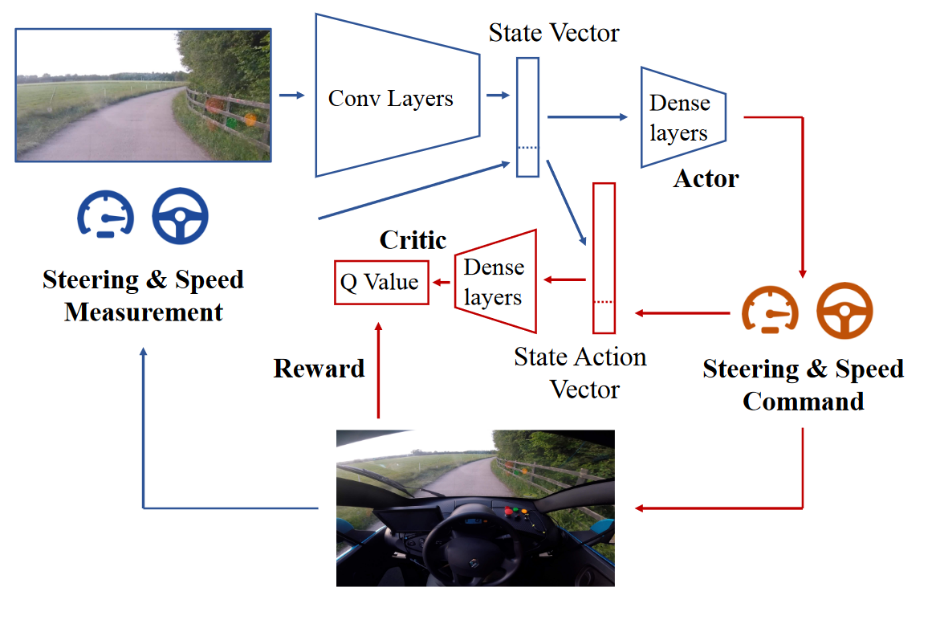
**Friday, November 17th, 2023**

I am fairly certain that a new motor controller will be necessary. I emailed the company which makes them to ensure I won’t have the same problems that I have been having with the two motor controllers we already have. They responded, saying that they ship their motor controllers with the latest firmware out of the box. Furthermore, they confirmed the output wires are capable of powering the servo motor for steering the car. They said they have all the parts so they will ship the next day (since I don’t want to be waiting too long for it to arrive).

I also found a reinforcement learning paper which tests their code on a small car just like I am trying to do:

“The agent receives an image from the on-board camera as input and commands desired throttle and steering angle…The primary reward is a weighted sum between a survival bonus (no intervention by the safety driver) and the commanded throttle. There is only the continuity cost as secondary reward. One episode terminates when the safety driver intervenes (crash) or after a timeout of 1 minute. Training is done directly on the robot”(<https://arxiv.org/pdf/2005.05719.pdf>)

The concept of a safety driver is one I haven’t seen before, but I think it is a good idea to implement in the real-world reinforcement learning. In simulation, the car can reset its position instantaneously to the start of the track. However, in real life this is obviously not possible. Keeping a replay buffer of previous actions and playing it backwards won’t work either due to tire slip and inaccuracies in the servo and throttle compounding over time. Thus, the idea of using a safety driver to keep the laps around the track continuously going is probably a good approach. I will be contacting the authors of the paper to see if I can get their code.

I also realized that I never linked the original paper showing a variational autoencoder applied to a full-sized car. It is here: <https://arxiv.org/pdf/1807.00412.pdf>. There is a really good diagram in the paper showing how the overall process works:

I sent a message to the first author of the first paper I linked (the one where they use a small RC car) asking for a few details about their safety driver implementation and how they integrated it with the reward function.

During 8th period, we laser-cut a redesigned laser cut plate. It fits much better onto the car:

A remote control car on a desk

Description automatically generated

**Monday, November 20th, 2023**

Given the situation with the motor controllers, I am verifying that the rest of the components are working. I tested the battery pack and confirmed that it is successfully charging. I am now working with the camera. I tried an example script which uses stereo depth to generate a point cloud map of the environment. Of course, this isn’t something I will need for my senior research project necessarily, but I wanted to ensure the camera is performing optimally.

A black and white image of a city

Description automatically generated(The picture to the left is from my point of view in the classroom).

Now that the battery pack is fully charged, I tested its discharge capability. First, I plugged my phone into it. According to the battery’s internal display, my phone is discharging at about 10 W. Then, I also plugged in my laptop’s charger. This was approximately a 70 W load. The battery pack did not increase in temperature significantly and it reported it would be able to continue charging my laptop for about 1 hour and 30 minutes. I estimate the maximum load the battery will have when deployed on the car is about 35 watts which means it will last for about three hours on battery, which is more than enough.

A close up of a device

Description automatically generated

I also tested the ability of the camera to recognize AprilTags, which are standardized markers used in industry/academia for localizing robots. The camera is able to detect AprilTags as long as it is within a few feet of the tag. This may be useful in the reinforcement learning process where I will need to know the car’s true position so that I can reset it after it crashes/leaves the track’s boundaries.

Finally, I began putting together a new purchase order for the motor controller and some other pieces I will need to assemble the car more easily. The biggest problem I am having right now is that each wire is far too long which is resulting in a terrible mess of cables on the car. Shorter wires/some specific adapter plugs will alleviate this issue.

I will need a some sort of test track to operate the car on. Ideally, the test track will have enough visual detail and consistency so that the auto-encoder can pick up the specific details of driving. I may also end up needing to incorporate vertical walls around the edges of the track to prevent the car from racing off somewhere else. I will research if pre-made tracks can be bought or fabricated. The easiest solution is to lay down tape on the ground.

**Monday, November 27th, 2023**

I received a response on my GitHub issue regarding the motor firmware. They told me that I needed to upgrade to an intermediate firmware version first in order to update the bootloader before upgrading to the actually correct version.

After trying out their instructions, I successfully updated the firmware on the motor controller!

I then uploaded a motor configuration file provided from a research group from UPenn. Their setup is similar enough to mine that most of the settings transfer over. I did have to do some tuning of the PID controller for RPM:

A screen shot of a graph

Description automatically generated

In this graph the RPM of the motor is graphed over time. Starting at a full stop, the motor is commanded to spin at 4000 RPM. However, to overcome the high static friction of the driveshaft, it has to spin up to 12500 RPM before settling down. I tuned the PID gains to eliminate the minor (5%) steady state error from the UPenn settings.

When the motor is already spinning, the step response is much better:

A screen shot of a graph

Description automatically generated

As you can see, the motor can precisely maintain a specific RPM value. Right now, I am spinning the wheels in the air, but the idea is that the RPM will automatically remain at the commanded value regardless of changes in torque, floor surface, turning, etc.

**Wednesday, November 29th, 2023**

Before ML (6th period) Shreyan and I went to laser-cut what should be our final mounting plate. The laser cutter wasn’t working so we will try again later this week.

I ensured the values I had tuned on Monday were still loaded onto the motor. They were. I was able to control the drive motor through Python as well. Next, I tried to control the servo motor, however, this did not work, either through the tool or Python. I will need to determine the source of this error, but servo control is simple enough that I can get it to work separately with extra components. It could be that the servo is faulty in which case I can replace it with the servo I ordered at the beginning of the year. Thus, I transitioned to getting the Nvidia board operating with the camera and motor controller.

First, I worked on establishing headless (without a keyboard, mouse, or monitor) control of the Jetson. I set up my computer to work as a wireless access point and manually connected the Jetson to it. I then set a static IP address for the Jetson so that I would not have to figure out what it is every time I turned it on. Then, I installed NoMachine on my computer and the Jetson. This is remote desktop software which will give me GUI access to the Jetson over Wi-Fi. Then, I started installing basic packages onto the Jetson. In particular, I want to make sure that machine learning libraries (TensorFlow/PyTorch/OpenCV) are installed with CUDA acceleration for maximum speed and efficiency.