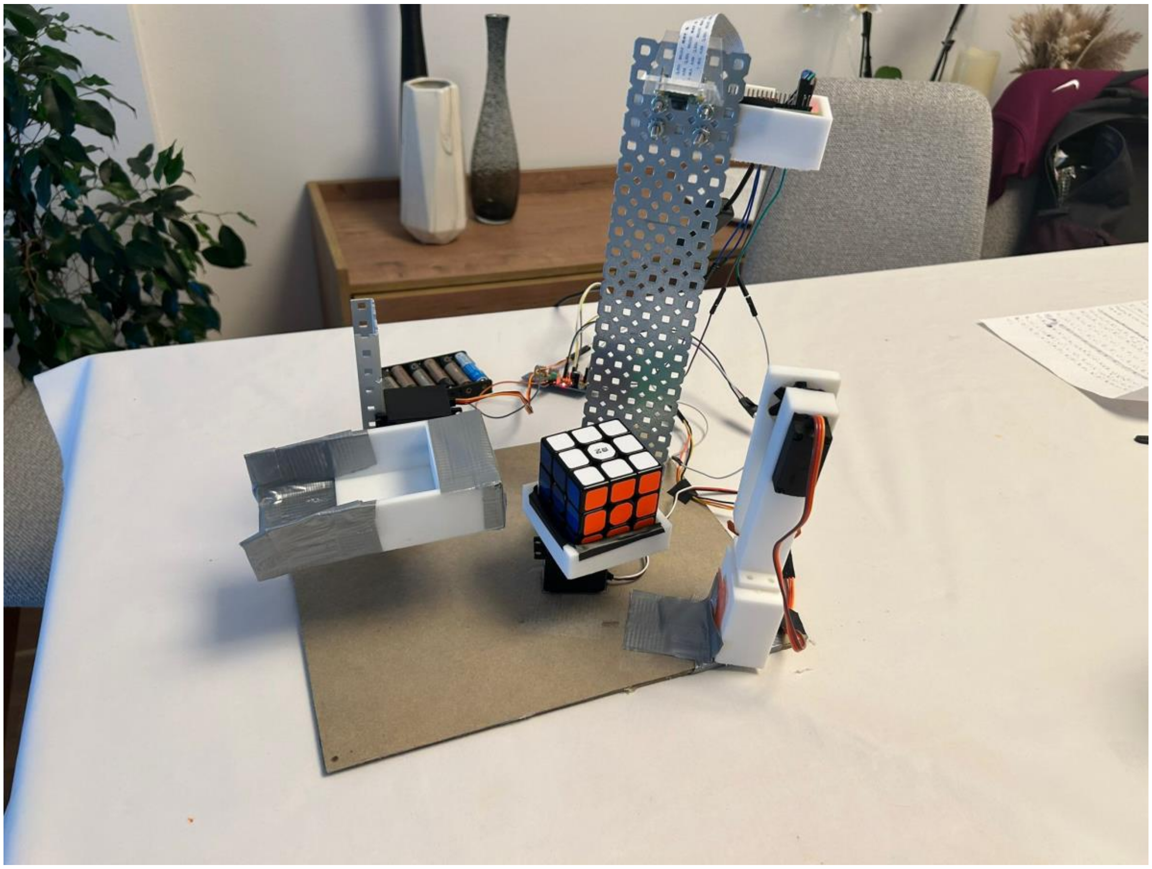


REPLICUBE

The robot that solves and replicates Rubik Cubes.

PROJECT Report.
DATE: 05 June 2023

Joel Paz Cepero
Gabriel Bardají Biescas
Pau Carracelas Expósito
Víctor Valero Carrasco

Table of Contents

REPLICUBE

The robot that solves and replicates Rubik cubes

Project description

This robot is equipped with advanced computer vision capabilities that enable it to solve a Rubik's Cube and replicate the pattern of another cube with high accuracy. The robot utilizes a camera to scan and capture the configuration of a Rubik's Cube, which is then stored in its memory for further processing.

Using sophisticated image processing algorithms, the robot analyzes the captured data to identify the current state of the Rubik's Cube, including the position and orientation of each colored cubelet. It then employs an efficient solving algorithm to calculate and execute the optimal sequence of moves to solve the cube.

In addition to solving the Rubik's Cube, this robot also has the unique ability to copy the pattern of another cube and replicate it. It uses its computer vision capabilities to scan and capture the pattern of the target cube, and stores the information in its memory.

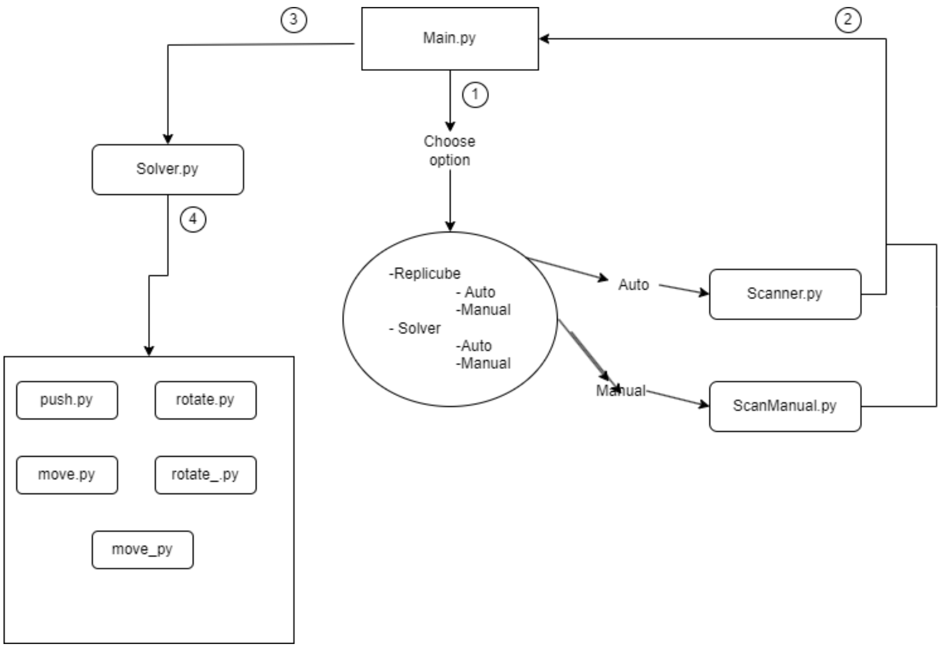
The robot then uses its solving algorithm to calculate the sequence of moves required to replicate the pattern on its own cube, and executes the moves to recreate the pattern with precision.

Electronic components

This is the list of the proposed components:

- Raspberry Pi 4 Model B x 1
- DF15RSMG 360 Degree Motor x1
- POWERHD 6001HB Analog Servo x1
- SPRINGRC SM-S4303R Motor CC x1
- Cámera 5MP OV5647 x 1
- Raspberry Pi official micro usb x 1
- PCA9685 Servo's controllers x1
- Bateria x6 pilas AA x1

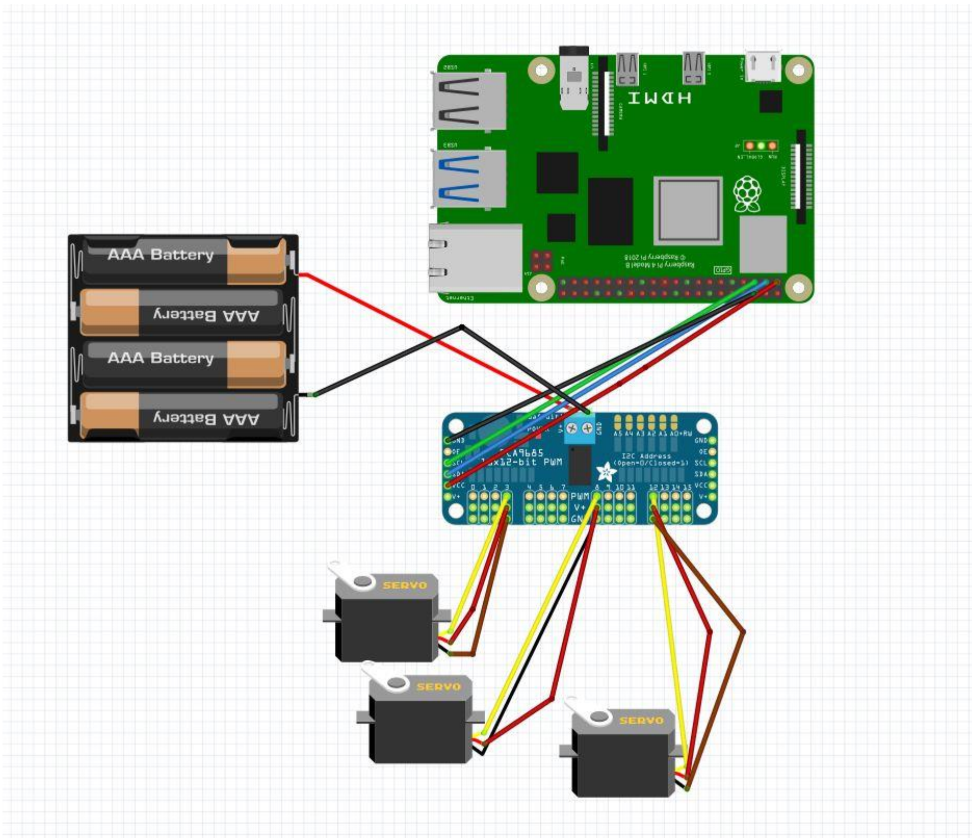
Software Architecture



Our software architecture is led by a main program in which we have a menu where we decide to solve or replicate the cube. This main program also calls the resolution algorithm and also the Computer Vision one. This last one is only called if we decide to scan the cube automatically instead of manually in the initial display(menu) of the main program. What we do in the scanning phase independently of the option we've chosen is, tell our program which is the color that we have in every "mini" square for all the 6 faces of the cube.

On the other hand, what concerns the solver algorithm, what we do is the different types of moves that exist for a cube in which we involve the few different moves we do with our robot (push, rotate, rotate_ ("_" invert side), move, move_). We have to do this because the moves that we can do with the robot are more limited than the ones we can do with our hands. What push does is move forward the cube so we can reach the face that is positioned frontally. If we do this various times we can get "down" and "back" faces too. Rotate moves the whole cube to one side or another. And move, holds the cube and only rotates the base, also, as rotate, to one side, or another.

Hardware Structure



Amazing contributions

1 - Dual functionality: Replicube's ability to both solve a Rubik's cube and replicate the pattern of another cube is a unique feature that sets it apart from other Rubik's cube-solving robots. This dual functionality adds versatility and practicality to the robot.

2 - Advanced solving algorithms: Replicube uses advanced algorithms to determine the most efficient solution for solving or replicating the Rubik's cube patterns. These algorithms enable the robot to operate quickly and accurately

3 - User interface: Replicube's LCD screen and button controls provide a user-friendly interface, allowing users to select the desired functionality and monitor the robot's progress during operation.

4 - Compact design: Utilizing a Raspberry Pi 4 Model B, Replicube is able to maintain a small form factor while still delivering powerful computational capabilities. This compact design allows for easy transport and use in various settings.

5 - Modular design: The modular design of Replicube allows for easy upgrades and modifications, such as adding new features or improving existing components. This adaptability makes the robot suitable for various applications and user preferences.

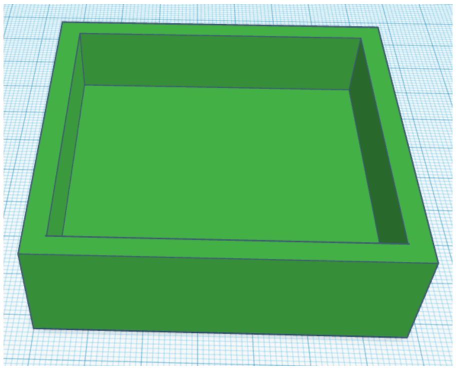
6 - Advanced computer vision capabilities: One amazing contribution of this project is the development of a robot with advanced computer vision capabilities. By utilizing a camera and sophisticated image processing algorithms, the robot is able to scan, capture, and analyze the configuration of a Rubik's Cube with high accuracy. This breakthrough in computer vision technology enables the robot to understand the current state of the cube, including the position and orientation of each colored cubelet.

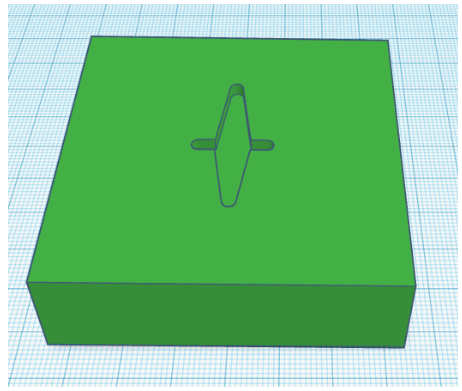
Replicube, the dual-function robot capable of solving and replicating Rubik's cube patterns, deserves a high score of 9 due to its innovative features and impressive capabilities. With computer vision as a new technology for the team, the project showcases the ability to adapt and incorporate cutting-edge techniques. The algorithmic complexity of the project demonstrates advanced problem-solving and adaptability, making it an impressive achievement.

The project's purpose is eye-catching and innovative, setting it apart from other robotics projects. Its compact design, enabled by the use of a Raspberry Pi 4 Model B and other carefully selected components, makes Replicube both portable and versatile. Furthermore, the project is budget-conscious, successfully fitting within the allocated 100-euro budget without compromising on functionality or quality.

Extra components and 3D pieces

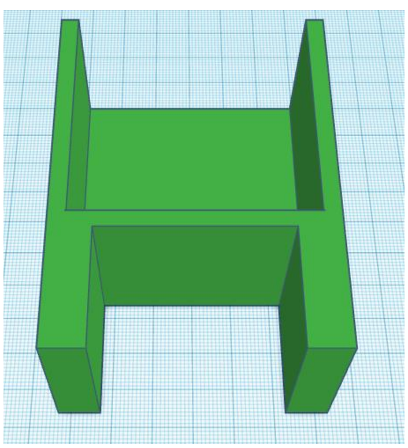
- *BASE*

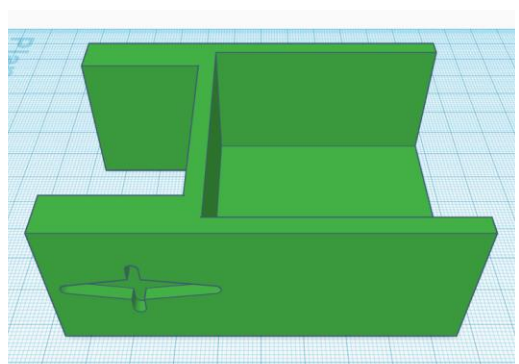




This piece will be the support for the Rubik's cube. On the bottom, it has a hole where the servo motor will be connected, and it will rotate, changing the state of the cube.

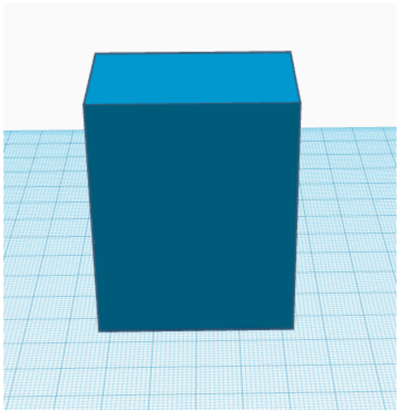
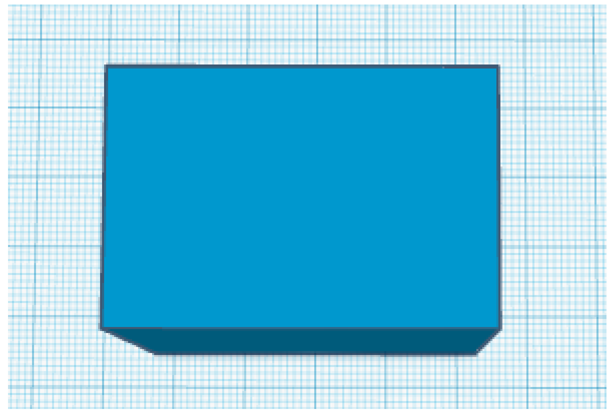
- *CUBE-HOLDER*





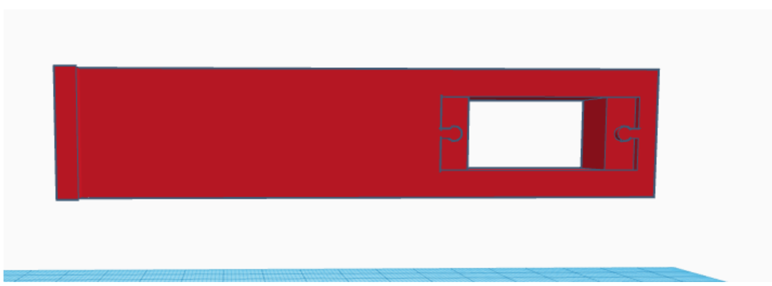
This piece will be responsible for holding the top part of the cube immobilized while the base rotates and moves the cube to the next state. On one side, it will have a hole that will fit with a servo motor which will rotate the piece 180º to hold or release the cube.

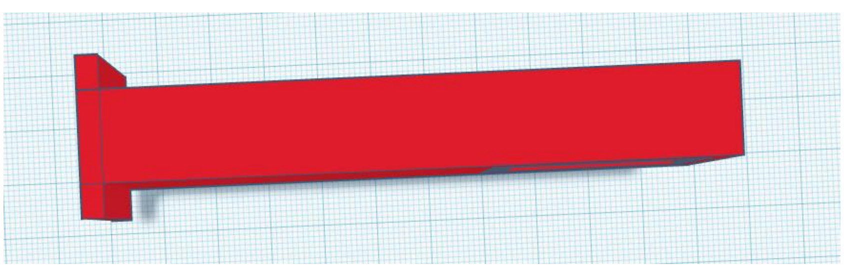
- *BASE-KNOCKER*

This piece will be attached to the base of the robot, and its main function will be to provide support for another 3D piece, named the Knocker.

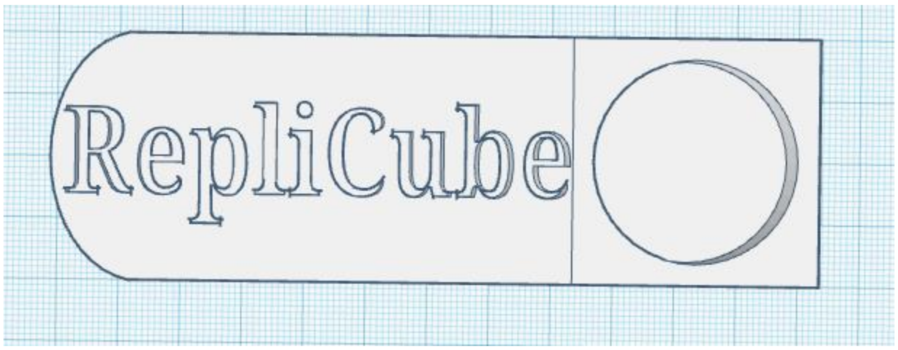
- *SERVO-HOLDER-KNOCKER*

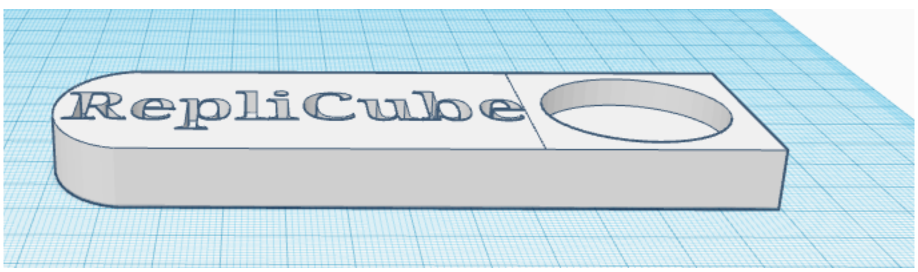




This piece will contain a servo motor inside, which will be responsible for rotating the Knocker. It will be attached to the previously mentioned 'BASE-KNOCKER' piece.

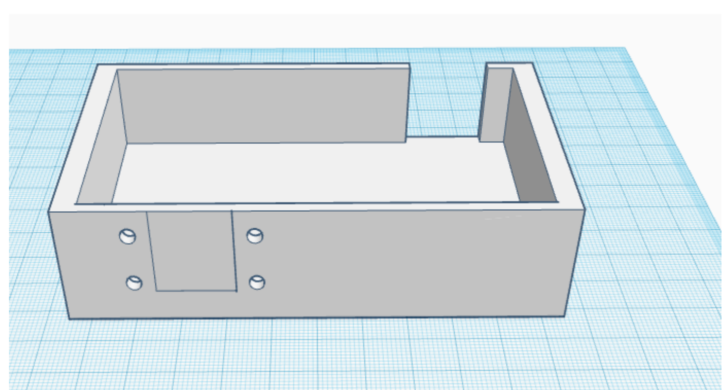
- *KNOCKER*

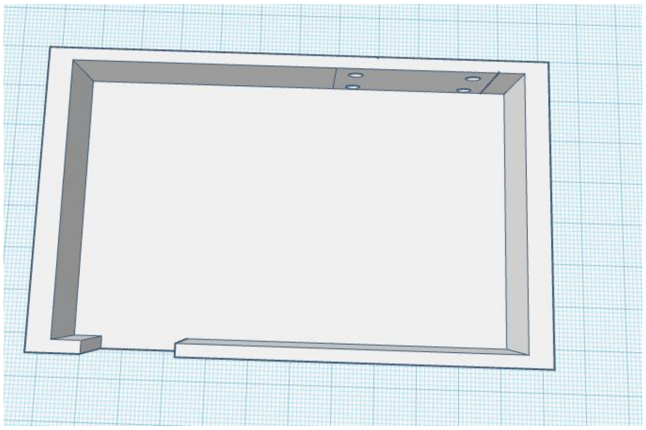




This piece will be attached to a servo motor and its main function will be to knock the Rubik's Cube, flipping it 90 degrees. It also features the Robot's name.

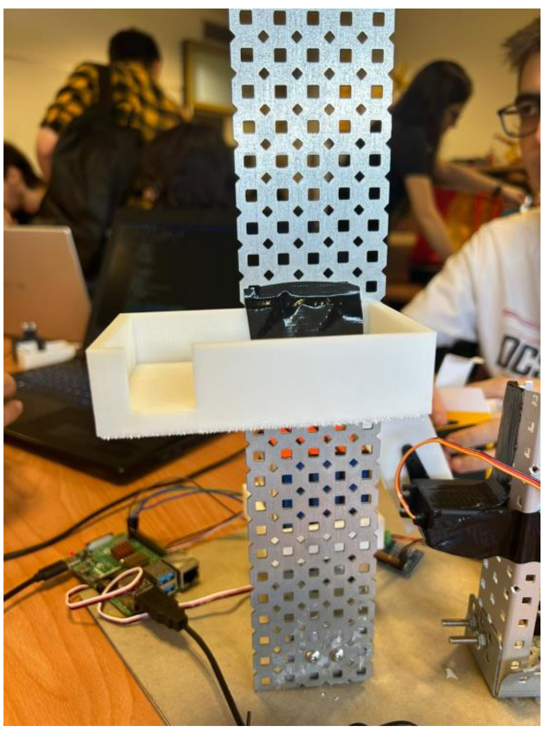
- *RASPY-HOLDER*





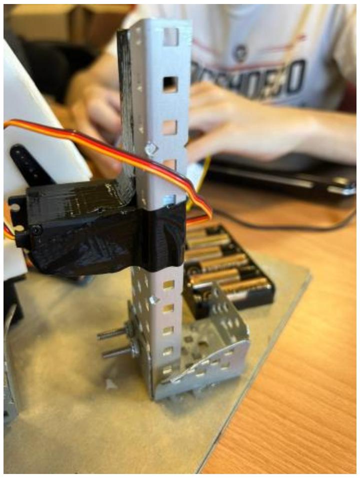
This piece will be attached to a support and will hold the Raspberry Pi 4 Model B. It is a necessary component because the camera cable is too short.

- *PIECE THAT HOLDS THE CAMERA AND THE RASPY-HOLDER*



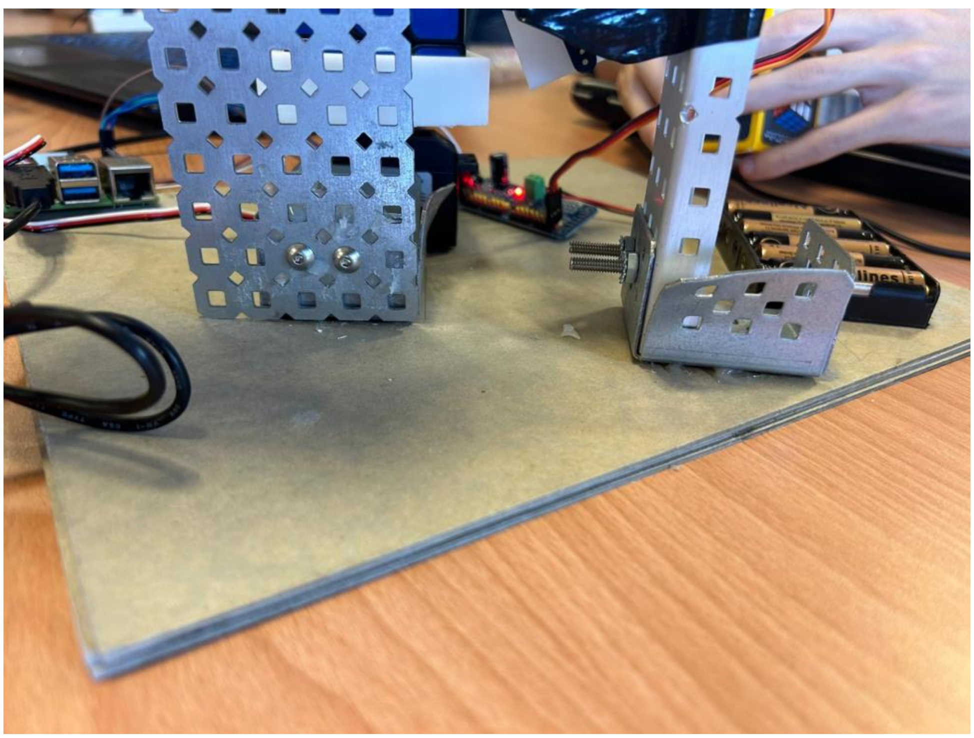
This piece is attached to the base, and holds the Raspy-Holder and the camera.

- *PIECE THAT HOLDS A SERVO*



This piece holds a servo motor and it is attached to the base too.

- *Robot base*



This is the base of the robot, its design makes it easy to transport and very light.

Strategy for validation, testing and simulation

First of all we will validate the computer vision part, because it's the key element of the project. It can be done by doing some recognition tests by checking the output that we got and what we expected.

Secondly, we have to test the solving algorithm. This can be done internally without requiring the robot by simulations in a computer.

Then we have to make a simulation of the robot in a simulation software like CoppeliaSim or something related.

We have to also test the replication pattern like the solving pattern.

Foreseen risks and contingency plan

Risk #	**Description**	**Probability** (High/Medium/Low)	**Impact** (High/Medium/Low**)**	**Contingency plan**
1	Accurately replicating the pattern of another cube may face challenges in the precision and alignment of movements to recreate the exact pattern	Medium	Medium	Use a precise calibration methodology to ensure the alignment and precision of the replication movements.
2	Complexity in Solving Algorithm Developmentin Solving Algorithm Development Designing and implementing an	High	High	Perform extensive tests on different cube configurations to identify and fix potential bugs in the algorithm

	efficient algorithm to solve the Rubik's Cube can be complex and require iterative testing and tuning			
3	The computer vision system may face challenges in accurately and reliably reading the cube configuration, including misidentification of colors or lighting issues.	Medium	High	Have a manual validation process to verify and correct the cube configuration in case of errors.
4	The CC motor, and de 360 servo may cause little errors that may lead to major errors solving or replicating the cube.	High	Low	We will conduct trial and error tests to minimize this persistent error. We need to ensure that it can perform numerous movements without any noticeable errors
5	The robot may encounter hardware malfunctions or failures, such as the camera malfunctioning or the motors experiencing issues.	Medium	High	Implement regular maintenance and testing procedures to identify and address potential hardware issues.
6	Environmental factors, such as changes in lighting conditions or	Medium	Medium	Conduct thorough testing in various environments.

	physical obstacles, may affect the robot's ability to accurately perceive and interact with the Rubik's Cube			

Updates Sprint 2

- All main 3D pieces have been printed.
- Solver algorithm has been implemented.
- Budget has been updated (we realized we had insufficient budget).
- PC design of the project with the main connections has been completed.
- Non-electronic components have been selected.
- GitHub has been created.

Updates Sprint 3

- Computer Vision algorithm has been created, but it has not been tested yet.
- All 3D pieces have been printed and are ready for testing.
- Code has been developed to test all hardware components.
- Raspberry Pi operating system has been installed.
- Code for testing components has been created.
- Hardware Structure diagram created.

Updates Sprint 5

- All hardware components tested.
- Tests to mark where the components must be.
- Unitary tests to determine if we need other components.
- The robot has been assembled.
- Test all connections and motors are tested separately.
- GitHub updates

- Software Architecture diagram created.
- All major components have been received.

Updates Sprint 6

- Robot has been assembled.
- GitHub has been updated.
- Starting with unitary tests.
- Change from Raspberry Pi Zero to a Raspberry Pi 4 Model B.
- Added a new 3D component to support the Raspberry and the camera.
- Requested a PCA9685 servo controller to control the movement.
- Tested a few movements from the solver algorithm.

Updates Sprint 7

- Main functionalities are now implemented.
- Final video recorded and edited.
- GitHub finished.
- README.md file written.
- Final presentation prepared.

References

Our GitHub project:

https://github.com/JPeaceK/RepliCube

This project has been inspired by the following Internet projects:

https://github.com/judith1304/Arduino-Rubiks-Cube-Solving-Robot

https://github.com/MohammadMouadi/Rubiks-Cube-Solver-Using-Arduino

https://projecthub.arduino.cc/hbolanos2001/6233e259-1629-458c-9f19-f8e9321a3060

https://rlpengineeringschooluab2018.wordpress.com/2018/05/30/rustic-rubiks-solver/