# Software Overview

A picture containing diagram

Description automatically generated

Since the Preliminary Design Review, we have looked critically at our original software architecture and redesigned it to be both more streamlined and efficient.

Our goal is to ultimately build a modular and extendable system. We achieve this by ensuring a single source of truth by ensuring the state of our system is detached from any dependent systems, very similar to the idea of a state redux. We have selected various communication protocols that will used across our of system (as well as many other systems), while ensuring the specific method chosen is both efficient and reliable.

This design uses two processors, a Jetson Xavier NX (referred to as the Jetson) and a raspberry pi 4 (referred to as the pi). Vision processing is handled on the Jetson while the primary game control and motor communication is handled by the pi. This ensures that each process is able to be completed in a timely manner as well as allowing for processors to be switched at the developers convenience.

Images are fed in to the Jetson, where the necessary filters and calculations are ran, at which point an X, Y coordinate as well as a velocity vector are sent to the pi, updating the ball position. On the pi, various applications are polling the game state, such as the Decision matrix and motor communications that use the data from the game state to make decisions and/or update other portions of the game state.

With this goal in mind, the game state is designed to be updated on the fly, during execution. This means that, while the format is rigid, the content of the game state is not. If a "key":"value" pair that doesn't currently exist within the game state is sent to the pi, the key, value pair is appended to the game state, rather than ignored. This means that in the future projects or labs, data can be easily added and used with little change needed to the overall system.

## Inter-process Communication

We picked several different communication protocols in order to allow for efficient communication and control across the application. For the motors, CAN is optimal as it is designed for controlling multiple controllers at once. The reason that vision processing isn't added is because if we did that, we would flood the bus with the sheer number of packets moving down the line. This motivated our decision to use UART which is a very straight forward protocol that is both easy to learn, easy to use, and easy to implement.

In the pi, TCP sockets are used as a form of inter-process communication. TCP implements reliable data transfer, and allows us to implement a standardized method of inter-process communication that allows for different portions of the code base to be changed independently. The TCP sockets are also keep-alive and will maintain a connection until explicitly closed, drastically improving performance.

While there are multiple forms of inter-process communication and, while they might be faster, it comes at the cost of increased overhead. Using another inter-process method, such as a shared memory space, would quickly lead to an ever increasing overhead, that would easily confuse both us and whoever wanted to change the code.

# Vision and Perception

## Goals and Constraints

For vision, our goal is to be able to locate the ball using blob/color detection and reliably calculate location and velocity vector. This information is enough for the decision matrix to get a pretty good idea as to where the ball is, and where it will be.

A challenge with real-time processing of images, and any real-time processing in general, is leveraging the rate of processing with the required amount of data to be useful.

In the early stages of approaching this problem we outlined two parameters that would define the constraints of our vision processing. The graphs show the required processing times, dependent on frame rate, and the distance covered between two frames, at various ball speeds and frames per second.

Chart, histogram

Description automatically generated

Chart

Description automatically generated

As stated in our deliverables, our goal is to keep the processing time under 10ms, this meaning processing 100 frames per second. We will still make use the 260fps camera as, the program grabs the most recent image available, so if an image is processed faster than 10ms the decision matrix will be able to make use of the additional frames, and thus improve resolution.

The decision matrix will be able to make use of the velocity vector to make rudimentary predictions regarding where the ball will be and react accordingly. This means that if the ball is moving at 4m/s (the fastest calculated speed of the ball) the decision matrix will be aware of this speed and respond to the balls future location (assuming response speeds are sufficient).

## Current Progress and Feasibility

Currently from our tests, we’ve been able to obtain a ball location and velocity vector, all under the 10ms processing constraint.

Text

Description automatically generated

From the image above, it is clearly shown that we have achieved our 10ms processing constraint. This test was done running all necessary processes that are needed to process each image, with the exception of the normalization equation.

While we don’t have the frame for the table, when we do get it, and mount the camera, lighting factors may come into play, which might require some tweaking (although I don’t anticipate any real issues). For the expo, \*to my understanding\* our presentation will be in the same room that our lab is, so I’m not concerned about changes in lighting for the final presentation.

Speaking of lighting, there are two specific things We’ve done to address this:

1. We’re making use of the HSV (Hue, Saturation, Value) spectrum rather than the RGB spectrum. The HSV spectrum is known to be more reliable in various lighting conditions while still allowing for accurate color detection.
2. Using a fairly broad threshold for our color, covering both brighter and dim lighting conditions and using appropriate applications of blob detection to filter out any potential “false positives”.

Ultimately, we are well positioned to deliver the goals set out in our project goals in the time we have left.

# Graphical User Interface

One element we didn’t talk about in our Preliminary Design Presentation, how the user will interact with our table.

While their are numerous methods that could work, we will be implementing a touch screen that will be mounted on the frame, as seen in this image. The touch screen will, at minimum, allow the user to start/stop the game and view the current score of the game.

To Design the application we’ll be ElectronJS and industry level cross-platform toolkit of which many popular applications have been built such as VSCode and Slack.

We chose this toolkit as its highly flexible and compatible with a variety of stacks, both front-end and back-end.

As a proof of concept, the app would look something a bit like this:

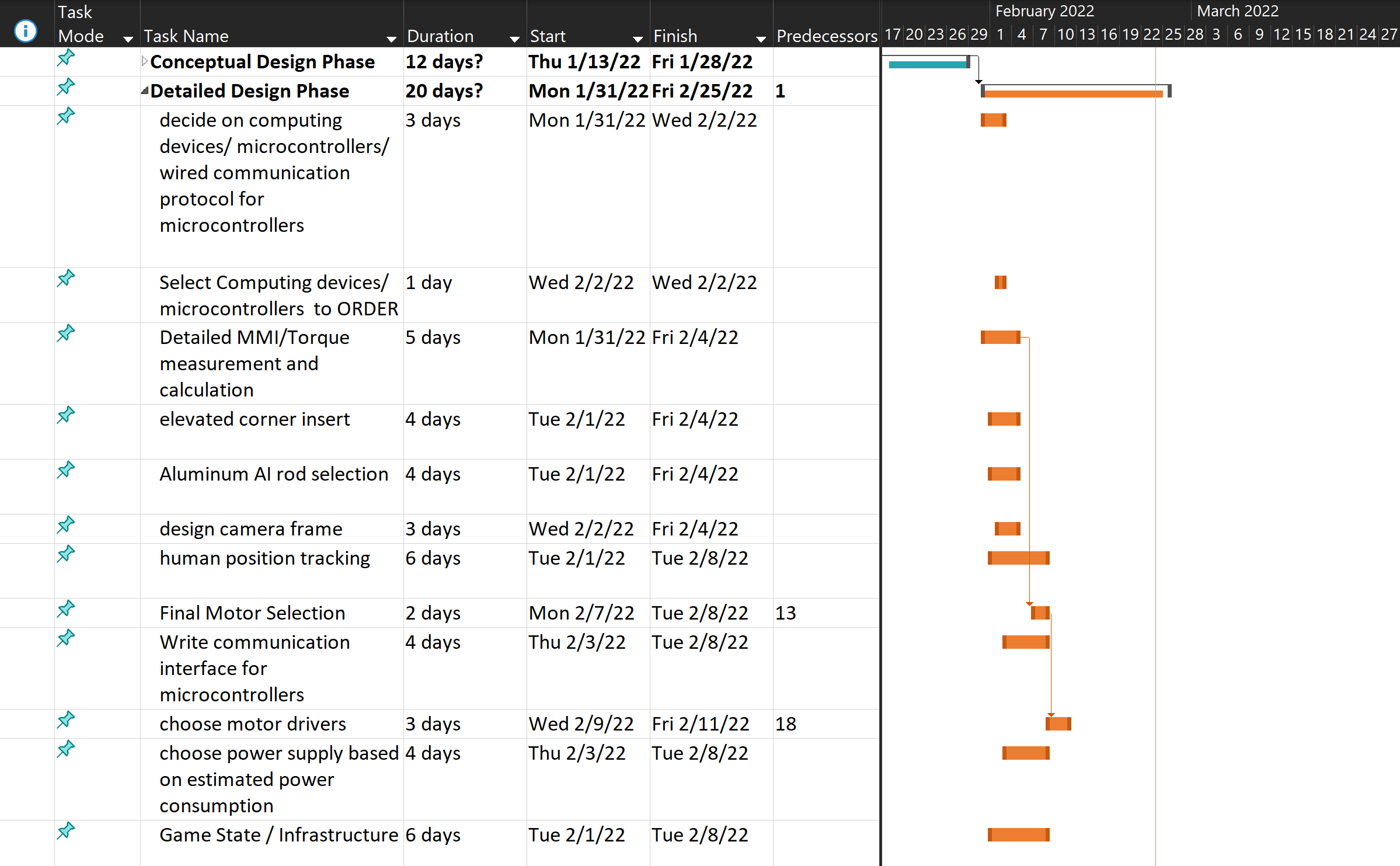
Graphical user interface, application, website

Description automatically generated

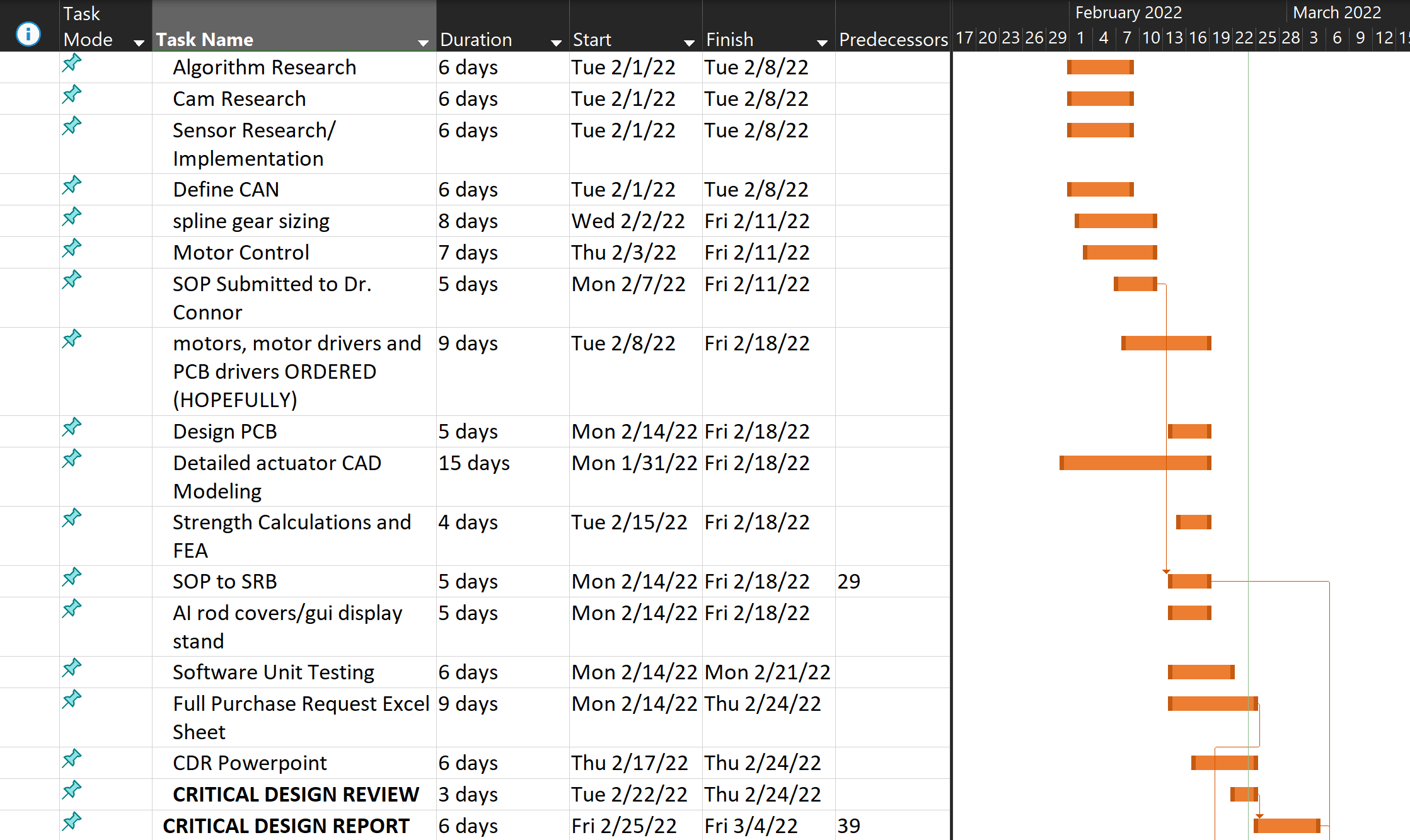
# Project Plan

Since Preliminary Design Review, we’ve been hard at work researching and fine tuning our approach. Taking into consideration much of the feed back from the PDR, we settled on our processor and motor selection, as well as designing the many CAD models needed to solve the mechanical challenges we face.

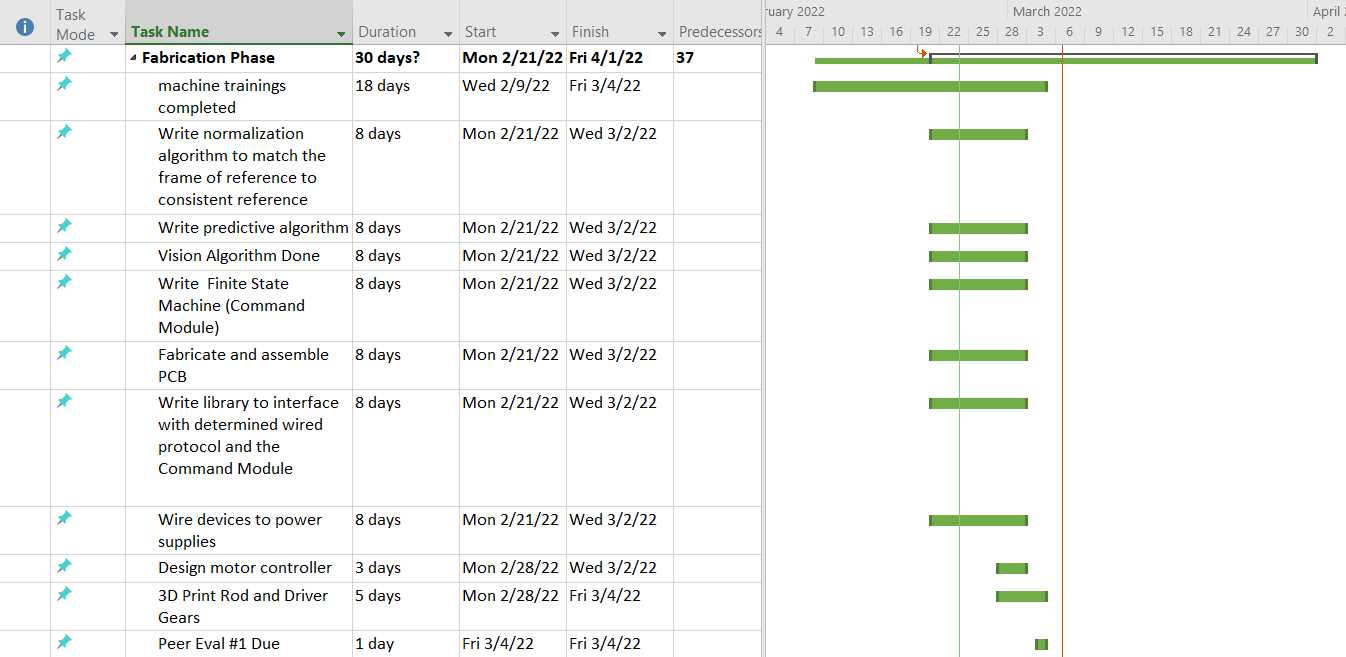
We’ve calculated power consumption to determine the necessary power supply we’d need and designed a the software infrastructure and game state.



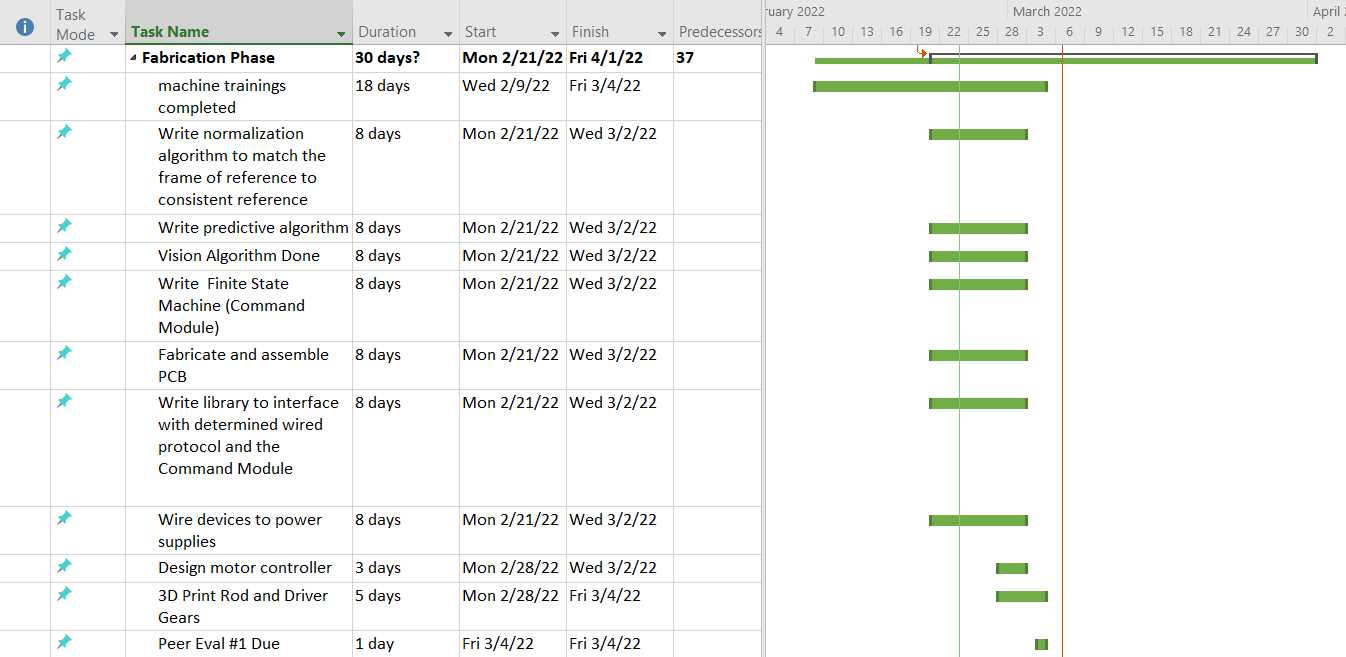
text



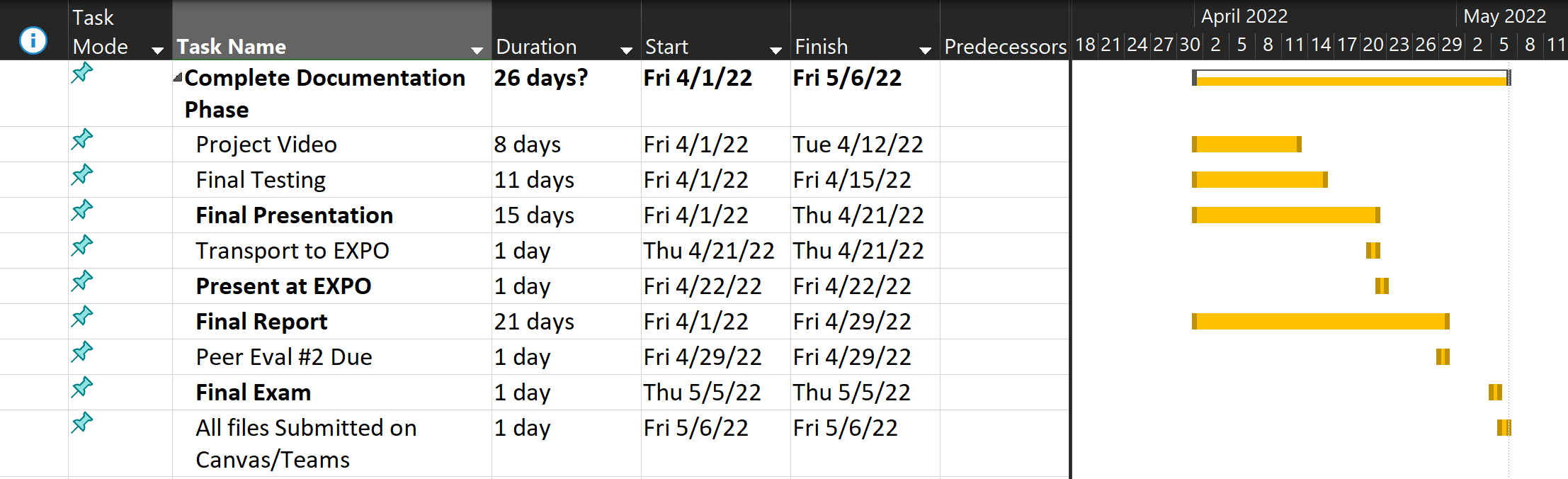
Text



text



Text



text