End of Quarter Report

Husky Automated Package and Parcel deliverY

Submitted by:

Nicholas Belbas

Brian Davis

Luis Cardona

Drake Lamb

Submission Date:

May 9, 2022

2. Table of Contents

3. Table of Figures

4. Formal Project Description

Robotics delivery platforms are becoming common around the world from small towns to big cities and even college campuses. These platforms allow for delivery of packages, groceries, or food from a delivery service without requiring extensive manpower. However, these platforms have flaws in that they are expensive for small universities to implement without sponsorship, they are difficult to repair due to proprietary components, and have their own software that cannot be modified to integrate with HBU's existing systems. We aimed to create a system that could solve all of these problems by combining the knowledge of the two electrical engineers, cyber engineer, and computer science major on our four-person team. We have named our system H.A.P.P.Y (Husky Automated Package and Parcel deliverY). By creating a system that was less than 1/5 the price of the competitors as well as one with off-the-shelf components and thoroughly documented open-sourced code, we could open up the door for private entities to create their own delivery systems with these robots. The robotics platform is based off an Nvidia Jetson Nano and controls four REV spark motor controllers to independently control each wheel. We then have a CSI camera module for obstacle avoidance using Nvidia's Inference libraries, and a ZED-F9P GPS sensor paired with an inexpensive magnetometer for accurate location tracking and navigation between buildings. The robot communicates with a command and control server via the existing wifi infrastructure on campus, and all of our code for the robot is written in Python with Flask running our web app for the monitoring dashboard.

5. Engineering Design Specification

With limitless creativity at our disposal, it was best to develop a set of Specifications for what we needed to do. Concentrating our focus on this area taught the team more about the problem at hand and how to approach this core rather than fluffy accessory features. This design must be both practical and robust to survive the elements and normal day to day use.

5.1 - Primary Intended Use

HAPPY is intended for small package deliveries on campus. This will include standard-sized paper as well as letters and official documents. This scales up to packages across all departments using a standard-sized box that can easily be replaced if it breaks. Students could send gifts to one another or deliver books back to the library without disrupting a needed study session.

5.2 - Predictable Unintended Use

If everything goes wrong, there will never be a productive package delivered. Students could use the robot to deliver inappropriate items to other students or faculty, and despite it being trackable, the students may assume they have anonymity when sending these items. Package verification and layered acceptance that allows a receiver to deny the package will be vital to limit the misuse of the service.

5.3 - Special Purpose Features

- High-accuracy GPS for positioning
- Single camera for edge detection and obstacle avoidance
- Large tires for stability going over rough sidewalks

- Box containing deliveries can be swapped instead of using the robot AS the box
- Will have a web-based UI for initiating deliveries and monitoring of position
- Push notifications for delivery confirmation
- Tamper detection

5.4 - Requirements

5.4.1 - Functional Performance

H.A.P.P.Y. must be able to navigate autonomously around campus within the specified geofence in the code. It should accept a job request via the JSON body of an HTTP POST request and navigate to the desired origin building, wait until the box is placed inside it, then navigate to the destination and await pickup. The robot should be able to avoid people and vehicles while driving to the destination, and should identify if a malicious party is attempting to steal the package or the robot itself. While self-charging docks would be very nice to have, a simple power plug on the front of the robot should suffice, as the robot should have a fair amount of driving time before a recharge is required.

5.4.2 - Operating Environment

Our robot will navigate solely on outdoor paths at HBU including sidewalks and ramps and will not drive up or down stairs. Additionally, we are not planning to navigate across roads to the sports fields, Presidents House, Alumni House, or even the UAC since a street or parking lot does need to be crossed to access these.

While operating outside, the robot must be able to withstand any weather typically expected in Houston. This can range from the low 30s in the winter up to ~100 degrees F in the summer. Given that the temperature range of our electronic components tops out at nearly 150F,

our robot should withstand extreme temperatures. Additionally, our robot should be waterproofed to withstand any humidity amount typically encountered on campus.

5.4.3 - Geometric Limitations

We want our robot to be as small as possible and do not have a maximum size for it. However, since we are using a standard-size shoebox from The Container Store as the delivery bin, the robot must handle at least carrying (fully inside itself) a 15in x 6in x 11in box.

5.4.4 - System Maintenance

Most components will not be soldered together and will instead use quick-release connectors so replacement parts can easily be installed. The power distribution system on the robot utilizes a simple screw-in terminal block, the sensors and motor controller use standard breadboard-style jumper pins, and the motors use a standard XT30 connector.

5.4.5 - Failures and System Recovery

The robot utilizes I2C for its sensors and controllers which allows for devices to be daisy-chained. However, in the event that a device fails, we do not want it affecting all other devices downstream, so we have wired our sensors up in a hub configuration so each sensor has a direct line to the I2C pins on the Jetson Nano. Additionally, all the code is on github and can be cloned to each robot with a single command, so a failure of the flash storage on the robot is easily repairable once a fresh OS is flashed to it.

6. Constraints

A project of this scope must be limited in some ways. Even with such boundaries in place, our team has spent countless hours designing and redesigning the optimal solution to our question: what is the best way to automate delivery on campus? The first one was time. This coincided with the second but almost equally important parameter of budget. These were the numerical constraints to a metaphysical question of what would be a project worthy of graduation after four years of learning about engineering? Enter our greatest use of time thus far, long and thorough discussions. Those hours spent deliberating are condensed into the 9 categories shown below.

6.1 - Economic

We understand that most venues utilizing delivery robots did not actually purchase the robots themselves and were instead sponsored by that delivery company (ie: Starship). Because of this, we realized if we wanted venues (or HBU) to purchase our robots, we needed to be as cost-effective as possible.

We kept our costs low, but as the supply chain issues that started during the pandemic intensified, we found some of our required components to be increasing in price rather substantially. For example, when we purchased our Nvidia Jetson board it was initially priced at $111 for a kit containing a brand new board, but now the price for a “Used, good condition” board starts at around $470!

6.2 - Environmental

Our environment consisted of Houston Baptist University's concourses, Happy will stay on the sidewalk and will not be crossing over grass. The team chose not to tackle stairs but to navigate ramps instead, giving access to the entire campus in the process. In relation to the global environment our battery-operated delivery service does not give off harmful emissions, odorous exhaust fumes or the loud cacophony of combustion engines.

6.3 - Social

As mentioned in section 6.2, the electronic motors make the unit relatively quiet to students on campus. Their relationship to the HAPPY delivery robot must be positive for the success of the delivery model. Our team wants to use the parcel carrier as the main vessel of advertising for future deliveries. Similar to HBU's "MyPrint" option students could receive a live update of their queue status and estimated delivery time.

6.4 - Political

Some students may not be happy with a system containing a camera that can detect people constantly driving around campus. While our robot does detect and track individuals within the line of sight of the robot, there is no facial recognition functionality for tracking an individual person over time.

6.5 - Ethical

The sensor data being collected is only used for robot location logging and delivery tracking. The robots cannot and will not go inside the buildings and invade people privacy in such a way that the sensor data leaks private information about that person. Additionally, the camera is on the front of the robot but cannot tilt up or down, so if someone hacked into the robot it could not be used to investigate places where they should not be allowed to look.

6.6 - Safety & Welfare

We should incorporate some kind of system for monitoring what exactly is being sent in the robot. This could include a small camera inside the robot for looking into the box, and possibly gas and temperature sensors to determine if a hazardous material is being moved.

The robot itself does not contain any dangerous materials besides the fully contained Lithium Iron Phosphate battery which has a built-in BMS to prevent against thermal runaway and overcharging.

6.7 - Public Health

This robot was initially conceived as the pandemic was winding down, so one of the main goals of the robot at the time was to allow for less foot traffic for deliveries and also less human interaction, thereby reducing the spread of Covid-19.

6.8 - Manufacturability

In its current form, this project is not ready for mass manufacturing. Everything was assembled by hand and modified slightly to make everything fit. In order to conform is to common manufacturing processes, we would need to make a few changes. First of all, components like sensors should be soldered to small hot-swap circuit boards and then connected to the main board via a connector that allows for quick release. This would mean that loose wires inside the robot would be greatly reduced without compromising the ability to replace components quickly. Additionally, we need to refine the metal frame pieces for the chassis so

that the dimensions are exact and not slightly off, since we had to bend some parts to make them fit. Finally, we would replace the 3D printed components like the front plate and “ribs” with injection molded plastic that is cheaper to produce and more stable.

6.9 - Sustainability

Since this was meant to be a functional prototype, little to no consideration for sustainable materials took place. We had originally had an idea to create the chassis out of wood, but for strength and waterproofing purposes we ultimately scrapped that idea. If we made another version of the robot, we would like to try our hand at making it more environmentally friendly.

7. Technical Details

To build our delivery robot, we first need to create a robotic platform for our code. This consists of a drive train comprised of motors and motor controllers, a power distribution system comprised of a battery as well as a charger circuit, sensors and a camera to gather data from the surrounding environment, and a central controller to handle all the inputs and outputs.

Below, we have listed the components we are utilizing for each of these parts of the robot, how they interact with each other, and how the full robot will function.

- Drive System:
    - 12V 182RPM DC Motors (36PG-3429-51-EN)
    - REV Robotics SPARK Mini brushed DC motor controllers
    - Sunfounder PCA9685 16-Channel PWM driver
- Power Distribution System
    - Nermak 12V 10Ah Lithium Iron Phosphate battery
    - Battery management system (10A rating, built in to battery)
    - Charging Circuit (purchased off-the-shelf, modified plug)
- Sensors
    - SparkFun GPS-RTK-SMA ZED-F9P GPS sensor
    - HMC5883L Triple-Axis magnetometer
    - ADS1115 Analog to Digital Converter
    - HiLetGo Voltage Sensor (just a voltage divider with resistors)
    - Waveshare IMX219-160 4K 160-degree camera module
- Processor

- Nvidia Jetson Nano 2GB
- Adafruit HT16K33 LED Bar
- High-power USB Wi-Fi antenna

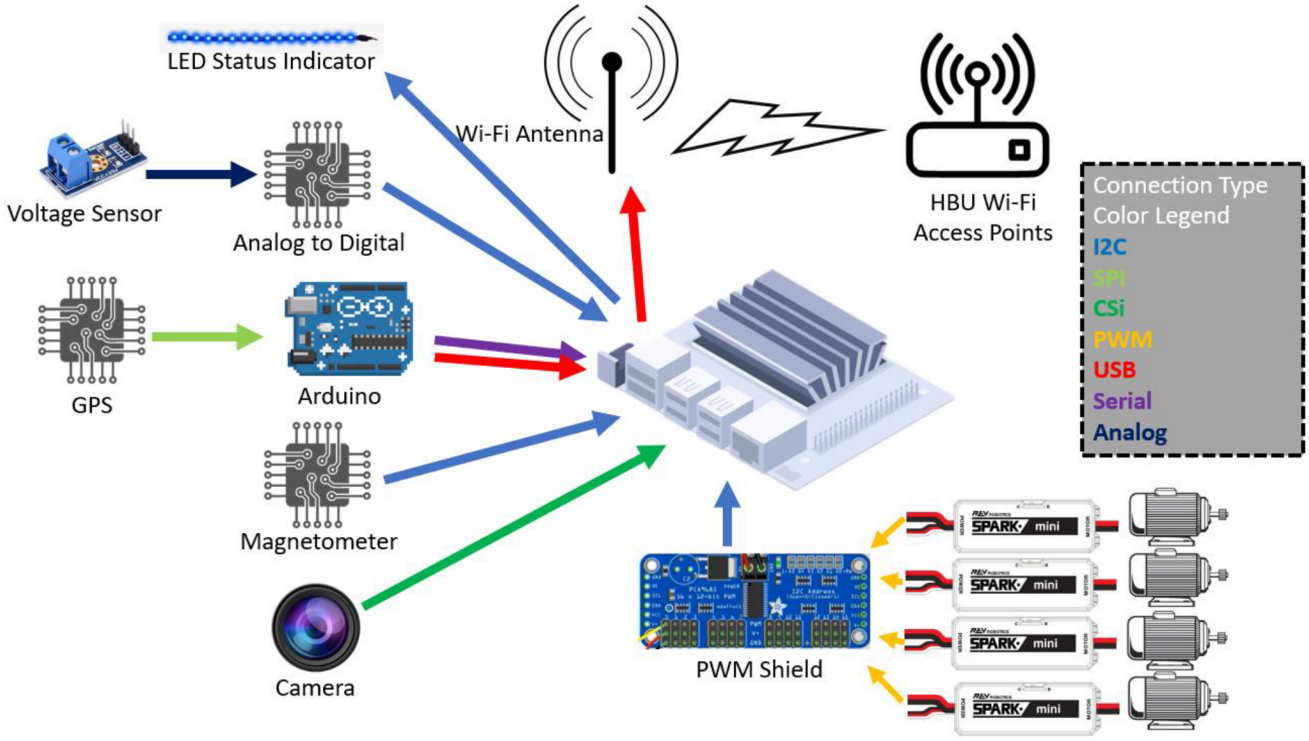


Figure 1: Component diagram

Figure 1 above shows the overview of how all these components are connected together logically. We will be using I2C for the magnetometer, LED status bar, PWM controller, and Analog to Digital Converter to communicate with the Jetson module. Additionally, the camera module uses the standard Camera Serial Interface (CSi) to connect to the Jetson. Our GPS module connectes via SPI to an Arduino Uno which acts as a value buffer and translator for the GPS sensor, and the GPS is connected via USB and communicating over serial to transmit the coordinates to the Jetson. We originally hoped to connect the REV SPARK Mini motor

controllers directly to PWM pins on the Jetson, but since it does not have any PWM capable pins, we are utilizing a PCA9685 servo driver connected to the Jetson via I2C.

We have had experience with PWM control of motors since the first semester of our freshman year at Houston Baptist University, and we have had some experience with Serial and I2C thanks to our Circuits and Microprocessors classes. Additionally, we programmed almost exclusively in Python as this is a language we have also used since freshman year here. Also, since Python is interpreted rather then compiled, we can make changes to the code as it runs and not require a full restart of the program. Finally, our electrical engineering teammates have had extensive experience working with circuit design and power draw calculations thanks to their classes, and this came in handy while designing our power distribution system, selecting out batteries and pack configuration, and finalizing our choice of battery management system (BMS), although due to spot welding issues we ultimately decided to use an off-the-shelf battery.

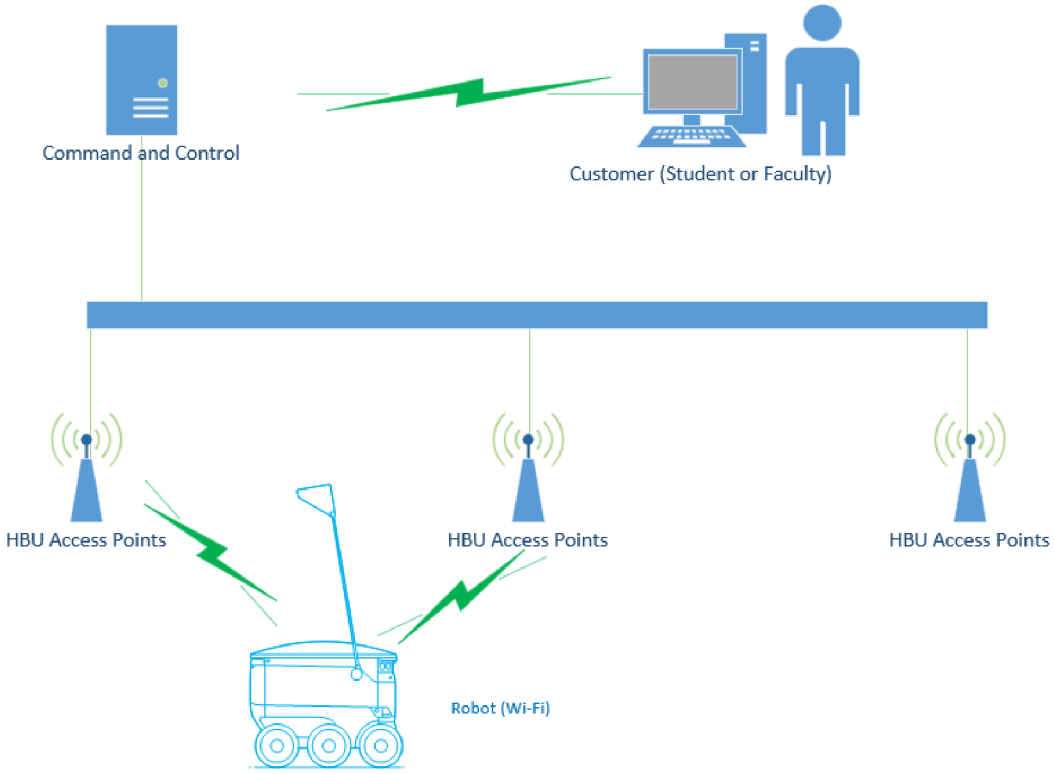


Figure 2: Network Layout

Our robot is designed to receive and execute delivery orders placed on a website by different users. The robot communicates with a central server by connecting to the existing HBU wi-fi access points using a high-power wi-fi antenna onboard the robot. The users are able to place orders for deliveries by selecting the origin building and the destination building as well as inputting contact information for the desired recipient. Conceptual views of this user interface are included below as Figures 3 through 13. The robot will then drive to the origin building (assuming it is not already there) and receive the package from the sender. Then, the robot will

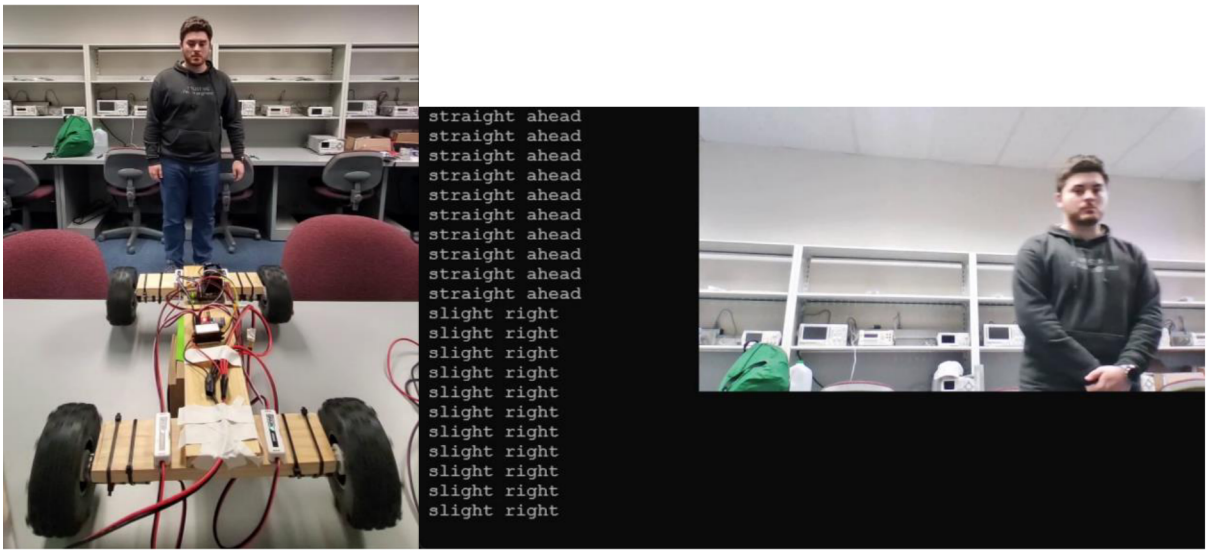
drive to the recipient's building and wait there until the recipient retrieves the package by entering a secure code on a keypad located on top of the robot.

In order to navigate between buildings on campus, our robot will utilize all of the above-mentioned sensors. The GPS sensor we have chosen has a high accuracy to within 10mm, and this is the primary sensor we will use. By creating a look-up table of paths and buildings on campus, our robot can easily plot the shortest distance between two buildings and drive there. The magnetometer will be used as a compass for the robot so it knows what direction it is pointed at all times. Additionally, an accelerometer which we hoped to add would be used as a sort of anti-tamper detection to ensure the robot is not picked up or broken into. Finally, the camera module will perform object detection and assist us with obstacle avoidance. By detecting people or vehicles in our way, we can override our navigation algorithm on the fly to ensure we do not hit anyone or anything. We were recommended early in the project that it would be smarter to run our navigation algorithm on the central server and simply send the steps to the robot, but this would not work if the path had to be modified in real-time as mentioned above.

Current Progress

We have successfully assembled the robot into a mostly working state. All of the sensors and controllers communicate properly and are queried using our code. However, we do not have autonomous code fully functional.

Object Detection

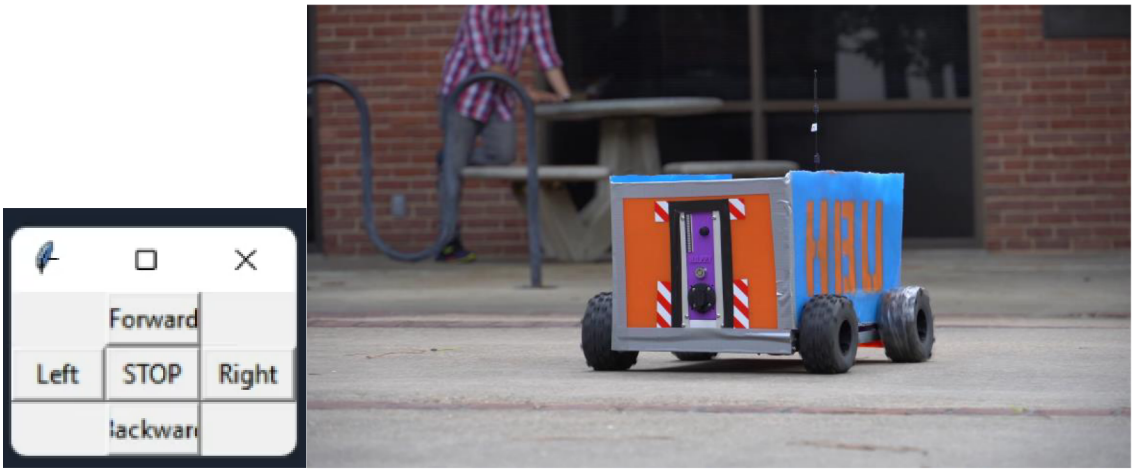


Figures 3 & 4: Prototype wooden chassis attempting test code to identify where the closest person is and drive around them. Terminal window shows output of neural network displaying current position of closest person.

We utilized the Nvidia Jetson Inference libraries along with a modified pretrained neural network from Nvidia to identify the closest person to the robot and print out their coordinates. The example code in the above photos simply tells us roughly where the person is for simplicity, but we can return the exact X/Y coordinates of each person the robot sees as well as a confidence level from 0-100%. This code works in a standalone module and can simply drive to avoid or

chase the nearest person, but we still have work to do before this can be implemented into the complete autonomous code.

Remote Control



Figures 5 & 6: The very simple UI on the client side (laptop) for remote control driving of the robot. An image of our robot driving under remote control, with the laptop and driver visible at the table in the background of the image.

We successfully programmed our PWM shield and all four motors and have created a simple program that uses the python socket library to open communication between the robot and a laptop to allow for direct remote control over the existing HBU network. We have a simple version of the socket communication, but we also implemented a more secure version that uses OpenSSL to secure the communication between the client (laptop) and the server (robot).

Web Dashboard (Robot Monitoring)

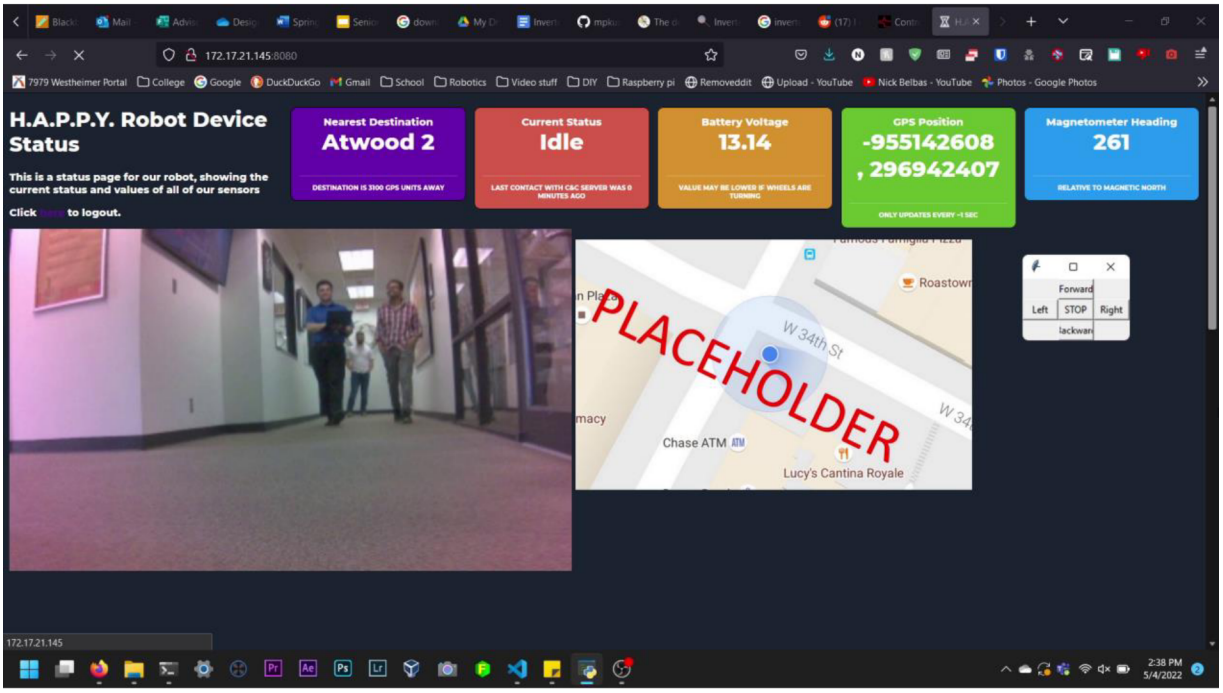


Figure 7: Robot Monitoring dashboard with remote control window overlaid.

Since we had all the sensors for autonomous code working even though we were only driving under remote control, we created a simple web interface that would display the updating values in real time from the different sensors on our robot. Going from right to left, we have our magnetometer heading with relation to magnetic north, our GPS position, our current battery voltage, current status (progress of job in progress or if bot is idle), and the nearest destination (calculated in real time by referencing a list of on-campus buildings' locations and comparing it to the current coordinates to find the closest one). Finally on the bottom row, we have a live camera view from the obstacle avoidance camera and a placeholder for a real-time plot of GPS coordinates. The sensor values update every second and the camera view runs at 480p at approximately 10 frames per second.

This web UI was created from scratch using the Flask Web Framework, and the dynamic text was accomplished using a free plugin known as TurboFlask. The camera feed was embedded by using OpenCV which we had to compile from source multiple times to get functional.

We did add a login function to the page, so only authenticated users would be able to view the data and camera views from the robot.

Web UI (Placing Orders)

Figure 8: Currently functioning web UI for placing orders

We created a simple web UI that would allow users to place an order and send the job info directly to the robot via an HTTP POST request with the content in JSON format.

Standards:

On our robotic platform, we utilized different existing standards for wireless and wired communication as well as control of various components. Our used standards are listed below.

- ***IEEE 802.11ax*** (Wireless communication via ***Wi-Fi 6***)
- ***Phillips InterIC*** (***I^2C*** for communicating with sensors)
- ***EIA RS-232*** (***Serial*** communication between Arduino and Jetson)
- ***Motorola Serial Peripheral Interface*** (***SPI*** between Arduino and GPS)
- ***MIPI Camera Serial Interface*** (***CSI*** camera module)
- ***Pulse Width Modulation*** (***PWM*** for communicating with motor controllers)
- ***Intel Universal Serial Bus*** (Serial over ***USB*** from Jetson to laptop for code modification)

8. Cost Analysis

Most of the listed items are discussed in detail in the previous section. Their prices and quantities are listed below as required to build our current robot version. Bear in mind, this has changed significantly as demand for the different hardware components of this project drastically increased during the spring semester. Some of these components have seen an inflation of four hundred percent.

Item	Individual Cost	Quantity	Total Cost (MSRP)	Total Cost (Current Price)
12V DC Motor	$28.70	4	$114.80	$114.80
SPARK Mini motor controller	$35	4	$140	$140
PCA9685 PWM controller	$10	1	$10	$10
LiFePO Battery	$49.99	1	$49.99	$49.99
GPS Sensor	$274.95	1	$274.95	$274.95
Magnetometer	$15.95	1	$15.95	$15.95
Analog to Digital	$14.95	1	$14.95	$14.95
Voltage Sensor	$6.99	1	$6.99	$6.99
Camera module	$29.25	1	$29.25	$38.89
Nvidia Jetson	$111	1	$111	$470
		TOTAL COST:	**MSRP: $767.88**	**Current Cost: $1136.52**

9. Design Alternatives

The initial impression to design alternatives was influenced by the budget constraints. The following will be a listing of the design alternatives we chose and a little about why we chose them. Because the robot was to deliver between buildings, it was decided that the robot would primarily operate outside. The use of GPS over a LIDAR was chosen because of the outdoor nature of the project. A GPS would satisfy the needs of the project and enable the robot to be tracked on the web interface. The project called for object tracking with a camera to have automatic object avoidance. Additionally, since the robot would operate outside, this motivated our decision to use traditional brushed DC motors rather than brushless motors due to the conditions the robot would have to endure outside.

10. Future Expansion of Project

The original purpose of this project was to create a robot that could receive requests and autonomously deliver packages to predetermined locations using AI (Artificial Intelligence) object tracking algorithms to avoid objects in its path. This idea could easily be expanded upon in many ways, but in an effort to limit the scope of the project we decided to work towards the current solution we have in mind. We will further consider prospects in the future if time permits. Some of these prospects may include but are not limited to suspension, a more durable chassis, holonomic wheels, more robots, a train-like carrier car, and many more. Currently the robot's design is made with PLA panels on the side, which are known to not withstand heavy heat – this will be a major point of improvement for the future of this project. The robot will need to be weatherproofed and sealed from the outside while also allowing the components to receive enough air to not thermal throttle within the chassis. Another implementation that would further the project in the future would be to produce many of these robots, a fleet so to speak, for use simultaneously. This will enable the robots to be more efficient in completing their job. The total number of robots needed will be dependent on the size of the campus in which this project is fully realized. HBU, for example, would need only two or three in total for its current size. This also implies that the robot will be malleable enough to use on other campuses of other universities. This is entirely up to the owner's discretion.