Self-Solving Rubik's Cube

Design Document

Team 29
Dr. Zambreno
Taylor Burton
Jacob Campen
Casey Cierzan
Joe Crowley
Annie Lee
Keegan Levings-Curry
Luke Schoeberle
sdmay20-29@iastate.edu
http://sdmay20-29.sd.ece.iastate.edu

Revised: November 3, 2019/Version 2

Executive Summary

Development Standards & Practices Used

- Follow IEEE standards
- Push early and push often
- Document every part of the process

Summary of Requirements

- Self-contained within the cube
- Easily scrambled by hand
- Solved within two minutes
- Lasts for three or more years
- Charges from an external source
- Fits within the typical senior design budget

Applicable Courses from the Iowa State University Curriculum

- ComS 309
- CprE 288
- ComS 228
- EE 224
- EE 311
- Phys 222

New Skills/Knowledge acquired that was not taught in courses

- Motors
- Clutches
- Rubik's cube algorithms
- 3D printing
- CAD tools

Table of Contents

List of figures/tables/symbols/definitions (This should be similar to the project plan)

1 Introduction

1.1 Acknowledgement

We would like to thank Dr. Joseph Zambreno and Lee Harker for their support.

1.2 Problem and Project Statement

Our project is to create a self-solving Rubik's cube by adding components within a cube. After the user scrambles the cube by hand, the cube must solve itself within two minutes.

This cube will be designed to inspire prospective students to pursue Electrical and Computer Engineering (ECpE). By witnessing the potential of an ECpE degree, they will be more motivated to study ECpE at Iowa State University (ISU).

1.3 Operational Environment

The finished cube must operate indoors on a fixed display table at room temperature.

1.4 Requirements

Here is a list of requirements:

- Can be scrambled by hand without requiring much force from the user
- Contains no external components (e.g. cameras, robot arms) except for charging devices
- Reliably solves the cube within two minutes
- Durable enough to last for at least three years
- The entire budget cannot exceed $750

1.5 Intended Users and Uses

Our cube's primary users are ISU tour guides and prospective families.

The tour guides will maintain the cube, which includes cleaning, charging, and lubing.

The prospective families, which may include high school students, parents, and small children, will scramble the cube.

1.6 Assumptions and Limitations

Here is a list of assumptions:

- The cube will always begin in a solved state
- Users will not need to indicate that they have finished scrambling the cube
- Users will only make turns in 90°-increments (e.g. 0°, 90°, 180°)

Here is a list of limitations:

- The cube must be solved within two minutes
- The cost for the cube cannot exceed $750
- The cube's side length must be in the range [5.7, 18] (in cm)

1.7 Expected End Product and Deliverables

Our end product is a self-contained, self-solving Rubik's cube. The final cube and all prototypes will be delivered to Dr. Joseph Zambreno for future tours. We will also provide a charging cord and an adapter. Additionally, we will provide our source code to Dr. Zambreno for future enhancements of this project. These deliverables will be submitted by Friday, May 8th.

2. Specifications and Analysis

2.1 Proposed Design

Our cube's side length must between 5.7 cm and 18 cm, and the cube will have a standard color scheme and layout. Each face will be independently controlled by electric motors. Since the cube must be easily scrambled by hand, we will be using stepper motors so that it can be scrambled without stripping the gears. To sense face movement, two Hall Effect sensors under each face will be paired with magnets within each face. When a turn occurs, the sensors will transmit meaningful analog voltages to the microcontroller, which will be interpreted by embedded software on the microcontroller.

The software will adhere to IEEE Standards. The software will control the clutches, ensuring that the motors are disengaged during the scrambling process. It will also track the cube's physical rotations during the scramble, which is used to simulate the rotations in software. After a period of inactivity, the software will initiate a solving algorithm based on the simulated cube's state. In doing so, it will send turn instructions to the motors until the cube is solved. By using algorithms that require few moves, the software will be able to solve the cube in less than two minutes.

At this point, the sensors and motors have been tested and the software rotation algorithms are in the testing phase. On the hardware side, we have printed a half sphere for housing our components and we have tested rotating one side, proving that we have enough torque to rotate the cube. Going forward, we will be finishing our housing sphere and working on connecting our microcontroller to our motors.

2.2 Design Analysis

As of 10/1, we have researched previous solutions, discussed multiple designs, and started to create our first prototype. Initially, we are using a large Rubik's cube for prototyping, and we have tested a few small magnets and Hall Effect sensors. We noticed that we may need different magnets since it is difficult to use the small magnets. Ultimately, we hope that we can use four magnets per face to capture the cube's rotation during the scrambling process. If we can do so, we will be able to keep every component within the Rubik's cube, as desired by our client.

By 11/1, we have started working on the basics of movement for our cube and defining the physical limitations of our motor system. We have started evaluating the advantages and disadvantages of a smaller cube and seeing how we will be limited by power constraints.

2.3 Development Process

For our project, we will be using a modified Agile workflow. We begin with the typical waterfall steps of high-level requirements specifications and design. However, we then follow an iterative Agile development, which will incrementally push us towards our final solution. By frequently displaying our progress to our client, we will understand our client's needs and desires more fully, so we will ultimately create a better final product.

For more details, see Figure 1 below:

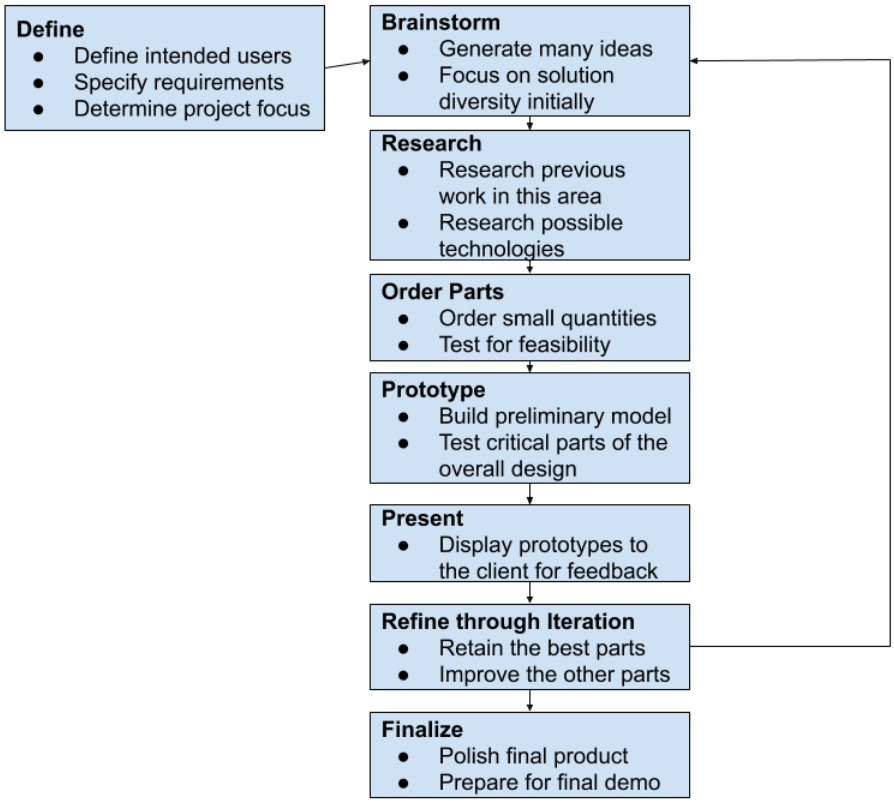

Figure 1: Development Process

2.4 Design Plan

We will initially design a large Rubik's cube that meets the basic requirements. Although a large cube may satisfy the requirements, it will not be ideal because larger cubes are more difficult to scramble. Once we have a functional larger cube, we will attempt to make a smaller cube by reducing the size of our internal components as much as possible.

3. Statement of Work

3.1 Previous Work And Literature

3.1.1 Similar Work

Two types of self-solving Rubik's cubes have been created in the past. The first type requires many external components, while the second cube uses only internal components.

Our client desires that we create the second type of cube.

3.1.1.1 External Solvers

The first self-solving cube type is a standard cube augmented with large external components. The cube operates within a custom-made stand that is connected to motors, cameras, and a desktop application. Generally, the motors turn the cube, the cameras determine the state of the cube, and the desktop application runs the solving algorithm.

See the following link for an example: Fast External Cube.

3.1.1.2 Internal Solvers

The second self-solving cube type is a custom cube augmented with tiny internal components. The cube is composed of 3D-printed parts, and it contains motors and microcontrollers within the cube's hollow center. Generally, the 3D-printed parts increase the size within the cube, while the motors turn the cube and the microcontroller runs the solving algorithm.

See the following link for an example: Self-Rotating Cube.

3.1.2 Solving algorithms

Many Rubik's cube solving algorithms exist, and they can be easily found online. In our case, our primary goals are algorithm efficiency and simplicity. However, these goals are often at odds in the solving algorithms, which complicates this issue overall. Ultimately, we will choose the fastest algorithm that we can fully implement.

See Table 1 for a discussion of a few possible algorithms:

Algorithm	Advantages	Disadvantages
Petrus	• Very efficient • Popular in speed solving • Uses the fewest moves	• Difficult to understand • Difficult to implement • Not designed for computers
Thistlethwaite	• Reasonably efficient • Designed for computers	• Not extremely efficient • Difficult to understand
Old Pochmann	• Simple to implement • Requires simple memorization	• Requires many moves • Ineffective for speed solving

Table 1: Solving Algorithms

3.1.3 Other resources

Additionally, we discovered other resources that may be useful during our project.

One article describes their design of an internal self-solving Rubik's Cube (Human Controller 2018]. It first illustrates how they created a large prototype before creating the standard-sized final cube. Additionally, it displays the additional internal components inside the cube. Overall, this article is very useful for us since we intend to create a very similar product.

See the following link for more information: Design of self-rotating cube.

We also found an open-source Rubik's cube applet ("Rubik's Cube Java Applet program"). This application provides a Rubik's cube simulator that allows us to visualize the cube easily, which is very helpful for testing. The applet also provides a few solving algorithms implemented in Java, which are very helpful examples of solving algorithms. Thus, this Java applet is an extremely useful resource for us.

See the following link for more information: Open Source Java Applet.

3.2 Technology Considerations

See Table 2 for a discussion of the previous approaches:

Previous Approach	Advantages	Disadvantages
External Solvers	• Identifies cube state with cameras • Powered with external motors • No space concerns • Solves the cube very quickly • Uses mostly standard cubes	• Not interactive • Requires external setup • Requires machine learning knowledge • Not portable
Internal Solvers	• Very intriguing for the audience • Portable • Contains no external components • Solves the cube reasonably fast • Easily turned by hand	• Limited identification of cube state • Expensive • Many space concerns • Requires a custom cube

Table 2: Discussion of Previous Approaches

In our case, our client desires an internal solver, which limits our possible solutions. However, even with these limitations, there are many technology considerations, such as motor type, state identification, and system integration.

By investigating different motor types, we determined that a stepper motor is best for our cube. Since the cube must be manually turned during the scrambling process, the motors cannot damage other components during these turns, which is a problem for servo motors. For servo motors, gearboxes ensure that manual turns do not damage its internal components, but gearboxes are typically large and difficult to handle. As a result, although stepper motors are generally less powerful than servo motors, we decided to use stepper motors because eliminating the gearbox provides more benefits than improving turn performance. Consequently, we have experimented with stepper motors, ultimately choosing motors that can move the cube easily without damaging the internal components during manual turns.

We will use Hall Effect sensors to determine the state of the cube. These sensors will use magnets embedded in the cube's face to sense magnetic field changes, which will transmit the cube's rotations to the microcontroller via analog voltage signals. voltages. This will ultimately allow the microcontroller to simulate the cube's rotation, assuming that the cube starts in a solved state. We

will use at least two sensors on each face to consistently detect rotations, which results in a total of at least twelve Hall Effect Sensors.

We will control the cube's operations with a Teensy 4.0 microcontroller, which was primarily selected for its small size and integration with Arduino libraries. This microcontroller has ample ports and uses a 3.3V power supply, which is ideal for our application. Ultimately, the Teensy will allow us to control the motors, receive rotation signals, and implement the solving algorithm within the cube, which is critical for system integration.

3.3 Task Decomposition

See Figure 2 for an overview of the tasks in this project.

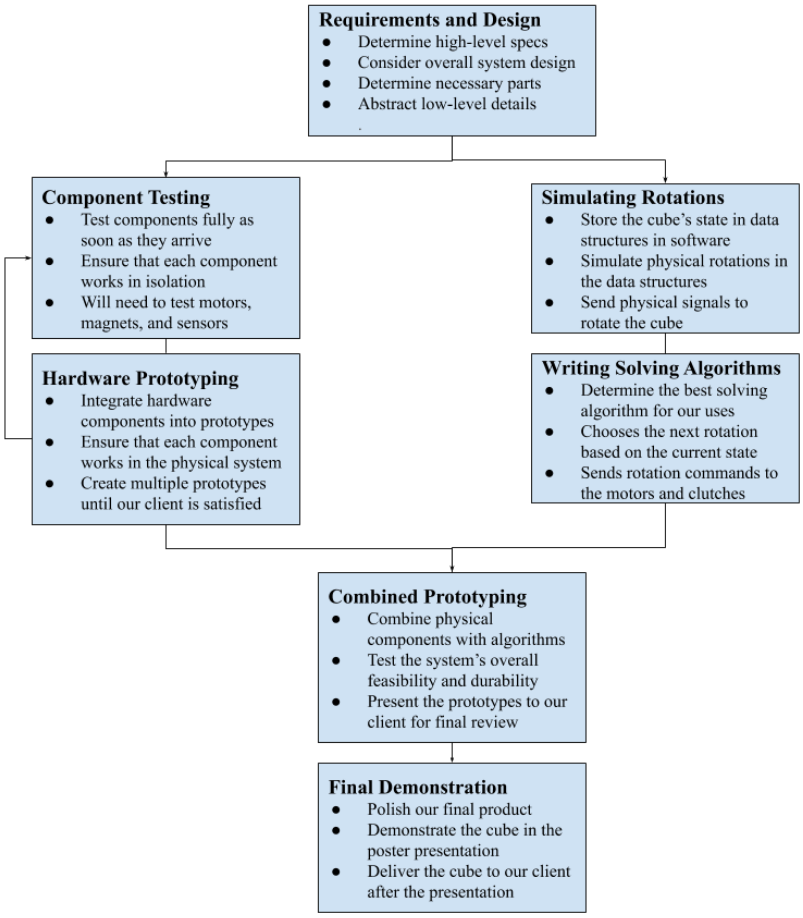

Figure 2: Task Decomposition

3.4 Possible Risks And Risk Management

Our project is potentially risky because of size concerns and our limited budget.

Since we need to create an internal solver, we will have very limited space for the internal hardware components, which introduces many possible issues. We may need to test multiple components

because the magnets and other electrical components may interact within the cube's small center. Additionally, the space restrictions will greatly increase the costs of our electrical components, which may ultimately exceed our budget.

We will manage these risks in two ways. We will carefully consider the interactions between our electrical components before we physically test them, which will mitigate the risk of interactions within the cube. We will also manage our budget by meticulously tracking our purchases and buying cheap components as much as possible.

3.5 Project Proposed Milestones and Evaluation Criteria

To ensure timely project completion, we have established milestones for this project. These milestones currently include Solving Algorithm Implementation and Hardware Prototype Completion, which are defined in the following subsections.

3.5.1 Solving Algorithm Implementation

This milestone is satisfied when our solving algorithm is fully functional in software. To achieve this milestone, we must develop and test the solving algorithms with simulated rotations. We also must send the correct hardware signals during this process.

We will test this milestone by manual testing and with the assistance of the open-source Java applet described above in section 3.1.3. Initially, we will manually ensure that the simulated rotations are functional, focusing on the simulated cube's new state and the signals for the motors. Later, we will ensure that the algorithms works correctly within the Java simulator.

3.5.2 Hardware Prototype Completion

This milestone is satisfied when our hardware prototype can consistently rotate the cube without damaging internal components. To achieve this milestone, we must first produce functional components in isolation before testing the full prototype. Later, we must design and implement a reliable system for performing rotations on the cube.

We will test this milestone by manual testing. Initially, we will ensure that each component is functional individually. Finally, we will test that the system can turn the cube as desired without damaging the insides of the cube. In doing so, we will specifically check for tangling wires by performing the same turn repeatedly.

3.6 Project Tracking Procedures

To ensure timely project completion by next May, we will follow the schedule described below in section 4.1 as closely as possible.

If we miss any deadlines, we will immediately schedule an emergency team meeting to resolve the issue. Ideally, this will never happen, but our team is prepared to attend this meeting if necessary.

3.7 Expected Results and Validation

Our desired outcome is a self-contained, self-solving Rubik's cube. See Sections 1 and 2 for more details about the final outcome.

If users can consistently scramble the cube, let it lie on a flat surface, and watch the cube solve itself within two minutes, our solution will be fully functional.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

See Figure 3 below for our project timeline. Week 1 denotes the first week of the Fall 2019 semester, and then Week 2 denotes the following week (and so on). Our timeline only includes weeks in which classes occur, so Thanksgiving week, winter break, and spring break are not included.

Note that this timeline is not rigid, so it may be modified slightly in the future if unexpected complications arise.

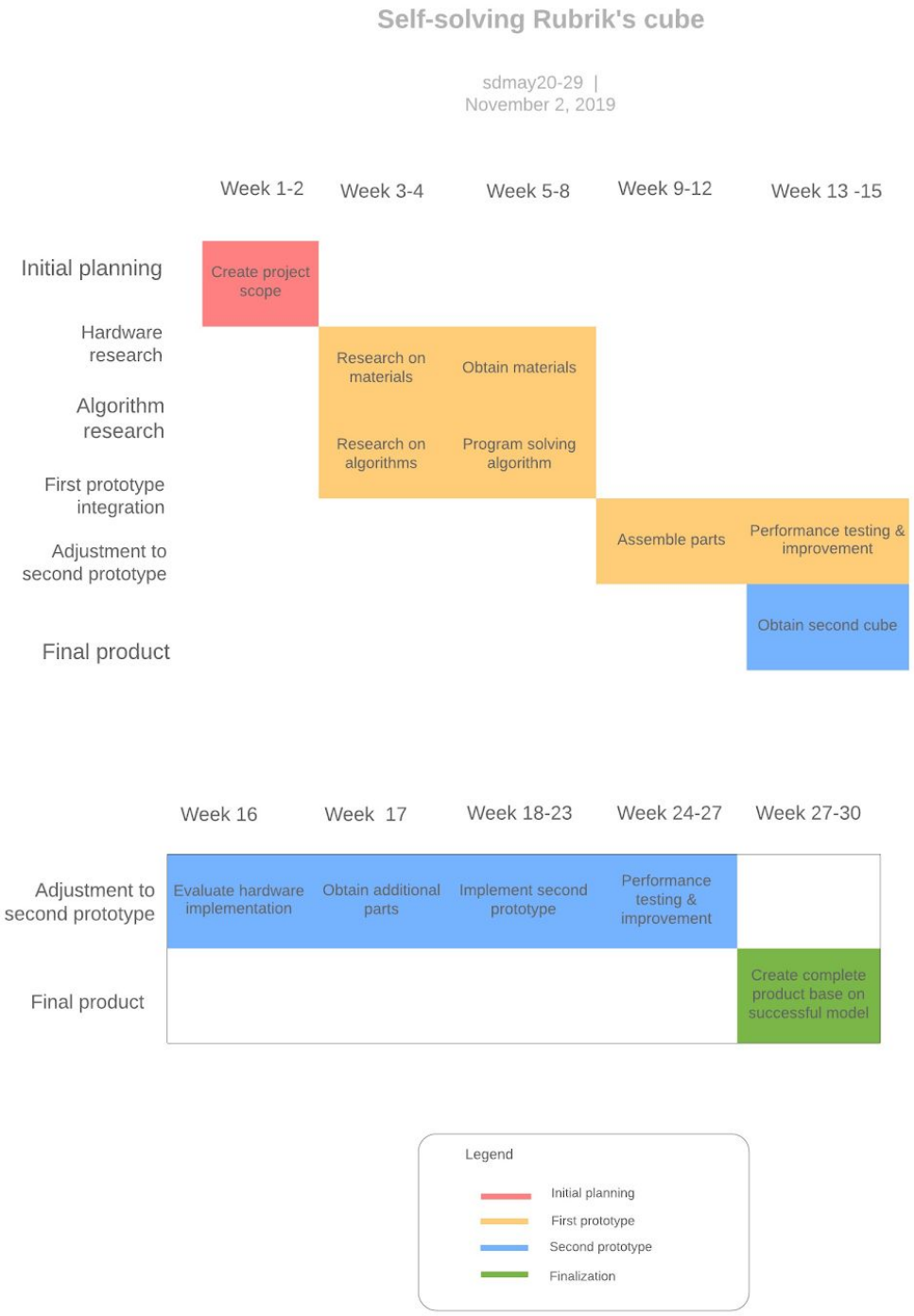

Figure 3: Project Timeline

4.2 Feasibility Assessment

Ultimately, creating a self-solving Rubik's cube will be challenging, but it is certainly possible for our team of electrical and computer engineers because we already have most of the skills needed to

solve this project. For instance, we know how to use microcontrollers and Hall effect sensors. Overall, we will face three major challenges during the project: size constraints, motor systems, and component costs.

First, since we are creating an internal solver, we will need to place every internal component inside the cube's tiny center. Consequently, it will be very difficult to fit these components inside the cube without skillful 3D modeling, which is not our team's strength. To alleviate these size constraints, we have decided to create a larger prototype to test our ideas, but size remains a large challenge in our project.

Second, since we need to turn the cube programmatically, we will need to use motors systems inside the cube, which is a challenge since our team is unfamiliar with motors overall. Consequently, we will need to do extra research and testing with motors to determine the best options for our cube, which will be time-consuming and potentially expensive. In this way, choosing the correct motors will be difficult since we are not mechanical engineers.

Lastly, component costs will be a challenge due to the size constraints and motor experiments. In general, miniature electrical components are exponentially more expensive than larger components, so component costs will become a large concern if we decrease the size of our cube. Additionally, motors are fairly expensive, so our additional motor testing will become expensive over time if we are not careful. Consequently, managing our budget will be difficult in this project.

4.3 Personnel Effort Requirements

See Table 3 below for a detailed task breakdown of our project:

Task	**Projected Effort**
Create Project Scope	**Entire team**: ● Discuss project scope with our client ● Determine acceptable solving time, cube size, and other requirements
Research on Materials	**Keegan, Casey, Jacob, Taylor**: ● Research batteries for optimal size-to-current ratios ● Research motors for optimal size-to-cost and size-to-torque ratios
Obtain materials	**Casey**: ● Order parts from ETG when necessary ● Manage our budget carefully
Implement Motors/Feasibility Testing	**Jacob, Taylor**: ● Wire the motors properly ● Ensure that the motors can reliably turn the cube's bottom face
Hardware Design and connectivity/wiring	**Jacob, Taylor**: ● Create a 3D model of the system

	● Design PCBs for motors and other circuits
Research on algorithms	**Joe, Annie, Luke**: ● Research solving algorithms online ● Choose the best algorithm for our cube
Problem solving algorithm	**Joe, Annie, Luke**: ● Write code for performing rotations and solving algorithms in C ● Write code that integrates properly with our hardware
Assemble parts	**Casey, Jacob, Taylor**: ● Ensure that we received the proper parts ● Retain usable parts from the first prototype and scrap the other parts
Initial Performance testing and improvement	**Keegan, Jacob, Taylor, Joe**: ● Test current and voltage requirements of our initial electrical components ● Assess the initial system's reliability
Obtain second cube	**Casey, Taylor**: ● Order a smaller cube if possible ● May also 3D-print a cube if necessary
Evaluate hardware implementation	**Joe, Taylor**: ● Test the system overall ● Ensure that the system reliably satisfies the original specifications
Obtain additional parts	**Casey, Taylor**: ● Order better parts if necessary ● Continue managing the budget carefully
Implement second prototype	**Jacob, Taylor**: ● Improve the design of the first prototype ● Physically implement the new hardware
Second Performance testing and Improvement	**Keegan, Jacob, Taylor Joe**: ● Test current and voltage requirements of our new electrical components ● Assess the new system's reliability
Create complete project based on successful model	**Entire team**: ● Tweak the second prototype to produce the final product ● Present the final cube to our client

Table 3: Effort Requirements

4.4 Other Resource Requirements

We are currently using the following parts:

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

4.5 Financial Requirements

The project must fit within the typical Senior Design budget of $750.

5. Testing and Implementation

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)
2. Define the individual items to be tested
3. Define, design, and develop the actual test cases
4. Determine the anticipated test results for each test case
5. Perform the actual tests
6. Evaluate the actual test results
7. Make the necessary changes to the product being tested
8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you've determined.

5.1 Interface Specifications

– Hall Effect Sensors

The Hall Effect sensors will transmit an analog signal to the microcontroller. This signal carries the position of the cube, which will have to be tracked in software. Total of 12.

– Clutches

The motors will be connected via electrically actuated clutch, of which the static state is engaged. This means the software will have to detect movement and disengage the clutch. 6 total.

5.2 Hardware and software

Hardware

- Full range of motion
- Repeated scramble and unscramble possible
- Time test, withstand repeated use

Software

- Unit tests for cases of unsolved cubes
- Be able to record a scramble of upwards of 100 moves
- Be able to translate scramble rotations into an unsolved cube in the program's memory

5.3 Functional Testing

Hardware

- Rotations must be 90 degrees
- Clutch must disengage during a scramble
- Clutch must engage during a solve
- Power supply must last through a solve
- All wired connections must stay connected during rotations

Software

- Given an unsolved cube, must be able to produce a list of rotations required to lead to a solved cube
- Must be able to send turn signals in an appropriate manner

Non-Functional Testing

Performance: Our cube must be durable, and last for a couple years. During a solve, it must not drain the battery fully and stop midsolve. Our cube must solve itself in 2 minutes.

Security: Our cube must not have sharp edges and must not shock a user if they touch the charging port.

Usability: Our cube must be able to be held effortlessly so that our end user can scramble. For the scramble, our sides need to turn easily without grinding the gears that will turn the sides during a solve.

Compatibility: We will need a power supply cable that can be plugged into a standard outlet.

5.4 Process

Our testing process will follow the same scheme as our design process. We start by defining what needs to be tested, and what our expected outcome should be able to do. We then brainstorm how to test for this quality, and then we jump to prototyping our testing environment. We will present the findings of our tests to each other first, in order to ensure that a good testing process was followed. Finally, we will refine our product through the results of our testing.

5.5 Results

- We had success in testing our Hall Effect sensors with our ordered magnets, and we are able to visualize how the control unit will fit in the middle of the cube.

- Our first setback is figuring out the motors and components needed to actually rotate our sides, and our lack of motor knowledge will be a hurdle to overcome.

-**Modeling and Simulation**: This could be logic analyzation, waveform outputs, block testing. 3D model renders, modeling graphs.

- To model our end product, we may use CAD software to see how it all fits together before buying all our components.
- To test our solving algorithms, we will be writing unit tests with different scrambles of a Rubik's Cube.

6. Closing Material

6.1 Conclusion

In conclusion, we have completed the initial design steps, and we have started to create prototypes. Currently, we plan on using magnets, Hall effect sensors, and stepper motors in our prototype. We plan to present this prototype to our client by the end of the semester.

6.2 References

Flatland, Jay. "World's Fastest Rubik's Cube Solving Robot - Now Official Record Is 0.900 Seconds." YouTube, YouTube, 11 Jan. 2016, https://www.youtube.com/watch?v=ixTddQQ2Hs4.

Controller, Human. "Self Solving Rubik's Cube." YouTube, YouTube, 17 Sept. 2018, https://www.youtube.com/watch?v=xCoH2AORcEQ.

"Petrus Method." Petrus Method - Speedsolving.com Wiki, https://www.speedsolving.com/wiki/index.php/Petrus_Method.

"Thistlethwaite's Algorithm." Thistlethwaite's Algorithm - Speedsolving.com Wiki, https://www.speedsolving.com/wiki/index.php/Thistlethwaite's_algorithm.

"全自動ルービックキューブ　Self Solving Rubik's Cube by Human Controller." DMM.make, https://media.dmm-make.com/item/4462/.

"Blind Solving Algorithms." SpeedCubeReview.com, https://www.speedcubereview.com/blind-solving-algorithms.html.

"Rubik's Cube Java Applet Program." Rubik's Cube Java Applet Program, https://ruwix.com/the-rubiks-cube/rubiks-cube-java-applet-software-program-josef-jelinek-animating-rubix/.

6.3 Appendices

We do not currently have any appendices.