Centro Federal de Educação Tecnológica de Minas Gerais
Programa de Educação Tutorial
Engenharia Mecatrônica

Guia de Montagem do Robô

Divinópolis

Resumo

Este guia reúne alguns dos conhecimentos básicos necessários aos que desejam participar da 11° competição interna de robótica (2025), uma iniciativa pensada como um meio para divulgação da tecnologia, da mecatrônica e da robótica, em especial da robótica móvel, para discentes do CEFET e das escolas públicas de ensino médio da região de Divinópolis. Este projeto é composto por três etapas:

1. Divulgação;
2. Treinamento online e desenvolvimento das equipes;
3. Encontro acadêmico, no qual inclui a competição de futebol de robôs.

A competição interna de robótica (CIR) constitui-se em uma iniciativa que espera, por meio do encontro dentro das regras e da ética, confrontar diferentes interpretações e implementações de soluções das tarefas propostas como desafios, premiando assim o conhecimento e o trabalho, enfim, a competência de nossos jovens. Apesar de ter um papel relevante no contexto do projeto, devemos destacar que a competição não é o mais importante, mas sim o processo ao longo de todo o projeto, desde a possibilidade de levar a oportunidade aos interessados em ter contato com a programação e com a robótica, de desenvolvimento ao longo do treinamento, do trabalho nas equipes e do encontro acadêmico que será feito em outubro. Portanto, senhores, agradecemos pela participação e pedimos que “apertem os cintos, pois a viagem vai começar”.

Sumário

1 A competição interna de robótica

As Competições Internas de Robótica da unidade de Divinópolis do CEFET-MG foram concebidas em 2011 pelo Grupo de Estudos em Robótica (GER), como uma forma de se levar aos alunos interessados uma possibilidade de aplicar conhecimentos vistos em disciplinas da área da robótica, bem como algo de programação, matemática, física, etc. Para isso, pensou-se que a organização de uma competição acadêmica, nos moldes das competições oficiais como a CBR (Competição Brasileira de Robótica) e a OBR (Olimpíada Brasileira de Robótica) poderiam auxiliar nesse propósito de uma forma construtivista, ou seja, incentivando o aprendizado por meio da aplicação dos conhecimentos abordados e a comprovação por parte das equipes de que suas ideias para a solução das tarefas propostas estavam (ou não) na direção certa. No entanto, um importante diferencial é que a partir de 2017, foi incluído nessa iniciativa a divulgação do projeto para as escolas públicas de nível médio e, principalmente, os treinamentos para os participantes das equipes, num efeito multiplicador do conhecimento.

Em 2022, a fim de que suas atividades pudessem acontecer a distância, manteve-se a reformulação implementada no ano de 2021. A edição do ano anterior mostrou-se eficaz ao iniciar todos os envolvidos no ambiente dos simuladores computacionais, que tanto tem revolucionado diversas áreas, dentre elas a da robótica. O software de simulação continuou sendo o Webots devido à sua qualidade, o uso de linguagens de programação conhecidas e úteis, além dos recursos disponíveis nele, como sensores e motores. A disponibilidade de instalação gratuita do simulador e sem prazo para utilização oferecida pela Cyberbotics foi importante e permitiu que a iniciativa se mantivesse totalmente gratuita para os participantes interessados.

A 10ª Competição Interna de Robótica de 2024/25 foi um marco para a CIR, na qual foi implementada pela primeira vez, uma competição real de futebol de robôs, ainda que controlados remotamente. Seguindo o padrão da 10ª edição, a 11ª CIR terá uma competição real de futebol de robôs, com a diferença dos robôs autônomos. A implementação de novos sensores, e da visão computacional será crucial para o acontecimento dessa edição.

A competição acontecerá no Laboratório de Robótica do CEFET, na mesa principal localizada no centro do laboratório. O jogo ocorrerá em um campo de MDF, e um suporte para a câmera será fixado no teto para capturar as imagens para a visão computacional, e auxiliar no monitoramento das partidas. A data prevista para a competição ainda não foi definida, mas ocorrerá no final do segundo semestre de 2025.

1.1 Inscrições

As inscrições serão realizadas por meio de formulário eletrônico no **Google Forms**, em data que será divulgada com antecedência. Todas as informações oficiais estarão disponíveis exclusivamente no site institucional do PET Engenharia Mecatrônica, acessível em: https://i3dpet.wixsite.com/pet-eng-mecatronica.

Atenção: como será necessário identificar as equipes e seus participantes, será solicitado além dos nomes completos, números de matrículas e contatos (e-mail e telefone de WhatsApp), serão necessárias fotos recentes (semelhantes ao padrão 3x4) dos componentes da mesma. Como dito acima, é pedido também o contato de celular de cada participante, preferencialmente de WhatsApp, com o intuito de viabilizar uma comunicação mais ágil entre a organização e as equipes, e também para reuniões informativas de possíveis alterações ou para a tomada de decisão a respeito de alguma questão que se apresente antes ou durante a competição. Vale destacar também que as fotos têm importante papel de registro e divulgação da iniciativa, sendo que os participantes ao se inscreverem para a competição estarão concordando com essas funções, autorizando o uso e divulgação das fotos para esses fins de organização e registro da CIR 2025. Para as equipes convidadas, será preciso também os comprovantes ou declarações de matrícula dos alunos participantes. Não serão aceitas inscrições de equipes após o período estipulado. Assim, indica-se o prévio planejamento da inscrição, e envio dos dados completos dentro do prazo determinado, para que possíveis transtornos possam ser evitados.

1.2 Etapas

A CIR é dividida em 2 etapas. A primeira terá início após as inscrições, na qual serão feitos encontros presenciais para auxiliar os participantes na montagem de alguns componentes, assim como vídeos nos quais ajudarão também na programação do robô, essa etapa servirá como um treinamento para as equipes. A segunda será a competição efetivamente, que ocorrerá em 2 fases, sendo a fase de grupos, com grupos de 3 equipes, em que o primeiro colocado e os dois melhores colocados seguintes avançam. Tem-se também o temido mata–mata, no qual apenas uma equipe avança, vale ressaltar que são disputas de JOGO ÚNICO. Além disso, destaca-se que será necessário o desenvolvimento de táticas para que a equipe supere o adversário e chegue ao objetivo maior do futebol, o GOL.

2 Regras da Partida

2.1 Duração da partida

Cada equipe irá jogar em um esquema de todos contra todos. Assim serão considerados jogos de ida e jogos de volta.

OBSERVAÇÃO: Pode ser que ocorram apenas jogos de ida, no sistema de todos contra todos.

Na etapa do chaveamento as partidas terão duração total de 4 minutos, divididos em dois tempos de 2 minutos cada. Serão computados 3 pontos para a equipe vencedora e 1 ponto para cada equipe em caso de empate. Caso haja empate na pontuação dos confrontos diretos, o número de vitórias e o saldo de gols, serão utilizados como critérios de desempate.

Os dois melhores times se classificarão para a final do campeonato, seguindo o mesmo padrão de tempo das partidas tradicionais. Nessas partidas finais, caso haja empate no tempo normal, será dada uma prorrogação de 1 minuto. Se na grande final a prorrogação permanecer empatada , haverá uma segunda prorrogação de 1 minuto, com a troca de lado entre os times Caso a prorrogação termine empatada, haverá um tempo extra no qual irá valer a regra do "gol de ouro".

2.2 Início da partida

A equipe mandante (cujo time jogará do lado esquerdo) iniciará a partida com a posse de bola. Antes de um chute direto ao gol, é obrigatório que o robô passe a bola pelo menos uma vez para outro robô da mesma equipe.

2.3 Faltas

Serão consideradas faltas qualquer tipo de contato físico entre os robôs adversários. Tendo o tempo pausado até a cobrança da falta. As infrações serão classificadas da seguinte forma:

- **Falta de Defesa:** Ocorre quando um robô defensor intercepta um adversário que está conduzindo a bola sem respeitar o espaço de movimentação do mesmo. Se o robô atacante estiver em posse da bola e se deslocando, qualquer colisão causada por um defensor

em movimento será considerada falta da defesa. Caso um robô cometa uma falta de defesa: uma falta leve de defesa resultará em uma advertência para a equipe. Se a equipe acumular três advertências, receberá um cartão amarelo. Uma falta mais forte poderá resultar diretamente na aplicação de um cartão amarelo. A aplicação de um segundo cartão amarelo equivalerá a um cartão vermelho para a equipe, mesmo que as infrações tenham sido cometidas por robôs distintos. Já uma falta grave ou que impeça um gol iminente será passível de cartão vermelho direto.

- **Falta de Ataque:** Caso o robô atacante colida com um defensor que já estava parado, a infração será considerada falta de ataque, seguindo as mesmas condições supracitadas anteriormente na falta de defesa. É permitido apenas um robô de cada equipe dentro da área do goleiro, caso haja mais de um robô, haverá a troca de posse de bola. Caso haja invasão com o robô da equipe adversária na área cobrança de falta, o mesmo será advertido. O tempo de posse de bola máximo é de 25 segundos.

- **Disputa de Bola:** Se dois robôs estiverem disputando a bola à mesma distância, sem vantagem clara de posição, o contato será considerado falta e a posse de bola será passada para na marcação mais perto, sendo alternado o infrator. No entanto, se um dos robôs estiver um pouco à frente e sofrer um impacto, será marcada falta contra o robô que estava atrasado

2.4 Regras de Movimentação e Chutes

Quando a bola estiver no meio de campo e prestes a sair dessa zona, o time que estiver com a posse precisa tocar na bola antes de avançar. Exceto pela regra acima, os robôs podem chutar a bola de um lado ao outro do campo livremente, inclusive tentando gols por tabela e de longa distância.

3 O Robô

Para o desenvolvimento deste projeto, será utilizado um robô composto pelos componentes descritos na Tabela 1 e na Tabela 2. A primeira tabela apresenta os componentes elétrônicos a serem adquiridos pela internet, incluindo informações de preço, quantidade e fornecedores. A segunda tabela reúne os componentes de caráter mecânico, como bases e carapaça em MDF, peças impressas em 3D, placas de circuito impresso (PCB) e elementos de fixação, com valores estimados.

Esses componentes são fundamentais para a montagem e integração dos diferentes circuitos que compõem o robô, possibilitando a aplicação prática de conhecimentos em eletrônica, robótica e programação. Além do caráter técnico, a construção do robô representa uma oportunidade única de aprendizado, uma vez que envolve desde o planejamento até a execução prática, estimulando a criatividade, a resolução de problemas e o trabalho em equipe.

A experiência de montar este robô vai muito além da simples junção de componentes: trata-se de um projeto desafiador e inovador, capaz de consolidar conteúdos teóricos em situações reais e práticas. Por isso, este projeto se destaca como uma atividade altamente enriquecedora, tanto em termos acadêmicos quanto profissionais, tornando o processo de desenvolvimento extremamente valioso para a formação dos estudantes.

Ressalta-se que a lista de componentes e fornecedores apresentada tem caráter sugestivo. Em caso de indisponibilidade de algum item, poderá ser realizada a substituição por outro equivalente, desde que assegurada a mesma função e desempenho necessários para garantir a continuidade e a eficácia do projeto.

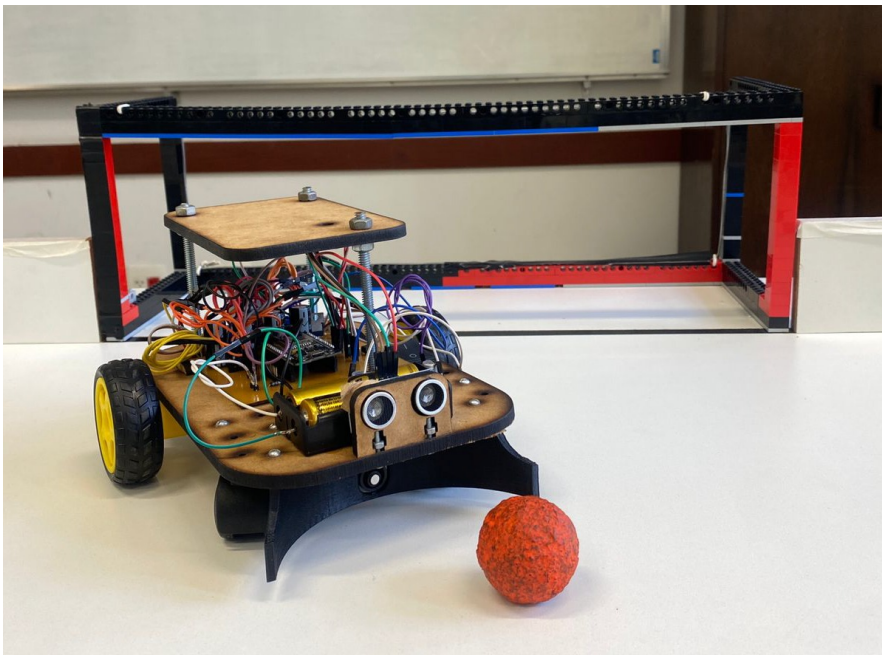

Figura 1 – Vista frontal do robô com a bola

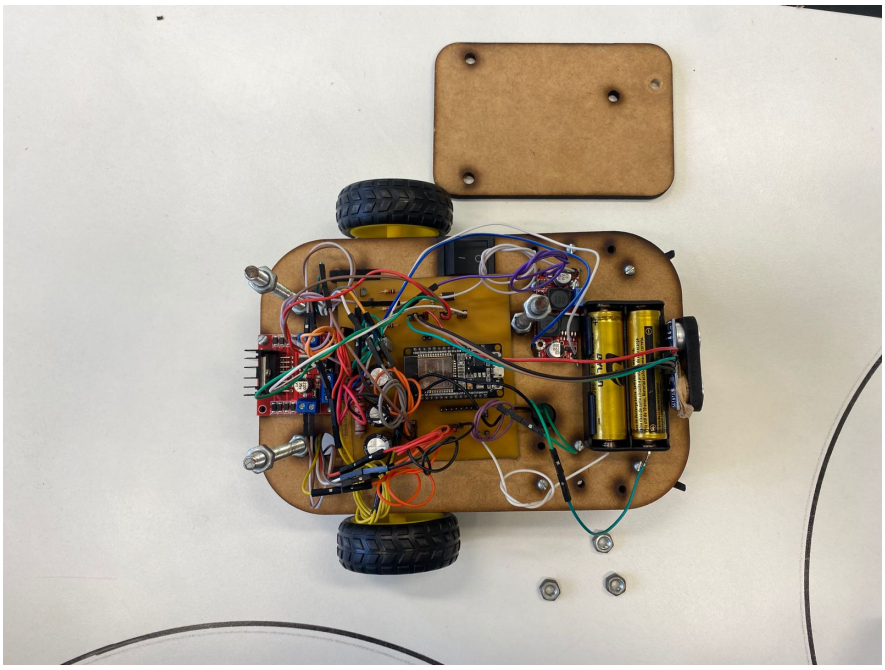

Figura 2 – Vista superior do robô

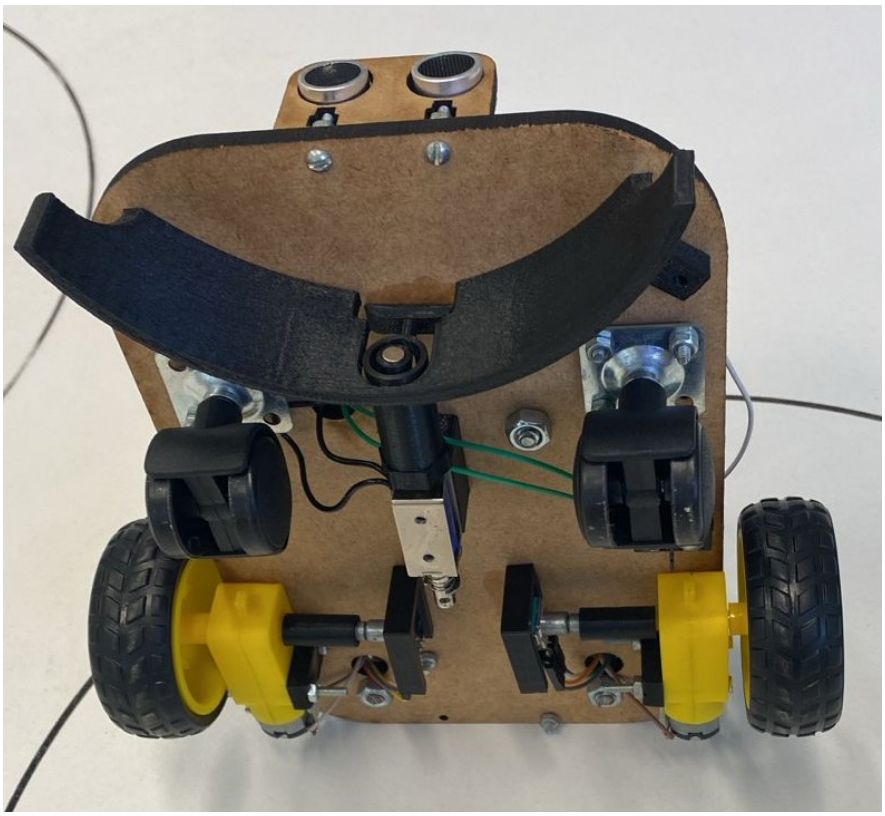

Figura 3 – Vista inferior do robô

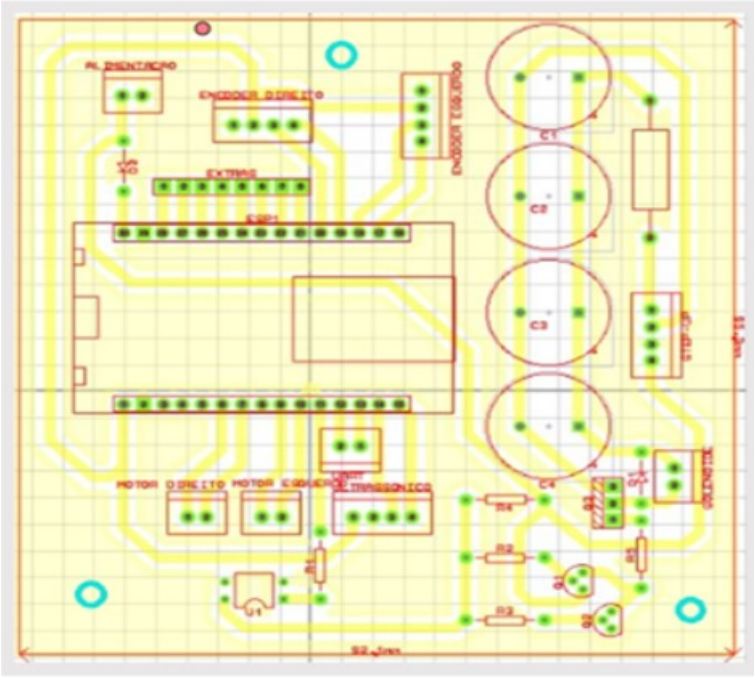

Figura 4 – PCI Lado dos Componentes

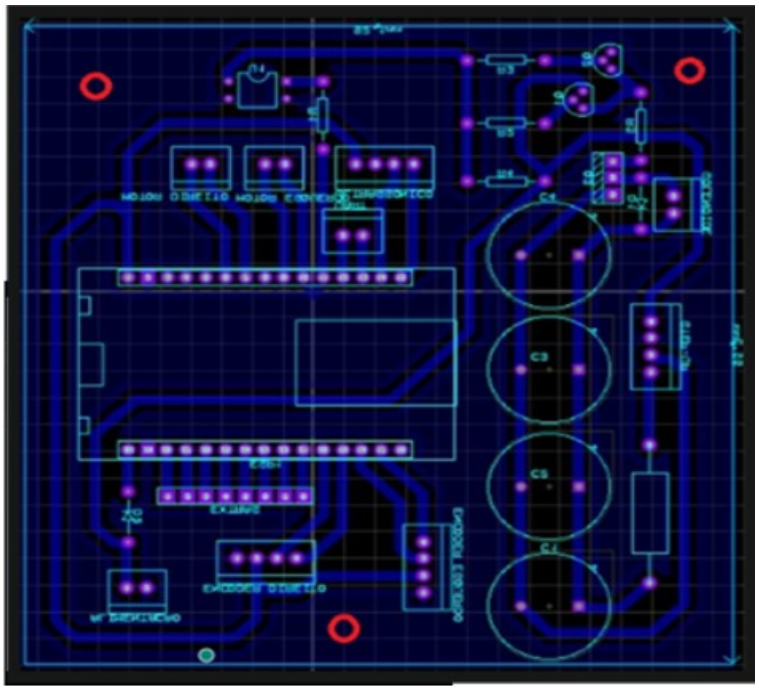

Figura 5 – PCI Lado da Solda

Produto	Qtd.	Valor	Loja
ESP32 WROOM DevKit V1	1	R$ 44,99	Shopee
Rodízio giratório de plástico 30mm	2	R$ 4,90 (unid.)	Eletrogate
Driver ponte H L298N	1	R$ 18,00	Shopee
Motor 3,6V com redução e roda	2	R$ 14,90 (unid.)	Eletrogate
Cabos jumper macho/fêmea 20cm	20	R$ 12,00	Shopee
Cabos jumper fêmea/fêmea 10cm	20	R$ 12,00	Shopee
Chave Gangorra KCD 1-101 (2 terminais)	2	R$ 1,50 (unid.)	Eletrogate
Sensor Encoder Rotativo KY-040	2	R$ 9,34 (unid.)	Eletrogate
Capacitor Eletrolítico 2200uF/35V	4	R$ 21,90 (kit c/10)	Shopee
Módulo Step-Up XL6009	1	R$ 14,50	Shopee
Mini Solenoide JF-0530B 12V	1	R$ 40,00	Shopee
Transistor IRF530N MOSFET	1	R$ 12,90	Shopee
Resistor 1kΩ	3	R$ 0,90 (10 unid.)	Eletrogate
Resistor 150Ω	1	<R$ 1,00	Eletrodil/Constel
Transistor BC558/557	1	R$ 0,34	Eletrogate
Transistor BC547	1	R$ 0,25	Eletrogate
Optoacoplador PC817	1	R$ 12,90	Shopee
Bateria lítio 3,7V – 6800mAh	2	R$ 19,80 (par)	Shopee
Módulo TP4056 (carregador)	1	R$ 5,90	Eletrogate
Imãs Neodímio 5×1mm	3	Fornecido pelo PET	–
Suporte 2 baterias 18650	1	R$ 5,90	Eletrogate
Diodo 1N4007	1	R$ 1,20 (10 unid.)	Eletrogate
Resistor 200Ω 5W	1	Aprox. R$ 5,00	Eletrodil/Constel
Suporte 1 bateria 18650	1	R$ 8,99	Shopee
Sensor Ultrassom HC-SR04	1	R$ 10,40	Eletrogate
Barra de Pinos Macho 1x40 - 180º	1	R$0,90	Eletrogate
Barra de Pinos Fêmea 1x40 - 180º	1	R$14,49 (10 unid.)	Eletrogate
Borne KRE 2 vias mini	1	R$0,95	Eletrogate

Tabela 1 – Lista de componentes utilizados no projeto com respectivas quantidades, valores e fornecedores.

Componente	**Valor Estimado**
Peças em MDF	Em torno de R$15,00
Peças impressas em 3D	Em torno de R$15,00
Placa de circuito impresso	R$5,00
Parafusos: M3 15mm (10 unid.), M3 30mm (4 unid.), M3 40mm (4 unid.); Porcas: M3 (20 unid.), 3/16 (12 unid.); Barra roscada 3/16 (30 cm)	Em torno de R$10,00

Tabela 2 – Componentes mecânicos e valores estimados

Observação: os respectivos componentes foram cotados no mês de Maio de 2025, é possível que os valores atuais sejam diferentes.

- Lojas Shopee: Baterias: Afanx.
- Solenoides: ARDUSHOP Componentes Eletrônicos.
- Optoacoplador, mosfet, capacitores: Saravati.
- ESP32, Case 1 bateria, pinos macho e fêmea p/ PCB: ArduKits.
- Step-Up, jumpers, ponte h: Telos e-comerce.
- Valor total do Robô: R338,47.

Tipo de controlador	K_c	T_i	T_d
P	$\frac{\tau}{K\theta}$	—	—
PI	$\frac{0{,}9\tau}{K\theta}$	4θ	—
PID	$\frac{1{,}2\tau}{K\theta}$	$2{,}375\theta$	$0{,}421\theta$

Tabela 3 – Parâmetros de sintonia do método CHR

3.1 ESP32 WROOM

O microcontrolador utilizado no robô foi o ESP32, uma vez que seu poder de processamento é 10x superior ao do Arduino e também oferece conexão Wi-Fi e Bluetooth, indispensáveis para a comunicação efetiva com outros controladores.

Figura 6 – ESP32 WROOM

Para utilizar o ESP32, é necessário baixar no computador o Arduino IDE, assim como sua biblioteca. O passo a passo está disponível em: https://blog.eletrogate.com/conhecendo-o-esp32-usando-arduino-ide-2/

3.2 Motores

Os motores DC são responsáveis pela movimentação do robô, ligados à ponte H, na qual controla a velocidade e o sentido de rotação dos mesmos.

Figura 7 – Motores DC e rodas

Antes de se iniciar as ligações elétricas é necessário fixar na parte traseira do robô a ponte H e os motores com os parafusos e porcas, assim como os dois rodízios giratórios, com uma porca em cada parafuso, entre o mdf e a base do rodízio, para o alcançar seu nivelamento.

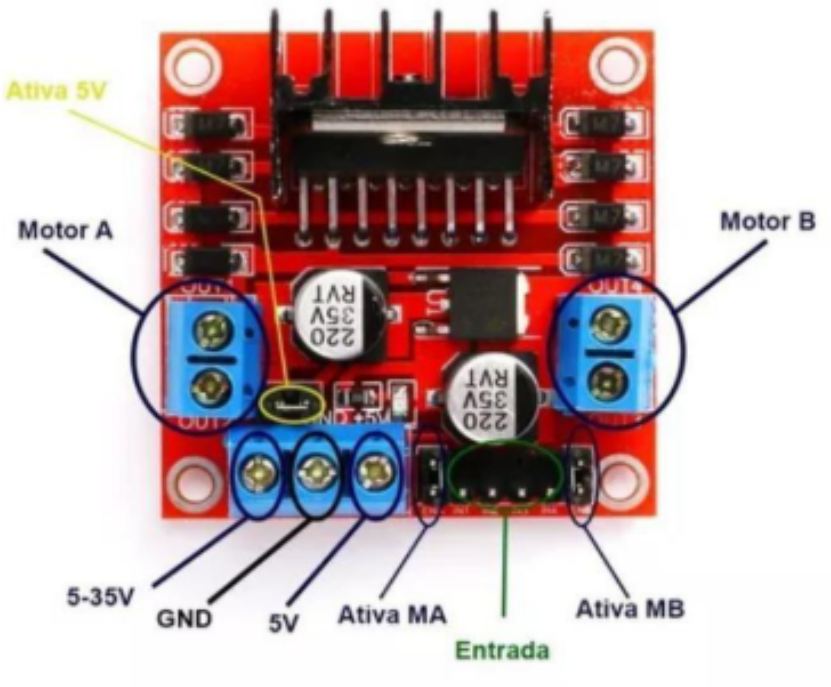

Figura 8 – Ponte H.

É possível notar que a ponte H possui 13 pinos para conexões. O motor direito será ligado nas entradas do "Motor A" e o motor esquerdo nas entradas do "Motor B". O pino "Ativa MA" é o enable do "Motor A" (motor direito), já o pino "Ativa MB" é o enable do "Motor B" (motor esquerdo), esses pinos são responsáveis pelo controle de velocidade dos motores.

Observação: nos pinos "Ativa 5V", "Ativa MA" e "Ativa MB" estão presentes um jumper, esses deverão estar conectados.

A alimentação do robô será feita por duas baterias de lítio ligadas em série para entregar uma tensão de aproximadamente 8 volts. O polo positivo desse conjunto deve ser ligado à chave gangorra e logo após no pino 5-35V, já o polo negativo será ligado no GND em comum com uma porta GND do ESP32 (Figura 9).

Observação: O robô deverá estar DESLIGADO para conectar o ESP32 no computador.

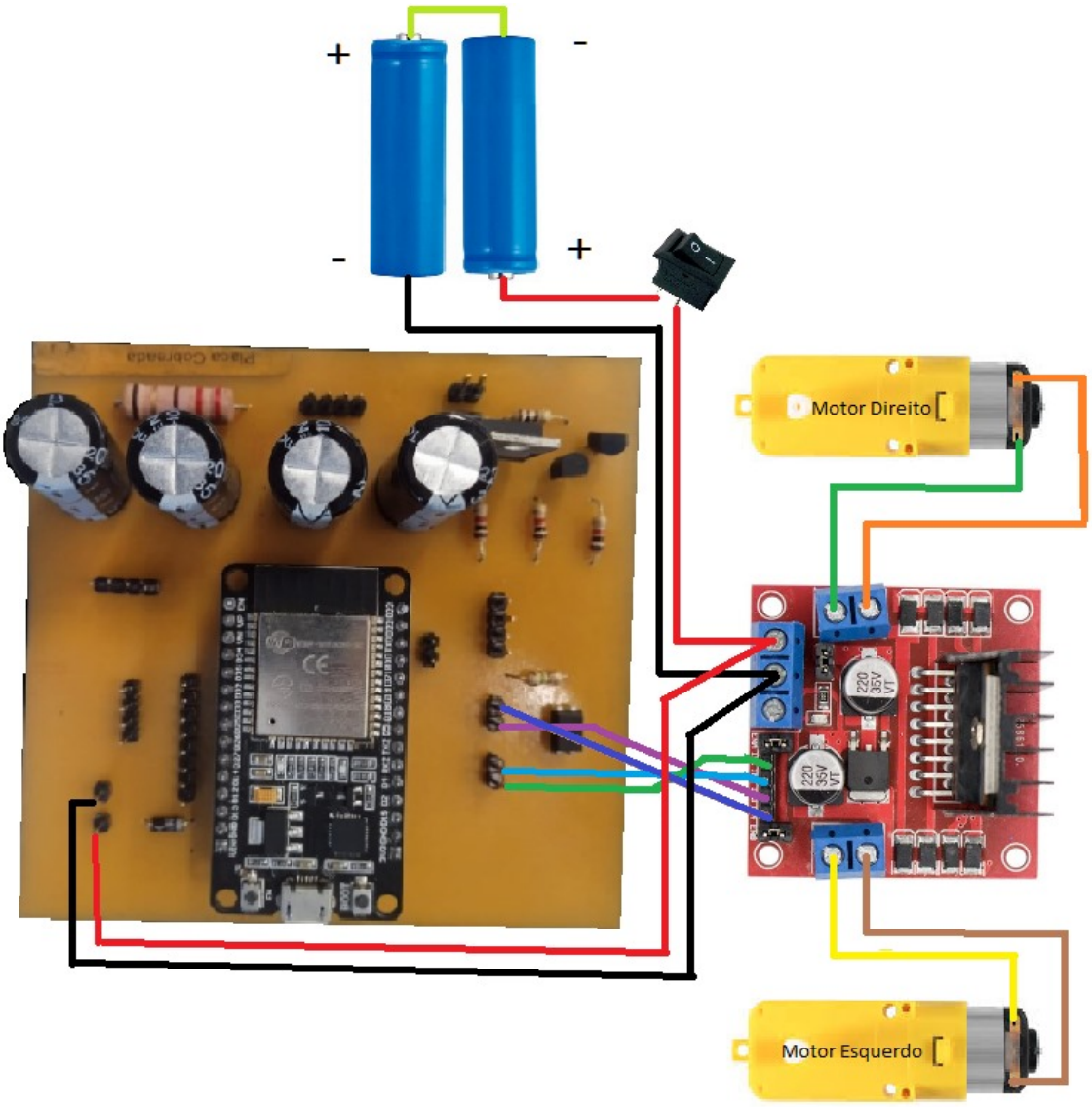

Figura 9 – Circuito completo dos motores.

3.3 Sensor Ultrassônico

O sensor ultrassônico será os “olhos” do robô, responsável por detectar obstáculos no campo de visão frontal. Denominado “Sensor 1”, esse recurso permite calcular a distância até os objetos, possibilitando que o robô identifique obstáculos, evite colisões e oriente seu deslocamento de forma eficiente.

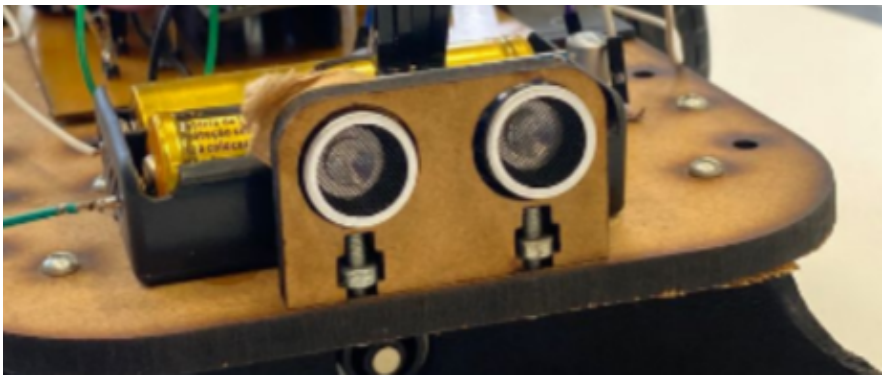

Figura 10 – Sensor descrito

Figura 11 – Sensor ultrassom HC-SR04

É possível notar que os sensores HC-SR04 possuem 4 pinos, ”Vcc” e ”Gnd” para a alimentação, ”Trig” e ”Echo” para a transmissão do sinal. A PCB a seguir foi utilizada para exemplificar e facilitar as ligações.

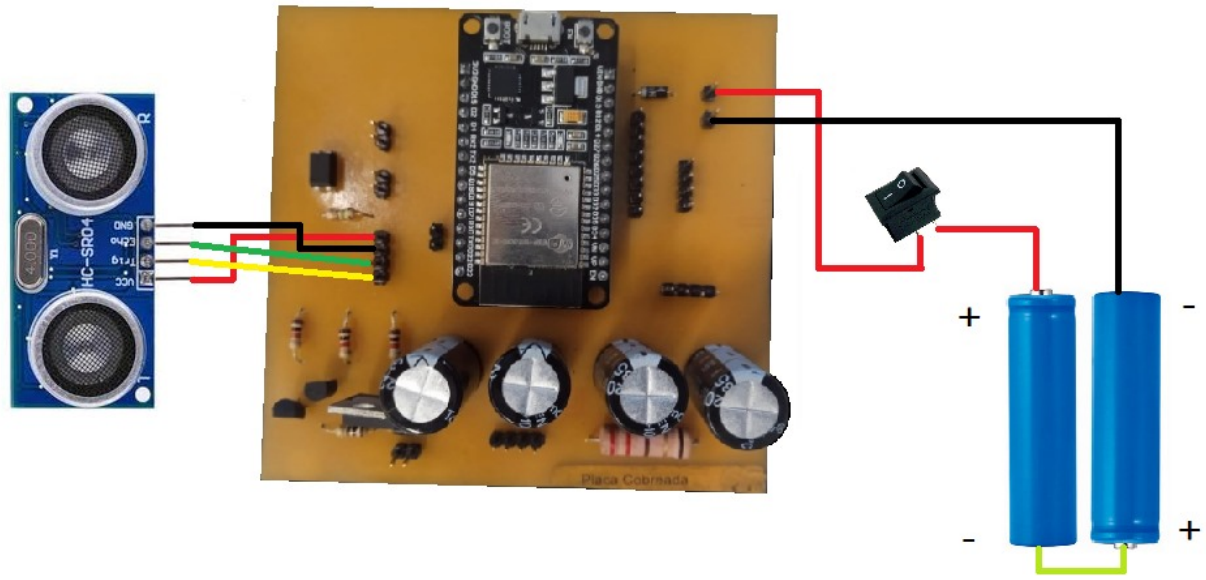

Figura 12 – Circuito completo do sensor

3.4 Encoders

Para monitorar a movimentação do robô são utilizados 2 encoders, sendo cada um acoplado ao eixo da respectiva roda atuada do robô. Assim, qualquer movimento do arranjo motor mais caixa de redução será indicado para o ESP32. Vale lembrar que a roda é encaixada num dos lados do eixo de saída da caixa de redução, sendo o outro lado acoplado ao eixo do encoder por meio de uma peça de ABS impressa. Já a placa do encoder é fixada à base do robô através de outra peça impressa. Conforme é possível notar na Figura 13, o encoder possui 5 pinos, sendo "+" e "Gnd" para a alimentação, "Clk" e "Dt" para a transmissão do sinal, já o pino "Sw" não será utilizado nessa montagem.

Figura 13 – Encoder rotativo KY-040

As respectivas ligações estão exemplificadas na Figura 14.

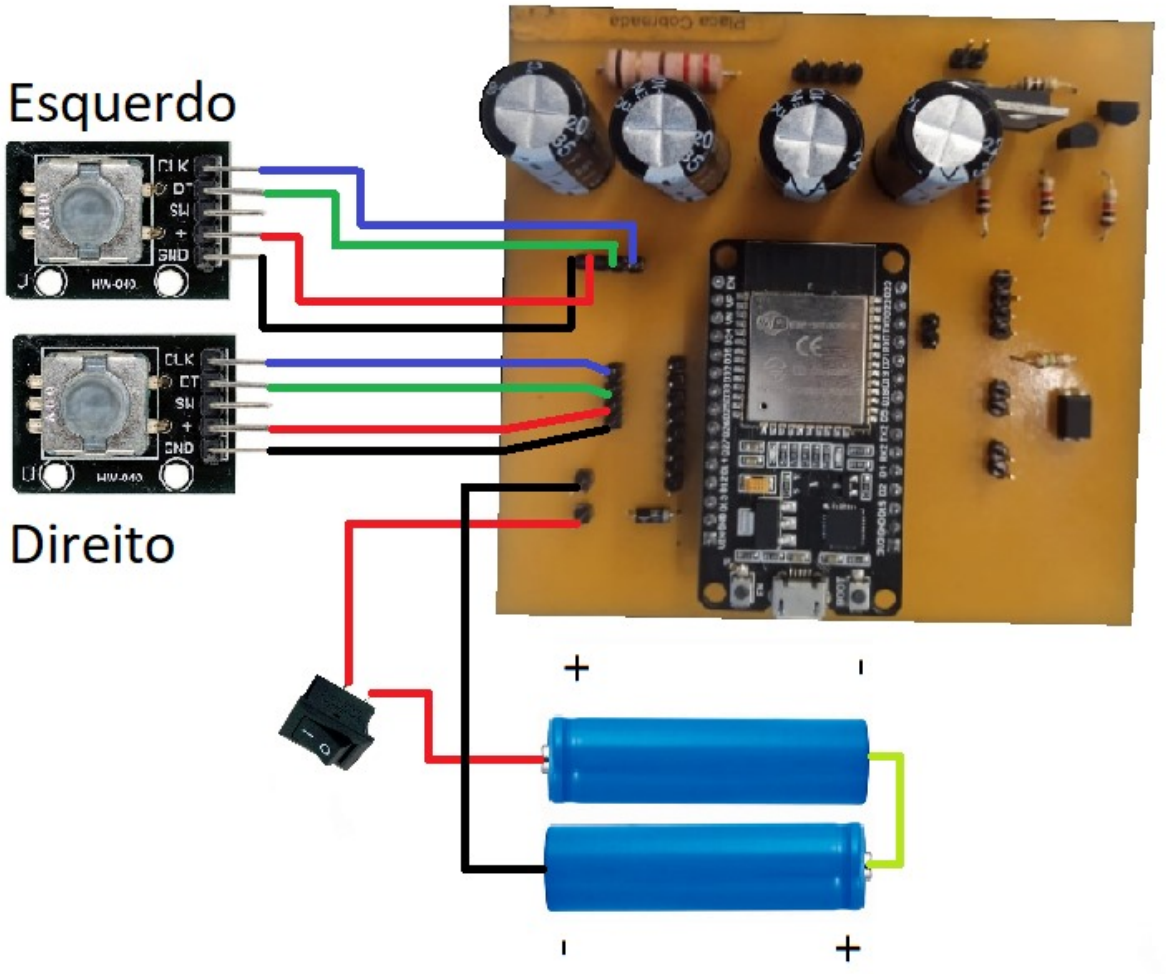

Figura 14 – Circuito completo dos encoders rotativos KY-040.

3.5 Chutador

Por fim, mas não menos importante, chega-se ao dispositivo chutador, sendo um arranjo fundamental para capacitar o robô móvel para jogar futebol. Tal arranjo pode ser dividido em sua parte mecânica e eletroeletrônica, descritas a seguir.

3.5.1 Parte mecânica do chutador

A parte mecânica do chutador é composta por um funil responsável por guiar a bola para o centro, no qual se encontra o arranjo responsável por empurrar a bola. Neste arranjo se encontra uma parte fixa com até 4 imãs de neodímio 5mm x 1mm acoplados. Este arranjo é responsável por evitar com que a bola escape do robô ao serem realizadas mudanças de

direção (a bola possui uma tinta com pó de ferro para criar a atração). Em volta dos imãs, há um extensor cilíndrico acoplado ao solenoide, que se move para frente junto ao mesmo, realizando assim o chute da bola. Todo arranjo pode ser visto na Figura 3, vista inferior do robô com alguns componentes, e somente as peças impressas, como mostrado na Figura 15.

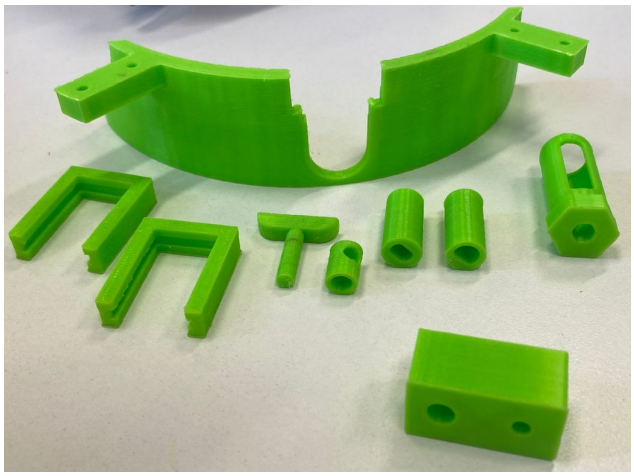

Figura 15 – Peças mecânicas do chutador impressas

3.5.2 Parte eletrônica do chutador

A parte eletroeletrônica do chutador do robô será composta pela bobina do solenoide de 12V e por um circuito responsável pelo seu acionamento, mostrado na Figura 17 e na Figura 16. Nele é empregado um banco de capacitores, composto por quatro capacitores eletrolíticos de 2200 uF / 35V (podendo ser usados capacitores de 50V, sem nenhum prejuízo para o funcionamento do circuito). Esse banco de capacitores é carregado através de um elevador de tensão Step-up de forma lenta, sendo a corrente de carga limitada pelo resistor de 200 Ôhms / 5 W. Seu tempo de carregamento gira em torno de 7 segundos. Já a descarga do banco de capacitores no solenoide ocorre por meio de um sinal ”HIGH” enviado pelo pino “D15” do ESP32 ao MosFet IRF530N, após uma pré-amplificação do sinal pelo transistor BC547 e o BC558.

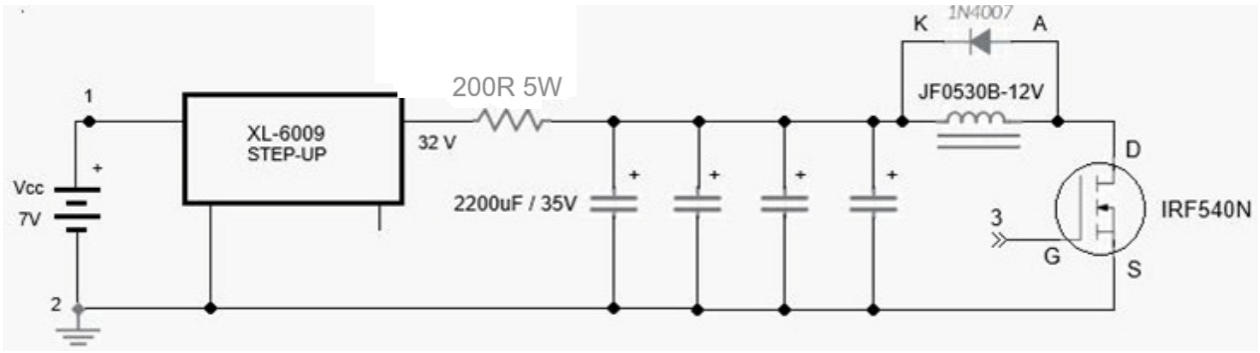

Figura 16 – Circuito de potência do chutador

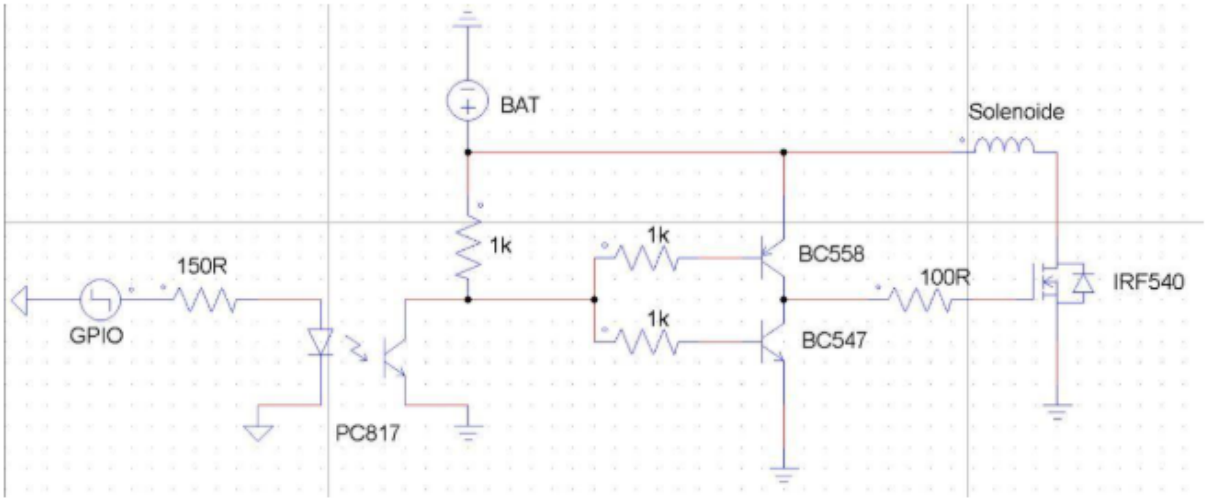

Figura 17 – Circuito de acionamento da bobina do solenóido do chutador

Figura 18 – Módulo elevador de tensão Step-up XL-6009.

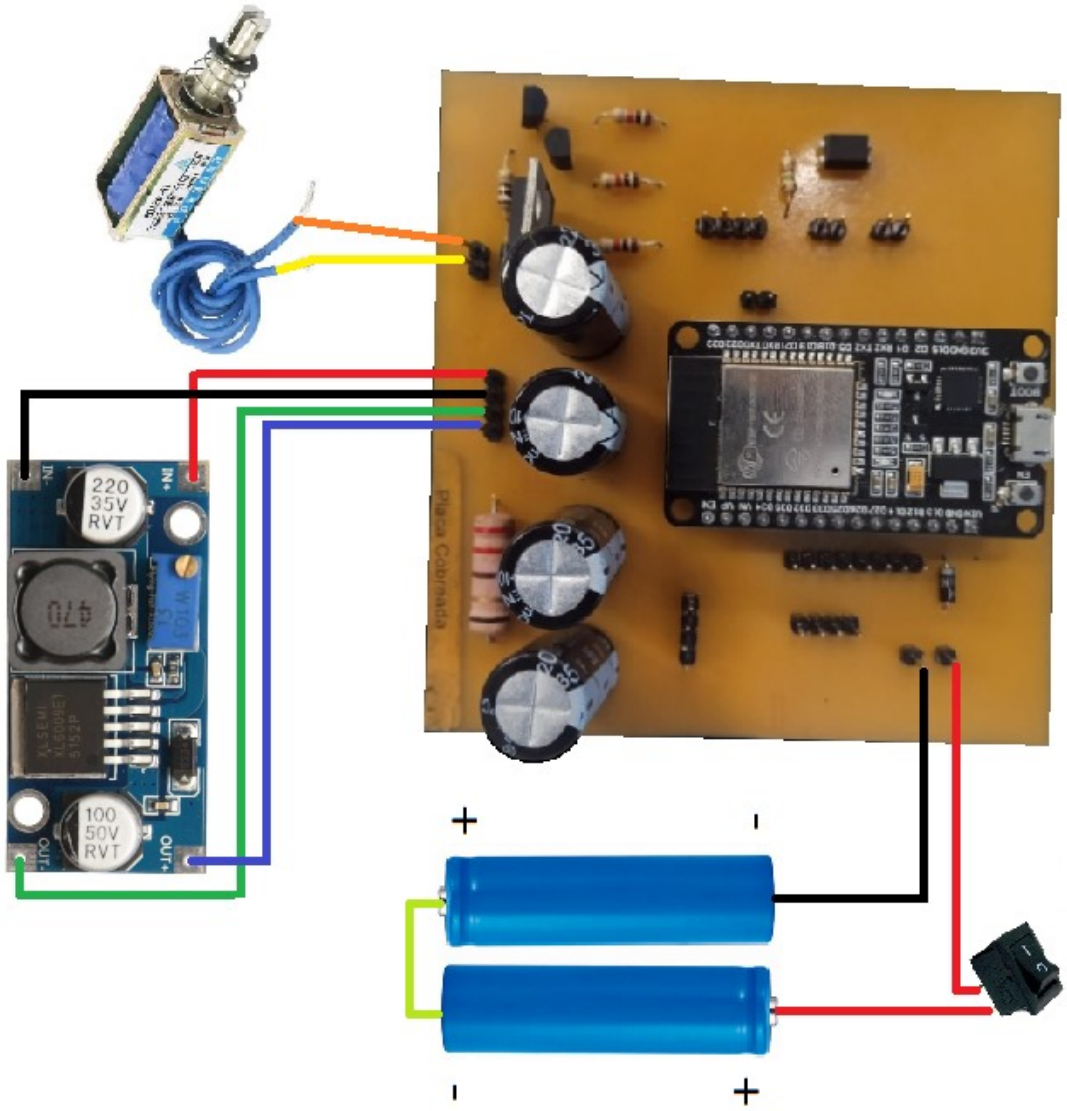

Figura 19 – Circuito completo do chutador

O funcionamento o circuito de acionamento do chutador do robô se baseia na ideia de armazenar uma quantidade significativa de energia elétrica no banco de capacitores, através da elevação da tensão da bateria e do carregamento do banco de capacitores por meio de uma corrente não muito alta (carregamento lento). No momento do chute, o ESP32 comanda o MosFet como uma chave, fechando-a e permitindo que a energia nos capacitores descarregue rapidamente, no solenoide, produzindo sua atuação que movimenta o tubo destinado a bater na bola, como descrito na seção anterior. A duração dessa descarga é inferior a um segundo.

É importante destacar que a força do chute pode ser modulada pela duração do pulso aplicado pelo ESP32 ao circuito do chutador. Assim, um pulso de 0,1 segundo irá produzir um chute forte, próximo à capacidade máxima do chutador, devendo ser usado para realizar

um chute de longa distância, como próximo ao meio campo. Já um chute a curta distância ou um passe para o outro robô próximo, deve ser feito com um pulso de intervalo menor, digamos de 0,01 segundo. Forças intermediárias de chute podem ser previstas usando-se valores intermediários aos citados acima.

Outro fato que é preciso destacar, é que os intervalos de tempo dos pulsos e consequentes forças obtidas nos chutes devem ser ajustadas experimentalmente, ou seja, numa sequência de tentativas e erros.

Os vídeos que auxiliam a montagem do robô estão disponíveis em: **CLIQUE AQUI!**

3.5.3 EXEMPLO DE PROGRAMAÇÃO DO CHUTADOR DO ROBÔ

No código abaixo, foram declaradas todas as portas e variáveis de cada componente, assim como suas explicações, no entanto, o "void loop" apenas move o robô para frente, realiza um giro e chuta a bola. Use-o como base para testar funções um pouco mais complexas, e tentar mais movimentos, assim como comutar o acionamento dos motores (sentido e velocidade) de acordo com a distância do robô em relação a algum objeto, divirta-se!

```
#include <Ultrasonic.h>

/*Define os sensores ultrassom.*/
Ultrasonic ultrasonic1(18, 5); //Sequência (trig, echo).
Ultrasonic ultrasonic2(21, 19);
Ultrasonic ultrasonic3(23, 22);

/*Define as portas da ponte H.*/
#define enable_direito 13
#define motor_direito_frente 14
#define motor_direito_reverso 12
#define enable_esquerdo 25
#define motor_esquerdo_frente 27
#define motor_esquerdo_reverso 26

/*Define as portas dos encoders.*/
#define CLK_D 32
#define DT_D 33
#define CLK_E 34
#define DT_E 35
/*Inicia uma contagem em zero, e cria variaveis para guardar as posições.*/
int count_D = 0;
int count_E = 0;
int currentStateCLK_D;
int lastStateCLK_D;
int currentStateCLK_E;
int lastStateCLK_E;
String currentDIR_D = "";
String currentDIR_E = "";

/*Define a porta do chutador, seu estado inicial como ligado, e o
tempo na qual ela ficará desabilitada (5 segundos).*/
const int chutador = 15;
bool chutar = true;
unsigned long tempo_porta_off = 5000;
/*ATENÇÃO: a porta 15 sempre manda um sinal "true" para o transistor, um sinal "false"
será usado apenas na hora do chute por um curto perído de tempo, caso contrário, haverá
sobrecarga nos capacitores.*/
```

Figura 20 – Variáveis Declaradas

```
void setup()
{
  /*Inicia o monitor serial, define as portas da ponte H e chutador como saídas,
  dos encoders como entrada, além de setar algumas condições iniciais como a
  velocidade do motor.*/
  Serial.begin(9600);
  pinMode(chutador,OUTPUT);
  digitalWrite(chutador, chutar);
  pinMode(enable_direito, OUTPUT);
  pinMode(motor_direito_frente, OUTPUT);
  pinMode(motor_direito_reverso, OUTPUT);
  pinMode(enable_esquerdo, OUTPUT);
  pinMode(motor_esquerdo_frente, OUTPUT);
  pinMode(motor_esquerdo_reverso, OUTPUT);
  analogWrite(enable_direito, 200);
  analogWrite(enable_esquerdo, 200);
  delay(100);

  /*Imprime as informações do encoder no monitor.*/
  Serial.println("Pronto!");
  Serial.println("Posição Encoder Direito: 0");
  Serial.println("Posição Encoder Esquerdo: 0");
  pinMode(CLK_D, INPUT);
  pinMode(DT_D, INPUT);
  pinMode(CLK_E, INPUT);
  pinMode(DT_E, INPUT);
  lastStateCLK_D = digitalRead(CLK_D);
  lastStateCLK_E = digitalRead(CLK_E);

  /*Lê as medidas do sensor em centimetros.*/
  ultrasonic1.read(CM);
  ultrasonic2.read(CM);
  ultrasonic3.read(CM);
}
```

Figura 21 – Configurações das portas

```
/*Função para o robô andar para frente.*/
void robo_frente()
{
  digitalWrite(motor_direito_frente, HIGH);
  digitalWrite(motor_direito_reverso, LOW);
  digitalWrite(motor_esquerdo_frente, HIGH);
  digitalWrite(motor_esquerdo_reverso, LOW);
}

/*Função para o robô andar para trás.*/
void robo_reverso()
{
  digitalWrite(motor_direito_frente, LOW);
  digitalWrite(motor_direito_reverso, HIGH);
  analogWrite(enable_direito, 150);
  digitalWrite(motor_esquerdo_frente, LOW);
  digitalWrite(motor_esquerdo_reverso, HIGH);
  analogWrite(enable_esquerdo, 150);
}

/*Função para o robô virar para a esquerda.*/
void robo_esquerda()
{
  digitalWrite(motor_direito_frente, HIGH);
  digitalWrite(motor_direito_reverso, LOW);
  analogWrite(enable_direito, 150);
  digitalWrite(motor_esquerdo_frente, LOW);
  digitalWrite(motor_esquerdo_reverso, HIGH);
  analogWrite(enable_esquerdo, 150);
  delay(1000);
}
```

Figura 22 – Funções para mover o robô

```
/*Função para o robô virar para a direita.*/
void robo_direita()
{
  digitalWrite(motor_direito_frente,LOW);
  digitalWrite(motor_direito_reverso,HIGH);
  analogWrite(enable_direito, 150);
  digitalWrite(motor_esquerdo_frente,HIGH);
  digitalWrite(motor_esquerdo_reverso,LOW);
  analogWrite(enable_esquerdo, 150);
  delay(1000);
}

/*Função para o robô ficar parado.*/
void robo_parado()
{
  digitalWrite(motor_direito_frente,LOW);
  digitalWrite(motor_direito_reverso,LOW);
  digitalWrite(motor_esquerdo_frente,LOW);
  digitalWrite(motor_esquerdo_reverso,LOW);
}

/*Função para o robô chutar.*/
void robo_chutar()
{
  digitalWrite(chutador, false);
  delay(tempo_porta_off);
  digitalWrite(chutador, true);
}
```

Figura 23 – Funções para mover o robô

```
void loop()
{
  robo_frente();
  delay(1000);
  robo_parado();
  delay(100);
  robo_esquerda();
  delay(500);
   robo_parado();
  delay(100);
  robo_chutar();
}
/*Nesse loop, o robô anda para frente, realiza um giro e chuta a bola.
Fique a vontade para usar sua criatividade!*/
```

Figura 24 – Loop criado para movimentar o robô

4 COMUNICAÇÃO INTELIGENTE NO FUTEBOL DE ROBÔS COM MQTT

No futebol de robôs, a comunicação eficiente é essencial para o desempenho das equipes. Por meio do MQTT, que é um protocolo de mensagens leve e eficiente, projetado para dispositivos com largura de banda limitada e conexões instáveis, o sistema de visão computacional pode enviar coordenadas e orientações em tempo real para os controladores dos robôs. Essa troca de informações permite que os robôs ajam de forma coordenada e estratégica durante o jogo, executando as táticas planejadas pela equipe de maneira precisa e sincronizada.

Para isso, o envio das coordenadas do computador ao micro controladores do robô foi atribuída a biblioteca paho.mqtt.client, que é usada para implementar a comunicação MQTT (Message Queuing Telemetry Transport) conforme a Figura 25 abaixo.

```
1  import paho.mqtt.client as mqtt
2  import time
3  import json
```

Figura 25 – Biblioteca MQTT

Em seguida, um cliente MQTT como na Figura 26 é criado com client = mqtt.Client ("PythonPublisher"). Aqui, (PythonPublisher) é o nome dado a esta instância do cliente, que será responsável por publicar as mensagens no broker MQTT. A conexão ao broker é estabelecida com client.connect(mqttBroker, 1883), onde mqttBroker é o endereço do broker (neste caso, test.mosquitto.org) e 1883 é a porta padrão para comunicação MQTT. Esta conexão permite que o cliente envie e receba mensagens do broker.

```
5  mqttBroker = "test.mosquitto.org"
6  client = mqtt.Client("PythonPublisher")
7  client.connect(mqttBroker, 1883)
```

Figura 26 – Conexão ao broker MQTT

A publicação das mensagens é realizada com o método client.publish (topic, message) apresentado na Figura 27. No código fornecido, os tópicos são formatados dinamicamente

como “topico:robo” para especificar a coordenada e o robô correspondente, enquanto as mensagens são os valores das coordenadas convertidos para strings. Por exemplo, uma mensagem poderia ser publicada no tópico cood(x):robo1 com o valor 10.0 referente a coordenada x do robô 1. Este método envia a mensagem para o broker MQTT, que pode então encaminhá-la para quaisquer clientes inscritos neste tópico.

```
38      # Publicar as coordenadas para cada robô e bola
39      for robo, info in coordenadas.items():
40          for topico, valor in info.items():
41              client.publish(f"{topico}:{robo}", str(valor))
42
43      time.sleep(1)
44
```

Figura 27 – :Publicação das coordenadas via MQTT.

Já para a coleta dos dados das coordenadas e identificação foi atribuída uma estratégia onde inicialmente foi preciso estabelecer o canal do broker a ser utilizado e definir as permissões e identificações para determinar quem teria acesso às informações recebidas como nas Figuras 28 e 29.

```
void Inicializacaocood(const char *ssid, const char *password, const char *mqttBroker, const char *nome) {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando ao WiFi...");
  }

  Serial.println("Conectado ao WiFi");

  client.setServer(mqttBroker, 1883); // Porta padrão do MQTT
  client.setCallback(callback);

  while (!client.connected()) {
    Serial.println("Conectando ao broker MQTT...");

    if (client.connect(nome)) { // Escolha um nome único para o cliente
      Serial.println("Conectado ao broker MQTT");

      // Inscreva-se em todos os tópicos
      for (int i = 0; i < 14; i++) {
        client.subscribe(mqttTopics[i]);
      }

    } else {
      Serial.print("Falha na conexão ao broker MQTT. Tentando novamente em 5 segundos...");
      delay(5000);
```

Figura 28 – Estabelecimento do canal do broker e conexão com o mesmo

```
4   const char *mqttTopics[] = {
5     "cood(y):robo1", "cood(x):robo1", "rotacao:robo1",
6     "cood(y):robo2", "cood(x):robo2", "rotacao:robo2",
7     "cood(y):robo3", "cood(x):robo3", "rotacao:robo3",
8     "cood(y):robo4", "cood(x):robo4", "rotacao:robo4",
9     "cood(x):bola", "cood(y):bola"
10  };
```

Figura 29 – Estabelecimento do canal do broker e conexão com o mesmo

Finalizando o envio da informação do ESP32 central e encerrando seu código. Assim, para simplificar ainda mais o código e facilitar a implementação da comunicação a partir dos dados, foi desenvolvida uma biblioteca abrangente contendo todas as funções necessárias para o broker MQTT.

Essa abordagem modular não apenas reduz a complexidade do código, tornando-o mais legível, mas também agiliza significativamente o processo de inicialização do broker. Ao encapsular as funcionalidades do broker MQTT em uma biblioteca, proporcionamos uma camada de abstração que permite aos desenvolvedores acessar facilmente as funcionalidades necessárias sem a necessidade de lidar diretamente com detalhes de implementação complexos. Isso não apenas acelera o desenvolvimento, mas também promove a reutilização de código, tornando-o mais eficiente e fácil de manter ao longo do tempo.

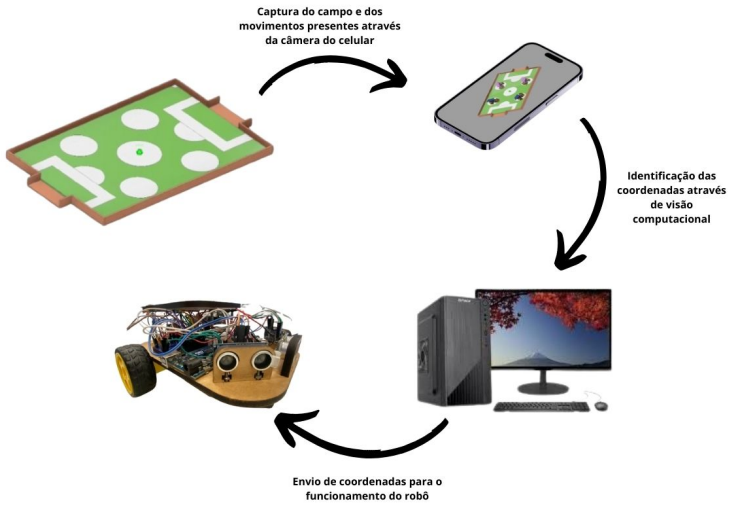

Figura 30 – Mapa mental sobre o funcionamento da comunicação

Falando sobre o sistema de visão computacional, ele funciona a partir de um código em Python baseado na biblioteca ”OpenCV”, que fica em execução durante todo o momento da partida. Com o auxílio de uma câmera localizada na parte superior do ”campo”, a imagem obtida por ela é trabalhada e processada a partir de cores, localizadas estrategicamente na parte superior dos robôs, orientando seus movimentos e ações. As informações obtidas pelo algoritmo Python são enviadas quase que instantaneamente para o ESP32, tornando o jogo dinâmico e sem interrupções.

```python
# Detecção de círculo rosa ou amarelo dentro do objeto azul
roi = frame[max(0, cY - 15):min(frame.shape[0], cY + 15),
            max(0, cX - 15):min(frame.shape[1], cX + 15)]

if roi.size == 0:
    continue

hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)

# Mudar o Upper e o Lower da cor do círculo (cor secundária 1)
lower_secondary1 = np.array([140, 50, 50])
upper_secondary1 = np.array([170, 255, 255])
secondary1_mask = cv2.inRange(hsv_roi, lower_secondary1, upper_secondary1)
secondary1_contours, _ = cv2.findContours(secondary1_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Mudar o Upper e o Lower da cor do círculo (cor secundária 2)
lower_secondary2 = np.array([20, 100, 100])
upper_secondary2 = np.array([30, 255, 255])
secondary2_mask = cv2.inRange(hsv_roi, lower_secondary2, upper_secondary2)
secondary2_contours, _ = cv2.findContours(secondary2_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Figura 31 – Definição dos limites de cores do landmark

```python
for secondary1_contour in secondary1_contours:
    secondary1_moment = cv2.moments(secondary1_contour)
    if secondary1_moment["m00"] != 0:
        secondary1_cX = int(secondary1_moment["m10"] / secondary1_moment["m00"])
        secondary1_cY = int(secondary1_moment["m01"] / secondary1_moment["m00"])

        secondary1_cX += max(0, cX - 15)
        secondary1_cY += max(0, cY - 15)

        # Objeto azul com círculo rosa
        cv2.drawContours(frame, [contour], -1, (255, 0, 0), 2)
        cv2.drawContours(frame, [secondary1_contour], -1, (0, 0, 255), 2)
        print(f"Objeto azul com círculo rosa: Objeto: ({cX}, {cY}), Círculo rosa: ({secondary1_cX}, {secondary1_cY})")
```

Figura 32 – Fragmento do código de identificação dos círculos do robô

Foi decidido que cada time seria representado por uma cor sólida no fundo do landmark, para indicação mais clara do time, além de um círculo orientando a frente do robô,

dividido entre duas cores, uma representando o robô "capitão" e a outra identificando o "secundário". Foi decidido também que o campo será feito de madeira MDF, com uma câmera suspensa em cima captando e orientando a movimentação dos robôs. A fim de testes, um protótipo de campo foi montado na mesa principal do laboratório com uma câmera de um smartphone presa a um tripé que, por sua vez, está preso ao teto.

Figura 33 – Protótipo do campo em MDF construído no AUTOCAD como ocorrerá a competição

Com a utilização do MQTT, a comunicação entre o sistema de visão computacional e os robôs se torna rápida, confiável e eficiente, contribuindo para a dinâmica e competitividade do futebol de robôs.

Figura 34 – Construção do campo no laboratório.

```
#data = {"coord(y):retanguloR1": cY }  # Dados para a coordenada "y" do retangulo do robo 1
#response = requests.post(esp32_central_url1, json=data)

#data = {"coord(x):retanguloR1": cX}  # Dados para a coordenada "x" do retangulo do robo 1
#response = requests.post(esp32_central_url2, json=data)

#data = {"coord(y):circuloR1": secondary1_cY}  # Dados para a coordenada "x" do retangulo do robo 1
#response = requests.post(esp32_central_url3, json=data)

#data = {"coord(x):circuloR1": secondary1_cX}  # Dados para a coordenada "x" do retangulo do robo 1
#response = requests.post(esp32_central_url4, json=data)
```

Figura 35 – Comunicação do código com o ESP por MQTT

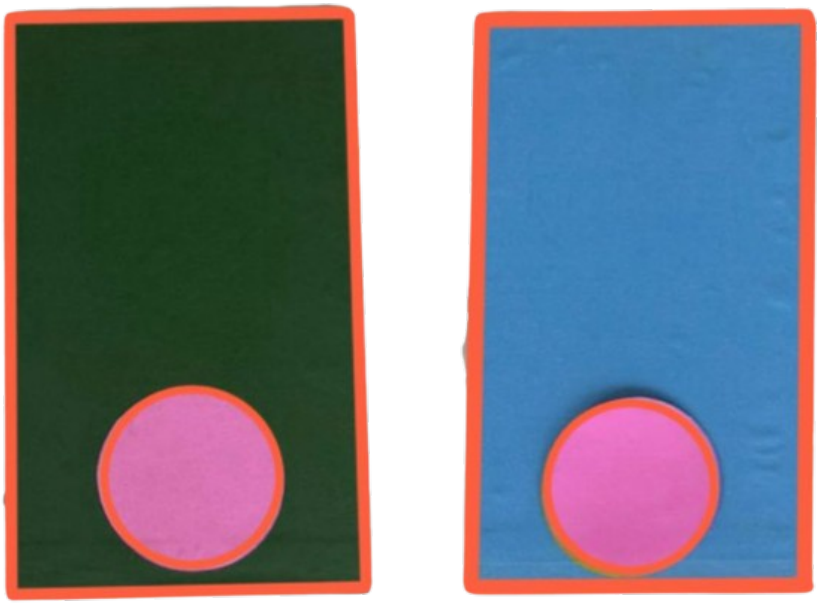

Figura 36 – detecção de cores nas landmakers.

```
Objeto azul com círculo amarelo: Objeto: (338, 115), Círculo amarelo: (337, 102)
Angulo do segmento de reta do robô azul com círculo amarelo: -94.39870535499554
A tangente desse ângulo é: 12.999999999999996
Objeto azul com círculo rosa: Objeto: (328, 304), Círculo rosa: (327, 290)
Angulo do segmento de reta do robô azul com círculo rosa: -94.08561677997488
A tangente desse ângulo é: 13.99999999999998
```

Figura 37 – Reconhecimento de coordenadas

5 Conclusão

Esse guia apresentou as partes componentes do robô móvel a rodas de baixo custo destinadas ao desafio do futebol de robôs promovido pelo grupo PET Eng. Mecatrônica do CEFET-MG / Unidade de Divinópolis em suas iniciativas Projeto Sementes (PS) e Competição Interna de Robótica (CIR). A ideia é portanto, realizar de forma física e presencial o desafio virtual realizado com o simulador realístico Webots na CIR 2022 e CIR 2023, cada equipe sendo composta por dois robôs, também é indicado que exista um robô reserva para imediata substituição em casos de falha técnica ou problemas de funcionamento durante uma partida. Recomenda-se que as equipes façam um bom uso do Webots e das bibliotecas e programas anteriores como uma forma de acelerar o aprendizado e aumentar a disponibilidade do desafio para os participantes da equipe. Ainda mais que ações de movimentação, chute, passe e defesa já se encontram disponíveis nesse formato virtual. Funcionalidades importantes para permitir que os robôs sejam capazes de realizar as diversas ações e jogar uma partida de futebol, como funções de comunicação e de visão computacional estão em fase final de desenvolvimento, encapsulamento em bibliotecas e documentação pelo grupo PET, devendo ser disponibilizada em breve para os usuários e demais interessados. Nota-se que isso reduz a dificuldade para que as equipes efetivamente consigam elaborar seus primeiros programas para fazer com que os robôs consigam jogar futebol, sendo complementado pelo treinamento previsto para a presente iniciativa. É inegável que o desafio de se jogar futebol com robôs não é trivial. No entanto, as oportunidades de aprendizado e de desenvolvimento nas partes componentes dele, tais como montagem mecânica, eletrônica, programação, ESP32 e robótica, é inegável. Esperamos que este guia possa ajudar na principal decisão dessa jornada, ou seja, dar os primeiros passos.

6 Referências

Guia projeto sementes: https://i3dpet.wixsite.com/pet-eng-mecatronica/extensao

Passo a passo ESP32: https://blog.eletrogate.com/conhecendo-o-esp32-usando-arduino-ide-2/

6.1 Glossário

Arduino: Arduino é uma plataforma programável de prototipagem eletrônica de placa única e hardware livre, que permite aos usuários criar objetos eletrônicos interativos e independentes.

Capacitor: Componente que armazena cargas elétricas num campo elétrico, acumulando um desequilíbrio interno de carga elétrica.

Encoder: Dispositivos de feedback usados para determinar a posição, velocidade e direção de um objeto mecânico em movimento.

Motor DC: Um motor DC é um motor elétrico que usa corrente contínua para produzir força mecânica.

Ponte H: Driver usado em motores de corrente contínua, que permite o motor girar tanto no sentido horário quanto no sentido anti-horário.

Solenoide: Dispositivo passivo capaz de armazenar a energia elétrica gerada num campo magnético.

Transistor: Semicondutor usado para amplificar ou trocar sinais eletrônicos e potência elétrica.