Project Proposal:
I propose, for my final project, to improve the IMU estimation via the use of an accurate Kalman Filter/Extended Kalman Filter for the gyroscope, magnetometer, and the accelerometer. While the Kalman Filter itself has been implemented several time, and open source libraries exist such as TinyEKF. The goal of this project, then, is to best determine the model of the system as well as its parameters, then use those to better estimate the true position and possibly make a predictive model *a la* SLAM.

Last year, this project was also proposed and implemented, but with limitied results. For example, although they showed reduced noise in roll and pitch, the overall implementation was incomplete, and latency as well as some offset issues persisted in the program, suggesting an improper implementation. Additionally, the previous project directly estimated the bias and variance of the IMU through averaging. Although this may work for simple estimations such as the accelerometer, it may fail when extending the filter to nonlinear noise sources such as the quaternion measurements. It seems from this project that I would need to implement a better way to estimate the model and parameters in order to succeed with this project.

This problem of adjusting the Kalman gain has lead to some research with the automatic implementation. One paper proposed using a neural network to approximate the Kalman filter with Gaussian priors, which works for a single-dimensional Gaussian-noised model, while another suggested using probabilistic population coding (finding the estimates of the as an alternative that more

accurately represents a Kalman filter behavior. The neural implementation, published afterward, described PPC as more accurate, but possibly less robust to noise.

In terms of the project plan, my plan for this week is to be able to implement what the previous team did on my computer, using TinyEKF and the current VRduino filter code. I believe the starter code from the assignment, as well as the results paper, will be enough of a starting point for this project to implement the code.

Additionally, it could be helpful to implement the Kalman filter, as well as any "smarter" implementation from the mentioned papers, in Matlab. That way, I can benchmark any improvements made in speed or accuracy for a given tests set with noise.

Then comes the benchmarking for improvements. The two major problems associated with the previous assignment were unknown offsets and latency issues. Therefore, I will address both issues when running the code. First, I will create a toy model in C++ (as it will directly implement the final Kalman filter and not just the Matlab simulation), and test it on both a noised set of rotations and the actual Arduino model. Then I can move on to plotting any improved results on the Arduino serial plotter, and compare results. This will be over the course of the next two weeks, up until the poster session.