**MEMORANDUM** 

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

# Design Overview

## Initial Goals

## Component & Sensor Selection

# Mechanical Design

## Initial Design

## Challenges & Setbacks

## Final Design

# Electrical Design

The electrical design was done by Rees Verleur and followed along with the class. Feedback from homework 4 and from the first revision was critical when creating the final design.

## Computer Design

After completing the minimum working module in homework 4. We used the feedback to revise the power circuitry and create a first revision of the board design. After receiving feedback on this first revision, we completed the final design. The attached Pin Allocation spreadsheet goes into detail on where sensors and other peripherals were connected to the MCU.

Figure 3.1 shows the final electrical CAD for our board. Full circuit schematics are attached to this report.

A picture containing circuit, text, screenshot, electronic engineering

Description automatically generated

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

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

## Assembly and Testing

The assembly process went smoothly with only a few of the MCU pins needing to be reworked by hand. After testing we realized that our Zener diode, which contributes to the functionality of our DC-DC switching converter, was installed backward. After this small correction, the circuit seemed to function perfectly. All the power rails were at the correct voltages, and the indicator LED (while much brighter than anticipated) confirmed that the 3.3V rail was operating correctly. We tested the design with the codes from mini labs 0 and 1. This confirmed that the debugging process was working (including the included UART connector). By looking at the output of the interrupt-based LED blinking code on the oscilloscope, we confirmed that the timing of the pulses was correct and consistent. This indicated that the crystal oscillator on the board was working properly.

At this point we were confident that the board would function for the competition so we moved on to writing the motor driver. However, on Monday, June 12 (two days before the competition) while writing a class to control the servo motors. We realized that the 5V regulator (which was working perfectly after the initial assembly) was no longer functioning. The “5V” line was actually at 10.7 V which caused the servo connected to the board to burn out. This left us very little time to fix the board before the competition. After swapping the 5V regulator out with a spare we had ordered just in case, we found that the problem persisted. This led us to believe that the 5V rail was somehow tied to a higher voltage somewhere else on the board. Visual inspection and multimeter testing were unable to locate this short, so we were not able to move forward with the current board. We did not have enough components (or time) to populate a second board so at this point we realized that we would be unable to complete the project.

The timing of this critical failure meant that we did not have enough time to come up with an alternative for the competition. Troubleshooting this failure and trying to repair it also took up much of the time we had planned to spend coding the main loop (as the servo motors were the last external component to need a class) and finishing the mechanical assembly.

## Final Board

The final board, although not completely 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.

A picture containing electronics, circuit, electronic component, electronic engineering

Description automatically generated

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

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

# Software

The programming and software design was primarily done by Rees Verleur. 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.

## 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 further adapted this to remove the Pixy2 and use Nathan’s OpenCV-based web system with a Bluetooth communication module.

### 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. The driver also implemented a checksum to be sure of data fidelity. The driver is easy to add on 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 ended up abandoning the Pixy2 in favor of Nathan’s system because it was inconsistent with its detection of objects.

### Color Sensor

We also had to write a driver for the I2C-based color sensor we decided to use. This was difficult because the I2C protocol appeared to be incorrectly implemented on the device we received. This deviation from the standard was not noted in the documentation and was extremely difficult to find/correct for. Eventually we were able to get the sensor to return color data which seemed to correlate correctly with the color placed in front of the sensor.

### Motor Driver

We created a motor driver to work with the on-board motor driver ICs. This driver allows for directly controlling the motor effort (-100% - 100%). Or allows for position control with one of two algorithms. The first is a standard PID controller. The second implements 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.

### Bluetooth Driver

In order to communicate with the system over Bluetooth to transmit direction data and the emergency stop command. We wrote a Bluetooth class that used the UART communication protocol. The class allowed for synchronization with a companion Python program so that data transmission could occur after requests were made. By using interrupt-based communication functions we were able to constantly watch for incoming messages 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 driver was not able to be tested before the failure of the 5V rail, so refinements and corrections to the class were not able to be made due to the time limitation. The companion python program is in a similar state.

## Main Loop

We did not have time to implement the main loop of our program due to the failure of the electrical system, and the late start of the mechanical system. 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.

Insert picture of FSM

## Final Submission

Our final submission consists of a number of classes written to control peripherals. Each class was originally written as part of its own project in an effort to keep things compartmentalized. Each of these projects’ main files contains a few lines of test and debug code as well as whatever variable declarations are required for the functionality of the system. 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.

# Conclusion