Progress so far:

The software up to this point is complete except for fixing bugs that future testing may reveal. The software works by first allowing for the user to crop the top of the image. This removes the competing robot which could be one possible source of noise during object detection. After that the software attempts to auto-calibrate the board edges. If this calibration is bad I have it set up so you can click the 4 corners of the board and it will use those as the edges. After we have the edges the user can set the score zones. Lastly it uses color blob detection to find the location of the puck and find the percentage of its x position between the edges. It then multiplies the percentage by the max encoder value and sends that to the fpga.

Issues:

Since last lab report I have found that the score zone lines are pretty much invisible to the camera due to a combination of both height and angle of the camera and how faint they are on the board. This means that I will not be implementing an automated system for detecting the score zones since anything I could do would be horrible inaccurate and have to be set manually anyways.

Along the same lines the board detection algorithm works really well but is not of the quality that I need in order to aim the shooter. Because of this we mostly use the manual calibration which due to some usability changes a made takes about 5 seconds which is not that large of an issue.

Another issue that came up was with integrating the software with the hardware through a BusPirate SPI interface. I was under the impression we would have a normal serial interface but was mistaken so I had to rewrite all of the serial code I had written to use the BusPirate library. Sadly the python BusPirate library was really slow when sending data so I had to extract the code from the library that did the sending and write my own version that was quicker. This solution got us near instant sending time.

We also had an issue with our aiming system. As I was sending positions to the FPGA they were always slightly off but is wasn't a normal function like I would expect from camera distortion. I still don't know what caused it but we managed to fix through a use of an offset in the value I send as well as an adjustment to the total encoder scale.

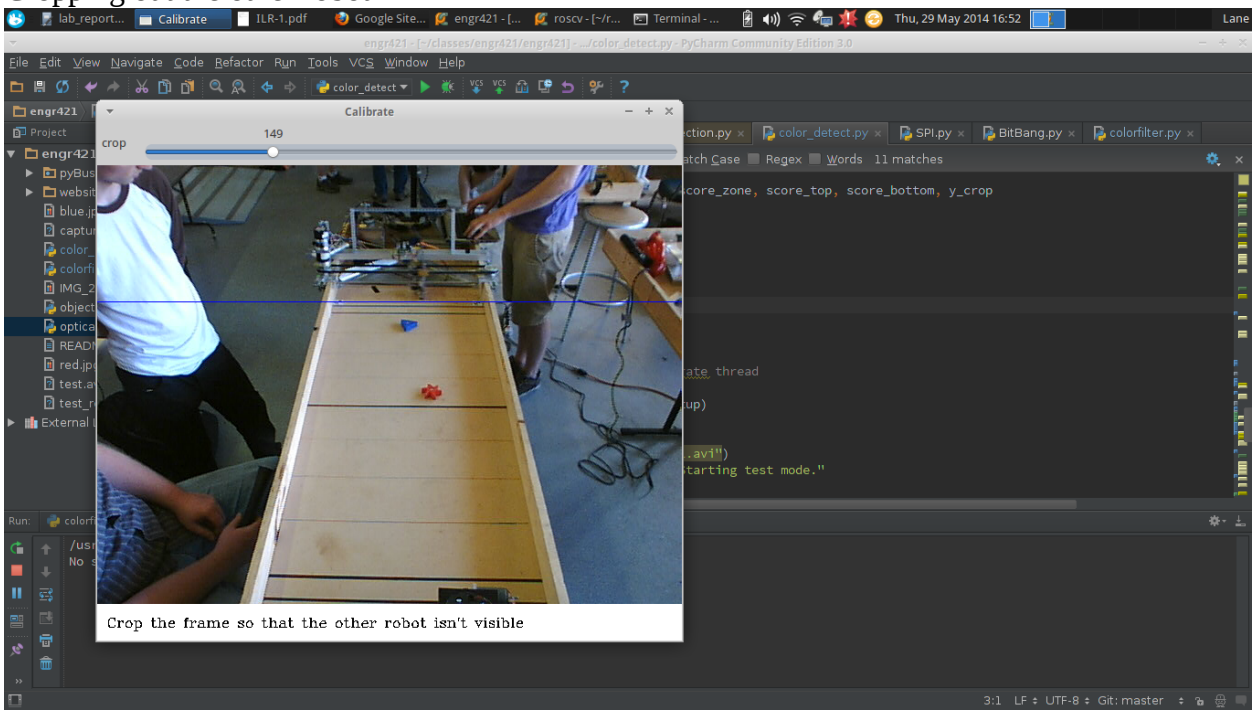
Team Goals:

The software being so close to completion is good because it allows for us to test the hardware more than we could have if I had been behind. It also gives us time to refine the system and find small bugs in the software like the aiming bug mentioned above.

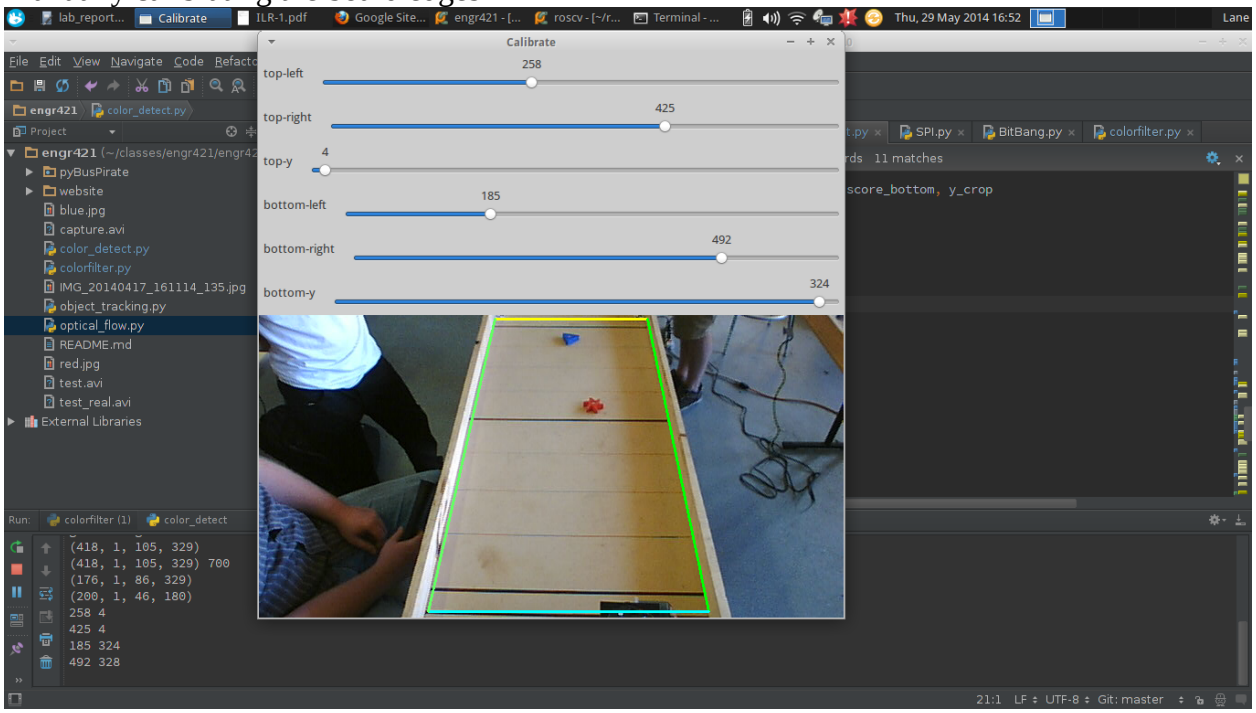
Plans for the following week:

I am pretty much done so I will continue testing and try to refine the software hardware interaction and make sure that we use the best possible strategy for the final competition so that we will win.

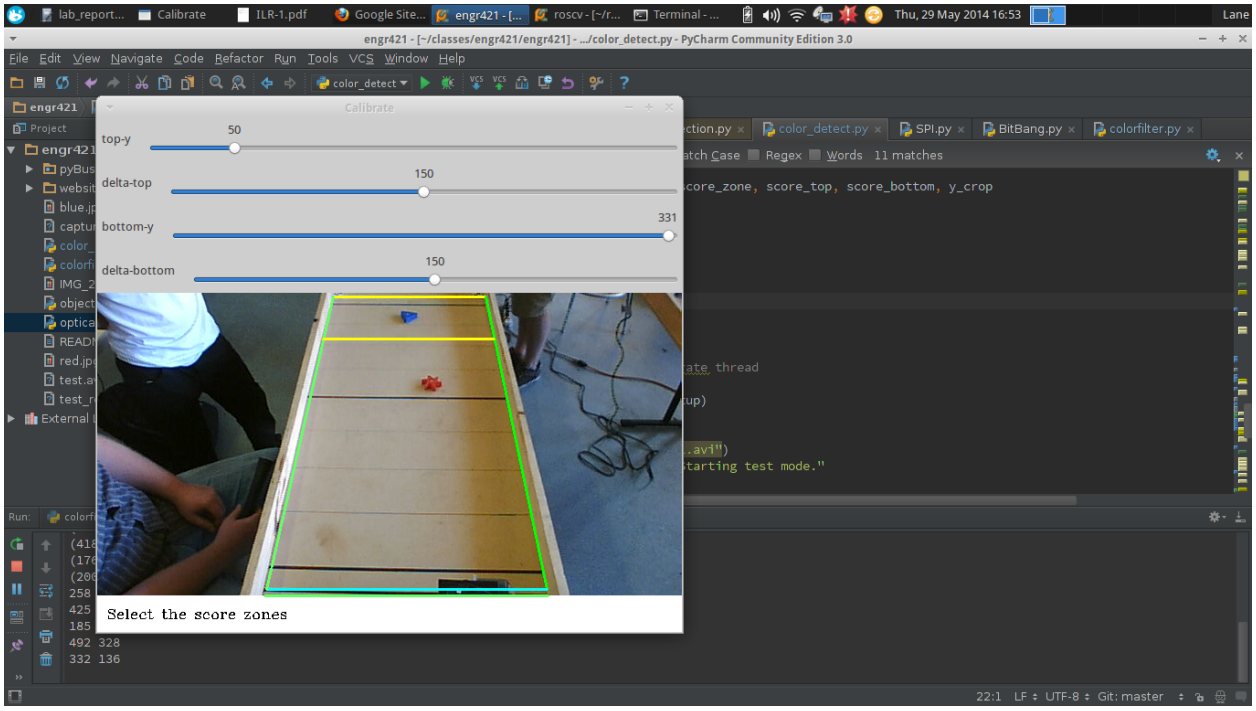
Cropping out the other robot



Manually calibrating the board edges



Set Score Zones:



Final Detection: