HIVE: The Grounded Swarm

Isaac Finley, Cooper Fitzgerald, John McClain, Cameron Nichols, Benjamin Palladino

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

***Abstract* -- HIVE: The Grounded Swarm is an autonomous robotic fleet with cloud-based computation that features modular attachment capabilities for an end-user to customize the use case of each agent on a moment's notice. To accomplish this, agents communicate LiDAR and odometry data from onboard hardware to a server via Wi-Fi to map and navigate their environment. A combination of CAN and SPI communication protocols alongside power outputs allows the end-user to add additional components of choice to modify the agent's application. This paper details the design of the hardware and embedded software used to implement this fleet.**

Index Terms* -- *Autonomous robots, mobile robots, adaptive systems, multi-robot systems, cloud computing.

I. Introduction

Based on trends toward market-wide implementation of robots in the last decade, combined with the consequences of the COVID-19 pandemic, robot automation is unarguably becoming a static component of numerous industries. With one report from Fortune Business Insights projecting growth of the autonomous robotics market by 23.7% in 2028 compared to pre-pandemic levels; it is evident that key market players will increasingly seek solutions to overcome the existing barriers of implementing robotic solutions.

Some of the major restraining factors hindering companies from utilizing such systems are high startup costs, difficulty deploying robots parallel to current company architecture, and limited adaptability. HIVE: The Grounded Swarm aims to address these shortcomings. Firstly, offloading strenuous calculations and processing to a server reduces the cost and complexity of onboard hardware which would normally be the most expensive component of the system. Not only is the agent itself cheaper, but capital is also saved in labor, as it is much more time-efficient to alter the needs of a system through software than to address them through hardware changes. For hardware that inevitably does need to be updated, this is where the modular attachment system comes into play. Traditionally, robotic systems are implemented to perform a small set of duties to improve a process or solve a problem. Contradictorily, the problems and needs of companies are always changing as they grow; leading them to waste time and money discarding and recreating existing robotic systems. The agents developed in this project allow for growth alongside the needs of the company, without needing to reinvent the wheel every time a system faces changes.

The agents each feature a set of modular attachment ports with power and data capabilities that allow users to adapt the agent, without changing the premise of the entire system, to new challenges. Given the standard aluminum extrusion used for the system combined with power and data ports, the system is apt both mechanically and electrically for a user to customize its application. The general functionality of the agent remains constant, as to provide a standard that companies can easily develop without needing complicated alterations. Agents come fit with the ability to autonomously navigate a given indoor environment utilizing onboard odometry and mapping sensors.

For the sake of fitting within the time constraints of this project, the team chose to develop one example of a custom attachment for the scenario of a warehouse operation, a common use case of autonomous mobile robots. The agents feature four payload manipulation mechanisms that can lock on a payload and deliver it to a given location within the environment. Additionally, while the software developed is capable of handling multiple agents, budget constraints led to the development of two agents, which demonstrate the fleet capabilities of the project without sacrificing additional capital.

Given these specifications, the final demonstrable products of this project are two robotic agents that communicate with a server to translate sensor data into autonomous navigation of an environment and interaction with a payload.

II. System Concept

Fig. 1 depicts the overall integration of the hardware system. The system begins with power, which consists of a rechargeable lithium-ion battery. This battery is directly connected to the input of three onboard voltage regulators which supply the PCB with three rails; 3.3 V, 5 V, and 12 V. These regulators deliver power not only to the main PCB but also the attachment modules which are discussed later in the paper.

The central point of the system is the microcontroller unit (MCU). As previously mentioned, the computation for the environment mapping and navigation of the agent is performed on a server, so this processor serves as the controller of just the embedded system. Firstly, the MCU interfaces with the motor drivers, which control the four

motors on the agent. The MCU also interfaces with the feedback system, which includes a LiDAR, an inertial measurement unit (IMU), and Hall effect sensors. Additionally, a status light is used to indicate that the agent is in use to warn users.

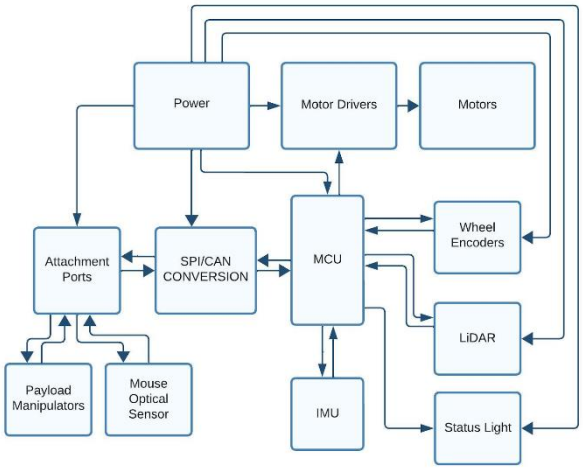

Fig. 1. Overall hardware block diagram.

Added attachments will communicate through a CAN bus. In our case, the solenoids used in the payload manipulator attachments were not able to directly interface with the CAN bus, so the ports for these devices feature an additional, slightly more primitive microcontroller, each fitted with a CAN/SPI conversion module. Likewise, this same conversion module is also on the main PCB. Essentially, for our specific attachments data originates on the SPI bus, converts to CAN and travels through the CAN bus to the main MCU, and then is converted back to SPI before being processed. The mouse optical sensor, although considered part of the feedback system, was also chosen to be an attachment in this application. This was done not only to demonstrate the flexibility of the modular attachment system but also because the mouse optical sensor would not necessarily be needed in every application.

Not shown in Fig. 1 is the software breakdown, which was developed by the computer science students on this interdisciplinary team. At a high level, the software can be broken down into four parts. Firstly, the web application functions as a user interface that can take user commands and send them to the robotic backend. The robotic backend is responsible for the majority of heavy lifting, relating to the concept of computation being offloaded from the physical agent. Here, tasks are conducted like storing and updating the mapping environment, as well as path planning. Furthermore, a Raspberry Pi acts as a router, providing a way for the agent and robotic backend to communicate. Lastly, there is the agent software - which is the portion of the software developed by the hardware side of the team. This software is responsible for interpreting the commands sent from the router and using these commands to control the drivetrain and any potential attachments. Additionally, the agent software relays sensor data to the router, which then sends this data to the backend, providing it with the information necessary to make informed path-planning decisions. This software is covered in further detail in the embedded software section.

III. System Components

The design consisted of five major components, including the microcontroller, power, feedback, attachment system, and drivetrain. Each section of the design played a pivotal role in meeting the overall goals of the project.

A. *Main Microcontroller*

We came across numerous processors that could have primitively met the agents' needs; however, optimizing for the simultaneous data transmission of multiple sensors and a diverse set of communication protocols led us to choose the ESP32-WROOM-32UE microcontroller. Firstly, this unit features a dual-core processor which provides parallel processing capabilities to enhance computational efficiency. This, combined with a processor speed of 240MHz was particularly important in being able to handle an overwhelming amount of sensor data while still being able to control locomotion and uphold communication with the main server. Secondarily, the ESP32 had an adequate number of GPIO pins that allowed us to easily connect all peripherals without sacrificing any pins that were required for pulling important debugging information during testing.

Support for various communication methods was another critical component playing into the selection of the ESP32. Due to the variety of sensors utilized on each agent, it was necessary to utilize more than one communication method. The optical sensor communicates through one of three UART channels. The IMU communicates through I2C, which was given a single channel. For the attachment system, it was decided that the MCU would interface through SPI to a CAN bus, of which the ESP32 featured two channels.

B. *Power*

Given the nature of an autonomous mobile robot, the only realistic option for a power source was a battery. The first step in designing the battery is to understand what loads the battery needs to provide. (1) explains the exact load that we expect the agent to require from the 12V, 5V, and 3.3V power rails to supply the agent and all attachments.

$$W_{total} = (12V*4A)+(5V*3A)+(3.3V*3A) \quad (1)$$

Once the load is understood, then the voltages that are required are next. After choosing to use a buck voltage design, the battery must supply above the highest voltage rail for the entirety of the drain cycle of the battery. Next considered are the battery types, these were researched with consideration of voltages, cycle life, current ratings, and cost. We decided to go with a LiPo battery in a 4-in-series configuration. This will achieve our required 12 volts at the minimum capacity requirement and is inexpensive to acquire. Lastly, the capacity is selected using the total wattage calculated before and the nominal voltage of the LiPo battery. (72.9 Watt / 14.8 Volts = ~5 Amps). Then converting the 5A rating over a 15-minute interval to the common mAh units, we receive a rating of at least 1,250 mAh. This is why we chose a 1,500 mAh LiPo battery.

To support the variety of hardware on each agent and the range of potential attachments an end-user may choose to utilize, it was necessary to have three stable and robust voltage rails on the board; namely 3.3, 5, and 12V. The 12V regulator was responsible for supplying current to the motors, up to 2.4A at a time; making it arguably the most critical regulator on the board. For this reason, considerations were taken to ensure the component was capable of handling this amount of current while still falling within reasonable thermal ranges.

Initially, three simplistic linear voltage regulators were used. Each device had a fixed output and required a small input and output filtering capacitor. The regulators upheld the agents for initial testing but overheated too quickly for long-term use. Moving forward, the AOZ6606PI switching voltage regulator was then selected for its high-efficiency ratings and ability to sustain a high output current without overheating. The variable output capabilities of this regulator appeared to work for all three voltage levels, however, in calculation, it was discovered that several of the capacitors needed for the 5V and 12V regulators needed capacitances in the mF range, which were only available in large package sizes.

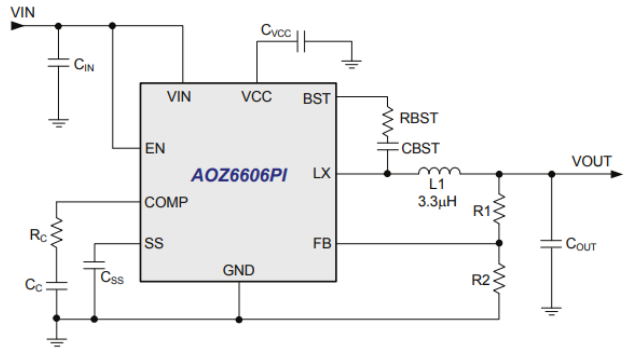

Fig. 2. AOZ6606PI 3.3V Regulator.

As shown in Fig. 2, the AOZ regulator requires 11 passive components. Given the spacious footprint of the regulator in its entirety, it would not have been possible to fit the necessary size capacitors - thus not allowing the regulators to function as intended. Because the 3.3V regulator had lower output expectations, the capacitor ratings necessary to support the system were much less demanding. Another regulator, the LM2576, was investigated and determined to be comparable to the latter, yet slightly less efficient. Given these facts, the team compromised by utilizing the AOZ6606PI for the 3.3V output, while using the LM2576 through-hole regulator series for the 5V and 12V outputs. The LM2576 is, similarly, a switching regulator, but has a footprint that accommodates the necessary size of the higher-rating filtering capacitors. Moving towards this design simultaneously allowed for maximum efficiency where physically viable, while taking the necessary steps to ensure that higher voltage outputs were both functional and stable.

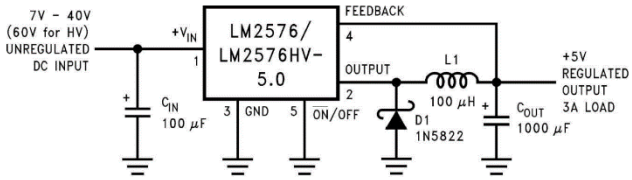

Fig. 3. LM2576 5V regulator circuit.

The most pivotal components in this system are the capacitors and inductors. According to the datasheet for the LM2576xx series of voltage regulators, C_{IN} needed to be at least a 100µF electrolytic capacitor. For the inductor, the datasheet provides application curves on page 21 based on maximum input voltage versus maximum load current for common output voltages, including 5 V and 12 V. Therefore, the inductor chosen was a 100µH inductor. As seen in Fig. 4, for an input voltage of 16.8V and a maximum load current of 3A for a 5V output, the L100 is what is recommended, which corresponds to a 100µH inductor.

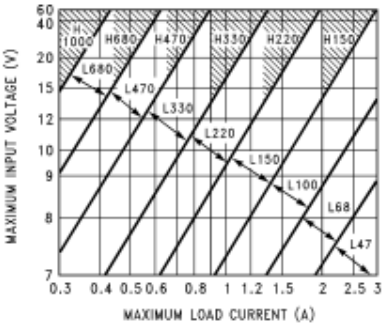

Fig. 4. LM2576 5V Regulator Inductor Graph

(2) highlights that the ripple current rating of C_{OUT} must be at least 50% higher than the peak-to-peak inductor ripple

current. Therefore, C_{OUT} should lie between 220µF and 1000µF.

$$C_{OUT} \geq 13300(V_{IN(MAX)})/(V_{OUT}*L) \quad (2)$$

A 1000µF capacitor was selected for the 5V voltage regulator as given the use of the 100µH inductor, the equation above was more than satisfied, and stable operation was assured. Additionally, the Schottky diode 1N5822 was chosen based on datasheet recommendations and component availability.

C. *Feedback System*

For the software team to effectively allow the agent to navigate an environment, several onboard devices were needed to relay important odometry, orientation, and mapping data.

The integration of an inertial measurement unit, or IMU, played a pivotal role in ensuring precise navigation and control. The IMU provides real-time data concerning the agent's acceleration, angular velocity, and local magnetic field, which is necessary for accurate localization of the agents. Continual monitoring of this information gives the fleet the ability to operate in the same constrained environment without interfering with one another. Without this data, the software would lack awareness regarding the fleet's spatial position and orientation which impedes its ability to effectively navigate them. The IMU selected for this project was the MiniIMU-9 v5 for several reasons. Firstly, this IMU had one of the lowest operation currents, 70 µA, out of its competitors; an important aspect for saving battery energy that is needed in larger quantities for the locomotive system. The system did have a larger zero rate output, which is the deviation of the actual output signal from the ideal output signal in the absence of acceleration; however when compared to major competitors the IMU had a lower noise density, zero-G offset, and zero-gauss level, all of which are crucial to minimizing the cumulative error of the device.

While the IMU provides critical data in determining the position and orientation of the agents, another sensor was needed to provide mapping data to the robotic backend. Options such as ultrasonic, infrared sensors, and overhead cameras were considered, but ultimately the team chose to use a LiDAR due to its superior data rate and range, both important criteria for the software team to optimally navigate the given environment. The LiDAR creates a full 360-degree two-dimensional scan of its surroundings, which is then sent through the attachment communication protocol. Out of the feedback devices, the LiDAR is most pivotal in the autonomous navigation of any given agent.

To improve the accuracy of driving, wheel encoders were implemented in the form of Hall effect sensors. With magnets being placed equally around the inner surface of each wheel, the Hall effect sensors are used to more accurately control the speed of the motors. Rather than just sending velocity commands from the server, the data provided by the Hall effect sensors allows the embedded software to correct the velocity in real time.

Additionally, a mouse optical sensor is to be used to provide additional odometry data, primarily the linear velocity of the device. Theoretically, this data could be gathered from the IMU, but it would be inaccurate and likely cause drift.

D. *Attachments*

With the defining principle of these agents being modularity, they were designed to support a variety of additional hardware that could change the application of the agent without altering the primary functionality of the system. For this reason, it was necessary that the attachment ports not only supply a variety of robust voltage lines but also host a versatile communication protocol that could interface with virtually any peripheral device. The concept is that given power and data capabilities, virtually any device could theoretically connect to the agent and interface with the main MCU.

Initially, it was planned to have all attachments on the single I2C channel, but it was discovered that an issue could arise with identical addresses amongst multiple of the same attachment being connected. To solve this, theoretically, the devices could be reprogrammed to have a unique address, but this would complicate the process for the end-user. For this reason, a CAN protocol was chosen.

CAN buses are proven to ensure reliability and efficiency in large systems, often being used as the main communication protocol amongst sensors in automobiles. With CAN, multiple devices can simultaneously exchange real-time data without bottlenecking. A data prioritization scheme allows for the most important bits to reach the MCU first, effectively creating a hierarchy that upholds the agent's functionality without compromising other important data. Many devices are capable of interfacing with the CAN bus directly; however, in application, an SPI/CAN conversion module was used, as the solenoids on the payload manipulators could not directly interface with the CAN bus. The team decided to use an MCP2515 stand-alone CAN controller. The MCP2515 includes masks and filters for filtering out unwanted messages, which reduces the host MCU's overhead, as well as decreases the complexity necessary in the embedded system. Likewise, the modules featured an interrupt pin which can be used to trigger the MCU when a message is received, another aspect helpful to software efficiency.

As depicted in Fig. 5, the protocol for our scenario begins with the main MCU interfacing the module via SPI. The MCP2515 then converts the SPI message into CAN, which

operates on two lines; CAN high, and CAN low. This message is broadcast across the CAN bus and any device connected to the two nodes. Then, opposite to the original broadcast, every other CAN module on the bus converts the message back through SPI, and to the secondary MCU, for our specific scenario. The embedded software on this secondary MCU is what decides whether or not to do something with the data. For the payload manipulators, the message sent through the bus will contain relevant bits that the team decided on to correlate with the solenoids, thus changing their state, or providing information about their current state. It is important to reiterate that some devices can natively interface with the CAN bus. In cases like this, the SPI/CAN converter as well as secondary microcontroller can be entirely omitted. Furthermore, if a secondary MCU was necessary, there are no limitations on the selection of that component other than relating to specific attachment specifications, and either SPI or CAN communication capabilities.

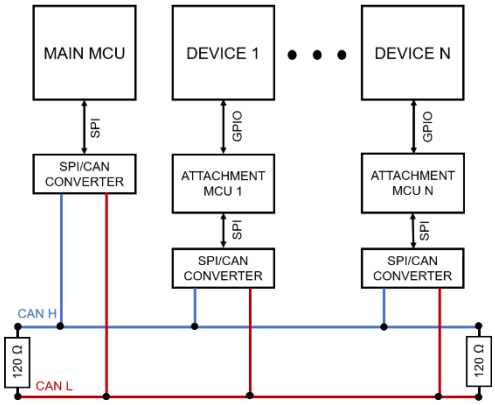

Fig. 5. Topology of the SPI/CAN communication system for N number of attachments for our specific scenario.

The number of nodes on a CAN bus is generally limited by the strength of the receiver. The MCP2515 can practically handle 110 unique devices, which leaves an ample amount of space for any attachments an end-user may select.

To prove the functionality of this system, an example attachment was developed. For the final demonstration of this project, our main goal was to navigate a warehouse setting and interact with payloads, to move them around the environment. For this reason, we created an attachment of payload manipulators, similar to how Autonomous Mobile Robots (AMRs) are used to bring items from point A to point B. To allow for optimal path planning, we planned on interacting with the payload vertically rather than horizontally. By doing this, the combined center of mass of the agent and payload would be relatively the same as just the agent alone, allowing for the best path planning capability. To do this, a mechanical attachment system needed to be designed to interact with the four legs of the payload. This was accomplished by designing a payload that envelops the footprint of the agent. As the agent drives under the payload, these mechanisms passively interact with the legs and then a solenoid is used to lock the position of the mechanism.

E. *Drivetrain*

Once the team defined the scenario for our robotic fleet and determined that the agent would utilize wheels to move around the environment, the next step was to choose the drivetrain type. To ensure that our robotic solution was as modular as possible, the team decided to use a holonomic drivetrain over a nonholonomic one. This drivetrain type would allow for three degrees of freedom, allowing the end user to have full control of the agent to best utilize its attachment system. Finally, it was decided to use a mecanum drive for the agent as the team thought that it would be most feasible due to its form factor and overall performance.

The main disadvantage to this drive-train type was the cost of the wheels. To overcome this, it was decided to completely design and manufacture the wheels in-house. Unlike traditional wheels, mecanum wheels generate a friction force at a 45-degree angle to the direction of motion, rather than normal to it. To generate the friction force in this direction, mecanum wheels have several rollers around the circumference of the main wheel frame where their axis of rotation is at a 45-degree angle to the axis of rotation of the main wheel. To accomplish this, the mechanical team created two revisions of this wheel. While the first revision was not successful it did show the team what needed to be done to develop a successful wheel in terms of mechanical design and manufacturing. With the resources at hand, the team decided to utilize FDM 3D printing to create the parts for the wheels. In this printing, we used PLA for structural parts and TPU for parts needing elastic properties.

Both PLA and TPU belong to the thermoplastics family of materials making them great for 3D printing. PLA has a higher yield strength than TPU making it a great option for parts of the wheel that will experience greater amounts of stress, such as the wheel frames, rods, and bushings. TPU has a much lower yield stress that allows for more elastic deformation making it a great option for the wheel's rollers and friction caps.

When it came to designing the mecanum wheels, it was decided to use a wheel frame that is an octagon allowing for the best balance of roller spacing and size, which ensures that one or more rollers will be touching the ground at any point in its rotation. This allows for the greatest roller-to-ground surface area ratio ensuring that the greatest amount

of friction force is always being created. TPU's elastic behaviors also allow for this contact patch to increase as it compresses due to the weight of the agent.

For mecanum drive to achieve three degrees of freedom, each wheel must be controlled independently by its motor. As mentioned above, the wheels themselves generate a friction force at a 45-degree angle, this means that the drivetrain must create a net force using a mirrored set of mecanum wheels to ensure that the agent drives in the correct position. Utilizing this concept is how mecanum drive can strafe, for example, to strafe right the right set of wheels must rotate towards each other while the left set rotates away from each other. The CAD model of this drivetrain is featured in Fig 6.

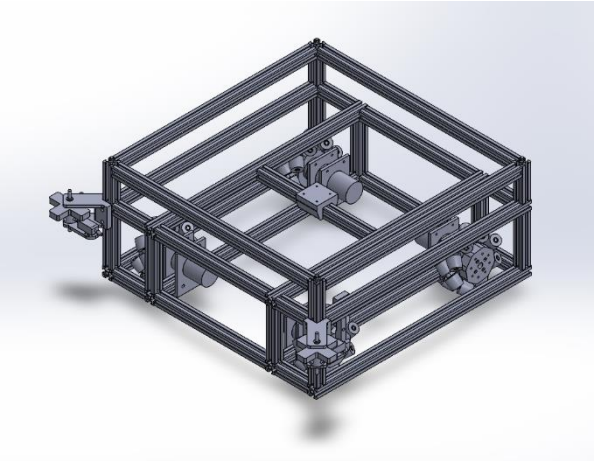

Fig. 6. CAD Model of Agent Drivetrain

Finally, the last portion of the drivetrain design was motor selection. To do this the team determined a combined weight threshold of 114.6N for the agent along with a static friction coefficient of 0.5 for TPU on polished concrete, it was also assumed that the center of mass of the agent would be roughly at the center of the agent. With this, the weight experienced on each of the wheels can be expressed through (3).

$$F_{Weight} = 114.6N/4 = 28.65N \quad (3)$$

With this, (4) shows the friction force generated by each wheel is equal to:

$$F_{Friction} = F_{weight} * 0.5 = 14.32N \quad (4)$$

The final assumption made is a no-slip condition, this allows us to set the friction force of the wheel equal to the tangential force created by the motor. Finally, assuming a wheel diameter of 100mm the required torque of the motor can be found through (5):

$$T_{Motor} = 0.100m * 14.32 \quad (5)$$

After converting units for appropriate scale, a motor with a rated stall torque of 6kg-cm was purchased for each wheel of the agent.

IV. Embedded Software Detail

The software contained on each agent consists of many different parts. This includes classes for motors, mecanum drive control, and the IMU. The software also features the main agent control program, which utilizes the aforementioned classes combined with additional logic and some external libraries, which allow for full control of the agent, as well as communicating with the attachment system and the robot router. To describe the overall functionality of the agent software, we will first examine the custom classes, then we will go over the main agent control program, which, as mentioned earlier, utilizes these classes.

A. *IMU Class*

The class for IMU control was developed by our team to combine the usage of the LIS3MDL magnetometer with the LSM6 accelerometer and gyroscope. This class includes the initialization function, which initializes the sensors and performs error calculations. The error calculations are performed by reading the results from all three sensors for five seconds and then saving the averages of the readings for all three axes for all three sensors. These error averages are later used in the functions that return the readings for each sensor. These functions all begin by calling the appropriate read function from the libraries that were developed for each sensor, subtracting the errors from the raw readings, and then converting these results into meters per second squared, degrees per second, and gauss, for the accelerometer, gyroscope, and magnetometer, respectively.

B. *Motor Class*

The motor class was developed by our team to introduce an easy method to control each motor and to use the Hall effect sensors to determine the speed at which each motor is spinning. This class includes the drive function, which accepts an integer between -255 and +255, which will determine the direction the motor spins as well as the PWM signal that is supplied to the motor driver. This function also will determine if the motor is saturated, meaning that it is being instructed to spin at full speed. This is useful to limit the speed of the other motors so that the agent will drive as expected. Another function is used to read the current revolutions per second (RPM) of the motor, and another is used to set the desired RPM.

C. *Mecanum Control Class*

The mecanum control class was developed by our team to easily control each of the four motors using the mecanum control principle. This class includes the drive function, which takes three floating point values as input: x, y, and z. These values represent forward/backward motion, strafing, and turning, respectively. This function calculates the necessary speed of each motor. This class also includes a function that is used to slow down all motors based on the fastest, saturated motor, if it exists.

D. *Main Agent Control*

Finally, we have the main agent control program. This program allows the agent to connect to the robot router's hotspot, communicate with the robot router via a WebSocket connection, interact with the attachment system, drive, and interface with the additional attached sensors, which include the mouse optical sensor and the LiDAR. One of these sensors will soon be implemented instead as an attachment and will interface with the agent via the CAN bus.

To achieve this functionality, we needed to utilize both cores of the MCU. Core 0 deals with all of the wireless aspects of the program. This includes establishing the Wi-Fi connection to the robot router and keeping the WebSocket connection alive. The agent needs to be able to receive commands from and send data to the robot router as fast as possible over the WebSocket connection, which is why this is all being done on Core 0 so that the work that is being done on Core 1 does not slow this process down. Core 1 deals with the remaining functionality of the agent software, which includes interfacing with the various sensors, the CAN bus, and driving the agent.

V. Testing Results

Given the multitude of components and complexity of integration, it was important to test individual systems before combining them.

A. *Voltage Regulators*

The initial set of regulators used were simple linear devices. Implementing the devices on a breadboard with a single input and output filtering capacitor on each regulator, and inputting a 14V input (an average voltage the battery would supply), the regulators were found to have the proper output and could supply power to charge a capacitor. However, testing the 12V regulator by connecting them to the motor drivers and driving the agent caused them to become extremely hot in a matter of minutes. Placing external heatsinks on these chips helped to dissipate heat, but not enough to ensure the long-term safety of the device.

Moving forward, the team decided to use switching regulators. By nature, this type of voltage regulator is better equipped for high-output currents.

B. *Communication*

The CAN conversion modules were tested by interconnecting two ESP32-WROOM-32UE modules by using the SPI/CAN conversion modules and transmitting data through the CAN bus. Reading from the ATTINY88 with an attached computer was difficult, so this is the way we decided to ensure each SPI/CAN conversion module was fully operational. Once this was confirmed, we ensured that each ATTINY88 could send data over the CAN bus, and ensured that data could be read by making an LED blink by sending the command over the CAN bus, which can transmit data at a rate up to 1Mb/s.

C. *Attachment*

Once communication through the CAN bus is achieved, there are two attachment modules with their respective microcontrollers and a standardized cable system to test.

The purpose of the standardized cable system is for the Attachments to connect to the agent. The ports include the three power rails 12 V, 5 V, and 3.3 V with a ground as well as the CAN high and low connections. This is done using 18 AWG wires and two MR-30 connectors per attachment port. This connector was chosen due to the 30A continuous rating and its inexpensive cost. After checking continuity on all connections and running 10A through the connector and wire chain, no issues were found.

Once the cable system is confirmed, next are the attachments. The first module is the Payload Manipulator Attachment using the ATTINY88. The ATTINY88 is the less powerful microcontroller of our two options and was confirmed to successfully receive commands from the CAN bus as well as perform simple GPIO switching. This was best suited for the Payload Manipulator Attachment which only requires a single GPIO connection to activate.

For the testing of the mechanical locking mechanism for the Payload Manipulator Attachment, several revisions of the design were tested to ensure seamless agent-to-payload interaction. The main challenges of this component were the clearance issues that arose due to the relatively close tolerances between the agent and payload. To avoid collisions of the attachment system and the payload legs the frame of the mechanism had to be altered accordingly. The other main issue with this setup was the fact that the legs and the passive component of the mechanism had the same cross-sectional shape (a square). To overcome this the payload legs had to be fitted with a 3D-printed part that allowed the legs to have circular cross-sections where the mechanism intercepts the

payload. After implementing these changes we were able to successfully interact with a payload through teleoperation.

The second attachment is the mouse optical sensor which can detect the displacement between time frames. This device was able to be read by the microcontroller for this attachment, the ESP32-WROOM-32U. Further testing for the microcontroller was performed to confirm the higher processing speed capability including additional tests with the LiDAR device.

D. *PCB*

A few iterations of the PCB were developed with the second version, pictured in Fig. 7, becoming the official design for the agent. Conceptually, the designs were not too different; but several of the first PCB's shortcomings were addressed in the second version.

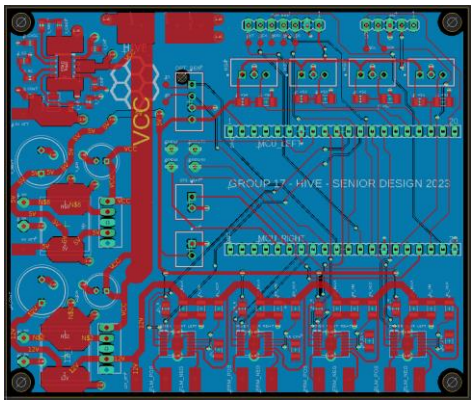

Fig. 7. Final interaction of the PCB.

Most importantly, more adequate thermal considerations were taken into account. For the 3.3V regulator, which is an SMD component, a large two-layer pour for the input and output pins of the regulators, as well as for the two pins of each inductor were placed. For the through-hole components, the 5V and 12V regulators, large copper pours were included for the inductors, and external heat sinks were placed on the heat pad of the chip itself. Additionally, a 2-pin JST connector for a fan was placed. The power section was arranged so the fan could provide airflow directly over all three regulators in a manner that doesn't push heat to other sections.

On a more functional level, we corrected many traces that were routed to unusable GPIO pins, as well as strategically moving certain components to other pins to leave important communication channels open if necessary. For example, the hall effect sensors were moved to input-only pins since they are not a bidirectional device. The team also opted to place test points for all communication lines, as well as leaving through-holes for any unused GPIO pins in the case that it was needed to connect a component for testing.

VI. Conclusion

The final design of the agents effectively met our main goal of creating a modular robotic solution with core functionality. At this point, the software is still being developed to use all agent sensor data optimally and efficiently, but the principle design of the hardware system is fully functional.

VII. Biographies

Isaac Finley is a 23-year old Electrical Engineering student, who has accepted a position with Black & Veatch as an Electrical Engineer 1 in the grid substation department after graduation.

Cooper Fitzgerald is a 20-year-old Electrical Engineering student. He has accepted a position with L3Harris as an Electrical Engineer after graduation.

John McClain is a 23-year-old Mechanical Engineering student. After graduation, he hopes to work in an engaging field where he can apply the technical concepts learned through his degree.

Cameron Nichols is a 22-year-old triple-major student, graduating with majors in Computer Engineering, Computer Science, and Electrical Engineering, as well as minors in Intelligent Robotics Systems and Mathematics. He has accepted a position with Prism Systems, Inc. as a Controls Engineer where he will be working with robotic systems, artificial intelligence/machine learning, and various other types of systems.

Benjamin Palladino is a 22-year-old Electrical Engineering student. After graduation, Benjamin will be working as Assistant Electrical Engineering for Burns & McDonnell, contributing to the design of substations.

References

[1] Fortune Business Insights. (2022, January) Autonomous Mobile Robots Market Size. Retrieved 9 November 2023, World Wide Web: https://www.fortunebusinessinsights.com/autonomous-mobile-robots-market-105055

[2] Alpha & Omega Semiconductor. (2018, November) AOZ6606PI Datasheet. Retrieved 11 November 2023, World Wide Web: https://aosmd.com/res/data_sheets/AOZ6606PI.pdf

[3] Texas Instruments. (2023, March) LM2576 Datasheet. Retrieved 11 November 2023, World Wide Web: https://www.ti.com/lit/ds/symlink/lm2576.pdf