2024-07-10_10-00_GabrielPancaRibeiro-VF

2024-07-03 - 10:02

Batalha da Ciência: é o nome do tapete :-) .. poderia colocar as duas imagens dos slides sobre os minijogos no Anexo e citar explicando no texto.
Mencionar no texto ser melhor fazer o processamento de reconhecimento local sem uso da rede porque muitas vezes não se tem rede aceitável nos locais de uso.

Slide do diagrama de atividade não é a figura 4 do artigo.

Figura 2: cor vermelha e laranja das setas ... muito iguais.
Podes refazer a figura e mencionar fonte adaptada.

Colocar a imagem do robô em cima do "tapete" que eu enviei :-)

No slide do processamento de imagem colocar uma imagem do fluxo de etapas embaixo da imagem do resultado. Ver exemplo dos TCC do Aurélio.

Imagem do aplicativo móvel.
Só aqui mencionou da diferença entre OpenCV da aplicação e móvel. Na versão usando Pyton .. não teria problema no OpenCV Unity.

CONSTRUÇÃO DO ROBÔ FÍSICO FURBOT PARA O ENSINO DO PENSAMENTO COMPUTACIONAL – BCC

Gabriel Panca Ribeiro, Miguel Alexandre Wisintainer – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

gpribeiro@furb.br, maw@furb.br

Resumo: *Este trabalho desenvolveu o robô físico Furbot para um minijogo interativo, equipado com rodas omnidirecionais acopladas a motores de passo, controladas por um Arduino, além de uma placa ESP32-CAM para visualização do ambiente. Um sistema em Python foi criado para controle e reconhecimento de imagens utilizando a biblioteca OpenCV, permitindo a movimentação do robô em planos quadriculados de diversos tamanhos. Os resultados demonstraram a capacidade autônoma de movimentação do robô em um plano quadriculado, tornando-o apto para utilização no minijogo, graças ao módulo de câmera e ao sistema de processamento de imagem.*

Palavras-chave*: Robótica móvel. Processamento de imagem. Arduino. ESP32-CAM. Python. OpenCV. Furbot.*

1 INTRODUÇÃO

Segundo uma pesquisa feita por Barrence (2023), até 2025, faltarão 530 mil profissionais na área da programação. O estudo investigou o mercado tecnológico no Brasil, dialogando com lideranças de startups, especialistas e jovens em formação. Está sendo enfrentado um problema estrutural na educação brasileira que vai além da graduação. A pesquisa destaca a importância de integrar a tecnologia ao longo da vida, enfatizando a urgência de ações.

Nesta linha, o ensino do pensamento computacional é apresentado como uma solução. Para Wing (2006), o pensamento computacional é uma habilidade crucial que se baseia no entendimento dos processos de computação, tanto realizados por seres humanos quanto por máquinas. Esse modo de pensar fornece a coragem para solucionar problemas e criar sistemas que ultrapassam capacidades individuais. A importância do ensino do pensamento computacional está em seu potencial para capacitar não apenas cientistas da computação, mas todos, integrando-se às competências analíticas essenciais para enfrentar os desafios contemporâneos, assim como a leitura, a escrita e a aritmética são fundamentais para a formação (WING, 2006).

Para inserir o pensamento computacional de forma efetiva no âmbito educacional, é fundamental integrá-lo ao currículo desde as primeiras etapas de aprendizagem (BARRENCE, 2023). Nesse contexto, surge o projeto educacional Furbot (REPÚBLIKA), com a missão de facilitar o ensino das habilidades do pensamento computacional para alunos do ensino fundamental. A plataforma oferece uma gama diversificada de exercícios em diferentes níveis de dificuldade, promovendo desafios de lógica e programação que auxiliam no desenvolvimento do raciocínio lógico e habilidades alinhadas ao universo digital.

Uma pesquisa feita por Kohler et al. (2021), na qual foi aplicado os jogos do Furbot, mostrou que exercícios que envolvem o desenvolvimento de algoritmos, possuem uma influência positiva no engrandecimento do raciocínio lógico. Observou-se um impacto significativo, com um aumento de aproximadamente 9% na classificação dos alunos do 2º ano e 52% na dos alunos do 5º ano. Este incremento demonstra o potencial desses exercícios no aprimoramento do pensamento computacional e na produção de algoritmos por estudantes do ensino fundamental, ressaltando a relevância do tema para a educação nesta etapa de ensino.

O Furbot físico é uma proposta que proporciona uma experiência prática e interativa aos alunos. O robô faz parte de um minijogo, permitindo que os estudantes controlem sua movimentação em um campo quadriculado. Inicialmente, sensores de infravermelho eram utilizados para detectar a posição do robô no espaço. Contudo, visando aprimorar a experiência do aluno e tornar as aulas mais eficazes e envolventes, está planejada uma atualização na detecção de movimentos por meio de uma câmera. Desta forma, esta melhoria proporcionará uma interação mais intuitiva e enriquecedora.

Assim, este trabalho propõe apresentar a construção e evolução do robô físico. Essa é uma ferramenta essencial para fomentar o pensamento computacional, não apenas para enfrentar os desafios cotidianos, mas também para preparar os futuros profissionais que moldarão o cenário da programação, uma das profissões mais promissoras e demandadas para o futuro.

Diante deste contexto, este trabalho tem o objetivo de desenvolver o robô físico do Furbot para a utilização no seu minijogo. Os objetivos específicos são: (i) construir o Furbot físico; (ii) programar o robô para o minijogo; (iii) desenvolver um sistema de reconhecimento de imagens para a movimentação do robô.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção descreve brevemente os assuntos que fundamentarão o estudo a ser realizado: robótica no ensino de pensamento computacional, robótica móvel e processamento de imagem, Furbot e os trabalhos correlatos.

2.1 ROBÓTICA NO ENSINO DE PENSAMENTO COMPUTACIONAL

Para Nogueira e Borges (2016), existe uma necessidade de uma abordagem pedagógica inovadora, considerando o ambiente tecnológico em que os jovens estudantes estão imersos. A robótica é apresentada como uma ferramenta pedagógica promissora no ensino de disciplinas como lógica e linguagem de programação. Disciplinas que ensinam Lógica de Programação (LP) são fundamentais para o desenvolvimento do raciocínio lógico e matemático. No entanto, a complexidade percebida e a dificuldade em desenvolver o raciocínio lógico podem desmotivar os estudantes, levando a altos índices de reprovação. São propostas abordagens para ensinar conceitos de LP utilizando a metodologia problem-based learning (PBL) e kits robóticos como recursos pedagógicos, visando melhorar o engajamento e a compreensão dos estudantes.

Ao revisitar trabalhos relacionados, destaca-se a experiência positiva com o kit robótico Lego Mindstorms, evidenciando seu impacto positivo na aprendizagem, com melhorias nas notas e alta aceitação pelos alunos. A revisão sistemática revela a ampla utilização da robótica educacional, principalmente nas regiões Sudeste e Nordeste do Brasil, destacando sua aplicação interdisciplinar. No entanto, para a disciplina de LP, observa-se uma lacuna nas aplicações práticas, ressaltando a relevância do presente artigo ao propor estratégias específicas para o ensino de programação usando o ambiente de desenvolvimento do kit Lego Mindstorms (DE JESUS; CRISTALDO, 2014).

É concluído que, para o ensino de LP e habilidades de pensamento computacional, a seleção adequada de um kit de robótica alinhado ao perfil dos alunos é crucial. Alcançar o objetivo com alunos iniciantes depende da identificação e utilização de estruturas simples e claras, facilitando a compreensão do conteúdo. Destacam também a viabilidade de abordar conceitos de maneira prática, despertando o interesse dos alunos e promovendo uma compreensão aprimorada, sem a necessidade de recursos complexos ou extensivos do ambiente de programação (NOGUEIRA; BORGES, 2016).

2.2 ROBÓTICA MÓVEL E PROCESSAMENTO DE IMAGEM

O estudo de Bertoncini (2022) aborda a importância da robótica móvel e sistemas autônomos, destacando sua capacidade de transformar tarefas anteriormente executadas por humanos. Enfocando a navegação de robôs móveis, é proposto um sistema de navegação autônoma baseado no processamento de imagens captadas por uma câmera, distinguindo-se por sua abordagem dinâmica que se adapta a obstáculos e integra conhecimentos discretos e contínuos sobre navegação. A visão computacional é enfatizada como uma ferramenta crucial para a autonomia desses sistemas, permitindo-lhes interpretar e reagir a ambientes complexos de maneira semelhante à percepção humana.

As áreas de robótica móvel e sistemas autônomos vem se tornando destaque, enquanto robôs de manipulação estática transformaram linhas de montagem, os robôs móveis agora desempenham um papel crucial na realização de tarefas complexas e dinâmicas (DESOUZA; KAK, 2002). Contudo, para alcançar eficácia nesses empreendimentos, é imperativo que os robôs possuam métodos robustos de localização, dependendo do tipo de robô, sensores, sistema de coordenadas e ambiente específico em que operam (ADORNI et al., 2009).

Para DeSouza e Kak (2002), a robótica móvel desempenhará um papel crucial na resolução de tarefas como limpeza, busca, entrega de mercadorias, exploração de áreas hostis e resgate em cenários de desastres, que são pouco aplicados atualmente. E que o avanço das pesquisas reflete a importância contínua da robótica móvel na atualidade, impulsionando inovações e soluções para desafios do mundo real.

2.3 FURBOT

Projetado para a educação básica o Furbot é gratuito e de fácil acesso, podendo ser usado em contexto educativo formal ou não formal, é constituído por atividades desplugadas e jogos eletrônicos (REPÚBLIKA, 2022). A plataforma foi desenvolvida para minimizar as dificuldades de aprendizagem na lógica de programação, as atividades incluem movimentação em quatro direções e detecção de obstáculos, com níveis de dificuldade progressivos (TRIDAPALLI, 2019).

Originalmente concebido para apoiar o ensino de Introdução à Programação nos cursos de graduação em Ciência da Computação e Sistemas de Informação FURB, o Furbot foi adaptado em 2017 para o ensino fundamental. Esta adaptação permite que crianças de 3º e 4º anos desenvolvam habilidades de pensamento computacional, utilizando uma versão da plataforma que inclui um gerador de mundos e um console de comandos (TRIDAPALLI, 2019).

Além da versão digital, o projeto Furbot inclui componentes para a prática de computação desplugada, como jogos de tabuleiro e de cartas. Essas atividades introduzem os comandos que serão utilizados na versão digital, fazendo uso da aprendizagem cinestésica e permitindo aos estudantes trabalharem de forma individual ou cooperativa (REPÚBLIKA, 2022).

Com este contexto, pode-se ressaltar que o minijogo determinado para o robô, será um plano quadriculado, que possibilita criar diferentes desafios e regras para os jogadores. Então é necessário que, quando for movimentado, o robô possa se movimentar para frente e rotacionar para os lados, andando um quadrado por vez.

2.4 TRABALHOS CORRELATOS

Nesta seção são apresentados trabalhos que são relacionados a aprendizagem de alunos com a interação com a robótica, tendo um objetivo de evoluir o pensamento computacional de alguma forma. No Quadro 1 mostra um trabalho onde foram construídos vários robôs para ensinar diversos assuntos (CHITOLINA; NORONHA; BACKES, 2016).

Quadro 1 – Robótica educacional como tecnologia potencializadora da aprendizagem

Referência	Chitolina, Noronha e Backes (2016)
Objetivos	Os autores têm o objetivo de integrar a robótica educativa com o processo de ensino e aprendizagem, apresentada como uma metodologia que altera os papéis tradicionais de professor e aluno, incentivando a participação ativa e autônoma dos estudantes na construção do conhecimento.
Principais funcionalidades	Permite a construção de robôs e programação de algoritmos para concluir desafios, proporcionando aos alunos uma abordagem prática e lúdica para a compreensão de conceitos de física e lógica de programação.
Ferramentas de desenvolvimento	Kits LEGO®.
Resultados e conclusões	O autor ressalta que a robótica educativa, quando incorporada ao ensino, não apenas facilita a compreensão de conceitos complexos, mas também promove a construção ativa do conhecimento pelos estudantes.

Fonte: elaborado pelo autor.

O trabalho do Quadro 2 foram montados e programados robôs que poderiam se mover de forma autônoma (FERNANDES *et al*., 2018).

Quadro 2 – Robótica educacional como ferramenta para ensino de lógica de programação no ensino fundamental

Referência	Fernandes *et al*. (2018)
Objetivos	Introduzir estudantes ao universo da robótica educacional com uma abordagem prática e lúdica, incentivando o raciocínio lógico e a resolução de problemas cotidianos.
Principais funcionalidades	Permite os alunos criarem robôs originais e implementar funcionalidades utilizando fluxogramas para resolver desafios, utilizando sensores para controles de atuadores.
Ferramentas de desenvolvimento	Kits eletrônicos e plataformas como o Arduino, além do Modelix.
Resultados e conclusões	Foi concluído que os resultados demonstraram a capacidade dos alunos de criar e controlar seus próprios equipamentos robóticos. Também foi perceptível a capacidade de abstração na construção dos programas, executando os conceitos teóricos sem demonstrar relutância ao operar o ambiente de programação do software Modelix, por possuir um visual simples e sendo operado apenas com formas geométricas utilizadas nos conceitos aplicados em aulas.

Fonte: elaborado pelo autor.

O Quadro 3 cita como foi utilizado um robô, construído previamente, para resolver desafios de programação utilizando a movimentação do robô (ZANETTI; OLIVEIRA, 2015).

Quadro 3 – Ensino de programação com robótica pedagógica

Referência	Zanetti e Oliveira (2015)
Objetivos	Analisar o uso do Pensamento Computacional e promover habilidades como a resolução de problemas e autoavaliação.
Principais funcionalidades	Utilizar o Scratch 4 Arduino (S4A) para desenvolver sistemas que se comunicam com o Arduino, fazendo leituras de sensores para detectar objetos e desviar de obstáculos a partir de entradas e saídas.
Ferramentas de desenvolvimento	Ambiente de programação S4A, derivado do projeto Scratch do MIT, junto com o Arduino e sensores.

Resultados e conclusões	Os autores descrevem que os resultados quantitativos e qualitativos obtidos demostram, de maneira positiva, que o método aplicado auxilia a composição da solução desenvolvida pelo aluno. A presença da robótica pedagógica apoia a abstração empírica, permitindo o aluno extrair informações mais concretas do objeto ou das ações sobre o objeto.

Fonte: elaborado pelo autor.

3 DESCRIÇÃO

Nesta seção são descritas a especificação do robô e dos sistemas que envolvem seu funcionamento, apresentando seus requisitos, comunicações entre as placas, fluxos de processos por meio de diagramas e a implementação do robô desde sua construção.

3.1 ESPECIFICAÇÃO

O Quadro 4 apresenta os Requisitos Funcionais (RF) descritos, e o Quadro 5 corresponde aos Requisitos Não Funcionais (RNF).

Quadro 4 – Requisitos funcionais

RF01:	O robô deve se movimentar para frente
RF02:	O robô deve rotacionar para esquerda e direita
RF03:	O robô deve receber comandos por controle
RF04:	O robô deve receber comandos pelo sistema de processamento
RF05:	O robô deve interagir corretamente com os minijogos propostos
RF06:	O robô deve utilizar o reconhecimento de imagem para a locomoção

Fonte: elaborado pelo autor.

Quadro 5 – Requisitos não funcionais

RNF01:	Permitir criar um campo para o minijogo em espaços diversos
RNF02:	Utilizar o Arduino e o Python para o desenvolvimento das funções
RNF03:	Utilizar uma carcaça construída em isopor para melhorar o visual do robô

Fonte: elaborado pelo autor.

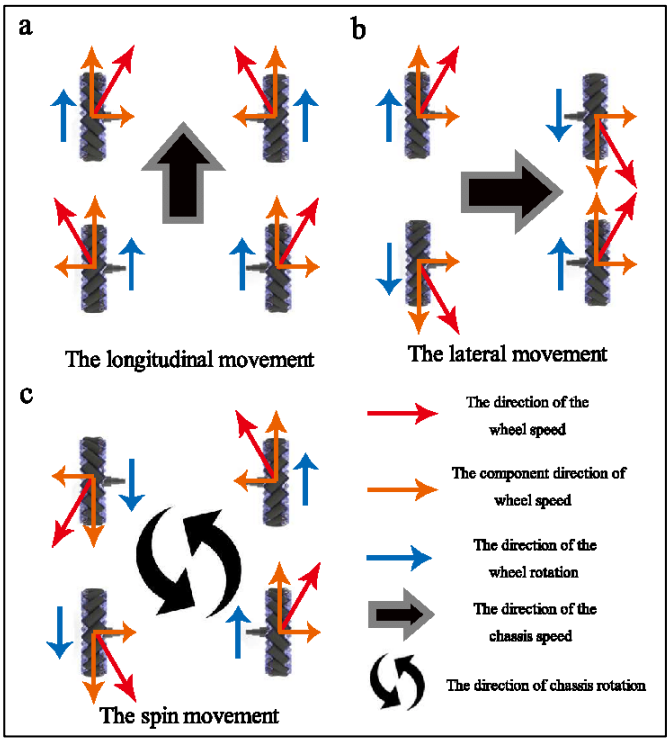
Na Figura 1 mostra as rodas omnidirecionais utilizadas na construção, que permitem que o robô se movimente para qualquer ângulo, neste trabalho foi utilizado a movimentação para esquerda e direita, sem necessariamente rotacionar o robô.

Figura 1 – Roda omnidirecional

Fonte: capturado pelo autor

Para a execução desses movimentos, basta rotacionar as rodas de maneiras específicas, conforme mostrado na Figura 2. Na seção (a) mostra a movimentação para frente, onde as rodas esquerdas giram de forma anti-horária e as rodas direitas rodam de forma horária. Na seção (b) explica como é possível fazer a movimentação para direita, onde as rodas da esquerda dianteira e direita traseira giram de forma anti-horária e as rodas da direita dianteira e esquerda traseira giram de forma horária. E por último na seção (c) demonstra como é possível rotacionar o robô para esquerda, bastando apenas girar todas as rodas em sentido horário.

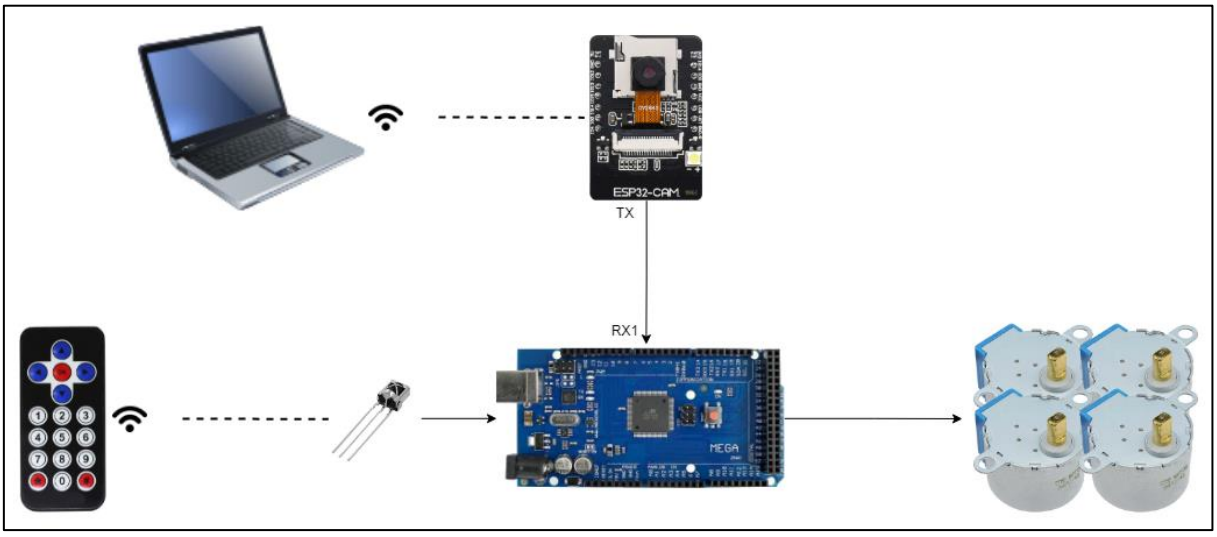
Figura 2 – Exemplo do uso de rodas omnidirecionais



Fonte: Yang *et al*. (2018)

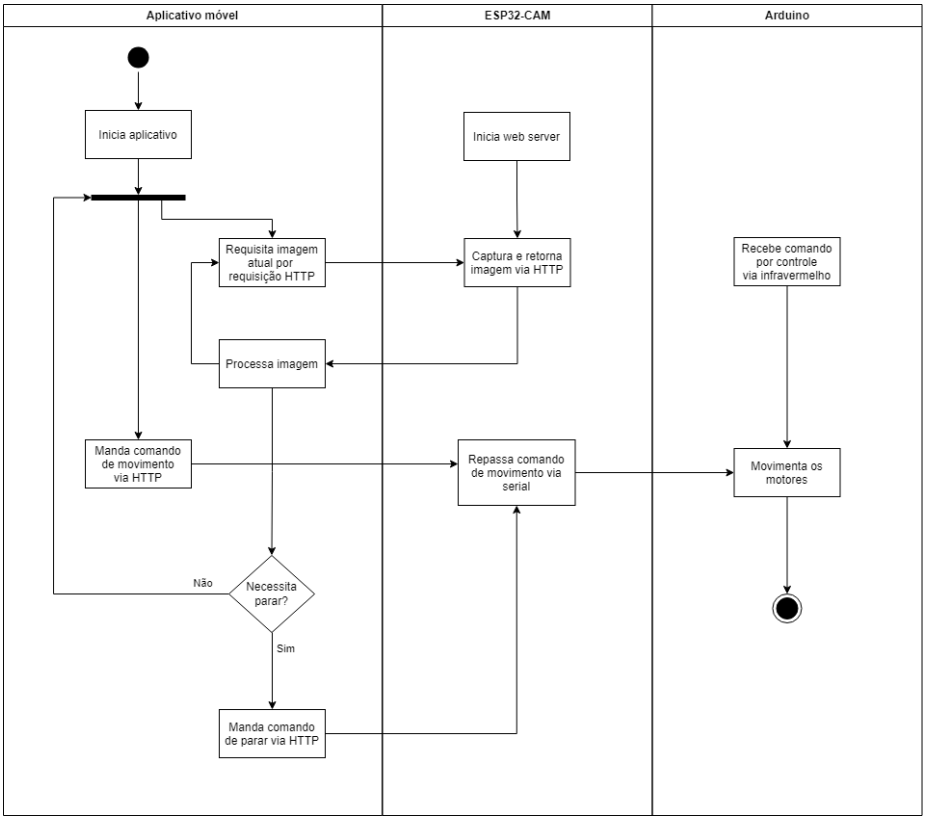
Para melhor entendimento das interações entre os *softwares* e *hardwares*, é apresentado o diagrama de distribuição na Figura 3, onde cita todas as partes separadas que fazem parte do sistema geral, iniciando a comunicação pelo sistema de processamento do computador, depois pelo ESP32-CAM, até o Arduino que finalmente manda os comandos para os motores, ou diretamente do controle infravermelho para o Arduino, até os motores. E o diagrama de atividade na Figura 4, descrevendo melhor a função de cada parte.

Figura 3 – Diagrama de distribuição



Fonte: elaborado pelo autor.

Figura 4 – Diagrama de atividade



Fonte: elaborado pelo autor.

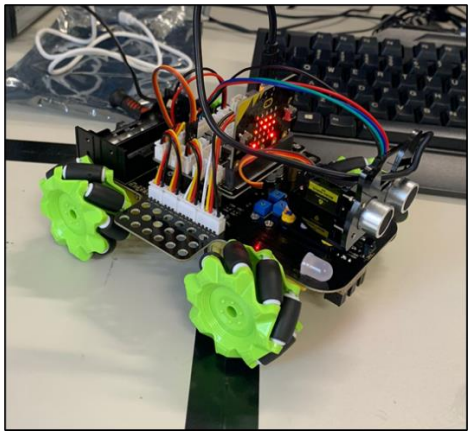
3.2 IMPLEMENTAÇÃO

Esta seção descreve a estrutura do robô e seu funcionamento, iniciando pela construção dos protótipos do robô físico, até sua versão final. Também será abordado como foi desenvolvido o fluxo de processamento e comandos necessários, incluindo o sistema de controle das rodas do Arduino, o servidor do ESP32-CAM e o sistema de processamento desenvolvido em Python. Os projetos podem ser encontrados no repositório do trabalho em: https://github.com/gpribeiro-furb/TCCFurbot

3.2.1 Construção do robô físico

A ideia da construção do robô físico foi iniciada antes deste trabalho, para contribuir com o projeto do Furbot, foi disponibilizado uma bolsa de estudo, na Furb, para construção e desenvolvimento do robô. Durante o programa, foram construídas três versões, o primeiro protótipo foi um conjunto de peças já preparadas para a montagem e funcionamento de um robô com controle em um aplicativo móvel, que serviu para testes do motor, sem sensores, onde foi verificado que o motor não atendia às necessidades do projeto, pela falta de torque necessário para uma movimentação mais precisa.

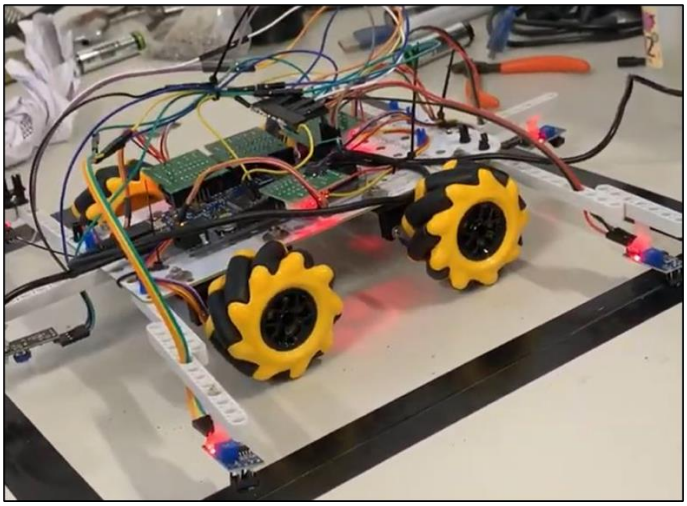
Figura 5 - Protótipo 1



Fonte: capturado pelo autor.

O segundo protótipo, foi construído com a colaboração do professor Miguel[1], foram compradas peças separadamente, onde iniciou a ideia de controlar quatro motores de passo, por meio de uma placa Arduino. Assim como no primeiro projeto, foram compradas rodas omnidirecionais, para possibilitar o robô andar de lado. Além de adicionar seis sensores infravermelhos, para o reconhecimento de espaço no minijogo, dois em cada lado, esquerdo e direito, um na frente, e um atrás.

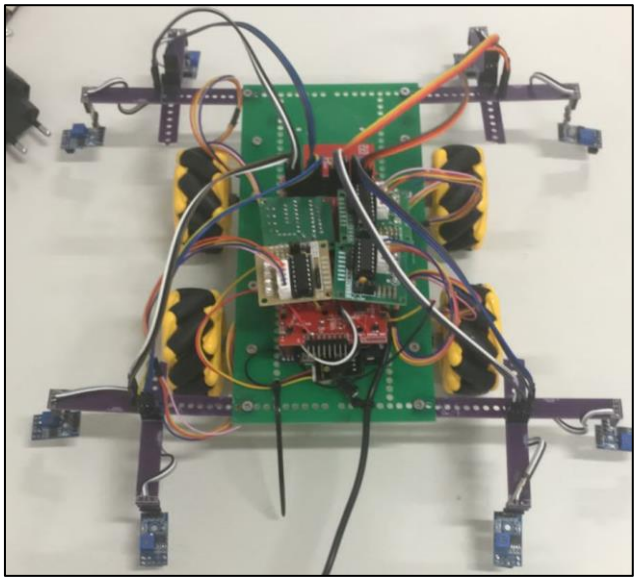
Figura 6 – Protótipo 2



Fonte: capturado pelo autor.

Para o terceiro protótipo, foi uma tentativa de ser uma versão final, onde funcionaria no minijogo, que também teve a ajuda do professor Miguel. Continha novas peças impressas em 3D para os motores, dois sensores infravermelhos a mais do que a versão anterior, um na frente e outro atrás, foi incluído uma câmera junto com uma placa separada para o reconhecimento de cartas utilizadas em outros minijogos do Furbot, além de um *shield* construído pelo professor para o Arduino que facilitaria as conexões das peças, como os *drivers* dos motores.

Figura 7 – Protótipo 3



Fonte: capturado pelo autor.

Neste trabalho foi iniciado um novo robô com algumas diferenças, foi construído um novo *shield* para o Arduino, foram removidos os sensores infravermelhos, as hastes que seguravam os mesmos e a função de reconhecimento de cartas. Para substituir os sensores, foi desenhado um *shield* para utilizar uma placa ESP32-CAM, onde foi acoplado a câmera utilizada para o processamento do ambiente ao redor do robô. A troca de sensores infravermelhos pela câmera foi necessária porque os sensores não apresentavam uma constância adequada, e a utilização de processamento de imagem permite que o robô seja usado em diversos ambientes e tabuleiros. O Arduino não pôde ser substituído pelo ESP32-CAM

[1] Miguel Alexandre Wisintainer, e-mail: maw@furb.br, currículo lattes: http://lattes.cnpq.br/0989671167867867.

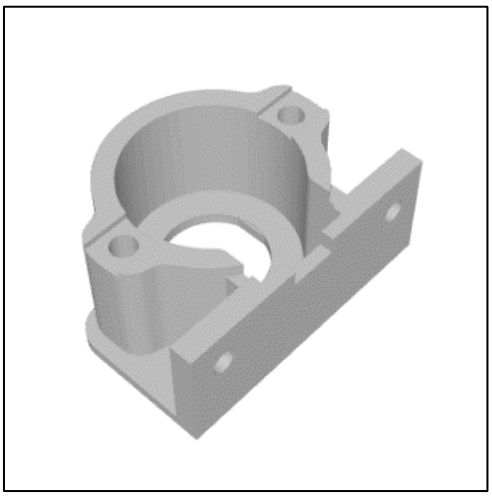
devido à falta de entradas e saídas suficientes, já que cada motor necessita de quatro entradas, além do módulo de receptor infravermelho. Além disso, a utilização das duas placas separadas oferece mais poder de processamento e facilita futuras melhorias.

Figura 8 – Modelo final do robô

Fonte: capturado pelo autor.

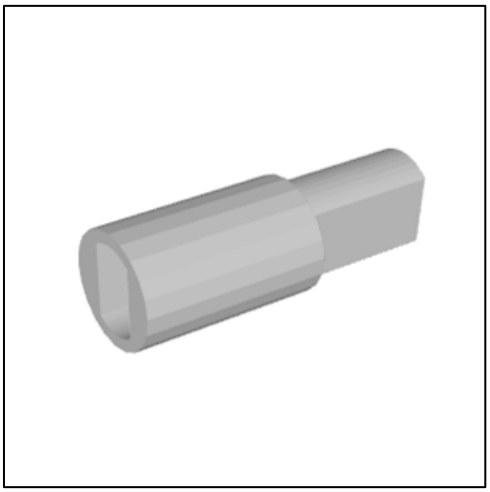
Foram utilizados dois modelos de peças impressas em 3D, uma para acomodar o motor de passo, que é parafusado na base do robô, mostrado na Figura 9, e outra que serve para encaixar a roda no motor de passo, como na Figura 10, onde originalmente, o buraco de encaixe da roda é maior do que a haste do motor.

Figura 9 – Suporte do motor de passo



Fonte: Robertson (2019).

Figura 10 – Adaptador da roda



Fonte: elaborado pelo professor Miguel.

Por questão de estética foi construído uma carcaça de isopor, com as cores e formato do Furbot, que fica por cima do robô e funciona como um suporte para a câmera, que é necessária para visualizar o ambiente na parte da frente. A câmera foi presa centralizada e mirada para baixo, por um pedaço de metal, parafusado na carcaça, deixando ela a aproximadamente 35 centímetros do chão. Foi configurado para a captura do vídeo ser na resolução 800x600, por questões de performance e nitidez. Um exemplo de captura é mostrado na Figura 11.

Figura 11 – Exemplo de captura pela câmera

Fonte: capturado pelo autor.

3.2.2 Funcionamento do sistema

Quando o Arduino é alimentado com energia, ele também alimenta o ESP32-CAM com 5V, que é o necessário para seu funcionamento. É possível alimentar o sistema por um cabo de energia, que deve ser conectado em uma tomada, ou por uma bateria de 6V. Ao ESP32-CAM ligar, ele cria uma rede Wi-Fi, onde é possível se conectar e utilizar o sistema de processamento desenvolvido com Python. Além disso, o ESP32-CAM liga seus serviços HTTP, se transformando em uma *representational state transfer* (REST) *application programming interface* (API).

Os *endpoints* utilizados foram o "/stream", onde permite o sistema de processamento criar uma conexão com o ESP32-CAM, que disponibiliza um fluxo contínuo de vídeo da visão da câmera em tempo real, e o "/comando", que é uma função do tipo POST, este serve para o sistema mandar textos para o ESP32-CAM, esses textos serão comandos, que em seguida serão enviados para o Arduino controlar a movimentação do robô. Esse texto é possível ser mandado por infravermelho, porém, por questão de precisão e velocidade, foi alterado para ser enviado via serial, pela porta TX principal do ESP32-CAM, para o RX1 do Arduino.

O sistema do Arduino contém uma função que executa em repetição, verificando se foi recebido alguma informação via serial ou por infravermelho. Os comandos possíveis são: ir para frente, girar para esquerda, girar para direita, ir para esquerda, ir para direita ou parar. Caso o comando for ir para frente, o Arduino controla os motores de passo para movimentar o robô para frente, quando é encontrado uma divisória é enviado o comando para parar, e logo após isso, comandos para girar ou andar, para esquerda ou direita, caso for necessário a correção de posição do robô no tabuleiro. Caso o comando for girar para esquerda ou direita, o Arduino controla os motores de passo para rotacionar o

robô para esquerda ou para direita respectivamente, e após esperar alguns segundos até o robô girar, é enviado comandos para correção do posicionamento do robô. E o comando parar, faz com que toda movimentação que esteja acontecendo seja cancelada, fazendo o robô parar.

Com esses comandos disponíveis, a única necessidade é visualizar o que a câmera está capturando, verificar a localização do robô no ambiente, e decidir o que deve ser feito. Para isso, é utilizado o sistema de processamento, desenvolvido em Python, onde ele faz toda a captura, processamento e envio dos dados.

O sistema de processamento também funciona em repetição, onde faz uma requisição inicial para conectar com a *stream* de vídeo do ESP32-CAM, e então processar cada quadro do vídeo. Com essas imagens em tempo real, é possível fazer um processamento de imagem para diferenciar divisórias, e com os dados dessa imagem processada, é feito um reconhecimento de linhas, em todos os quadros capturados. Os únicos dados de uma linha são duas coordenadas na imagem, o valor X e o valor Y, do início e do fim da linha.

Para o tratamento da imagem capturada pela câmera, foi utilizado o OpenCV para a seguinte sequência de processos: (i) conversão da imagem para tons de cinza; (ii) aplicado a função de desfoque gaussiano para normalização dos valores; (iii) aplicado a função de limite adaptativo, o algoritmo calcula um limite para pequenas regiões da imagem, o que possibilita lidar com diferentes condições de iluminação. Utilizando um tamanho de vizinhança de 15x15 *pixels*, aplica uma soma ponderada gaussiana, subtrai o valor constante 8 do limite calculado e inverte o resultado do limite binário, definindo os valores de pixel como 255, onde o valor do pixel original está abaixo do limite calculado, e como 0, caso contrário; (iv) para a detecção de bordas foi utilizado a função Canny, onde foi informado 50 como o primeiro limite para o procedimento de histerese, onde qualquer valor abaixo é considerado sem borda, e 150 para o limite superior, que qualquer valor acima deste é considerado uma borda forte; (v) e finalmente para a detecção de linhas, foi aplicado a função de transformação de linha Hough, onde foi informado o valor de 1 para a resolução da distância no acumulador, em pixels, 1 grau de precisão angular como a resolução do ângulo no acumulador, em radianos, 100 como *threshold*, que é o número mínimo de interseções necessárias no acumulador para detectar uma linha, 50 como o tamanho mínimo de uma linha para ser considerada e 50 como distância máxima entre segmentos de linha, que ainda serão considerados parte de uma única linha. O Quadro 6 demonstra o trecho do código fonte onde são utilizadas as funções.

Quadro 6 – Código fonte do sistema de processamento

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 0)
thresh = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 15, 8)
edges = cv2.Canny(thresh, 50, 150)
lines = cv2.HoughLinesP(edges, 1, np.pi / 180, threshold=100, minLineLength=50, maxLineGap=50)
```

Fonte: elaborado pelo autor.

Com essas informações em tempo real, de detalhes do ambiente detectados na parte frontal do robô, foi feito um sistema original, para agrupamento e processamentos das linhas. O objetivo, é saber se existe uma divisória na frente do robô, enquanto o robô se mexe. Para isso, em cada quadro do vídeo, é feito o reconhecimento de grupos de linhas, pois a mesma divisória reconhecida na imagem, pode conter várias linhas. Então, agrupando por orientação e valores médios das coordenadas, é possível ter um reconhecimento aproximado de divisórias sendo visualizadas pela câmera.

Foi definido que os comandos no sistema Python são as teclas "W" para andar para frente, "A" para girar para esquerda e "D" para girar para direita. Ao iniciar a movimentação para frente, o sistema ignora a linha que está à frente do robô, caso exista, e manda o comando para o ESP32-CAM, que envia para o Arduino, onde orienta os motores para andar para frente. Quando uma divisória reconhecida chega a um valor determinado, que significa que está perto do robô, o sistema manda o comando para parar, fazendo o Arduino parar todos os motores. Após parar, é verificado o ângulo da divisória em relação ao robô e calculada a quantidade de rotação necessária para que o robô fique paralelo a ela. Além disso, as divisórias à esquerda e à direita, que estão mais próximas do centro da visão, são verificadas para confirmar se o robô está no meio do quadrado. Caso necessário, é feito um cálculo para mover o robô para a esquerda ou direita, centralizando o Furbot no tabuleiro. Esses cálculos foram baseados em testes de movimento do Arduino com os motores de passo. Os valores foram testados manualmente e, por exemplo, concluiu-se que 255.000 execuções dos motores na velocidade máxima são necessárias para rotacionar o robô em 90 graus. A partir disso, foi feita uma equivalência para determinar quantas execuções são necessárias para rotacionar o robô de forma precisa. O mesmo método foi aplicado para centralizar o robô no quadrado.

Os comandos "A" e "D" enviam comandos para rotacionar o robô exatamente 90 graus, para esquerda ou direita respectivamente, sem a necessidade de verificar se foi encontrada alguma divisória. Também é verificado se são necessárias as correções de ângulo e centralização, porém, é feito após aproximadamente 7 segundos depois de iniciar a rotação.

Para a utilização dos motores de passo, foi utilizada a biblioteca AccelStepper, uma interface orientada a objetos para motores de passo e drivers de motor de 2, 3 ou 4 pinos. Ao iniciar o sistema do Arduino os motores são configurados com suas entradas e com a velocidade máxima, como mostra o Quadro 7, além de um exemplo de como funciona a

definição das rotações, inclui a função que faz o robô rotacionar para a direita, definindo a velocidade desejada nos motores, e a função para ativá-los.

Quadro 7 – Código fonte do Arduino

```
// Configurações dos pinos
//Esquerda dianteira
AccelStepper stepper1(AccelStepper::HALF4WIRE,17,15,16,14);
//Esquerda traseira
AccelStepper stepper2(AccelStepper::HALF4WIRE,57,55,56,54);
//Direita dianteira (inverso)
AccelStepper stepper3(AccelStepper::HALF4WIRE,8,10,9,11);
//Direita traseira
AccelStepper stepper4(AccelStepper::HALF4WIRE,29,25,27,23);

// Configuração da velocidade
stepper1.setMaxSpeed(1000);
stepper2.setMaxSpeed(1000);
stepper3.setMaxSpeed(1000);
stepper4.setMaxSpeed(1000);

// Função para rotacionar para direita
void girarDireita() {
  stepper1.setSpeed(-velocidade);
  stepper2.setSpeed(-velocidade);
  stepper3.setSpeed(velocidade);
  stepper4.setSpeed(velocidade);
}

// Função para ativar os motores
void runSteppers() {
  stepper1.run();
  stepper2.run();
  stepper3.run();
  stepper4.run();
}
```

Fonte: elaborado pelo autor.

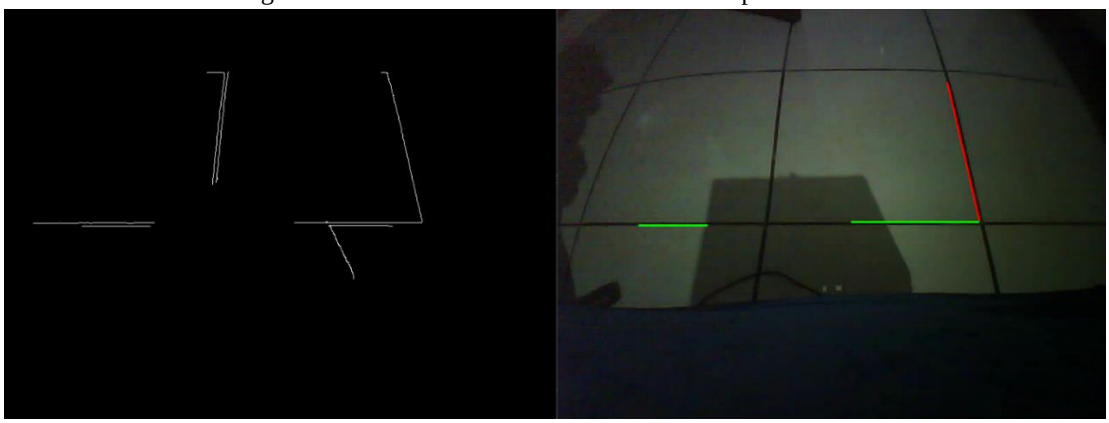
4 RESULTADOS

Nesta seção é apresentado os testes realizados com métodos de processamento de imagem e agrupamento de linhas detectadas, é citado o aplicativo móvel que substituiria o sistema de processamento e a carcaça feita com impressão 3D.

4.1 PROCESSAMENTO DE IMAGEM

No início do desenvolvimento do sistema de processamento, foram testadas algumas funções diferentes para o processamento de imagem, além de métodos de agrupamento e leitura das linhas encontradas nas imagens, que foram descartadas, pelos resultados incapazes de ter precisão e constância. O primeiro teste mostrado na Figura 12 foi feito sem a função de limite adaptativo, o que fez com que sombras, mesmo que fracas, impedissem a detecção correta de bordas. Assim, não era possível detectar divisórias se houvesse qualquer sombra, até mesmo a do robô.

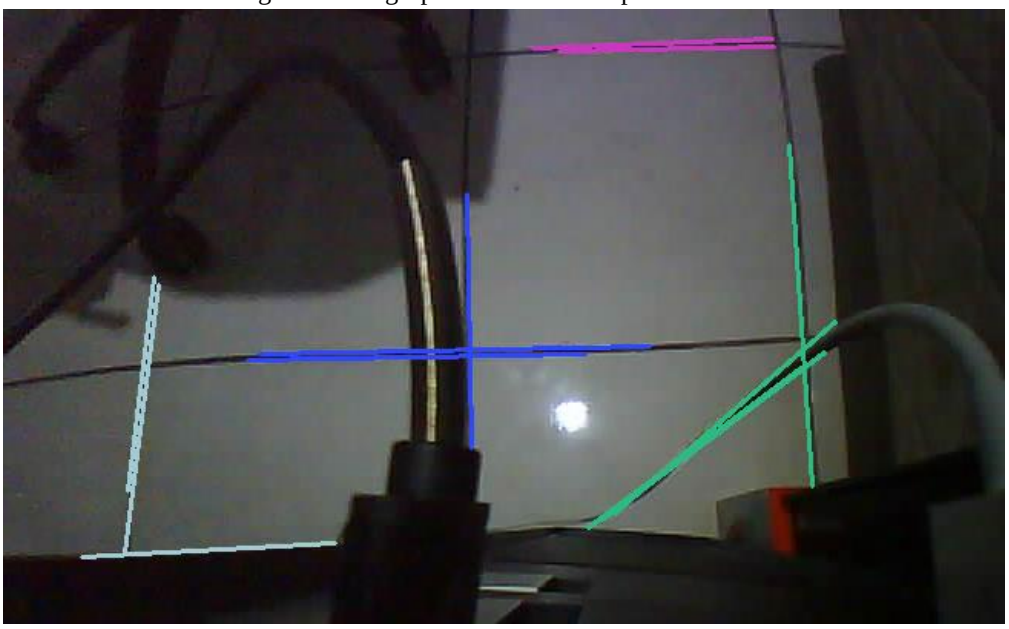
Figura 12 – Reconhecimento de linhas afetado pela sombra



Fonte: capturado pelo autor.

Antes de fazer o agrupamento manual das linhas, pelo posicionamento e ângulo, foi utilizado o algoritmo Density-Based Spatial Clustering of Applications with Noise (DBSCAN), que serve para encontrar *clusters* em um conjunto de dados com base na densidade de pontos de dados, o que não seria útil para a diferenciação de divisórias, pela sua falta de previsibilidade e constância, a Figura 13 mostra um exemplo de aplicação desse algoritmo.

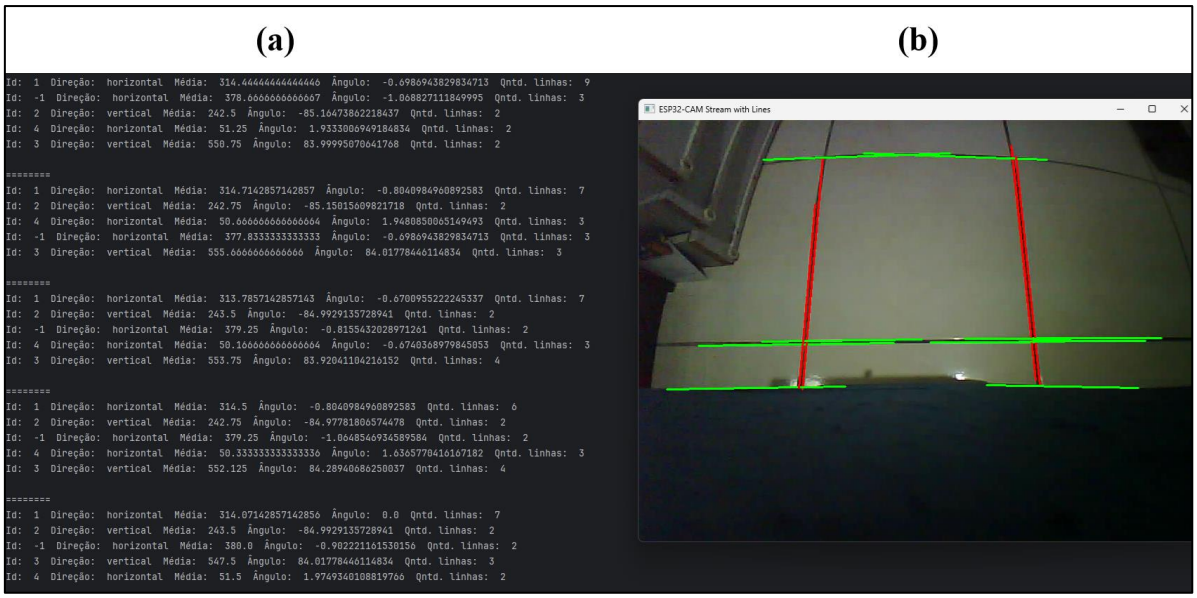
Figura 13 – Agrupamento de linhas por DBSCAN



Fonte: capturado pelo autor.

A Figura 14 demonstra um exemplo do resultado do sistema atual, onde a seção (a) mostra o *console* do sistema de processamento, que imprime, em todos os quadros, as divisórias reconhecidas, assim como sua direção, horizontal ou vertical, a média das coordenadas, dos valores de X para divisórias verticais e de Y para divisórias horizontais, a média da angulação das linhas e a quantidade de linhas reconhecidas na divisória. E as linhas detectadas, desenhadas com cores verdes para linhas horizontais e vermelhas para linhas verticais, por cima da visão original, na seção (b).

Figura 14 – *Console* do sistema de processamento (a) e a visão original com as linhas reconhecidas (b)



Fonte: capturado pelo autor.

4.2 APLICATIVO MÓVEL

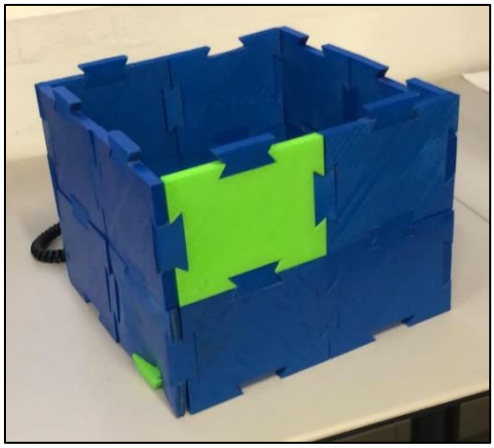
Durante o trabalho foi idealizado um aplicativo para celular que funcionaria assim como o sistema de processamento feito em Python. O projeto foi desenvolvido para Android, mas não foi citado por dois motivos, o maior problema era a impossibilidade de processar a visualização da câmera por *stream* de vídeo, assim, utilizava-se a função de captura de imagem do servidor criado pelo ESP32-CAM, o que causava uma grande perda de velocidade na visualização em tempo real do ambiente. Além disso, era necessário um celular com alto poder de processamento devido

à exigência de rapidez e capacidade para lidar com a grande quantidade de imagens por segundo. Então é necessária uma melhora de performance para usá-lo efetivamente.

4.3 CARCAÇA FEITA COM IMPRESSÃO 3D

Juntamente com o aluno Jonas[2], foi iniciado uma carcaça feita por uma impressora 3D, construída por ele. Como mostrado na Figura 15, foi concluído a parte do torso, com peças encaixáveis, tendo o mesmo tamanho da carcaça de isopor.

Figura 15 – Carcaça feita com impressão 3D



Fonte: capturado pelo Jonas.

5 CONCLUSÕES

Através do desenvolvimento do trabalho, foi possível cumprir com o objetivo principal de desenvolver o robô físico do Furbot para a utilização no seu minijogo, ele é capaz de se movimentar em qualquer plano quadriculado, onde as divisórias são pretas e o resto do piso branco, sem necessidade de manipulação do código, além de ser programado para se ajustar automaticamente ao centro do quadrado, facilitando o uso do robô.

A construção do robô físico foi desenvolvida de uma forma eficiente, depois de vários protótipos, que viabiliza alterações e melhorias no seu *hardware*, como adições de novos módulos, ou até uma carcaça com outros materiais, assim como a versão impressa em 3D do modelo original do Furbot.

O objetivo de programar o robô para o minijogo foi implementado, onde o robô poderá receber comandos de ir para frente ou girar para esquerda ou direita, podendo se locomover livremente para qualquer quadrado do tabuleiro. Isso possibilita integrações futuras com alguma aplicação, por ser apenas um serviço HTTP, como desafios similares ao aplicativo do Furbot, para uma experiência mais interativa.

O objetivo de desenvolver um sistema de reconhecimento de imagens para a movimentação do robô foi atendido, que possibilita o uso do robô em qualquer ambiente quadriculado, independentemente do tamanho do quadrado, desde que as linhas laterais possam ser vistas pela câmera, ao contrário dos primeiros protótipos com sensores infravermelhos, que apenas funcionariam em tamanhos específicos de quadrados no tabuleiro.

Então, pode-se concluir que o robô físico foi construído de forma com que possa ser utilizado para o minijogo proposto. Além de garantir mais constância e liberdade de mudança no ambiente de jogo, pela implementação de processamento de imagem por câmera ao invés de detecção por sensores infravermelhos. Suas limitações são relacionadas ao sistema de processamento, que neste trabalho foi desenvolvido em Python, por questões de performance.

A partir deste contexto, as possíveis extensões para esse trabalho são:

a) desenvolvimento de um minijogo virtual para integração e facilidade de controle do robô;

b) melhorar a performance do aplicativo móvel;

c) melhoria da detecção de linhas para anular ruídos de brilhos e sombras;

d) finalizar a carcaça feita na impressora 3D para melhorar a estabilidade e suporte da câmera;

e) implementar comandos por voz.

[2] Jonas Fernando Schuh, e-mail: jonasschuh@hotmail.com.

REFERÊNCIAS

ADORNI, Giovanni; CAGNONI, Stefano; ENDERLE, Stefan; KRAETZSCHMAR Gerhard, K.; MORDONINI, Monica; PLAGGE, Michael; RITTER, Marcus; SABLATNÖG, Stefan; ZELL, Andreas. Vision for mobile robot navigation: a survey. 2002. Disponível em: https://ieeexplore.ieee.org/document/982903. Acesso em: 5 dez. 2023.

BARRENCE, André. A escassez dos profissionais de tecnologia no Brasil e seu consequente impacto no ecossistema de startups. 2023. Disponível em: https://blog.google/intl/pt-br/produtos/a-escassez-dos-profissionais-de-tecnologia-no-brasil-e-seu-consequente-impacto-no-ecossistema-de-startups/. Acesso em: 18 set. 2023.

BERTONCINI, Joao Paulo Scarabelo. Coordenação de robô autônomo por meio de visão computacional. 2022. Dissertação (Mestrado em Informática) - Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2022. Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/29473. Acesso em: 29 set. 2023

CHITOLINA, Renati F.; NORONHA, Fabrícia; BACKES, Luciana. A Robótica Educacional como tecnologia potencializadora da aprendizagem: das ciências da natureza às ciências da computação. **EDUCAÇÃO, FORMAÇÃO & TECNOLOGIAS**, Monte da Caparica, v. 9, n. 2, p. 56-65, 2016. Disponível em: https://eft.educom.pt/index.php/eft/article/view/199/174. Acesso em: 25 set. 2023.

DE JESUS, Leandro; CRISTALDO, Marcia F. Uma abordagem utilizando LEGO Mindstorms Education EV3 para verificar o desempenho acadêmico dos estudantes do Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso do Sul do Câmpus Aquidauana. 2014. Disponível em: http://milanesa.ime.usp.br/rbie/index.php/sbie/article/view/3066. Acesso em: 5 dez. 2023.

DESOUZA, Guilherme N.; KAK, Avinash C. Vision for mobile robot navigation: a survey. 2002. Disponível em: https://ieeexplore.ieee.org/document/982903. Acesso em: 5 dez. 2023.

FERNANDES, Manasses; SANTOS, Camila A.; SOUZA, Edmar E.; FONSECA, Marcos Uma abordagem utilizando LEGO Mindstorms Education EV3 para verificar o desempenho acadêmico dos estudantes do Instituto. In: WORKSHOP DE INFORMÁTICA NA ESCOLA (WIE), 24. , 2018, Fortaleza, CE. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2018. p. 315-322. Disponível em: https://sol.sbc.org.br/index.php/wie/article/view/14343/14188. Acesso em: 25 set. 2023

KOHLER, Luciana P. ; MATTOS, Mauro M.; LOPES, Mauricio Capobianco; FRONZA, Leonardo; SILVEIRA, Heitor Ugarte Calvet da; FIBRANTZ, Guilherme; ROSA, Vitor Lourenço da; SON, Lucas Hong Lae. Análise dos Resultados de um Estudo de Caso Aplicando Pensamento Computacional no Ensino Fundamental com Foco na Produção de Algoritmos. In: WORKSHOP DE INFORMÁTICA NA ESCOLA (WIE), 27. , 2021, On-line. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021. p. 106-115. Disponível em: https://sol.sbc.org.br/index.php/wie/article/view/17839/17673. Acesso em: 25 set. 2023.

NOGUEIRA, Flávia Z.; BORGES, Marcos A. PBL e robótica no ensino de conceitos de Lógica de Programação. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 24. , 2016, Porto Alegre. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2016. p. 2293-2302. ISSN 2595-6175. Disponível em: https://sol.sbc.org.br/index.php/wei/article/view/9673/9574. Acesso em: 29 set. 2023

REPÚBLIKA. Furbot, 2023. Disponível em: https://furbotldtt.wixsite.com/my-site-1. Acesso em: 4 dez. 2023.

ROBERTSON, John. Stepper Motor Base for 28BYJ-48, 2019. Disponível em: https://www.thingiverse.com/thing:3959310/files. Acesso em: 17 jul. 2024.

TRIDAPALLI, Joan G. Pensamento computacional e gamification: relato de um experimento na plataforma Furbot. 2019. Disponível em: https://www.furb.br/dsc/arquivos/tccs/monografias/2019_1_joan-gianesini-tridapalli_monografia.pdf. Acesso em: 19 jul. 2024.

WING, Jeannette M. Pensamento computacional. **Educação e Matemática**, (162), v. 49, n. 3, p. 2-4, mar. 2006.

YANG, Xiong; ZHANG, Hongbin; CHENG, Tianyou; NI, Xuebin. An Omnidirectional and Movable Palletizing Robot based on Computer Vision Positing. 2018. Disponível em: https://www.researchgate.net/publication/328995037_An_Omnidirectional_and_Movable_Palletizing_Robot_based_on_Computer_Vision_Positing. Acesso em: 19 jul. 2024.

ZANETTI, Humberto A.; OLIVEIRA, Claudio L. Práticas de ensino de Programação de Computadores com Robótica Pedagógica e aplicação de Pensamento Computacional. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (CBIE), 4., 2015, Maceió. **Anais** [...]. Maceió: Universidade Federal de Alagoas, 2015, p. 1236-1245. Disponível em: http://milanesa.ime.usp.br/rbie/index.php/wcbie/article/viewFile/6268/4389. Acesso em: 29 set. 2023.

ANEXO A – LISTA DE COMPONENTES ELETRÔNICOS

Neste anexo é descrito todas as peças necessárias para a montagem do robô no Quadro 8.

Quadro 8 – Peças

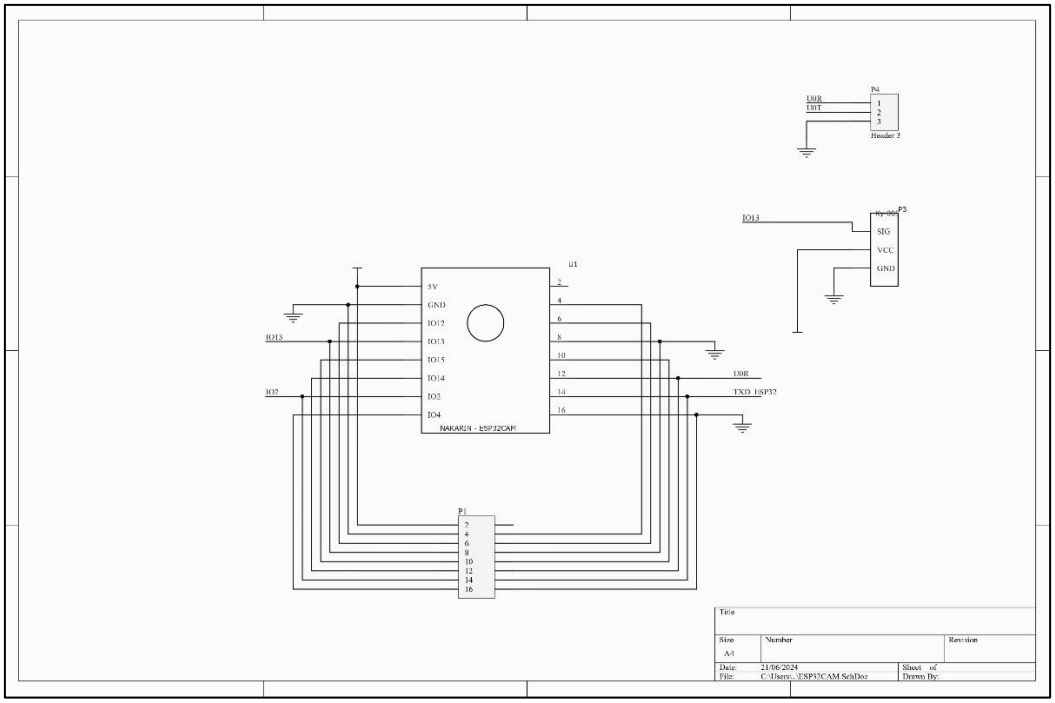
Descrição	Quantidade
Placa Mega 2560	1
ESP32-CAM OV2640	1
Módulo de câmera fisheye para ESP32-CAM	1
Motor de Passo 5V - 28BYJ-48	4
Drive Para Motor de Passo 5V - ULN2003	4
Roda Omnidirecional 60mm	4
Módulo LM 2596 S Step Down	1
Chave Táctil KFC-A06-3X6X2,5 - 2T 180G SMD	1
VS1838B - Foto Transistor Receptor Infravermelho - 3 Terminais	1
Modulo Emissor LED KY-005	1
Bateria Selada Unipower 6V / 2,8A	1

Fonte: elaborado pelo autor.

ANEXO B – ESQUEMAS ELÉTRICOS

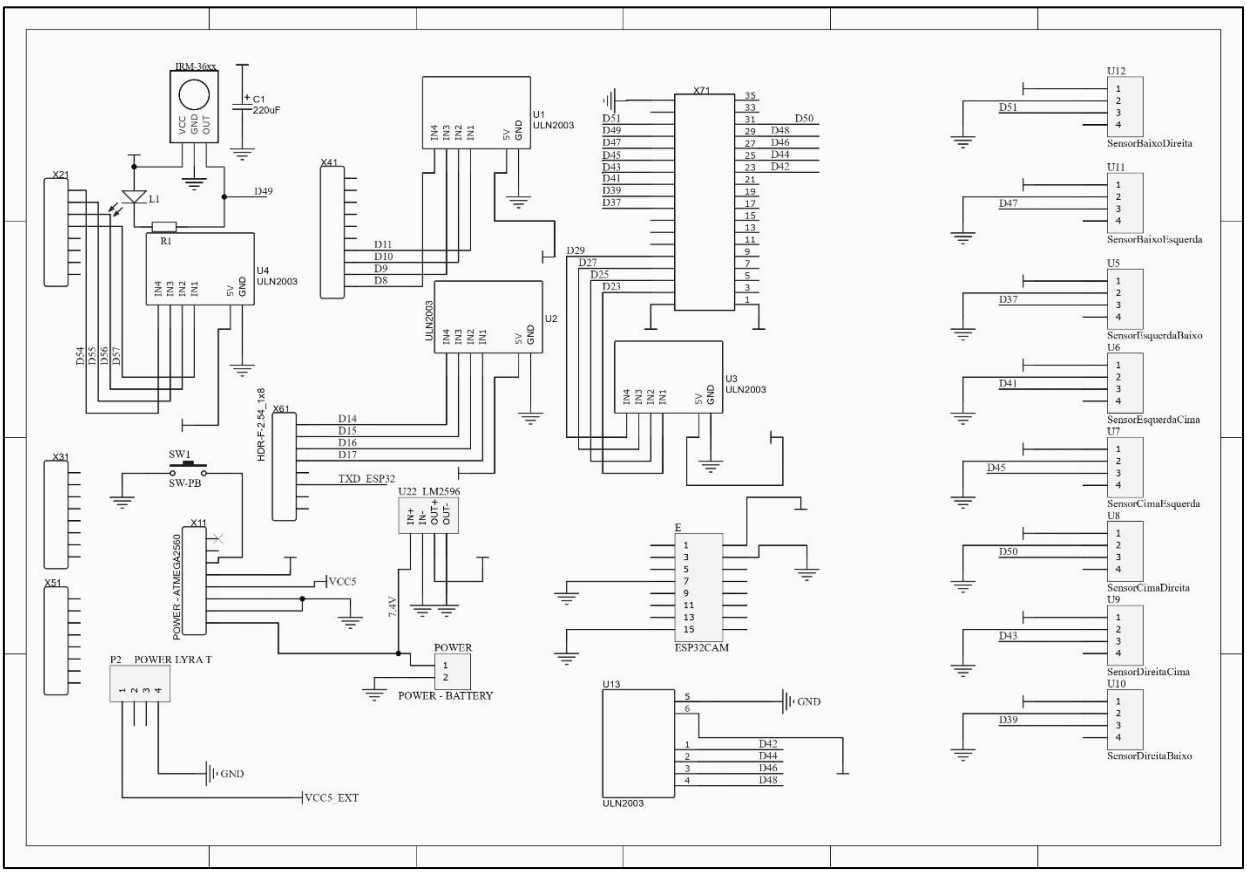
Neste anexo são apresentados os esquemas elétricos construídos pelo professor Miguel, a placa utilizada placa o ESP32-CAM demonstrada na Figura 16 e para o Arduino na Figura 17.

Figura 16 – Esquema elétrico da placa utilizada para o ESP32-CAM



Fonte: elaborado pelo professor Miguel.

Figura 17 – Esquema elétrico da placa utilizada para o Arduino



Fonte: elaborado pelo professor Miguel.