MEMORANDUM

To:	Charlie Refvem, Department of Mechanical Engineering, Cal Poly SLO crefvem@calpoly.edu	
From:	Rees Verleur rverleur@calpoly.edu	Jack Ellsworth gjellswo@calpoly.edu
Group:	Hippotronics	
Date:	June 13, 2023	
RE:	**ME 507 Final Project**	

I. Design Overview

1.1 Project Strategy

Our goal was to develop a robot that could sweep around the arena to collect as many balls as possible while sorting them during intake. We intended to acquire targets through a video sensor with object recognition. Our system needed two brushed DC motors for travel, two for our intake system, and two servos for ball sorting and delivery.

1.2 Component & Sensor Selection

We selected drive motors for their size, which was limited by the project requirements, and torque. We wanted to avoid a long gear train if possible and planned to attach the wheels directly to the motor shafts. From some simple calculations with the weight of the bot, desired speed, and rolling resistance, we determined that the motors we chose should have no problem driving the wheels directly. Our intake motors were similarly selected, with an additional requirement of being as low-cost as possible.

We purchased many sensors to support our design. Many had redundant functions since these were purchased early in the project to give us flexibility while developing the design. For locating our robot during the competition, we selected a 9-axis IMU, Hall effect sensor, and IR line sensors, all redundant with our drive motor encoders. We also purchased a Pixy2 camera for object identification. Late in the project, we replaced our RC receiver with a Bluetooth module to interface with team Serious Blunder's overhead camera for path planning and positioning, making all these sensors unneeded. We also selected a color sensor for ball sorting.

II. Mechanical Design

The main goal of the mechanical design was to hold the selected components in space and facilitate ball capture, rejection, and delivery. We developed two testing rigs to aid design, one for drive system testing and one for intake geometry validation. Since our complete mechanical design was custom-made, designing and manufacturing the system was very time intensive, as all assembly components were meant to be 3D printed or laser cut. This design style could have been accomplished if it had been started in earnest earlier in the quarter, but since it was left to the last minute, manufacturing was left incomplete.

Our drive testing rig (shown in Figure 1 below) was 3D printed and would have allowed us to test our drive system software. This was not a trivial task, as we intended for our computer to receive information from the overhead video, apply a path planning algorithm to determine where the robot should go, and send the path to the robot as a series of instructions to turn and then drive forward a specified amount. This rig included standoffs to connect to our PCB's mounting holes and open space in the center for our battery and Bluetooth module. Due to PCB issues (discussed in Section 3), we could not use this tester to validate these systems. This would have validated half of our project, with the remaining systems being intake, sorting, and storage.

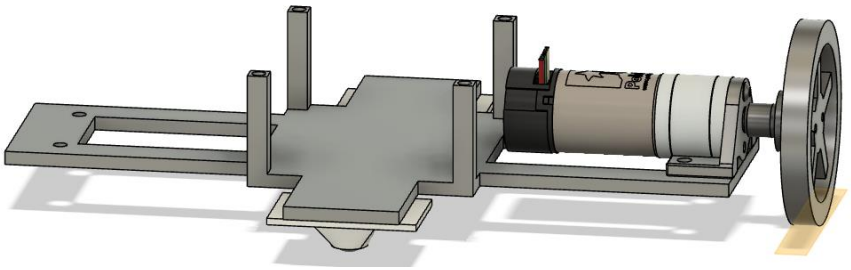

Figure 2.1 Drive Testing Rig

Our intake test rig was very successful, allowing us to verify our concept and adjust key dimensions to find the best geometry. This rig is shown in Figures 2a and 2b. Our intake system is comprised of a polychord loop and a resin-printed ramp, with support structures allowing adjustment of the rig using flathead screws. We designed this system so that we could tension the cord and adjust the loop's offset from the ramp base. The ramp could snag a ball as soon as the polychord made contact, requiring minimal pressure for the loop to pull a ball along. This validated our design and motivated its use moving forward. We had a few other ideas for our intake system, namely a belt-based system and an arm-based system. However, we were unsure which idea would be best, so it was a significant milestone for the mechanical design when this tester worked as intended. From this design, we also learned that the polychord is more rigid than expected and would behave better using a larger pulley diameter.

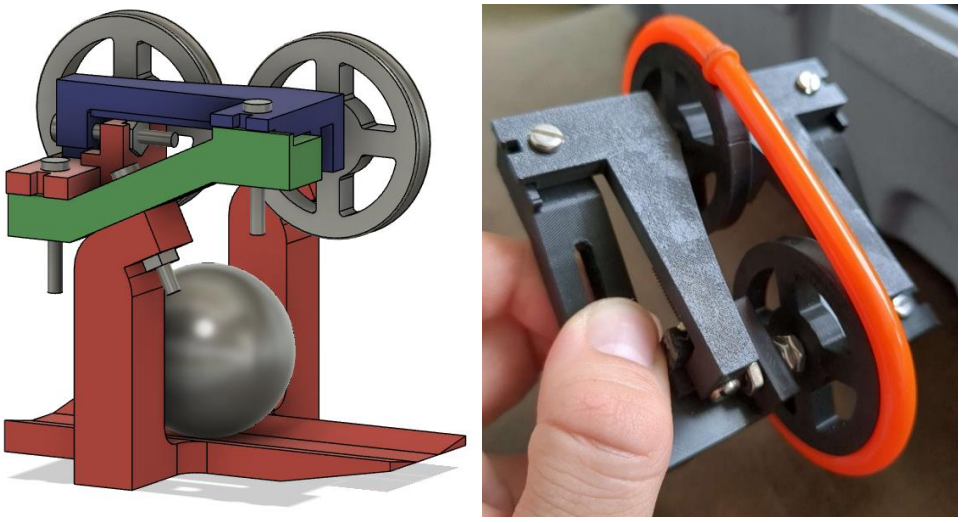

Figure 2.2 Polycord Intake Testing Rig

Our final design was intended to integrate the lessons learned from these test rigs alongside a sorting and storage system. The progress made on this design is shown in Figure 3. Most of the time in this model went into designing the intake system. This assembly utilized a spring tensioning system to keep the polychord tight and also used springs to adjust the vertical offset of the loop. The larger mounting structure was left to be manufactured last so we could accommodate as many of the remaining systems as possible (sensor mounting and ball storage), and we intended to make this out of acrylic for rapid fabrication of large structures. However, due to our previously discussed setbacks, we decided not to fabricate these structures to conserve materials.

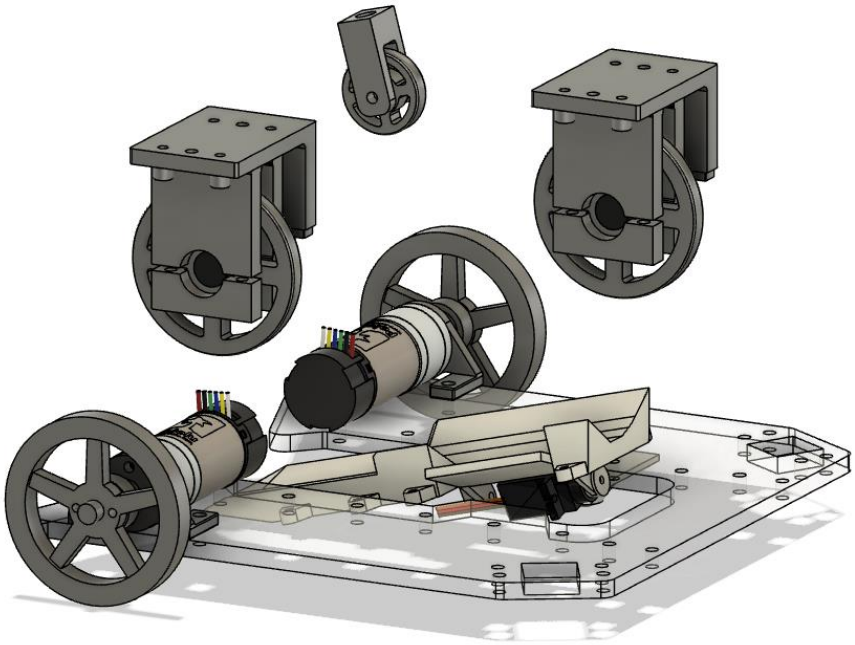

Figure 2.3 System Design Progress

In summary, our mechanical system was built based on our testing done along the way, but since the full system was designed late in the project, we weren't able to finish its manufacturing and assembly before the deadline. This, combined with critical issues with the electronics, resulted in us canceling the last phase of design and manufacturing, leaving us with few physical parts to demonstrate.

III. Electrical Design

The electrical design followed along with the class. Feedback from homework four and the first revision was critical when creating the final design. Assembly went smoothly, but the board failed during some last-minute component testing.

3.1 Computer Design

Our board design used homework four as a launching point and underwent two revisions based on instructor feedback before being ordered for manufacturing. Figure 3.1 shows the final electrical CAD for our board,

and complete circuit schematics are attached to this report. The attached Pin Allocation spreadsheet details where sensors and other peripherals were connected to the MCU.

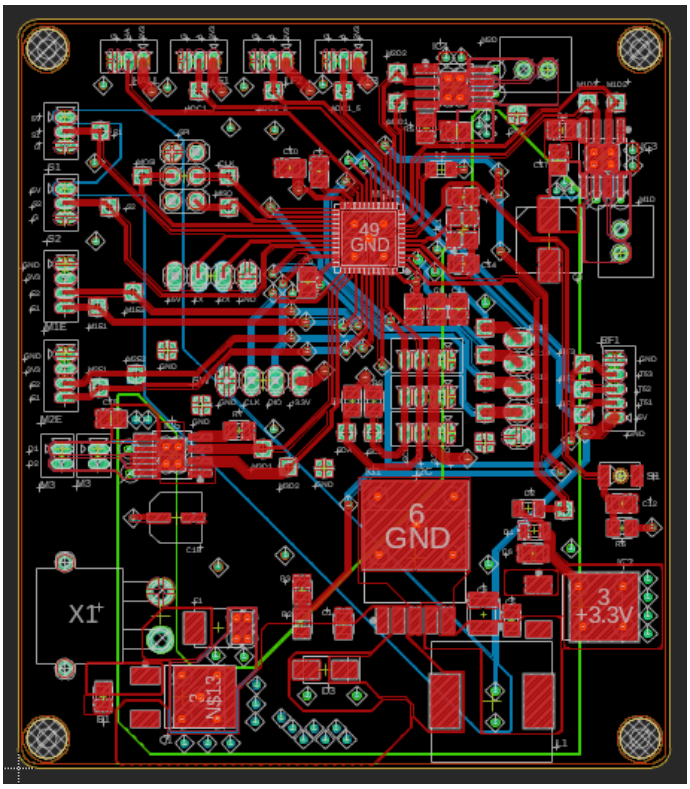

Figure 3.1 Electrical Routing. This figure shows the final routing used for our circuit board. Polygon fills have been turned off so traces can be viewed more easily.

Based on the feedback we received on the first revision of this design (feedback mostly pertained to our misunderstanding of how encoders functioned on the STM32), we were reasonably confident that the second revision would function as desired, so we ordered the PCBs from jlcpcb.com.

3.2 Assembly and Testing

Our board assembly went remarkably smoothly, with only minor reworking needed (a few MCU pins and a flipped Zener diode). We confirmed our power circuitry using test points and an LED on the 3.3V line, and we verified the MCU and crystal functionality by running code from mini labs 0 and 1 on our custom board. With these checks completed, we began developing our remaining drivers on our new hardware.

Due to timeline issues, we began developing the servo driver (our last component to receive attention) on our PCB on Monday, June 12th, two days before the competition. Our servos failed immediately, and during troubleshooting, we noticed that the 5V output of our switching regulator was instead outputting 10.7V, which likely caused our servo to fail. This issue persisted after replacing the regulator, and if we connected any of our peripherals to this line (all of which needed 5V power except for our drive motors), they would likely break as well. For these reasons and the lack of time available to seek a substitute, we were unable to start on system integration.

3.3 Final Board

The final board, although not wholly functional, does still allow for the control of the motors. The 3.3V linear regulator is still working correctly (for the time being), and we have verified that the motor driver/controller we wrote is working as expected. Figure 3.2 shows a picture of the fully assembled board.

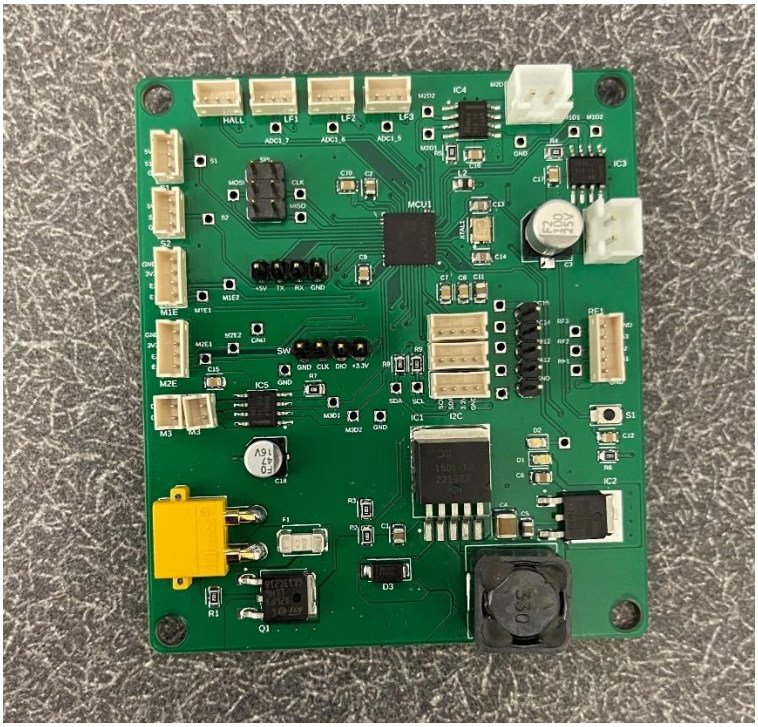

Figure 3.2 Assembled Board. This figure shows the fully assembled board.

The board was fully functional before the critical failure on Monday, June 12. All the motor power outputs worked properly, as did all the sensor ports. There was also no indication that we had misallocated pins, as all the classes we wrote to control sensors used the same pin allocations we had planned for this board.

IV. Software

The software development was done in the STM Cube IDE, which allowed for easy pin allocation and interfacing with the HAL API. We used a C++ object-oriented approach to communicate with all sensors and planned to use a finite state machine (FSM) based main loop to control the robot.

4.1 Custom Drivers

Classes were written for all intended sensors. We decided against using the line followers and hall effect sensors in favor of an image-based approach with Pixy2. We later adapted this to remove the Pixy2 and use Nathan's OpenCV-based web system with a Bluetooth communication module.

4.1.1 Pixy2 Driver

We wrote a custom pixy2 driver using an SPI protocol. The driver allowed for querying the camera to get position data for each detected object. It also implemented a checksum to be sure of data fidelity. The final class is easy to add to if additional functionality (such as line following) is required, as the only thing that changes is the message to send to the Pixy2. We shared our code with our classmates to help them develop their own code.

This class was difficult to get working due to some inaccuracies in the Pixy2 documentation, and our inexperience working with SPI. However, we did get it to return position data for the largest detected object of a specific color. A video of this functionality has been submitted alongside this report.

We ended up abandoning the Pixy2 in favor of Nathan's system because we felt that a top-down view would be far more useful for object detection and mapping than a view from the robot itself.

4.1.2 Color Sensor

We also had to write a driver for the I2C-based color sensor we decided to use. This wasn't easy because the I2C protocol appeared to be incorrectly implemented on the device we received. Based on our testing, a memory request would always result in the return of a command byte (the byte we sent) followed by the number of bytes requested starting from byte 0 of the device memory. This deviation from the standard should have been noted in the documentation and was extremely difficult to find/correct for. Eventually we got the sensor to return color data which depended on the color placed in front of the sensor. We did not have time to implement a correction for the detected colors, but changing the color in front of the sensor does change the values returned.

4.1.3 Motor Driver

We created a motor driver to work with the onboard motor driver ICs. This driver allows for directly controlling the motor effort using an input between -100 and 100. It also allows for position control with one of two algorithms: a standard PID controller, or a two-state controller which allows for the integral action to be turned on only when the system gets close to the final position. This means that the integral error term does not accrue while the system is saturated from the proportional gain.

4.1.4 Bluetooth Driver

To communicate with the system over Bluetooth, we wrote a Bluetooth class that used the UART communication protocol. The class allowed for synchronization with a companion Python program running on a laptop so that data transmission could occur after requests were made by the robot. By using interrupt-based communication functions, we were able to watch for incoming messages constantly, and if a stop message was received (by triggering a keyboard interrupt on the Python side) the robot could be stopped on the next pass through the main loop. This class allowed us to offload the processing of position data to the laptop which, in theory, would be able to process the information much more quickly.

This driver was not tested before the failure of the 5V rail, so refinements and corrections to the class could not be made. The companion Python program is in a similar state as both sides of the communication are required to test either piece properly.

4.2 Main Loop

We did not have time to implement the main loop of our program due to the electrical system's failure and the mechanical system's late start. However, we did create a diagram of how the program should work as an FSM. This diagram can be seen below in Figure 4.1.

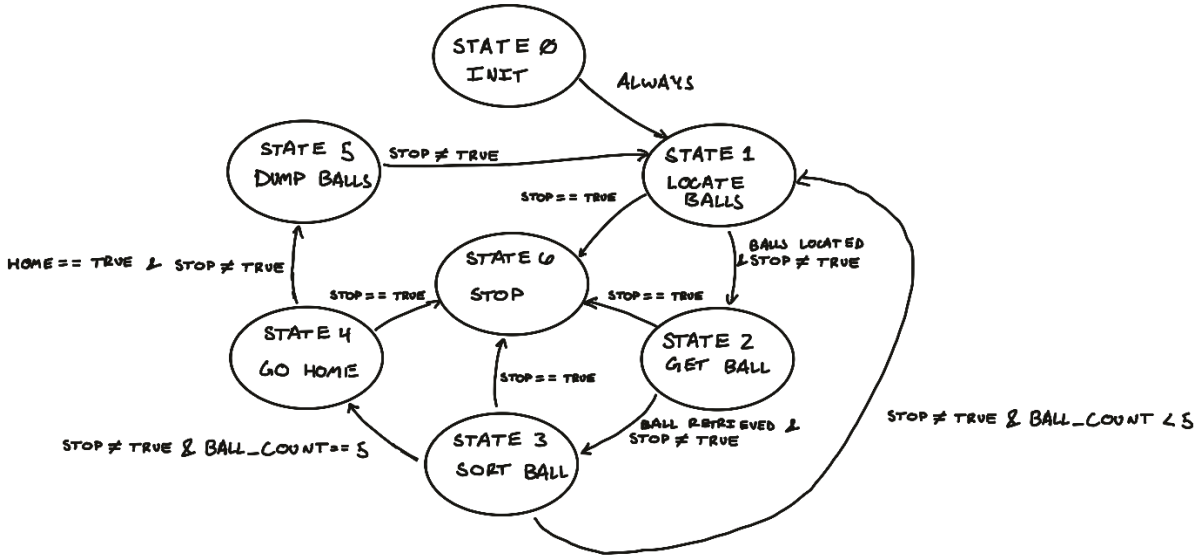

Figure 4.1 Finite State Machine. This diagram shows the preliminary program structure we intended to use during the competition.

This diagram is still in a crude state since we could not write the main loop due to the previously discussed setbacks.

4.3 Final Submission

Our final submission consists of several classes written to control peripherals. Each class was written as part of its own project to keep things compartmentalized, with these projects' main files containing a few lines of test and debug code as well as whatever variable declarations are required for the demonstration We do not have one project which contains all the classes, nor do we have a main control loop due to the setbacks previously discussed.

V. Conclusion

Like many others in the class, we were unable to bring our robot to a state that was competition ready. A combination of factors, namely commitments to other courses and last-second problems with the electrical system, prevented us from finishing the mechanical design and the software. If we were to do this project again, we would spend more time, in the beginning, working on the mechanical design. This would free up both group members to put more time into the software. We would also exercise more care when testing on our final board to avoid damaging the components. We feel that with 2 more weeks of time, we would have been able to have a functional design that would have been competitive with the other robots.