Cesar Wen
Francisco Kim Gaieski

Robô Para Solução do Cubo de Rubik

Brasil
2020

Cesar Wen
Francisco Kim Gaieski

Robô Para Solução do Cubo de Rubik

Escola Politécnica da Universidade de São Paulo - POLI USP
Departamento de Engenharia Mecatrônica e Sistemas Mecânicos - PMR
Trabalho de Conclusão de Curso

Orientador: Jun Okamoto Jr.

Brasil
2020

Resumo

O cubo de Rubik é um quebra-cabeça de extrema popularidade que atrai a atenção de públicos diversos, incluindo engenheiros. Robôs capazes de solucionar o quebra-cabeça vêm sido criados com base em três objetivos, a maior velocidade de solução, a maior acessibilidade na montagem do robô e a comercialidade do produto criado. A partir da análise de robôs já existentes, o trabalho visa a construção de um robô com capacidade de solucionar o cubo de Rubik de forma autônoma e se comunicar por meio de *Bluetooh Low Energy* com um aplicativo de *smartphone* personalizado. O aplicativo, por meio de visão computacional, captura o estado inicial do quebra-cabeça e, utilizando o algoritmo de solução de Kociemba, gera e aciona os movimentos que levem ao estado solucionado. A partir da união de cada seção do trabalho, o robô é capaz de solucionar o quebra-cabeça de forma autônoma, em média, em 1 minuto e 8 segundos.

Palavras-chaves: Robô; Robô solucionador; cubo de Rubik; *Bluetooh Low Energy*; visão computacional; OpenCV; Kociemba.

Abstract

The Rubik's cube is an extremely popular puzzle that attracts the attention of diverse audiences, including engineers. Robots capable of solving the puzzle have been created based on three objectives, the highest speed possible of solving the puzzle, the greatest accessibility in the assembly of the robot and the commerciality of the created product. Based on the analysis of existing robots, the project aims to build a robot with the ability to solve the Rubik's cube autonomously, capable of communicating through the protocol *Bluetooh Low Energy* with an *smartphone*. This application has the ability, through computer vision, to capture the initial state of the puzzle and using the Kociemba solution algorithm to generate and trigger the movements that lead to the solved state. From the joint work of each section, the robot is able to solve the puzzle autonomously, on average, in 1 minute and 8 seconds.

Key-words: Robot; Solving robot; Rubik's cube; OpenCV; *Bluetooh Low Energy*; computer vision; Kociemba.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BLE	*Bluetooth* de Baixo Consumo
CAD	Desenho Assistido por Computador
PCB	Placa de Circuito Impresso
PLA	Poliácido Láctico
PWM	Modulação por Largura de Pulso
RGB	Vermelho Verde Azul
HSV	Matriz Saturação Valor

Sumário

1 Introdução

O cubo de Rubik, conhecido também como cubo mágico, é um clássico quebra-cabeça inventado há mais de 40 anos pelo arquiteto húngaro Erno Rubik. Desde seu lançamento, o brinquedo ganhou notoriedade com a venda de mais de 450 milhões de unidades [14]. Devido à essa extrema popularidade, o cubo mágico tem sido do foco de não apenas passatempos, mas também de pesquisas científicas.

Principalmente matemáticos se viram trabalhando no Cubo de Rubik como um problema de otimização discreto, tentando encontrar maneiras eficientes de resolvê-lo [15]. Todos à procura do que é chamado de *God's Number* (Número de Deus) [16]. Tal termo representa o menor número de movimentos requeridos a fim de se resolver qualquer configuração do cubo mágico.

Engenheiros vêm construindo robôs capazes de solucionar um cubo de Rubik, aplicando os algoritmos encontrados pelos matemáticos em computadores. Assim, é possível a criação de um robô sofisticado que examina o cubo e gira as faces para sua solução sem a intervenção humana.

Esse trabalho consiste no desenvolvimento de um robô capaz de solucionar um cubo de Rubik de forma autônoma, incluindo uma interface com o usuário baseada num aplicativo para *smartphone*.

Para tal, são discutidas as diferentes soluções abordadas em literaturas disponíveis. Dentre elas, há aquelas referentes à velocidade, com o intuito de quebrar recordes de velocidade, à facilidade de construção, que são desenvolvidos com a mentalidade de fácil reprodução e aqueles que focam na fabricação de um produto comercial, focado em aparência e tamanho.

Para o desenvolvimento do robô proposto neste trabalho, o processo foi dividido em três partes, a mecânica, a eletrônica e a programação.

Com relação à mecânica, foi realizada uma pesquisa das diferentes configurações de robôs já feitos por trabalhos antecedentes. A partir de observações, foi realizado um projeto em Desenho Assistido por Computador (CAD), o qual foi possível realizar sua construção por impressão 3D.

O desenvolvimento da eletrônica consistiu na criação de um circuito elétrico o qual comunica o microcontrolador, responsável pela aquisição dos comandos vindos do aplicativo, aos atuadores.

Na parte de programação foi desenvolvido um programa capaz de controlar os movimentos necessários dos motores que correspondem às rotações do cubo de Rubik. As

rotações são comandos recebidos por *Bluetooth* [17] de um *smartphone* o qual o usuário opera. Foi desenvolvido um aplicativo de processamento de imagens para capturar o estado inicial das faces do cubo, utilizando o *OpenCV*[18]. Assim, a partir dessas faces iniciais, o algoritmo computacional de solução gera os movimentos necessários para a solução do quebra-cabeça.

Portanto, o projeto visa desenvolver um robô de aproximadamente $21cm$ de largura e comprimento, com capacidade de solucionar o cubo de Rubik de forma autônoma em um tempo menor que 5 minutos. Para tal, o usuário deve escanear o estado inicial do cubo com um celular, para que o aplicativo possa calcular o trajeto necessário a ser feito para chegar na forma resolvida do quebra-cabeça. Assim, o robô terá os movimentos necessários para rotacionar suas faces e enfim, solucioná-lo de forma autônoma.

O documento obedece a seguinte estrutura: a próxima seção traz uma contextualização da literatura que aborda as estratégias utilizadas para o desenvolvimento e construção de um robô solucionador de cubo de Rubik, assim como as abordagens técnicas relacionadas ao algoritmo de solução. A seção três aborda a descrição do robô proposto, assim como os requerimentos e funcionalidades do projeto. A seção quatro traz uma análise do *hardware* utilizado, apresentando os conceitos envolvidos na tomada de decisão referente ao microcontrolador e os motores escolhidos. A seção cinco traz uma breve descrição da estrutura mecânica e seu funcionamento, podendo então visualizar o mecanismo que movimenta o robô. A seção seis documenta a parte de programação desenvolvida no projeto. Nela está contida a explicação do funcionamento da comunicação entre microcontrolador e aplicativo, a lógica de movimentos dos motores, a visão computacional e o algoritmo de solução. A seção sete discute os resultados obtidos por meio das da integração das partes discutidas anteriormente. Assim, essa parte traz uma análise dos testes de viabilidade do projeto, para em seguida, observar se os objetivos iniciais foram atingidos. Finalmente, o projeto é concluído na seção oito.

2 Robôs para Resolver o Cubo de Rubik

Na atualidade, existe uma ampla gama de robôs que visam resolver o cubo de Rubik. Uma forma de categorizar tais máquinas é por meio da classificação de seus objetivos:

- **Quebrar recordes** - Máquinas com o objetivo de quebrar recordes visam ter a mais alta velocidade de resolução;
- **Acessibilidade** - Máquinas com o objetivo de reprodução de projeto visam estruturas de fácil obtenção e custos acessíveis;
- **Produtos comerciais** - Máquinas com o objetivo de venda priorizam a estética, assim como produtos de baixo custo a fim de atender um preço atrativo para a venda;

2.1 Velocidade

Entre as máquinas construídas para resolver o cubo de Rubik, destacam-se os que visam quebrar o recorde de velocidade. Assim, para conseguir resolver o cubo em tempos como 0.38s, atual recorde mundial [1], são necessárias otimizações de *hardware* e de *software*.

Com o intuito de maximizar a velocidade entre movimentos, cada face do cubo está conectada ao eixo de um motor de alta velocidade, dedicado apenas a rotacioná-la. Considerando esse objetivo, o atuador comumente escolhido por sua agilidade e precisão é o motor de passo [19] [2]. No entanto, o atual recordista de velocidade utiliza o motor DC *Kollmorgen ServoDisc U9/N9* com um *enconder* acoplado[1].

Sua fixação ao cubo também é especializada. São utilizados cubos especificamente feitos para o funcionamento em seu respectivo robô. O cubo pode ser usinado a fim de ser acoplado por pequenas garras que segure apenas a peça central [2]. Ou mesmo um cubo específico em que caiba um acoplamento sem modificações irreversíveis à face [19] [1].

A utilização de diversos motores e seus respectivos *drivers*, controladores e, em alguns casos iluminação [1][19], gera a necessidade de uma fonte de energia de alta capacidade [1].

Para a identificação do estado inicial, podem ser utilizadas duas câmeras, como o projeto da figura 1, ou mesmo quatro câmeras como o projeto de Albert [2]. Assim, todas as faces podem ser analisadas ao mesmo tempo sem necessidade de movimentação do cubo. Porém, a identificação das cores pode gerar dificuldades. No projeto atualmente recordista

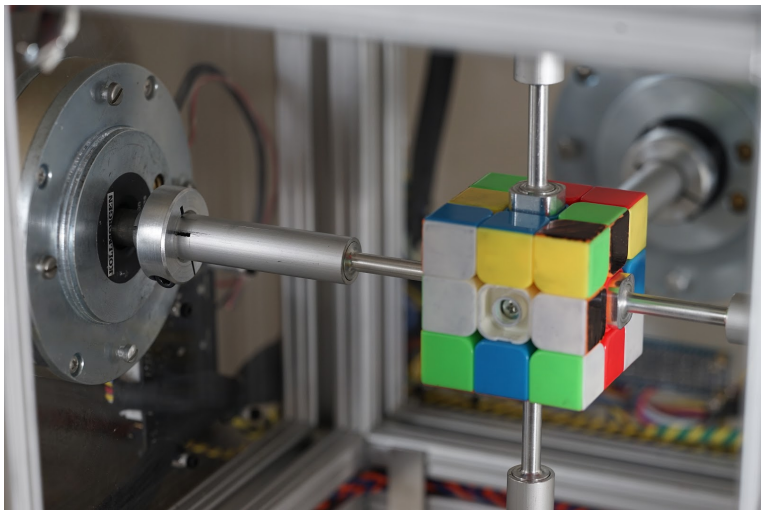


Figura 1 – Motor DC especializado e seu cubo específico com a face da peça central removida. Fonte: [1]

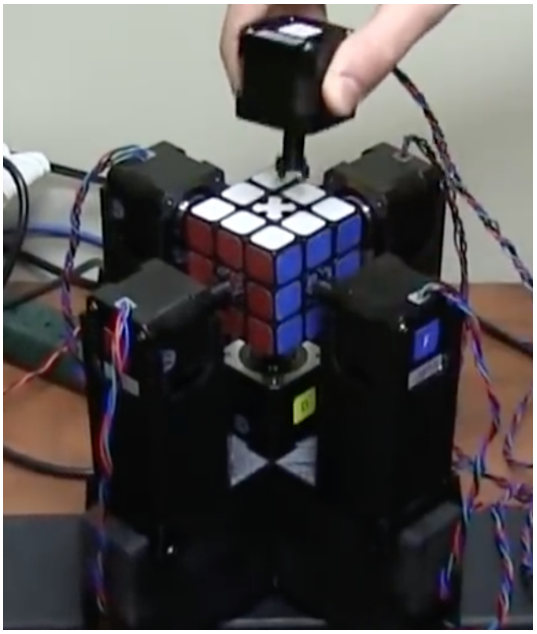


Figura 2 – Método de fixação e o respectivo cubo modificado. Fonte: [2]

[1], a distinção entre a cor vermelha e laranja se demonstrou um problema. Portanto, para contornar tal problema, as faces laranjas foram pintadas de preto para melhor contraste.

Por fim, o algoritmo popularmente utilizado é o de Herbert Kociemba [20]. Uma vez que, quando comparado a outros métodos populares, como os de Thistlethwaite e Korf, o algoritmo escolhido tem pequeno custo computacional [21] e uma boa eficiência. O algoritmo garante em torno de 20 movimentos para sua resolução, próximo ao *God's Number* [16].

2.2 Acessibilidade

Uma categoria de robôs para resolver o cubo de Rubik é aquela que prioriza a disseminação da acessibilidade.

Para que isso seja possível, os robôs desta classe são caracterizados por uma estrutura de fácil montagem com produtos de fácil acesso. Desse modo, há um espectro grande com diferentes tipos possíveis de robôs e seus componentes que o determinam.

Em um projeto caseiro [3], foram utilizados componentes de fácil acesso para que qualquer um pudesse reproduzir. Este robô em questão utiliza apenas dois servomotores como atuadores e palitos de picolé em conjunto com chapas de madeira compensada para sua estrutura mecânica. Com apenas dois servomotores, fica claro que a rapidez de solução do cubo não é uma prioridade. Para seu controle é utilizado um Arduino UNO, capaz de controlar tais servos, e um computador, que recebe o estado inicial do cubo manualmente e envia os comandos de solução para o microcontrolador.

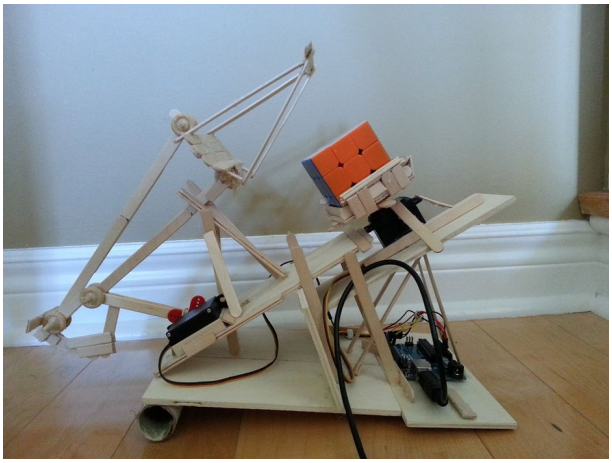


Figura 3 – Projeto caseiro utilizando madeira e dois servos. Fonte: [3]

Uma mecânica amplamente utilizada em projetos acessíveis para robôs que solucionam o cubo de Rubik é o produto LEGO Mindstorms. Por possuírem centenas de peças e componentes distintos em seu repertório, é possível diversas configurações diferentes para a construção deste tipo de robô. O mecanismo discutido previamente demonstra

uma montagem comum entre as possibilidades. Mais especificamente, há apenas dois motores para todas as movimentações do cubo [4] e [22]. Ou também é possível que o cubo seja segurado por quatro garras em suas faces laterais, totalizando oito motores, metade para o movimento de tais garras e a outra para o movimento rotativo das faces [5]. Para o controle, podem ser utilizados tanto os próprios controladores da LEGO [22], como também microcontroladores de terceiros, como o Raspberry Pi. Dada sua entrada comum USB, o Raspberry Pi possibilita uma maior variedade de opções de *hardware* para a detecção do estado inicial do cubo [22].

Figura 4 – Projeto em Mindstorms controlado por Raspberry Pie. Fonte: [4]

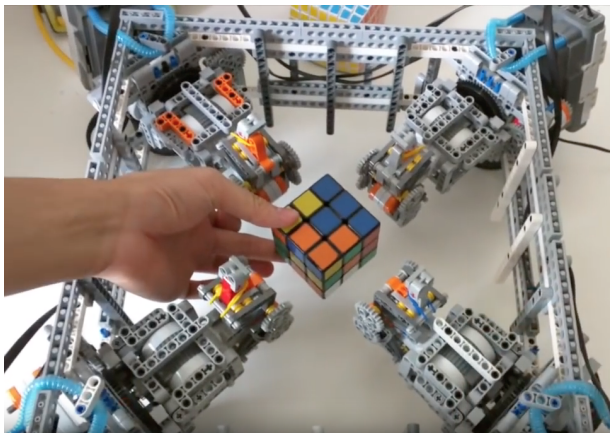


Figura 5 – Projeto de Minstorm utilizando 4 braços com garras. Fonte: [5]

No entanto, não é mutuamente exclusivo um projeto ser acessível e não possuir velocidade de solução como objetivo. O CubeStormer é um projeto feito a partir da base de LEGO Mindstorms que, previamente, era o robô solucionador de cubo de Rubik mais veloz que existia [6]. Diferente dos citados anteriormente, o CubeStormer atingiu um tempo de solução de 3.253s por meio do uso de um *smartphone* para a computação e aquisição de dados. O robô também utiliza oito motores, quatro sendo para movimentação de garras. Uma vez que possui braços menores, é capaz de realizar um movimento rotacional mais veloz.

Figura 6 – Antigo recordista de velocidade, CubeStormer. Fonte: [6]

2.3 Comercialidade

Pela própria natureza de produtos, os robôs comerciais visam ter além da capacidade de solucionar o quebra-cabeça, melhores acabamentos, funcionalidades extras, melhor estética e baixo custo.

O projeto OTVINTA [7] consiste em um robô solucionador do cubo de Rubik com uma estrutura completamente feita em impressão 3D. O robô opera com oito servomotores com um mecanismo similar ao *Cubestormer*, no qual metade dos servos rotacionam as faces laterais e os quatro servomotores restantes garantem o movimento linear dos efetuadores que se acoplam ao cubo. A utilização de servomotores garante um baixo custo de manufatura. No entanto, devido às suas limitações de velocidade, o robô não é focado para resoluções de alta rapidez. Por ser feito sob medida, é garantido um acabamento personalizado no qual todos seus fios, dos motores e fonte de alimentação, estão fora de vista.

Outro robô comercial é o GAN ROBOT da empresa GANCube [8]. O GAN ROBOT possui uma estrutura compacta de plástico, tendo a possibilidade de dobrar as estruturas que contêm os atuadores a fim de diminuir sua altura. Seu acionamento é composto de cinco servomotores para cinco faces do cubo. Está incluso em sua compra um cubo personalizado no qual os acoplamentos dos motores possam encaixar em sua peça central. A utilização de poucos servomotores garante o uso de uma pequena fonte de alimentação, além de um baixo custo de componentes.

Figura 7 – Robô da Otvinta. Fonte: [7]

Figura 8 – Robô da GANCube. Fonte: [8]

3 Sistema Proposto

O objetivo do projeto é a construção de uma máquina pequena que consiga manipular e resolver o cubo de Rubik por meio de comandos de um aplicativo de celular.

Sua estrutura foi restringida para um tamanho em torno de $210x297mm$. Isso foi estipulado a fim de ter uma máquina de fácil manuseio e transporte.

3.1 Alternativas de Implementação

Para a construção do robô, foram idealizadas possibilidades que envolvessem desde dois até oito atuadores. Alguns modelos foram descartados, como foi o caso de um modelo semelhante ao projeto Brickuber [4], discutido e ilustrado na figura 4. Isso devido à baixa velocidade de solução do sistema.

Como avaliação das máquinas, foram considerados 20 movimentos, de acordo com o *God Number*, distribuídos igualmente entre os seis lados do cubo. Assim, como não existem frações de movimentos, foram atribuídos quatro movimentos para cada face, totalizando 24.

Em seguida, são contados os números de movimentos necessários dos servomotores para que se resolva o quebra-cabeça. Para tanto, foi definido como um passo da solução a movimentação de 90^o junto a face, de soltar, de retornar os 90^o e por fim, voltar a segurar o cubo.

Para facilitar a comparação, cada robô foi classificado de acordo com o número de garras.

- **Seis Garras**

  Havendo uma garra conectada a cada face, não há necessidade de rotacionar o cubo como um todo para ter acesso à algum lado. Desse modo, o número de passos é simplesmente de 24.

- **Quatro Garras**

  Dado que há quatro garras, existem duas faces que estão sempre livres. No caso de tentar rotacionar uma destas faces, é necessário rotacionar o cubo por inteiro. Portanto, movimentos em faces que já estão conectadas equacionam apenas como um passo da solução, enquanto das duas faces livres, três. Portanto, totalizam-se 40 passos necessários para a solução.

- **Duas Garras**

No caso de duas garras, o cubo possui quatro faces livres. Com um raciocínio análogo ao caso de quatro garras, as duas faces adjacentes às faces conectadas terão três passos necessários para uma movimentação. Porém, as outras duas opostas necessitam de cinco. Assim, são totalizados 72 passos necessários para a solução.

Considerando apenas o número total de passos para a solução, seis garras se destacam. Porém, pelo fato de haver atuadores em todas as faces, é dificultado o manuseio do usuário para colocar e retirar o cubo. Outro ponto negativo é diminuir a visibilidade para o usuário, prejudicando a experiência. Como o esforço de 72, característico das duas garras, é muito grande, foi escolhido o mecanismo com quatro garras.

Deve ser analisado também o método com que o robô conseguirá segurar o cubo. Desta forma, foi feita uma categorização em 3 tipos, encaixe, pinça e customizado.

Para melhor avaliar os métodos, primeiramente é necessário saber o mínimo de recuo necessário. Considerando que o cubo de Rubik descreve uma circunferência perfeita em torno de seu eixo central, sabemos então que a garra deve recuar ao menos metade da diagonal da face ($D = 80.61mm$). Ou seja, a garra deve recuar para fora da circunferência de raio $40.30mm$.

- **Encaixe**

  Para o encaixe simples, a garra não possui partes móveis, desta maneira, seu recuo deve ser no mínimo metade da diagonal menos metade do lado ($\frac{D}{2} - \frac{L}{2} \approx 11mm$).

- **Pinça**

  No caso de operação da garra do tipo pinça, seria necessário um mecanismo capaz de realizar o movimento de abrir e fechar do acoplamento, como por exemplo, um bloco deslizante ligado às hastes da garra. Neste caso, o comprimento do recuo do bloco deslizante dependeria do ângulo de abertura da garra.

  Outra possibilidade é o arranjo de um pinhão e duas cremalheiras para as pinças se moverem linearmente. Como o recuo é nulo, este já deve ter a distância mínima.

- **Customizado**

  Uma possibilidade é ter algo customizado. Esse tipo de encaixe necessitaria de modificações no cubo de forma que seja possível fixar a garra apenas na peça central.

  O uso desse tipo encaixe customizado foi descartado devido ao fato de não aceitar um cubo qualquer, apenas aquele que for modificado.

O mecanismo da pinça, embora tenha a menor necessidade de recuo, a dimensão é compensada com o próprio mecanismo, o qual minimiza o ganho dimensional. Assim,

pelo fato da garra do tipo "encaixe" possuir maior simplicidade e não apresentar perdas dimensionais, foi a opção escolhida.

3.2 Configuração Final

Foi então construído em CAD o protótipo e impresso por impressão 3D.

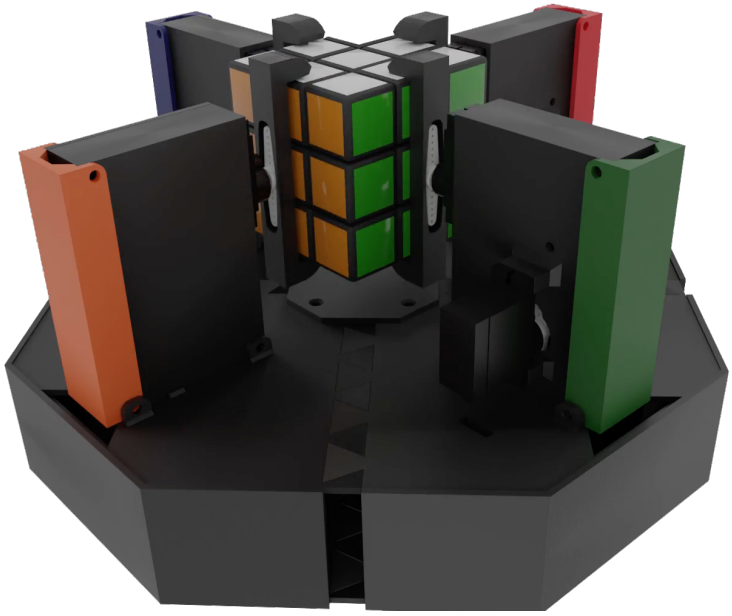


Figura 9 – *Render* do *Assembly* final do Robô em CAD. Fonte: Própria

Como é possível observar na figura 9, o robô consiste em quatro torres que abrigam dois servomotores cada. Enquanto um deles controla a garra que se prende à face do cubo a fim de rotacioná-la, o segundo é responsável pelo movimento linear do primeiro para liberar ou prender a face do cubo. Com oito servomotores no total, essa estrutura consegue, portanto, reproduzir todos os movimentos fundamentais do cubo de Rubik descritos no A.

Por baixo das torres, encontra-se o encapsulamento que é a base. Nela estão presentes os componentes eletrônicos que controlam e acionam os atuadores. Os comandos de movimentos dos motores são recebidos por meio de um sistema de comunicação sem fio [17] com um *smartphone*.

Para tal, foi desenvolvido um aplicativo dedicado para o *smartphone* com o sistema operacional iOS. O estado inicial do cubo é detectado por meio da câmera de alta qualidade já presente no celular. A partir desse estado inicial, é possível a resolução do mesmo por meio do algoritmo desenvolvido por Herbert Kociemba. algoritmo o qual também é processado pelo aplicativo, devido não só à maior capacidade de processamento do celular, resultando em uma resolução mais veloz, como também ao menor volume de dados a ser transmitido [20]. O aplicativo enviará então os comandos de movimentos necessários para o microcontrolador presente no robô. Também há a possibilidade de enviar comandos

manualmente, a fim de manipular o cubo sem a intervenção de um algoritmo de resolução. Para essas duas opções, o microcontrolador recebe do celular um *string* de caracteres que corresponde aos movimentos desejados.

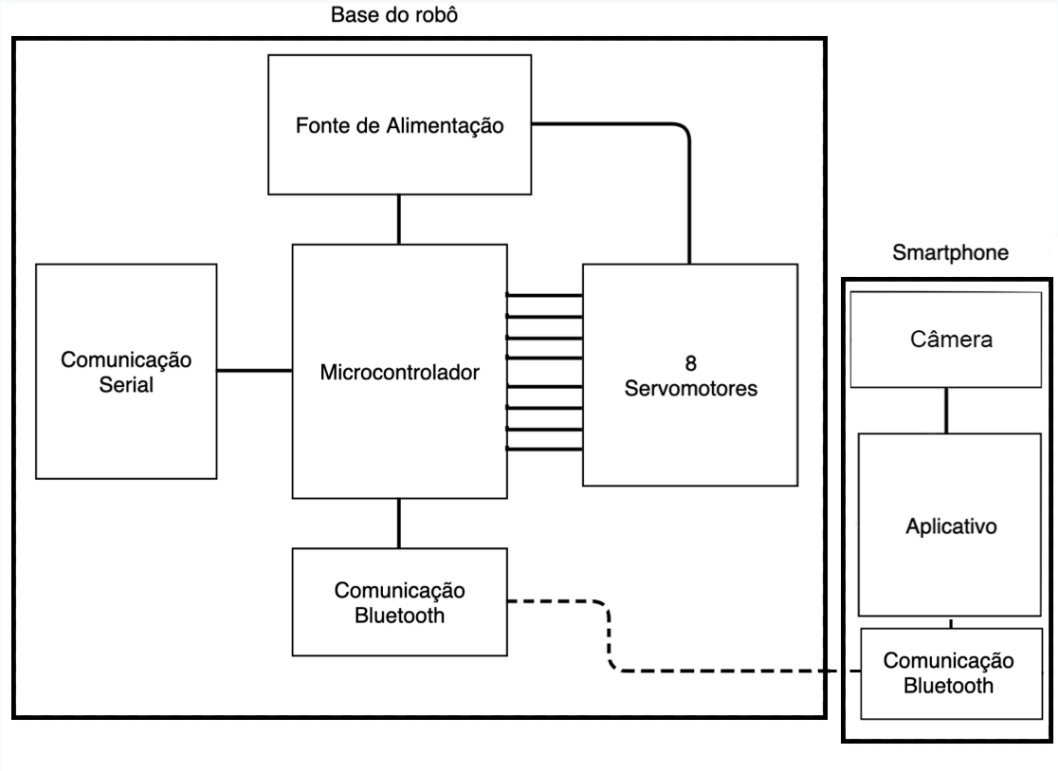


Figura 10 – Diagrama de blocos do sistema de controle. Fonte: Própria

4 Documentação Elétrica

O circuito elétrico do robô é composto por um microcontrolador, um conversor de nível de tensão e saídas para os oito atuadores. O microcontrolador é um ESP32. Programado por sua interface serial, o microcontrolador envia os sinais de Modulação por Largura de Pulso (PWM) de controle para os servomotores. No entanto, como esse sinal é de 3.3V de amplitude, é necessário um conversor de nível de tensão para 5V, requerido pelos servomotores.

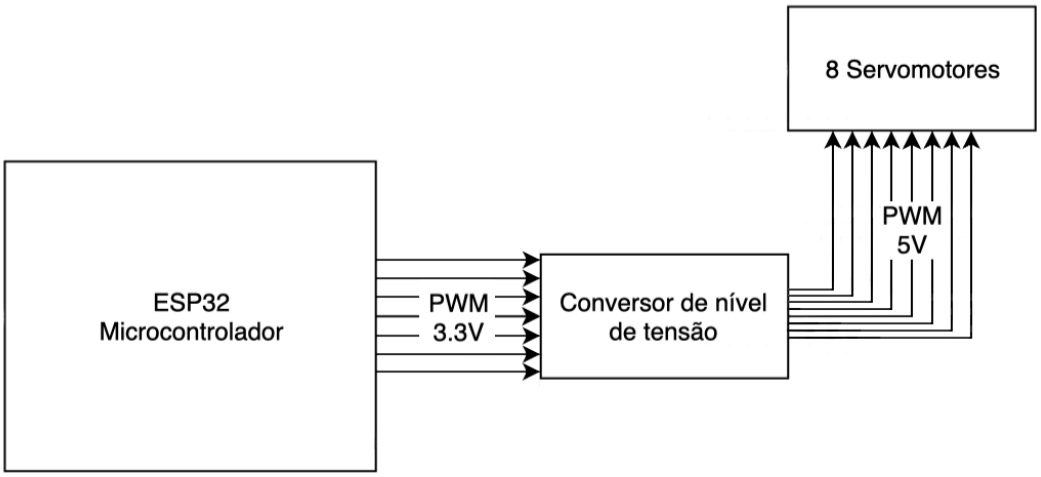


Figura 11 – Diagrama de blocos do circuito. Fonte: Própria

4.1 Componentes

Para a seleção de componentes, primeiramente foram escolhidos os atuadores e, a partir deles, o microcontrolador que atende às necessidades do projeto. Ou seja, aquele com a capacidade de controlar todos os oito atuadores, além de ser capaz de realizar a comunicação *Bluetooth* de Baixo Consumo (BLE).

4.1.1 Servomotor

Os parâmetros considerados na escolha do atuador foram o torque necessário para a rotação de uma face do cubo de Rubik, suas dimensões e sua disponibilidade.

Para a análise do torque necessário, foi desenvolvida uma peça a qual se prende a um dos lados cubo, cujo comprimento é $57mm$. Nesta mesma peça é possível adicionar

um recipiente no qual pesos são posicionados. Desta forma, uma vez que se sabe o peso adicionado, pode-se calcular o torque realizado por meio da fórmula 4.1.

$$T[kg.cm] = m[kg] * \frac{L_{Cubo}[cm]}{2} \tag{4.1}$$

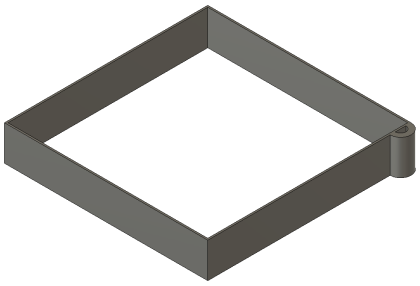


Figura 12 – Mecanismo desenvolvido para medição de torque. Fonte: Própria

A peça foi utilizada em 3 diferentes tipos de cubos, o original da marca Rubik, um desenvolvido para profissionais, cujo foco é resolver o cubo rapidamente, e um genérico de baixo custo.

Tipo de Cubo	**Teste 1 [g]**	**Teste 2 [g]**	**Teste 3 [g]**	**Torque Médio[kg.cm]**
Original	231	209	241	0.647
Profissional	66	101	88	0.251
Genérico	90	34	87	0.216

Tabela 1 – Medidas e Cálculo do Torque. Fonte: Própria

Como pode-se observar na tabela 1, o torque necessário para rotacionar a face do cubo original é muito maior do que os demais. Portanto, para que o robô consiga com sucesso manipular os 3 cubos testados, este dever ser capaz de gerar um torque maior que 0.647 kg.cm.

A partir disso, foi escolhido como atuador do robô o servomotor MG90S da *Tower Pro*. Além de possuir o torque necessário para rotacionar o cubo de Rubik [9], é um servo do tipo RC com PWM periódico de $20ms$ e largura de pulso entre 1 e 2 ms. Tal servomotor é um dos mais utilizados comercialmente sendo de fácil obtenção dentro do mercado brasileiro.

Especificações	
Peso	13.4 g
Dimensão	22.5x12x35.5 mm
Stall torque	1.8 kgf·cm (4.8V), 2.2 kgf·cm (6 V)
Velocidade de operação	0.1 s/60 deg (4.8 V), 0.08 s/60 deg (6 V)
Voltagem de operação	4.8 V - 6.0 V

Tabela 2 – Especificações do servomotor MG90S. Fonte: [9]

Figura 13 – Servomotor MG90S da Tower Pro. Fonte: [9]

4.1.2 Microcontrolador

Devido à comunicação *Bluetooth* entre o robô e o *smartphone* do usuário e a existência de sinais PWM necessários para o controle dos 8 servos, foi escolhido o ESP32 como microcontolador. Mais especificamente o módulo ESP-WROOM-32 na placa Devkit DOIT[10]. O ESP32 é um microcontrolador de baixo custo e baixo consumo de energia com Wi-Fi e, por requisito do projeto, *Bluetooth* integrado. O microcontrolador possui 16 canais diferentes para o uso de PWM, podendo então, controlar os oito servomotores sem a necessidade de um *driver* externo.

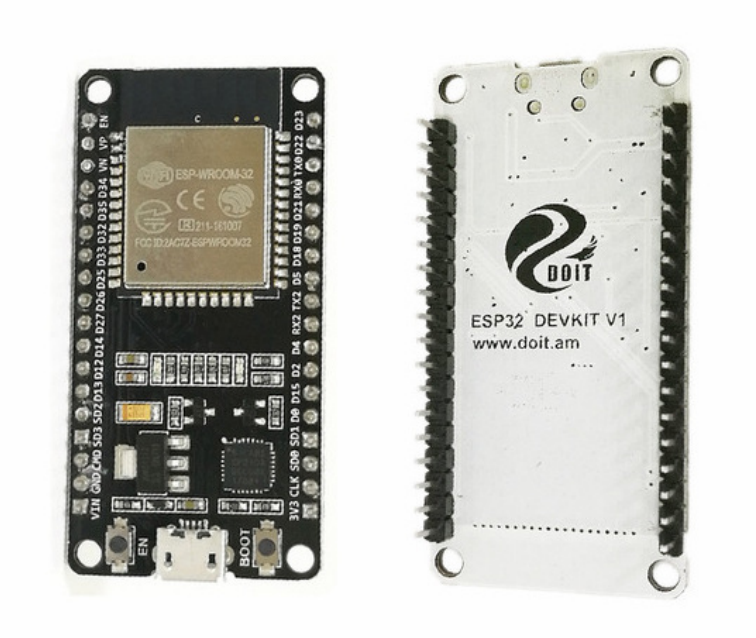


Figura 14 – DOIT ESP32 DevKit. Fonte: [10]

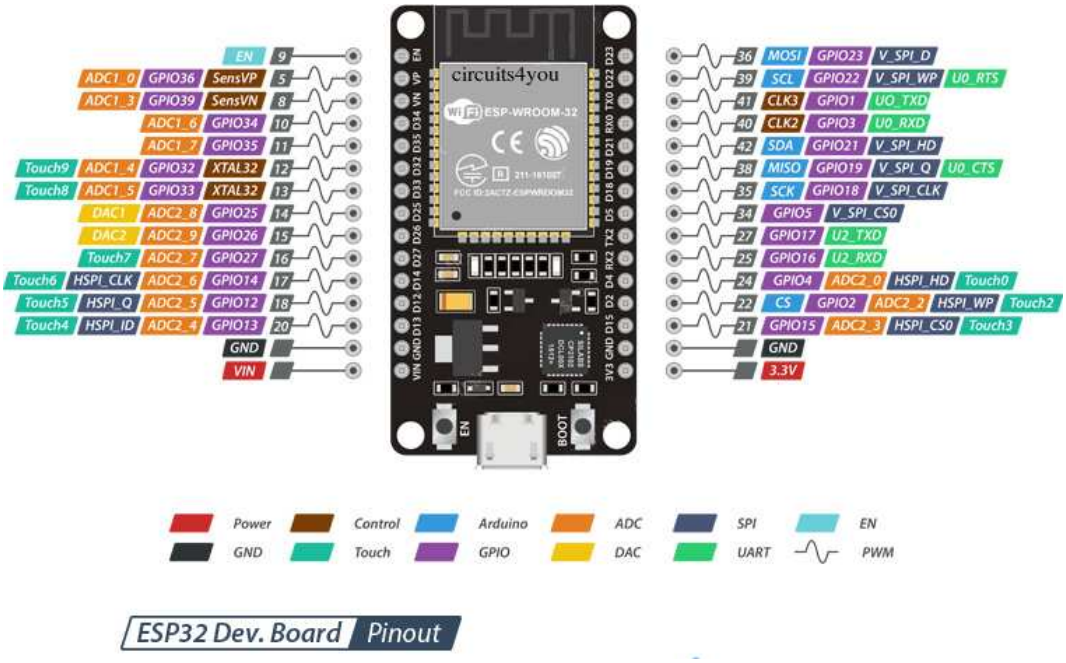


Figura 15 – DOIT ESP32 DevKit Pinout. Fonte: [11]

4.1.3 Compatibilização do ESP32 com os Servomotores

Como o módulo ESP32 possui apenas saídas de 3.3V e o servomotor MG90S opera em ao menos 4.8V, foi necessário a adição de um conversor de nível de tensão. Isto é, um circuito com o intuito de converter sinais de um nível lógico de tensão para outro, no caso, de 3.3V à 5V, sem a inversão do sinal. Foi então desenvolvido o circuito presente na figura 16.

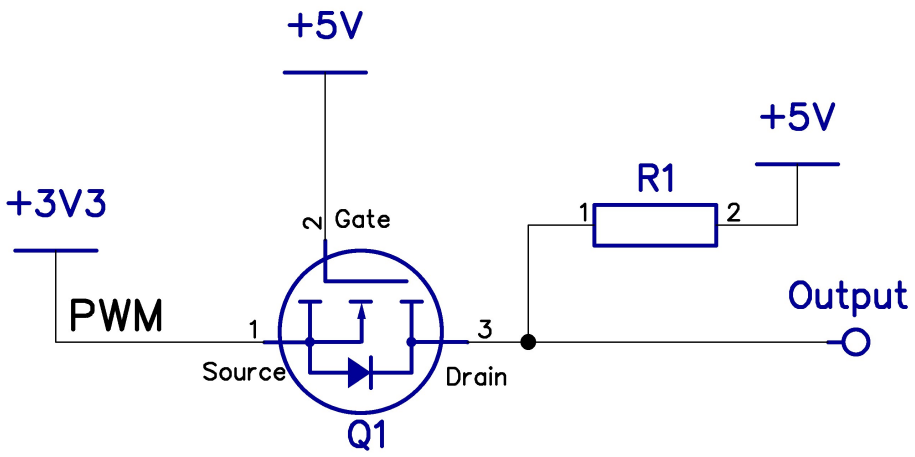


Figura 16 – *Level shifter*. Fonte: Própria

Como é possível observar, o *gate* está vinculado ao Vcc de 5V, advindo da fonte de tensão, enquanto o PWM, ao *source*. Quando o PWM vai para o estado *LOW*, o MOSFET conduz, levando o *drain* para *LOW* também. Quando o *source* transiciona para o estado *HIGH*, o MOSFET para de conduzir, deixando o *drain* em *HIGH*. Ou seja, o sinal do PWM conduz o MOSFET para que o *output* desejado fique com o mesmo *pulse width*, mas com a amplitude de 5V acoplada ao *drain*.

Para tal lógica, foi utilizado o MOSFET de canal N 2N7000, dado que o componente possui um *Gate Threshold Voltage* de 2.1V[23]. É possível ver sua aplicação no esquemático apresentado no apêndice B.

4.2 Construção Final e Testes

Para o teste dos componentes, foi impressa parte da estrutura final para não só a checagem das medidas, como o controle da movimentação requerida. Desse modo, foram desenvolvidos dois programas, um no ESP32 e outro no celular. Enquanto o celular envia apenas comandos pela comunicação BLE, o microcontrolador recebe e os interpreta para um movimento fundamental entre as duas estruturas impressas. Os possíveis movimentos são os lineares e de rotação.

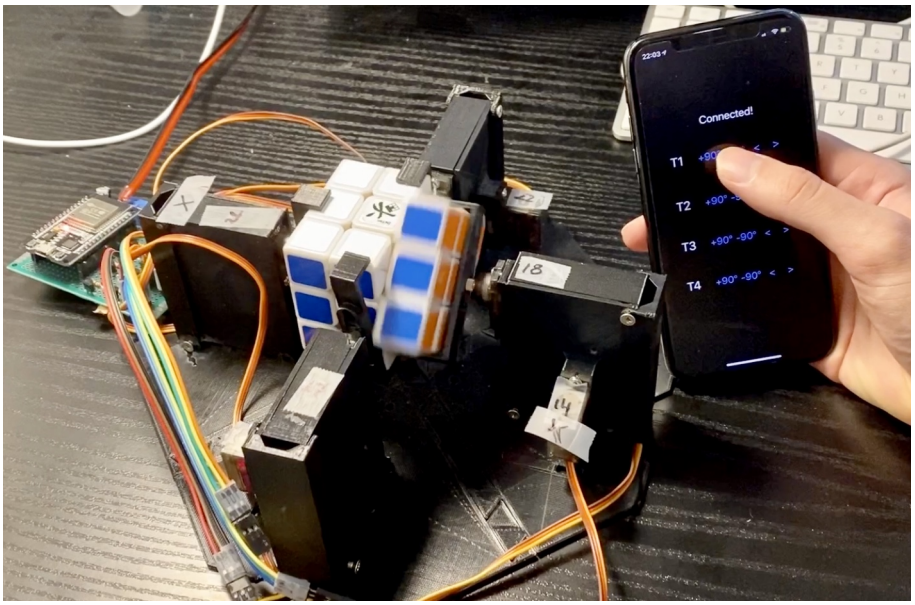


Figura 17 – Teste dos componentes em conjunto com o controle do celular por *Bluetooth*. Fonte: Própria

5 Documentação Mecânica

Uma vez escolhido o tipo de mecanismo desejado, foi desenvolvida a estrutura do robô.

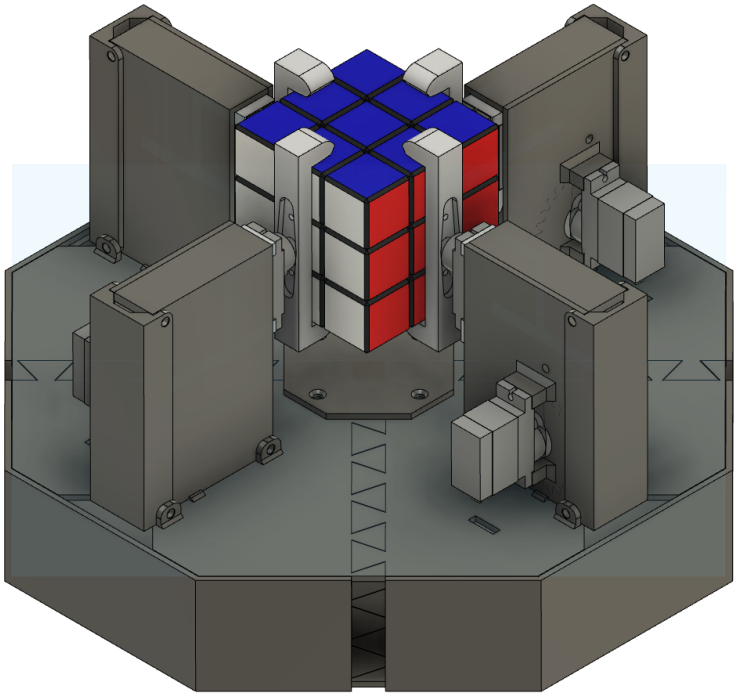


Figura 18 – *Assembly* do Robô em CAD. Fonte: Própria

Como é possível observar na figura 18, o mecanismo é composto por uma base, que contém a parte elétrica, figura 21, e quatro estruturas verticais, figura 20, denominada como torre. Cada torre encapsula dois servomotores com funções distintas.

A partir da figura 19 é possível observar tais funções. O sistema baseado numa cremalheira, ou seja, o servomotor acoplado à parede da estrutura rotaciona uma engrenagem. Essa rotação causa o movimento linear do outro servomotor acoplado à um trilho também dentado, gerando o ato de acoplar e desacoplar da garra ao cubo de Rubik. Isso garante que a face do cubo possa ser rotacionada em 360^o, a partir de movimentos de 90^o. Somado os movimentos das quatro torres, é possível garantir que todas as faces do cubo de Rubik possam sofrer rotações.

Com o *assembly* do CAD completo, foram construídas as peças a partir da tecnologia de impressão 3D. As peças fixas por meio de diversos parafusos criam o sistema final conforme a figura 22.

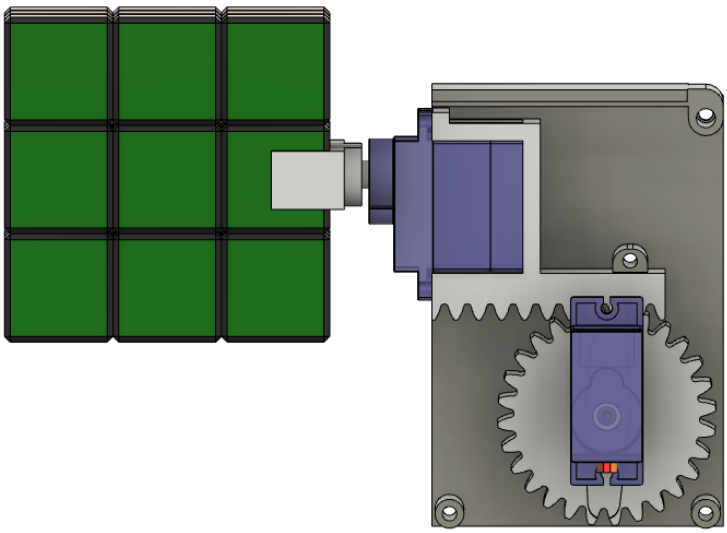


Figura 19 – Mecanismo da estrutura de torre. Fonte: Própria

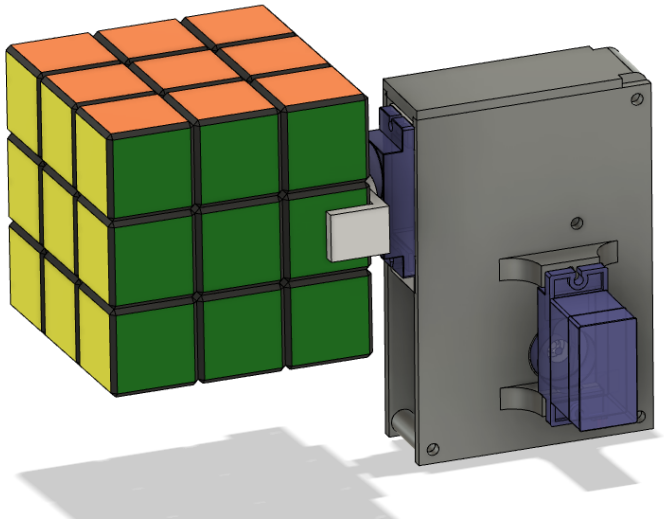


Figura 20 – *Assembly* da estrutura de torre do robô. Fonte: Própria

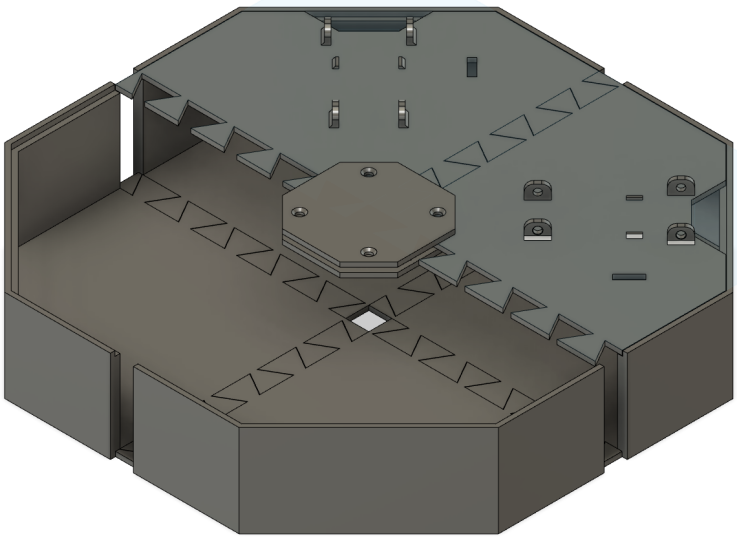


Figura 21 – *Assembly* da base do robô. Fonte: Própria

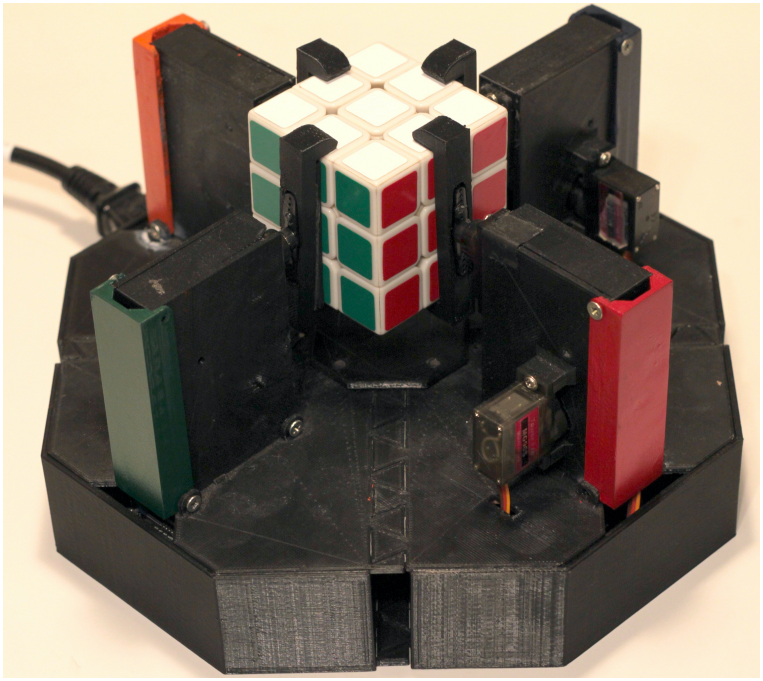


Figura 22 – Mecanismo impresso. Fonte: Própria

6 Documentação de *Software*

A figura 23 representa simplificadamente os objetivos do *software*. Isto é, um aplicativo capaz de enviar comandos de ao microcontrolador através de BLE. E o microcontrolador, por sua vez deve interpretar os comandos recebidos e convertê-los em sinais de comando para os servomotores.

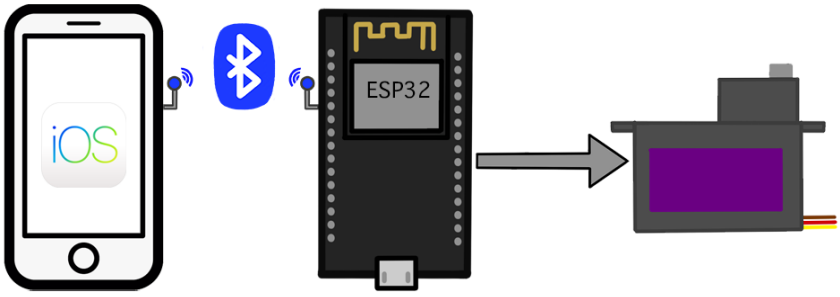


Figura 23 – Representação da comunicação entre *softwares* e *hardware*. Fonte: Própria

Conforme os diagramas de caso de uso apresentados nas figuras 24 e 25, o aplicativo apresenta duas funcionalidades, o controle manual e o modo de solução autônoma. O controle manual do cubo de Rubik consiste no usuário selecionar um entre 18 botões distintos, os quais correspondem a todas as rotações possíveis do quebra-cabeça. Os movimentos e sua nomenclatura estão detalhadas no apêndice A. Já o modo de solução autônoma, consiste no uso de visão computacional para aquisição de parâmetros do algoritmo de solução. Capturando fotos das faces do cubo, o celular é capaz de detectar as cores de cada peça do quebra-cabeça para que o algoritmo retorne os passos necessários de solução para a configuração em questão.

Os movimentos são feitos a partir dos movimentos básicos do robô, mover o atuador para frente, ou para trás, ou rotacionar a garra $+90^o$ ou -90^o. Feita a conexão *Bluetooth* entre o celular e o ESP32, o aplicativo envia o comando requerido pelo usuário. Uma vez recebido, o microcontrolador encaminha o sinal ao servomotor correspondente ao movimento desejado.

6.1 Comunicação BLE

Os dispositivos que trabalham com BLE podem ter duas funções diferentes em uma conexão: dispositivo central ou dispositivo periférico. O primeiro se trata de dispositivos que recebem dados. Já o segundo se refere aos dispositivos *low power* que se conectam ao dispositivo central. A estrutura pode ser entendida como a de um cliente e servidor, no qual o cliente recebe dados transmitidos do servidor [24]. Dada uma conexão, é feita uma

Figura 24 – Diagrama de casos de uso para o aplicativo. Fonte: Própria

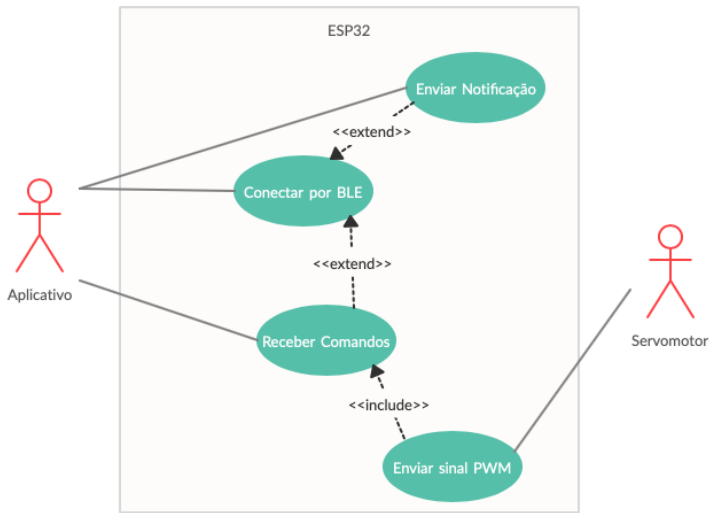


Figura 25 – Diagrama de casos de uso para o ESP32. Fonte: Própria

estrutura de dados hierárquica, conhecida como GATT (*Generic Attribute Profile*), que define o modo que dois dispositivos recebem e transmitem dados. Para o nível mais alto se encontra o *Profile*, que é composto por um ou mais serviços necessários para atender a um caso de uso. Um serviço é composto de características. Uma característica é um valor usado em um serviço, juntamente com propriedades e informações de configuração sobre como o valor é acessado e informações sobre como o valor é exibido ou representado [12]. No caso, é utilizada as propriedades *notify* e *write* das características.

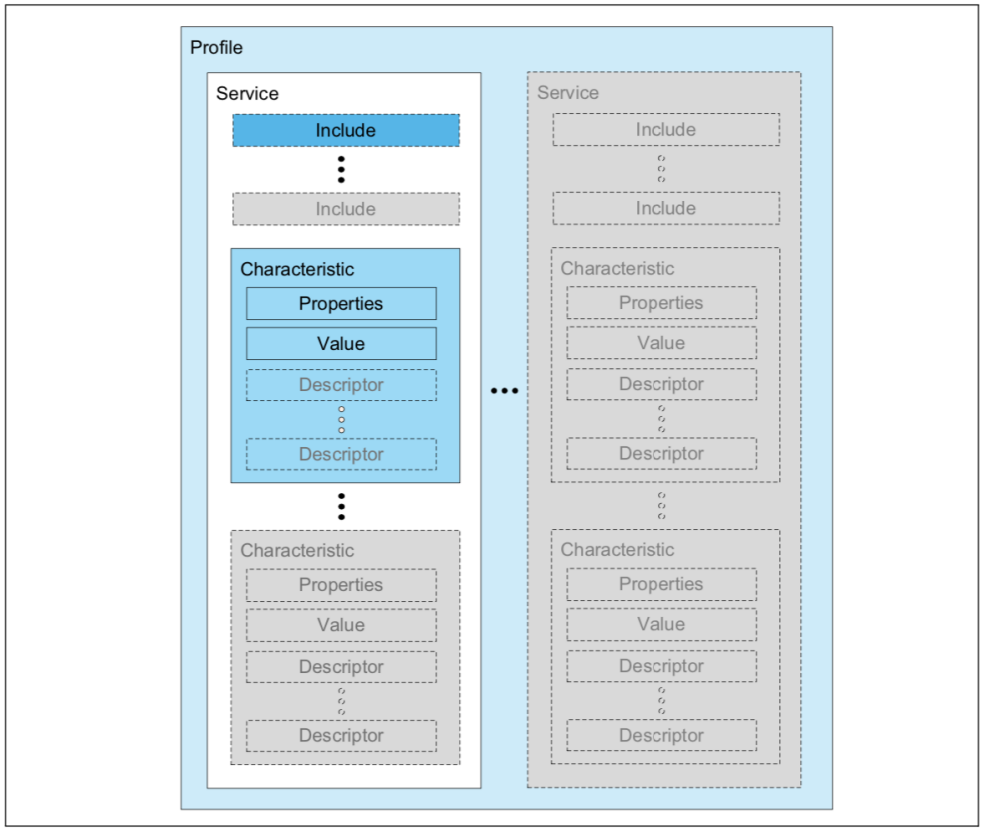


Figura 26 – Hierarquia de perfis baseados em GATT. Fonte: [12]

Assim, o ESP32 no sistema possui a função de servidor e o aplicativo como cliente. A partir do serviço criado no microcontrolador, é feita uma característica de *notify* para o envio de notificações ao aplicativo e uma de *write* para o recebimento dos comandos. O celular então consegue reconhecer o ESP32 como um dispositivo periférico e então utilizar essas características para envio e recebimento de dados.

6.2 Modos de Operação do Aplicativo

Como anteriormente descrito, o aplicativo possui dois modos de funcionamento. O primeiro consiste no modo manual. Tal modo dispõe ao usuário todas as três movimentações para cada uma das seis faces do cubo, totalizando 18 possibilidades (figura 27).

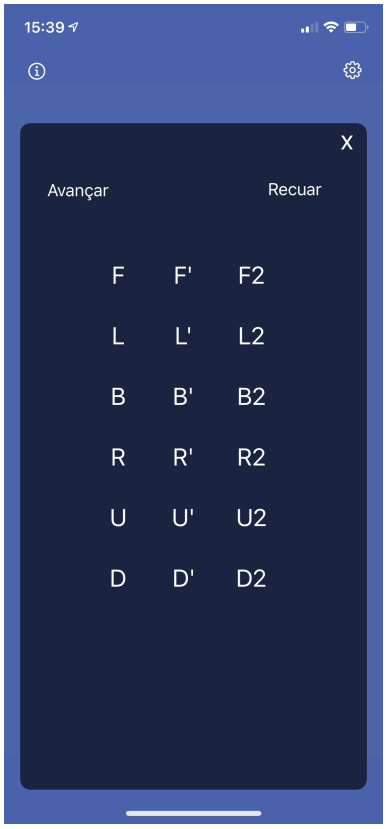


Figura 27 – Tela do aplicativo desenvolvido para o controle manual. Fonte: Própria

O segundo, consiste no modo de solução autônoma. Neste modo, o usuário é instruído a adquirir as fotos do estado inicial do quebra-cabeça, de modo que o aplicativo compreenda todas as posições relativas das peças (figura 28).

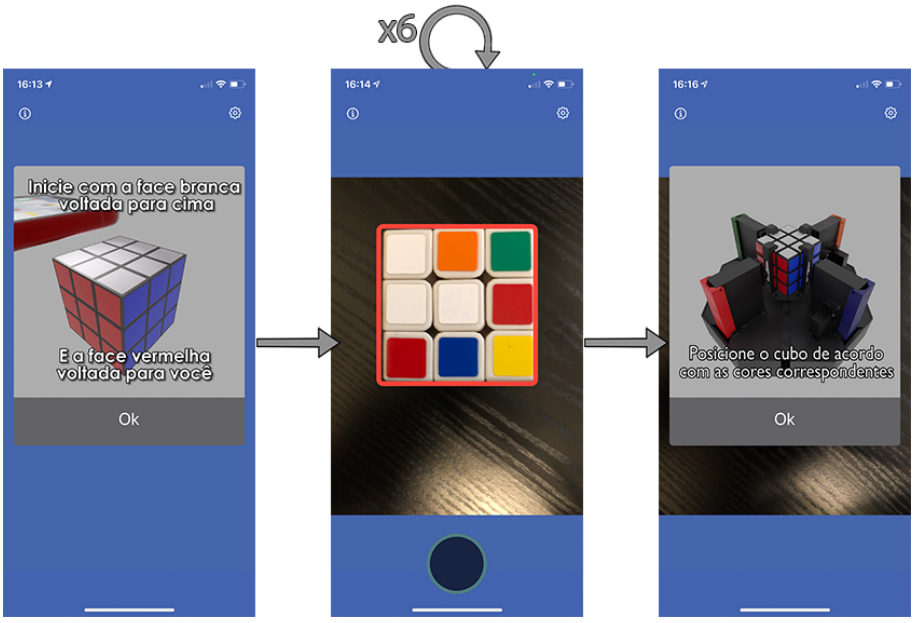


Figura 28 – Telas do aplicativo desenvolvido para a solução autônoma. Fonte: Própria

Tais fotos do estado inicial são interpretadas por um algoritmo de visão computaci-

onal que, por sua vez, possibilita a geração da solução a partir do algoritmo de Herbert Kociemba. Ambos modos de operação serão detalhados nas seções seguintes.

6.2.1 Visão Computacional

Para o processamento das imagens e, portanto, a aquisição das cores das peças do cubo, é necessária a aplicação de visão computacional. Foi utilizada então a biblioteca *OpenCV* [18].

Primeiramente, para que todas as imagens tenham um tamanho padrão, um *layout* (figura 29) foi desenvolvido. Desta forma, contanto que o cubo esteja dentro das margens, suas dimensões em *pixels* estão dentro de um padrão esperado.

Figura 29 – Tela do aplicativo desenvolvido para o dimensionamento do cubo. Fonte: Própria

6.2.1.1 Pré-Processamento da Imagem

Para que a imagem seja adequadamente analisada, é necessário realizar primeiramente uma etapa de tratamento. Tal etapa consiste em redimensionar a imagem para que tenha uma largura de 600 *pixels* e uma altura que siga a mesma proporção da imagem original. A redimensão traz duas vantagens. A primeira, permite que independentemente da resolução da câmera da qual foi capturada a foto, o quebra-cabeça apresente sempre as mesmas dimensões aproximadas. A segunda, acelera o processamento do algoritmo uma vez que a resolução final é menor.

Em seguida, é aplicada uma máscara para cada cor que compõe o cubo de Rubik. Tal máscara ajuda na consistência do algoritmo, uma vez que este elimina da imagem

possíveis geometrias também contidas na foto que pudessem ser confundidas como um elemento do quebra-cabeça. Em conjunto com a máscara, é realizada uma conversão do espaço de cor RGB para o HSV, dado que este é o mais consistente sob diversas condições de iluminação [25].

Então, é necessário borrar levemente a imagem. Tal tratamento auxilia na remoção de ruídos [26], permitindo que as cores sejam detectadas com maior precisão.

Para o processamento da imagem, é utilizado o algoritmo *"Canny"* disponível na biblioteca do *OpenCV*. Esse algoritmo se baseia em quatro etapas, começando pela redução de ruído, passando a encontrar o gradiente de intensidade da figura, realizando então supressão de não máximos e finalizando com uma histerese de limites [27]. Isso gera uma imagem com contornos que permitem a detecção das peças individuais.

Finalmente, é realizada uma operação de dilatação morfológica no resultado do algoritmo *"Canny"*. Tal processo ajuda a remover pontos de contorno fora da alvo desejado [26]. Ao final do pré-processamento, o resultado obtido, com exceção da alteração de resolução, pode ser observado na figura 30.

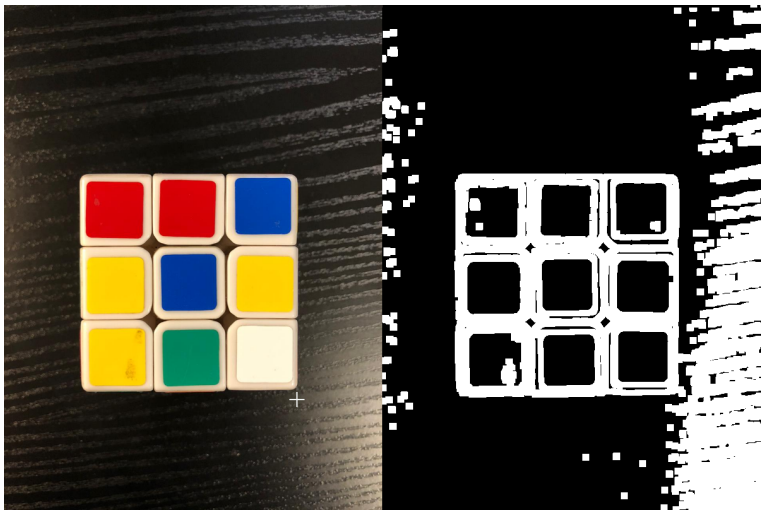


Figura 30 – Imagens antes e depois do pré-processamento. Fonte: Própria

6.2.1.2 Detecção das Peças e suas Cores

Uma vez terminado o pré-processamento, inicia-se então a detecção das peças do quebra-cabeça. É primeiramente utilizada a função `findContours` disponível pelo *OpenCV*. São então obtidos todos os contornos encontrados pelo método em um vetor.

Para cada contorno, são aplicados três filtros. Primeiro, a proporção entre os lados deve ser sempre entre 0.8 e 1.2 para que a forma seja próxima de um quadrado. O segundo e o terceiro são limites de área e perímetro. Assim, são identificados os contornos de tamanho próximo ao de uma peça.

Encontrados os contornos desejados, sabe-se que em seu interior está a cor desejada. Assim, basta entender onde estão localizadas as coordenadas das cores dentro do campo HSV (figura 31).

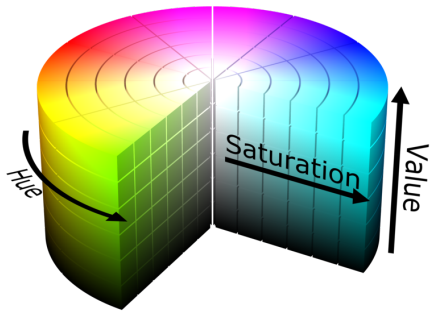


Figura 31 – Campo de cores HSV. Fonte: [13]

6.2.2 Algoritmo de Solução

Dentre as 4.3 **x** 10^{19} permutações possíveis do cubo de Rubik, apenas uma configuração é a a correta. Assim, desenvolver uma solução aplicável a todos esses estados não é algo trivial. Apesar disso, há diversas maneiras de solucioná-lo. Os métodos podem ser divididos em duas categorias, os algoritmos humanos e os computacionais. Os humanos já são bem conhecidos e sabidos pela comunidade entusiasta, que podem também ser divido em métodos para iniciantes, para rapidez e para aqueles de olhos vendados.

Os computacionais, por outro lado, são métodos baseados em algoritmos que iteram diversos caminhos possíveis para chegar à solução. Por sua alta taxa de processamento, são soluções que comumente apresentam menos movimentos necessários para solucionar o quebra-cabeça.

Os principais algoritmos utilizam uma aplicação da teoria matemática dos grupos, em particular, aqueles que têm uma estrutura comutativa. Por haver subgrupos na solução, o algoritmo é feito a partir de vários "níveis de dificuldade" independentes. Por exemplo, um desses subgrupos poderia envolver a solução de cubos que foram embaralhados usando apenas voltas de 180 graus. Esse princípio é o que consta no algoritmo de Kociemba [28].

6.2.2.1 Algoritmo de Duas Fases

Herbert Kociemba no começo da década de 90 criou sua versão do algoritmo computacional para a solução do cubo de Rubik, hoje o algoritmo é popular entre os robôs solucionadores [29].

De acordo com as notações de movimentos do cubo do Apêndice A, dado um cubo resolvido, se não utilizar os movimentos R, R', L, L', F, F', B e B', o algoritmo afirma que será gerado apenas um subconjunto de todos os cubos possíveis, sendo denotado por

$G1 =< U, D, R2, L2, F2, B2 >$. Neste subconjunto, as orientações dos cantos e bordas não podem ser alteradas. Ou seja, a orientação de uma aresta ou canto em um determinado local é sempre a mesma.

A partir disso, o algoritmo é dividido em duas partes, denominado de *Two Phases* (duas fases). Na fase 1, procura-se por movimentos que irão transformar um cubo embaralhado em G1. Ou seja, as orientações dos cantos e bordas devem ser restringidas e as bordas da fatia UD devem ser transferidas para essa fatia. Para encontrar esse estado de objetivo, o algoritmo realiza uma pesquisa heurística $h1$. Uma vez que G1 é um subconjunto de todos os embaralhamentos possíveis, é mais rápido de chegar do que ir de um cubo embaralhado para um resolvido em uma única etapa. Assim, a função heurística é uma tabela de pesquisa baseada em memória e permite a remoção de até 12 movimentos com antecedência.

Na segunda fase, o programa restaura o cubo no subgrupo G1 utilizando apenas movimentos deste subgrupo. Restaura a permutação dos 8 cantos, a permutação das 8 arestas das faces U e D e a permutação das 4 arestas das fatias UD. A função heurística $h2$ estima o número de movimentos que são necessários para atingir o estado objetivo porque existem muitos elementos diferentes em G1. Portanto, uma estimativa é feita no número de movimentos necessários para a Fase 2. O algoritmo continua a encontrar soluções cada vez mais curtas usando alguns valores de Fase 1 subóptimo que produzem valores de Fase 2 mais ótimos.

O algoritmo não para quando uma primeira solução é encontrada, segue a procurar soluções mais curtas, realizando a fase 2 a partir de soluções sub-ótimas da fase 1. Por exemplo, se a primeira solução tem 10 movimentos na fase 1 seguidos por 12 movimentos na fase 2, a segunda solução poderia ter 11 movimentos na fase 1 e apenas 5 movimentos na fase 2. O comprimento das manobras da fase 1 aumenta e o comprimento das manobras da fase 2 diminui. Se o comprimento da fase 2 chegar a zero, a solução é ótima e o programa pára [30].

6.3 Movimento dos Servomotores

Como citado previamente no documento, os servomotores são movimentados em ângulos de 90^o. Para que o robô seja capaz de realizar todos os movimentos necessários para resolver o cubo de Rubik, cada servomotor deve se mover de forma ordenada. Por exemplo, se o movimento for L, a garra à esquerda rotacionará 90^o, para que em seguida o servomotor interno mova a cremalheira de forma a recuar a garra do cubo. Esse recuo possibilita que o servomotor da garra possa retornar à sua posição inicial e a movimentação é terminada com o servomotor interno movendo novamente a fim de avançar a garra ao cubo.

Os movimentos são enviados a partir do aplicativo, o qual pode representar um único comando no caso do modo de controle manual, ou uma série de comandos ordenados no caso do modo de solução automática. Estes comandos são então processados pelo microcontrolador e enviados aos servomotores correspondentes a cada movimento.

7 Resultados e Discussões

Uma vez que as três divisões do projeto estão funcionando em conjunto, foram feitos testes de suas funcionalidades de forma a verificar se o objetivo inicial do projeto foi cumprido. São também discutidas as dificuldades encontradas durante os experimentos de forma a apresentar as suas respectivas soluções.

7.1 Objetivos Atingidos

O trabalho tem como objetivo a construção de um robô capaz de solucionar o cubo de Rubik de forma autônoma que utiliza um *smartphone* como controle remoto. Além disso, o esperado era que o robô conseguisse solucionar o quebra-cabeça em um tempo máximo de 5 minutos com dimensões máximas de aproximadamente $21cm$ de largura e comprimento.

O robô construído possui $21cm$ de largura e comprimento, e é capaz de utilizar o *smartphone* para processar o estado inicial do cubo por meio de reconhecimento de imagens por visão computacional. A partir das imagens, aplica-se o algoritmo de solução de Kociemba. Tudo implementado sob uma interface intuitiva ao usuário e capaz de enviar os dados de solução ao microcontrolador.

Entre os robôs já desenvolvidos, especialmente os classificados como de acessibilidade, há uma abundância de metodologias em que o usuário introduz manualmente o estado inicial do quebra-cabeça. Esse foi um problema solucionado a partir do uso de *smartphone*, visto que sua presença é comum nos dias de hoje.

Quando comparado aos robôs classificados como de velocidade, o robô não é capaz de solucionar o cubo de Rubik em tempos comparativos. No entanto, além de não ter sido o foco do projeto, o robô realiza o proposto muito abaixo do objetivo inicial de 5 minutos, além de também possuir outras funcionalidades, como o de controle manual.

Assim, o robô desenvolvido no projeto pode ser comparado aos robôs referentes à parte de comercialidade. Foi realizada a construção de um robô com relevância em estética e componentes de baixo custo, além da possibilidade comercial dos arquivos de sua estrutura e do *software* como o aplicativo que o compõe.

Utilizando então essa interface em conjunto aos componentes do robô, o sistema proposto é capaz de solucionar o quebra-cabeça em um tempo médio de 1:08 minuto.

Desse modo, todos os objetivos foram atingidos.

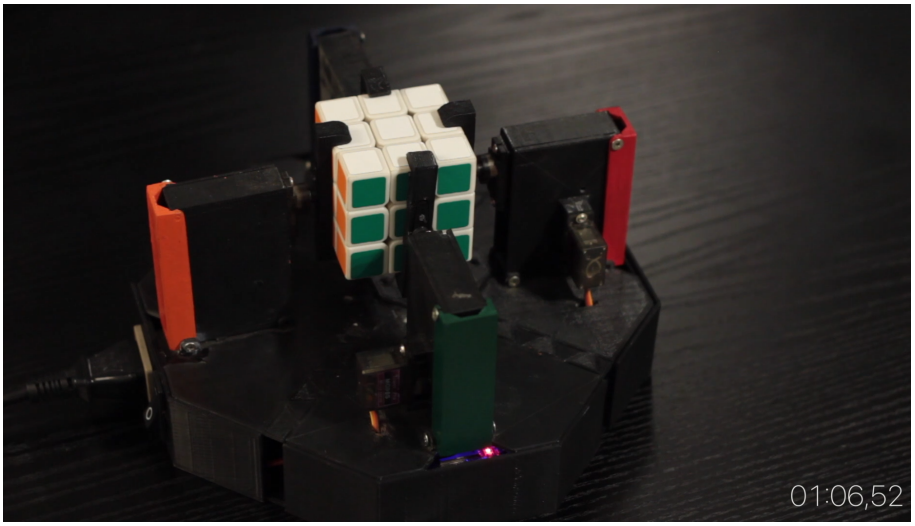


Figura 32 – Cubo resolvido com tempo total de 1:06 minuto de uma solução autônoma. Fonte: Própria

7.2 Análise de Soluções

7.2.1 Ajuste do Movimento dos Servomotores

Por meio do uso do controle manual do sistema, apresentado na figura 27, foi observado que os servos, apesar de alta precisão, possuem uma baixa acurácia.

De forma a confirmar o observado, foram realizadas coletas de dados para uma análise de precisão e acurácia. Para tal, quatro diferentes servomotores foram rotacionados em 90^o ida e volta de forma a anotar o erro a cada rotação feita. Vale ressaltar que os ângulos coletados possuem imprecisão devido à falta de instrumentos capazes de medir pequenos ângulos. No entanto a figura 33 já demonstra a falta de acurácia do atuadores.

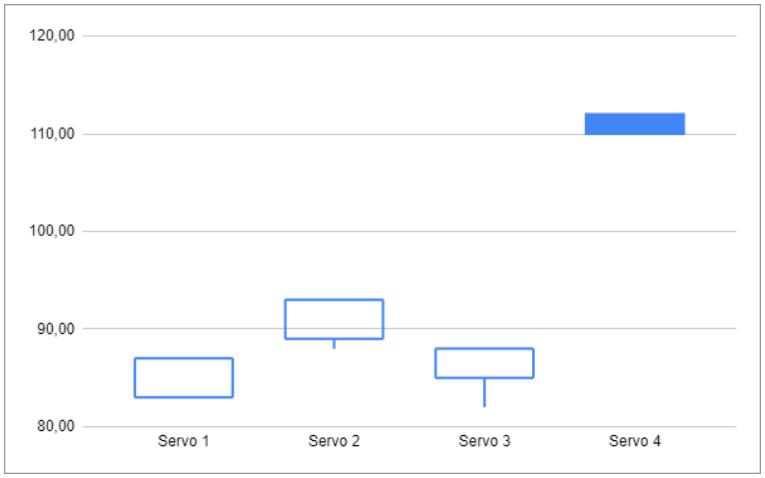


Figura 33 – Dispersão dos testes de acurácia dos servomotores, valor esperado de 90^o. Fonte: Própria

É possível notar que o observado foi confirmado com a coleta de dados. Cada

servomotor possui a precisão esperada quando enviado o sinal de rotação de 90^o, ou seja, o atuador rotaciona consistentemente o mesmo ângulo para o mesmo sinal. No entanto, quando acoplado ao cubo de Rubik, cada servomotor garante um erro consistente à esses 90^o, o qual um servomotor garante que sempre atingirá, por exemplo, 95^o, enquanto outro, 86^o.

Esses erros, quando ocorridos durante uma solução autônoma com diversas rotações seguidas em sequência, são capazes de incapacitar o robô. A figura 34 ilustra uma rotação em que o servomotor ultrapassa o ângulo de 90^o enviado pelo microcontrolador. Esse erro já seria o suficiente para impedir a rotação seguinte das faces adjacentes do cubo, anulando a possibilidade de uma solução autônoma.

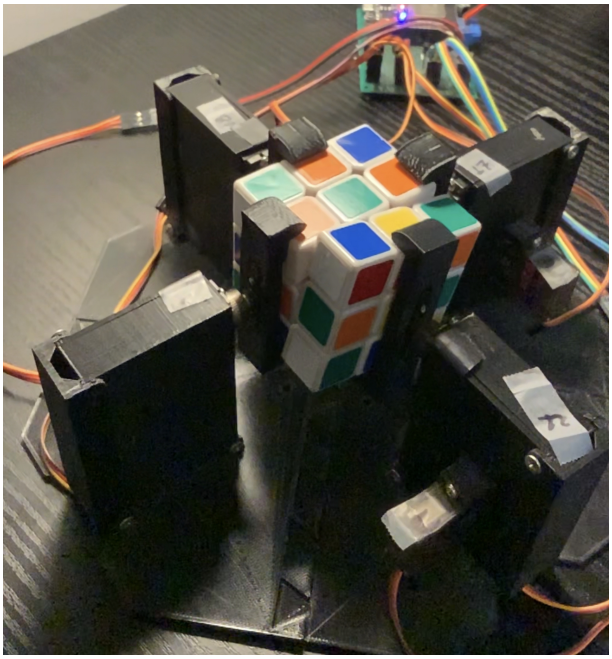


Figura 34 – Resultado de uma rotação de 90^o de um servomotor de baixa acurácia. Fonte: Própria

Foi observado que, ao mandar um sinal de ângulo do erro, somado aos 90^o desejados, o servo rotaciona a face do cubo de Rubik como esperado. Desse modo, é necessário uma compensação de ângulos para cada servo a fim de fazê-los rotacionar o ângulo esperado. Para tal, foi desenvolvida uma interface no aplicativo capaz de testar e armazenar essas compensações a fim de enviá-las ao microcontrolador quando uma movimentação for acionada.

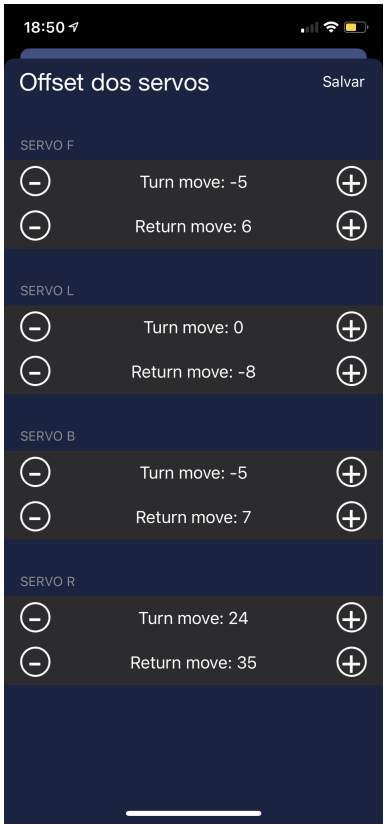


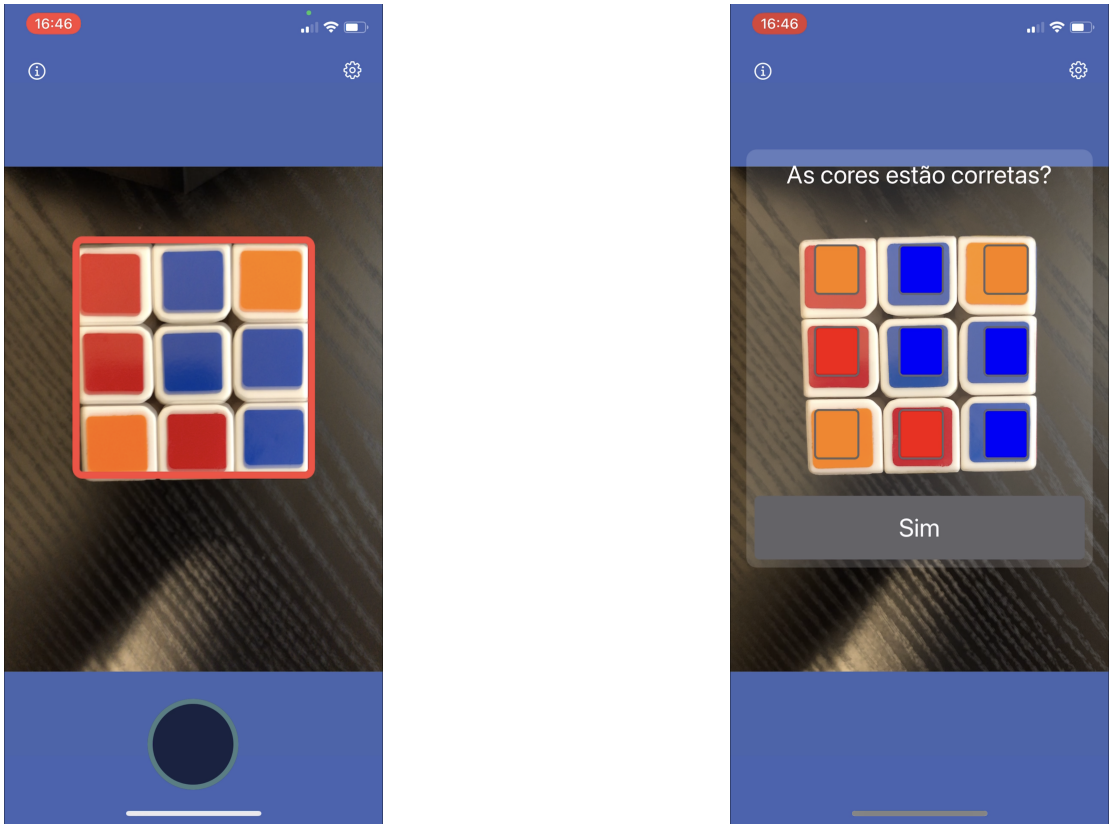
Figura 35 – Interface para o ajuste de acurácia dos servomotores. Fonte: Própria

7.2.2 Visão Computacional

O *software* de visão computacional implementado no aplicativo de controle foi desenvolvido tendo em vista as diferentes luminosidades possíveis em um ambiente, podendo ser iluminado, por exemplo, por uma luz incandescente ou pela luz do sol.

No entanto, imprevistos podem ocorrer de modo a sair das situações previstas que foram implementadas. A figura 36a ilustra um imprevisto em que a foto tirada pelo usuário se apresentou borrada. Isso causou um clareamento na cor vermelha da peça do cubo de Rubik, o que o aplicativo interpretou como uma peça laranja.

Esses resultados errôneos podem ser gerados de diferentes maneiras além do descrito. Por exemplo, a iluminação desigual pode gerar uma maior exposição de forma parcial no cubo, gerando uma imprecisão na detecção das cores. O contrário também poderia ser verdade, ocorrendo em regiões de sombra parcial no cubo. A fim de combater os possíveis imprevistos na captura das fotos, foi desenvolvida uma interface que necessita da confirmação do usuário de forma à garantir se a detecção de cor foi correta. Caso errada, o usuário tem a possibilidade de corrigir por meio de botões com as cores correspondentes na interface. A figura 36b demonstra tal interface do aplicativo.



(a) Foto capturada borrada.

(b) Solução à erros de detecção de cor

Figura 36 – Visão computacional falhas e sua solução. Fonte: Própria

7.2.3 Trabalhos Futuros

Os testes realizados elucidaram possíveis etapas para o avanço do projeto. O sistema poderia ser desenvolvido nos seguintes quesitos:

- **Aplicativo Android**

  O aplicativo desenvolvido no trabalho foi feito na linguagem *Swift* para a plataforma iOS. No entanto, a transferência das funções já desenvolvidas para o sistema Android resultaria numa maior compatibilidade de celulares para o uso do robô.

- **Movimentos mais eficientes dos servomotores**

  Os testes demonstraram uma ineficiência de movimentos dos servos. Para cada rotação de face, os servomotores e o cubo de Rubik são retornados para as posições iniciais prévias ao movimento. Ou seja, caso seja necessário realizar o movimento de rotação da face superior, por exemplo, o cubo por inteiro será rotacionado para que tal face esteja acessível ao servomotor. Uma vez que a face superior é rotacionada, o cubo por inteiro é retornado para sua posição inicial. A falta de eficácia consta na possibilidade do movimento seguinte ser novamente o movimento de rotação da face superior, ou mesmo inferior. Neste caso, o cubo teria que ser novamente rotacionado

para que as faces superior e inferior possam ser acopladas aos atuadores, gerando então rotações desnecessárias entre movimentos.

Assim, um desenvolvimento de análise do caminho de solução para aumentar a eficácia de movimentos na solução do cubo de Rubik beneficiaria a velocidade de solução do robô.

- **Funcionalidades além da solução**

  O projeto cumpriu os objetivos estipulados, sendo capaz de solucionar o cubo de Rubik de forma autônoma. No entanto, caso o projeto fosse distribuído na categoria de comercialidade, seria benéfico o desenvolvimento de funcionalidades além do estipulado do projeto. Um exemplo seria um sistema de embaralhamento do cubo de Rubik, no qual o usuário gostaria de resolver o quebra-cabeça a partir de um estado imparcial de embaralhamento. Para tal, seria possível que tal usuário colocasse o cubo no robô para que o quebra-cabeça fosse embaralhado de forma aleatória.

8 Conclusão

O projeto têm como objetivo a construção de um robô capaz de solucionar o cubo de Rubik de forma autônoma, por meio de um aplicativo de *smartphone* cuja câmera será utilizada para uma visão computacional do quebra-cabeça.

Por meio de uma pesquisa bibliográfica, foi possível o conhecimento de diversos tipos de robôs com o mesmo objetivo do trabalho, mas em contextos variados. Desde projetos que visam a quebra do recorde mundial de velocidade de solução, que utilizam seis motores potentes acoplados à cada face de um cubo especificamente modificado para a acomodação desses atuadores. Até projetos que visam uma construção mais acessível. Para alcançar esse objetivo, utilizam estruturas baseadas em LEGO Mindstorms, madeira e impressão 3D. Foram também pesquisados robôs cujo o objetivo é a comercialidade como produto. Esse tipo foca em uma estrutura de boa estética, assim como funcionalidades além da solução autônoma.

Com base então nessa análise da literatura, uma configuração efetiva proposta foi a de uma estrutura impressa em plástico PLA, com servomotores MG90S como atuadores. O sistema possui oito servos no total, metade para que cada face lateral do cubo de Rubik possa ser rotacionada, e os demais quatro para que os atuadores possam acoplar e desacoplar às faces do quebra-cabeça. Tais atuadores possuem um torque de 1.8 kgf·cm, além do suficiente para a rotação de uma face do cubo de Rubik, um controle PWM de servo do tipo RC e disponibilidade no mercado brasileiro.

Para o controle destes atuadores, foi utilizado o microcontrolador ESP32. O ESP32 é um microcontrolador de baixo custo que possui *Bluetooth* integrado. O microcontrolador possui 16 canais diferentes para o uso de PWM, com capacidade de controlar os oito servos.

Assim, o ESP32 envia comando para os servomotores para realizar a rotação das faces do cubo de Rubik. Essa rotação é possível devido ao sistema baseado em uma cremalheira que causa o ato de acoplar e desacoplar da garra ao cubo. Somando os movimentos das quatro torres no total, é possível garantir que todas as faces do quebra-cabeça possam sofrer rotações.

Os comandos de rotação recebidos pelo ESP32 são enviados por um aplicativo de celular por meio do uso do protocolo BLE. O aplicativo possui dois modos, o modo de controle manual e o modo de solução autônoma.

Para o controle manual do sistema foi desenvolvida uma interface que é possível enviar comandos representados pelos três movimentos possíveis de rotação das 6 faces do

cubo de Rubik. Já o modo autônomo é caracterizado pelo uso do algoritmo do Kociemba, Funcionando com base na teoria de grupos da matemática, o algoritmo atua de acordo com o estado inicial das cores do cubo iterando diferentes possibilidades de resolução. Assim, como parâmetros do algoritmo, fotos das faces do cubo são utilizadas a partir do uso da câmera já embutida no celular. A partir do programa desenvolvido de visão computacional, é possível a obtenção das cores de cada peça para cada face.

A partir da conclusão das seções de mecânica, da elétrica e da programação, foram realizados testes a fim de assegurar a viabilidade do projeto para uma solução autônoma do cubo de Rubik. Tais testes revelaram a necessidade de ajustes em relação à acurácia dos atuadores quando aplicado torque à sua rotação. Para compensar essa falta, o aplicativo possui uma interface capaz de alterar o ângulo de rotação enviado ao atuador.

Outro ajuste necessário em relação à solução autônoma foi a de diferentes exposições da foto capturada a ser processada pela visão computacional. Quando presente um feixe de luz que afeta parcialmente o cubo, a exposição da câmera afeta o resultado. Para tal, foi desenvolvido um passo em que o usuário há de conferir e corrigir as cores processadas.

Uma vez que as soluções dessas perturbações da solução autônoma foram implementadas, todas as seções trabalham em conjunto resultando no robô sendo capaz de solucionar o cubo de Rubik com uma média de 1 minuto e 8 segundos. A partir disso, o projeto cumpre de forma satisfatória os objetivos propostos em sua concepção inicial.

Referências

1 KATZ, J. D. C. B. *The Rubik's Contraption*. Acessado em: 2020-03-24. Disponível em: <http://build-its-inprogress.blogspot.com/2018/03/the-rubiks-contraption.html>.

2 ROSE, J. F. P. *Robot created by software developers solves Rubik's cube in record time*. Acessado em: 2020-03-24. Disponível em: <https://www.guinnessworldrecords.com/news/2016/2/rubiks-cube-robot>.

3 MATT2UY. *Rubik's Cube Solver*. Instructables, 2017. Acessado em: 2020-03-19. Disponível em: <https://www.instructables.com/id/Rubiks-Cube-Solver/>.

4 BRICKUBER - How To Build a Raspberry Pi Rubiks Cube Solving Robot. Acessado em: 2020-03-19. Disponível em: <https://www.dexterindustries.com/projects/brickuber-project-raspberry-pi-rubiks-cube-solving-robot-project/>.

5 HOANG, L. T. *Lego Robot Rubik's Cube Solver*. Acessado em: 2020-03-19. Disponível em: <https://www.youtube.com/watch?v=Q8BYKwbwZSM>.

6 CUBESTORMER 3. Wikimedia Foundation, 2019. Acessado em: 2020-03-19. Disponível em: <https://en.wikipedia.org/wiki/Cubestormer_3>.

7 OTVINTA®. *3D-PRINTED RUBIK'S CUBE ROBOT*. Acessado em: 2020-03-19. Disponível em: <http://www.rcr3d.com/intro.html>.

8 GANCUBE. *GAN ROBOT*. Acessado em: 2020-03-19. Disponível em: <https://www.gancube.com/gan-robot>.

9 PRO, T. *MG90S*. Acessado em: 2020-02-15. Disponível em: <https://www.towerpro.com.tw/product/mg90s-3/>.

10 ZHILIAN, T. S. Acessado em: 2020-04-12. Disponível em: <http://doit.am/>.

11 THAKUR, R. et al. *ESP32 DevKit ESP32-WROOM GPIO Pinout*. 2018. Acessado em: 2020-03-14. Disponível em: <https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/>.

12 PROPRIETARY, B. S. Bluetooth core specification. *Bluetooth® Specification*, 2019.

13 WIKIPéDIA. *Fichier:HSV color solid cylinder.png*. Acessado em: 2019-09-19. Disponível em: <https://fr.wikipedia.org/wiki/Fichier:HSV_color_solid_cylinder.png>.

14 SNYDER, C. *How the Rubik's Cube became one of the bestselling toys in history*. Business Insider, 2018. Acessado em: 2020-03-19. Disponível em: <https://www.businessinsider.com/how-rubiks-cube-became-one-of-bestselling-toys-in-history-erno-rubik-2018-10>.

15 EL-SOURANI, N.; BORSCHBACH, M. Design and comparison of two evolutionary approaches for solving the rubik's cube. In: SPRINGER. *International Conference on Parallel Problem Solving from Nature*. [S.l.], 2010. p. 442–451.

16 ROKICKI, T. et al. *God's Number is 20*. 2010. Acessado em: 2020-02-20. Disponível em: <https://cube20.org/>.

17 VISHNURAM, L. et al. *Radio Versions*. Acessado em: 2019-12-7. Disponível em: <https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/radio-versions/>.

18 TEAM, O. *OpenCV*. 2020. Acessado em: 2019-12-13. Disponível em: <https://opencv.org/>.

19 BEER, A. *Fastest robot to solve a Rubik's Cube*. Acessado em: 2020-03-24. Disponível em: <https://www.guinnessworldrecords.com/world-records/fastest-robot-to-solve-a-rubiks-cube>.

20 KOCIEMBA, H. *Two-Phase Algorithm Details*. Acessado em: 2020-02-20. Disponível em: <http://kociemba.org/math/imptwophase.htm>.

21 LU, S. L.; HUANG, M.; KONG, F. R. The design of a Rubik's Cube robot. *Advanced Materials Research*, v. 709, p. 432–435, 2013. ISSN 10226680.

22 DIEGO-MANTECÓN, J. M. et al. An engineering technology problem-solving approach for modifying student mathematics-related beliefs: Building a robot to solve a rubik's cube. *International Journal for Technology in Mathematics Education*, v. 26, n. 2, 2019.

23 SEMICONDUCTOR, O. *2N7000 / 2N7002 / NDS7002A*. Semiconductor Components Industries LLC, 2017. Acessado em: 2020-03-4. Disponível em: <https://www.onsemi.com/pub/Collateral/NDS7002A-D.PDF>.

24 BAUERMEISTER, G. *Bluetooth Low Energy com ESP32 e DHT11*. 2019. Acessado em: 2019-12-13. Disponível em: <https://www.filipeflop.com/blog/bluetooth-low-energy-com-esp32-e-dht11/>.

25 WHITAKER, J. et al. Herbert: Autonomous Rubik ' s Cube Solver. 2016.

26 PAVANI, M.; VILLANI, T. Sistema de Pouso Autônomo de Drones Assistido por Imagens. 2018.

27 TEAM, O. *Canny Edge Detection*. 2020. Acessado em: 2019-09-19. Disponível em: <https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html>.

28 RUBIK'S Cube. Wikimedia Foundation, 2020. Acessado em: 2020-09-22. Disponível em: <https://en.wikipedia.org/wiki/Rubik's_Cube>.

29 MCALEER, S. et al. Solving the Rubik's cube with approximate policy iteration. *7th International Conference on Learning Representations, ICLR 2019*, p. 1–13, 2019.

30 KOCIEMBA, H. *Two-Phase Algorithm*. Acessado em: 2020-09-20. Disponível em: <http://kociemba.org/cube.htm>.

31 RUWIX, *Rubik's Cube Notation*. Acessado em: 2020-04-20. Disponível em: <https://ruwix.com/the-rubiks-cube/notation/>.

Apêndices

APÊNDICE A – Notação do Cubo de Rubik

Para facilitar a resolução de um cubo de Rubik, é padrão utilizar a terminologia e orientação usado em análises do cubo.

Como esperado, o cubo em questão apresenta 6 faces diferentes, a frontal (**F**), a lateral direita (**R**), a lateral esquerda (**L**), a traseira (**B**), a superior (**U**) e a inferior (**D**). Como demonstra a figura 37, esta notação de orientação é relativa à posição na qual o usuário segura o cubo. Por exemplo, se alinhar a face azul na direção do usuário, a face azul é então definida como a face frontal. No entanto, é comum se utilizar a face branca como a face superior (U), e por consequência, a amarela como inferior (D).

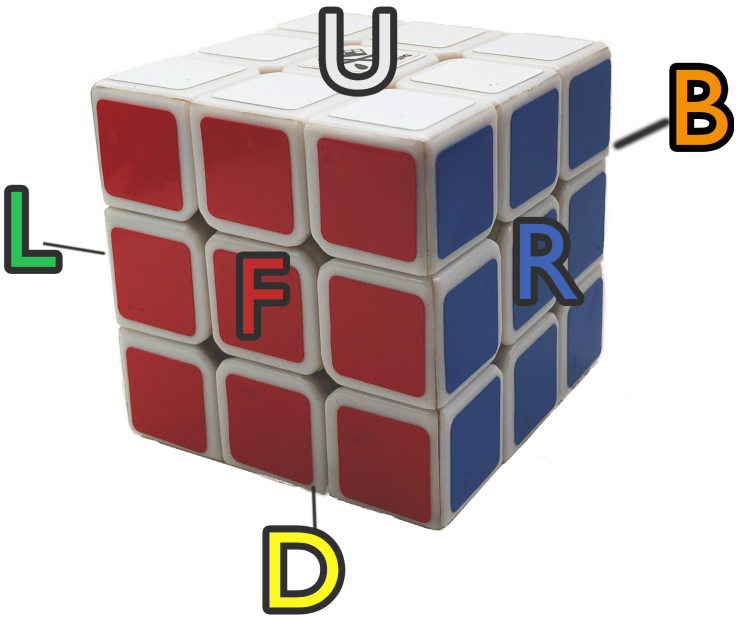


Figura 37 – Notação das faces do cubo de Rubik. Fonte: Própria

Cada face pode ser rotacionada em duas direções diferentes, no sentido horário ou anti-horário. Esses movimentos são definidos a partir da direção de rotação como se o usuário estivesse encarando a face diretamente. Em vista disso, são definidos os movimentos fundamentais. Onde, por exemplo, **R** define a rotação da face direita em 90^o no sentido horário, de modo que a ponta no canto superior direito gire para trás.

São também definidos modificadores. Ao juntar o **R** com o modificador **'**, **R'**, define-se a rotação da face direita em 90^o no sentido anti-horário, de modo que a ponta no canto superior direito gire para frente. Há também o modificador **2**, onde **R2** significa uma rotação da face direita em 180^o [31].

Movimento	**Descrição**
F	Indica uma rotação da face frontal em 90^o no sentido horário
F'	Indica uma rotação da face frontal em 90^o no sentido anti-horário
F2	Indica uma rotação da face frontal em 180^o
U	Indica uma rotação da face superior em 90^o no sentido horário
U'	Indica uma rotação da face superior em 90^o no sentido anti-horário
U2	Indica uma rotação da face superior em 180^o
R	Indica uma rotação da face direita em 90^o no sentido horário
R'	Indica uma rotação da face direita em 90^o no sentido anti-horário
R2	Indica uma rotação da face direita em 180^o
L	Indica uma rotação da face esquerda em 90^o no sentido horário
L'	Indica uma rotação da face esquerda em 90^o no sentido anti-horário
L2	Indica uma rotação da face esquerda em 180^o
B	Indica uma rotação da face traseira em 90^o no sentido horário
B'	Indica uma rotação da face traseira em 90^o no sentido anti-horário
B2	Indica uma rotação da face traseira em 180^o
D	Indica uma rotação da face inferior em 90^o no sentido horário
D'	Indica uma rotação da face inferior 90^o no sentido anti-horário
D2	Indica uma rotação da face inferior 180^o

Tabela 3 – Rotações fundamentais. Fonte: Própria

APÊNDICE B – Projeto Eletrônico

Foi desenvolvida uma Placa de Circuito Impresso (PCB) para dar suporte ao ESP32 de forma que existam conectores onde o microcontrolador e a fonte possam ser inseridos. Também há pinos de saída para os servomotores a serem controlados.

Para o *design* da PCB foi utilizado o *DipTrace* que é um *software* EDA(*electronic design automation*) para a criação de diagramas esquemáticos e disposição de componentes para a montagem da placa de circuito impresso.

B.1 Esquemático

No *DipTrace* foi então esquematizado o circuito a ser implementado no robô. Como é possível observar na figura 38, a placa do ESP32 é o centro do circuito. No microcontrolador estão conectados os oito conversores de nível de tensão onde se conectam os servomotores por meio de *headers*. Também nela, um *header* para a soldagem de um botão de *reset* e um *LED* para notificação do estado da máquina. À esquerda, a entrada para a fonte de alimentação de 5V que alimentará o circuito.

B.2 *Layout*

Com o esquemático feito, o *software* DipTrace possibilita com facilidade a sua transformação a um PCB. Seu resultado final está apresentado nas figuras 39 e 40 e sua lista de componentes encontra-se descrita na tabela 4.

Referência	**Valor**	**Nome**	**Quantidade**
C1	100uF	Capacitor	1
D1	LED verde	LED	1
J1, J2, J3, J4, J5, J6, J7, J8	Saída para o servomotor	*Header* 3x1	8
J9	RESET	*Header* 2x1	1
J10	POWER	Conector Fit JS-3027-02	1
Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8	MOSFET	2N7000	8
R1, R2, R3, R4, R5, R6, R7, R8	10k	Resistor	8
R9	330	Resistor	1
U2	ESP32	ESP32 DOIT Devkit	1

Tabela 4 – *Bill of Materials*. Fonte: Própria

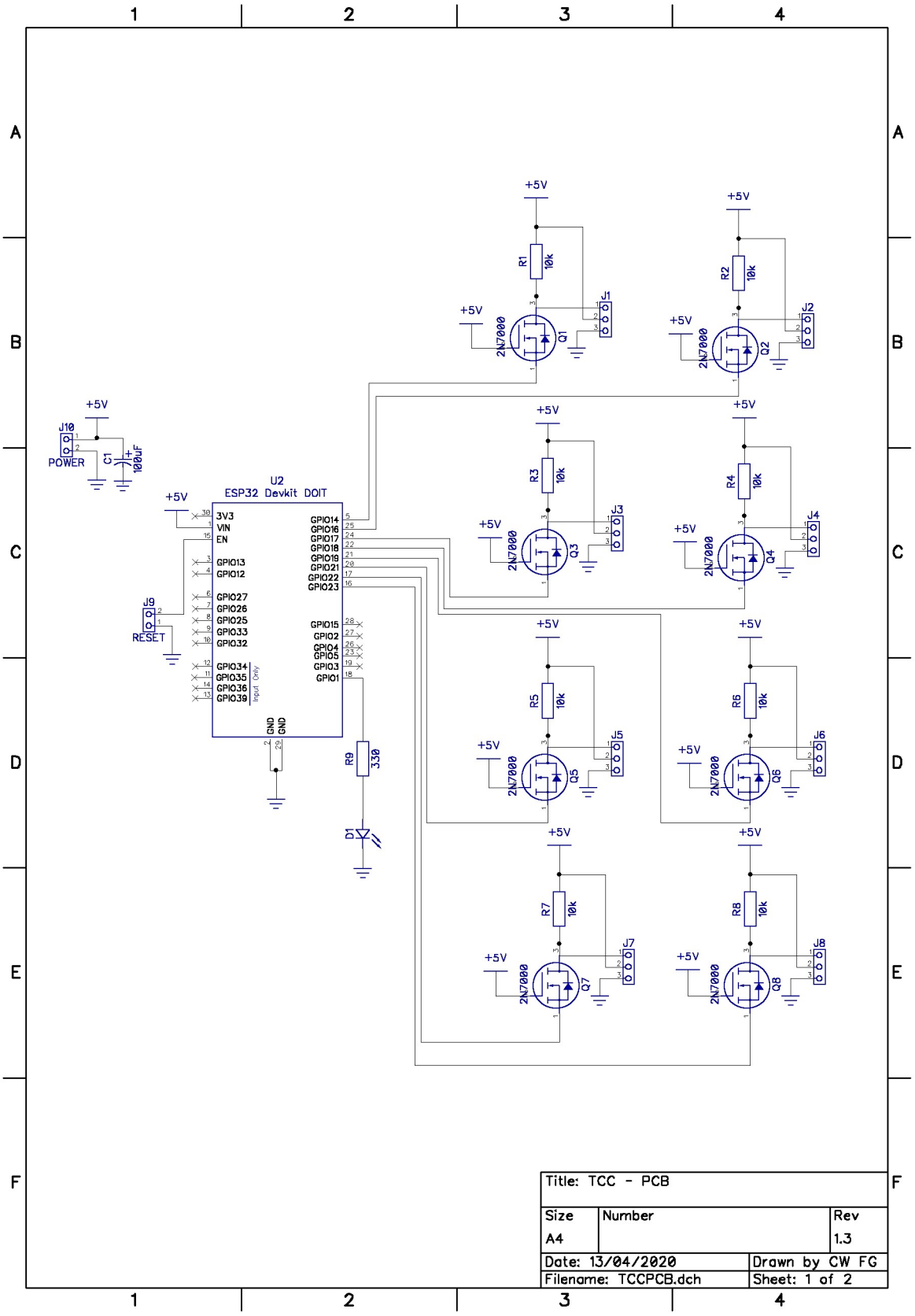


Figura 38 – Esquemático. Fonte: Própria

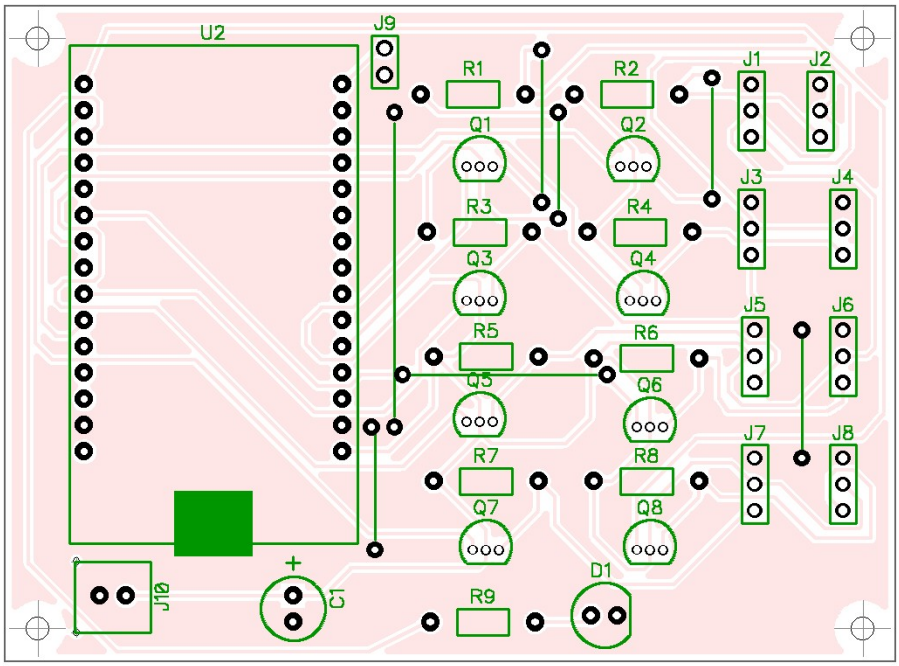


Figura 39 – Parte superior do layout da PCB. Fonte: Própria

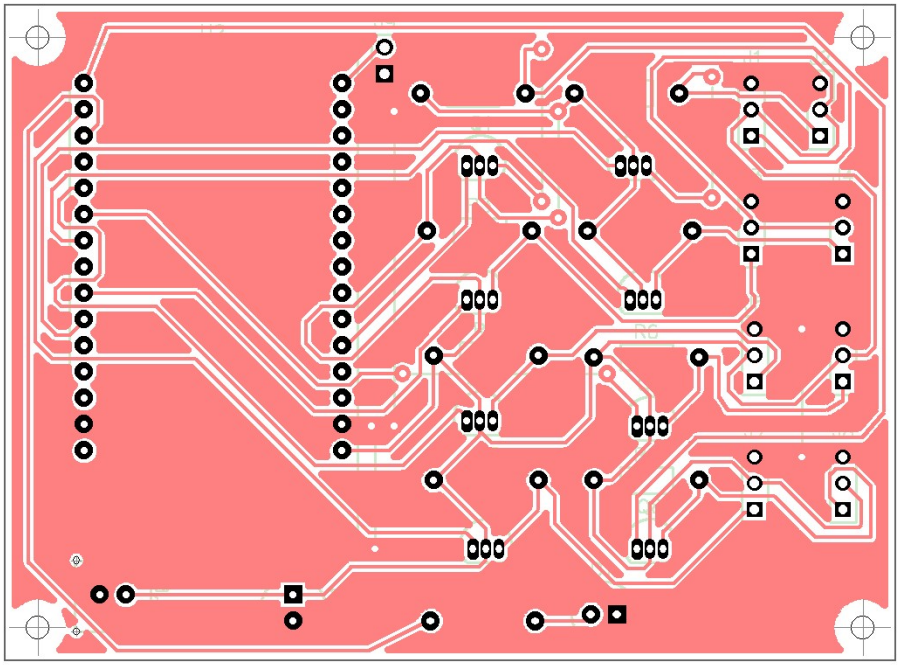


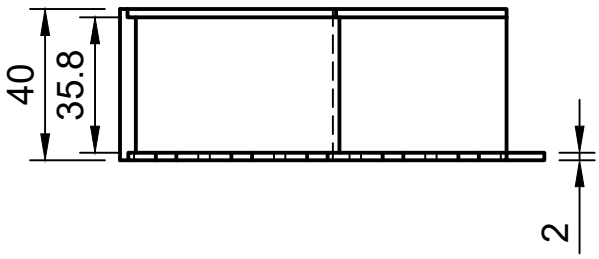
Figura 40 – Face de solda da PCB. Fonte: Própria

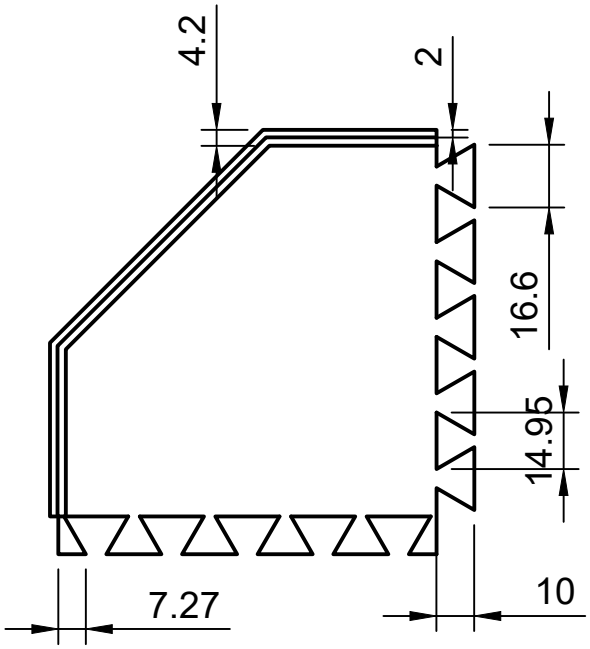
APÊNDICE C – Projeto Mecânico

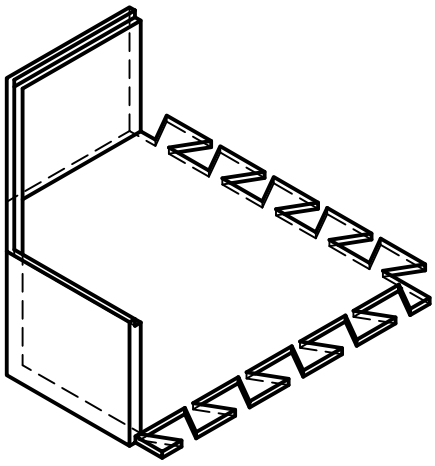
Para a estrutura mecânica do robô, foi desenvolvido um *design* em CAD capaz de abrigar os componente eletrônicos e realizar as movimentações necessárias dos servomotores.

Sua elaboração foi realizada no *software Fusion 360*, capaz de modelar estruturas tridimensionais de forma precisa. Outra vantagem do *software* é sua capacidade de conversão dos objetos tridimensionais em arquivos compatíveis com impressoras 3D.

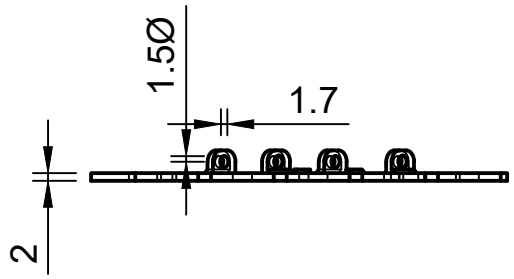
Assim, o robô foi elaborado visando a facilidade de montagem, respeitando as limitações dimensionais e de construção de arcos de impressoras 3D. Tais peças foram feitas para serem montadas com elementos de fixação como parafusos ou por meio de juntas de encaixe, como encontradas nas peças *bottom* e *basePart* da base.

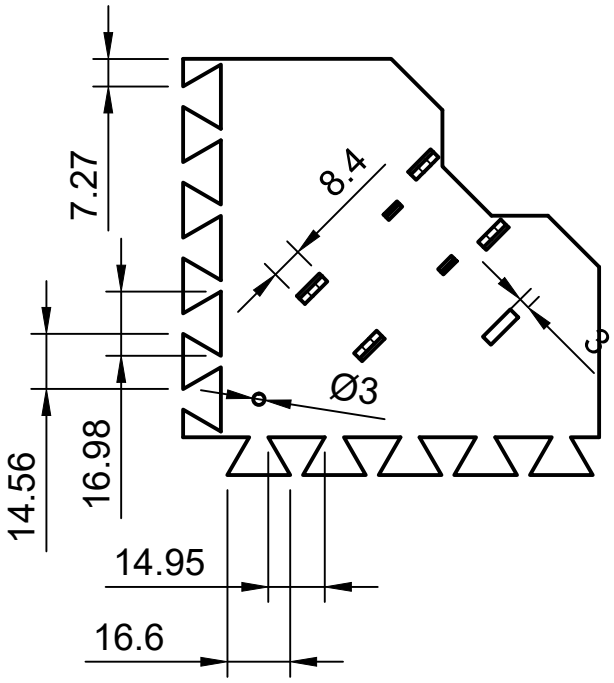


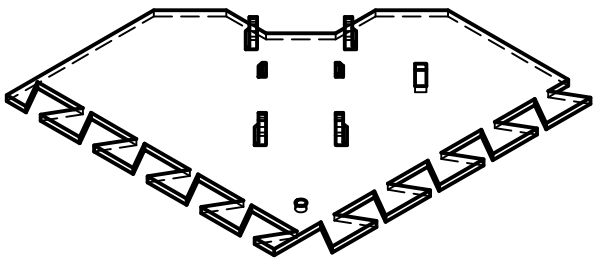




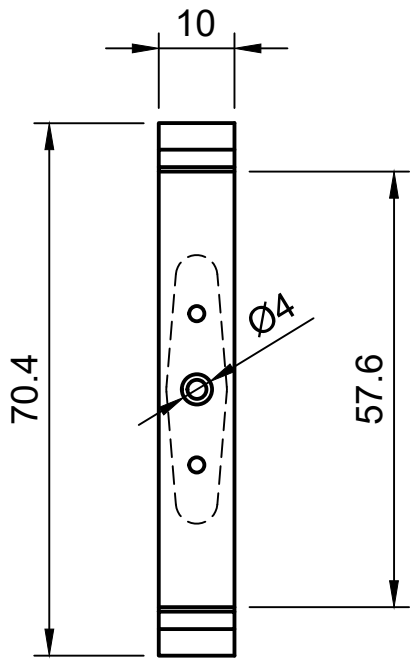
Dept.	Technical reference	Created by	Approved by	
		CW FG		
		Document type	Document status	
		Desenho de fabricação		
		Title	DWG No.	
		basePart	1	
			Rev.	Date of issue / Sheet
			20	08/04/2020 / 1/1

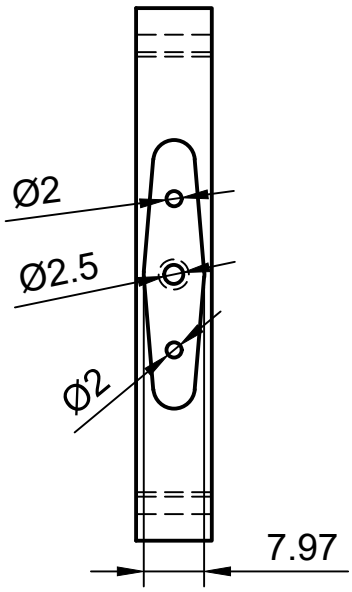


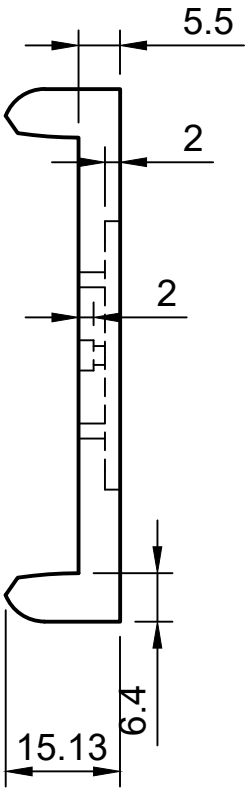


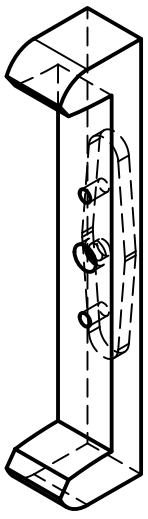


Dept.	Technical reference	Created by	Approved by		
		CW e FG			
		Document type	Document status		
		Desenho de fabricação			
		Title	DWG No.		
		bottom			
			Rev.	Date of issue	Sheet
			2	08/04/2020	1/1

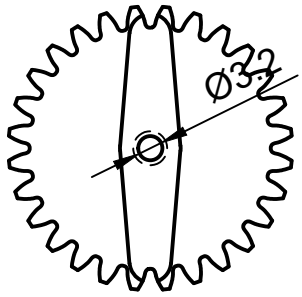




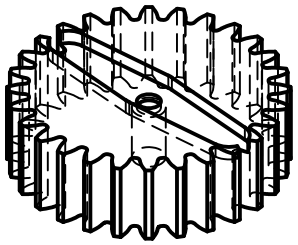




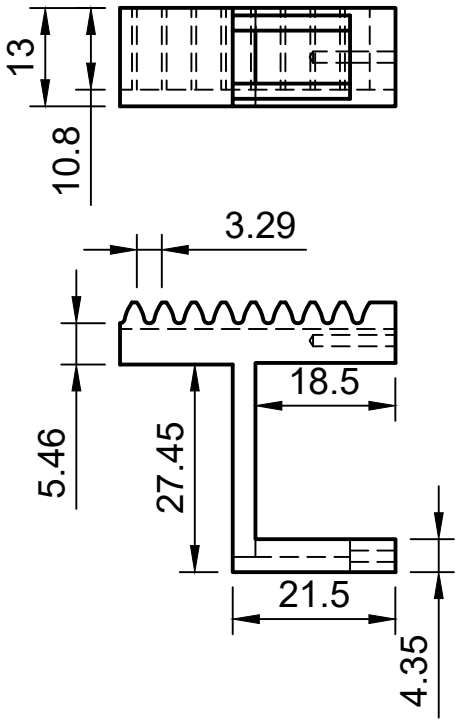
Dept.	Technical reference	Created by	Approved by	
		CW e FG		
		Document type	Document status	
		Desenho de fabricação		
		Title	DWG No.	
		interferenceClaw		
			Rev.	Date of issue / Sheet
			14	14/08/2020 / 1/1

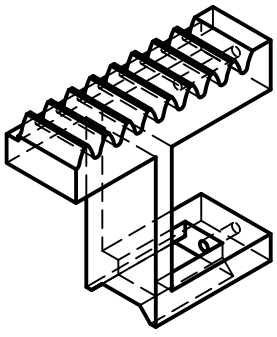


Number of teeth	28
Pitch diameter	16.6 mm
Outside diameter	18.86 mm
Inside diameter	15.9 mm

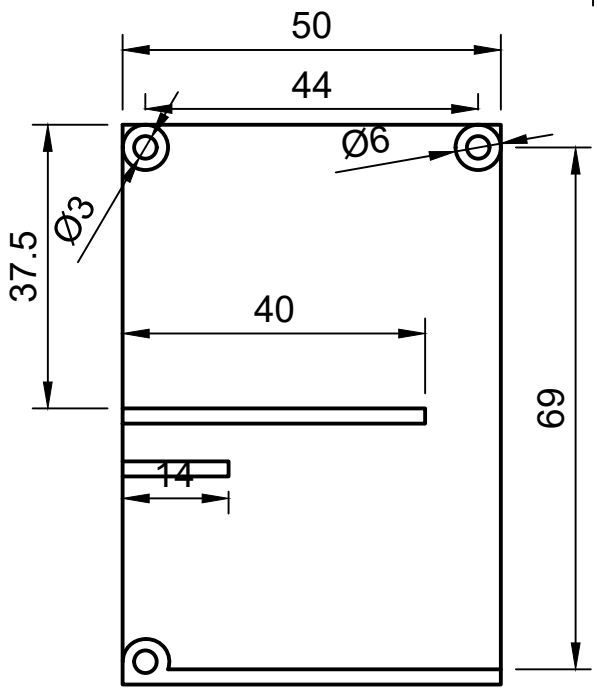


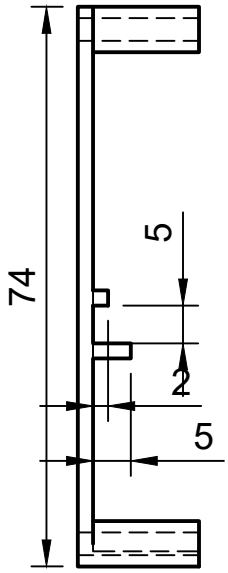
Dept.	Technical reference	Created by	Approved by		
		CW e FG			
		Document type	Document status		
		Title	DWG No.		
		pinion			
			Rev.	Date of issue	Sheet
			10	08/04/2020	1/1

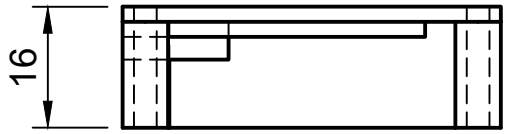


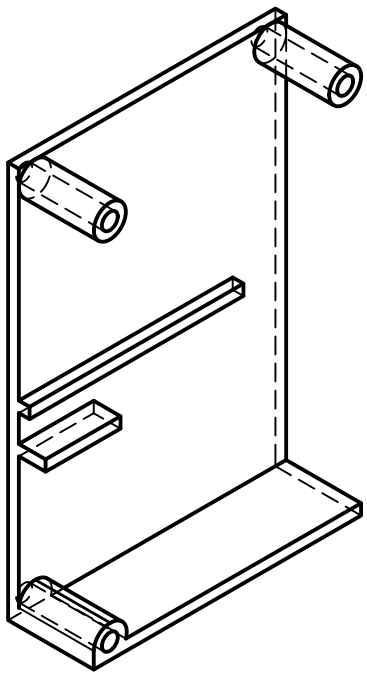


Dept.	Technical reference	Created by	Approved by	
		CW e FG		
		Document type	Document status	
		Desenho de fabricação		
		Title	DWG No.	
		rack		
			Rev.	Date of issue / Sheet
			13	04/04/2020 / 1/1

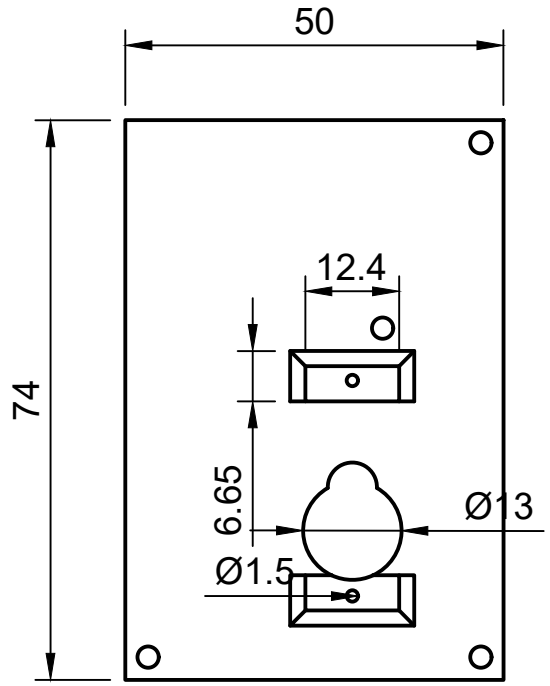


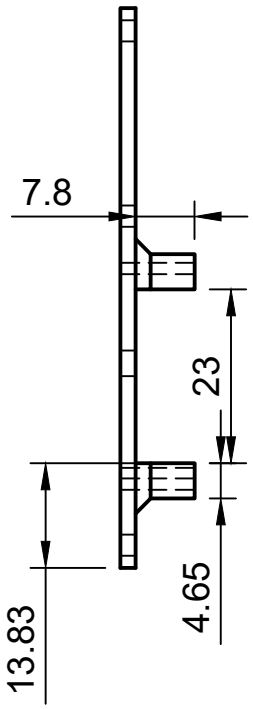


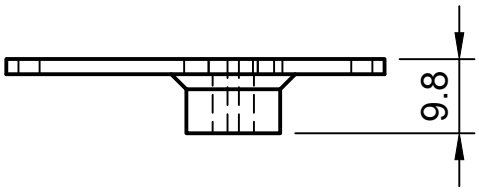


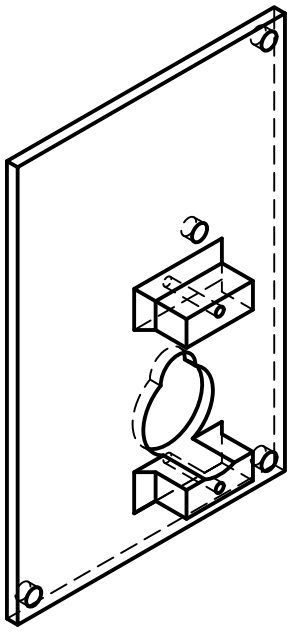


Dept.	Technical reference	Created by CW e FG	Approved by	
		Document type	Document status	
		Title towerCounterWall	DWG No.	
			Rev. 23	Date of issue 03/04/2020	Sheet 1/1









Dept.	Technical reference	Created by	Approved by	
		CW e FG		
		Document type	Document status	
		Desenho de fabricação		
		Title	DWG No.	
		towerServoWall		
			Rev.	Date of issue / Sheet
			22	04/08/2020 / 1/1