FACULDADE ÚNICA DE IPATINGA

ISAQUE DE ALMEIDA COSTA
CARLOS EDUARDO VIEIRA RIBEIRO
KENEDY KEVYN TOSTES MIRANDA
ROBERT JHON LAGE MOREIRA

MAGIC GLOBE - A RUBIK'S CUBE SOLVER MACHINE

IPATINGA
2019

ISAQUE DE ALMEIDA COSTA
CARLOS EDUARDO VIEIRA RIBEIRO
KENEDY KEVYN TOSTES MIRANDA
ROBERT JHON LAGE MOREIRA

MAGIC GLOBE - A RUBIK'S CUBE SOLVER MACHINE

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação da Faculdade Única de Ipatinga como requisito parcial para obtenção de título de Bacharel em Ciência da Computação.

Orientador(a): Júlio Cézar da Silva Costa

IPATINGA
2019

FACULDADE ÚNICA DE IPATINGA

ISAQUE DE ALMEIDA COSTA
CARLOS EDUARDO VIEIRA RIBEIRO
KENEDY KEVYN TOSTES MIRANDA
ROBERT JHON LAGE MOREIRA

MAGIC GLOBE - A RUBIK'S CUBE SOLVER MACHINE

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação da Faculdade Única de Ipatinga como requisito parcial para obtenção de título de Bacharel em Ciência da Computação.

Aprovada em 10/12/2019

BANCA EXAMINADORA

Professor: Júlio Cézar da Silva Costa

Professor: Thiago de Medeiros Gualberto

Professor: Wander Marcelo Camilo de Sousa

AGRADECIMENTOS

Agradecemos à todas as pessoas que nos ajudaram ao longo deste projeto: à família que sempre prestou todo suporte necessário e aos amigos que nos apoiaram e incentivaram nesta longa caminhada. Agradecemos também aos professores: Filipe Costa que nos apresentou como desafio, Júlio Cezar que nos apoiou a realizá-lo, Josimar que nos ajudou com a estrutura do primeiro protótipo, e principalmente ao técnico de laboratório João Victor Rocha que com muita boa vontade nos ajudou nesta reta final. Também agradecemos ao Marcos da Cubo3D que apoiaram a ideia e nos ajudaram a concretizá-la com excelência. Por fim agradecemos em especial ao Renato, pai do autor Isaque de Almeida Costa, que desde o início deste trabalho tem sido peça fundamental para realização do mesmo. Abraçou a ideia antes mesmo de nós, não mediu esforços em nenhum momento, logo no início nos presenteou com as primeiras peças usadas no projeto e participou ativamente do desenvolvimento do início ao fim, sem ele não teríamos chegado até aqui!

"Se você não consegue explicar algo de forma simples, você não entendeu suficientemente bem."

Albert Einstein

RESUMO

O presente trabalho de conclusão de curso tem como objetivo a resolução de qualquer Cubo Mágico 3x3 em qualquer um de seus 43.252.003.274.489.856.000 (quarenta e três quintilhões, duzentos e cinquenta e dois quadrilhões, três trilhões, duzentos e setenta e quatro bilhões, quatrocentos e oitenta e nove milhões e oitocentos e cinquenta e seis mil) possíveis estados. A máquina é composta por uma estrutura de peças modeladas em 3D, um Cubo Mágico, 6 motores de passo, um driver controlador para cada motor, duas câmeras, uma placa Arduino MEGA com módulo ESP32, um computador e um *iPad* para controlar toda a máquina. A resolução do Cubo Mágico começa com a captura do *feed* das duas câmeras para identificar cada um dos 54 *cublets* que são divididos pelos 6 lados do cubo, o computador utilizando técnicas de visão computacional processa essas imagens e traduz essa informação em uma *string* que determina o estado atual do cubo, essa *string* é enviada para a biblioteca Kociemba que usa essa informação para determinar a sequência de movimentos que irá solucionar o Cubo, toda essa sequência de movimentos é enviada para a placa Arduino pela porta serial ou pela rede e, por fim o driver executa todos esses movimentos em sequência resolvendo o Cubo Mágico. Este processo leva de 1 a 2 segundos.

Palavras-Chave: Cubo mágico, resolução, visão computacional, OpenCV, velocidade.

ABSTRACT

The present course completion paper aims to solve any 3x3 Magic Cube in any of its 43,252,003, 274,489,856,000 (forty-three quintillion, two hundred and fifty-two quadrillion, three trillion, two hundred and seventy-four billion, four hundred and eighty-nine million and eight hundred and fifty-six thousand) possible states. The machine is made up of a 3D modeled part structure, a Rubik's Cube, 6 stepper motors, a controller driver for each engine, two cameras, an Arduino MEGA board with ESP32 module, a computer and an iPad to control the entire machine. . The resolution of the Magic Cube begins with capturing the feed from the two cameras to identify each of the 54 cublets that are divided by the 6 sides of the cube. The computer using computer vision techniques processes these images and translates this information into a string that determines the current state of the cube, this string is sent to the kociemba library which uses this information to determine the motion sequence that will solve the cube, all this motion sequence is sent to the Arduino board via the serial port or network, and finally The driver performs all these moves in sequence by solving the Magic Cube. This process takes up to 1 or 2 seconds.

Keywords: Rubik's Cube, speedcubing, computer vision, OpenCV, robot.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

3D	Tridimensional.
TI	Tecnologia da Informação.
LED	*Light-emitting Diode* (Diodo Emissor de Luz).
USB	*Universal Serial Bus* (Porta universal).
ABS	*Anti-lock Braking System*.
STL	Tipo de arquivo, abreviação da palavra estereolitografia.
SSE	*Streaming SIMD Extensions* (Extensões SIMD para *streaming*).
HTML	*Hyper Text Markup Language*.
CUDA	*Compute Unified Device Architecture* (Arquitetura de Dispositivo de Computação Unificada).
OpenCV	*Open Source Computer Vision Library* (Biblioteca de visão computacional *open source*).
Mbps	*Megabits per second* (*Megabits* por segundo).

SUMÁRIO

1 INTRODUÇÃO

1.1 Tema

Desenvolvimento de robô para resolução do Cubo Mágico 3x3, utilizando visão computacional e sistemas distribuídos.

1.2 Problema

Como aplicar de forma eficiente os avanços tecnológicos na área da TI para solucionar um dos brinquedos mais vendidos de todos os tempos?

1.3 Hipótese

Através do *machine learning* (OpenCV), seria possível identificar as cores de uma imagem, e a eletrônica possibilitaria desenvolver um robô para resolver o Cubo Mágico.

1.4 Justificativa

Para alcançar uma solução do Cubo Mágico de forma extremamente rápida, como conseguiu-se ao concluir o projeto, é necessário colocar em prática diversas habilidades da área de TI, o que proporcionou um ganho profissional útil em diversos cenários.

A visão computacional por exemplo, pode ser utilizada para automação de um estacionamento ou para facilitar na execução de uma intravenosa, a robótica tem inúmeras aplicações para automação residencial e industrial, além das demais áreas como a da saúde onde é utilizada para o desenvolvimento de próteses cada vez melhores e mais próximas do membro o qual pretende-se suprir.

De forma geral, este projeto foi escolhido por que possibilitou de forma prática aprender mais sobre as tecnologias as quais acreditou-se serem promissoras no ramo da TI.

1.5 Objetivos

1.5.1 Geral

A resolução do Cubo Mágico através de uma máquina, de forma rápida, superando os melhores competidores (humanos).

1.5.2 Específicos

- Explorar as aplicações e obter maiores conhecimentos sobre Arduino e Impressão 3D.
- Demonstrar que a resolução do Cubo Mágico pode ser realizada utilizando-se de técnicas de visão computacional, robótica e outras tecnologias.

2 REFERENCIAL TEÓRICO

2.1 Rubik's Cube (Cubo Mágico)

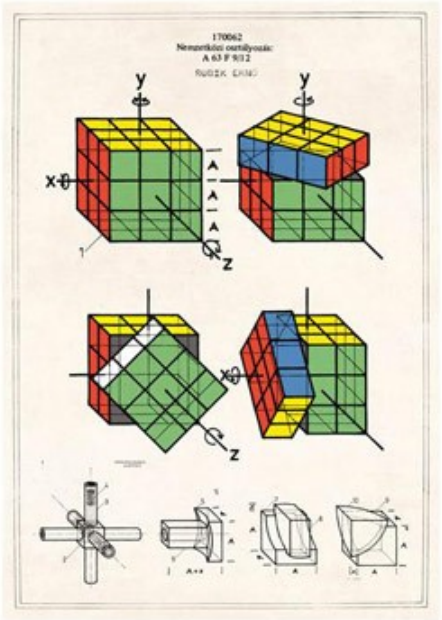
Figura 1 – Ernő Rubik, o inventor do Cubo Mágico.

Fonte – (RUBIK, 1974).

Em 1974, (RUBIK, 2019) ainda um jovem professor de arquitetura de Budapeste (Hungria) criou um objeto que não deveria ser possível. Mesmo após de ter sido girado, o cubo não quebrou ou desmontou. Com adesivos coloridos em suas faces, o cubo foi embaralhado e assim surgiu o primeiro "Cubo de Rubik". Ernő precisou de um pouco mais de um mês para conseguir solucionar este quebra-cabeça. **Mal ele sabia que o cubo se tornaria o brinquedo mais vendido do mundo.**

Como professor, Ernő (Figura 1) sempre procurava por coisas novas, maneiras mais emocionantes para transmitir informação, então ele usou o primeiro modelo do cubo para ajudá-lo a explicar aos seus alunos sobre relações de espaço. Ernő sempre viu o cubo como uma peça de arte, uma escultura móvel simbolizando contrastes da condição humana: problemas desconcertantes e inteligência triunfal; simplicidade e complexidade; estabilidade e dinâmica, ordem e caos. Para este objeto mágico se tornar o brinquedo mais vendido do mundo, algumas tentativas foram necessárias. (RUBIK, 2019).

Figura 2 – Patente registrada por Ernő Rubik em 1975.



Fonte – (RUBIK, 1975).

Assim como as maiores invenções do mundo, o Cubo de Rubik não teve um começo fácil. Depois de mostrar o protótipo para seus alunos e amigos, Ernő começou a perceber o potencial de seu cubo. O próximo passo era fabricá-lo. Os primeiros cubos (Figura 2) foram produzidos e distribuídos na Hungría pela Politechnika. Estes primeiros cubos, comercializados como “Cubo Mágico”, eram o dobro do peso dos que seriam fabricados mais tarde. Nos anos 70, a Hungria fazia parte do regime comunista e qualquer tipo de importação ou exportação era altamente controlado. Como que a invenção de Ernő, que se tornou um grande sucesso na Hungria, acabaria nas mãos das crianças dos anos 80? (RUBIK, 2019).

Figura 3 – Primeiro modelo comercializado do Cubo Mágico em 1977.

Fonte – (RUBIK, 1977).

O primeiro passo para o cubo ganhar reconhecimento mundial seria ser exportado da Hungria. Um parte disso foi feita pelos matemáticos que levavam o cubo para conferências internacionais e a outra parte por um empresário húngaro que levou o cubo para a feira de brinquedos de Nuremberg em 1979 (Figura 4). Foi lá que Tom Kremer, um especialista em brinquedos, concordou em vendê-lo para o resto do mundo. A confiança implacável de Tom sobre o cubo finalmente resultou na "Ideal Toy Company" assumindo distribuição do Cubo Mágico (Figura 3). Os executivos da Ideal Toy acharam que o nome possuía conotações de bruxaria, e depois de passar por diversas possibilidades, o nome "Cubo de Rubik" foi decidido, e o ícone nasceu. (RUBIK, 2019).

Figura 4 – Feira internacional de brinquedos realizada na Alemanha em 1979.

Fonte – (RUBIK, 1979).

Desde o seu lançamento internacional em 1980, **estima-se que foram vendidos mais de 350 milhões de cubos**. Aproximadamente uma a cada sete pessoas já brincaram com o quebra-cabeça. Este pequeno cubo de seis cores passou a representar uma década. Ele apareceu em obras de arte, vídeos famosos, filmes de Hollywood e **até teve o seu próprio programa de TV**, ele representava tanto genialidade quanto confusão, **deu início a um novo esporte (speedcubing)**, e já até foi para o espaço. (RUBIK, 2019).

A beleza do Cubo de Rubik é que quando você vê um embaralhado, você sabe o que exatamente precisa fazer, sem alguma instrução. Porém, sem instrução é quase impossível de se resolver, fazendo com que ele seja umas das invenções mais frustrantes e viciantes já produzidas. (RUBIK, 2019).

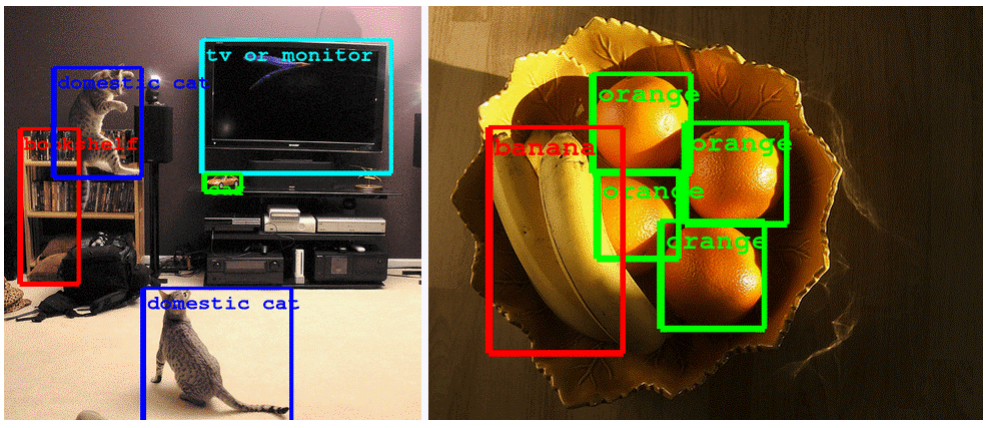
2.2 Visão Computacional

Segundo SAS (2019) as primeiras experiências em visão computacional aconteceram nos anos 1950, com o uso de algumas das primeiras redes neurais para detectar os limites de um objeto e para classificar objetos simples em categorias como círculos e quadrados. Nos anos 1970, o primeiro uso comercial de visão computacional interpretou textos manuscritos e digitados usando reconhecimento ótico de caracteres. Esse avanço tinha como objetivo interpretar textos escritos para deficientes visuais.

Com o amadurecimento da internet nos anos 90, grandes volumes de imagens foram disponibilizados on-line para análises e o desenvolvimento de programas de reconhecimento facial explodiu. Esses crescentes conjuntos de dados ajudaram a possibilitar que máquinas identifiquem pessoas específicas em fotos e vídeos. SAS (2019).

Hoje, inúmeros fatores convergiram para reanimar as pesquisas em visão computacional, como por exemplo: o poder computacional tornou-se mais barato e compreensível, tecnologias móveis com câmeras embutidas saturaram o mundo com fotos e vídeos, *hardwares* projetados para visão computacional e suas análises estão mais disponíveis, novos algoritmos como redes neurais convolucionais podem aproveitar melhor as capacidades dos *hardwares* e *softwares*, dentre outros. SAS (2019). A figura 5 demonstra exemplo de algoritmo utilizando técnicas de visão computacional.

Figura 5 – Aplicações da visão computacional.



Fonte – ACADEMY (2019)

Os efeitos desses avanços no campo da visão computacional têm sido surpreendentes. As taxas de precisão de identificação e classificação de objetos foram de 50 para 90% em menos de uma década e os sistemas de hoje são ainda mais precisos que seres humanos nas rápidas detecções e reação a estímulos visuais. SAS (2019).

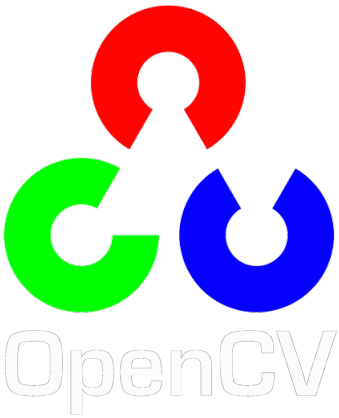
A visão computacional lembra um quebra-cabeça, pois um computador monta conteúdos visuais da mesma maneira que você monta um quebra-cabeça.

Pense em como você monta um quebra-cabeça. Você tem todas as peças e precisa montá-las em uma imagem: é assim que as redes neurais funcionam para a visão computacional. Elas distinguem partes diferentes da imagem, identificam seus limites e modelam os subcomponentes. Usando filtros e uma série de ações através de camadas de rede profundas, elas podem juntar todas as partes da imagem assim como você faria com um quebra-cabeça. O computador não olha para a imagem final na caixa de quebra-cabeça mas, muitas vezes, é alimentado com centenas ou milhares de imagens relacionadas para treiná-lo a reconhecer objetos específicos. SAS (2019).

Em vez de treinar computadores para procurar por bigodes, rabos e orelhas pontudas na intenção de reconhecer um gato, os programadores fazem upload de milhões de fotos de gatos e, em seguida, o modelo aprende por conta própria as diferentes características de um gato. SAS (2019).

2.3 OpenCV

Figura 6 – Logo OpenCV.



Fonte – OPENCV (2019)

OpenCV (Figura 6) é uma biblioteca de *software* de visão computacional e aprendizado de máquina de código aberto. O OpenCV foi desenvolvido para fornecer uma infraestrutura comum para aplicativos de visão computacional e acelerar o uso da percepção da máquina nos produtos comerciais. Por ser um produto licenciado pela BSD, o OpenCV facilita para as empresas e desenvolvedores em geral a utilização e modificação do código. OPENCV (2019).

A biblioteca possui mais de 2500 algoritmos otimizados, que incluem um conjunto

abrangente de algoritmos clássicos e avançados de visão computacional e aprendizado de máquina. Esses algoritmos podem ser usados para detectar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, rastrear movimentos de câmeras, rastrear objetos em movimento, extrair modelos 3D de objetos, produzir nuvens de pontos 3D a partir de câmeras estéreo, costurar imagens para produzir uma alta resolução imagem de uma cena inteira, encontre imagens semelhantes de um banco de dados de imagens, remova olhos vermelhos de imagens tiradas com *flash*, siga os movimentos dos olhos, reconheça o cenário e estabeleça marcadores para sobrepor à realidade aumentada etc. O OpenCV tem uma comunidade de mais de 47 mil pessoas e número estimado de *downloads* superiores a 18 milhões. A biblioteca é amplamente utilizada em empresas, grupos de pesquisa e órgãos governamentais. OPENCV (2019).

Figura 7 – Grandes empresas que utilizam o OpenCV.

Fonte – Do autor (2019).

Juntamente com empresas bem estabelecidas (Figura 7), como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda e Toyota, que empregam a biblioteca, existem muitas *startups*, como*Applied Minds*, *VideoSurf* e Zeitera, que fazem uso extensivo do OpenCV. Os usos implementados do OpenCV abrangem o alcance de costurar imagens de *streetview*, detectar invasões em vídeos de vigilância em Israel, monitorar equipamentos de minas na China, ajudar robôs a navegar e pegar objetos na *Willow Garage*, detectar acidentes de afogamento de piscinas na Europa, executar arte interativa Espanha e Nova York, verificando pistas em busca de detritos na Turquia, inspecionando rótulos de produtos em fábricas ao redor do mundo para detecção rápida de faces no Japão. OPENCV (2019).

Possui interfaces C ++, Python, Java e MATLAB e suporta Windows, Linux, Android

e Mac OS. O OpenCV se inclina principalmente para aplicativos de visão em tempo real e tira proveito das instruções MMX e SSE, quando disponíveis. As interfaces CUDA e OpenCL com todos os recursos estão sendo desenvolvidas ativamente agora. Existem mais de 500 algoritmos e cerca de 10 vezes mais funções que compõem ou suportam esses algoritmos. O OpenCV é escrito nativamente em C ++ e possui uma interface de modelo que funciona perfeitamente com contêineres STL. OPENCV (2019).

2.4 Python

Figura 8 – Logo do Python.

Fonte – PYTHON (2019)

Segundo o site BRASIL (2019), Python (Figura 8) é uma linguagem de programação criada por Guido van Rossum em 1991. Os objetivos do projeto da linguagem eram: produtividade e legibilidade. Em outras palavras, Python é uma linguagem que foi criada para produzir código bom e fácil de manter de maneira rápida. Entre as características da linguagem que ressaltam esses objetivos estão:

- O uso de identação para marcar blocos.
- Quase nenhum uso de palavras-chave voltadas para a compilação.
- Coletor de lixo para gerenciar automaticamente o uso da memória.
- Baixo uso de caracteres especiais, o que torna a linguagem muito parecida com pseudo-código executável.

Além disso, Python suporta múltiplos paradigmas de programação. A programação procedimental pode ser usada para programas simples e rápidos, mas estruturas de dados complexas, como tuplas, listas e dicionários, estão disponíveis para facilitar o desenvolvimento de algoritmos complexos. Grandes projetos podem ser feitos usando técnicas de orientação a objetos, que é completamente suportada em Python (inclusive sobrecarga de operadores e herança

múltipla). Um suporte modesto para programação funcional existe, o que torna a linguagem extremamente expressiva: é fácil fazer muita coisa com poucas linhas de comando. E também possui inúmeras capacidades de meta-programação: técnicas simples para alterar o comportamento de comportamentos da linguagem, permitindo a criação de linguagens de domínio específico. PYTHON (2019).

Python tem uma biblioteca padrão imensa, que contém classes, métodos e funções para realizar essencialmente qualquer tarefa, desde acesso a bancos de dados a interfaces gráficas com o usuário. E, logicamente, já que esse é o objetivo deste grupo, existem muitas ferramentas para lidar com dados científicos. Essa característica da linguagem é comumente chamado baterias inclusas, significando que tudo que você precisa para rodar um programa está na maior parte das vezes presente na instalação básica. PYTHON (2019).

Por fim, e não menos importante, Python é uma linguagem livre e multiplataforma. Isso significa que os programas escritos em uma plataforma serão executados sem nenhum problema na maioria das plataformas existentes sem nenhuma modificação. E, caso a plataforma objetivo não tenha uma versão de Python, desenvolvedores têm a liberdade de estudar e modificar o código da linguagem para fazer com que ela rode onde quer que seja. PYTHON (2019).

2.4.1 Por que Python?

É fácil ver que a linguagem tem facilidades incríveis para uso geral. A pergunta é: por que Python é a linguagem ideal para aplicações científicas? As respostas são muitas, mas podemos resumir algumas aqui. A primeira razão, e provavelmente a principal, é: Python é uma linguagem expressiva, em que é fácil traduzir o raciocínio em um algoritmo. Em aplicações científicas, o raciocínio é essencialmente complicado essa é a natureza das ciências. É um problema adicional para o cientista ter que se preocupar com, além do assunto básico de sua pesquisa, a correção do programa em detalhes pouco relevantes: alocação de memória, gerenciamento de recursos, etc. Python faz isso tudo automaticamente de maneira muito eficiente, permitindo ao cientista se concentrar exclusivamente no problema sendo estudado. BRASIL (2019).

Python é extremamente legível. Isso significa que é muito fácil compreender programas escritos há algum tempo. É muito comum que os programas em atividades científicas sejam criados a partir da evolução de algoritmos anteriores. Portanto, é extremamente importante ser capaz de entender o que foi feito antes. Uma vez que as palavras-chave da linguagem Python

são voltadas para a estruturação dos programas (e não para indicar ao computador como compilar ou interpretar trechos de código), não existem trechos de código que são inúteis para o raciocínio. BRASIL (2019).

Python tem uma comunidade ativa e vibrante, espalhada por todo o mundo. E, sendo uma linguagem livre, todos os seus usuários estão dispostos a contribuir (este site é um exemplo disso, já que todos seus contribuidores são voluntários). Isso faz com que a documentação seja abundante e existam módulos para executar virtualmente qualquer tarefa necessária. Isso é importante: não há tempo para reinventar a roda, então poder contar com módulos prontos é ótimo. Mas, mais que isso, uma vez que os programas em Python são distribuídos na forma de código-fonte, qualquer pessoa pode alterar, corrigir e melhorar os algoritmos. Isso faz com que os módulos sejam maduros e seguros, testados contra diversas situações e diversas vezes. A robustez alcançada é um fator importante. BRASIL (2019).

Python é, além disso, uma linguagem de propósito geral. Muitas vezes, é necessário lidar com tarefas laterais: buscar dados em um banco de dados remoto, ler uma página na internet, exibir graficamente os resultados, criar uma planilha, etc. Linguagens de cunho especificamente científico têm um sério problema aí, mas, uma vez que Python é utilizada em praticamente todo tipo de tarefa, encontram-se módulos prontos para realizar essas tarefas que podem ser tornar complicadas. Novamente, é uma preocupação a menos para quem está desenvolvendo aplicações científicas. BRASIL (2019).

Por esses e ainda outros motivos, Python tem conquistado uma grande popularidade entre a comunidade científica. É uma linguagem simples que dá conta do recado e não fica entre o cientista e a resolução do seu problema. Essa frase provavelmente resume todos os motivos necessários para sua utilização. BRASIL (2019).

2.5 O algoritmo de Kociemba

A descrição a seguir tem como objetivo fornecer uma idéia básica de como o algoritmo funciona.

As 6 faces diferentes do cubo são chamadas U (p), D (own), R (direita), L (eft), F (ront) e B (ack). Enquanto U indica um quarto de virada para cima na face de 90 graus no sentido horário, U2 indica uma volta de 180 graus e U ’indica um quarto de volta de 90 graus no sentido anti-horário. Uma sequência como UDR’D2 do cubo se move é chamada de manobra. KOCIEMBA (2011).

Se você virar as faces de um cubo resolvido e não usar os movimentos R, R', L, L', F, F', B e B', você somente gerará um subconjunto de todos os cubos possíveis. Esse subconjunto é indicado por G1 = <U, D, R2, L2, F2, B2>. Nesse subconjunto, as orientações dos cantos e arestas não podem ser alteradas. Ou seja, a orientação de uma aresta ou canto em um determinado local é sempre a mesma. E as quatro arestas na fatia UD (entre a face U e a face D) permanecem isoladas nessa fatia. KOCIEMBA (2011).

Na fase 1, o algoritmo procura manobras que transformarão um cubo embaralhado em G1. Ou seja, as orientações dos cantos e arestas devem ser restringidas e as arestas da fatia UD devem ser transferidas para essa fatia. Nesse espaço abstrato, um movimento apenas transforma um triplo (x, y, z) em outro triplo (x', y', z'). Todos os cubos de G1 têm o mesmo triplo (x0, y0, z0) e este é o estado do objetivo da fase 1. KOCIEMBA (2011).

Para encontrar esse estado de objetivo, o programa usa um algoritmo de busca chamado de aprofundamento iterativo A* com uma função heurística de limite inferior (IDA *). No caso do cubo, isso significa que ele repete todas as manobras de comprimento crescente. A função heurística h1 (x, y, z) estima para cada estado do cubo (x, y, z) o número de movimentos necessários para alcançar o estado do objetivo. É essencial que a função nunca superestime esse número. No Cube Explorer 2, ele fornece o número exato de movimentos necessários para alcançar o estado do objetivo na Fase 1. A heurística permite a remoção durante a geração das manobras, o que é essencial se você não quiser esperar muito, muito tempo antes o estado do objetivo é alcançado. A função heurística h1 é uma tabela de pesquisa baseada em memória e permite a remoção de até 12 movimentos com antecedência. KOCIEMBA (2011).

Na fase 2, o algoritmo restaura o cubo no subgrupo G1, usando apenas movimentos desse subgrupo. Restaura a permutação dos 8 cantos, a permutação das 8 arestas das faces U e D e a permutação das 4 arestas da fatia UD. A função heurística h2 (a, b, c) estima apenas o número de movimentos necessários para alcançar o estado do objetivo, porque existem muitos elementos diferentes em G1. KOCIEMBA (2011).

O algoritmo não para quando uma primeira solução é encontrada, mas continua a procurar soluções mais curtas executando a fase 2 a partir de soluções abaixo do ideal da fase 1. Por exemplo, se a primeira solução tiver 10 movimentos na fase 1 seguidos por 12 movimentos na fase 2, a segunda solução pode 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 algoritmo para. KOCIEMBA (2011).

Na implementação atual, o algoritmo bifásico não procura algumas soluções ideais em geral, aquelas que devem entrar e sair da fase 2. Isso aumenta a velocidade consideravelmente. KOCIEMBA (2011).

O algoritmo que fornece uma solução ótima no sentido de que não existe uma solução mais curta é chamado algoritmo de Deus. Neste trabalho foi utilizado o algoritmo de Hebert Kociemba por ser o mais rápido no que se refere a tempo de execução. Existem posições do cubo (por exemplo, o superflip que vira todas as 12 arestas), que são conhecidas por terem um comprimento de manobra mais curto de 20 movimentos a serem resolvidos. Após 30 anos, foi finalmente demonstrado em julho de 2010 que todas as posições do cubo podem ser resolvidas em 20 movimentos ou menos. KOCIEMBA (2011).

2.6 Porta Serial

Figura 9 – Conectores serial e USB.

Fonte – AWSLI (2019)

A comunicação serial na placa Arduino é um poderoso recurso que possibilita a comunicação entre a placa utilizada e um computador ou entre a placa e outro dispositivo, como por exemplo um módulo GPS ou um módulo GSM. É através desse canal que é realizado o upload do código para a placa. EMBARCADOS (2014).

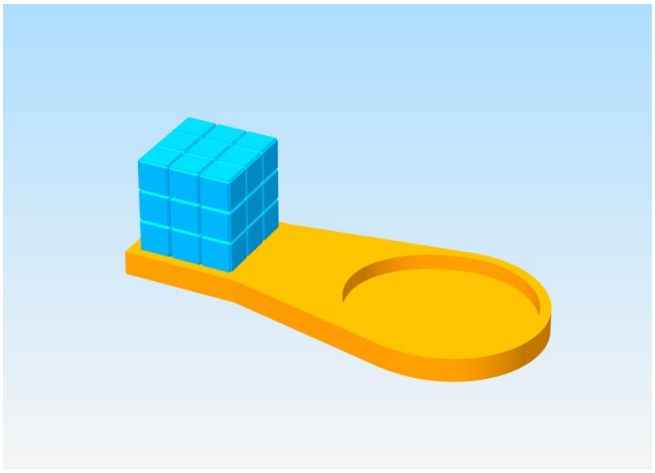
Na interface serial, os bits são transferidos em fila, ou seja, um bit de dados de cada vez. Esta interface é comunmente utilizada para conectar dispositivos como impressoras, scanners, modems, câmeras, e tantos outros equipamentos ao computador. As portas seriais são bidirecionais, ou seja, elas permitem envio e recebimento de dados. É um meio de cabeamento simples e de fácil utilização. NOTEPLACE (2017).

2.7 Modelagem 3D

RODRIGUES (2018) do site Mundo do Desenho Digital, descreve a modelagem 3D como o processo de se elaborar um dado objeto com 3 dimensões através de *softwares* com especificidade para este fim. Esta técnica possibilita a simulação de objetos dos mais diversos tipos sendo aplicada a diversas áreas, como cinema, jogos, arquitetura e ilustrações.

Para desenvolver a criação de qualquer cena 3D, por meio de modelagem em computadores, é importante ter em mente a sua aplicação, a complexidade e o estilo desejados. Esta cena pode ser uma simulação da realidade ou uma cena estilizada como de filmes para crianças que utilizam o estilo *cartoon*, por exemplo. RODRIGUES (2018). A figura 10 demonstra um modelo 3D desenvolvido pelos autores neste trabalho de conclusão de curso.

Figura 10 – Modelo 3D da base de leitura do cubo mágico, usada no primeiro protótipo apresentado na feira de robótica.



Fonte – Do autor (2019).

Apesar de a princípio parecer algo complexo e difícil de fazer, quando se utiliza o *software* adequado a modelagem 3D ou o desenho feito em 3 dimensões torna-se algo simples que qualquer um pode aprender. RODRIGUES (2018).

Quando falamos em 3D a maioria das pessoas já associa aos filmes e desenhos animados que estão extremamente realistas nos últimos tempos. Entretanto, não é só para isso que a modelagem 3D é utilizada. As possibilidades de aplicação da modelagem digital ou modelagem 3D é muito ampla. Ela pode ser usada como *hobby* e de forma profissional em muitos segmentos como: projeto de móveis, *design* de calçados, *design* de joias, *design* de interiores, design de produtos, engenharia, arquitetura e muitos outros. RODRIGUES (2018).

2.8 Corel Draw

Segundo a CORPORATION (2019), desenvolvedora do *software*, CorelDRAW é um programa de desenho bidimensional para aplicação em *design* gráfico desenvolvido pela Corel Corporation, localizada no Canadá. É um aplicativo de ilustração que possibilita a criação e a manipulação de vários produtos, como por exemplo: desenhos artísticos, publicitários, logotipos, capas de revistas, livros, etc.

O CorelDRAW permite ao usuário criar e diversificar imagens já criadas com compatibilidade com vários formatos de arquivos de imagem incluindo AI, PSD, PDF, JPG, PNG, SVG, DWG, DXF, EPS, TIFF, entre outros. CORPORATION (2019).

O CorelDRAW facilita a preparação de documentos para impressão. Com um poderoso mecanismo de gerenciamento de cores, você pode controlar a consistência das cores em diferentes mídias e certificar-se da exatidão das cores antes da impressão. CORPORATION (2019).

Também traz em sua biblioteca conteúdo atraente para a *Web* com uma coleção de predefinições e ferramentas gráficas para a *Web*. Com o recurso de publicação no *WordPress*, você pode carregar vários formatos de imagem diretamente no seu *site WordPress*. CORPORATION (2019). A figura 11 demonstra um desenho realizado no Corel Draw pelo autor Isaque de Almeida Costa.

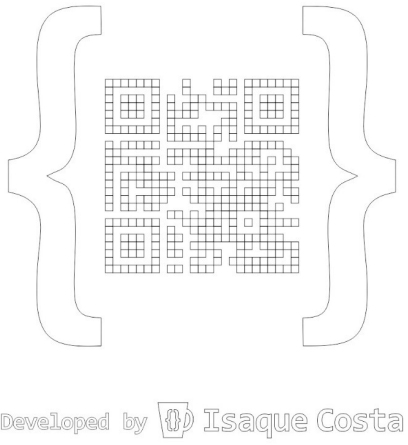
Figura 11 – Arte feita utilizando Corel Draw.

Fonte – Do autor (2019).

2.9 DXF

Segundo KONDRASOVAS (2016), arquivos DXF (*Drawing Exchange Format*) são os mais comuns na troca de desenhos e projetos de peças para serem utilizadas em *softwares* de CAM e máquinas de corte automáticas. São populares porque promovem a troca aberta destes arquivos entre programas desenvolvidos por diferentes empresas.

Figura 12 – Exemplo de um arquivo DXF.



Fonte – Do autor (2019).

2.10 Fusion 360

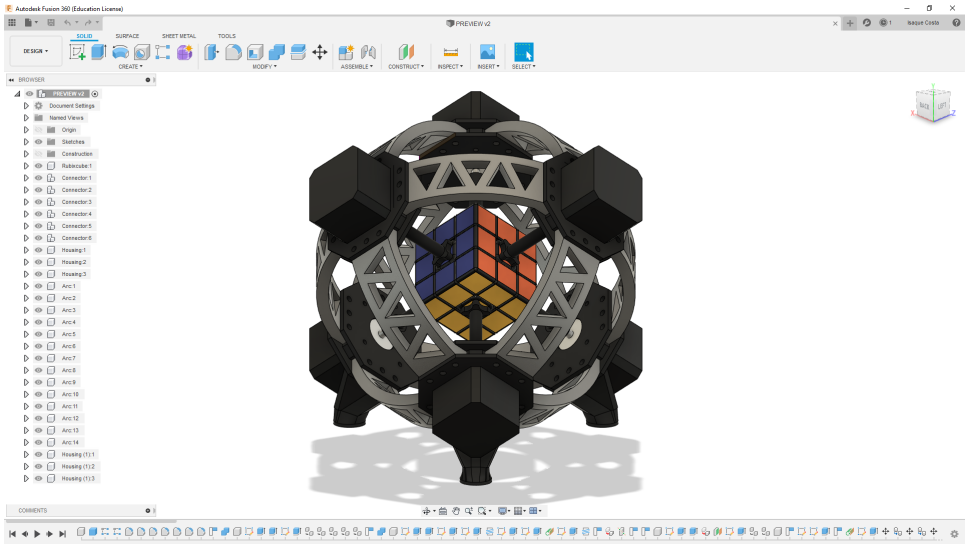
Segundo a AUTODESK (2019), *Fusion* 360 é um *software* de CAD, CAM e CAE 3D o modelado para a nuvem para modelos de projeto conceptual com formas livres ou técnicas de modelação de sólidos e para partilhá-los com outros colaboradores.

Leve como nenhum outro programa de seu nicho, o *Fusion* 360 é realmente um produto interessante. Ele é 100% integrado ao Autodesk 360 (serviço de *cloud computing da Autodesk*), permitindo que você guarde seus documentos na nuvem, compartilhe arquivos com amigos, trabalhe em projetos em equipe e faça renderizações *online* usando os servidores da empresa. AUTODESK (2019).

De acordo com Souza (2014), o que mais chama a atenção no *Fusion* 360, porém, é a forma como o programa consegue ser simultaneamente simples e poderoso, o *software* é realmente fácil de usar, mas sem abrir mão de ferramentas e recursos profissionais encontrados em utilitários mais tradicionais (como o modo de modelagem paramétrica e a possibilidade de aplicar propriedades físicas de um material em certo objeto).

A figura 13 demonstra o projeto final feito no Fusion360.

Figura 13 – Interface do Fusion360.



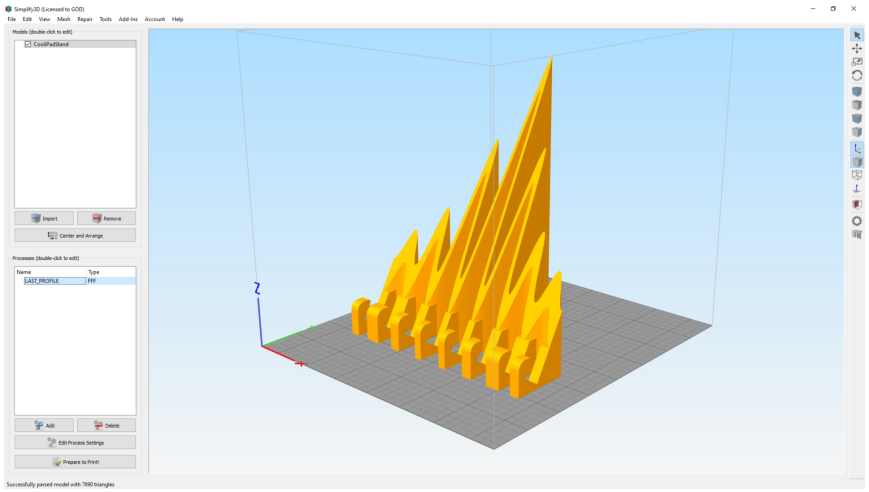
Fonte – Do autor (2019).

2.11 Simplify3D

Segundo a desenvolvedora, SIMPLIFY3D (2014) o *software* é um dos melhores e mais famosos *software* de fatiamento para impressoras 3D que existem no mercado atualmente. Ele converte modelos 3D em instruções claras para a impressora, melhorando assim a qualidade das impressões, uma simples atualização de *software* faz toda a diferença no mundo.

De fato, mais de 90% dos especialistas concordam que o *software* de impressão 3D tem o maior impacto na qualidade da impressão, ainda mais que a própria impressora 3D. SIMPLIFY3D (2014). A figura 14 demonstra a interface do software.

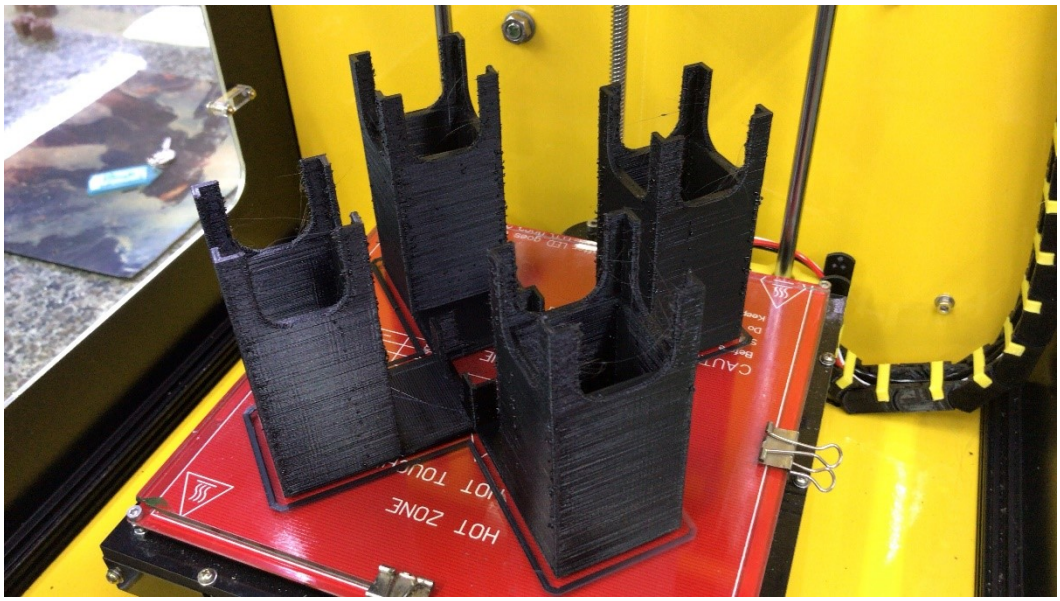
Figura 14 – Interface do *software* Simplify3D.



Fonte – Do autor (2019).

2.12 Impressão 3D

Figura 15 – Base do primeiro protótipo funcional ainda na cama da impressora 3D.



Fonte – Do autor (2019).

Acima pode ser visto o processo de impressão 3D concluído pelos autores, que segundo a AUTODESK (2019), grande empresa do mercado de *design* digital, é o processo pelo qual objetos físicos são criados pela deposição de materiais em níveis de camadas, embasando-se em um modelo digital pré-estabelecido. Todos os processos de impressão 3D requerem o trabalho conjunto de *software*, *hardware* e materiais para que o projeto seja concluído com êxito e qualidade.

A tecnologia de impressão 3D pode ser usada para criar tudo, de protótipos e peças simples a produtos finais altamente técnicos, como peças de avião, construções sustentáveis, implantes médicos que salvam vidas e até mesmo órgãos artificiais com o uso de camadas de células humanas. AUTODESK (2019).

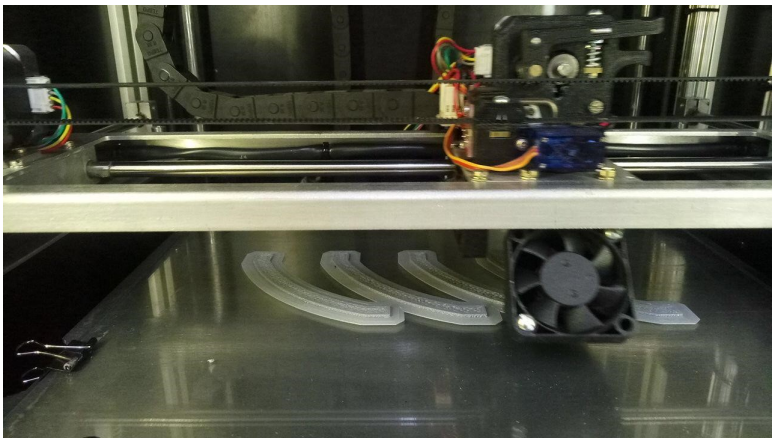
esta é uma tecnologia que avança rapidamente e está se tornando um recurso confiável para a fabricação em massa de peças. A cada dia, cientistas e inventores estão descobrindo novas maneiras de aplicá-la. AUTODESK (2019).

2.12.1 Fabricação de filamentos fundidos (FFF)

Também conhecida como modelagem por deposição de material fundido (FDM). Este método de impressão 3D aquece e efetua a extrusão de materiais plásticos depositando-os na cama de impressao(base de vidro ou metal aquecida para melhor aderência do filamento). Ele

é comum em impressoras 3D de pequeno porte e profissionais. Como exemplos de impressoras 3D temos a impressora da Faculdade Única (Figura 43). A figura 16 demonstra o processo FFF e a figura 17 os filamentos utilizados neste processo de impressão. AUTODESK (2019).

Figura 16 – Processo FFF de impressão 3D.



Fonte – Do autor (2019).

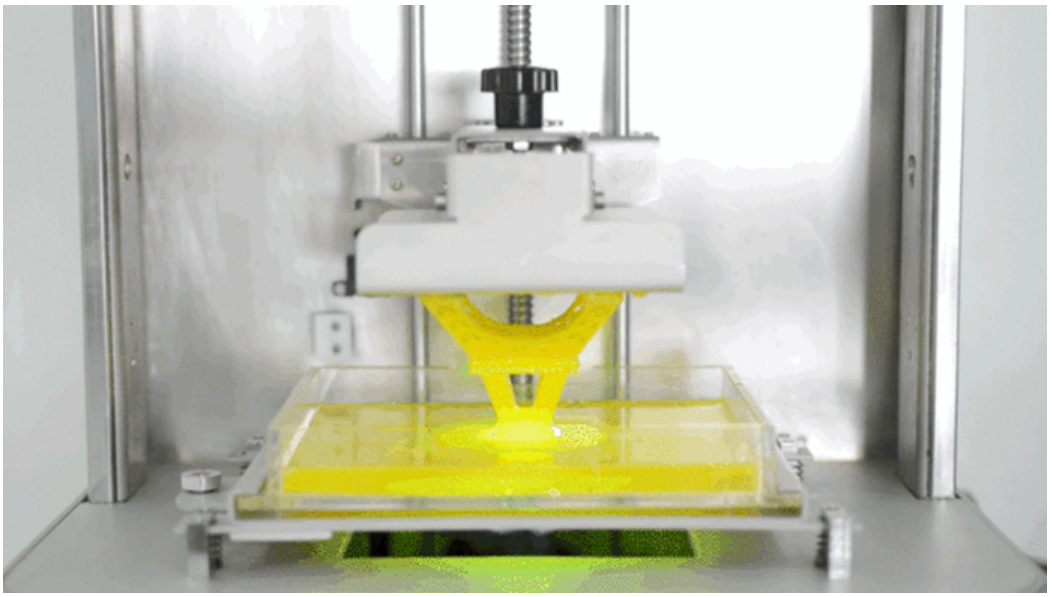
Figura 17 – Filamento preto feito de plástico ABS.

Fonte – Do autor (2019).

2.12.2 Estereolitografia (SLA, Stereolithography)

Este método de impressão 3D usa luz UV para curar ou enrijecer resinas, camada por camada. Como exemplos de impressoras 3D temos a Autodesk *Ember* e a Formlabs Form 1. A figura 18 mostra este processo em execução. AUTODESK (2019).

Figura 18 – Processo de estereolitografia de impressão 3D.



Fonte – http://s2.glbimg.com/JOj2XaizyeUqZQqCDE1CDqSqGjs=/695x0/s.glbimg.com/po/tt2/f/original/2016/02/25/make-x.png (2019).

2.12.3 Sinterização seletiva a laser (SLS, Selective laser sintering)

Comum na fabricação industrial, este método de impressão 3D usa laser para fundir materiais em pó, camada por camada. Como exemplos de fabricantes de impressoras 3D temos a EOS e a 3D *Systems*. A figura 19 e um exemplo de impressora 3D que utiliza este método, um equipamento como este começa a custar na faixa de USD 115.000,00. AUTODESK (2019).

Figura 19 – Impressora de sinterização seletiva a laser P100.

Fonte – https://res.cloudinary.com/engineering-com/ image/upload/$w_6$40,$h_6$40,$c_l imit/image$001$_w ibqjd.jpg$(2019).

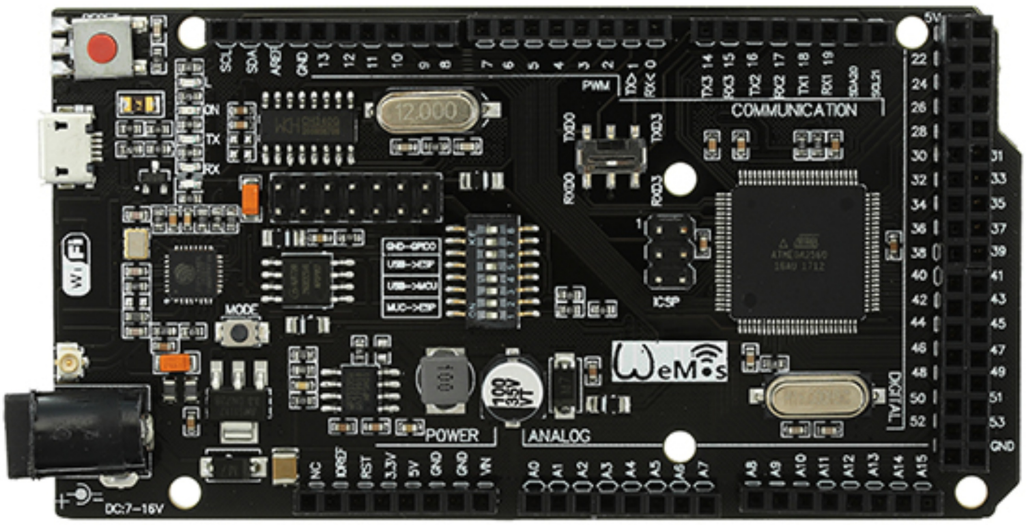
2.13 Arduíno MEGA 2560

Falando em termos práticos, as placas Arduino possuem funcionamento semelhante ao de um pequeno computador, no qual, pode-se programar a maneira como suas entradas e saídas devem se comportar em meio aos diversos componentes extErnős que podem ser conectados nas entradas I/O. KOYANAGI (2017a).

O Arduino é uma plataforma *open-source* de prototipagem eletrônica com *hardware* e *software* flexíveis e fáceis de usar, destinado a artistas, *designers*, hobbistas e qualquer pessoa interessada em criar objetos ou ambientes interativos. Ou seja, O Arduino é uma plataforma formada por dois componentes: A placa, que é o *Hardware* que usaremos para construir nossos projetos e a IDE Arduino, que é o *Software* onde escrevemos o que queremos que a placa faça. KOYANAGI (2017a).

A maior vantagem dessa plataforma de desenvolvimento sobre as demais é a sua facilidade de sua utilização, pois, pessoas que não são da área técnica podem aprender o básico e criar seus próprios projetos em um intervalo de tempo relativamente curto. A figura 20 demonstra a placa Arduino MEGA WiFi. KOYANAGI (2017a).

Figura 20 – Arduino MEGA WiFi.



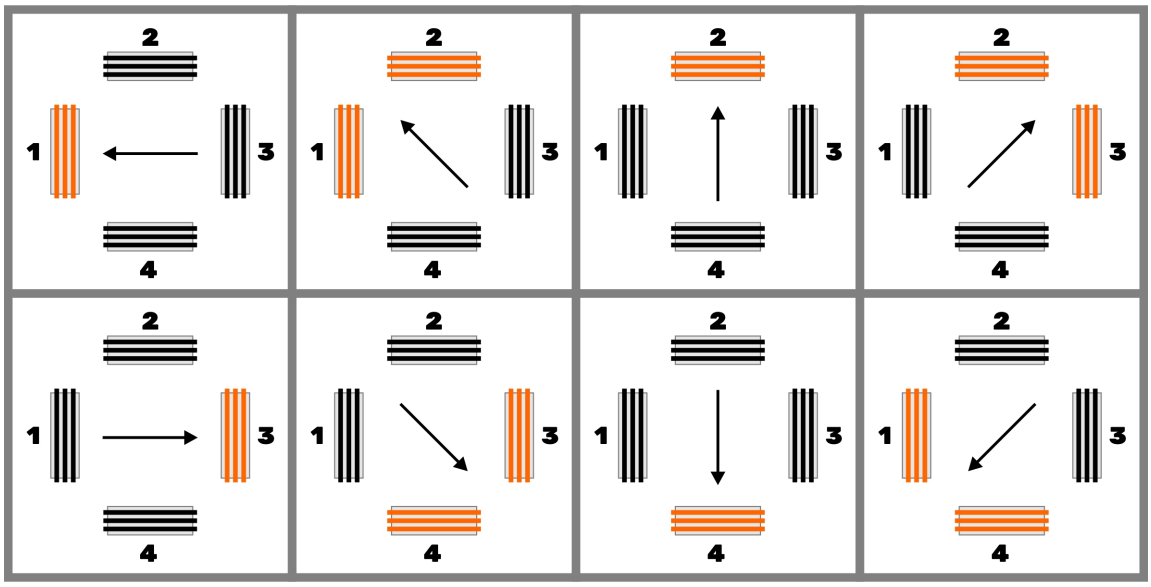
Fonte – Do autor (2019).

2.14 Motor de passo Nema 17

O motor de passo é um tipo de motor elétrico, especialmente projetado, que desloca um ângulo de 1,8º a cada pulso recebido do *drive* de controle. Ou seja, o motor de passo é usado quando se tem que posicionar algo de forma muito precisa ou rotacionada em um ângulo exato. KOYANAGI (2017b).

O número de passos que o motor gera é exatamente igual ao número de pulsos recebidos e a velocidade do motor é igual a frequência de entrada de pulsos. O motor de passo possui um imã muito forte e é controlado por uma série de campos eletromagnéticos que são ativados e desativados eletronicamente. Esses campos são produzidos por bobinas instaladas em torno de um motor. O funcionamento do motor de passo depende principalmente da topologia empregada. Os principais tipos de motor de passo são os de Relutância Variável, Imã Permanente e Híbrido. A figura 21 demonstra o funcionamento do motor de passo, que nada mais é do que um conjunto de imãs (geralmente 4) que ligam e desligam de forma a atrair o eixo do motor, fazendo-o rotacionar determinado ângulo (geralmente 1,8º). Cada acionamento de um conjunto de imãs que gera o movimento do eixo do motor é chamado de passo. KOYANAGI (2017b).

Figura 21 – Exemplificação do funcionamento do motor de passo.



Fonte – Do autor (2019).

Segundo PISCALED (2019b) explica, que o motor de passo Nema 17 (Figura 22) como sendo muito utilizado em máquinas pequenas e de precisão como Impressoras 3D e Router CNC. Possui um torque de 4,2 kgf, sendo que utiliza até 1,7A por fase e opera com tensão de 12V.

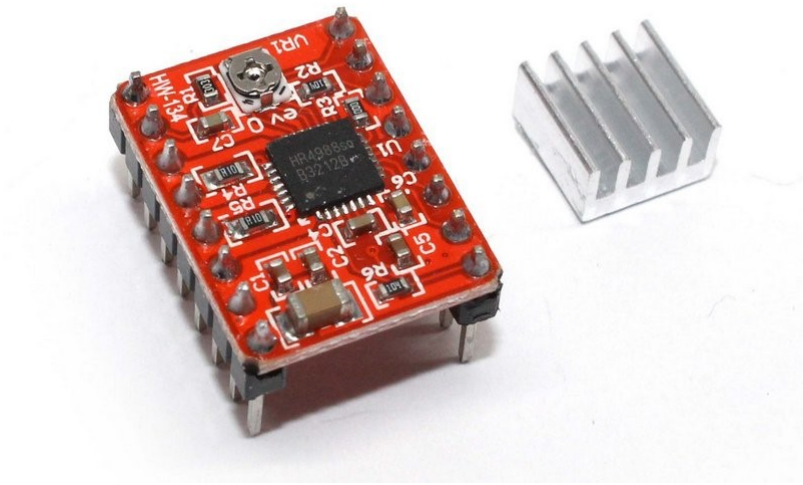
Figura 22 – Motor de passo Nema 17.

Fonte – Do autor (2019).

2.15 Driver A4988

A loja PISCALED (2019a), descreve o *driver* A4988 (Figura 23), como capaz de controlar, com micropassos, um motor de passo bipolar. É um módulo útil para o controle da robótica e mecânica, excelente custo benefício para estudantes, *hobbystas* e iniciantes da área de robótica/eletrônica.

Figura 23 – Driver de motor de passo A4988.



Fonte – Do autor (2019).

2.16 HTML

HTML é uma das linguagens que utilizamos para desenvolver *websites*. O acrônimo HTML vem do inglês e significa *Hypertext Markup Language* ou em português Linguagem de Marcação de Hipertexto. EIS (2011).

O HTML é a liguagem base da internet. Foi criada para ser de fácil entendimento por seres humanos e também por máquinas, como por exemplo o Google ou outros sistemas que percorrem a internet capturando informação. EIS (2011).

Tim Berners-Lee. Esse é o nome do homem que criou o HTML. Ele criou o HTML para a comunicação e disseminação de pesquisas entre ele e seu grupo de colegas. O HTML ficou bastante conhecido quando começou a ser utilizada para formar a rede pública daquela época, o que se tornaria mais tarde a internet que conhecemos hoje. EIS (2011).

O HTML é uma linguagem baseada em marcação. Nós marcamos os elementos para mostrar quais informações a página exibe. Por exemplo, um título importante. Aquele título do artigo, da manchete do *site*, nós marcamos com uma tag/elemento chamado H1. Segue exemplo:

Figura 24 – Estrutura básica do HTML.

```
<!DOCTYPE html>

<html>

<head>
<meta charset="utf-8">
<title>Title here</title>
</head>

<body>
Page content goes here.
</body>

</html>
```

Fonte – Do autor (2019).

Utilizando as tags, nós dizemos para o navegador o que é cada informação. O que é um título, o que é um parágrafo, o que é um botão, um formulário etc. Dizemos também o que é cada coisa para os sistemas de busca, como o Google. O Google, nesse caso, para exibir os resultados de busca, ele precisa saber o que é um parágrafo e o que é um título. Ele sabe disso através das tags. EIS (2011).

3 METODOLOGIA

O primeiro passo deste trabalho foi a pesquisa por outros projetos similares já desenvolvidos por pessoas da área. Foi adotado como exemplo um projeto que segundo Flatland (2016) manteve por um certo período o título de "O robô mais rápido a resolver o Cubo Mágico".

Segundo Gil (2002, p. 41) a pesquisa exploratória tem como objetivo “proporcionar maior familiaridade com o problema, com vista a torná-lo mais explícito ou a construir hipóteses”. Portanto, podemos classificar este trabalho como tal.

Pode-se afirmar que o presente trabalho possui caráter bibliográfico e documental, pois para sua fundamentação foram investigados artigos, livros, revistas e redes eletrônicas dos principais conceitos e práticas associados ao tema.

Utilizou-se uma abordagem qualitativa através do procedimento técnico de levantamento. Pois, segundo Gil (2002) , este procedimento envolve a interrogação direta das pessoas cujo comportamento se deseja conhecer. Que neste caso, se limita apenas aos autores e orientadores deste trabalho.

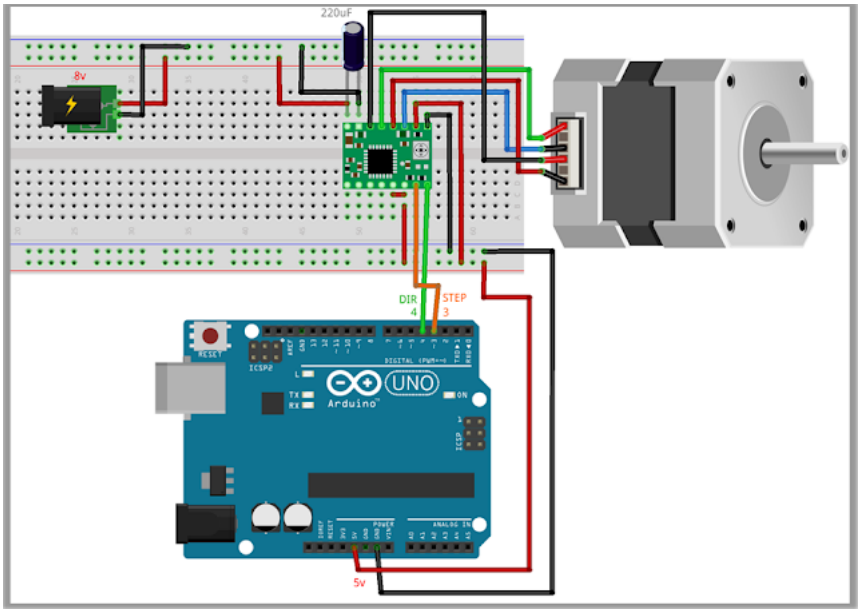
Os resultados serão demonstrados através de imagens de forma qualitativa, pois este trabalho busca o aprimoramento profissional no que se refere ao conhecimento adquirido durante o seu desenvolvimento e também a reação do público ao ver o robô solucionando o cubo.

4 DESENVOLVIMENTO

4.1 Circuito

O circuito final foi desenvolvido a partir de um modelo apresentado por KOYANAGI (2017b) que utiliza o *driver* A4988, pois segundo ele, "utilizando este *driver* você comanda com precisão e com força o motor de passo". Justamente o que é necessário para se mover o Cubo Mágico exatos 90º para ambos os sentidos. Logo no primeiro teste obteve-se sucesso. A figura 25 apresenta o circuito modelo usado para testes do motor NEMA 17.

Figura 25 – Circuito modelo, usado como base.

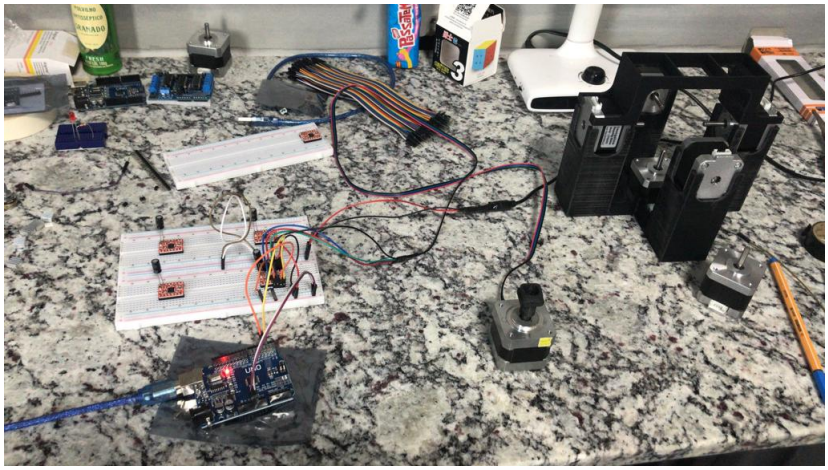


Fonte – KOYANAGI (2017b).

O próximo passo foi aprofundar os conhecimentos sobre a eletrônica, que segundo o Mundo (2019), nada mais é do que o controle do movimento dos elétrons, ou melhor dizendo, o controle da corrente elétrica. Tendo como base os conhecimentos adquiridos foi desenvolvido um circuito mais robusto, para controlar todos os 6 motores de passo.

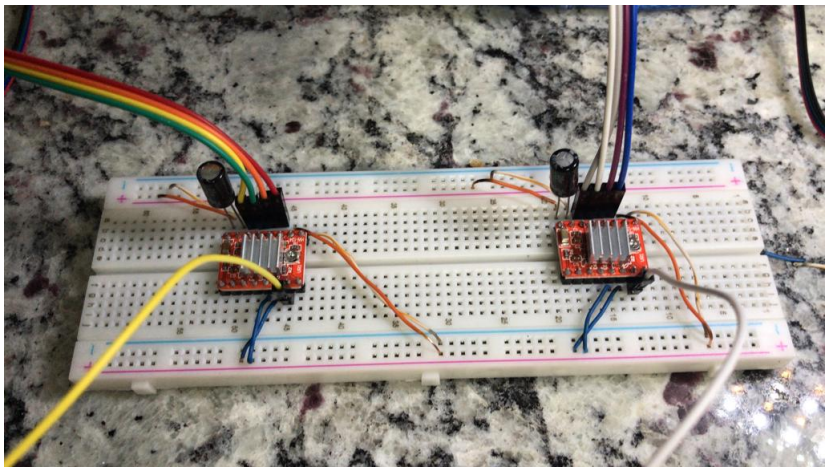
As figuras abaixo demonstram as etapas do processo de montagem do circuito deste trabalho de conclusão de curso:

Figura 26 – Primeiro teste com o circuito modelo, para apenas um motor.



Fonte – Do autor (2019).

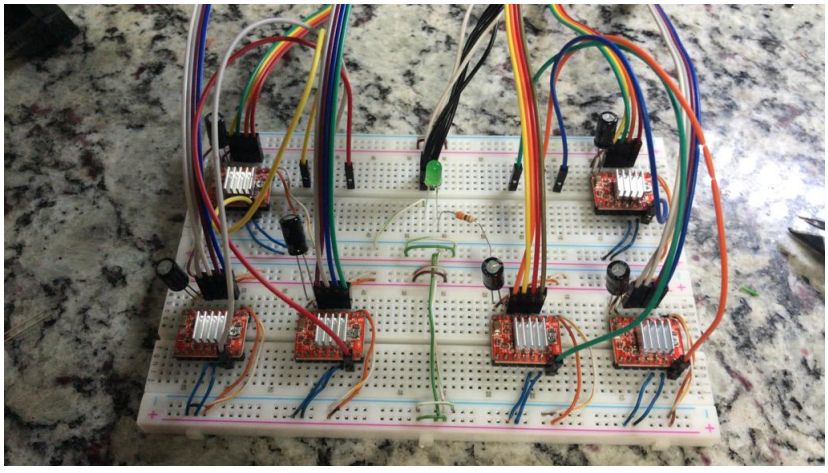
Figura 27 – Primeiro teste de duplicação do circuito modelo.



Fonte – Do autor (2019).

Após o sucesso, ao duplicar o circuito modelo, foi desenvolvido este circuito da figura abaixo para controlar os 6 motores.

Figura 28 – Primeiro circuito feito para controle de 6 motores



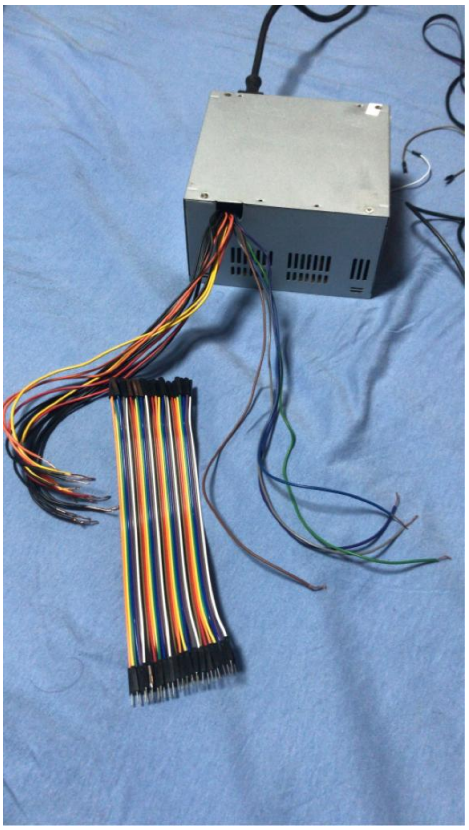
Fonte – Do autor (2019).

Para fazer a alimentação dos drivers, motores e demais componentes do circuito de forma segura e eficiente, foi necessário uma fonte robusta e com diversos mecanismos de proteção. A opção mais viável foi uma fonte ATX de 200W, apresentada nas figuras 29,30 e 31.

Figura 29 – Fonte utilizada para alimentar o circuito.

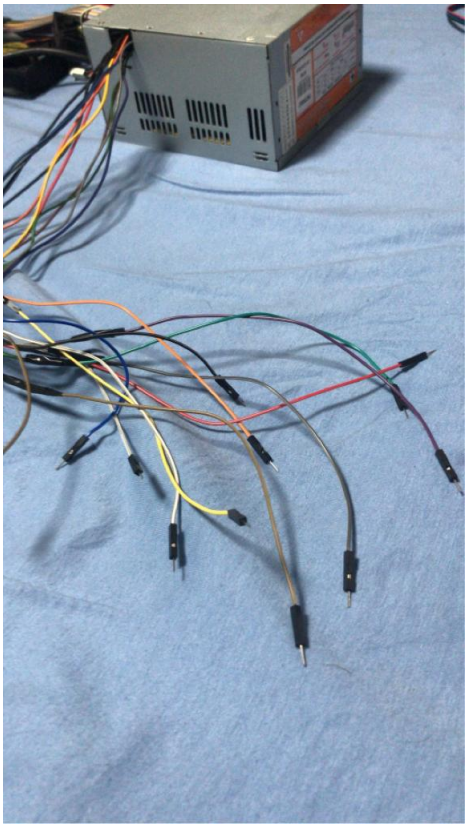
Fonte – Do autor (2019).

Figura 30 – Processo de adaptação da fonte para alimentar a protoboard.



Fonte – Do autor (2019).

Figura 31 – Fonte de adaptada.



Fonte – Do autor (2019).

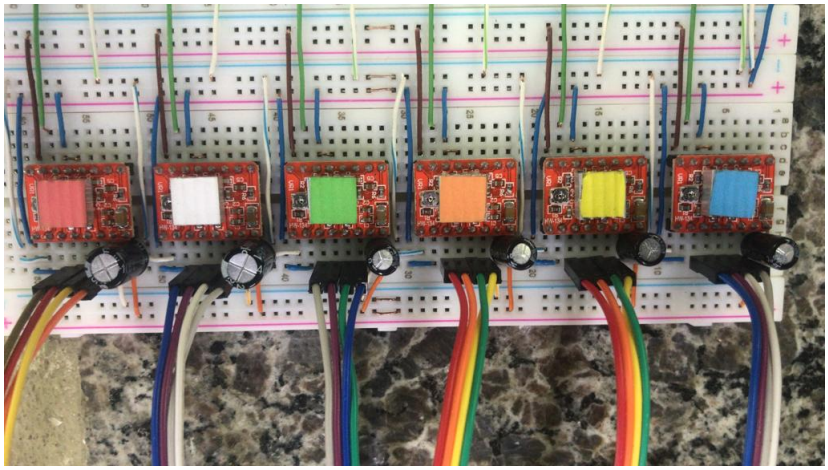
Foram impressos adesivos para indentificar os motores correspondentes a cada face do Cubo Mágico, facilitando o desenvolvimento e melhorando o visual do projeto, como ilustrado na figura 32.

Figura 32 – Identificação dos motores.

Fonte – Do autor (2019).

A figura 33, apresenta a reorganização dos *drivers* para melhor aproveitamento do espaço na protoboard.

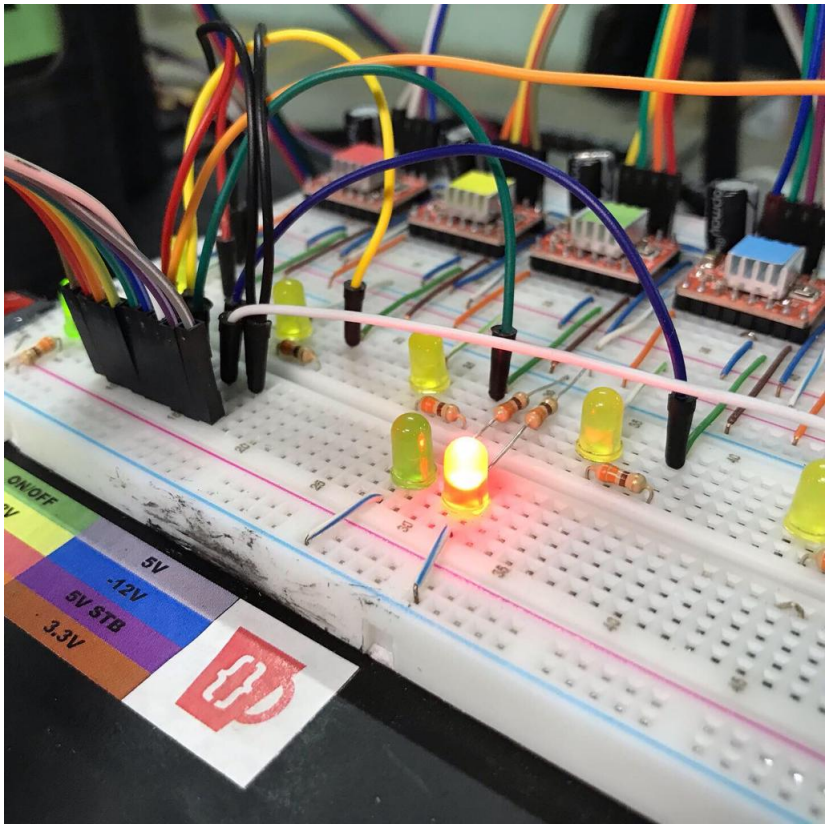
Figura 33 – Reorganização dos *drivers* na protoboard.



Fonte – Do autor (2019).

A figura 34, apresenta o primeiro circuito 100% funcional, utilizado na apresentação da feira de robótica e IA na Faculdade Única em Junho de 2019.

Figura 34 – Primeiro circuito finalizado, utilizado no primeiro protótipo.



Fonte – Do autor (2019).

Os componente eletrônicos utilizados foram:

- 1 LED verde para indicar o estado do circuito ON/OFF.
- 6 conectores, para conectar os motores ao circuito.

- 6 capacitores de 220uF, para armazer a energia necessária para dar o *start* nos motores.
- 6 *drivers* A4988 para controlar os motores de passo, como citado acima.
- 6 LEDs amarelos para indicar quando o comando for enviado do arduino para o *driver*.
- 1 LED verde para indicar o sentido no qual o motor irá girar, horário/anti-horário.
- 1 LED vermelho para indicar o estado dos motores ON/OFF.
- 9 resistores 330 Ohms, para cada um dos LEDs.
- 1 Arduino MEGA com módulo WiFI da RobotDyn.
- 1 Fonte de Computador de 200W adaptada.

4.2 Protótipo Inicial

A estrutura inicial utilizada foi idêntica a usada por Flatland (2016) para bater o record mundial no dia 06 de fevereiro de 2016 resolvendo o Cubo Mágico em 0.9 segundos. As peças que compõem esta estrutura são:

- 1 Cubo Mágico.
- 1 Base para leitura das faces do Cubo Mágico.
- 6 conectores, eixo do motor ao eixo do Cubo Mágico.
- 6 motores de passo, modelo Nema 17.
- 1 suporte para os motores de passo laterais e o motor inferior.
- 1 suporte para o motor de passo superior.

Figura 35 – Estrutura de Flatland (2016).

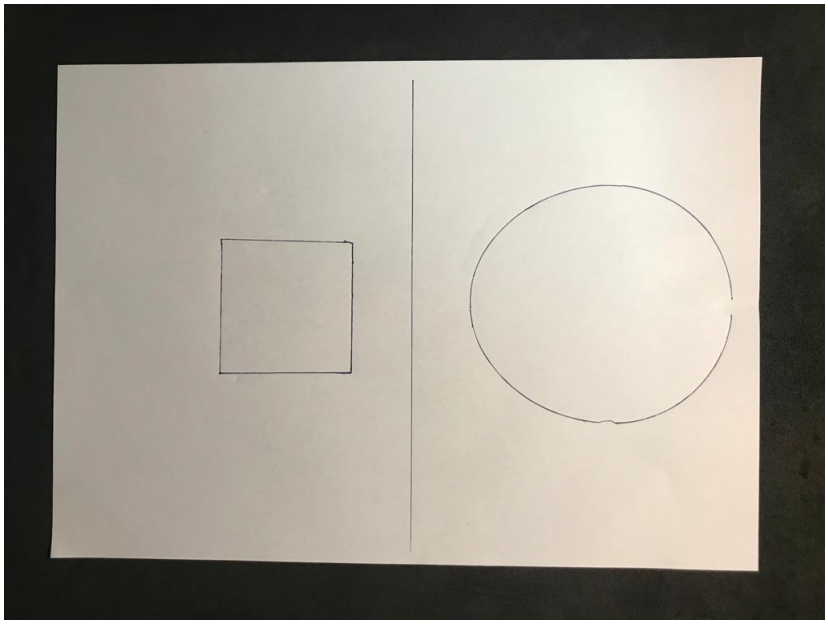
Fonte – Flatland (2016).

4.2.1 Modelagem da base de leitura do Cubo Mágico

Para fazer a leitura das faces do Cubo, era necessário que o mesmo permanecesse no mesmo local, imóvel, para que as coordenadas de cada um dos *cublets* ficassem na mesma posição. Para garantí-lo, foi desenvolvido uma base para encaixe do Cubo Mágico e da câmera que faria a leitura.

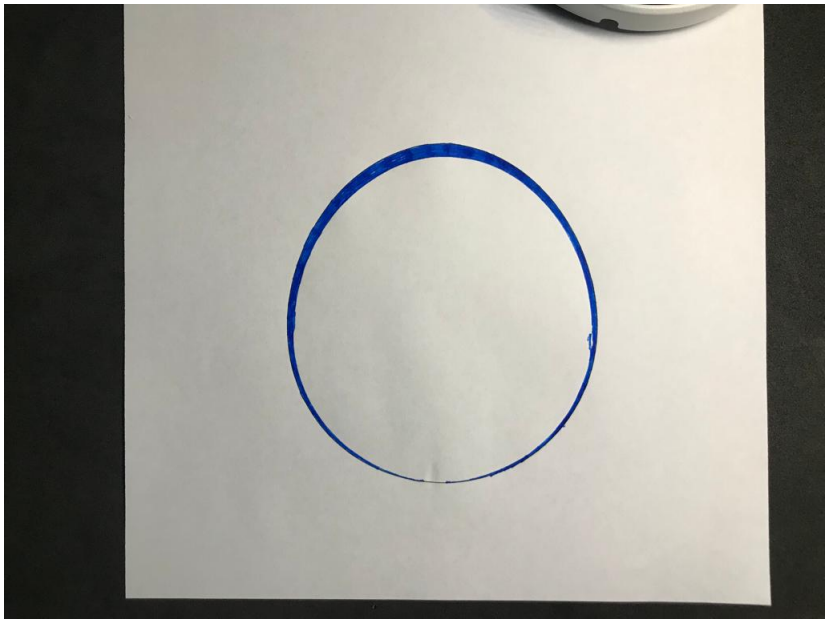
A câmera utilizada foi a VIP 1120 D G2 da Intelbras, que possui uma base oval não simétrica. Para garantir um encaixe perfeito foi necessário vários passos para conseguir um modelo 3D nas proporções exatas, conforme apresentado nas figuras 36 à 41.

Figura 36 – Contorno em folha A4 do Cubo Mágico e da câmera.



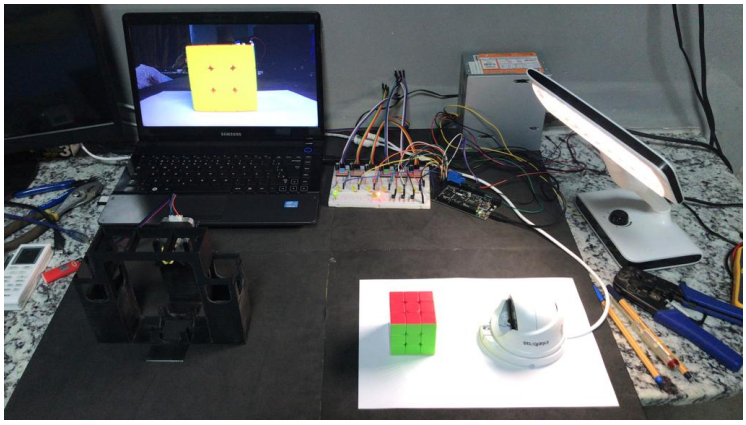
Fonte – Do autor (2019).

Figura 37 – Contorno da câmera aprimorado para facilitar a digitalização.



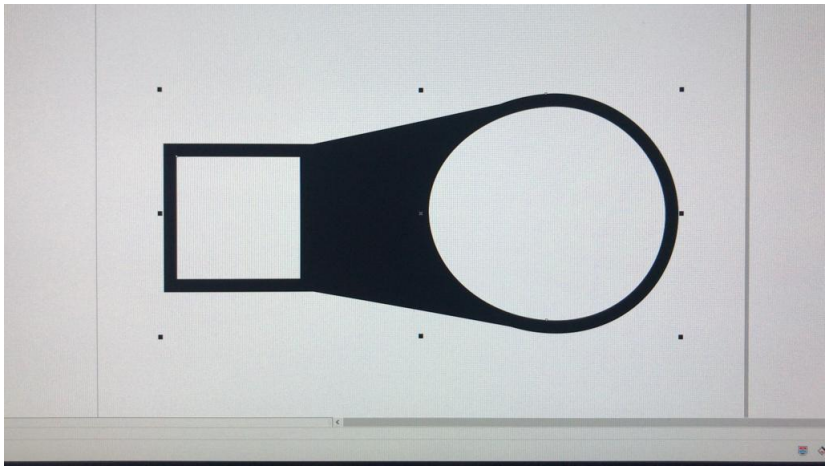
Fonte – Do autor (2019).

Figura 38 – Conferindo a distância necessária para leitura.



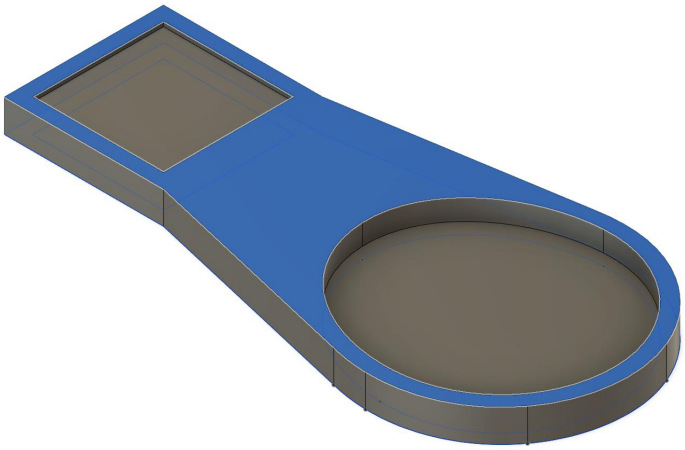
Fonte – Do autor (2019).

Figura 39 – Modelo 2D da base criado utilizando o *software* Corel Draw.



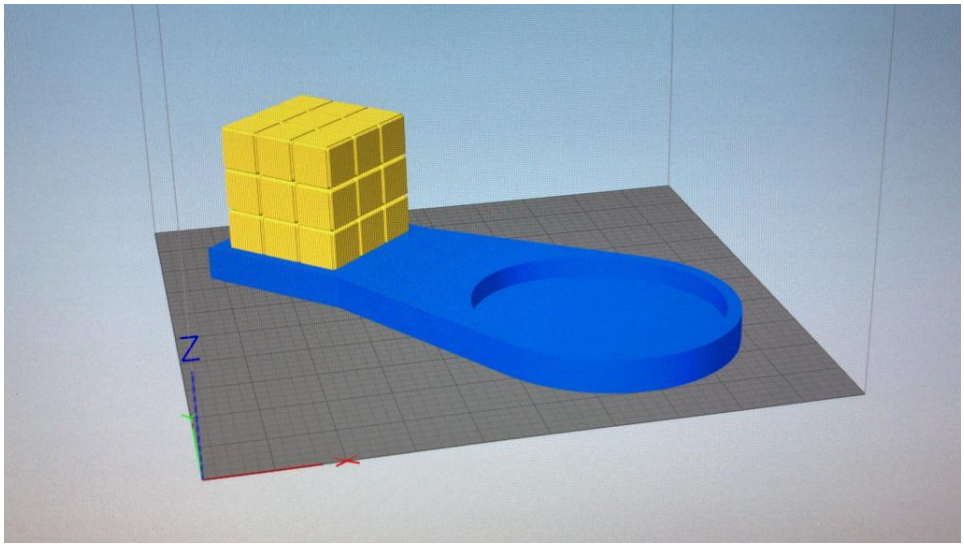
Fonte – Do autor (2019).

Figura 40 – Extrusão do arquivo DXF utilizando o *software* Fusion360 para criação do modelo 3D.



Fonte – Do autor (2019).

Figura 41 – Conferindo medidas e preparando o arquivo STL para impressão.



Fonte – Do autor (2019).

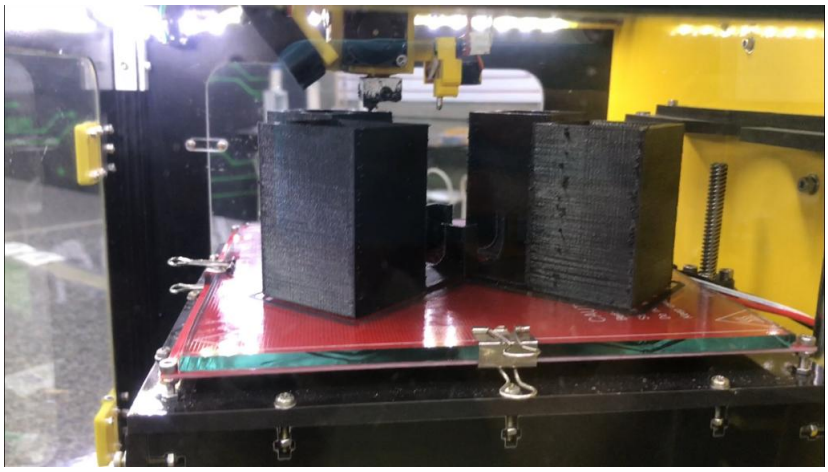
4.2.2 Impressão do protótipo

A impressão de todas as peças utilizadas até este ponto, foram realizadas na impressora 3D da faculdade Única (Factor 3D) pelos autores, conforme apresentado nas figuras 42 à 47.

Figura 42 – Impressora da Faculdade Única de Ipatinga.

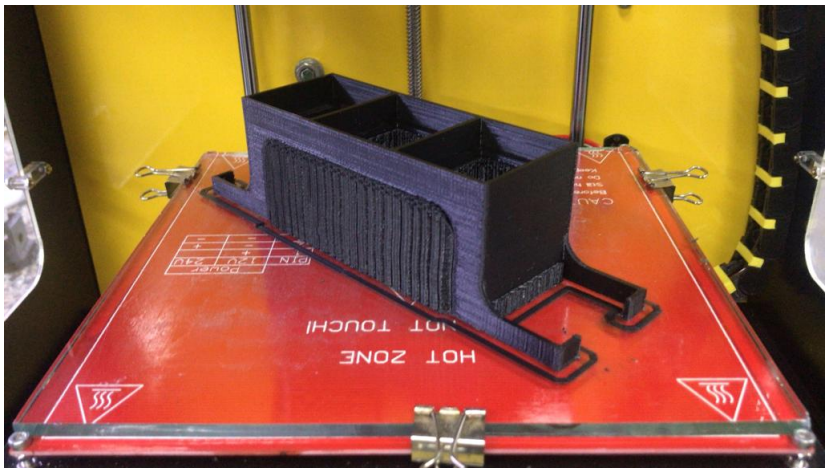
Fonte – Do autor (2019).

Figura 43 – Impressão da base do protótipo.



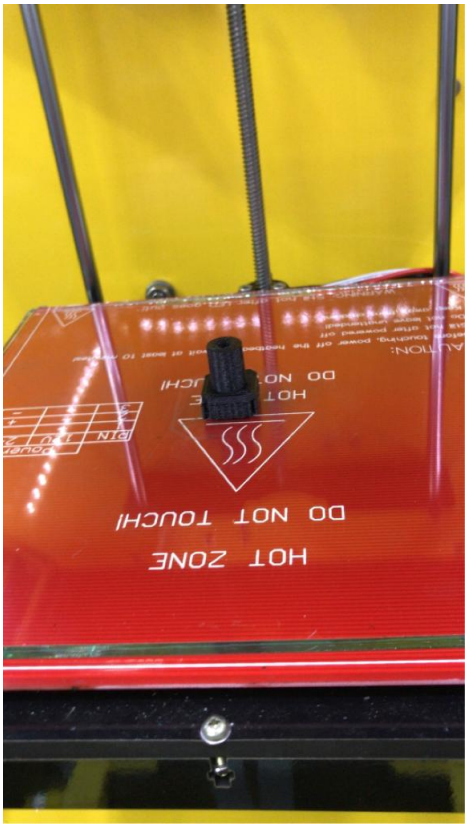
Fonte – Do autor (2019).

Figura 44 – Impressão do suporte do motor superior do protótipo.



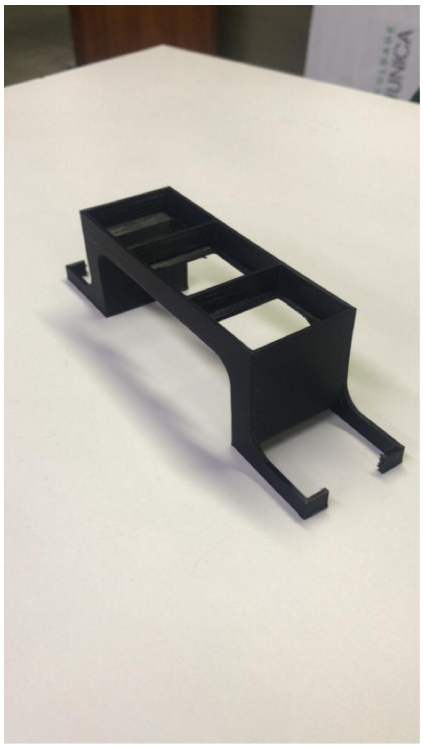
Fonte – Do autor (2019).

Figura 45 – Impressão de um conector de teste.



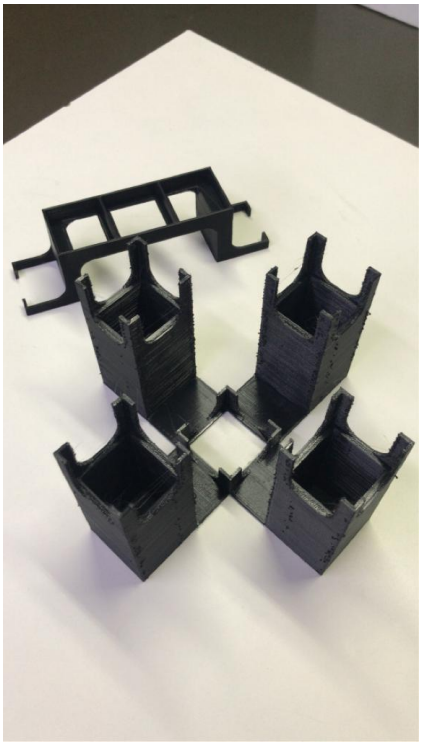
Fonte – Do autor (2019).

Figura 46 – Base do protótipo 100% impressa.



Fonte – Do autor (2019).

Figura 47 – Suporte do motor superior do protótipo 100% impresso.



Fonte – Do autor (2019).

4.3 Feira de Robótica e IA

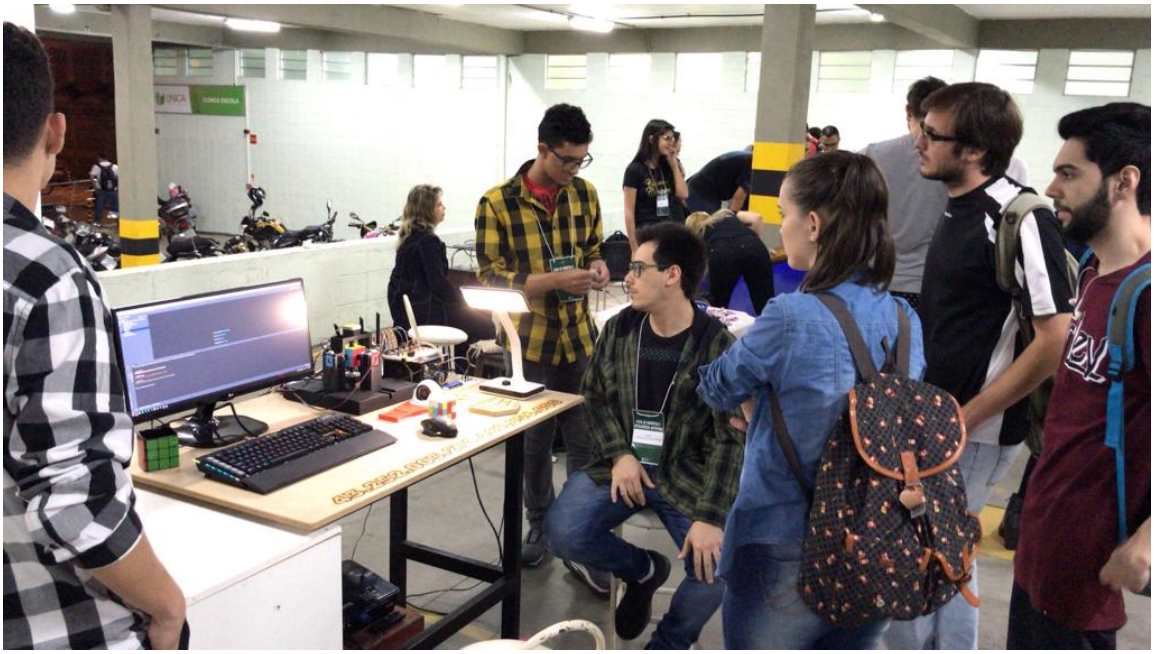
Durante o desenvolvimento do trabalho, houve a oportunidade de apresentar o primeiro protótipo funcional na feira de robótica da Faculdade Única. E na mesma pode-se receber um *feedback* das pessoas que visitaram o *stand*.

As figuras 49 à 53 demonstram o projeto sendo apresentado na feira de Robótica e IA da Faculdade Única:

Figura 48 – *Stand* do Cubo Mágico na feira.

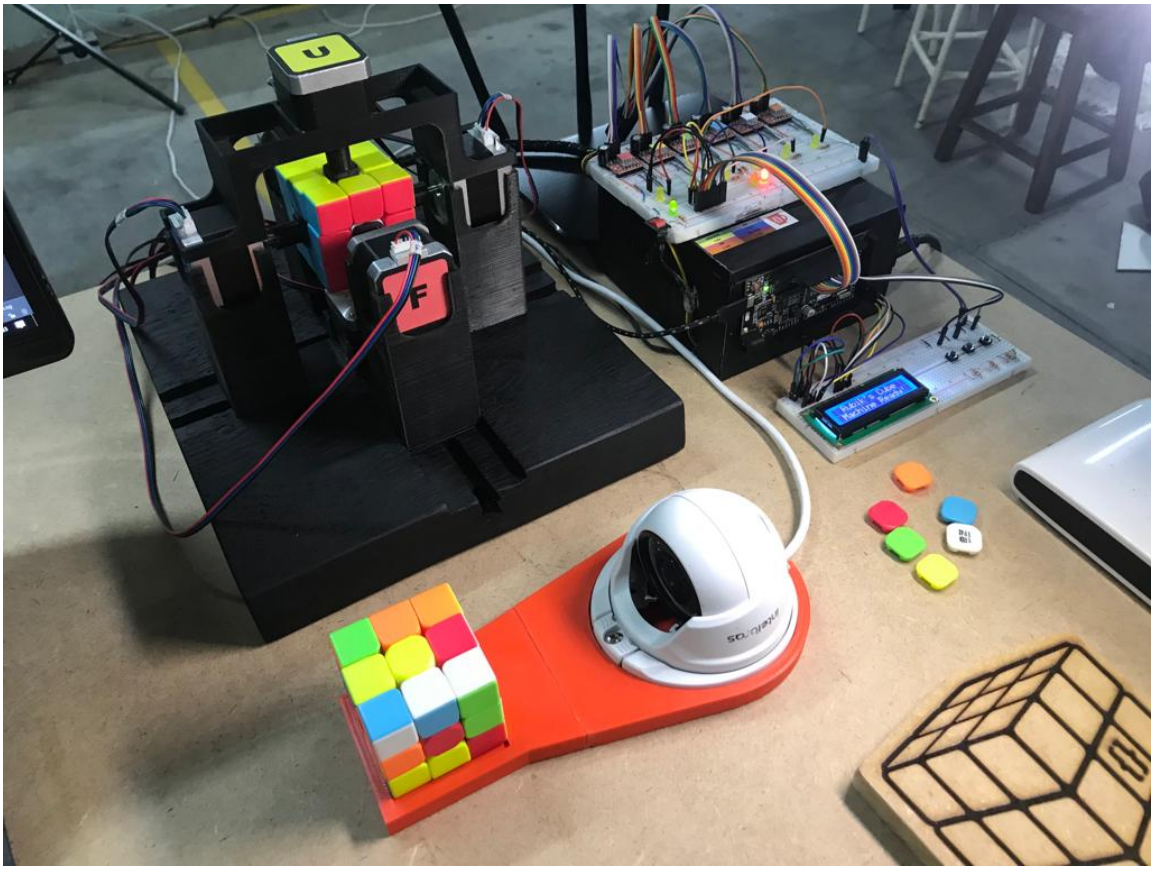
Fonte – Do autor (2019).

Figura 49 – Alguns visitantes do *stand* na feira.



Fonte – Do autor (2019).

Figura 50 – Primeiro protótipo do Robô, utilizado na feira.



Fonte – Do autor (2019).

Figura 51 – Base de leitura das faces do Cubo Mágico do primeiro protótipo.

Fonte – Do autor (2019).

Figura 52 – Processo de leitura das faces do Cubo Mágico do primeiro protótipo.

Fonte – Do autor (2019).

Figura 53 – Kenedy Tostes, Isaque Costa, Professor Júlio Cezar, Carlos Eduardo e Robert Jhon.

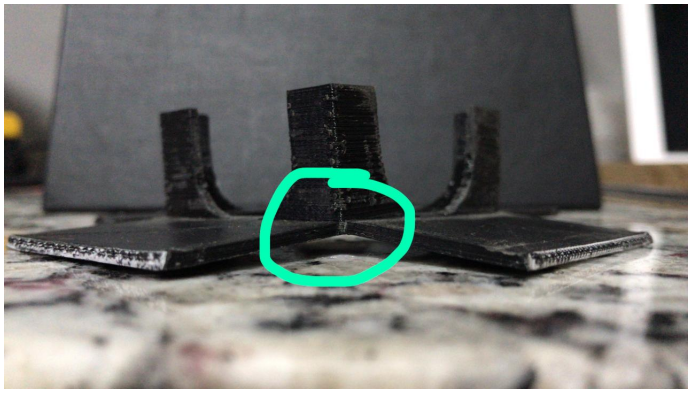
Fonte – Do autor (2019).

O trabalho foi contemplado com nota máxima e diversos elogios tanto dos curiosos, alunos visitantes, amigos e colegaa, bem como o corpo docente da Faculdade Única, que enfatizaram o fato do trabalho, abordar praticamente todas as diciplinas do curso de Ciência da Computação.

4.4 O Globo Mágico (Magic Globe)

A estrutura utilizada no primeiro protótipo, apresentou diversos problemas. Entre eles estão problemas decorrentes do próprio modelo em si, como por exemplo a necessidade de se utilizar várias câmeras para captar todos os *cublets* do Cubo Mágico, e outra grande parte por conta do equipamento de baixa qualidade utilizado para realização da impressão, como: diferenças de tamanhos e *WARP* apresentado na figura 54 (empeno da base da peça devida adesão pobre da mesma a cama de impressão).

Figura 54 – Exemplo de *WARP*.



Fonte – Do autor (2019).

Sendo assim, foi necessário uma mudança na estrutura através de uma nova abordagem, a partir dos requisito abaixo:

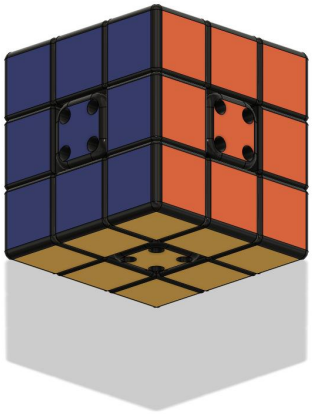
- A estrutura deverá sustentar os 6 motores de passo, de forma que cada um deles fique com o seu respectivo eixo, centralizado com a face do cubo a qual irá rotacionar.
- A mesma deve possibilitar um bom ângulo de visão para as câmeras que serão instaladas posteriormente, permitindo a visualização de todos os 54 *cublets* espalhados pelas faces do Cubo Mágico, portanto, a distância do motor ao cubo deve ser o suficiente para satisfazer tal requisito.

Sendo assim, o método adotado para a construção desta estrutura foi novamente a impressão 3D, que possibilitou um *design* arrojado atendendo todos os pré-requisitos. As peças que compõem esta estrutura são:

- 1 Cubo Mágico.
- 6 conectores, eixo do motor ao eixo do Cubo Mágico.
- 6 motores de passo, modelo Nema 17.
- 3 suportes com pés para os motores de passo inferiores.
- 3 suportes para os motores de passo superiores.
- 12 arcos de conexão.

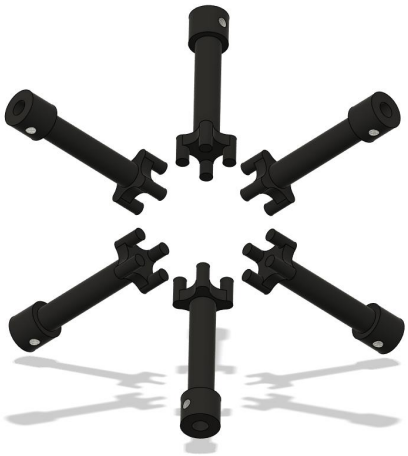
As figuras 55 à 61 demonstram as peças listadas acima.

Figura 55 – Cubo Mágico em sua respectiva posição.



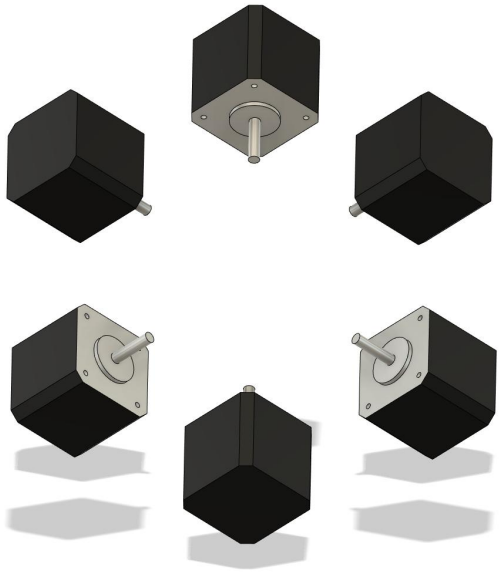
Fonte – Do autor (2019).

Figura 56 – Conectores em suas respectivas posições.



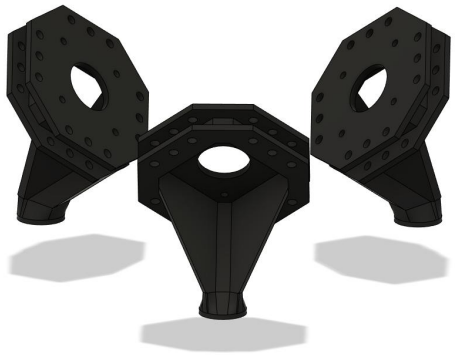
Fonte – Do autor (2019).

Figura 57 – Motores de passo (Nema 17) em suas respectivas posições.



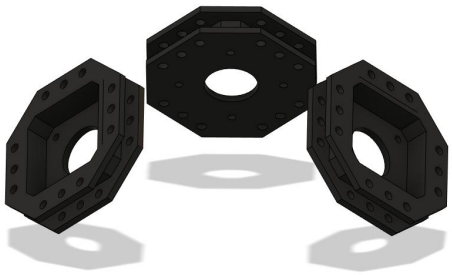
Fonte – Do autor (2019).

Figura 58 – Suportes dos motores inferiores em suas respectivas posições.



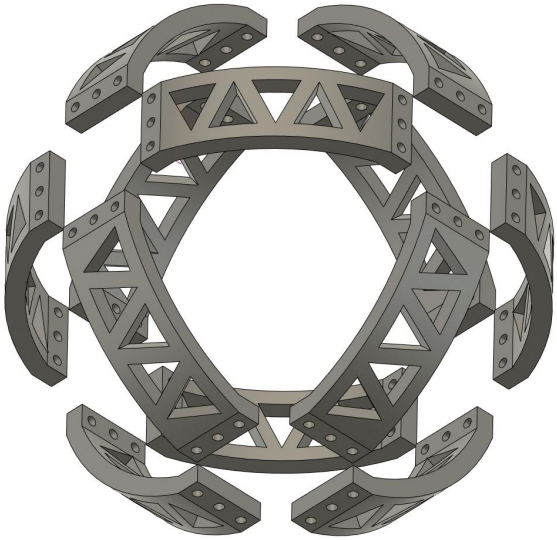
Fonte – Do autor (2019).

Figura 59 – Suportes dos motores superiores em suas respectivas posições.



Fonte – Do autor (2019).

Figura 60 – Arcos de conexão em suas respectivas posições.



Fonte – Do autor (2019).

Figura 61 – Resultado esperado com todas as peças.

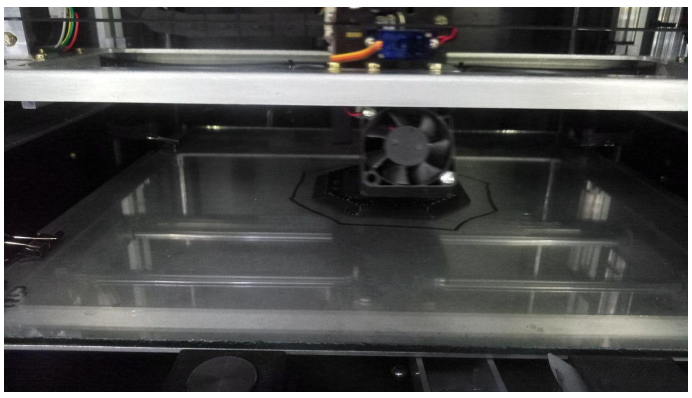
Fonte – Do autor (2019).

Todas as peças citadas acima, com exceção do Cubo Mágico e os motores de passo foram disponibilizadas por Horstkötter (2019) através do Thingiverse (2019).

4.4.1 Impressão do Globo

Todas as peças foram impressas utilizando o processo de Fabricação de filamentos fundidos (FFF) pela empresa Cubo3D, o material utilizado foi o ABS. Todo o processo de impressão levou 50 horas, sendo impresso peça por peça que, segundo a Cubo3D (2019), é importante para se reduzir os riscos de problemas no processo como um todo. A figura 62 mostra o processo de impressão de um dos suportes.

Figura 62 – Processo de impressão de um dos suportes.



Fonte – Do autor (2019).

O próximo passo após a impressão é o de limpeza das peças. É necessário remover

todos os suportes gerados pela impressora, rebarbar os cantos e nos casos em que temos pequenos furos, é preciso abrí-los novamente com a assistência de uma micro-retífica. Este processo levou 2 horas e é ilustrado na figura 63.

Figura 63 – Processo de limpeza das peças.

Fonte – Do autor (2019).

4.4.2 Montagem do Globo

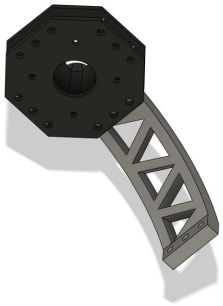
Para montagem da estrutura foi necessário 72 parafusos M3 de 30mm e 24 parafusos M3 de 10mm. São estes parafusos que irão manter a conexão dos arcos com as demais peças e prender os motores de passo em seus respectivos suportes.

O processo de montagem foi feito na seguinte ordem:

Ao fim de cada passo, foram adicionados e apertados levemente os parafusos M3.

- Passo 1: Conexão de um dos suportes inferiores a um arco de conexão como mostra a figura 64.

Figura 64 – Passo 01 do processo de montagem.



Fonte – Do autor (2019).

- Passo 2: Repetir o primeiro passo com outras duas peças e depois conectar o conjunto ao anterior como mostra a figura 65.

Figura 65 – Passo 02 do processo de montagem.

Fonte – Do autor (2019).

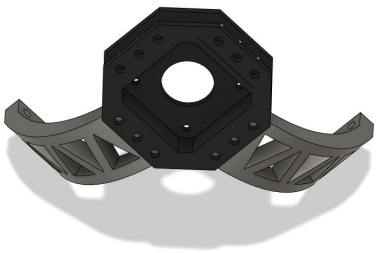
- Passo 3: Repetir o segundo passo com outras duas peças e depois conectar o conjunto ao anterior como mostra a figura 66.

Figura 66 – Passo 03 do processo de montagem.

Fonte – Do autor (2019).

- Passo 4: Conexão de dois arcos de conexão aos encaixes inferiores de um dos suportes superiores, ver figura 67.

Figura 67 – Passo 04 do processo de montagem.



Fonte – Do autor (2019).

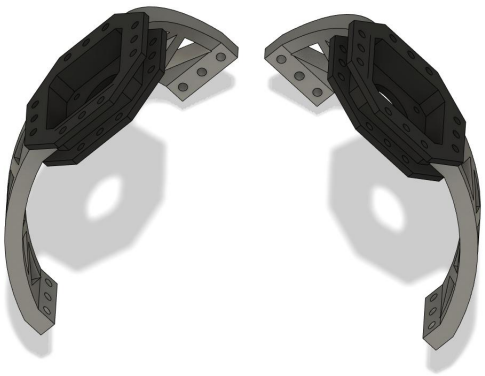
- Passo 5: Conexão do conjunto criado no passo anterior ao restante da estrutura, ver figura 68.

Figura 68 – Passo 05 do processo de montagem.

Fonte – Do autor (2019).

- Passo 6 e 7: Repetir o quarto passo outras duas vezes e depois conectar o conjunto ao anterior como mostram as figuras 69 e 70.

Figura 69 – Passo 06 do processo de montagem.



Fonte – Do autor (2019).

Figura 70 – Passo 07 do processo de montagem.

Fonte – Do autor (2019).

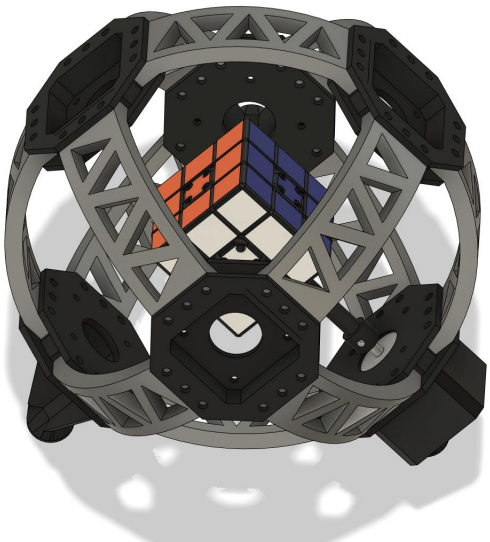
- Passo 8: Conexão dos últimos arcos de conexão como mostra a figura 71.

Figura 71 – Passo 08 do processo de montagem.

Fonte – Do autor (2019).

- Passo 9: Posicionamento do Cubo Mágico no centro com o auxílio de, um conector e um motor, como mostra a figura 72.

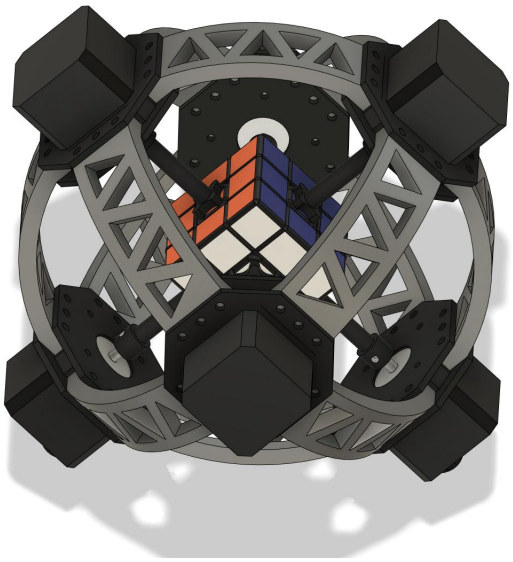
Figura 72 – Passo 09 do processo de montagem.



Fonte – Do autor (2019).

- Passo 10: Conexão dos demais motores e conectores como mostra a figura 73.

Figura 73 – Passo 10 do processo de montagem.



Fonte – Do autor (2019).

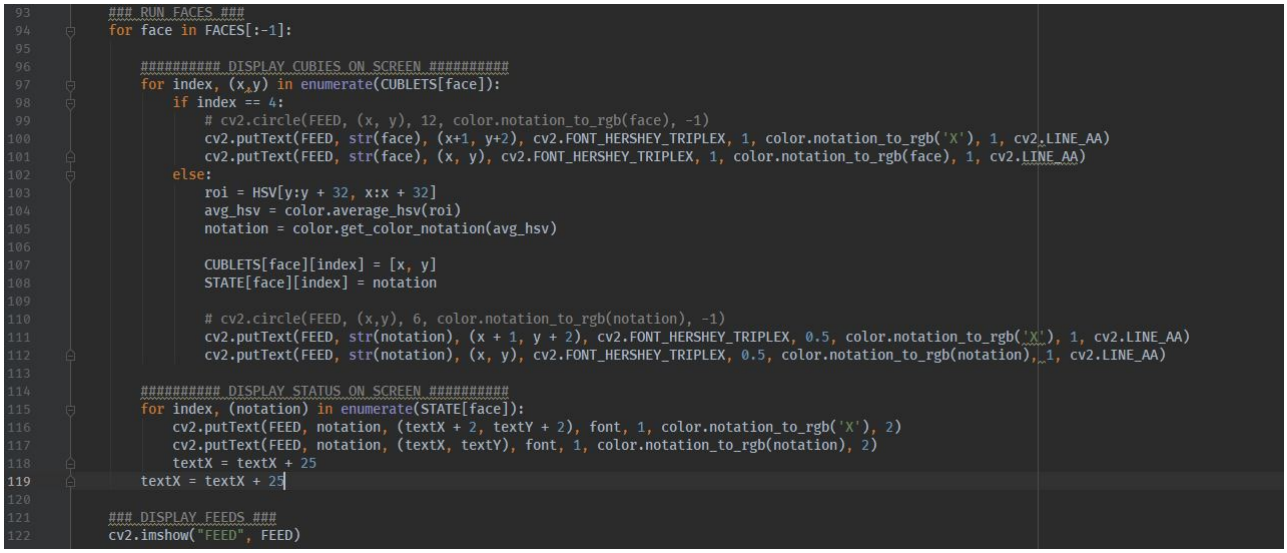
4.5 Os *softwares*

Para executar todas as tarefas necessárias para se resolver o Cubo Mágico com a estrutura acima descrita, foram necessários o desenvolvimento de 4 *softwares* diferentes, iremos agora explicar cada um deles de forma separada e depois apresentar como eles funcionam juntos.

4.5.1 *Software* 1: Detecção de Cores utilizando o OpenCV

Para determinar o estado atual do Cubo Mágico foi desenvolvido um *software* em python utilizando a biblioteca OpenCV (Figura 74). Este programa faz a captura dos *feeds* das webcams instaladas na estrutura para visualizar todos os 54 *cublets* do Cubo Mágico. Estes *feeds* são concatenados em um para simplificar o código. A partir daí, convertemos o esquema de cores dessa imagem para HSV e rodamos por todos as 54 posições pre-armazenadas para determinar a cor presente em cada uma delas:

Figura 74 – Rodando as posições.



```
    ### RUN FACES ###
    for face in FACES[:-1]:

        ########## DISPLAY CUBIES ON SCREEN ##########
        for index, (x,y) in enumerate(CUBLETS[face]):
            if index == 4:
                # cv2.circle(FEED, (x, y), 12, color.notation_to_rgb(face), -1)
                cv2.putText(FEED, str(face), (x+1, y+2), cv2.FONT_HERSHEY_TRIPLEX, 1, color.notation_to_rgb('X'), 1, cv2.LINE_AA)
                cv2.putText(FEED, str(face), (x, y), cv2.FONT_HERSHEY_TRIPLEX, 1, color.notation_to_rgb(face), 1, cv2.LINE_AA)
            else:
                roi = HSV[y:y + 32, x:x + 32]
                avg_hsv = color.average_hsv(roi)
                notation = color.get_color_notation(avg_hsv)

                CUBLETS[face][index] = [x, y]
                STATE[face][index] = notation

                # cv2.circle(FEED, (x,y), 6, color.notation_to_rgb(notation), -1)
                cv2.putText(FEED, str(notation), (x + 1, y + 2), cv2.FONT_HERSHEY_TRIPLEX, 0.5, color.notation_to_rgb('X'), 1, cv2.LINE_AA)
                cv2.putText(FEED, str(notation), (x, y), cv2.FONT_HERSHEY_TRIPLEX, 0.5, color.notation_to_rgb(notation), 1, cv2.LINE_AA)

        ########## DISPLAY STATUS ON SCREEN ##########
        for index, (notation) in enumerate(STATE[face]):
            cv2.putText(FEED, notation, (textX + 2, textY + 2), font, 1, color.notation_to_rgb('X'), 2)
            cv2.putText(FEED, notation, (textX, textY), font, 1, color.notation_to_rgb(notation), 2)
            textX = textX + 25
        textX = textX + 25

    ### DISPLAY FEEDS ###
    cv2.imshow("FEED", FEED)
```

Fonte – Do autor (2019).

Após a detecção das cores, é realizado a normalização para a notação "URFDLB"e concatenado tudo em uma única *string*. Pois é neste *software* que é feito o cálculo da solução do Cubo Mágico, e para isso é utilizado o algoritimo de KOCIEMBA (2011), que recebe uma *string* informando o estado atual do Cubo Mágico, e retorna uma nova *string*, contendo o conjunto de movimentos em uma determinada ordem que irão solucionar o cubo.

4.5.2 *Software* 2: Arduino MEGA

Este foi o primeiro *software* desenvolvido (Figura 75), ele é o responsável pelo controle dos motores e foi desenvolvido utilizando a linguagem C.

No código há essencialmente 5 funcões, uma para ler as requisições vindas do *web-server*, outra para ler as requisições vindas da porta serial, outra para controlar o estado dos 6 motores (*idle*/*running*), outra para *tokenizar* a *string* recebida nessas requisições e enviar para última função que normaliza e executa estes movimentos nos motores:

Figura 75 – Função que lê requisições vindas da porta serial.

```
199 void listenForAlgorithm() {
200   String algorithm = "";
201   char character;
202
203   // Enquanto receber algo pela serial
204   while (Serial.available() > 0) {
205     // Lê byte da serial
206     character = Serial.read();
207     // Ignora character de quebra de linha
208     if (character != '\n') {
209       // Concatena valores
210       algorithm.concat(character);
211     }
212     // Aguarda buffer serial ler próximo character
213     delay(10);
214   }
215   algorithm.toUpperCase();
216
217   if (algorithm != "")
218     executeAlgorithm(algorithm);
219 }
```

Fonte – Do autor (2019).

4.5.3 *Software* 3: ESP8266

Este foi o ultimo *software* desenvolvido (Figura 76), ele é o responsável por enviar requisições para a placa Arduino MEGA vindas de uma aplicação *web* com a intenção de controlar o Cubo Mágico que fica dentro da estrutura.

No código tem-se essencialmente uma implantação de um *webserver* que irá hospedar e rodar uma aplicação web dentro da placa ESP8266.

Figura 76 – Trecho do código do *webserver*.

```
144 ////// FUNCTIONS ////////////////////////////////////////////////////////////////
145
146   void turnCube();
147
148 void setup()
149 {
150   Serial.begin(115200);
151   WiFi.begin(ssid,password);
152   while(WiFi.status()!=WL_CONNECTED)
153   {
154     delay(500);
155   }
156   Serial.println(WiFi.localIP());
157   Serial.println(WiFi.macAddress());
158
159   server.on("/",[](){server.send_P(200,"text/html", webpage);});
160   server.on("/turn", turnCube);
161   server.begin();
162 }
163
164 void loop()
165 {
166   server.handleClient();
167 }
168
169 void turnCube()
170 {
171   REQUEST = server.arg("command");
172
173   for(int i = 0; i <= REQUEST.length(); i++)
174     Serial.write(REQUEST[i]);
175
176   REQUEST = "";
177   delay(1);
178
179   server.send(200,"text/plain", "OK");
180 }
```

Fonte – Do autor (2019).

4.5.4 *Software* 4: Aplicação Web

Este *software* é uma adaptação do Chrome (2019), que é uma aplicação web desenvolvida essencialmente JavaScript, CSS e HTML. As implementações feitas neste código permitem que uma *string* contendo o estado atual do Cubo Mágico, seja passada para criar um Cubo 3D interativo que esteja no mesmo estado embaralhado que o Cubo físico se encontra.

Algumas funções adicionadas para possibilitar esta funcionalidade:

Figura 77 – Trecho 1 do código da aplicação *web*.

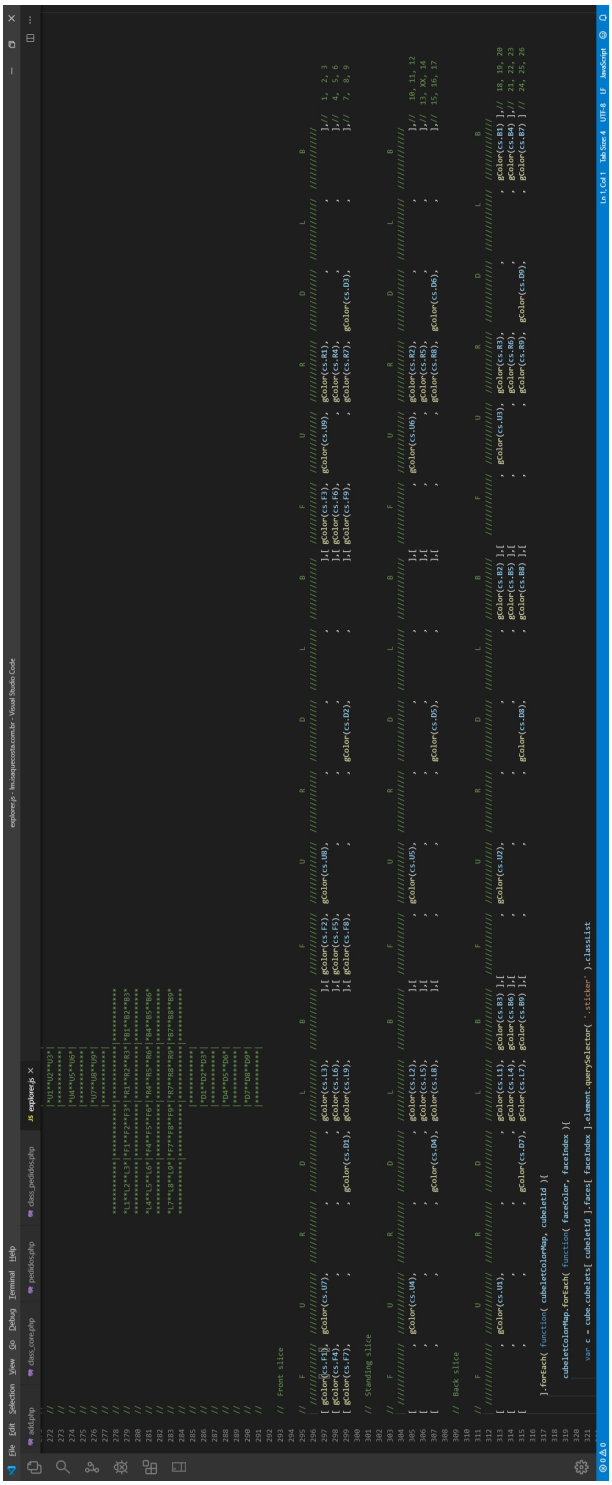
```
gColor = function (s)
{
        if (s.toUpperCase() == 'D' || s.toUpperCase() == '1')
        return ERNO.WHITE
    else if (s.toUpperCase() == 'B' || s.toUpperCase() == '2')
        return ERNO.ORANGE
    else if (s.toUpperCase() == 'L' || s.toUpperCase() == '5')
        return ERNO.BLUE
    else if (s.toUpperCase() == 'F' || s.toUpperCase() == '4')
        return ERNO.RED
    else if (s.toUpperCase() == 'R' || s.toUpperCase() == '3')
        return ERNO.GREEN
    else if (s.toUpperCase() == 'U' || s.toUpperCase() == '6')
        return ERNO.YELLOW
    else
        return ERNO.COLORLESS
}

translateTwist = function (algorithm)
{
    let movesCount = 0
    let commands = algorithm.split(' ')
    let newCommands = ""

    commands.filter(command => {

            if (command.includes("'"))
                command = command.replace("'", '').toLowerCase()

            if (command.includes('2'))
                command = command.replace('2', '').repeat(2)

        movesCount++
        newCommands = newCommands + command
    });

    cube.twist(newCommands)
    console.log('movesCount => ', movesCount)
    console.log('algorithm => ', algorithm, ' ===> ', newCommands)
}
```

Fonte – Do autor (2019).

No código há diversas adaptações, mas sem dúvidas a mais custosa é ilustrada no trecho da figura 78. Foi necessário identificar as posições do array que determina as posições do

Cubo Mágico, para chamar a função mostrada no primeiro trecho da figura 77, que recebe como parâmetro uma posição da *string* contendo o estado desejado para o cubo e retorna o objeto que contém as propriedas necessárias para desenhar o cubo na tela.

Figura 78 – Trecho 2 do código da aplicação *web*, aqui basicamente é feito a designação correta de cada posição da *string* recebida para o cubo digital.

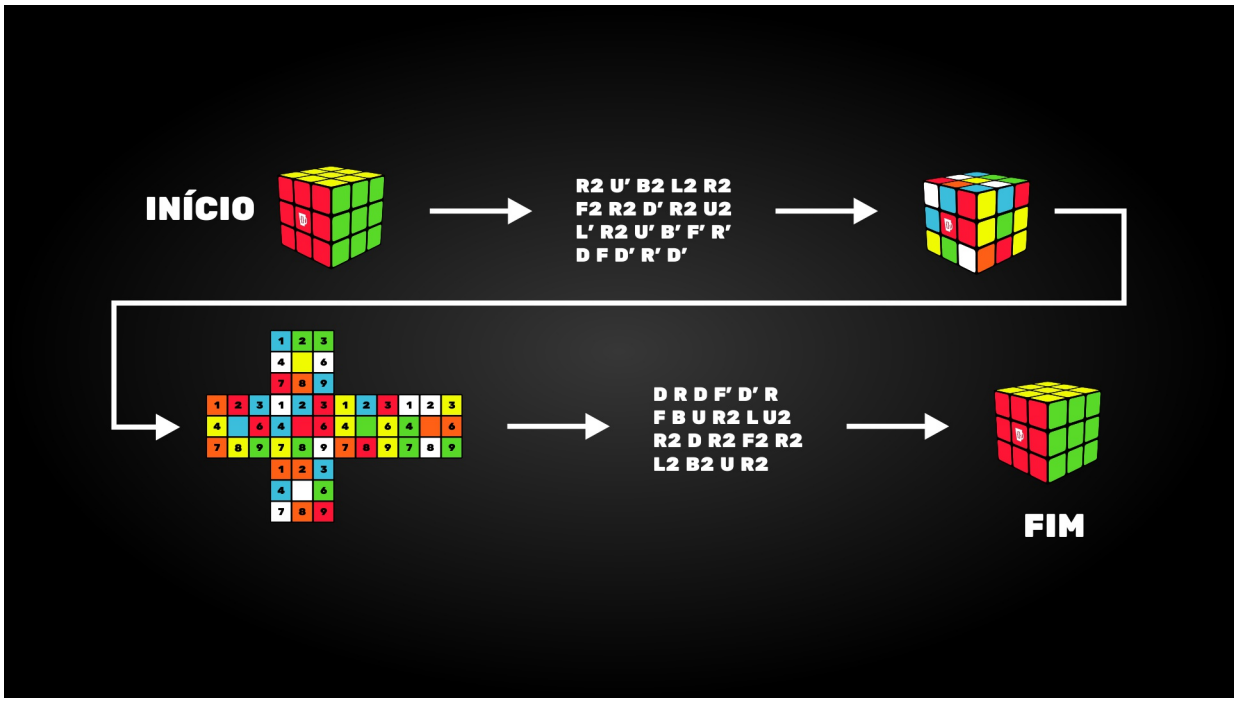


Fonte – Do autor (2019).

4.6 Solucionando o Cubo Mágico

O funcionamento de todo projeto acontece em diversas partes como exemplificado na Figura 79.

Figura 79 – Fluxograma.

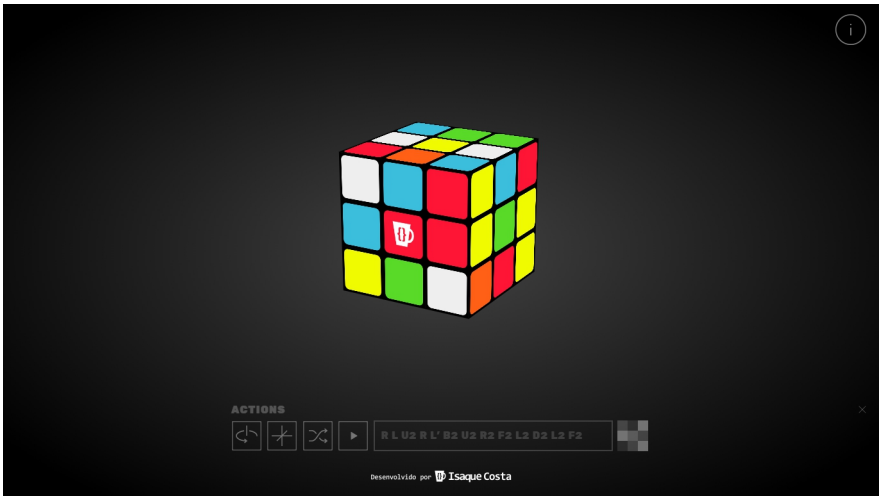


Fonte – Do autor (2019).

Inicialmente, a estrutura é composta por um Cubo Mágico solucionado, que é embaralhado utilizando a aplicação *web* manualmente pelo usuário ou de forma aleatória gerada pelo *software*. Agora com o cubo mágico embaralhado entra em ação a leitura do estado do cubo que resulta em uma *string* contendo este estado embaralhado. Este estado é analizado e uma solução para o mesmo é calculada. Uma vez que a solução é encontrada, ela será enviada para a placa Arduino que executa os movimentos deixando o Cubo Mágico resolvido novamente.

Na página seguinte, as figuras 80 e 81 demonstram alguns exemplos.

Figura 80 – Cubo no estado 1.



Fonte – Do autor (2019).

Entrada do usuário: R2 U’ B2 L2 R2 F2 R2 D’ R2 U2 L’ R2 U’ B’ F’ R’ D F D’ R’ D’

Estado normalizado: LRRDUDFBLULFURUBFUDLFLFFURDBBLLDRDBFBFLULFBURDDURBBRDR

Solução calculada: D R D F’ D’ R F B U R2 L U2 R2 D R2 F2 R2 L2 B2 U R2

Figura 81 – Cubo no estado 2.

Fonte – Do autor (2019).

Entrada do usuário: U’ R2 D’ U2 L2 B2 U F2 D F2 R D2 R2 D’ B’ F2 D R D2

Estado normalizado: DRLUUBFBRBLURRLRUBLRDDFDLFUFUFFDBRDUBRUFLLFDDBFLUBLRBD

Solução calculada: D2 R’ D’ F2 B D R2 D2 R’ F2 D’ F2 U’ B2 L2 U2 D R2 U

4.7 Problemas e soluções

Durante todos os 12 meses de desenvolvimento deste projeto, diversos desafios foram vencidos para alcançar uma máquina funcional. Dentre eles estão:

- Energia e Aquecimento

PROBLEMA: Nos primórdios do desenvolvimento do primeiro protótipo, encontrou-se problemas para alimentar todos os 6 motores simultâneamente, a primeira fonte não suportava a entrega dos 12V necessários para alimentar os 6 motores simultâneamente por conta de não suportar uma amperagem necessária.

SOLUÇÃO: Substituição da fonte simples por uma fonte ATX de 200W que entrega até 12 amperes. Foi necessário a adaptação das saídas da fonte para conectá-la a protoboard.

- Calibração da detecção de cores

PROBLEMA: A estrutura final, apesar de ter diversos pontos positivos, conta com um pequeno problema de projeção de sombras em alguns pontos do Cubo Mágico.

SOLUÇÃO: Foi acrescentado uma fita de LED ao redor de cada câmera para criar uma iluminação homogênea.

4.8 Cronograma das atividades

Para o desenvolvimento deste projeto foi definido um cronograma de atividas apresentado na figura a seguir:

Figura 82 – Cronograma das atividades.

Mês/Ano	Resumo das Atividades
01/2019	Pesquisa e levantamento dos requisitos do projeto.
02/2019	Início das impressões da estrutura do primeiro protótipo.
03/2019	Início do desenvolvimento do software, compra dos componentes e montagem do circuito.
04/2019	Fim das impressões da estrutura e montagem do trabalho.
05/2019	Fim do desenvolvimento do software, testes e ajustes do protótipo.
06/2019	Apresentação do primeiro protótipo na Feira de Robótica.
07/2019	Planejamento e pesquisa da nova estrutura.
08/2019	Desenvolvimento da aplicação web.
09/2019	Desenvolvimento da aplicação web.
10/2019	Impressão da nova estrutura e adaptações no software.
11/2019	Documentação do projeto.
12/2019	Apresentação do projeto para a banca.

Fonte – Do autor (2019).

4.9 Gastos com o Projeto

Estes são os gastos referentes apenas ao que se foi utilizado na versão final do projeto, ignorando os custos do primeiro protótipo e materias descartados.

Figura 83 – Tabela de gastos.

Item	Valor
Impressão da Estrutura	R$ 1.000,00
6 motores Nema 17	R$ 420,00
6 drivers A4988	R$ 131,40
3 protoboards	R$ 174,70
Jumpers	R$ 97,40
2 Webcams	R$ 799,76
Arduino MEGA + WiFi	R$ 125,00
MDF + Cortes	R$ 225,00
Parafusos e porcas	R$ 95,00
Fonte ATX e cabos	R$ 382,00
Adesivos dos motores	R$ 21,90
Despesas com transporte (Fretes, SEDEX, gasolina)	R$ 878,63
Soldas, fita isolante, espaguete termo retrátil, LEDs e outros.	R$ 187,90
Valor Total	**R$ 4.538,69**

Fonte – Do autor (2019).

4.10 Trabalhos Futuros

Pretende-se o desenvolvimento de um mecânismo de retirada rápida dos motores, proporcionando a oportunidade de remover o cubo da estrutura para o embaralhamento fisico, ao invés de utilizar-se apenas o iPad para tal procedimento.

5 CONSIDERAÇÕES FINAIS

Desenvolver uma máquina para bater o recorde mundial atual, aumentaria de forma considerável o custo final do trabalho. Porém, com esta máquina chegou-se próximo, alcançando a marca de 1,5 segundos para solução completa do Cubo Mágico. Através deste trabalho, foi possível vislumbrar que a visão computacional e sistemas distribuídos possuem inúmeras aplicações que podem auxiliar diretamente na melhoria de diversos processos.

O presente trabalho cumpre com seu objetivo geral proposto: A resolução do Cubo Mágico através de uma máquina, de forma rápida, superando os melhores competidores (humanos). Contudo, é importante resaltar o maior impecílio para funcionamento correto da leitura do Cubo Mágico: a iluminação uniforme, que foi um desafio para ser superado, pois a estrutura apesar de seus diversos pontos positivos, tem um ponto negativo forte, que são os pontos de sombra, gerados pela mesma.

Além disso, todos os objetivos secundários foram atendidos, porém em trabalhos futuros, pretende-se o desenvolvimento de um mecanismo de retirada rápida dos motores, proporcionando a oportunidade de remover o cubo da estrutura para o embaralhamento fisico, ao invés de utilizar-se apenas o iPad para tal procedimento.

REFERÊNCIAS

ACADEMY, D. S. **Data Science Academy Website.** 2019. Disponível em: <https://i2.wp.com/datascienceacademy.com.br/blog/wp-content/uploads/2017/01/computer-vision.png?resize=978\%2C413>. Acesso em: 31 de Outubro de 2019.

AUTODESK. **O que é impressão 3D?** 2019. Disponível em: <https://www.autodesk.com.br/solutions/3d-printing>. Acesso em: 31 de Outubro de 2019.

AWSLI. **AWSLI Content Delivery Network.** 2019. Disponível em: <https://cdn.awsli.com.br/600x450/404/404053/produto/15881345/1488ab4c63.jpg>. Acesso em: 31 de Outubro de 2019.

BRASIL, P. **Python: O que é? Por que usar?** 2019. Disponível em: <http://pyscience-brasil.wikidot.com/python:python-oq-e-pq>. Acesso em: 31 de Outubro de 2019.

CHROME, G. **Chrome Cube Lab**. 2019. Disponível em: <https://chrome.com/cubelab>. Acesso em: 31 de Outubro de 2019.

CORPORATION, C. **CorelDRAW**. 2019. Disponível em: <https://www.coreldraw.com>. Acesso em: 31 de Outubro de 2019.

CUBO3D. **Impressão 3D**. 2019. Disponível em: <https://cubo3d.com.br>. Acesso em: 31 de Outubro de 2019.

EIS, D. **O básico: O que é HTML?** 2011. Disponível em: <https://tableless.com.br/o-que-html-basico/>. Acesso em: 31 de Outubro de 2019.

EMBARCADOS. **Arduino - Comunicação Serial**. 2014. Disponível em: <https://www.embarcados.com.br/arduino-comunicacao-serial/>. Acesso em: 31 de Outubro de 2019.

FLATLAND, J. **World's Fastest Rubik's Cube Solving Robot**. 2016. Disponível em: <https://www.youtube.com/watch?v=ixTddQQ2Hs4>. Acesso em: 01 de Fevereiro de 2019.

GIL, A. C. **Como elaborar projetos de pesquisa**. [S.l.]: Atlas S/A, 2002. Acesso em: 31 de Outubro de 2019.

HORSTKöTTER, A. **Rubiks Cube Robot**. 2019. Disponível em: <https://www.thingiverse.com/thing:3515432>. Acesso em: 27 de Junho de 2019.

KOCIEMBA, H. **Solve rubik's cube with the Kociemba algorithm**. 2011. Disponível em: <https://kociemba.org>. Acesso em: 31 de Outubro de 2019.

KONDRASOVAS, I. **O que é arquivo DXF?** 2016. Disponível em: <https://www.otimizenesting.com.br/single-post/2016/09/29/Arquivo-DXF>. Acesso em: 31 de Outubro de 2019.

KOYANAGI, F. **Arduino Mega com WiFi Embutido ESP8266**. 2017. Disponível em: <https://www.fernandok.com/2017/11/arduino-mega-com-wifi-embutido-esp8266.html>. Acesso em: 27 de Junho de 2019.

KOYANAGI, F. **Motor de passo com Arduino e o Driver A4988**. 2017. Disponível em: <https://www.fernandok.com/2017/12/motor-de-passo-com-arduino-e-o-driver.html>. Acesso em: 27 de Junho de 2019.

MUNDO, M. do. **Manual Maker**. 2019. Disponível em: <https://www.youtube.com/watch?v=RN5PxezvIzE&list=PLYjrJH3e_wDPwKigz0AcIgzk9BF4lqDuy>. Acesso em: 31 de Outubro de 2019.

NOTEPLACE. **Entendendo a porta serial e o USB**. 2017. Disponível em: <http://www.noteplace.com.br/busca/?termo=entendendo>. Acesso em: 31 de Outubro de 2019.

OPENCV, E. **Sobre o OpenCV**. 2019. Disponível em: <https://www.opencv.org/about/>. Acesso em: 31 de Outubro de 2019.

PISCALED. **DRIVER MOTOR DE PASSO BIPOLAR A4988 COM DISSIPADOR**. 2019. Disponível em: <https://www.piscaled.com.br/driver-motor-de-passo-bipolar-a4988-com-dissipador>. Acesso em: 31 de Outubro de 2019.

PISCALED. **MOTOR DE PASSO NEMA 17 MODELO 17HS4401 4,2KGF.CM 4,2KG**. 2019. Disponível em: <https://www.piscaled.com.br/motor-de-passo-nema-17-modelo-17hs4401-42kgfcm-42kg>. Acesso em: 31 de Outubro de 2019.

PYTHON. **About Python**. 2019. Disponível em: <https://www.python.org/about/>. Acesso em: 31 de Outubro de 2019.

RODRIGUES, A. **O que é modelagem digital ou modelagem 3D**. 2018. Disponível em: <https://mundodesenhodigital.com.br/o-que-e-modelagem-digital-ou-modelagem-3d-e/>. Acesso em: 18 de Outubro de 2019.

RUBIK, E. **Ernő Rubik, o inventor do Cubo Mágico.** 1974. Disponível em: <https://www.rubiks.com/media//gene-cms/e/r/erno-1970s-with-cube_1.jpg>. Acesso em: 31 de Outubro de 2019.

RUBIK, E. **Patente registrada por Ern o Rubik em 1975.** 1975. Disponível em: <https://www.rubiks.com/media//gene-cms/e/r/erno-patent.jpg>. Acesso em: 31 de Outubro de 2019.

RUBIK, E. **Primeiro modelo comercializado do Cubo Mágico em 1977.** 1977. Disponível em: <https://www.rubiks.com/media//gene-cms/h/u/hungarian-magic-cube-with-box_3.jpg>. Acesso em: 31 de Outubro de 2019.

RUBIK, E. **Feira internacional de brinquedos realizada na Alemanha em 1979.** 1979. Disponível em: <https://www.rubiks.com/media//gene-cms/g/e/german-toy-fair-1979_1.jpg>. Acesso em: 31 de Outubro de 2019.

RUBIK, E. **About us | Rubik's Official Website**. 2019. Disponível em: <https://www.rubiks.com/en-us/>. Acesso em: 31 de Outubro de 2019.

SAS. **Visão Computacional, O que é e qual sua importância?** 2019. Disponível em: <https://www.sas.com/pt_br/insights/analytics/computer-vision.html>. Acesso em: 31 de Outubro de 2019.

SIMPLIFY3D. **Simplify3D Software | All-In-One 3D Printing Software**. 2014. Disponível em: <https://www.simplify3d.com/>. Acesso em: 31 de Outubro de 2019.

SOUZA, R. de. **Fusion 360: novo software da Autodesk quer democratizar a modelagem 3D**. 2014. Disponível em: <https://www.tecmundo.com.br/impressora-3d/56381-fusion-360-novo-software-autodesk-quer-democratizar-modelagem-3d.htm>. Acesso em: 31 de Outubro de 2019.

THINGIVERSE. **Digital Designs for Physical Objects**. 2019. Disponível em: <https://www.thingiverse.com>. Acesso em: 31 de Outubro de 2019.