Rubik's Cube Labview Code Review

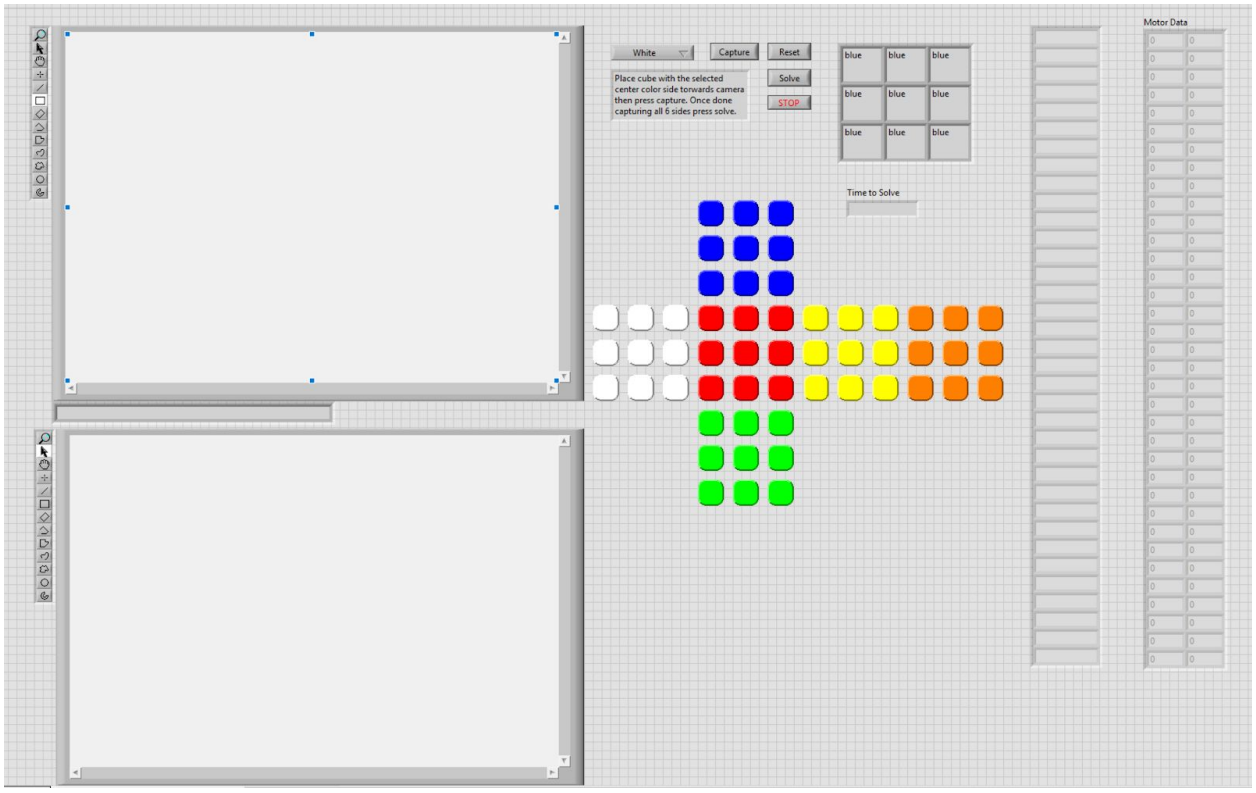

Figure 1: Computer front panel GUI. This was used to get image data from a cube placed on a table and also live feed a video stream of the cube inside of the robot.

One of the biggest challenges with this front panel was making it look nice while not messing up the code for the coloring of the decorations. To have the interactive colors on the front panel changed we had to use the indexing that originated from the order that the decorations were placed in. I then placed the squares in the specific order needed. To get around issues of placing text blocks that count as decorations that would mess up the indexing I used indicators and wrote the instructions in string constants. This way I could update one string indicator to have different instructions without interfering with the decorations. I also tried to make the front panel streamlined and easy to use so that it would be easy for anybody to use.

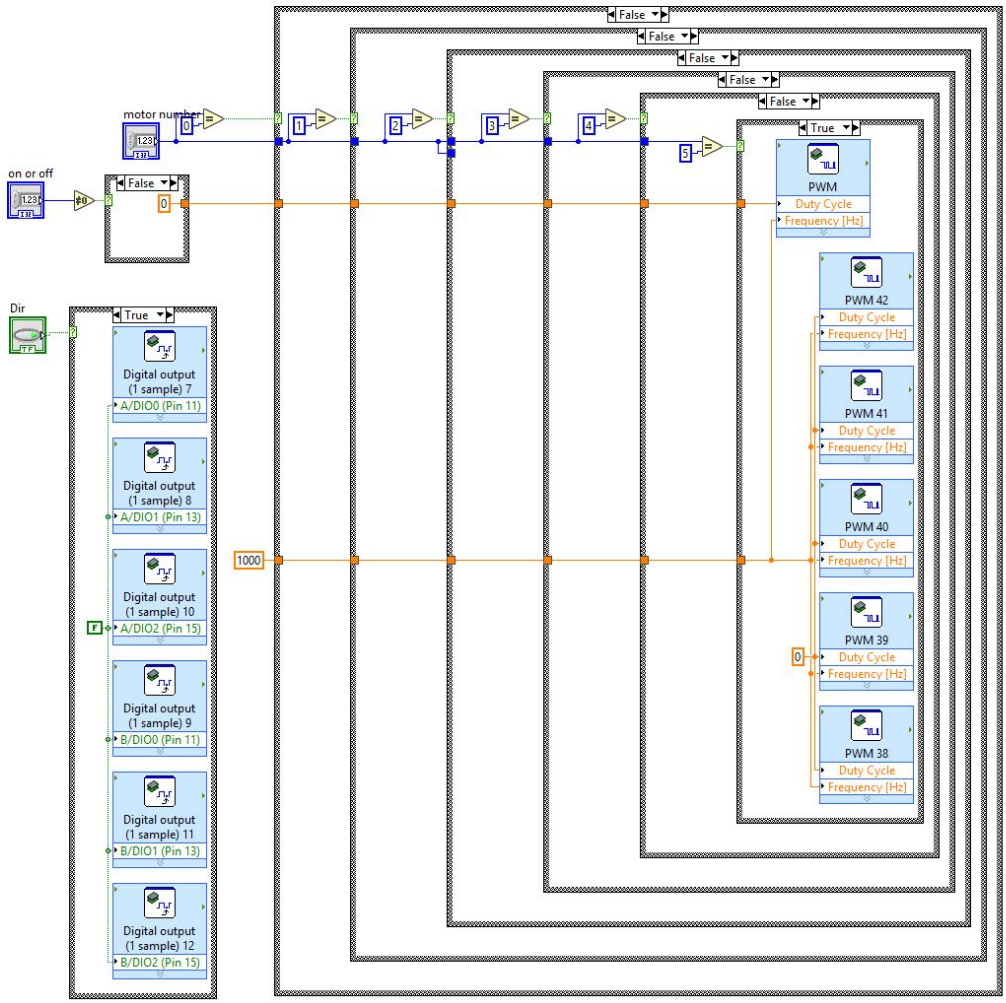

Figure 2: First attempt at writing a pwm signal to motors with a while loop.

This is the first attempt of writing pwm values this was done by writing to 6 different signals. While this was effective it caused a lot of strain on Labview's graphical interface which started lagging. We completely avoided this as it had to main express vi's.

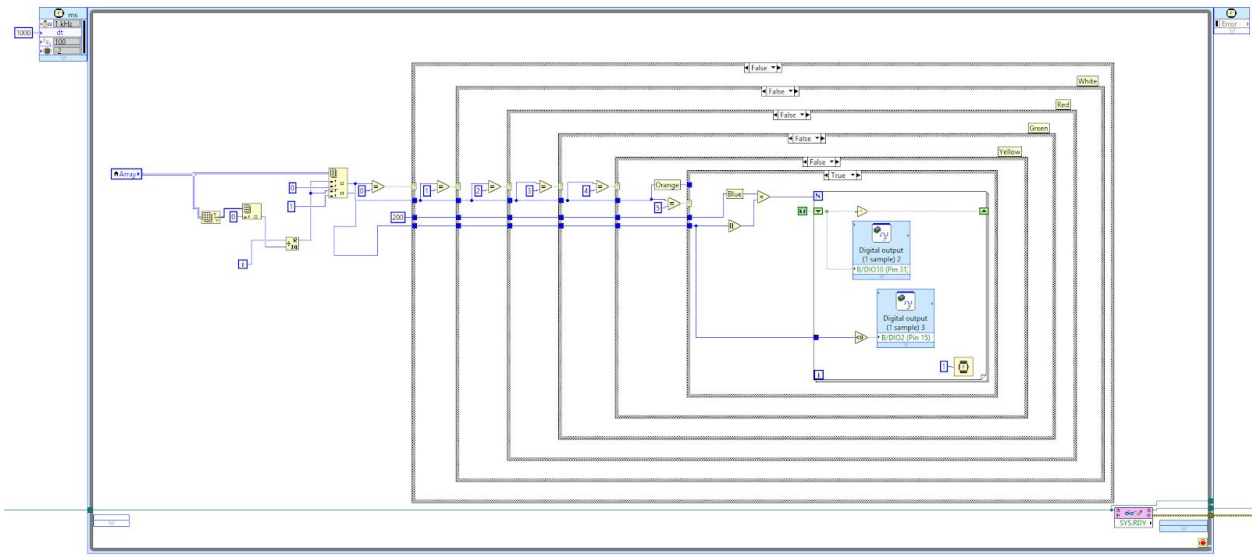

Figure 3: This was code worked to write to the motor pwm. The only issue was that the frequency of the pwm signal was at 1000 Hz which was very slow.

This was our first attempt at creating a working motor writer. This worked with the only issue being that the motors did not turn very fast. This was our first working attempt with labview and in the future to achieve better results we want to write our own program on the FPGA level to write a digital signal to actuate a stepper motor 90 and 180 degrees. This was the optimal solution to our issue however time was the largest constraint against us.

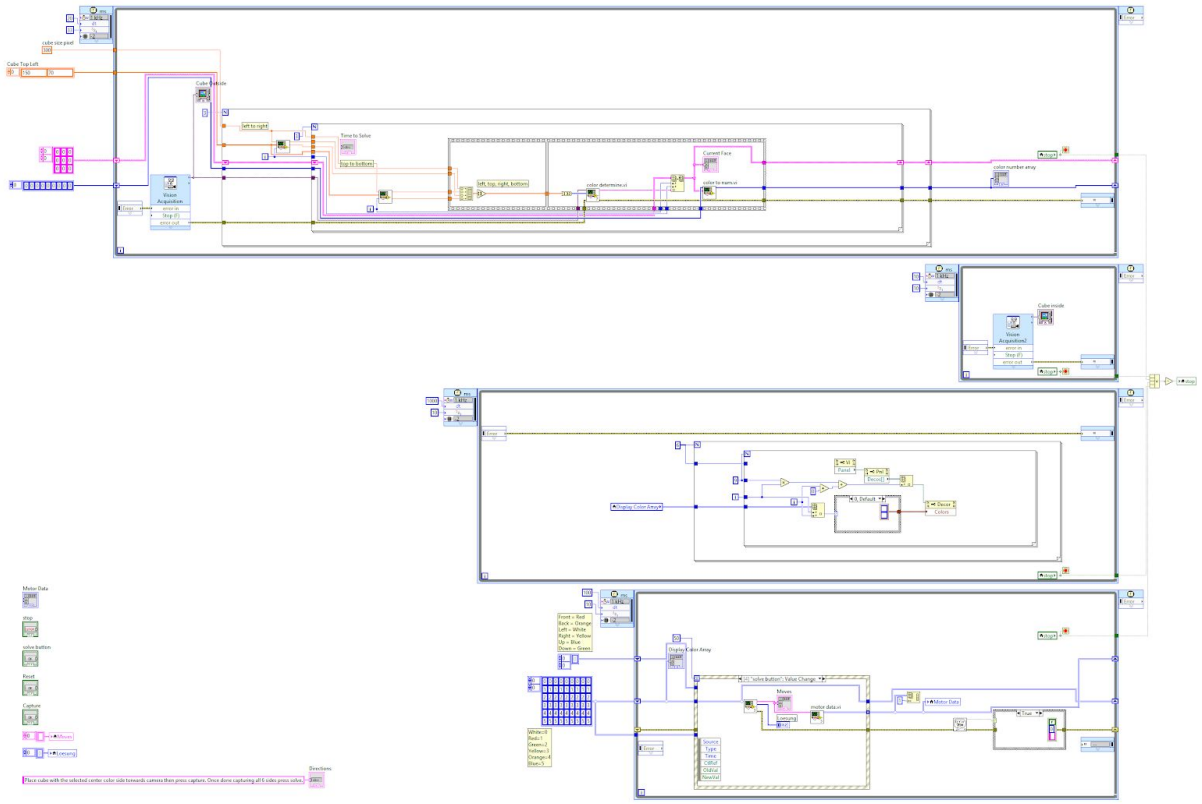

Figure 4: Computer GUI block diagram.

The structure of the computer GUI block diagram is made with 4 while loops. The top loop runs the color algorithm that was custom written to improve performance. This was done by averaging the RGB values and custom writing a tree diagram (see Figure 5) to solve what color is being scanned. Using Labview's vision assistant would have added extra strain on the computer and allowed us to run our front panel at a very high speed. The second loop was the camera looking inside of the robot just to show what was happening on the inside. The third while loop changed the decorations on the front panel to match what was being scanned on the outside rubik's cube. The last timed loop changed the array of the cubes current situation and also had an event structure which would go to an invent to solve the cube.

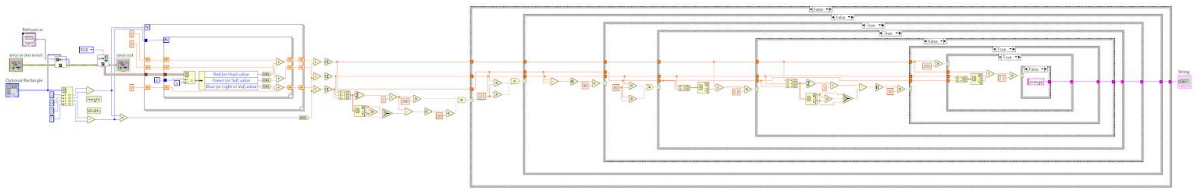

Figure 5: Color determine block diagram.

Conclusion: To continue use with the MyRIO we would need to write some code to make 90 and 180 degree turns. We would do this by writing our own custom FPGA and write a digital square wave that could actuate the motors at a high frequency. In the FPGA we could even write a ramping style

acceleration curve which would allow us to be able to do even faster turns. The main limitation of our project was time and the ability to tune the robot, and how well we knew the fpga programming.