Self-Solving Rubik's Cube

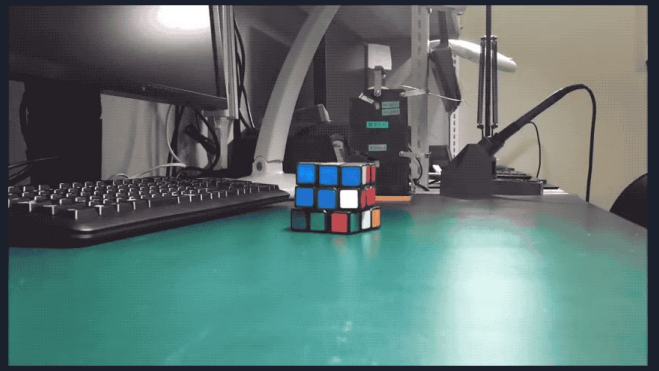
sdmay20-29

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

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

Similar Product Survey

- External Solvers
- Internal Solvers

Source: Jay Flatland

Requirements

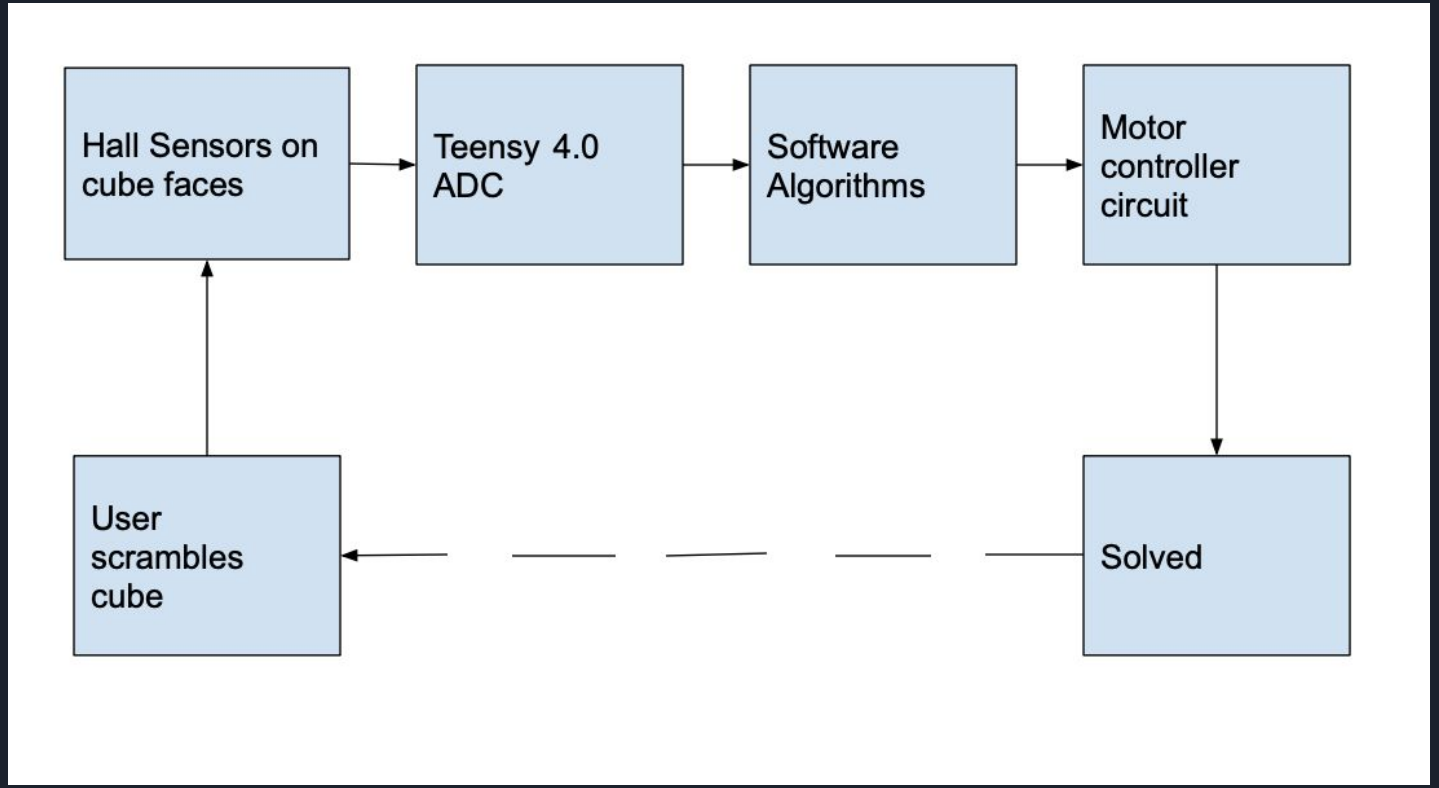
Functional Requirements

- Must be solved in 2 minutes or less
- Battery should last for at least 1 solve
- System should not rely on external devices (like cameras or robot arms)
- Cube starts in a solved state

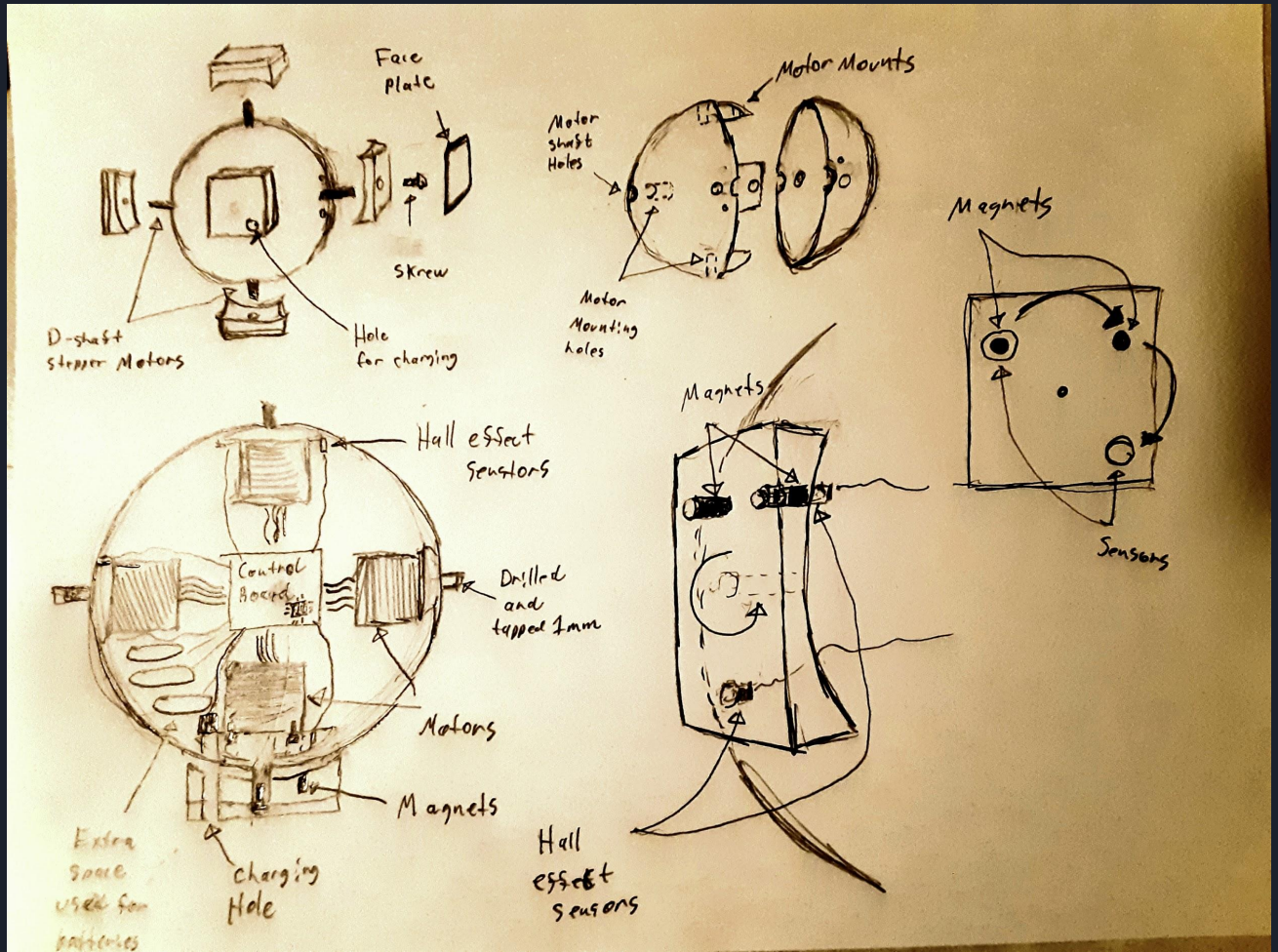
Nonfunctional Requirements

- System should be look like a standard Rubik’s Cube
- Cube faces must be easily turnable by the user
- Cube should last at least 3 years
- Cube side length should be between 5.7 cm and 18 cm
- Budget should not exceed $750

System Block Diagram



Conceptual Sketch

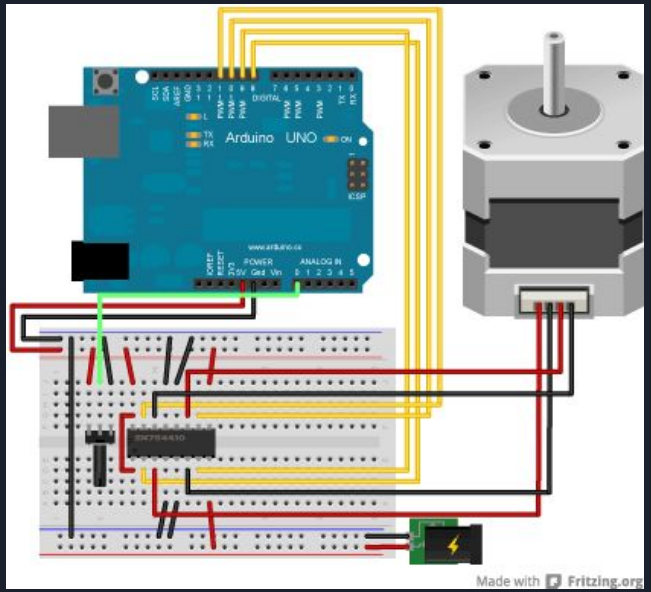


User Interface Description

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

Hardware Design

- Hall Effect Sensors for rotation detection
- Mechanical Considerations
  - Size of cube and internal space
  - Size of motors
  - Operating Environment
    - Flat tabletop
- Stepper Motors
  - Can be turned manually to scramble
- Teensy 4.0 Microcontroller
- Battery



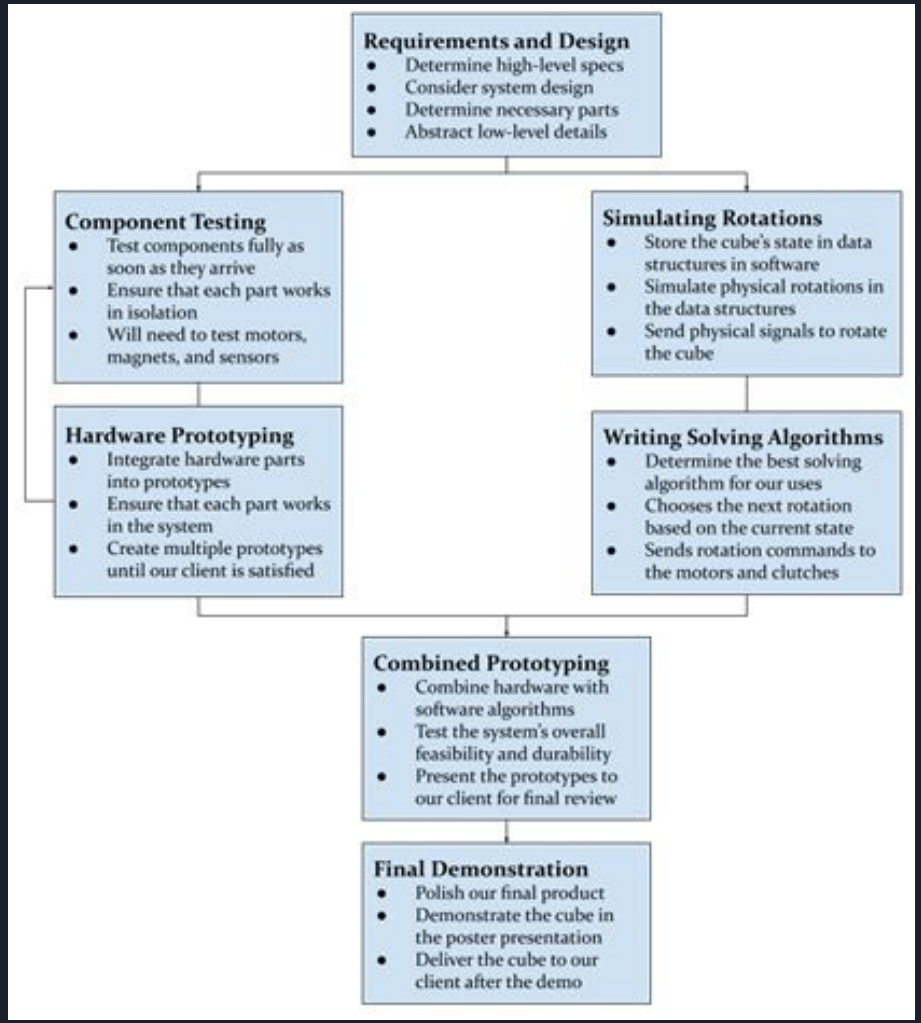
Software Design

- 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

Resource Requirements

- 1 18x18x18 Rubik’s cube
- 6 stepper motors
- 12 Hall Effect Sensors
- 18 magnets
- 1 Teensy 4.0 microcontroller
- 2 3D-printed hemispheres
- Rechargeable batteries
- 1 battery charging port

Work Breakdown



Risks

- Size limitations
- Budget constraints
- High capacity batteries
- Safety during construction

Project Plan - Metrics to quantify progress

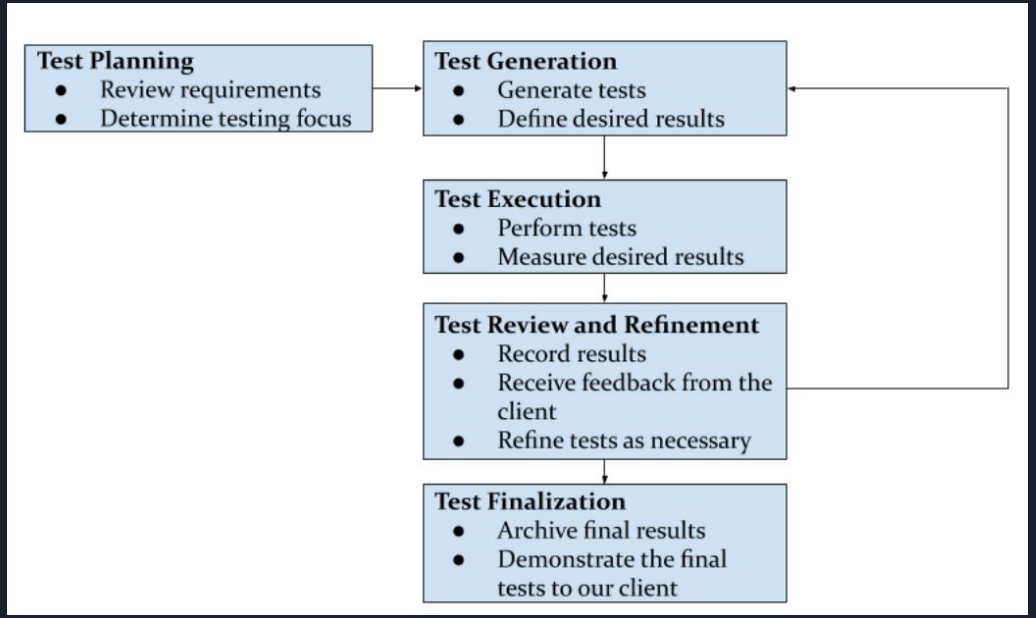
- Rotation Accuracy: error from 90° or 180°
- Solving Accuracy: percentage of correct cubes when the algorithm ends
- Solving Space: deviation from the table area
- Solving Time: acceptable only if less than two minutes
- Battery Time: acceptable only if it lasts for the entire solving process

Project Schedule - Milestones

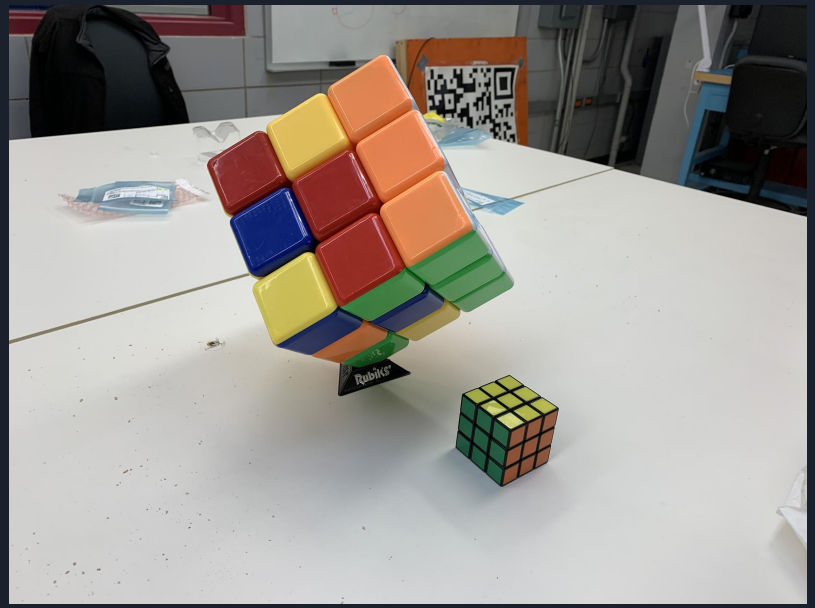
- 1/26: Hardware Prototype Completion
- 2/16: Full Prototype Completion
- 3/8: Full CAD Design Completion
- 3/29: Final Cube Completion
- 5/1: Final Cube Submission

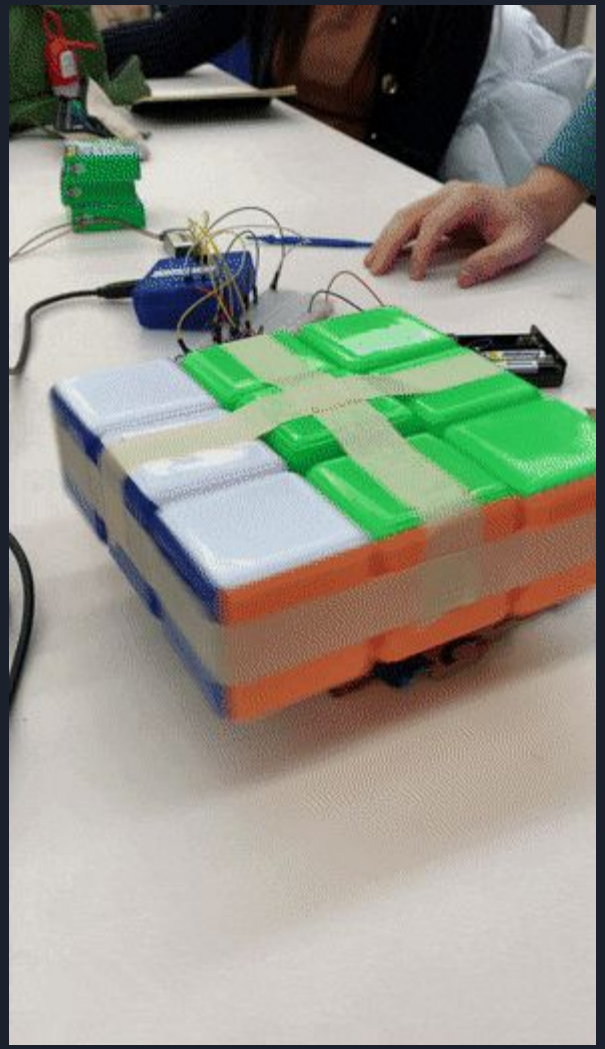
Testing Process

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

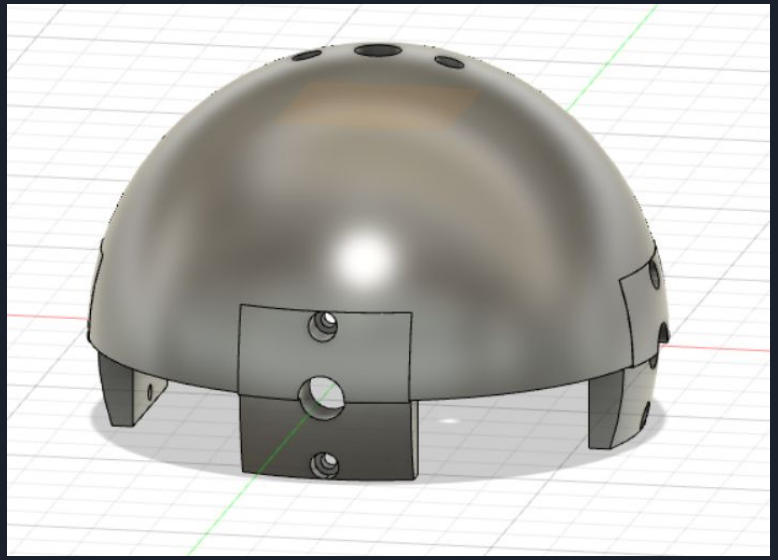


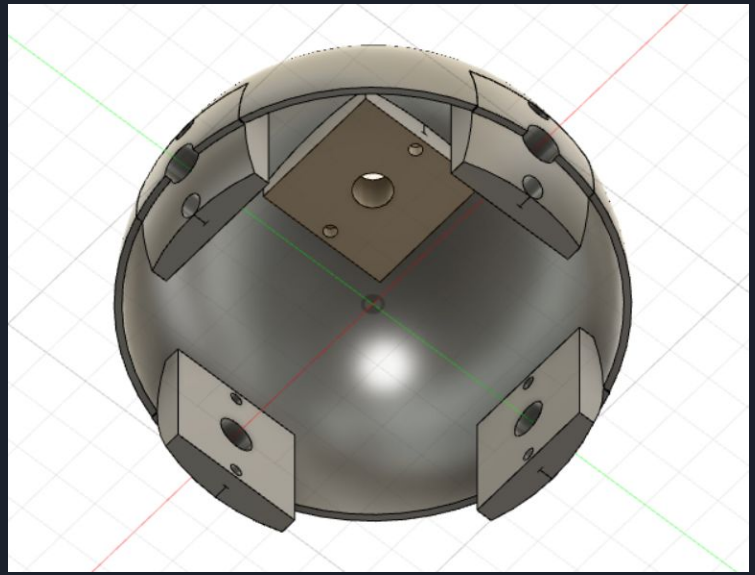
Prototyping





Prototyping





Deliverables

- 11x11x11 final product
- Charging equipment
- Source code in GitLab
- Delivered to Dr. Zambreno at the end of the project

Questions?