IOWA STATE UNIVERSITY

Electrical and Computer Engineering Senior Design

Self-Solving Rubik's Cube

Taylor Burton (Systems Lead)
Jacob Campen (Hardware Lead)
Casey Cierzan (Materials Lead)
Joe Crowley (Testing Lead)
Annie (Yung-Hsueh) Lee (Algorithms Lead)
Keegan Levings-Curry (Administrative Lead)
Luke Schoeberle (Software Design Lead)

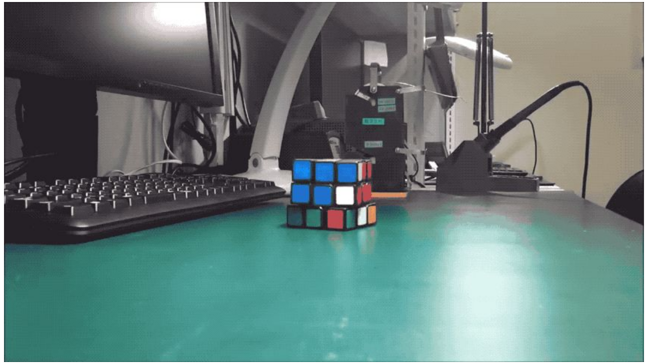
Client/Advisor: Dr. Zambreno

sdmay20-29

http://sdmay20-29.sd.ece.iastate.edu

Problem Statement and Project Vision

- A self-contained, self-solving Rubik's cube
  - Can be scrambled by hand
  - Solves itself with no intervention
- Use for recruitment at ISU
  - Displays the possibilities of our degree
  - Hands-on recruitment tool



Source: Takashi Kaburagi

Requirements

Functional Requirements

- Solved in 2 minutes or less
- The battery lasts for at least one full use case
- Does not rely on external devices (like cameras or robot arms)
- Starts in a solved state

Nonfunctional Requirements

- Resembles a standard Rubik’s cube on the exterior
- Easily turned by the user
- Lasts for at least 3 years
- Side length should be 11 cm
- The costs should not exceed $750

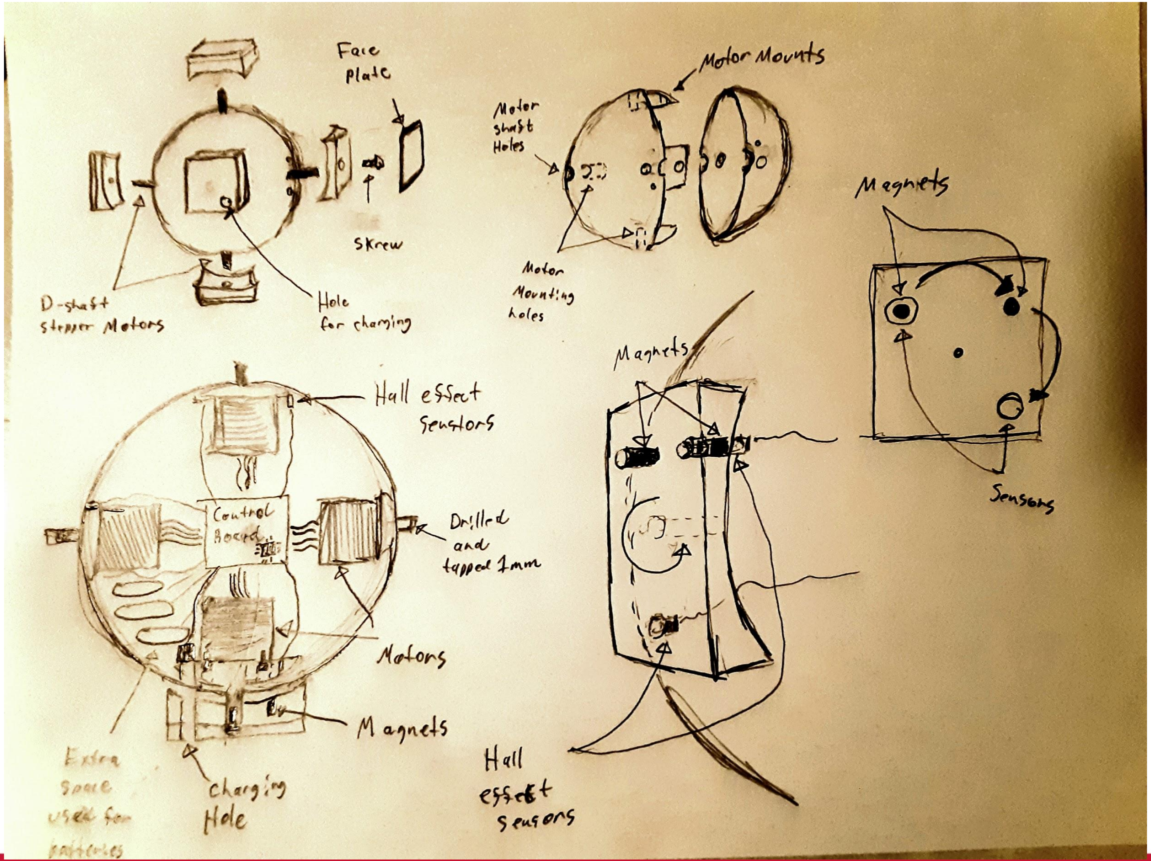
Engineering Standards and Design Practices

- Follow IEEE standards (hardware and software)
- Push early and often
- Document everything (e.g., schematics and meeting notes)
- Follow a tight budget ($390.30/$750)
- Ensure maintainability (at least 3 years)

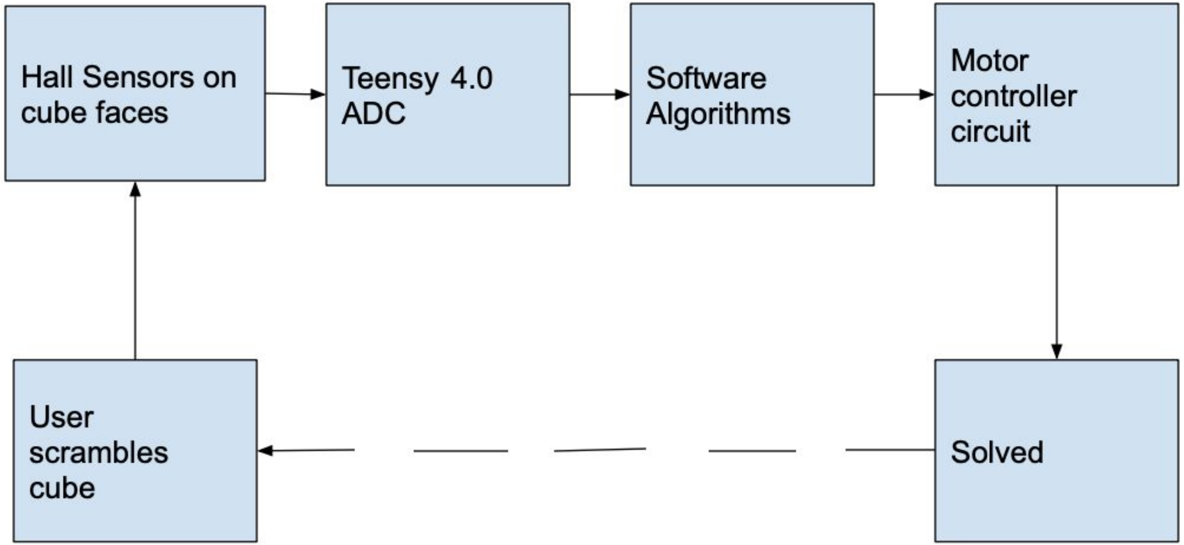
Final Deliverables due to COVID-19

- Materials for the physical prototype
- Completed solving algorithms
- Untested system code
- CAD models
- Documentation

Conceptual Sketch



System Block Diagram



Risks and Mitigation

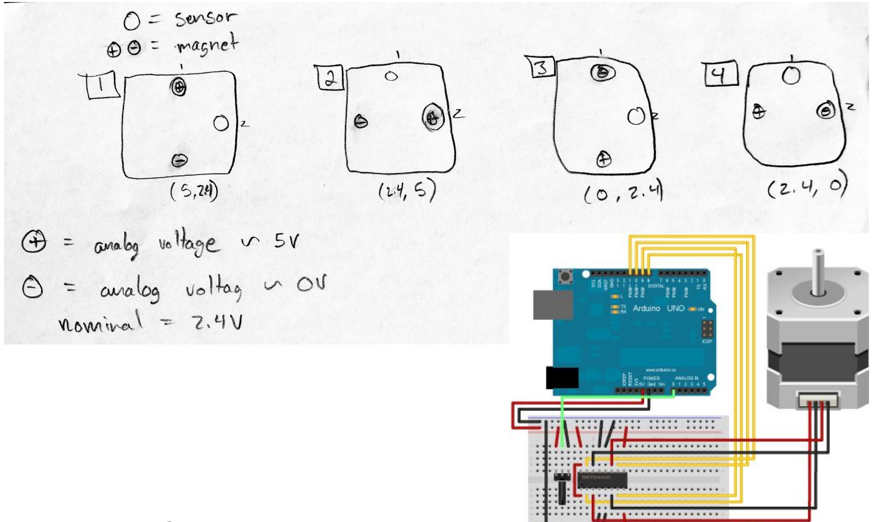
Risks	**Mitigations**
1. Size limitations	1. Agreed side length is 11 cm
2. Budget constraints	2. The budget is $750
3. High capacity batteries	3. Batteries are not high capacity
4. Safety during construction	4. Most of system is 3D-printed

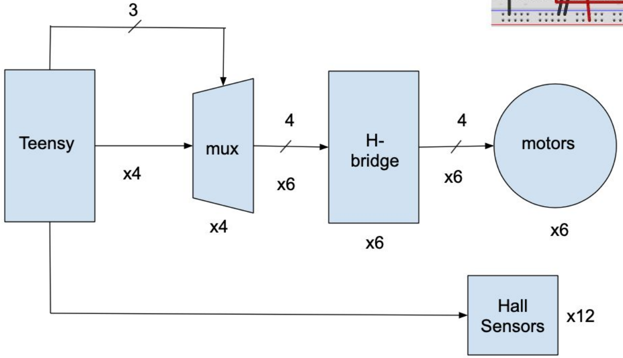
User Interface Description

- From a user's perspective, the interface is mostly the same as a normal cube
- Turned by rotating the outside faces as usual
- There are a few minor differences in our cube
    - Users may feel the motors' resistance when they scramble the cube
    - Users can charge the cube by using the port on the white face's center

Hardware Design

- Hall Effect sensors
- Mechanical considerations
  - Size of cube and internal space
  - Size of motors
  - Operating environment
    - Flat tabletop
- Stepper motors
  - Can be turned manually
- Teensy 4.0 microcontroller
- Batteries



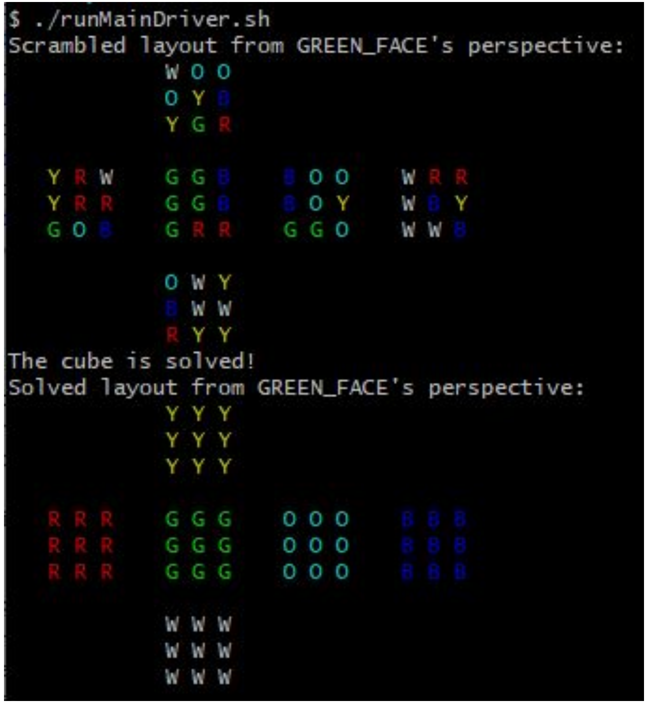


Software Design

- Embedded software on our Teensy microcontroller
- A mix of pure C code and Arduino code
- Consists of four main parts:
  - Rotation detection software
  - Rotation simulation algorithms
  - Solving algorithms
  - Motor control software

Solving Algorithm Overview

- Implements a layer-solving algorithm in C
- Solves the green face first due to our data structures
- Records the rotations in a linked list
- Consists of four main parts:
  - Utility functions
  - First-layer algorithms
  - Second-layer algorithms
  - Third-layer algorithms



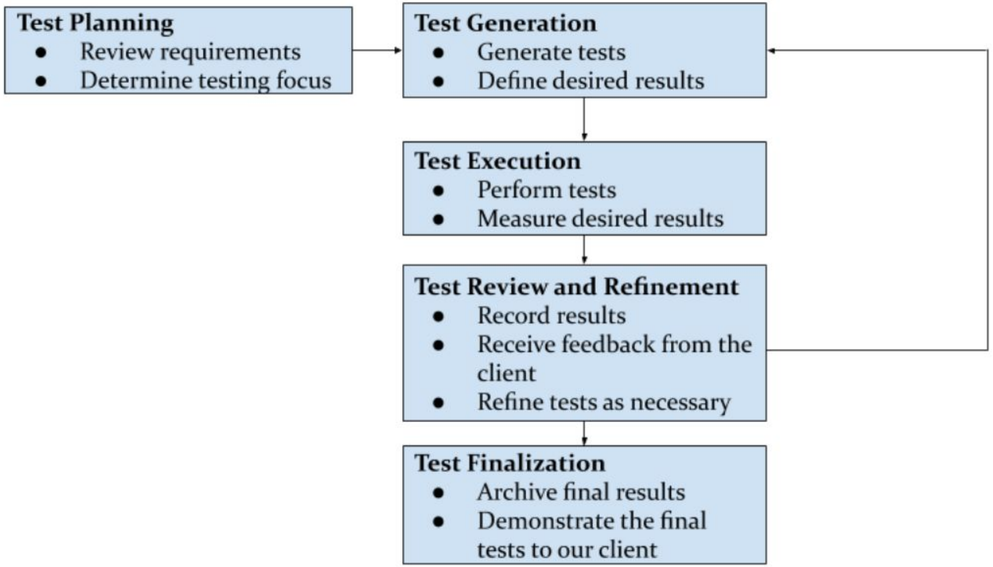
```
$ ./runMainDriver.sh
Scrambled layout from GREEN_FACE's perspective:
        W O O
        O Y B
        Y G R

  Y R W G G B B O O W R R
  Y R R G G B B O Y W B Y
  G O B G R R G G O W W B

        O W Y
        B W W
        R Y Y
The cube is solved!
Solved layout from GREEN_FACE's perspective:
        Y Y Y
        Y Y Y
        Y Y Y

  R R R G G G O O O B B B
  R R R G G G O O O B B B
  R R R G G G O O O B B B

        W W W
        W W W
        W W W
```

Future Improvements to the Solving Algorithms

- Consider reversing the input rotations as a possible solution
- Choose the best starting face for the current algorithm
- Reduce the number of rotations from 150 to 100 in the current algorithm
- Implement other efficient solving algorithms
- Minimize the cube's spatial movement during the algorithm

Testing Process

- Unit Testing
  - Ex: Motor control circuit
- Integration Testing
  - Ex: Motor integration
- System Testing
  - Ex: Holistic verification



Test Planning
- Review requirements
- Determine testing focus

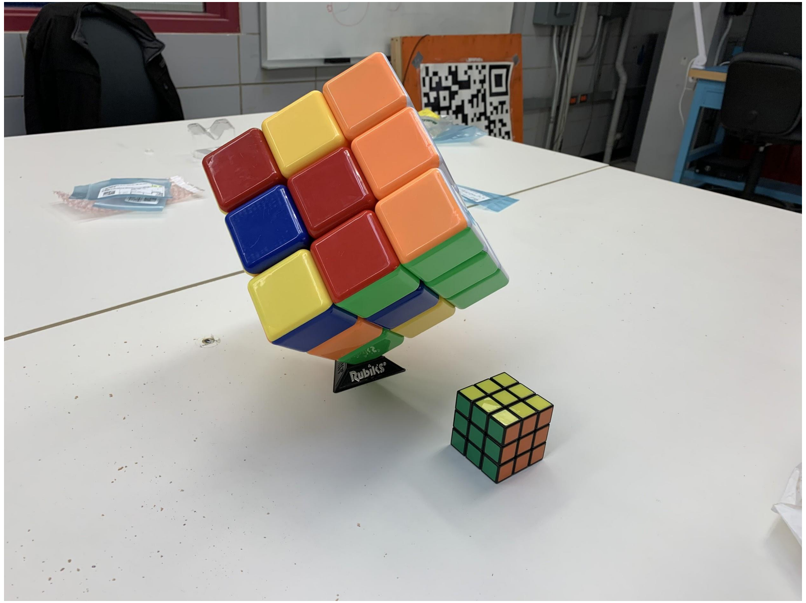
Test Generation
- Generate tests
- Define desired results

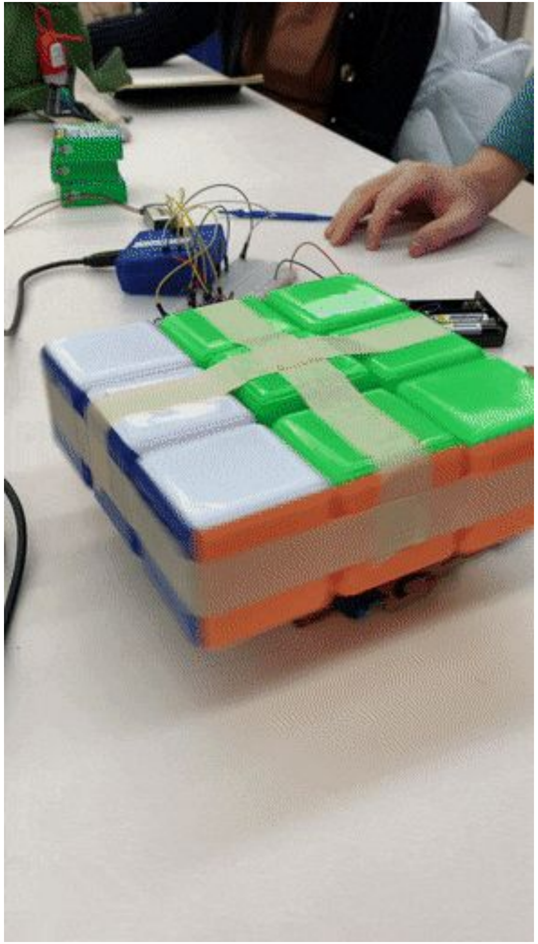
Test Execution
- Perform tests
- Measure desired results

Test Review and Refinement
- Record results
- Receive feedback from the client
- Refine tests as necessary

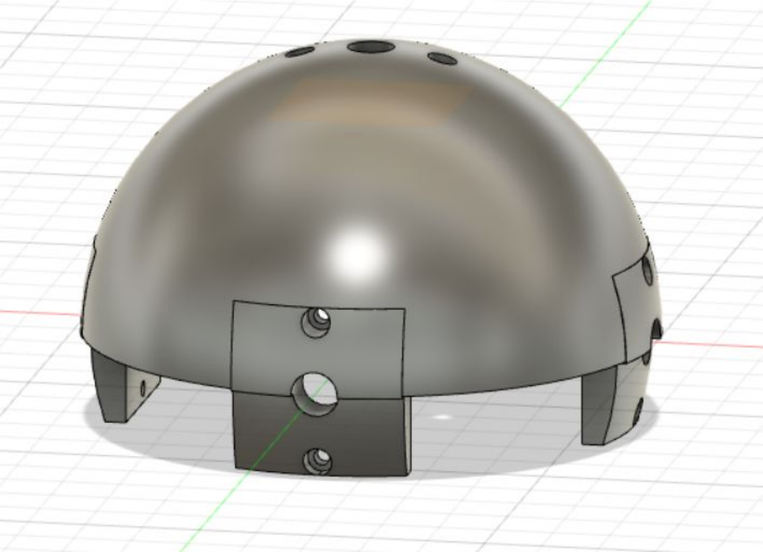
Test Finalization
- Archive final results
- Demonstrate the final tests to our client

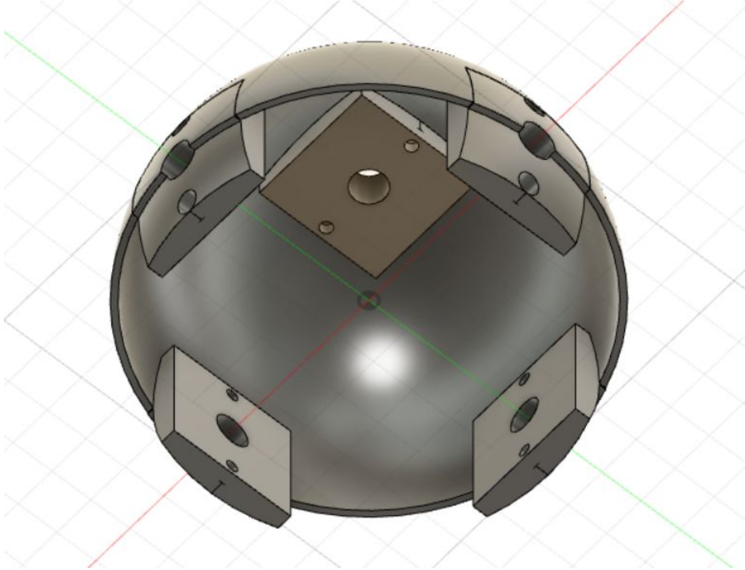
Prototyping



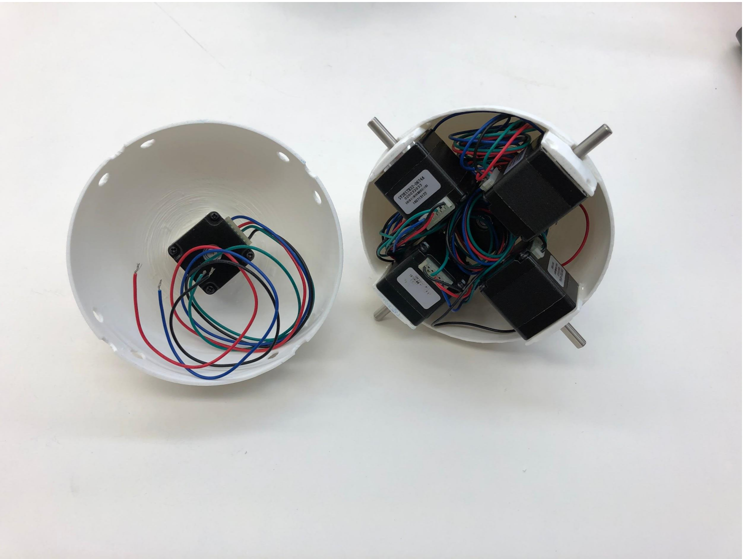


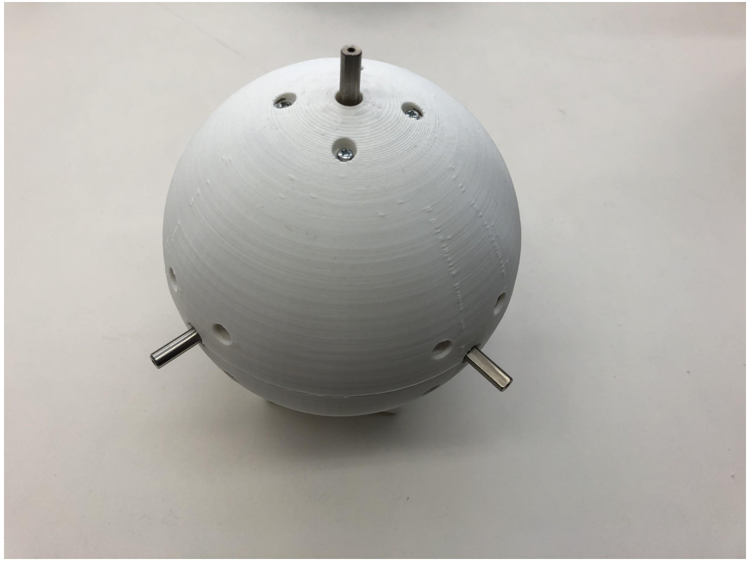
Prototyping



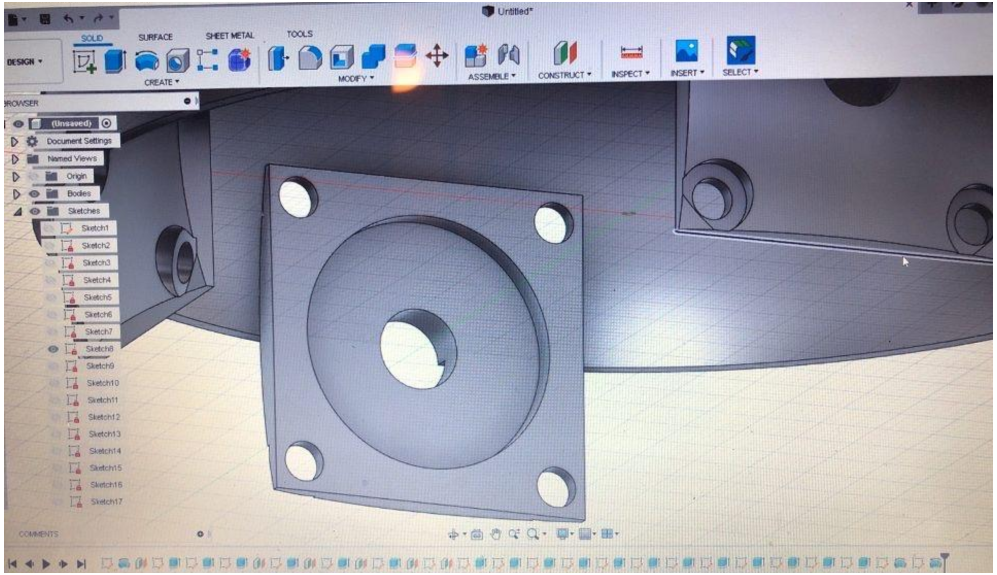


Prototyping

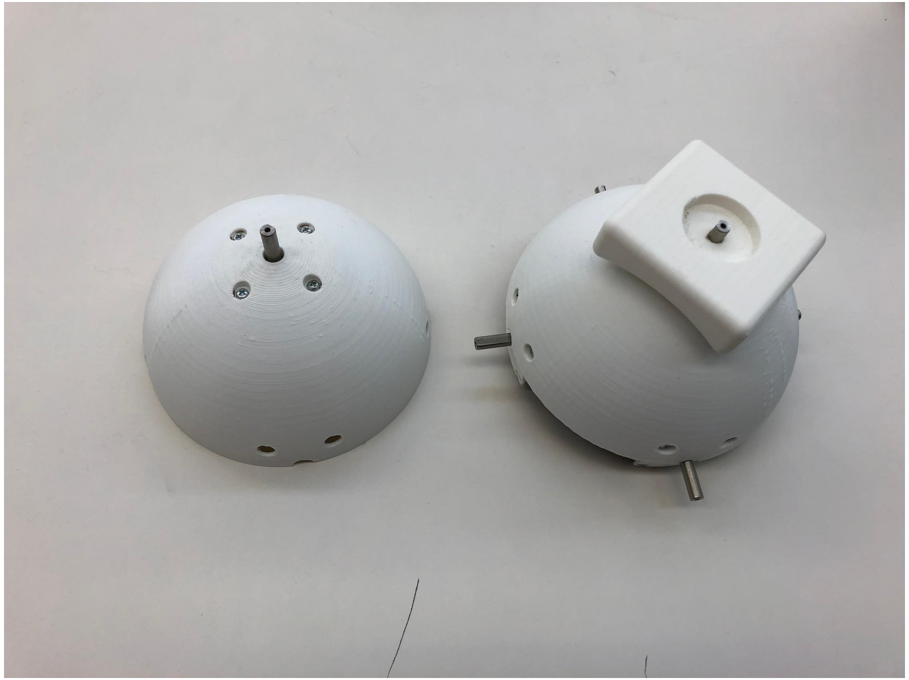




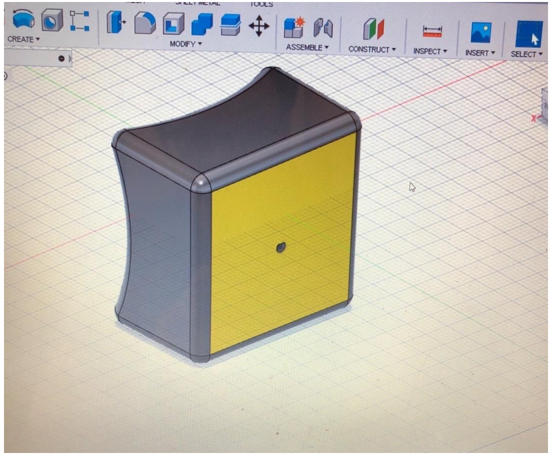
Prototyping

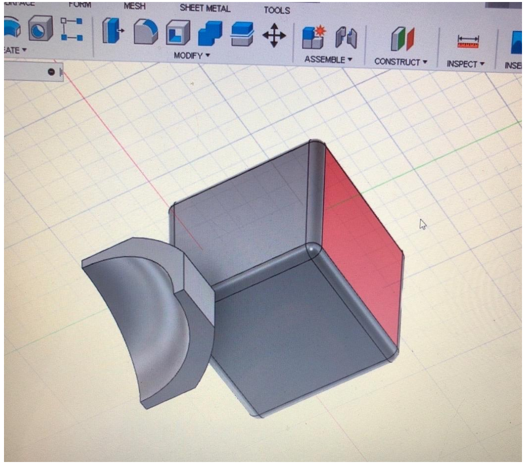


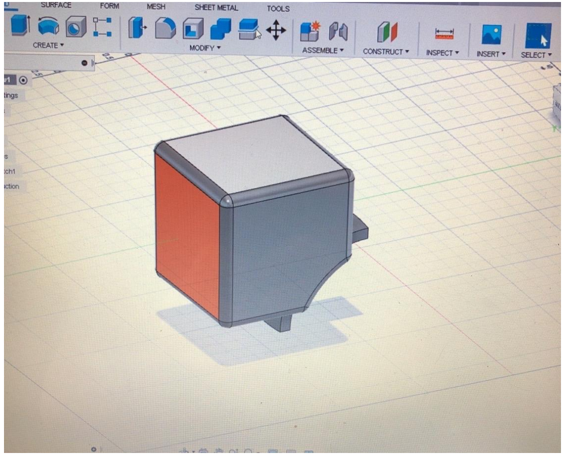
Prototyping



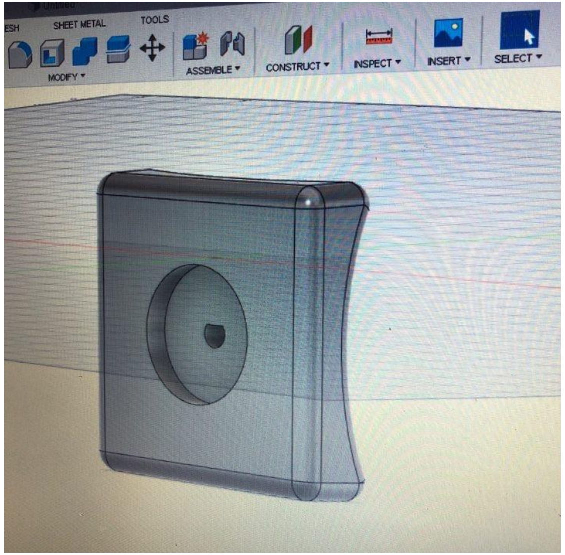
Prototyping

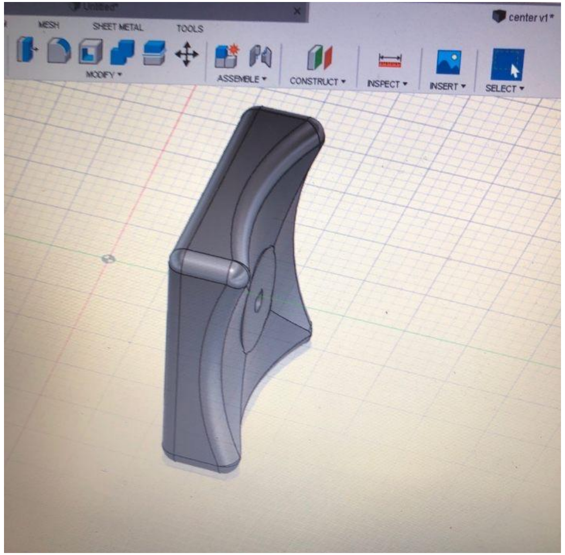




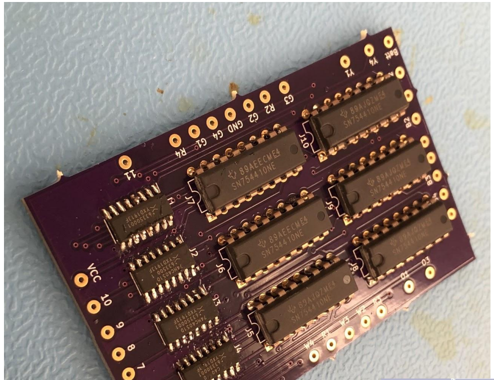


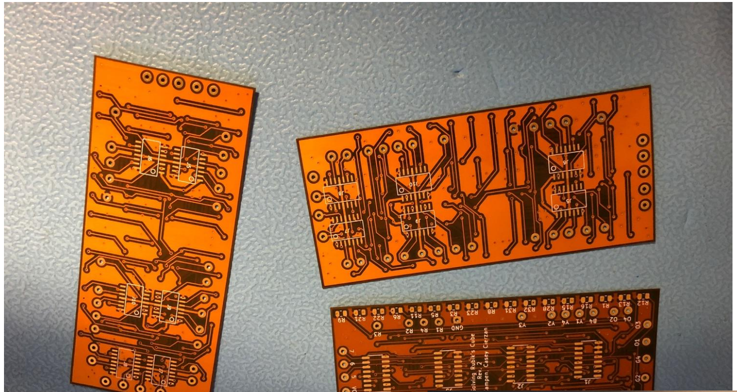
Prototyping

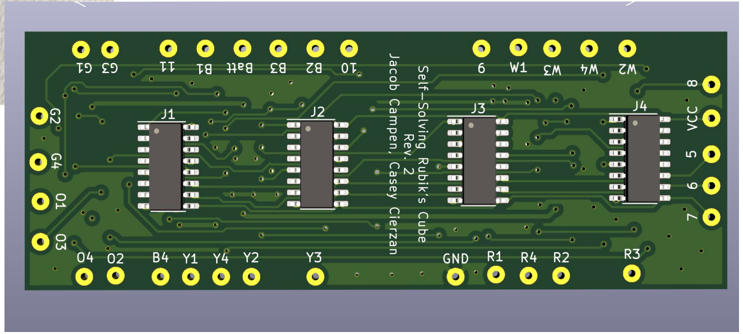


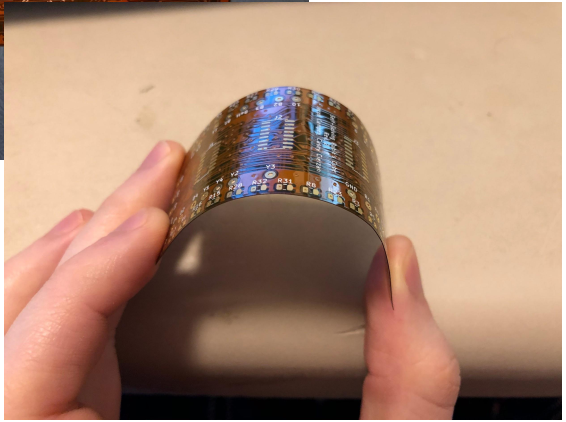


Prototyping









Contributions

- Rotation simulation algorithms and solving algorithms - Luke and Annie
- Rotation detection code and motor control code - Joe
- Mechanical design and construction - Taylor
- PCB design and hardware selection - Jacob
- Hardware selection and schematic drafting - Casey
- Battery selection and general logistics - Keegan

Future Status

- Obtained all the parts for the prototype
- CAD models are ready for 3D-printing
- Performed unit testing on most of the components
- Need to verify the PCB, the system code, and the full mechanical system
- The basic solving algorithms are fully completed
- Overall, future teams should be able to complete this project

Thank you!