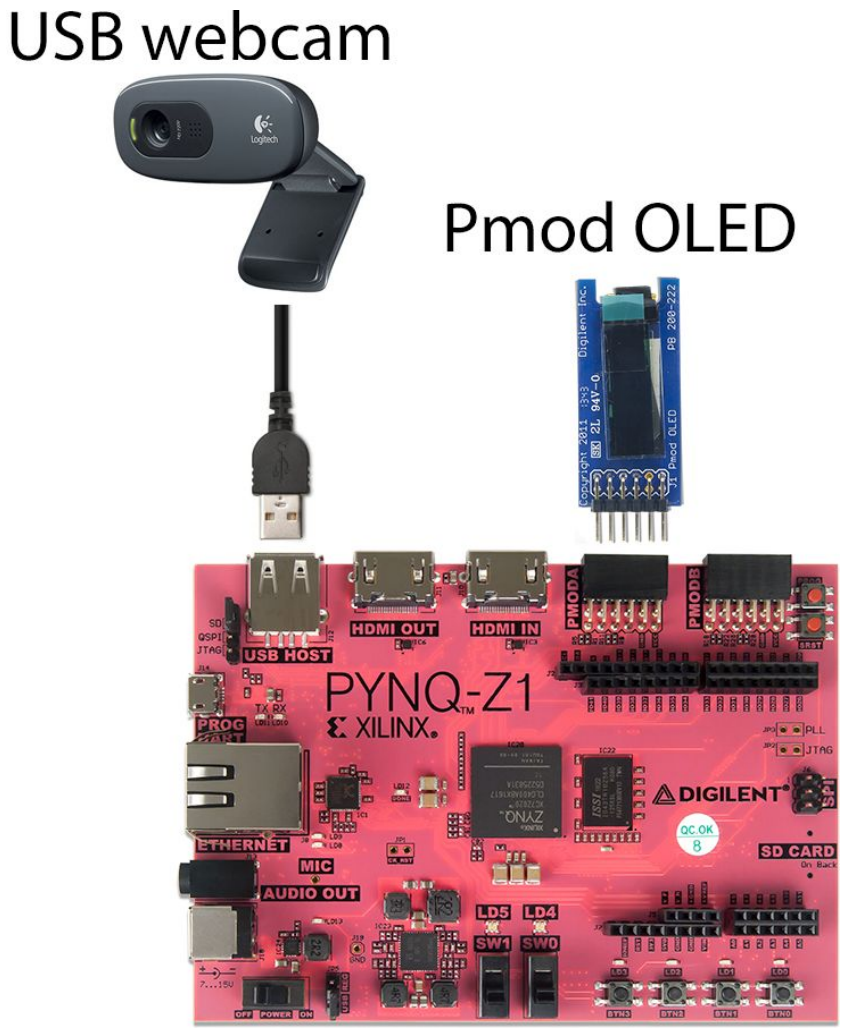
William Scotten
47821486
EECS 113 Spring
6-5-17

Actioniks Cube Solver

Introduction

For this project, I decided to make a Rubik's cube solver. The basis of the project is that a person with no knowledge of how to solve a Rubik's cube can solve one by having the Pynq board take pictures of the cube using the USB webcam and follow the instructions given to them on the Pmod OLED. If a person follows the instructions correctly, they will hold a solved cube in their hand when they are done. All of the programmed logic is written in python and ported to the board through the Jupyter Notebook environment. Once the user runs the code in Jupyter, they will be guided step-by-step through the Pmod OLED.

Schematic

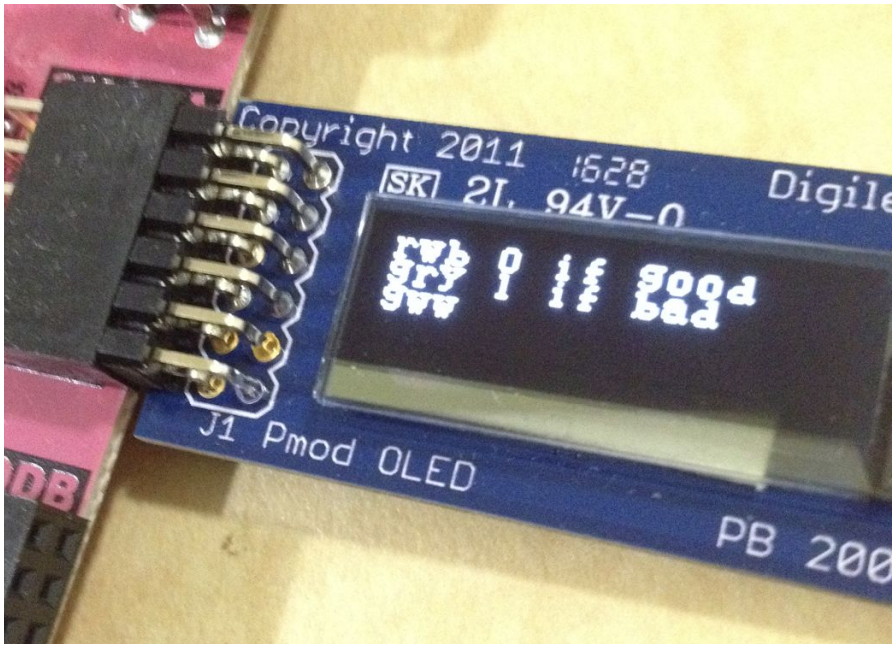


How to Use

Taking Pictures of the Cube:

To take a picture of the first side, the user places the cube with the white side face up beneath the usb webcam. In order to snap the photo, the user clicks button 0 on the Pynq board. The Pmod OLED will then print out what the board registered as the nine colors of the board. The user then can choose whether those are colors are correct to move on to the next side, or if they are not, they can retake the picture of that side of the cube by pressing Button 1 on the Pynq board. The order of taking the pictures is Front > Top > Back > Bottom > Left > Right. The board can only solve the cube if the front facing side is the white side. Once all of the pictures have been taken, the board will calculate the solution, assuming that a possible cube orientation was provided by the pictures.

Example Output after Picture is Taken:



Following the Solution from the Pmod:

Once the board is done calculating the solution (it should only take a second or two), it will the user the first move to make. The board assumes that the user starts solving the cube with the same starting orientation as the one in the pictures. For instance, holding the cube with the green side on top but having the Pynq board think the orange side is the top will result in an unsolved cube. The Pmod prints out the moves one by one, and waits for the user to press Button 0 before moving on to the next move. The names of the moves the Pmod reads back to the user are the same as what people in the Rubik's Cube community understand. For instance, "Right Inverted" as understood by Cube solvers around the world is the same "Right Inverted" that is read back to the user. The visual representations of the moves are shown below. When the Pmod reads "Inverted" after a move's name, it just means go in the opposite direction of that move. When the Pmod reads "Twice" after a move's name, that just means execute that move twice.

Example Solution Output after all Sides are Inputted:

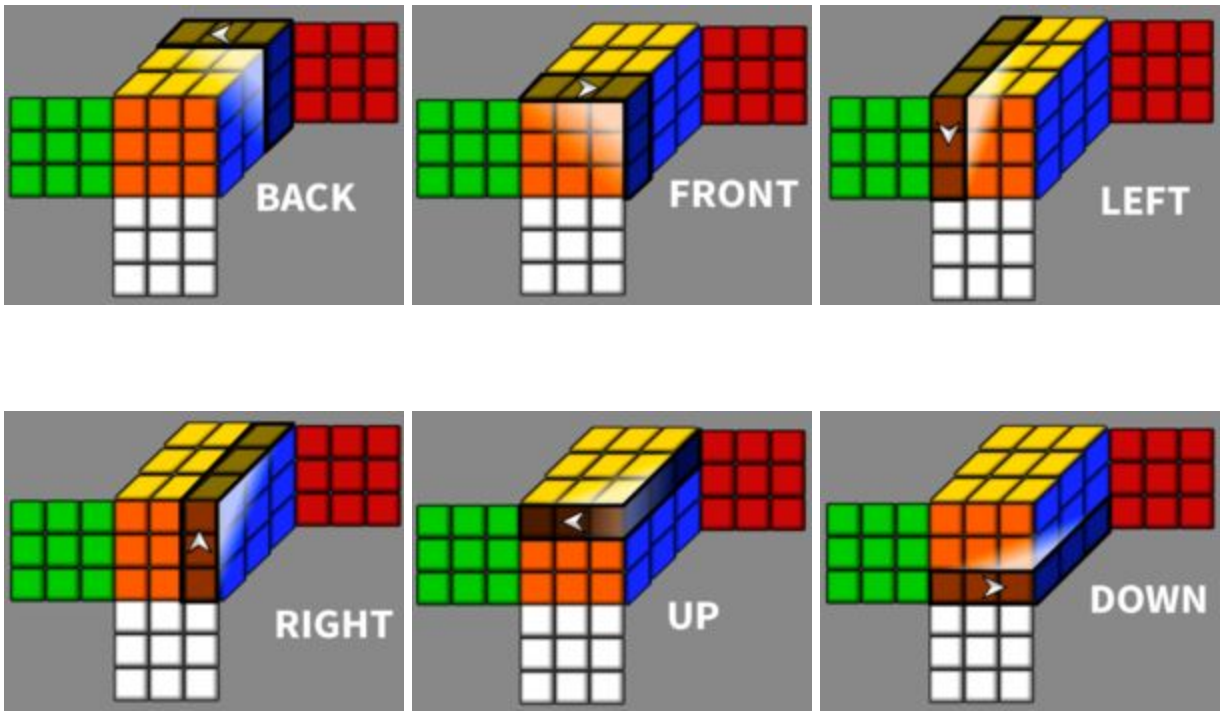
```
        o o o
        o o o
        o o o

g g g | w w w | b b b | y y y
g g g | w w w | b b b | y y y
g g g | w w w | b b b | y y y

        r r r
        r r r
        r r r
```

```
["R'", "B'", 'R', 'B', 'B', 'C', "R'", "B'", 'R', 'B', 'B', 'C', "R'", "B'", 'R', 'B', 'C', 'B', 'C', "R'", "B'",
 'R', 'B', 'C', "R'", "B'", 'R', 'B', "R'", "B'", 'R', 'B', 'C', 'B', 'C', "R'", "B'", 'R', 'B', "R'", "B'", 'R',
 'B', "R'", "B'", 'R', 'B', "R'", "B'", 'R', 'B', "R'", "B'", 'R', 'B', 'C', 'C', "B'", "L'", 'B', 'L', 'B', 'U',
 "B'", "U'", 'B', 'B', "L'", 'B', 'L', 'B', 'U', "B'", "U'", 'C', 'C', "B'", "C'", "B'", "L'", 'B', 'L', 'B', 'U',
 "B'", "U'", "C'", "B'", "C'", 'B', 'R', "B'", "R'", "B'", "U'", 'B', 'U', 'C', 'C', "B'", "C'", 'B', 'R', "B'",
 "R'", "B'", "U'", 'B', 'U', 'C', 'U', 'R', 'B', "R'", "B'", 'U', 'R', 'B', "R'", "B'", "U'", 'C', 'C', 'R', 'B',
 "R'", 'B', 'R', 'B', 'B', "R'", 'B', 'B', 'R', "B'", "L'", 'B', "R'", "B'", 'L', "R'", "F'", 'R', 'F', "R'", "F'",
 'R', 'F', "B'", "R'", "F'", 'R', 'F', "R'", "F'", 'R', 'F', "R'", "F'", 'R', 'F', "R'", "F'", 'R', 'F', 'B']
['Right Inverted', 'Back Inverted', 'Right', 'Back Twice', 'Rotate Clockwise', 'Right Inverted', 'Back Inverted', 'Ri
ght', 'Back Twice', 'Rotate Clockwise', 'Right Inverted', 'Back Inverted', 'Right', 'Back', 'Rotate Clockwise', 'Bac
k', 'Rotate Clockwise', 'Right Inverted', 'Back Inverted', 'Right', 'Back', 'Rotate Clockwise', 'Right Inverted', 'Ba
ck Inverted', 'Right', 'Back', 'Right Inverted', 'Back Inverted', 'Right', 'Back', 'Rotate Clockwise', 'Back', 'Rotat
e Clockwise', 'Right Inverted', 'Back Inverted', 'Right', 'Back', 'Right Inverted', 'Back Inverted', 'Right', 'Back',
'Right Inverted', 'Back Inverted', 'Right', 'Back', 'Right Inverted', 'Back Inverted', 'Right', 'Back', 'Right Invert
ed', 'Back Inverted', 'Right', 'Back', 'Rotate Clockwise Twice', 'Back Inverted', 'Left Inverted', 'Back', 'Left', 'B
ack', 'Up', 'Back Inverted', 'Up Inverted', 'Back Twice', 'Left Inverted', 'Back', 'Left', 'Back', 'Up', 'Back Invert
ed', 'Up Inverted', 'Rotate Clockwise Twice', 'Back Inverted', 'Rotate Counter-Clockwise', 'Back Inverted', 'Left Inv
erted', 'Back', 'Left', 'Back', 'Up', 'Back Inverted', 'Up Inverted', 'Rotate Counter-Clockwise', 'Back Inverted', 'R
otate Counter-Clockwise', 'Back', 'Right', 'Back Inverted', 'Right Inverted', 'Back Inverted', 'Up Inverted', 'Back',
'Up', 'Rotate Clockwise Twice', 'Back Inverted', 'Rotate Counter-Clockwise', 'Back', 'Right', 'Back Inverted', 'Right
Inverted', 'Back Inverted', 'Up Inverted', 'Back', 'Up', 'Rotate Clockwise', 'Up', 'Right', 'Back', 'Right Inverted',
'Back Inverted', 'Up', 'Right', 'Back', 'Right Inverted', 'Back Inverted', 'Up Inverted', 'Rotate Clockwise Twice',
 'Right', 'Back', 'Right Inverted', 'Back', 'Right', 'Back Twice', 'Right Inverted', 'Back Twice', 'Right', 'Back Inv
erted', 'Left Inverted', 'Back', 'Right Inverted', 'Back Inverted', 'Left', 'Right Inverted', 'Front Inverted', 'Righ
t', 'Front', 'Right Inverted', 'Front Inverted', 'Right', 'Front', 'Back Inverted', 'Right Inverted', 'Front Inverte
d', 'Right', 'Front', 'Right Inverted', 'Front Inverted', 'Right', 'Front', 'Right Inverted', 'Front Inverted', 'Righ
t', 'Front', 'Right Inverted', 'Front Inverted', 'Right', 'Front', 'Back']
```

Pmod Instructions in Visual Form



Testing

Testing was crucial when making sure the program could effectively solve the cube. The algorithm for solving the cube was the most time consuming part of the project, and in making sure it was as robust as possible was crucial to the project coming together. In order to test my code, I used Selenium-Webdriver and Node.js to send automated clicks to the website https://rubiks-cube-solver.com/ to compare its results to those of my program. For the first part of testing, I used the website to simulate rotations of the their cube and compared the resulting cube's orientation with the orientation of my cube. This verified that my program could rotate the cube correctly, and allowed me to move on to the crux of the assignment, the solving algorithm.

I only tested the algorithm after I had fully wrote it, and before that I was just working through solving one specific orientation of the cube. Once the algorithm was able to solve that one orientation of the cube, I wrote a unit test for 20 orientations of the cube through Node.js and Selenium-Webdriver once again to send clicks to the website and compare its results to my own. I tested 20 different cube configurations, and I had to go back and tweak almost every part of the algorithm because not all 20 cubes would make it past each of the checkpoints. Once I fixed those bugs, all 20 test cases passed, and I was able to rest easy knowing that the algorithm worked.

Goals Met

- The Webcam can read in the sides of the cube accurately.
- The algorithm works and solves every possible orientation of the cube.
- The Pmod reads each of the steps of the solution back to the user.
- The user can walk away with a solved cube.

Conclusion

The algorithm took probably 70% of the time, and was much more difficult than I had expected. I wrote the entire thing from scratch, and used the method for solving the cube based off of a variation of the layer by layer method found here: https://how-to-solve-a-rubix-cube.com/. Because getting the initial algorithm working took much of the development time, optimizing and reducing the number of steps needed to solve the cube was not able to be completed. Solving the cube currently takes on average more than 10 minutes to solve, and if the user makes one mistake then they will have to start the program over again and take the pictures of the cube once more. However, this is a prototype and has a base functionality of being able to read a cube and give the user the solution. I learned a lot from this project, and am happy I have a bit of Python under my belt now.