Developer Documentation

My robot arm was functional, but required very accurate positions. The way I created my coordinate grid required the paper to be just inside the outer edge of the camera frame, too close and it will swing wide. Too far and it will fall short. My camera is effectively dangling and requires that no one hit or jostle the table it is on. Excessive shadows on the paper will cause issues with misinterpreted shapes or phantom images.

To start my robot you send the Arduino code, then you turn on the C# code. Once both are ready then you start the program with a well labeled “Start” button in C# form. Just ensure the shapes are placed before start is pushed. The rest happens on its own once start is pushed. The robot will locate and sort all shapes usually in the order of closest to farthest away.

Known issues include that the code does not stop when there are no shapes or even if C# is stopped. The Arduino will continue after phantom shapes forever. The shutdown procedure is kill the power supply to the robot once it is done. It will always go after real shapes first before going after phantom shapes.

Agile User Story

My project was straight forward from the beginning. The original design I imagined in my head was very nearly the final functioning design I used to complete the project. Only a few small things like arm total length and the shape of the base changed. The build had a few small issues, but was quite sturdy. It does not allow access to one of my servos without significant work, but with the properly mounted bearing absorbing all the weight, the servo will only have stress when it is turning the base upon its proper axis of rotation. None of my servos have any sideways or shear forces thanks to proper mounts and bearings. I had an issue with my arm being too flimsy and short and it needed to be rebuilt twice. I could use the old arm as a guide for re-drilling my holes which decreased the time and effort of that process.

Coding the robot was a much tougher process than the physical construction of the arm. I had to work through many iterations of code with a lot of trial and error. The largest issue was serial communication. I still struggled to fully comprehend what a few fragments of code in that interaction fully did. With that in mind I tried to minimize how much was being sent. C# created a coordinate grid and identified triangles and squares. All that information was conveyed with 3 whole numbers. Arduino did the complex math for converting arbitrary grid numbers into usable angles for the servos. The Arduino code was surprisingly simple baring the actual communication part. The C# code took much more debugging and testing.

In short, physical construction was straight forward and went very well. The programing had elements go well, but more elements that did not go well and required extra time to make function properly.