Rubik's Cube Solver Bot

Presented By

Amritesh Singh

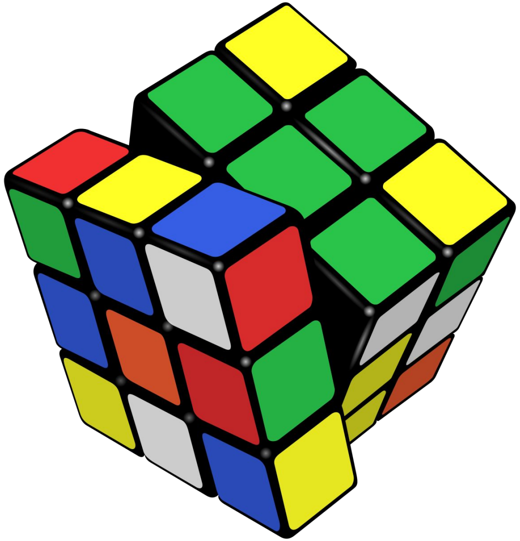
IRM2013020

Mentor: Dr. Pavan Chakravorty

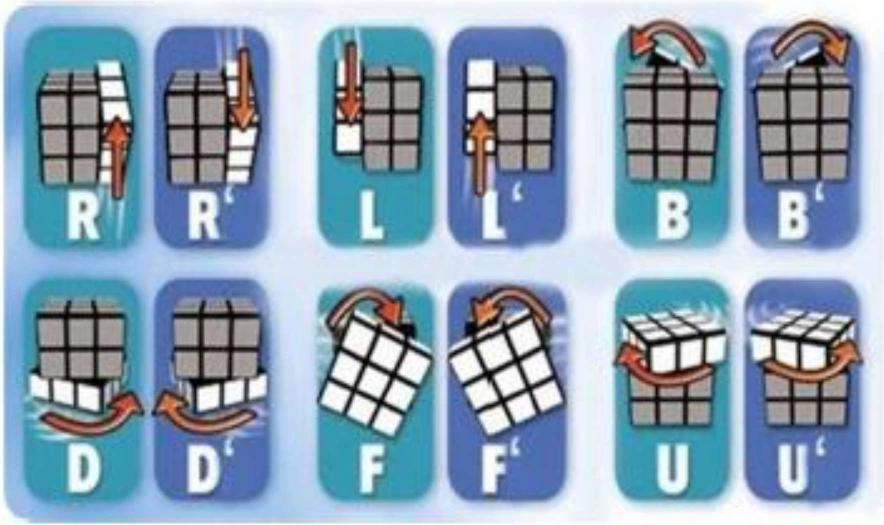
Thesis

- A Rubik’s cube consists of 6 faces – each having 9 square cells of one of 6 colours - yellow, white, blue, red, green, and orange, with each face capable of rotation independent of the others.
- Standard way of representing moves –F, F', R, R', B, B', L, L', U, U’, D, D’.
- **Problem definition:** To make mechanical design capable of solving cube using very less resources that can created at low cost.

Introduction



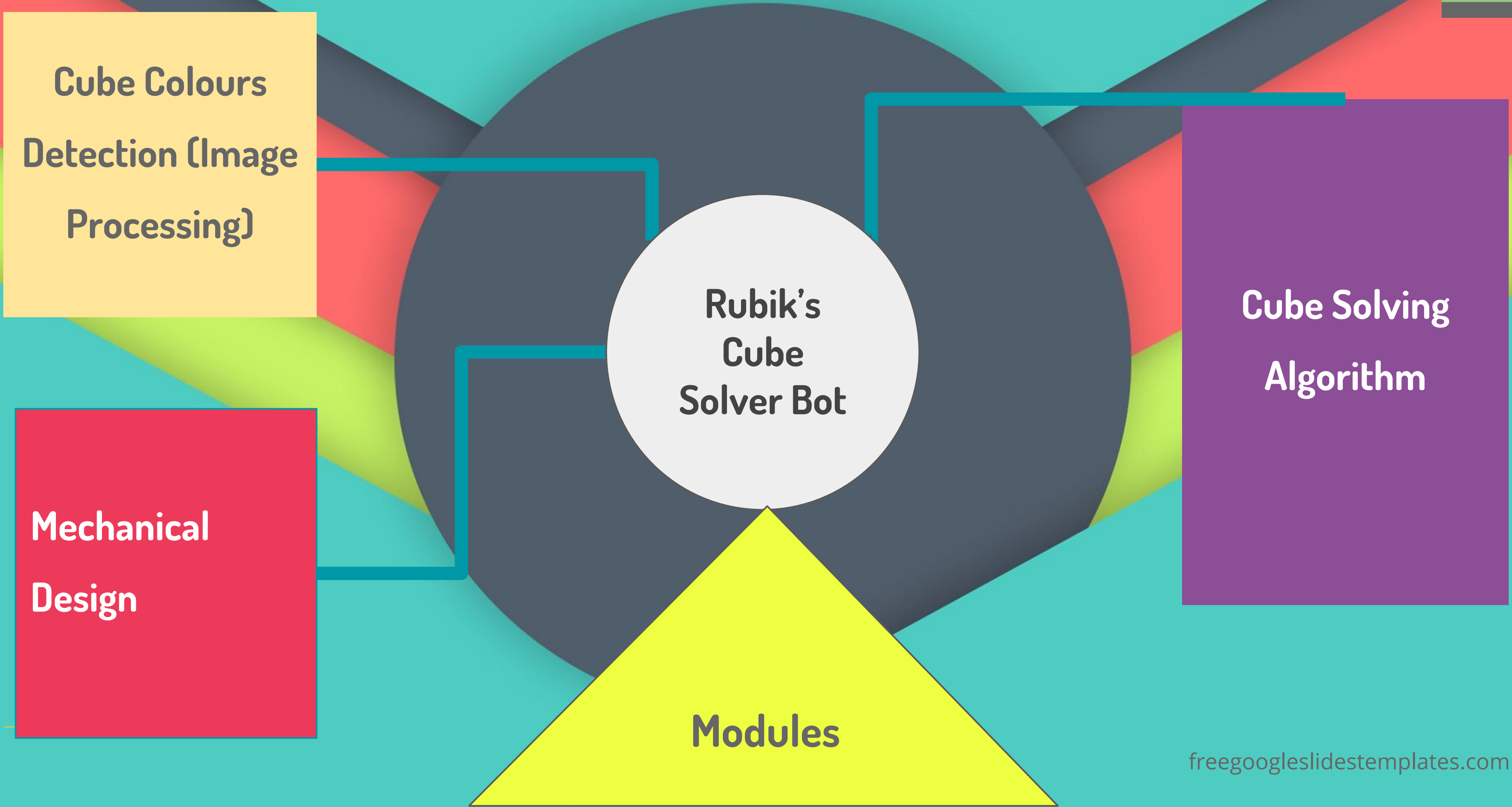
List of moves available in Rubik's cube



Introduction

- Code and mechanical design remain relatively decoupled.
- External webcam and image processing modules used to directly detect state of Rubik's cube.
- Sequence of moves generated to allow manual solving.
- Bot can be used for testing purposes of new algorithms.

Motivation



Rubik's Cube Solver Bot

Cube Colours Detection (Image Processing)

Cube Solving Algorithm

Mechanical Design

Modules

- OpenCV in Python 3 has been used.
- Each cube face is detected by screen capture using webcam.
- Steps involved:

❑ Canny edge detection is performed by converting RGB image to grayscale image.

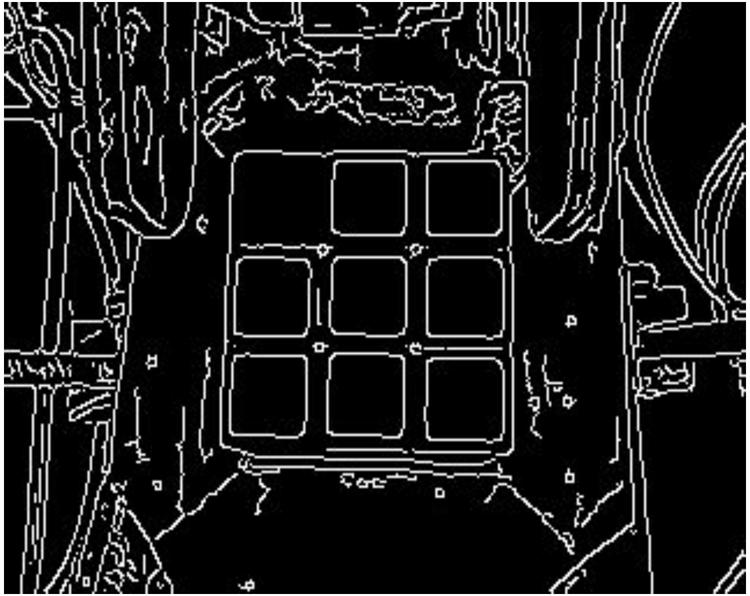
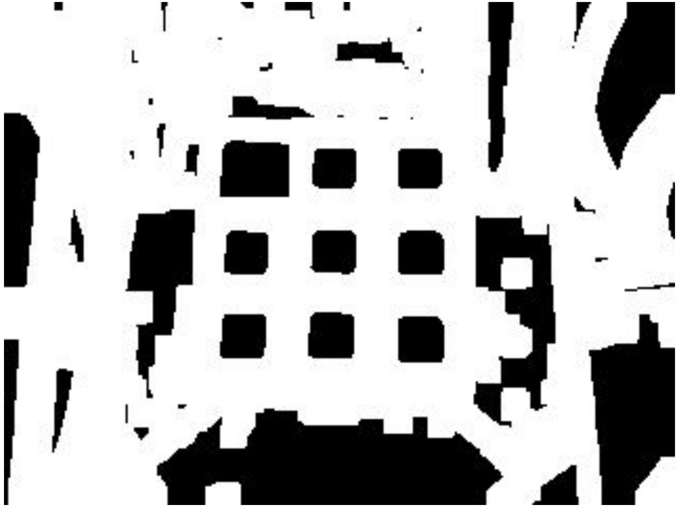


Image Processing

- ❑ Dilation is done to thicken the boundaries.
- ❑ Contours are detected.
- ❑ Contour approximation is done to generate the 9 squares on each face.
- ❑ Colour of each cell is separately identified by taking Numpy mean of RGB values in specific regions of each cell and checking if it lies in the range of that colour.



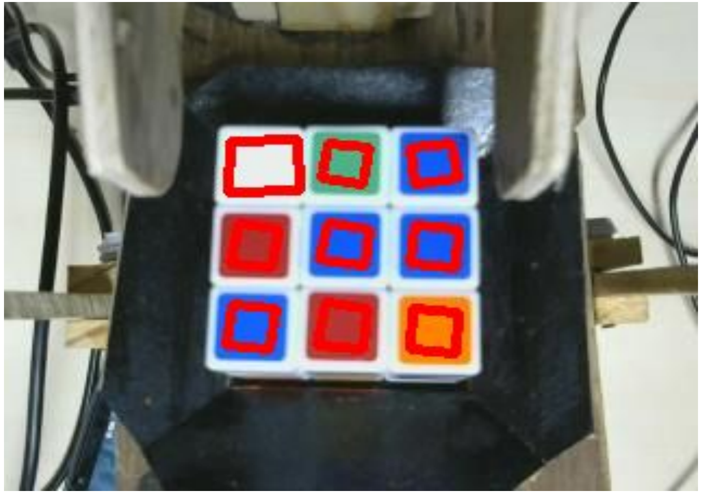


Image Processing

Solving Algorithm

- Layer by layer method used.
- Involves fives steps:

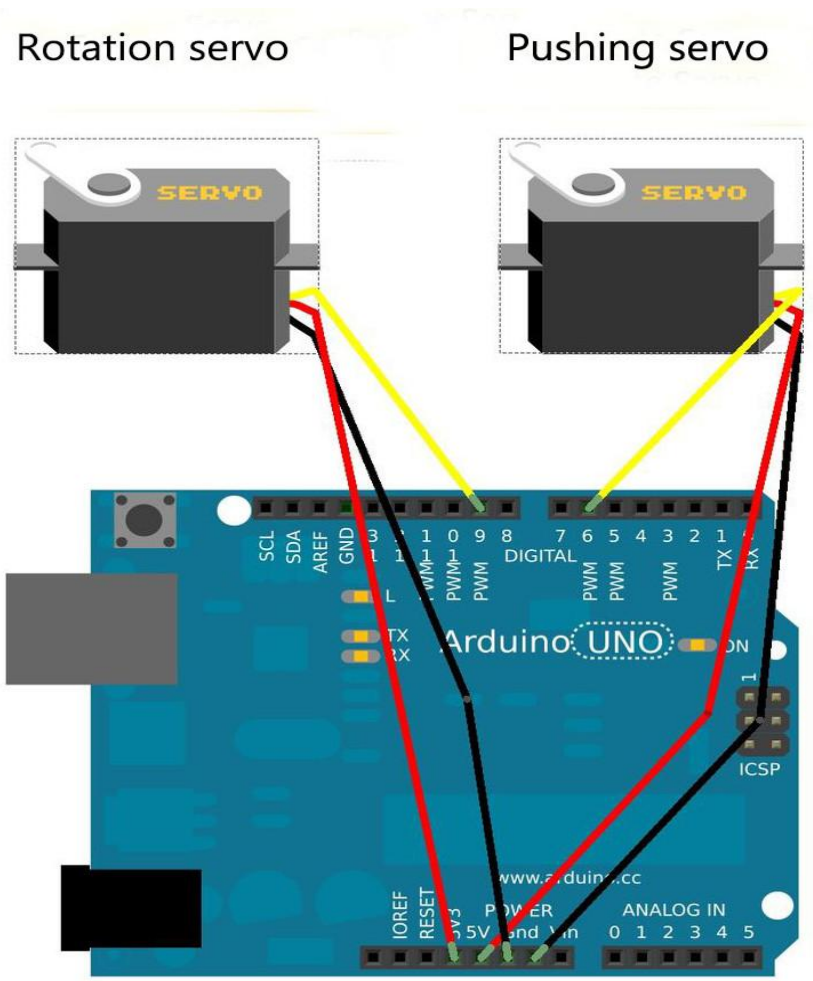
❑ Solving the cross

❑ Solving the first layer

❑ Solving the second layer

❑ Orienting the last layer

❑ Permuting the last layer

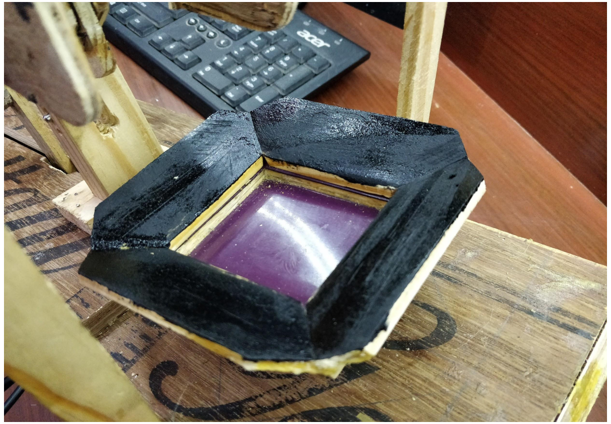


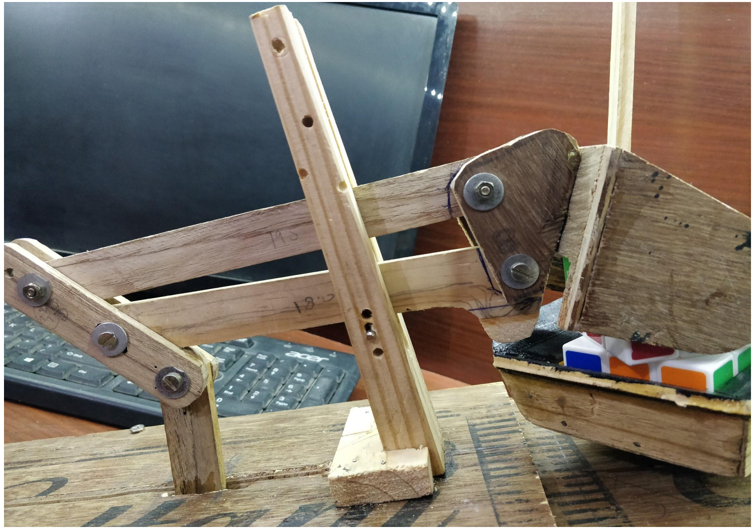
Arduino Connections

- Two servo motors used.
- Two basic mechanical components controlled by the servo motors:

❑ **Arm**: To push or hold cube.

❑ **Platform:** To rotate or hold cube.





Mechanical Design

- The arm and platform components perform following operations on the cube’s faces:

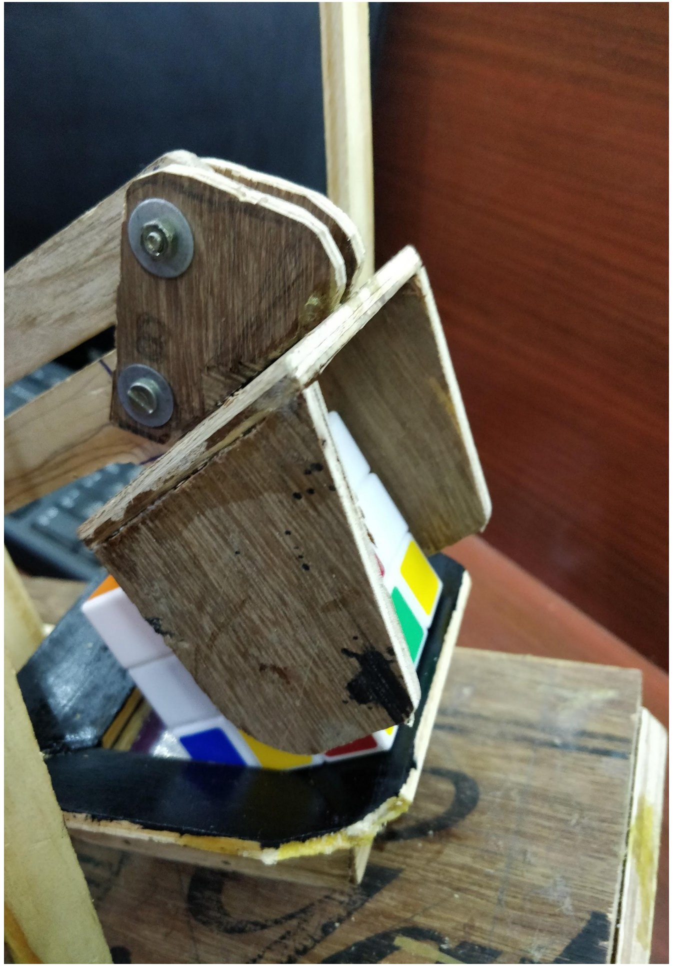
❑ Push

❑ Hold

❑ Release

❑ Rotate

Mechanical Design



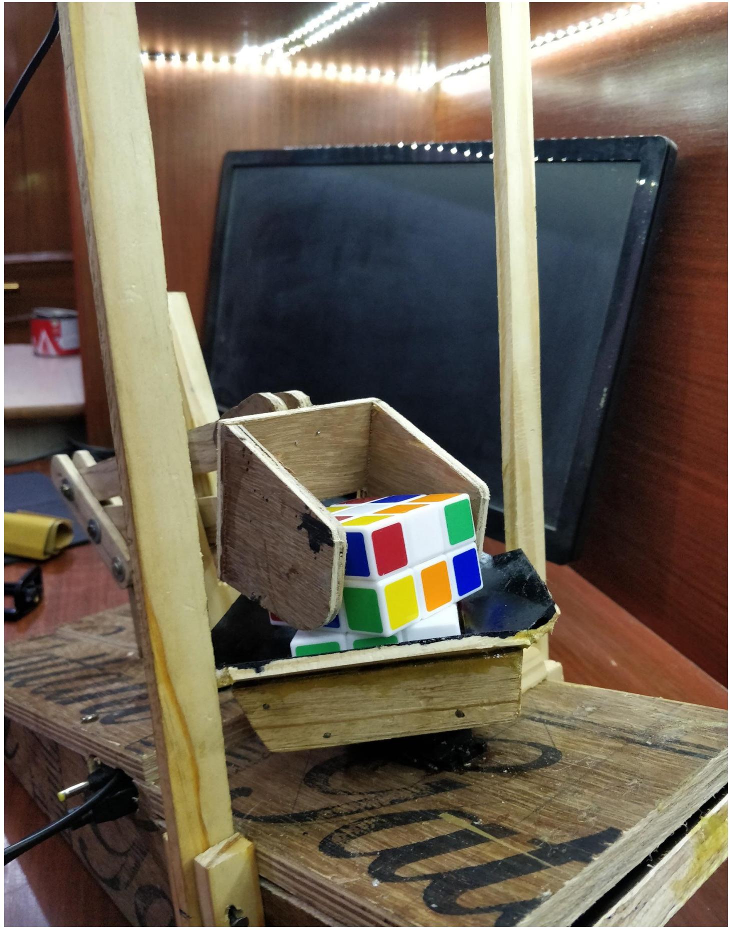
- Three combination operations possible:

❑ Push

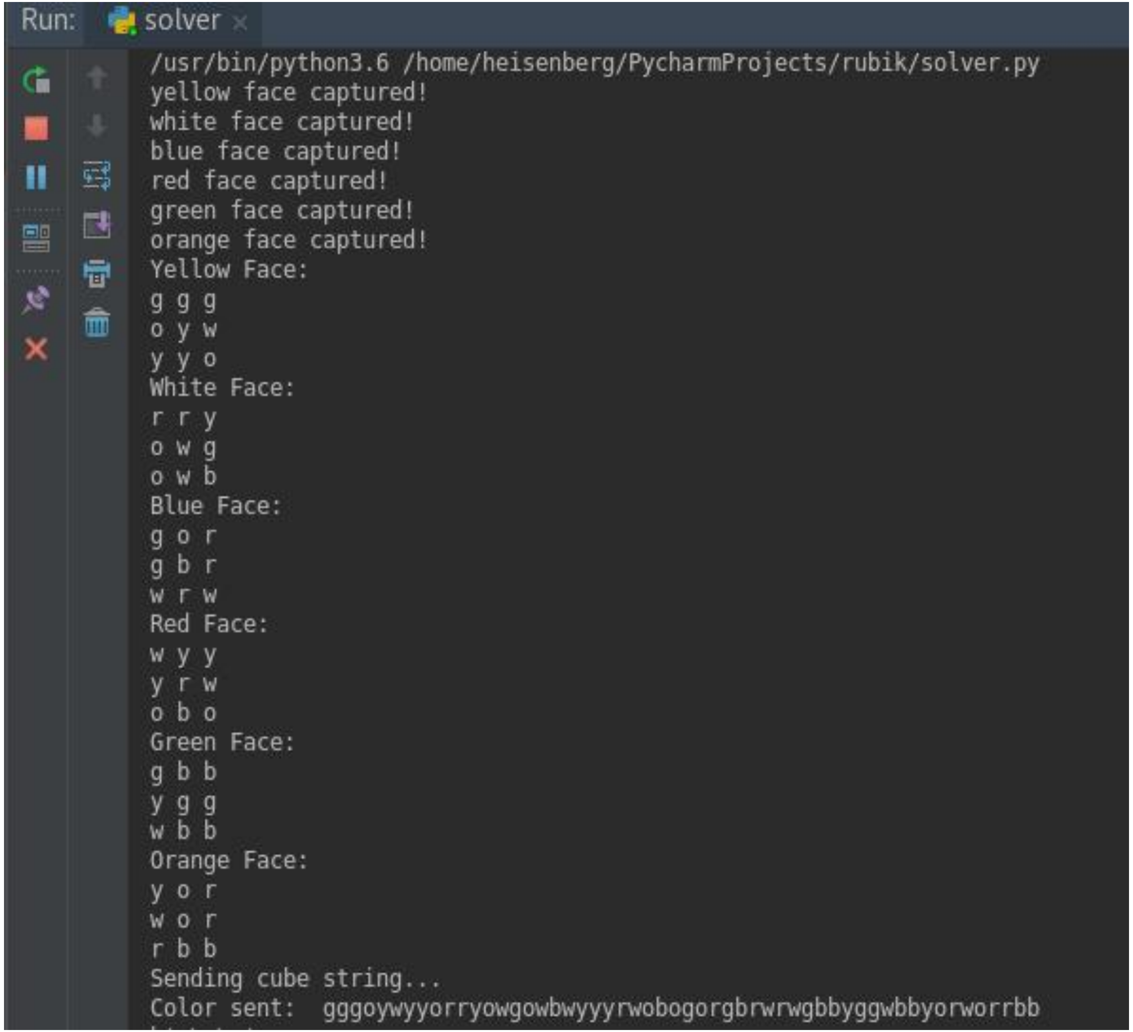
❑ Hold and Rotate

❑ Release and Rotate

Mechanical Design



- All cube face colours accurately detected through webcam and image processing.
- Raw string sent to Arduino.

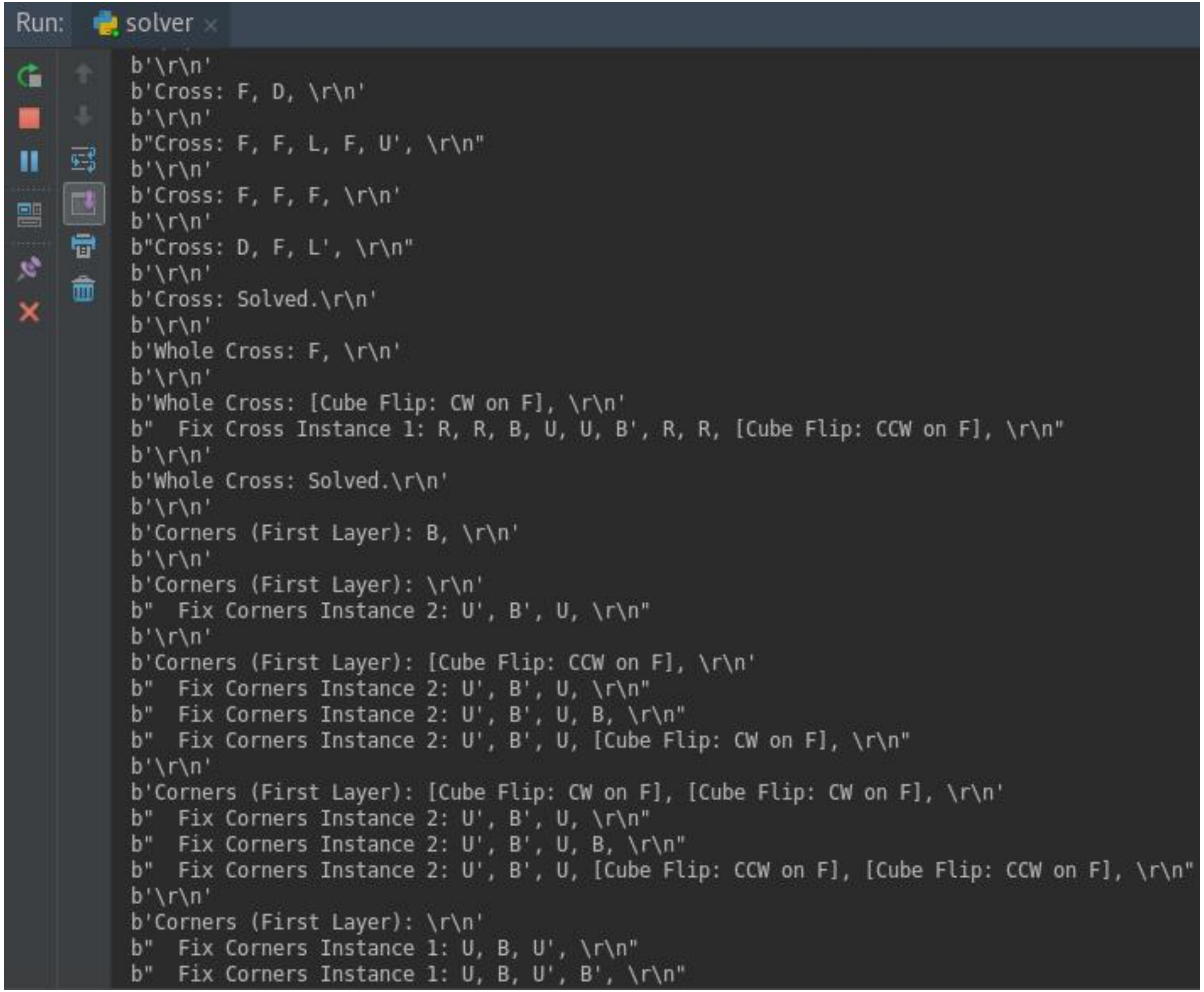


```
/usr/bin/python3.6 /home/heisenberg/PycharmProjects/rubik/solver.py
yellow face captured!
white face captured!
blue face captured!
red face captured!
green face captured!
orange face captured!
Yellow Face:
g g g
o y w
y y o
White Face:
r r y
o w g
o w b
Blue Face:
g o r
g b r
w r w
Red Face:
w y y
y r w
o b o
Green Face:
g b b
y g g
w b b
Orange Face:
y o r
w o r
r b b
Sending cube string...
Color sent:  gggoywyyorryowgowbwyyyrwobogorgbrwrwgbbyggwbbyorworrbb
```

Results

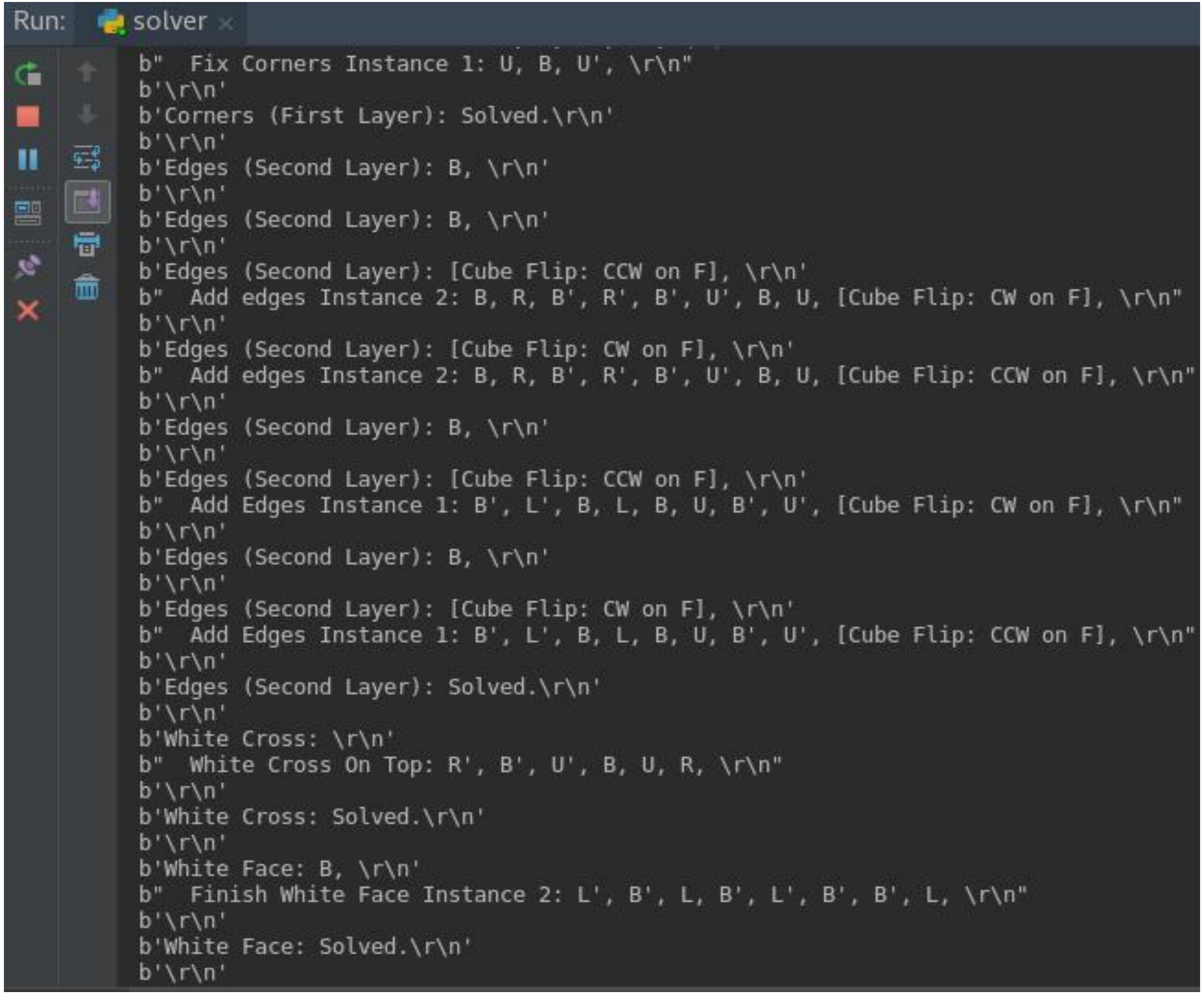
Results

- Sequence of moves is generated.



```
b'\r\n'
b'Cross: F, D, \r\n'
b'\r\n'
b"Cross: F, F, L, F, U', \r\n"
b'\r\n'
b'Cross: F, F, F, \r\n'
b'\r\n'
b"Cross: D, F, L', \r\n"
b'\r\n'
b'Cross: Solved.\r\n'
b'\r\n'
b'Whole Cross: F, \r\n'
b'\r\n'
b'Whole Cross: [Cube Flip: CW on F], \r\n'
b"  Fix Cross Instance 1: R, R, B, U, U, B', R, R, [Cube Flip: CCW on F], \r\n"
b'\r\n'
b'Whole Cross: Solved.\r\n'
b'\r\n'
b'Corners (First Layer): B, \r\n'
b'\r\n'
b'Corners (First Layer): \r\n'
b"  Fix Corners Instance 2: U', B', U, \r\n"
b'\r\n'
b'Corners (First Layer): [Cube Flip: CCW on F], \r\n'
b"  Fix Corners Instance 2: U', B', U, \r\n"
b"  Fix Corners Instance 2: U', B', U, B, \r\n"
b"  Fix Corners Instance 2: U', B', U, [Cube Flip: CW on F], \r\n"
b'\r\n'
b'Corners (First Layer): [Cube Flip: CW on F], [Cube Flip: CW on F], \r\n'
b"  Fix Corners Instance 2: U', B', U, \r\n"
b"  Fix Corners Instance 2: U', B', U, B, \r\n"
b"  Fix Corners Instance 2: U', B', U, [Cube Flip: CCW on F], [Cube Flip: CCW on F], \r\n"
b'\r\n'
b'Corners (First Layer): \r\n'
b"  Fix Corners Instance 1: U, B, U', \r\n"
b"  Fix Corners Instance 1: U, B, U', B', \r\n"
```

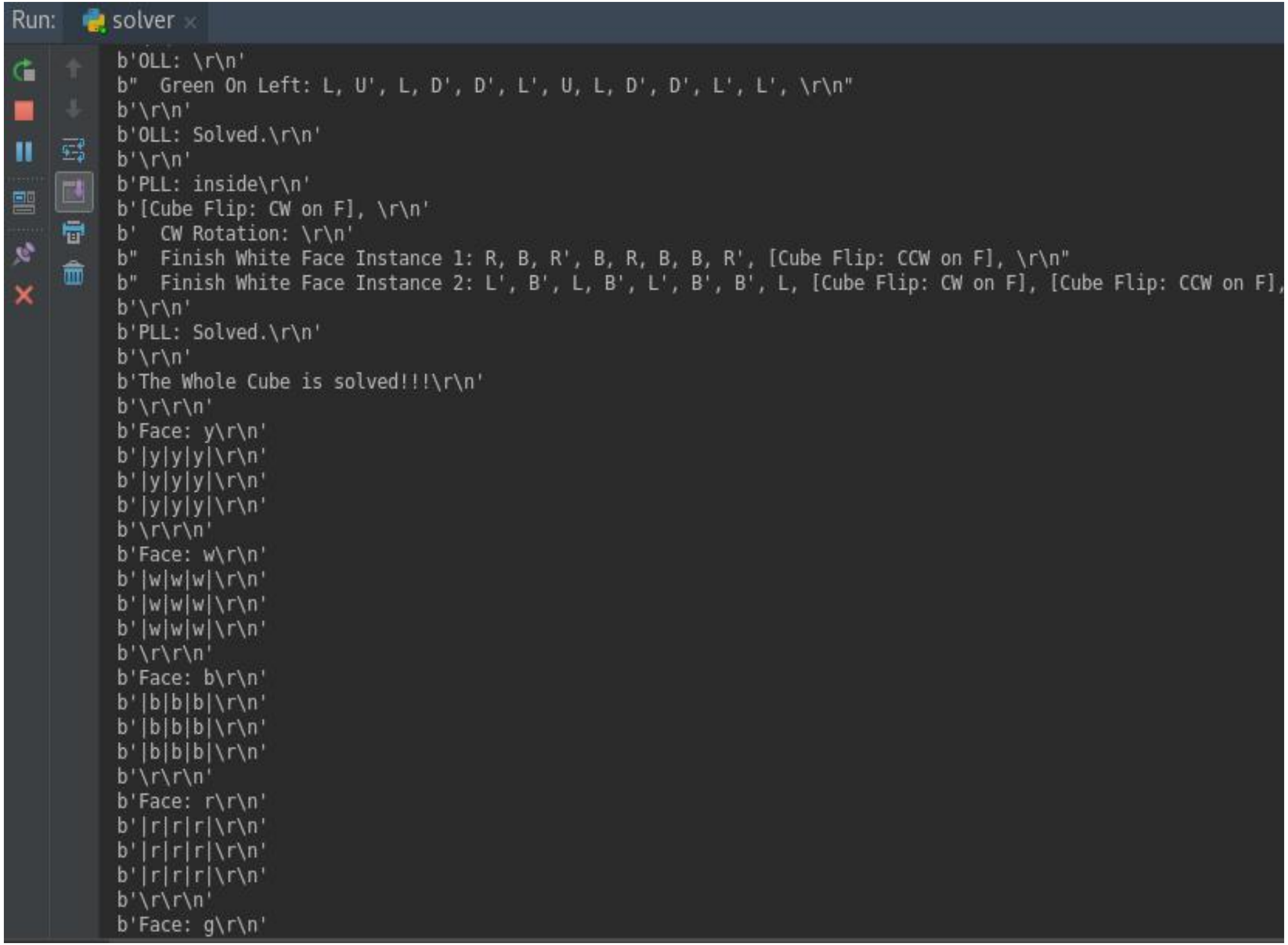
- Simultaneously, the mechanical components solve the cube using the 3 operations.



```
b"  Fix Corners Instance 1: U, B, U', \r\n"
b'\r\n'
b'Corners (First Layer): Solved.\r\n'
b'\r\n'
b'Edges (Second Layer): B, \r\n'
b'\r\n'
b'Edges (Second Layer): B, \r\n'
b'\r\n'
b'Edges (Second Layer): [Cube Flip: CCW on F], \r\n'
b"  Add edges Instance 2: B, R, B', R', B', U', B, U, [Cube Flip: CW on F], \r\n"
b'\r\n'
b'Edges (Second Layer): [Cube Flip: CW on F], \r\n'
b"  Add edges Instance 2: B, R, B', R', B', U', B, U, [Cube Flip: CCW on F], \r\n"
b'\r\n'
b'Edges (Second Layer): B, \r\n'
b'\r\n'
b'Edges (Second Layer): [Cube Flip: CCW on F], \r\n'
b"  Add Edges Instance 1: B', L', B, L, B, U, B', U', [Cube Flip: CW on F], \r\n"
b'\r\n'
b'Edges (Second Layer): B, \r\n'
b'\r\n'
b'Edges (Second Layer): [Cube Flip: CW on F], \r\n'
b"  Add Edges Instance 1: B', L', B, L, B, U, B', U', [Cube Flip: CCW on F], \r\n"
b'\r\n'
b'Edges (Second Layer): Solved.\r\n'
b'\r\n'
b'White Cross: \r\n'
b"  White Cross On Top: R', B', U', B, U, R, \r\n"
b'\r\n'
b'White Cross: Solved.\r\n'
b'\r\n'
b'White Face: B, \r\n'
b"  Finish White Face Instance 2: L', B', L, B', L', B', B', L, \r\n"
b'\r\n'
b'White Face: Solved.\r\n'
b'\r\n'
```

Results

- Ultimately, the bot successfully solves the cube.



```
b'OLL: \r\n'
b"  Green On Left: L, U', L, D', D', L', U, L, D', D', L', L', \r\n"
b'\r\n'
b'OLL: Solved.\r\n'
b'\r\n'
b'PLL: inside\r\n'
b'[Cube Flip: CW on F], \r\n'
b'  CW Rotation: \r\n'
b"  Finish White Face Instance 1: R, B, R', B, R, B, B, R', [Cube Flip: CCW on F], \r\n"
b"  Finish White Face Instance 2: L', B', L, B', L', B', B', L, [Cube Flip: CW on F], [Cube Flip: CCW on F],
b'\r\n'
b'PLL: Solved.\r\n'
b'\r\n'
b'The Whole Cube is solved!!!\r\n'
b'\r\r\n'
b'Face: y\r\n'
b'|y|y|y|\r\n'
b'|y|y|y|\r\n'
b'|y|y|y|\r\n'
b'\r\r\n'
b'Face: w\r\n'
b'|w|w|w|\r\n'
b'|w|w|w|\r\n'
b'|w|w|w|\r\n'
b'\r\r\n'
b'Face: b\r\n'
b'|b|b|b|\r\n'
b'|b|b|b|\r\n'
b'|b|b|b|\r\n'
b'\r\r\n'
b'Face: r\r\n'
b'|r|r|r|\r\n'
b'|r|r|r|\r\n'
b'|r|r|r|\r\n'
b'\r\r\n'
b'Face: g\r\n'
```

Results

THANK

YOU