Team 3 - Project Proposal

Team Number: 3

Team Members: Diego Garcia, Luis Belmontes, Ankeet Prasai, Karen Liu, Xingtong Zhang

Project Name: **Solved^3**

Project Synopsis:
Software and Hardware application that will use motors to solve a rubik's cube in any state by using an algorithm.

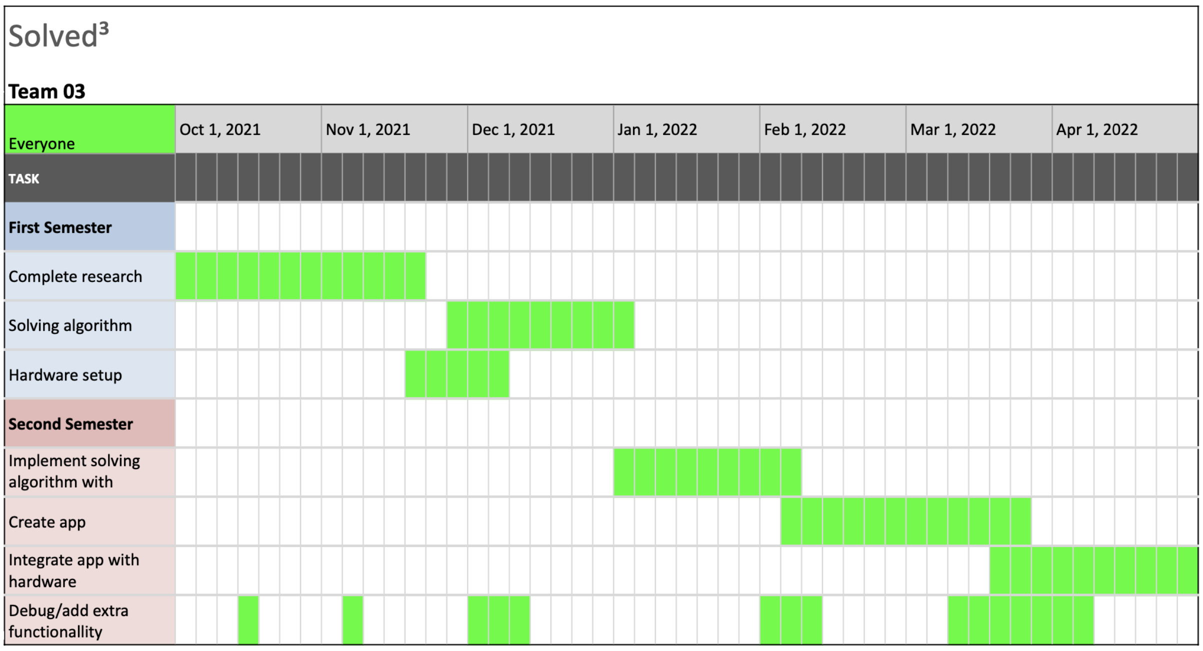
Project Description:
The project came to light by exploring the unique problem of solving a rubik's cube. Our team shares the joy of solving rubik's cubes. We then decided it would be interesting to develop software that can use known algorithms to indicate what steps to follow to solve a cube in any state. The project will also involve image processing as we'll have to use cameras to capture the state of the cube and translate the data. Fearing that our project lacked complexity, a hardware component was decided upon where claw-like components powered by motors will physically change the state of the cube based on the encoded algorithm to solve it. The opportunity to work with hardware was appealing to all team members since we were all computer science majors who don't have many opportunities to work with hardware in our degree program. It will also be to great benefit that our team works with hardware because we'll learn more about the integration of software and hardware. Ultimately, the end result of our project is to have a fully functional hardware application that can be given a rubik's cube in any state and be solved mechanically by the integration of hardware, software, and algorithms.

Project Milestones:
First Semester:

- Finish initial research on hardware and software implementation, for hardware research best parts to use, potential 3-D printing, integration of all the different systems together.
  Deadline: 10/17/2021
- Have the Rubik's cube solving algorithm ready, this means that the application will be able to tell what is the next step based on the state of the cube.
  Deadline: 12/1/2021
- Have our hardware set up so that we have the mechanics of how to move the cube. Essentially just making sure that our hardware implementation will be ready to begin.
  Deadline: 12/3/2021

Second Semester:

- Implement the solving algorithm with hardware so that the Rubik's cube is solved physically.
  Deadline: 2/11/2022
- Create an app that will use our algorithm to tell a user the move required to solve the cube.
  Deadline: 3/11/2022
- Integrate the app with our hardware so that the app shows what the next step is, while the hardware physically does it.
  Deadline: 3/25/2022
- Add extra functionality depending on how much we have advanced. Fine tune any small details and optimize our implementation.
  Deadline: 4/29/2022



Project Budget:

Items	Quantity	Price	Vendor
Raspberry Pi	1	$ 175.99	Amazon | link
Servo Motors	5	$ 25	Amazon | link
3-D Printing	1	$ 25-400	N/A
Wiring Kit + Arduino	1	$ 42.89	Elegoo | link
Raspberry pi cam	1	$ 26.39	Weewooday | link
Rubiks Cube	3	$ 8.79	D-fantix | link
Speed Cube	2	$ 71.45	Gan | SpeedCubeShop
Soldering Iron	1	$ 128.99	SainSmart | link
Total		**$ 568.53**	**Total excluding possible 3-D printing costs.**

Ethical Issues:

- In order to monitor the current stage of our user's rubik's cube, the use of a camera is necessary to capture every face of rubik's cube. The application will then process the algorithm and output the move instruction to the user. This can possibly lead to user privacy problems, since every taken image needs to be processed through the application, those images could include background information that might violate the user's privacy. Since taking images is required for the use of this application, a possible way to address this issue is to remind users at the starting phase of camera use and to obtain permission from them. That way, it's their responsibility whether or not they want to share their images through our application.
- The fun of the rubik's cube is the process of learning to solve. Having a program available to teach about how to solve rubik's cube might affect people's willingness to learn to solve the rubik's cube themselves since there's a program available that may be more efficient at solving the rubik's cube when compared to the majority of human beings. The popularity of automating physical tasks may lead society to always rely on technology and disconnect us from reality. So although a project like this takes us actual work, it still may discourage people from trying on their own with no help from technology because it's just not necessary anymore and not fun anymore.
- Health and safety of users: The product design must include design constraints so that the hardware cannot in any way harm the users. As such the design must be eventually made into a closed system with all parts and internal hardware enclosed in a shell not easily accessible by the user.

Intellectual Property Issues:

- We will be the first to admit that we are not the first to create a rubik's cube program or algorithm. The algorithms to solve rubik's cubes are open source and available all over the internet. However we understand that we will not grow if we merely copy off of what is available over the internet. Our goal for this project is to first learn from the algorithms and solutions available online and to create our implementation. Therefore we will be trying our absolute best to come up with our own original ideas, and everytime we use either open source libraries or free to use algorithms we will credit the source.

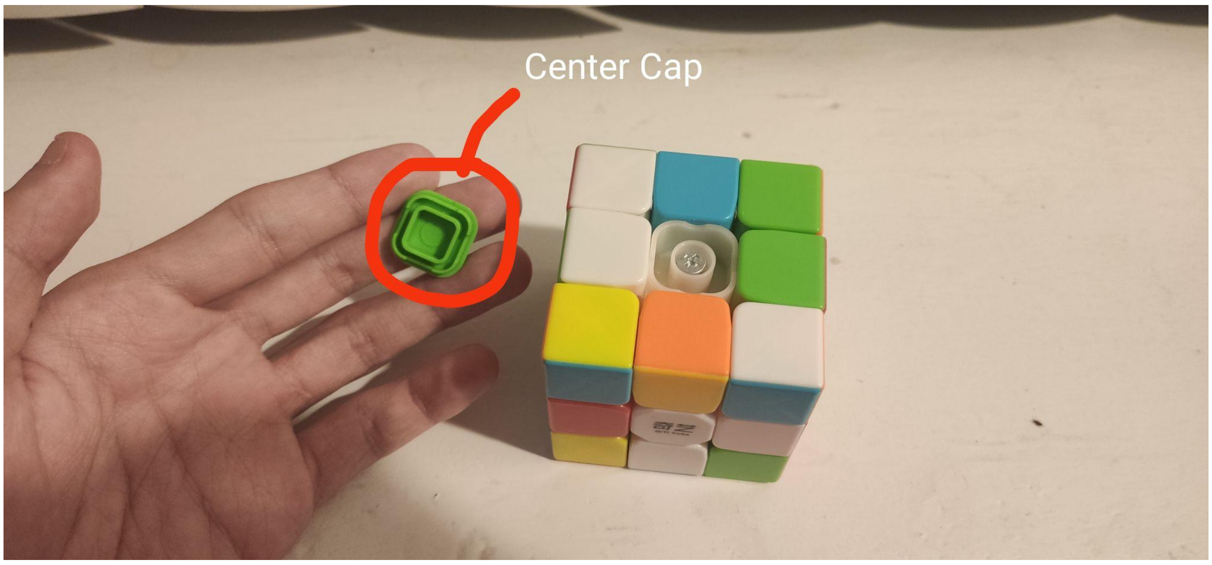
Change Log:

The deadlines of our milestones shifted drastically since we underestimated the amount of research that needed to be done to move forward with designing or formulating possible implementations.

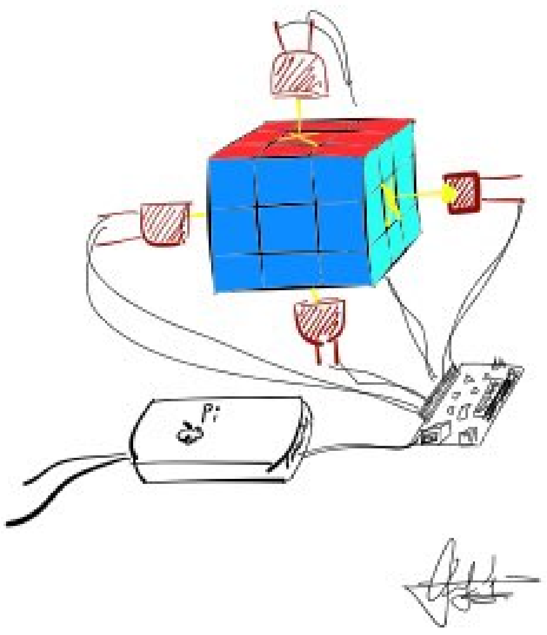
Initially, we thought that we would need to use a lower level language to program the hardware but since we're just changing the movement of the servo motors and taking the camera's input. We discovered that we can program the hardware with python alone or even with a separate python library.

Design Constraints:

When thinking about the way that we are going to implement the hardware, we arrived at a point in which we had to decide how the cube is going to be turned. We decided that there are two main design constraints that need to be addressed. Since we plan on using servo motors, we had to make sure that we choose proper motors that will be able to turn the cube with ease. This way, we avoid putting too much stress on the motors and potentially damaging or even breaking them, this is the first part of our design constraint. To solve this, we chose servo motors that will have enough torque to be able to turn a Rubik's cube with ease. Another way to ensure that the Rubik's cube can be turned properly is by purchasing Rubik's cubes that are designed for speed solving. These speed solving cubes require very little force to turn the edges, and their tension can be easily adjusted using a screwdriver or a tool of some sort. This will help us avoid any issues that can come up if our servo motors end up not having enough torque. These cubes can also be lubricated if taken apart, adding on to our measures of avoiding this issue. The second part of this design constraint comes when deciding how we are going to be able to use the servo motors. For this, we are going to design, and 3D print some basic components that can be fitted into the center of the cube. The center pieces can be removed easily, and when removed, it creates a slot in which we can insert a 3D printed part that can be attached to the motors. By doing so, this will allow for the cube's faces to be turned properly.



(Credit: Xingtong Zhang)



Credit: (DIEGO GARCIA)

Our rubik's cube implementation will follow two popular beginner friendly algorithms. These two algorithms don't have an official name but they have a similar approach to solving the cube. The algorithm solves the cube layer by layer which makes the implementation slightly easier because we don't have to look at very specific colors, cells or states of the cube to perform an operation. Advanced and speed solving algorithms require this but will be much harder to implement since our team members aren't at all familiar with these algorithms. One can say that the constraints of our implementation depends on the complexity of the algorithms we intend to use. The algorithms that we use start off by first picking a face that represents a color which is the color cell in the middle. In this face, a cross made up of the same color must be obtained. Next, the corners are changed to match the color of any face that it touches by doing so the layer on the adjacent faces is being completed. The cross should also have been completed in manner so that the colors cell match the adjacent faces. To complete the second layer of the cube, specific steps need to be followed to move pieces in the right place without undoing the progress made in the first layer. Similarly, this will be the case for completing the last layer of the cube. The first layer of cube will likely be the trickiest because specific colors can be anywhere whereas for the 2nd and 3rd layer the color cells needed will be in similar locations. So, it'll be easier just to apply the algorithms to solve the cube. This is a brief explanation of the process that goes behind completing a rubik's cube with basic algorithms. To implement this with our python code, our idea is to create functions that can carry out specific steps of the algorithm that will modify our data structure that represents a rubik's cube. The solver function will look for specific states of the cube to apply the correct function to move forward in the completion of the cube. It's important that we don't alter the rubik's cube data structure in a way that isn't physically possible. This is also key because our implementation will be carried out by the hardware so it could damage the cube. The same functions that modify the rubik's cube data structure will be programmed to send signals to servo motors to perform operations on the actual cube. Lastly, the image processing piece of our project will be carried out by a camera connected to the raspberry pi. The python library opencv will be used for image processing that will ultimately translate the physical state of the cube to match the data structure of our cube in our python code.

We decided to use python as our base coding language for the Rubik’s Cube project. It offers a large amount of python libraries that can be useful for our future development on the image processing for the Rubik’s Cube. The complexity and shortage of C++ language isn’t ideal for this project, while on the other hand, python is easy to lean and easy to put in practice when comparing to other low-level programming language for example C++ which becomes harder as the program advance its features, there might be a case where one of the team members isn’t familiar with the python language, the simplicity and easy to learn features allows the team member to quickly catch up with the development of our projects. The simplicity and easy to use features of python allows us to write simple and easily readable code, this is helpful when we are trying to develop algorithm learning and image processing. Also, the python libraries play a crucial role in developing image and data manipulation applications for our Rubik’s Cube, there are tons of libraries out there that fit our needs for image processing for both the application itself and the future development on our hardware side. The overall reason why we are moving from a low-level language to python is for the simplicity and easy use purpose and the libraries that can speed up the process when developing the key features of the Rubik's Cube solver.