My experience at Space Dynamics Lab (SDL), in terms of computer programming, has been very good. I have mostly programmed in C++ and Matlab. I learned a little about C# as well working on a specific project they had me do.

Some of the general concepts pertaining to C++ were: functions, namespaces and anonymous namespaces, structures, classes, the boost library, the eigen library, the wxwidgets library and Anthemion Dialog Blocks (drag and drop application to making wxWidgets projects), in-house libraries at SDL, project setups, debugging experience, code standards experience, code reviews, pull request experience, github experience, git bash and git extensions experience, Jenkins software experience, and cmake experience.

For Matlab and C#, some of those concepts I have learned are: the windows form library, image processing, project setup, graphing and plotting, complex values, libraries from another company (Innovative-DSP), garbage collection, workspace viewing and manipulation.

There have been many different tasks that they have had me do over the past two and a half years. These tasks are in a chronological order. Some of these tasks include:

- I added functionalities to a graphical user interface (gui) that let the user input a comma separated values (.csv) file with certain headers and based on the lines following the header, it would populate the gui as needed. My work helped this gui be able to read in the .csv files as well as output new .csv files, giving you the ability to edit one specific .csv file, and it outputting that version of information, then editing it again. This .csv file setup different flight lines with different geometries.
- I created a unit test for some of the code that already existed. This set of code generated a couple images using an RGB value. This code also compared the images with each other and made sure that the images were different and created a new image showing the differences. The unit test was designed for when someone broke the code, this would catch it and prevent that person from introducing broken code into main repository of code. This unit test used the Eigen Value
- I interpreted different formulas and implemented them in Matlab to ensure they worked properly with image formation, then I took the code from Matlab and rewrote it in C++ to add it to some of the different gui's. Most of the formulas had to deal with image formation and different ways to process imagery in terms of time, processing power, and quality.
- I added different panels to a graphical user interface that was used for processing imagery. Based on the many sensors SDL uses, the data we obtain is slightly different between sensors. We must select the type of sensor so the application knows how to process it. The panel I created helped us distinguish different settings for the different sensors and aided with processing. The panel creation is mostly done in Anthemion Dialog blocks.
- I wrote a Matlab script that read certain fields from a database, grabbed the coordinates from different targets and created a .kmz file so it could be plotted on Google Earth and you could see the where each of the targets were located.

- I obtained an application from a different company that was written in C# and I learned how to modify their application. I learned how to draw on the application as well so we could get visual feedback from different settings that we were using. I also graphed the waveform that we were going to be using. This gui communicates to some field-programmable gate array (FPGA) cards and sets them up to send waveforms and receive returns from the waveforms for a radar application.
- I used some of the code I wrote for the application programmed in C# and created a new gui programmed in C++. I did this because we needed this applications to be cross-platform. The libraries mentioned above aid in applications being cross-platform. I made a lot of the panels and functionalities for this gui. The new gui's functions allow us to specify where the data is saved to, upload different waveforms, setup parameters for the FPGA card, and graphing capabilities for the waveforms as well as the incoming real-time data. Another main function of this application is after every event (whenever a field changes), it stores the values in a registry. When you have the settings as you want for the data collection, the values will be updated and saved. You are then able to click on a different button that will start the collect. When starting the collect the other functions will grab those values, calculate the timing required based on the parameters and set them as a hexadecimal value and convert that to a standard string (in order to set the FPGA card). It then sets the FPGA card with those values as well as the waveform and other constant values that we defined for our specific application. Also after you click the button to start the collect, I save out a new directory using the boost library. I save out a txt file that specifies what values were set on the specific registers on the FPGA card. I also copied the waveform file, stripped off the header, and saved both the original copy of the file as well as created a new file of just the waveform. This was required because the FPGA needed to send only the waveform, but we are required by work standards to have the header on the file.

There may be a handful of smaller tasks that I did, but these are the main programming tasks that I have done while I have been at SDL.

Trevor Brown