

Trabalho de Conclusão de Curso

# DESENVOLVIMENTO DE UM TIME DE FUTEBOL DE ROBÔS PARA A CATEGORIA IEEE VSSS

Luis Felipe Monteiro da Fonte  
ORIENTADOR: TÉO CERQUEIRA REVOREDO



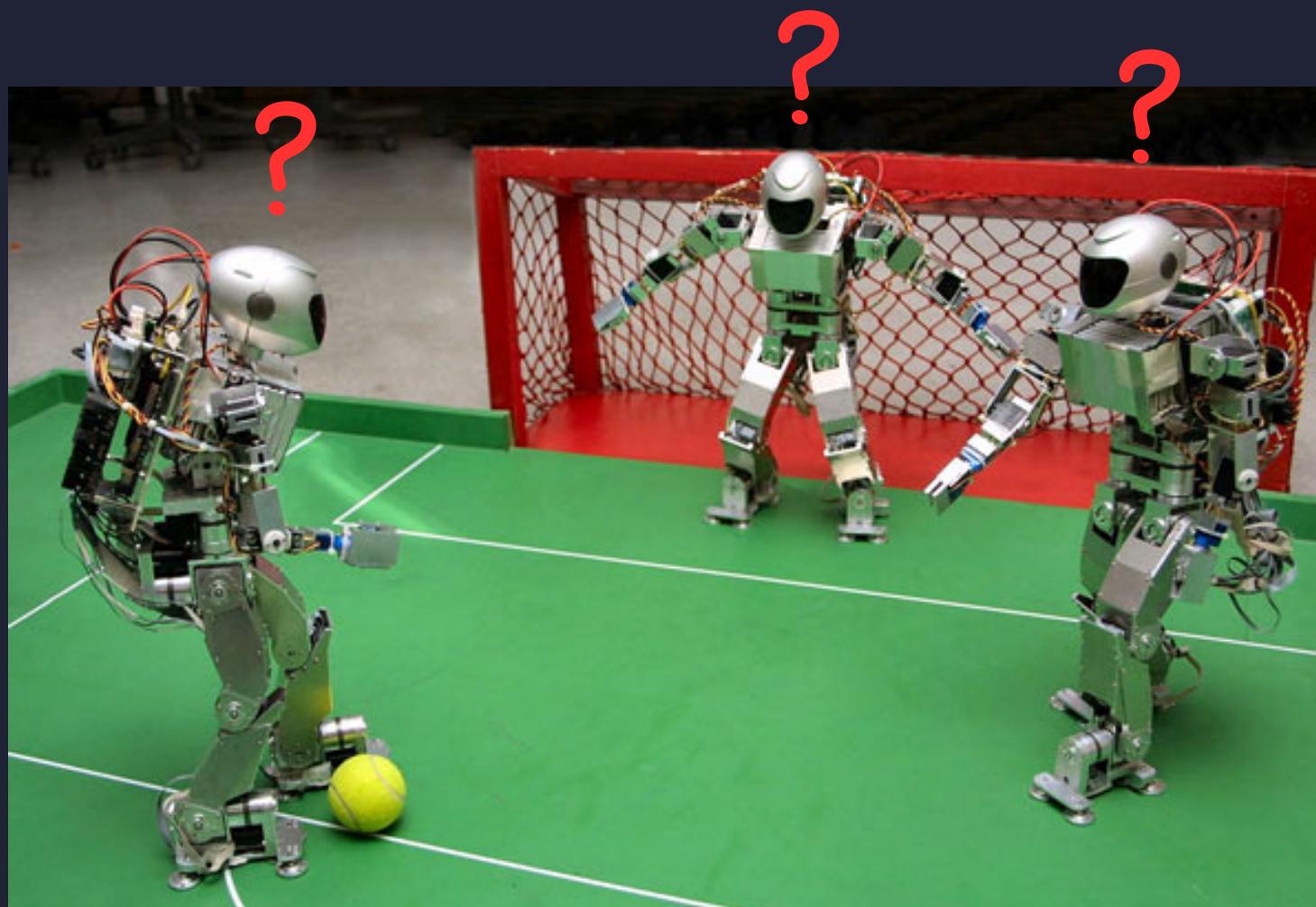
# Sumário

- 1 INTRODUÇÃO
- 2 MONTAGEM DOS ROBÔS E ELEMENTOS DO JOGO
- 3 VISÃO COMPUTACIONAL E INTEGRAÇÃO
- 4 MODELAGEM MATEMÁTICA
- 5 CONTROLADOR DE VELOCIDADE ANGULAR
- 6 MOVIMENTAÇÃO AUTÔNOMA
- 7 VALIDAÇÃO EXPERIMENTAL
- 8 CONCLUSÃO
- 9 REFEÊNCIAS

# Introdução

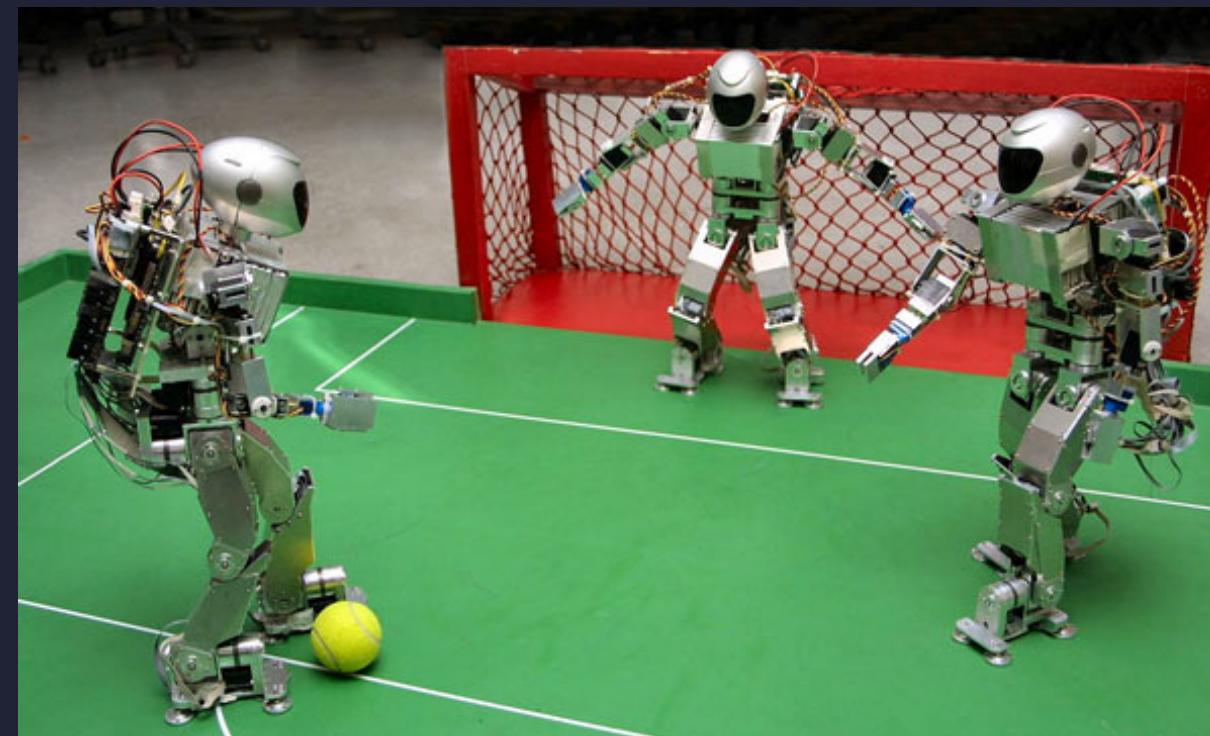
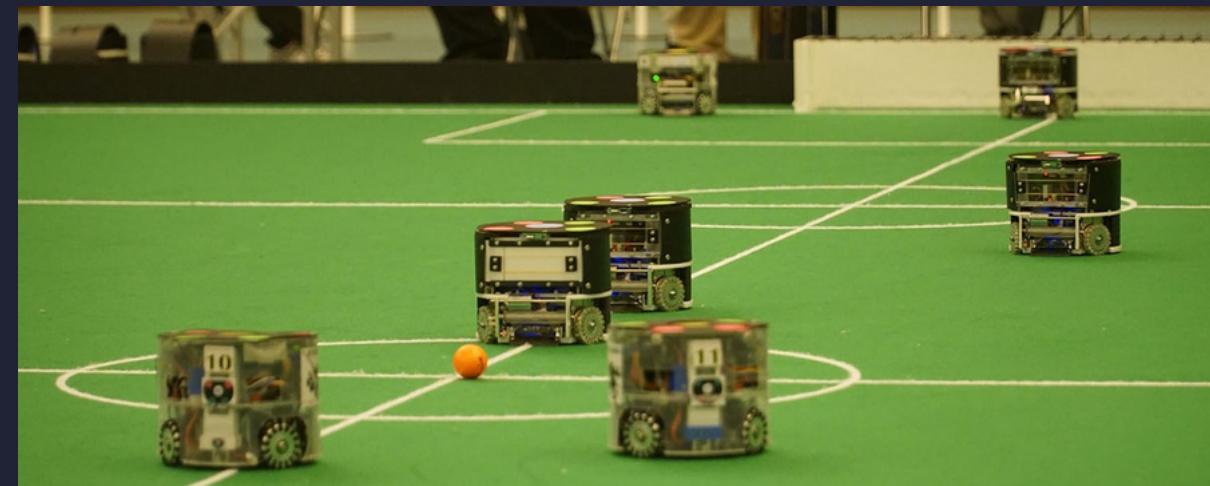
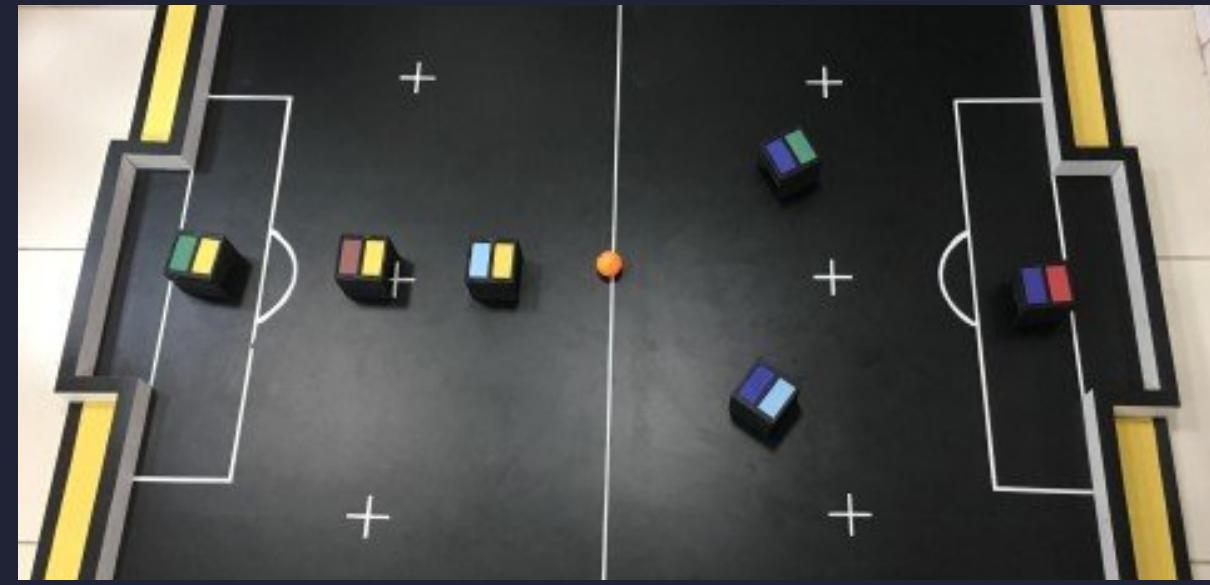
[VOLTAR AO SUMÁRIO](#)

# Por que o Futebol de robôs ?



Competições de futebol com robôs são formas lúdicas de incentivar a realização de PD&I na área de robótica autônoma multi-agente e permitir a implantação de sistemas experimentais de baixo custo no ambiente universitário (e escolar). Um time de robôs com capacidade de jogar futebol, na prática, é um time de robôs que pode atender a muitas outras aplicações.

- planejamento de trajetória
- controle PID para robôs móveis
- visão computacional
- Coordenação de sistemas multirrobóticos



# O que é o Futebol de robôs ?

- O Futebol de robôs é uma adaptação do esporte com humanos.
- Existem várias categorias como VSSS, SSL e Humanóides. As diferenças entre elas estão principalmente nos tipos de robôs e nas dimensões do campo e da bola.

# Objetivos

Desenvolver um time de futebol de robôs da categoria IEEE VSSS

## 1. Construção:

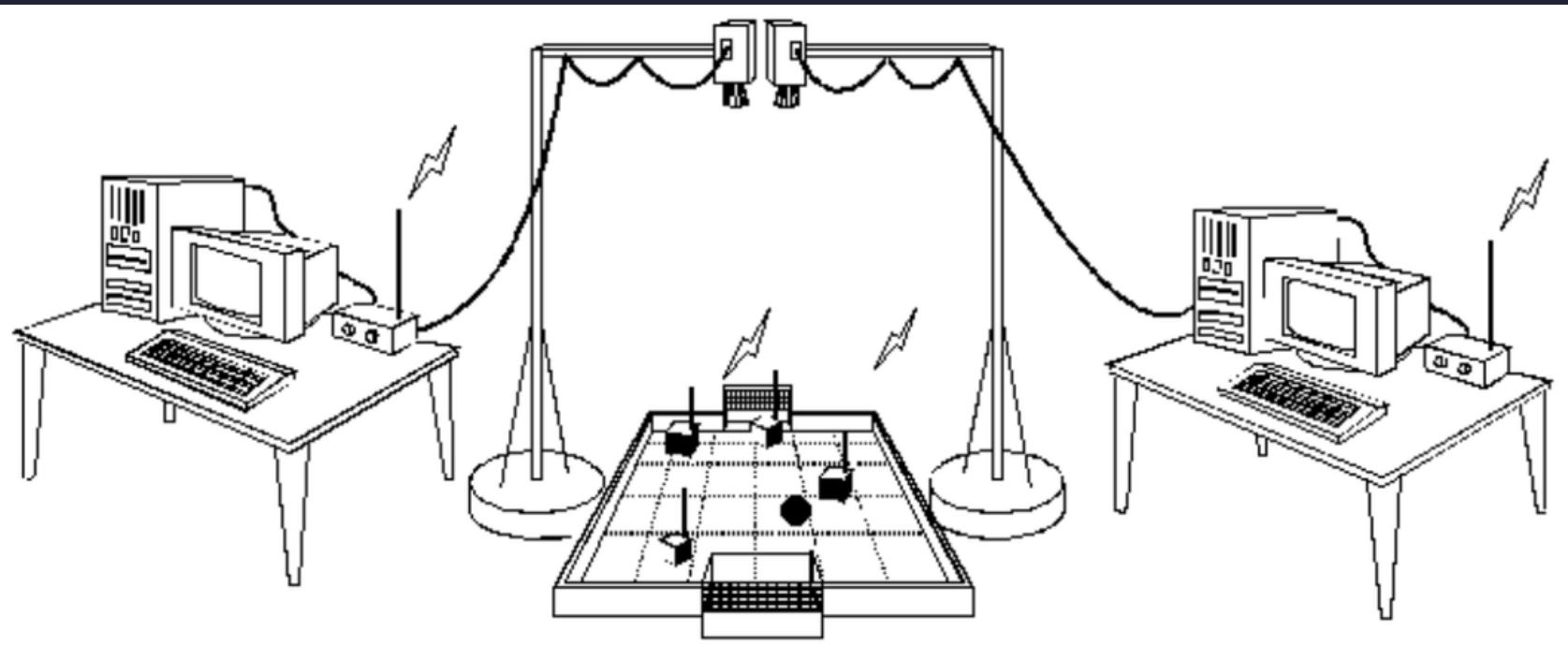
1. Fabricação dos robôs, sua estrutura mecânica dos robôs e sistema de controle embarcado;
2. Construção do campo de jogo e estrutura menor para testes;
3. Fabricação de estrutura auxiliar para posicionamento de câmeras.

## 2. Desenvolvimento do(a):

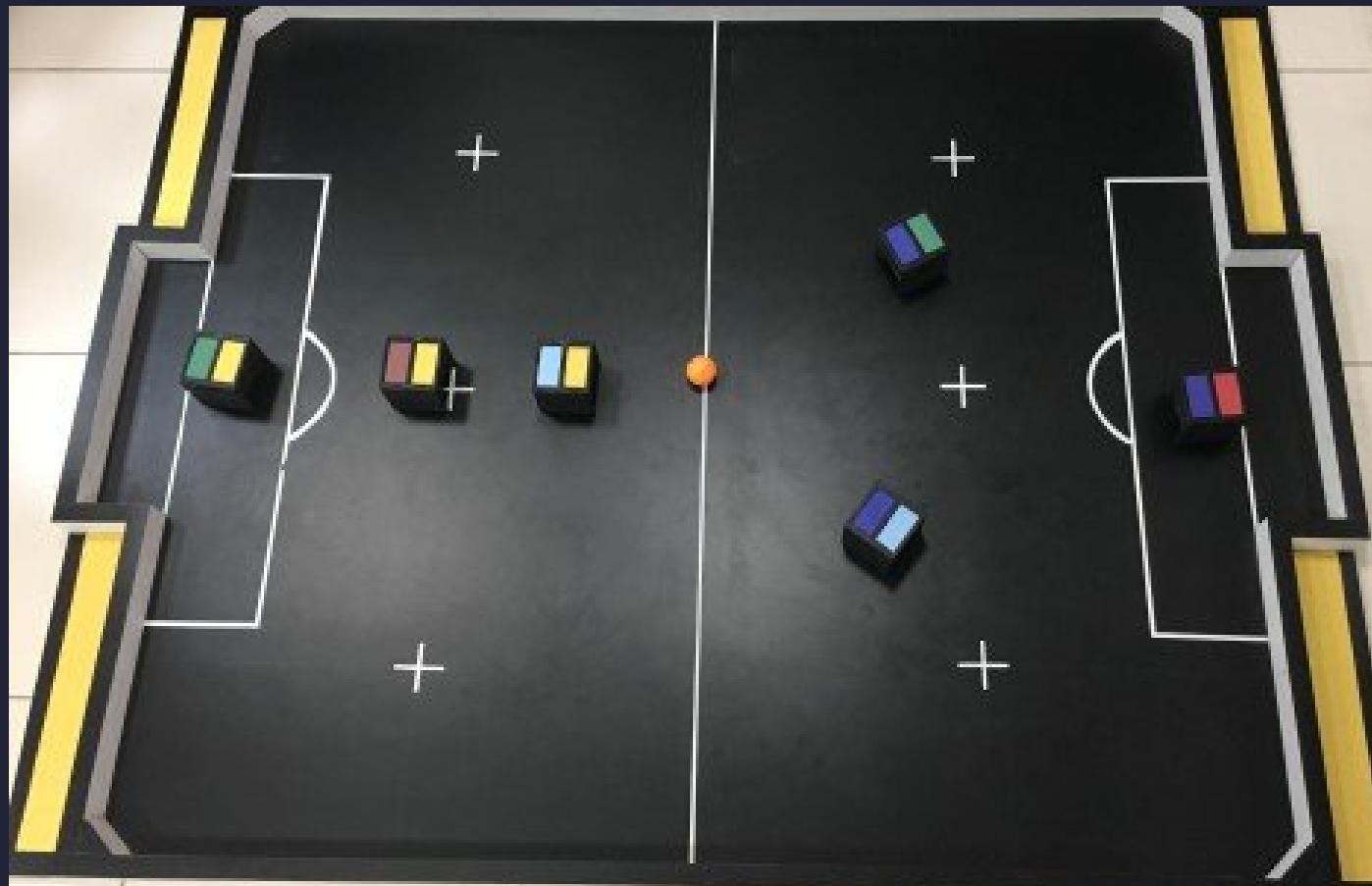
1. Canal de comunicação do computador com os robôs em campo;
2. Algoritmo de visão computacional capaz de identificar os jogadores, e a bola;
3. Interface gráfica para monitoramento do jogo e comunicação com os jogadores;
4. Algoritmos de controle para movimentação autônoma dos robôs.

# A categoria IEEE VSSS

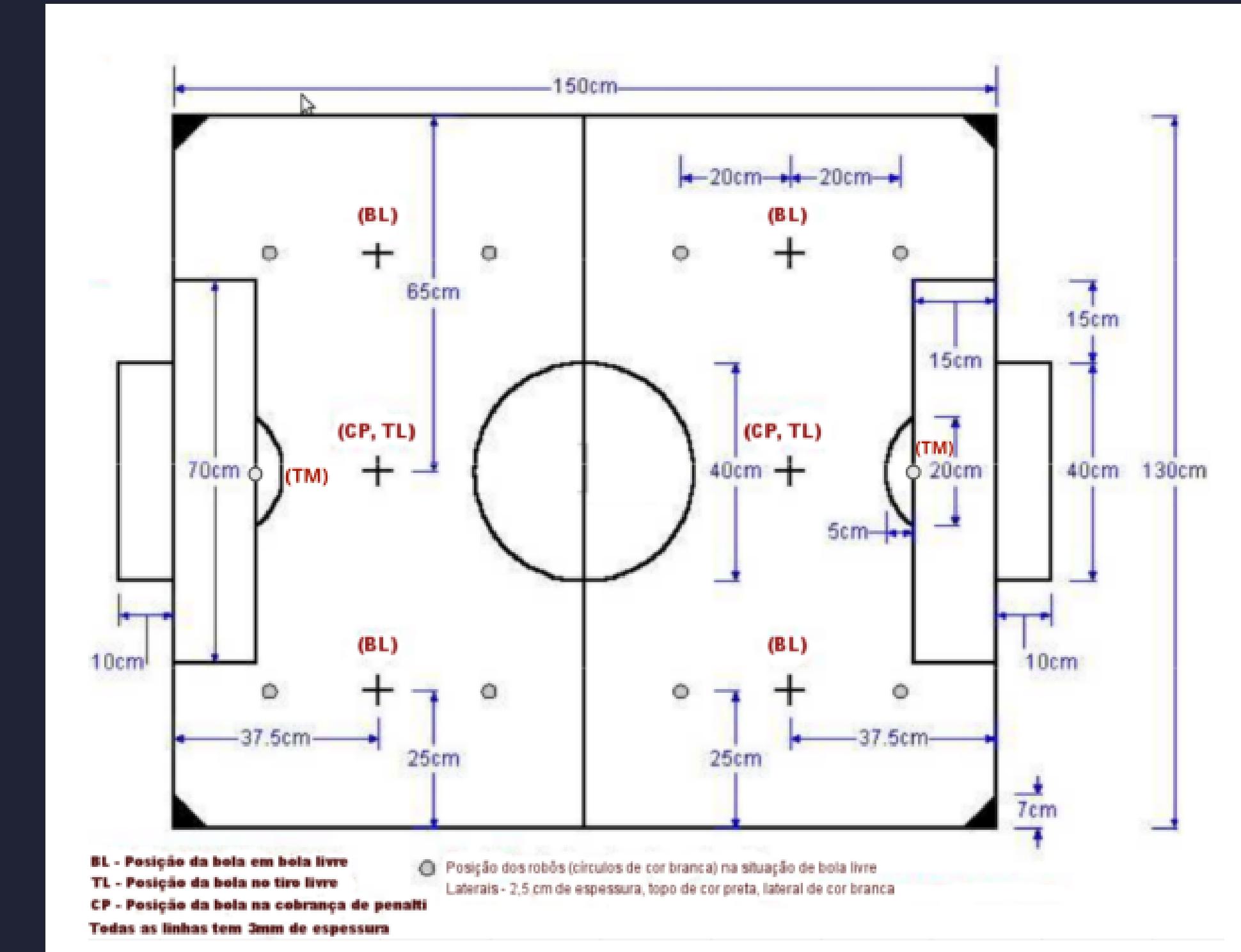
- VSSS significa Very Small Size Soccer, é uma das categorias padronizadas de futebol de robôs pelo IEEE. Jogam dois times com 3 jogadores cada.
- As disputas ocorrem em dois tempos de 5 minutos cada, com intervalo de 8 minutos entre elas.
- Cada time é composto por 3 jogadores, um sistema de visão computacional com a camera e suporte, um computador e um transmissor.
- O sistema de visão afere a pose dos jogadores e da bola. O computador recebe os dados de visão e processa as jogadas, definindo a movimentação dos jogadores que são transmitidas através do transmissor.



# O campo



O campo possui dimensões de 150cm por 130cm. Ele é feito em madeira, com fundo pintado de preto fosco e marcações em branco brilhante



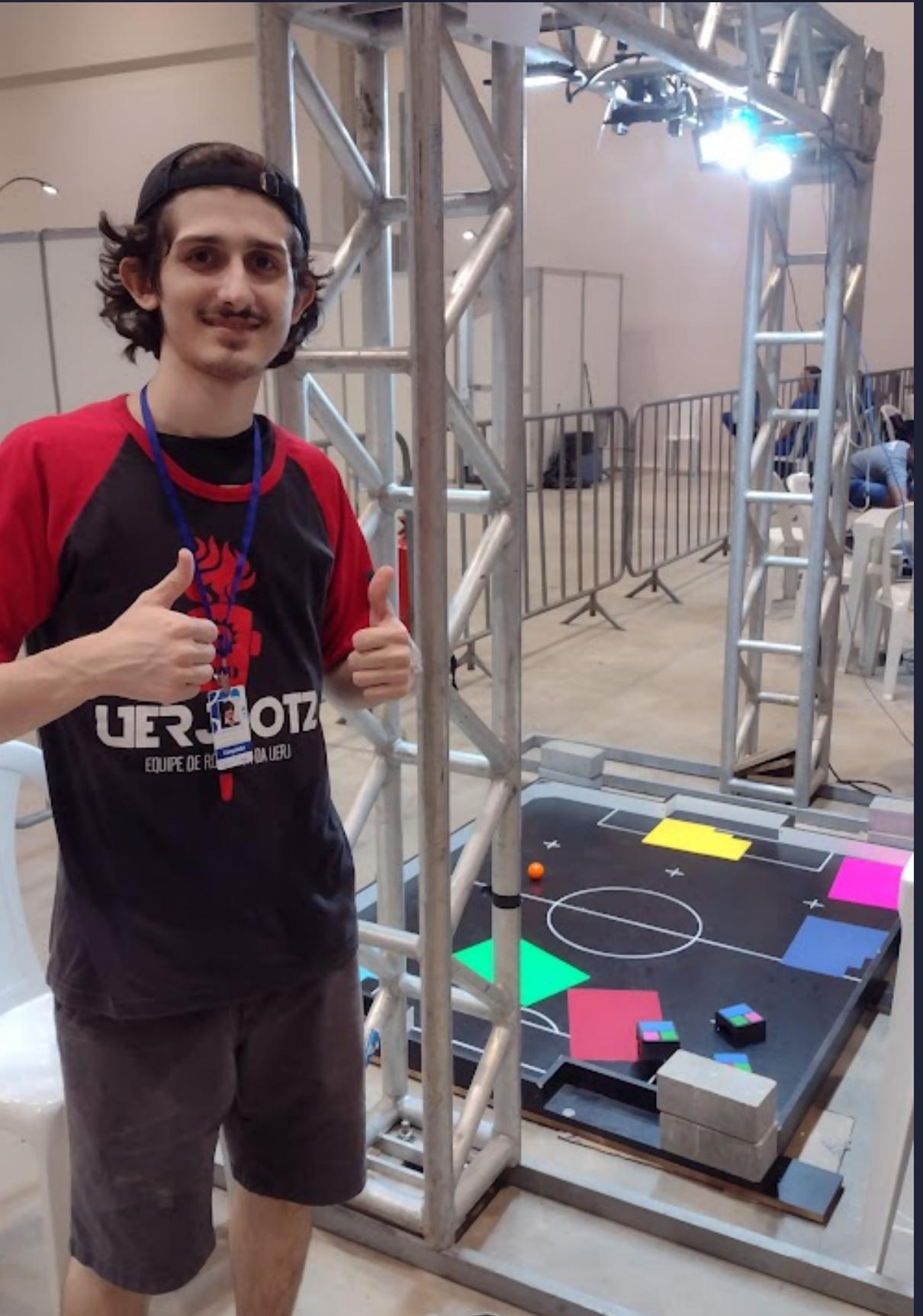
# A bola



Bola é laranja, similar a uma bola de golfe, com 42,7mm de diâmetro e 46g de peso.

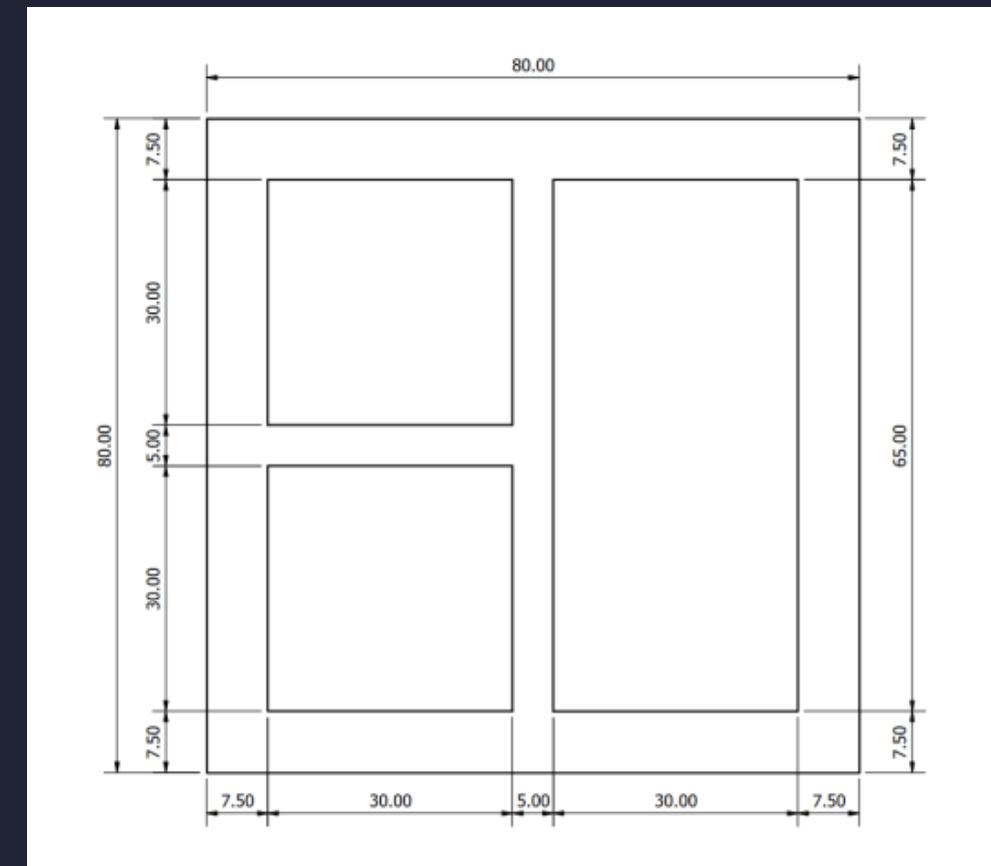
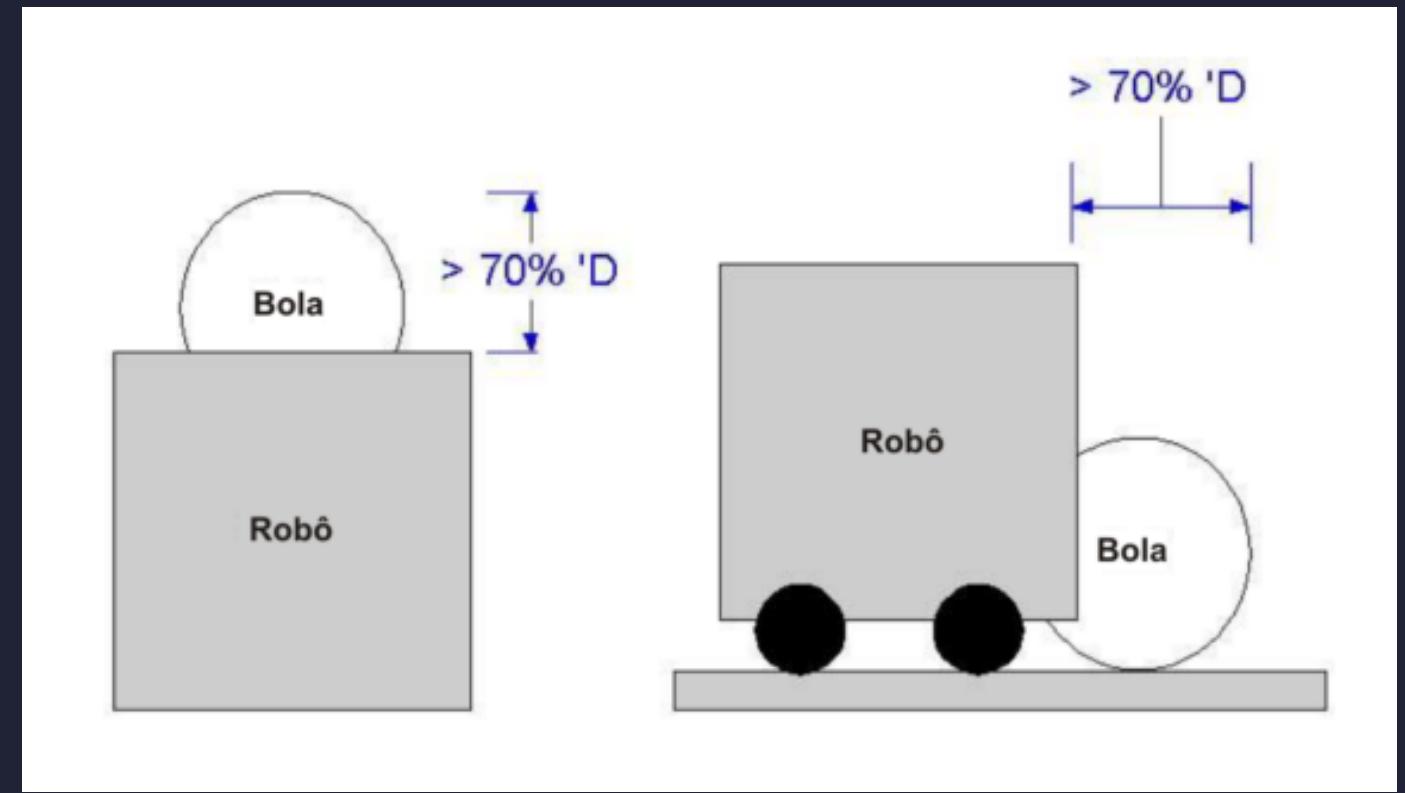
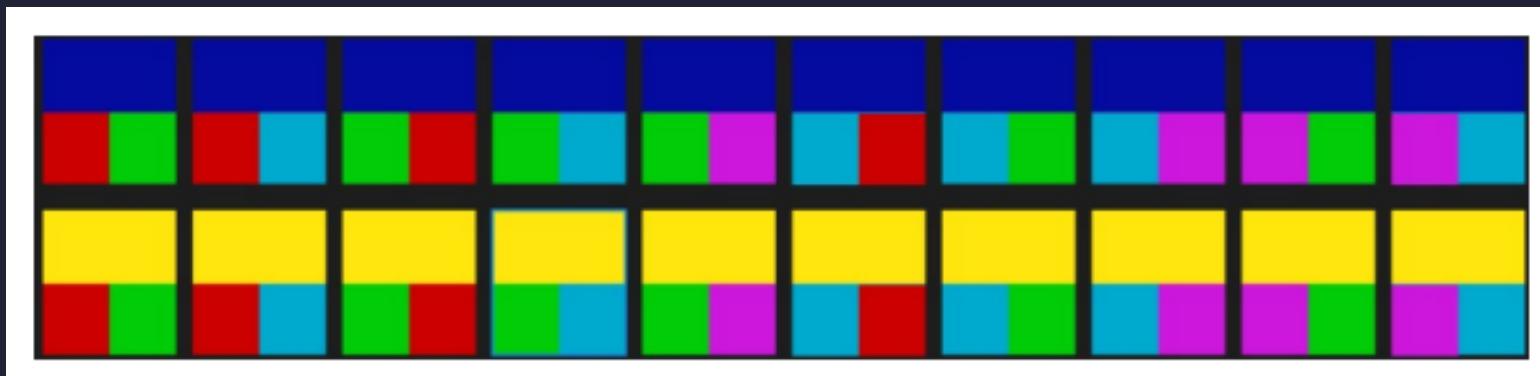
# Sistema de visão

A parte física do sistema de visão é composto por uma câmera fixada em uma estrutura lateral que garante distância adequada entre a câmera e o campo, que de acordo com as regras da categoria deve ser de 2 metros. Além disso existe sistema de iluminação.



# Jogadores

- Controle sem fio (geralmente IR ou Rádio)
- dimensões limites de 75x75x75mm, com uniforme podem ter até 80x80x80mm.
- No topo possuem etiquetas, que indicam a cor do time, amarelo ou azul, o número do jogador. E funcionam como identificador para o sistema de visão.
- Cada time pode designar um jogador para a função de goleiro. É permitido apenas ao goleiro segurar a bola em sua própria área e encobrir mais do que 30% do diâmetro da bola.



# Montagem dos robôs e elementos do jogo

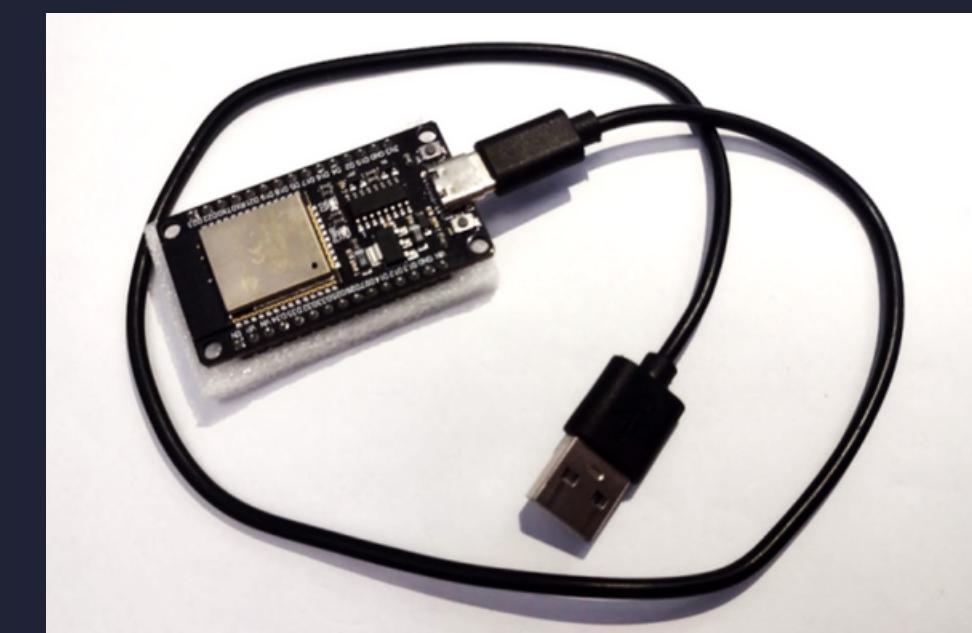
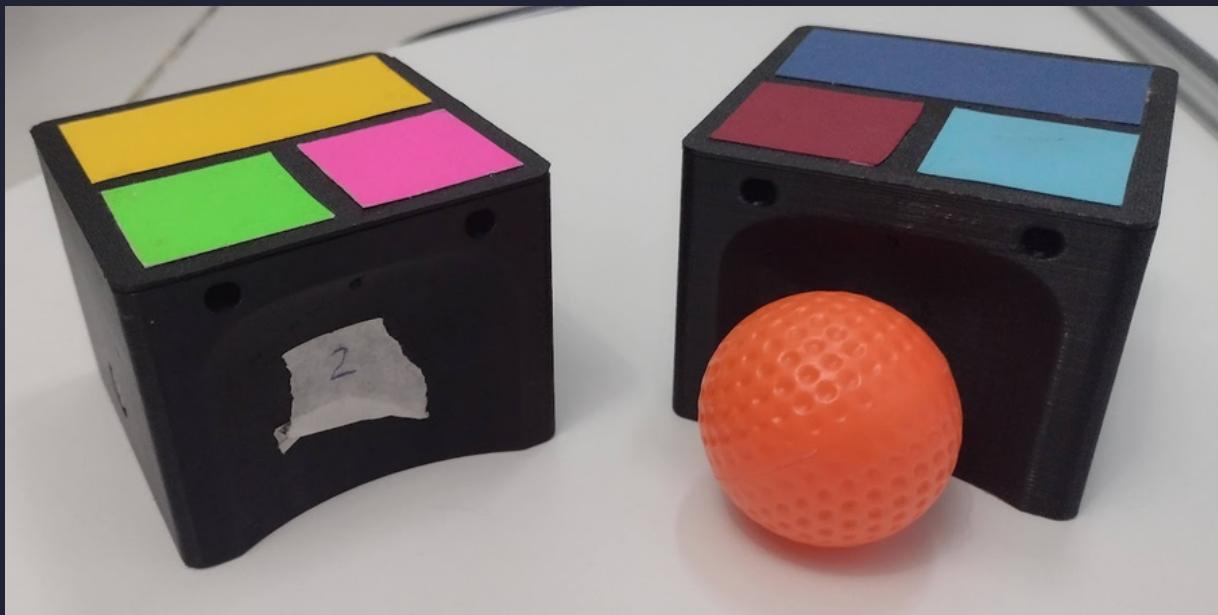
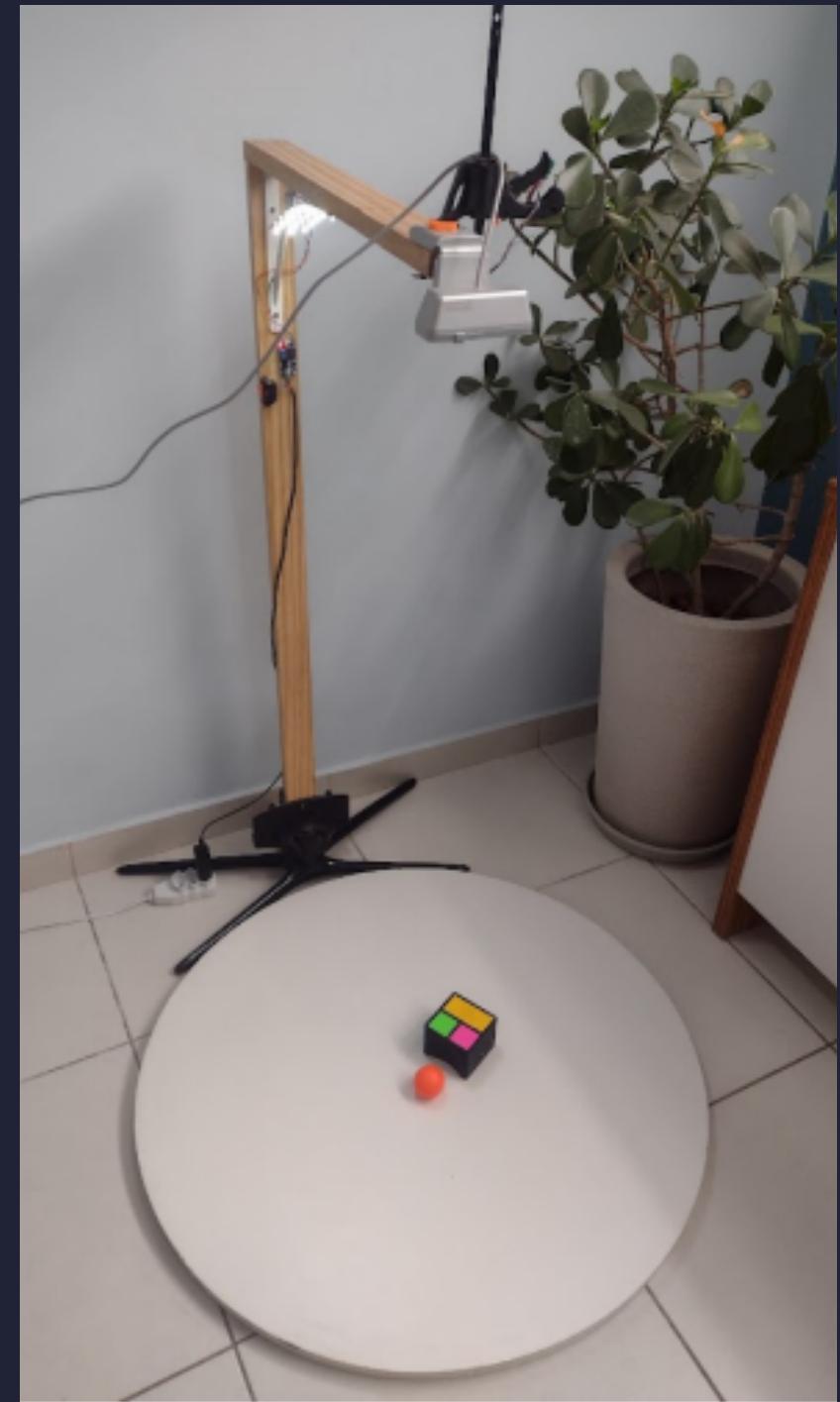
[VOLTAR AO SUMÁRIO](#)



# Montagem dos robôs e elementos do jogo

Foram desenvolvidos:

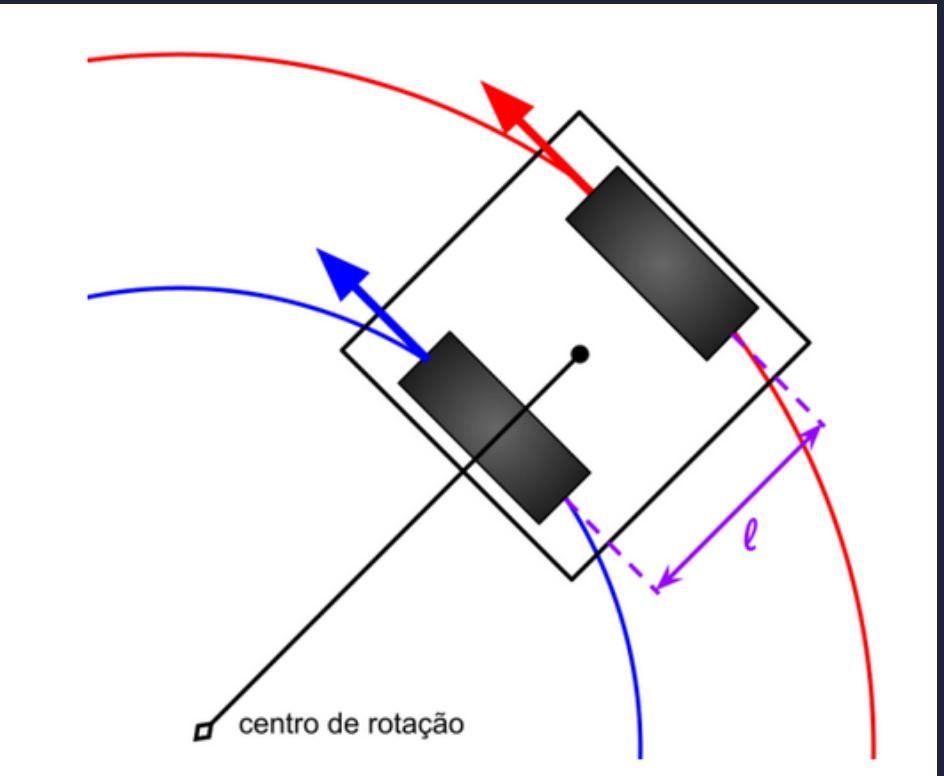
- Jogadores
- Campo
- Estrutura da Camera
- Transmissor



# Montagem dos robôs – Sistema de locomoção

Foi adotado o sistema de locomoção do tipo diferencial, por conta da facilidade de montagem e maior liberdade de movimentação.

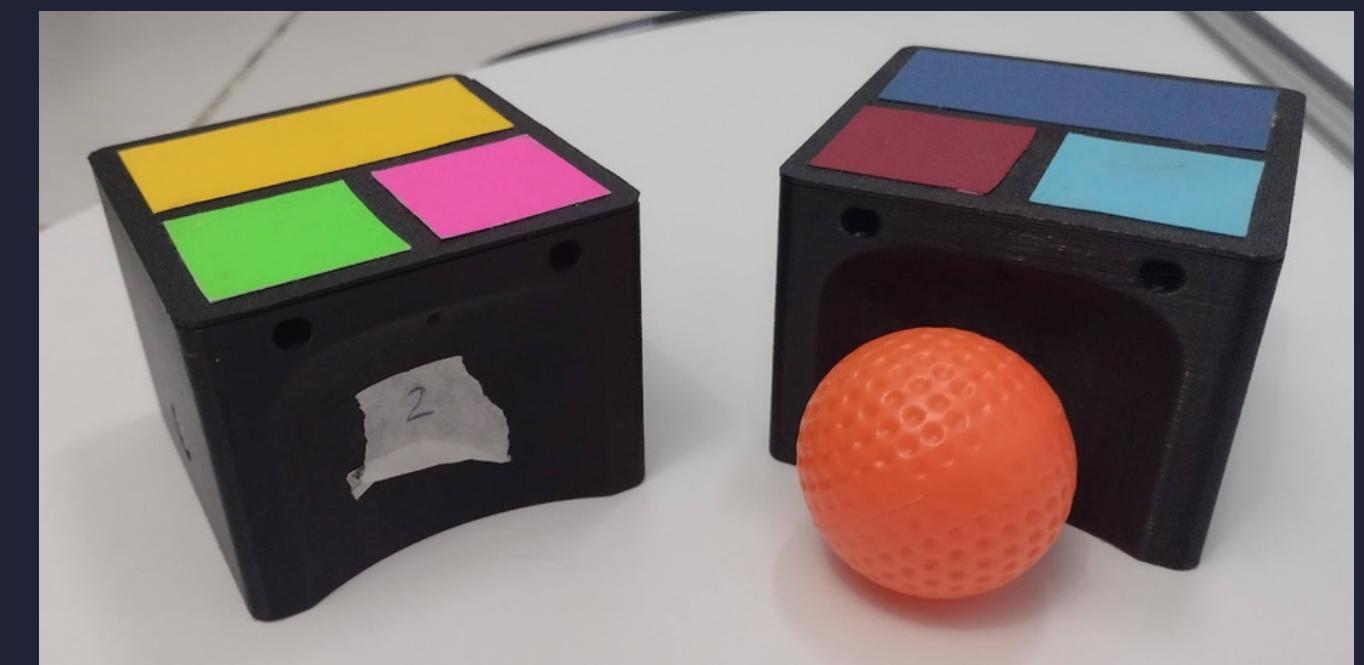
- Dois Motores N20:
  - Micromotores de corrente contínua escovados de ímãs permanentes.
  - Cada um pesa 9,6g
  - Micro-redução na saída.
  - Velocidade nominal de 500RPM
  - Tensão nominal de 6V
- Duas Rodas
  - Penus de silicone
  - 34mm de diâmetro



# Projeto Mecânico

As partes mecânicas dos robôs são impressas em material PLA (Biopolímero ácido poliláctico) e PETG (Polietileno Tereftalato Glicol). A modelagem da estrutura dos robôs foi realizada usando o software CAD (Computer Aided-Design) Onshape. Requisitos:

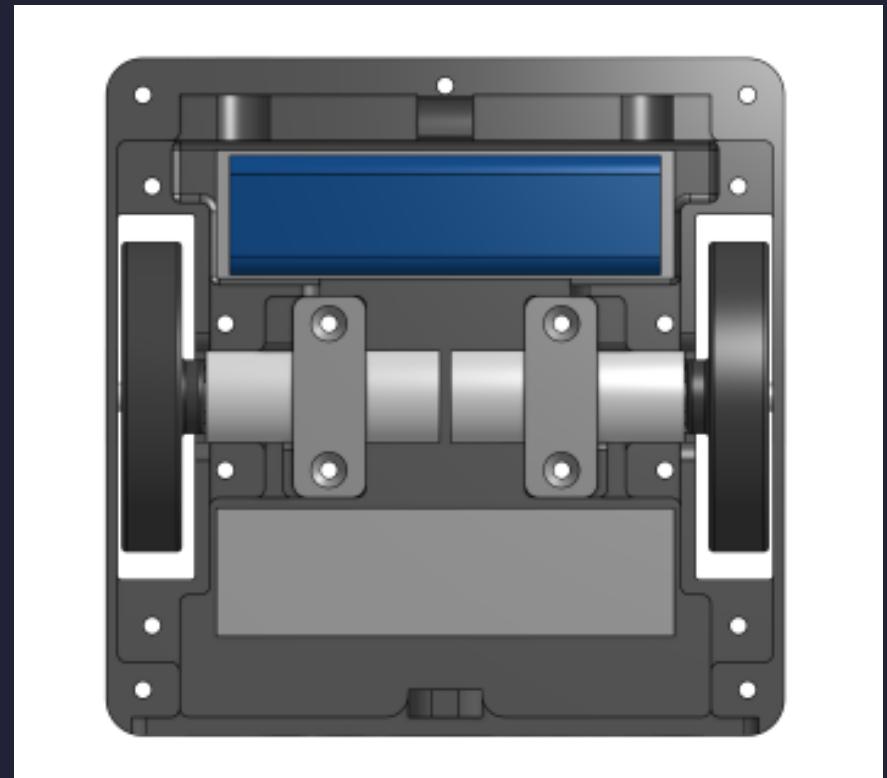
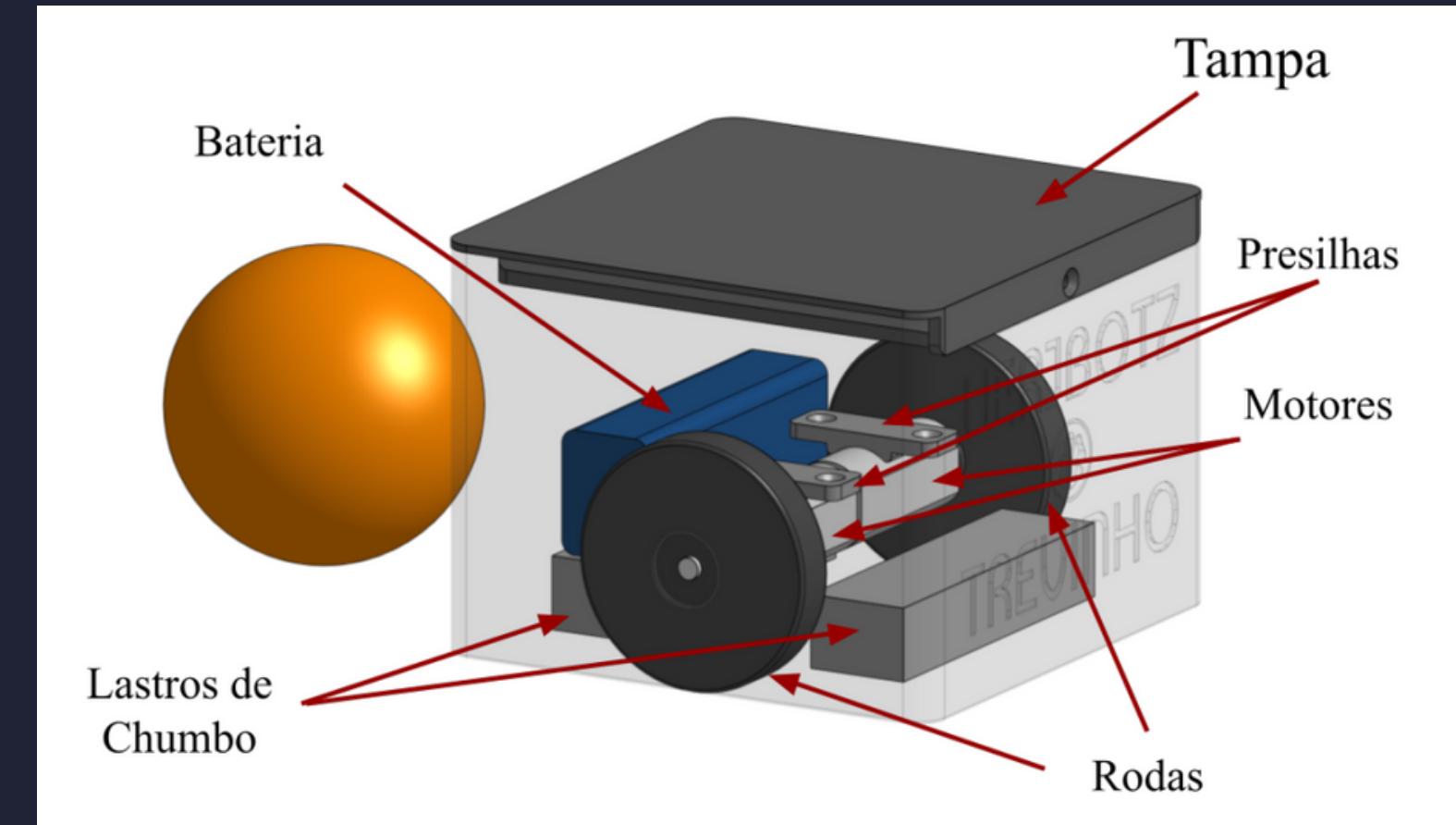
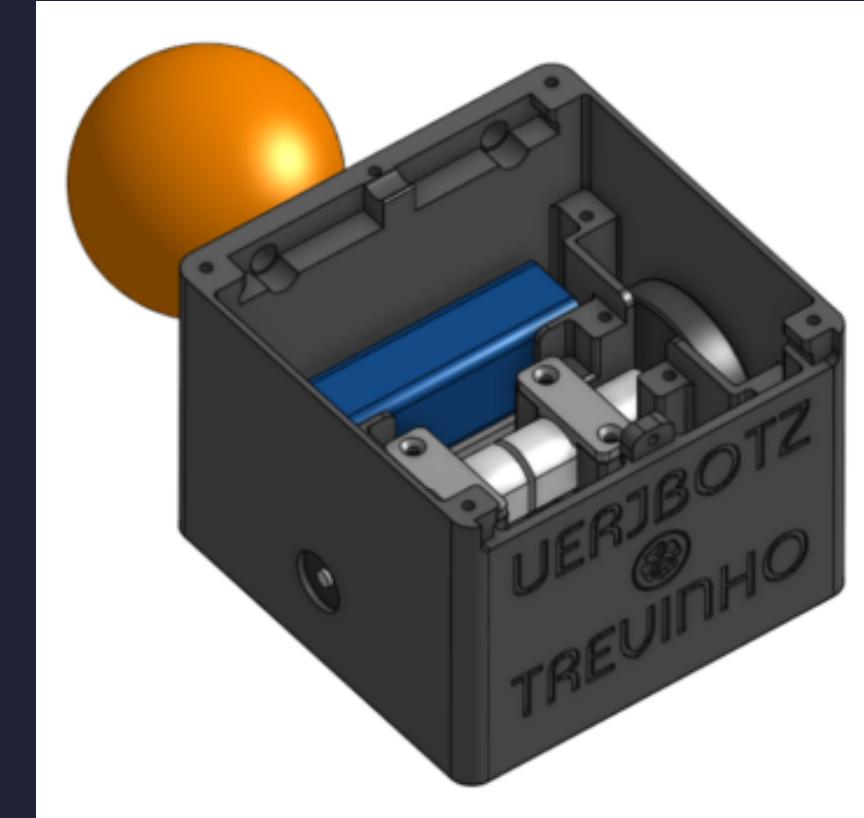
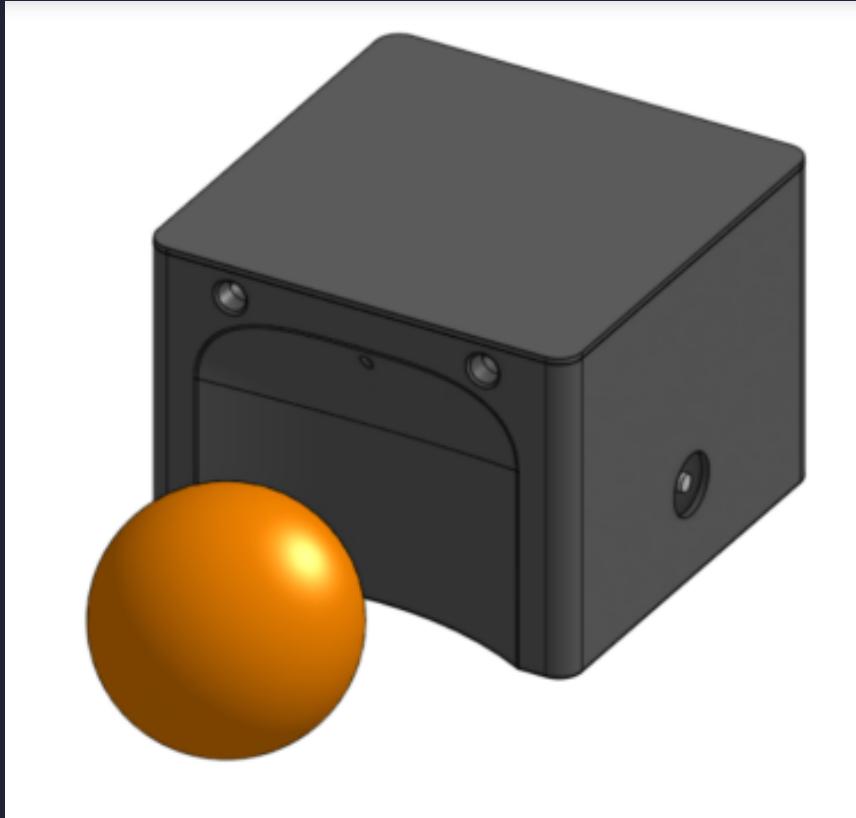
1. Peças viáveis para impressão 3D, minimizando a dependência de suportes durante a impressão;
2. Imitação de espaço de até 75 x 75 x 75mm;
3. Espaço interno suficiente para acomodar a bateria, os componentes eletrônicos e lastro de chumbo;
4. Centro de massa no centro do robô em vista superior e o mais baixo possível em relação ao chão.
5. O robô não deve encobrir mais que 30% do diâmetro da bola em vista superior;
6. Tampa fácil e rápida de ser trocada;
7. Facilidade de montagem e troca de peças;
8. Preferência ao uso de encaixes ao invés de parafusos;
9. Menor número possível de peças;
10. Locomoção diferencial;



# Projeto Mecânico – Modelo final

Todos os requisitos foram atendidos com sucesso. Cada jogador possui duas rodas com pneus de silicone e 4 peças impressas.

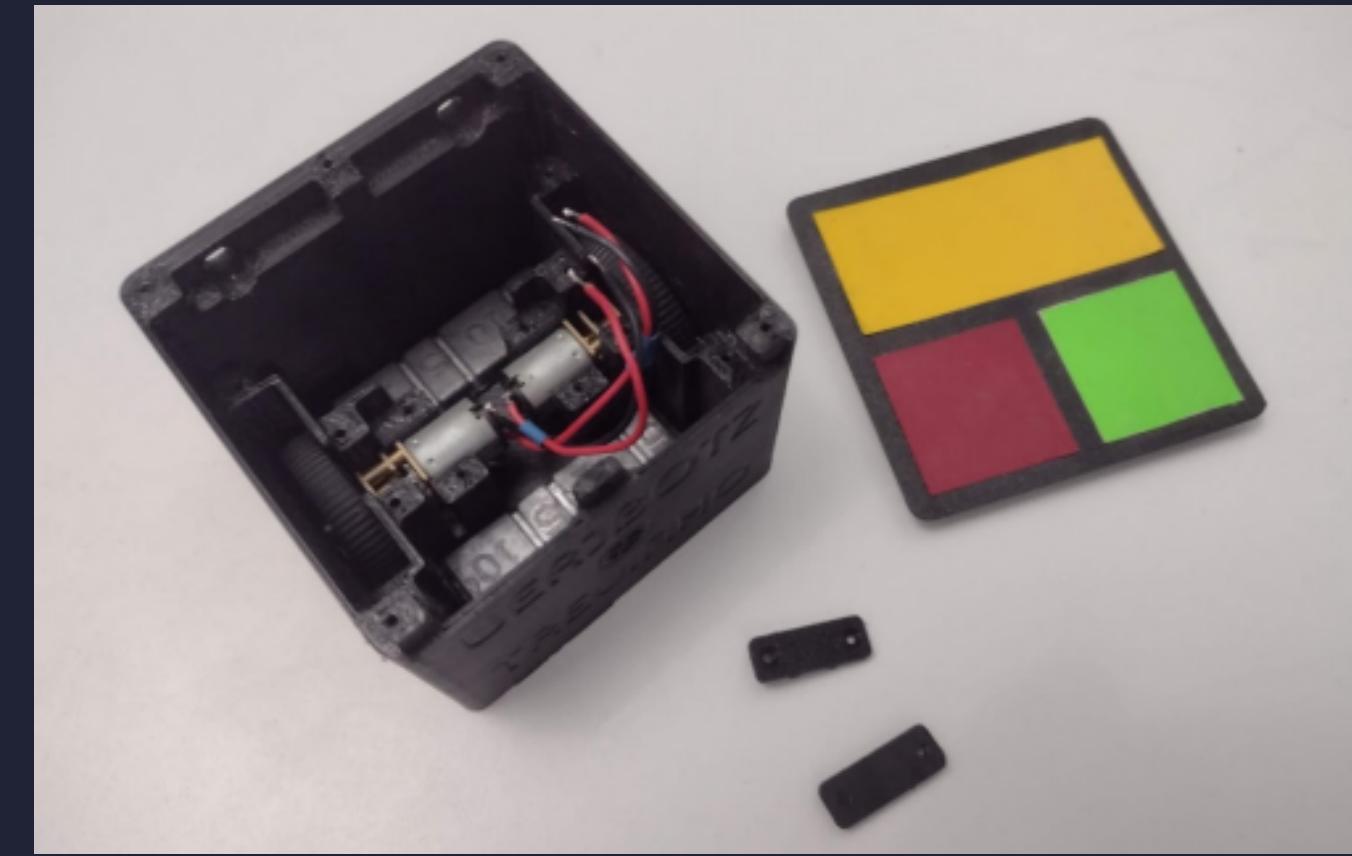
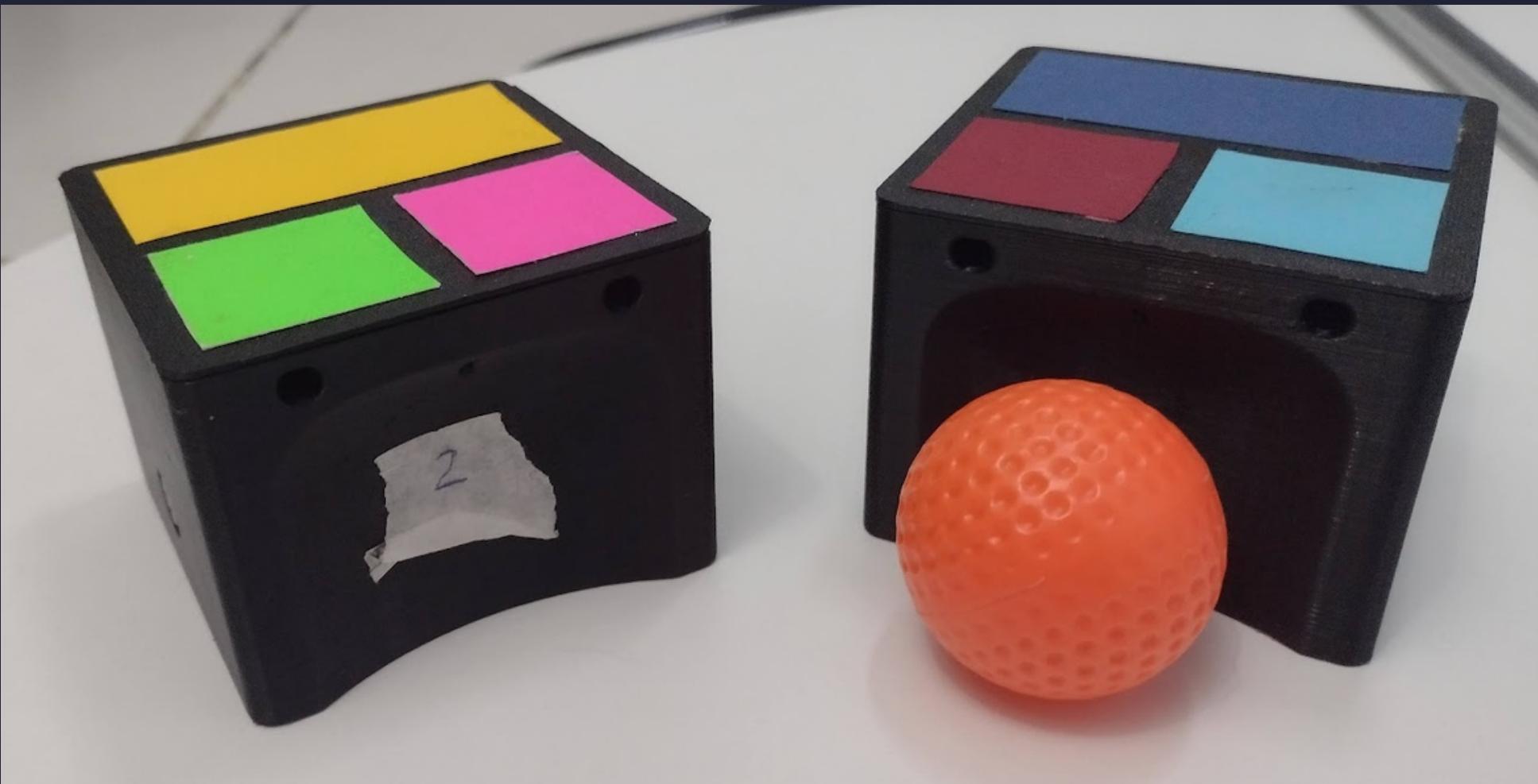
- Carcaça lateral;
- Duas presilhas de motores;
- Tampa removível.



# Projeto Mecânico – Resultado

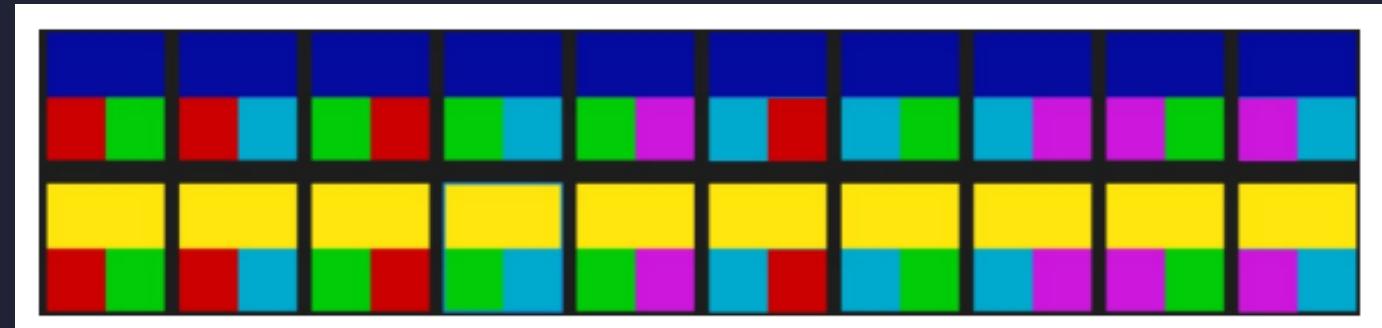
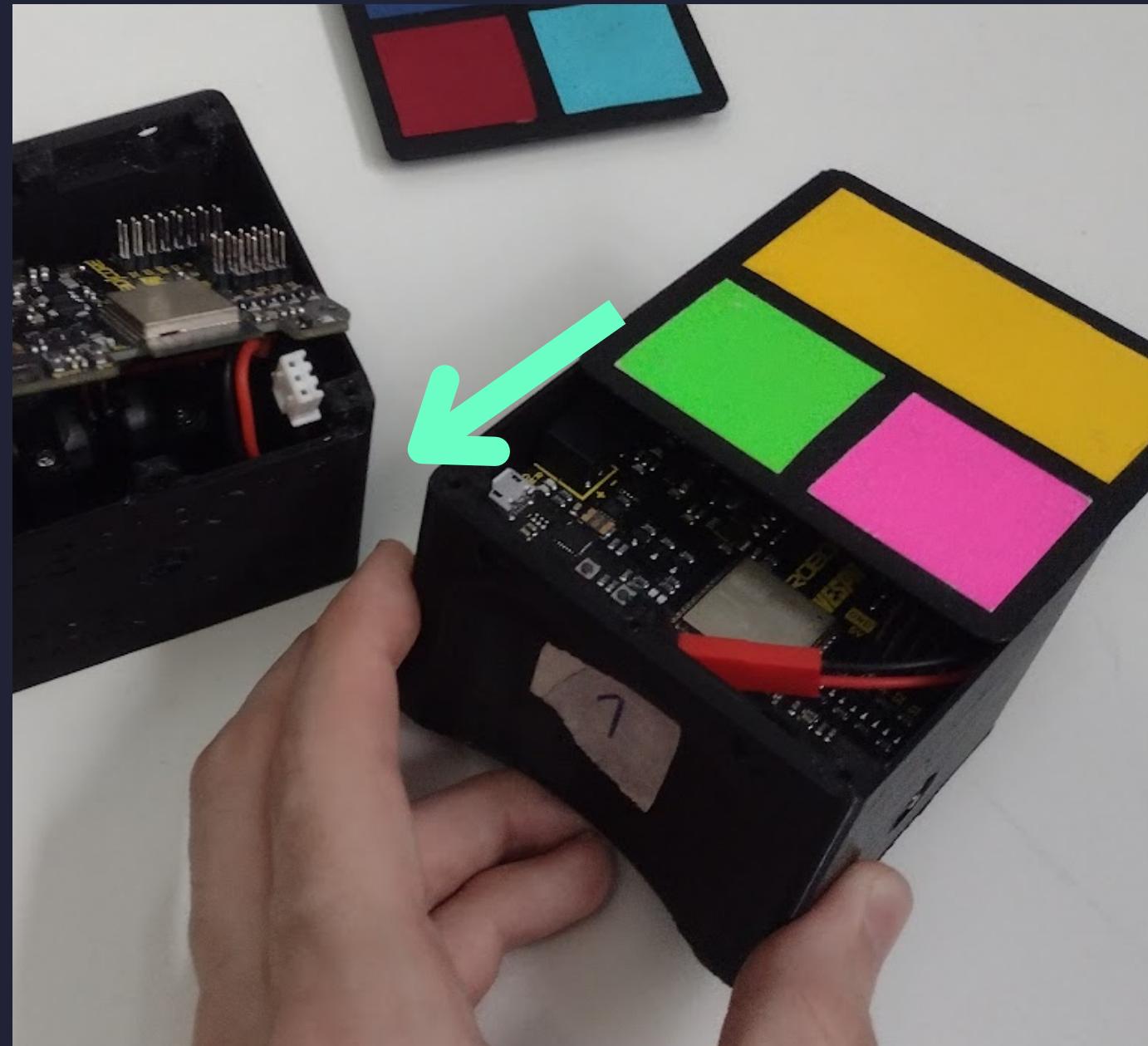
4 peças impressas.

- Carcaça lateral;
- Duas presilhas de motores;
- Tampa removível.



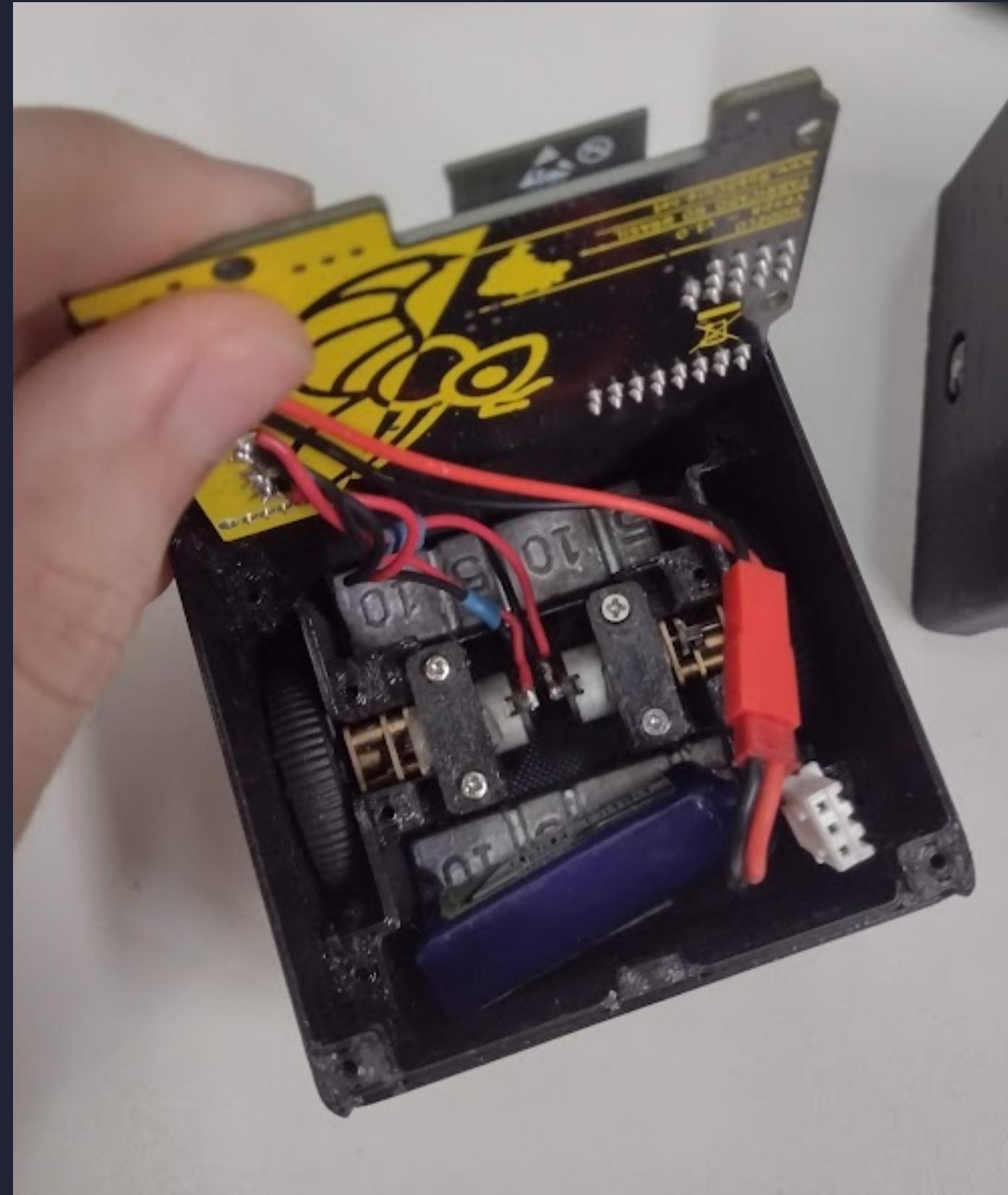
# Projeto Mecânico – As etiquetas

- Removíveis
- Etiquetas no padrão da categoria



# Projeto Mecânico - Lastro

O lastro de chumbo serve para aumentar a tração nas rodas dos robôs. A tração é importante para garantir força em disputas de bolas e aceleração.



# Sistema Eletrônico e seus requisitos

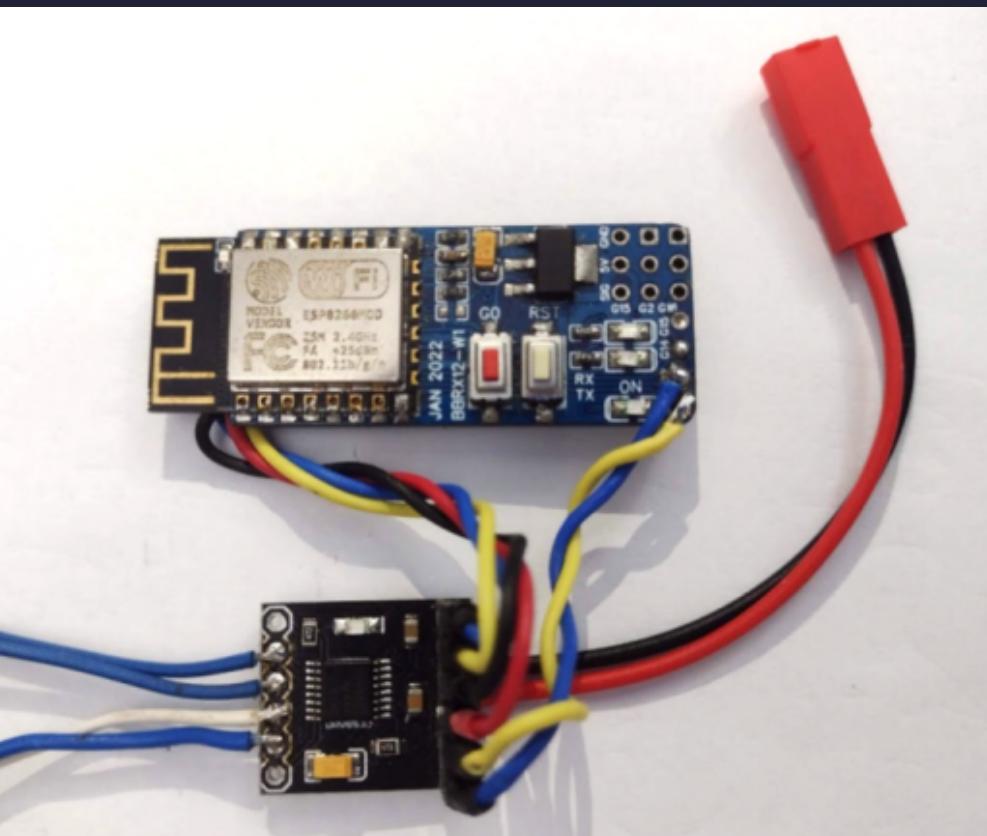
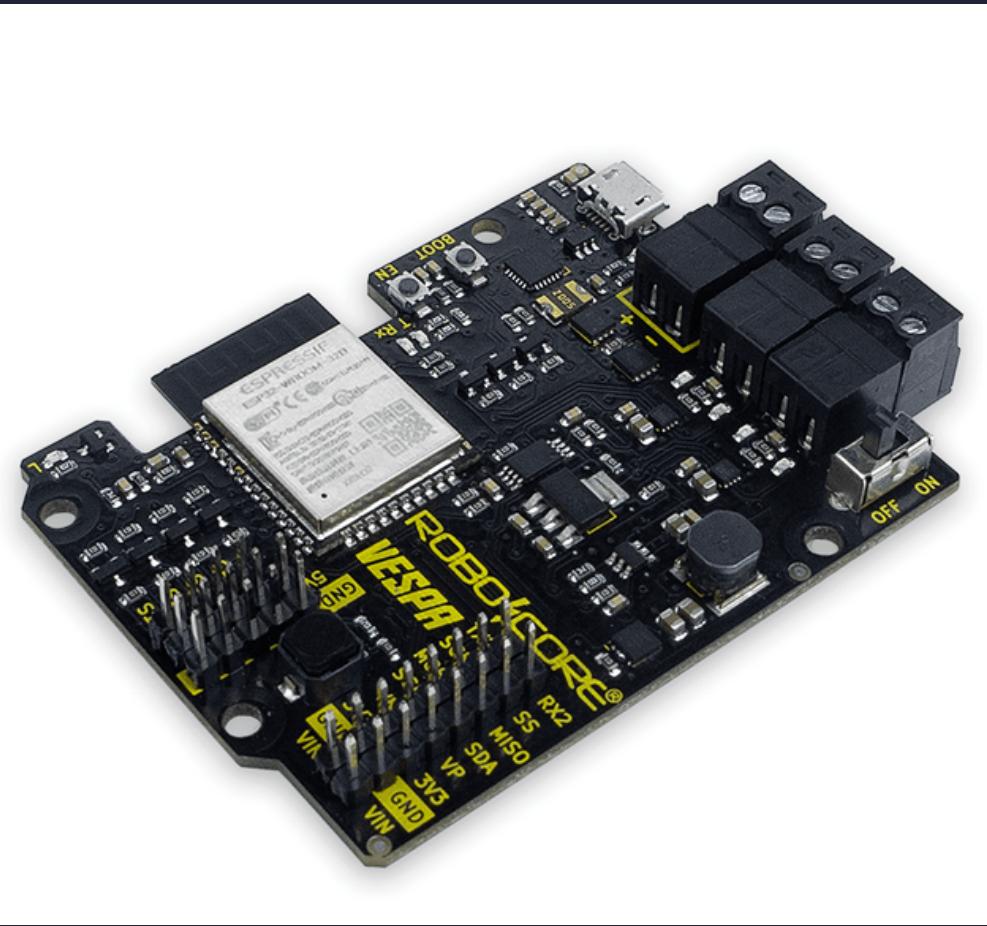
A função do sistema é controlar a velocidade angular e tangencial dos robôs em função dos comandos recebidos pelo computador. São requisitos:

1. Usar microcontroladores da Espressif que permitam o uso do protocolo ESPNOW.
2. Possuir circuito de alimentação capaz de receber baterias 2S (com duas células) ou seja, tensão máxima de entrada superior a 8,4V.
3. Possuir circuito de proteção contra polaridade reversa, para proteger em caso do conector da bateria ser conectado ao contrario.
4. Ser capaz de medir a tensão da bateria para sinalizar a necessidade de troca.
5. Possuir duas pontes H que possam ser alimentadas com tensão 2S e possuir corrente continua de ao menos 600mA, para garantir força e aceleração adequadas dos motores.
6. Possuir unidade inercial para balizar a regulação de velocidade angular.

# Sistema Eletrônico – Abordagens

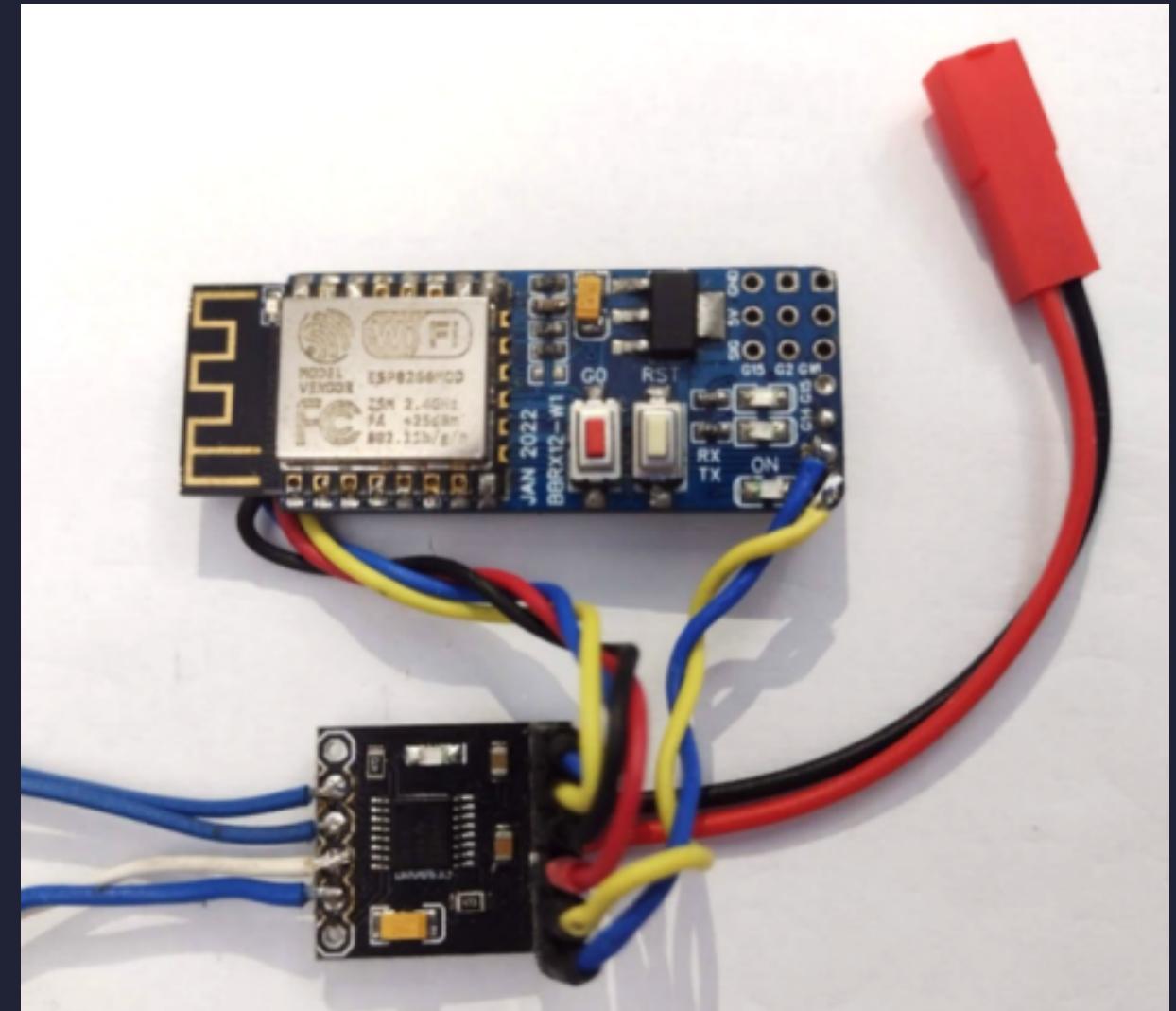
Foram adotadas duas abordagens equivalentes na montagem dos sistemas eletrônicos, visando permitir a confecção de mais robôs. Abordagens:

1. Montar a placa de controle com componentes discretos, módulos e circuitos integrados.
2. Utilizar a placa Vespa, fabricada pela empresa Robocore, que contém quase todos os circuitos necessários pra controlar os robôs.

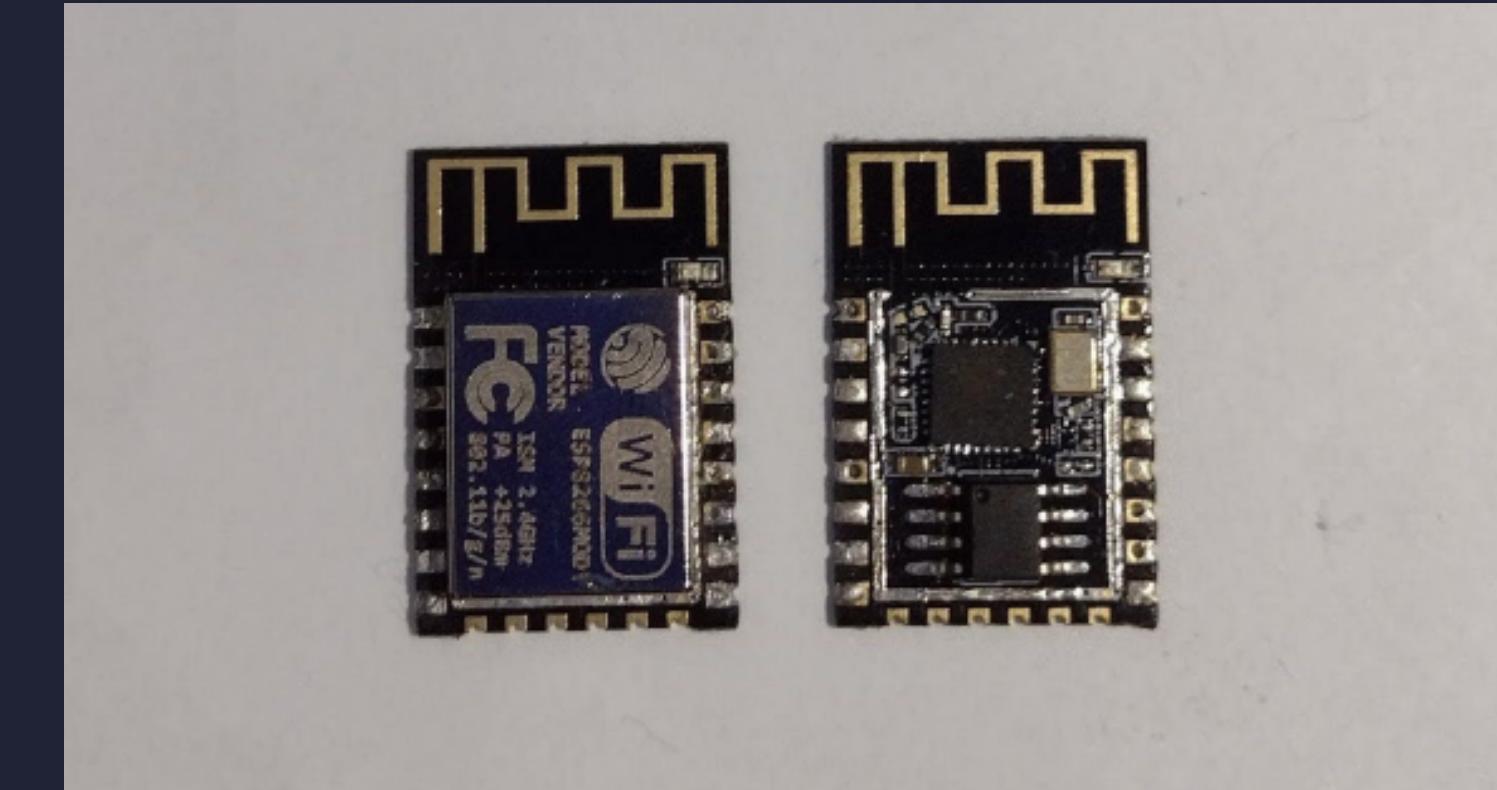
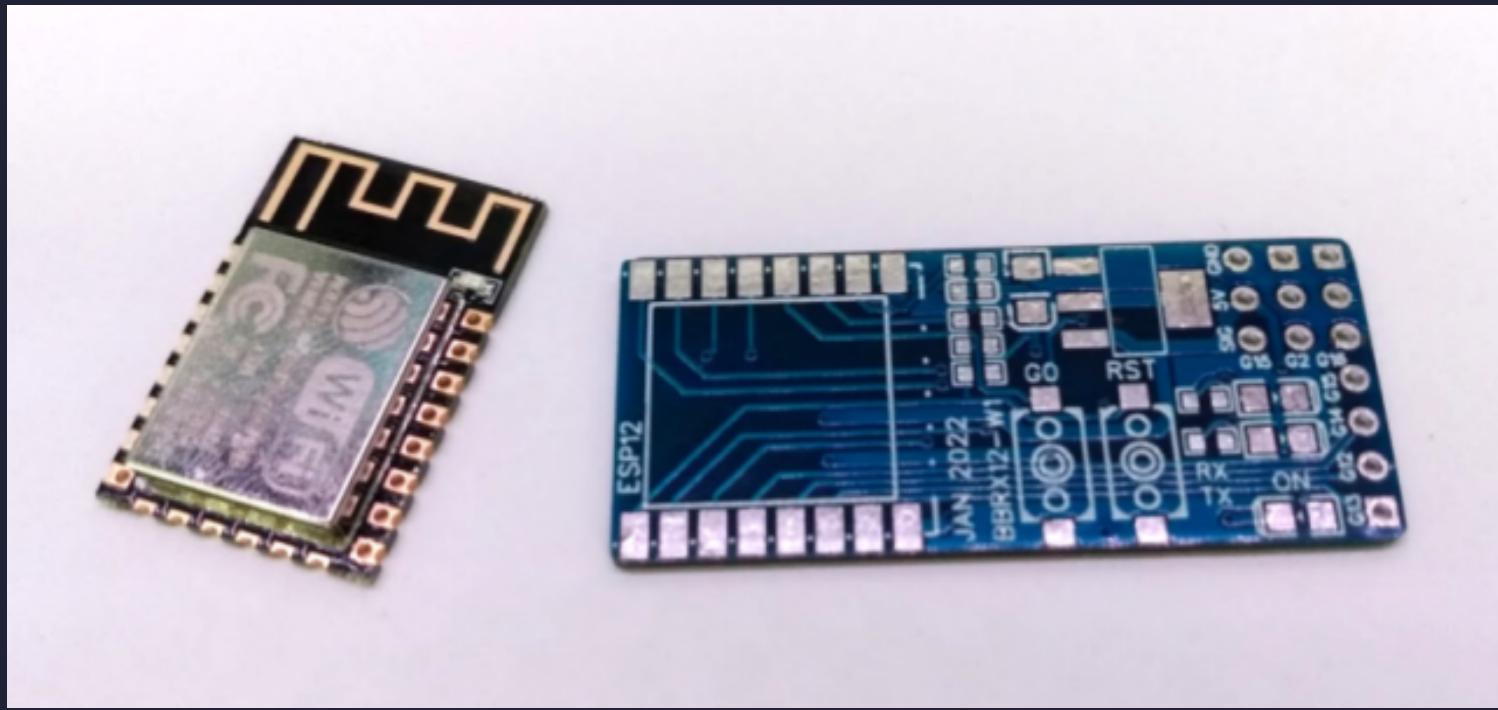
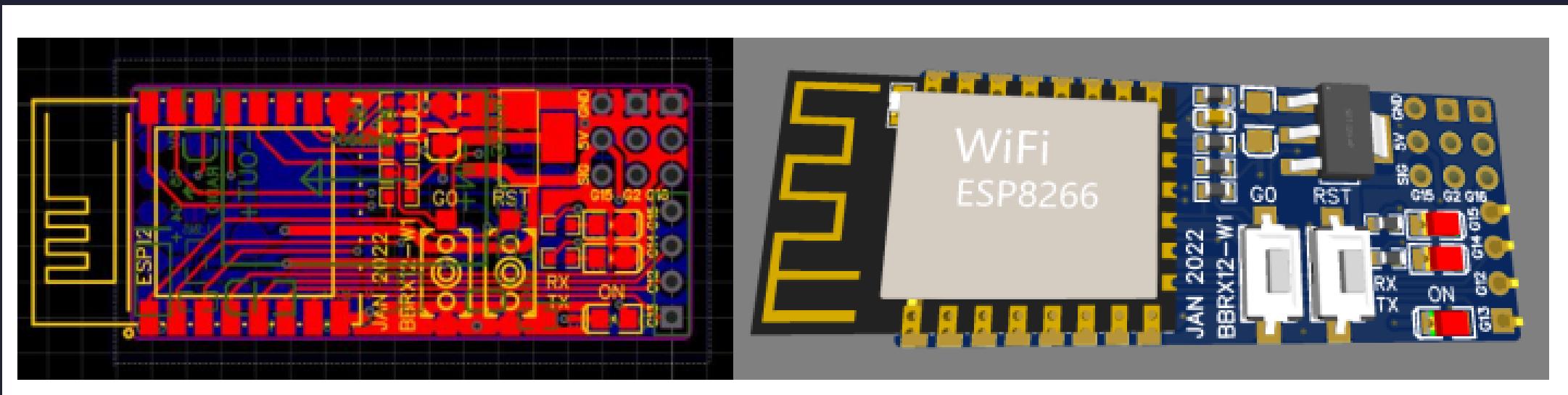


# Sistema Eletrônico – Abordagem 1

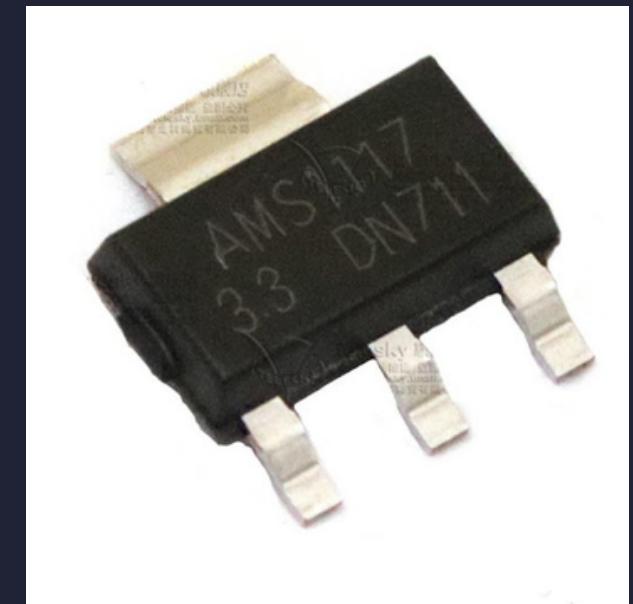
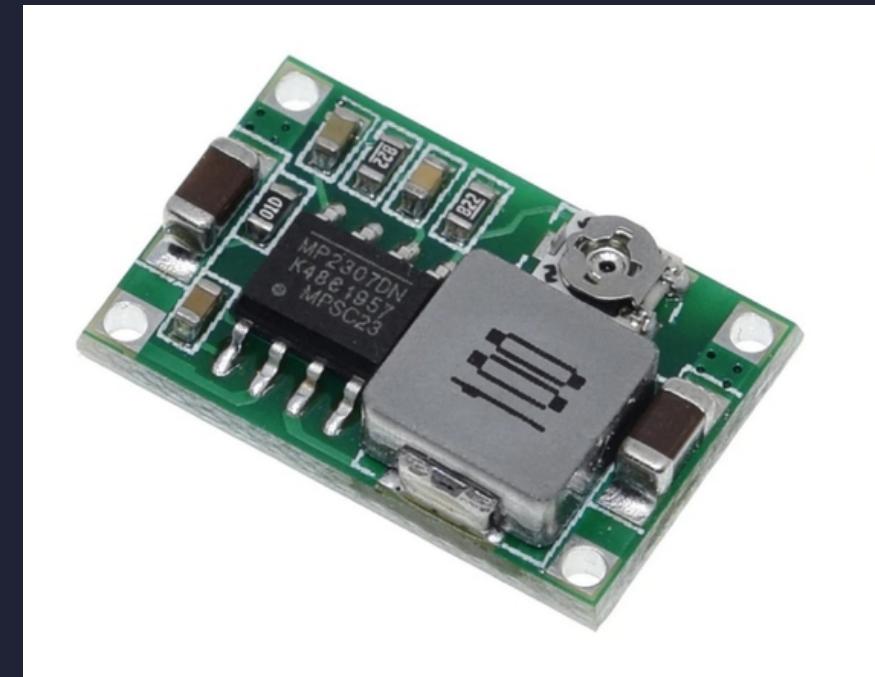
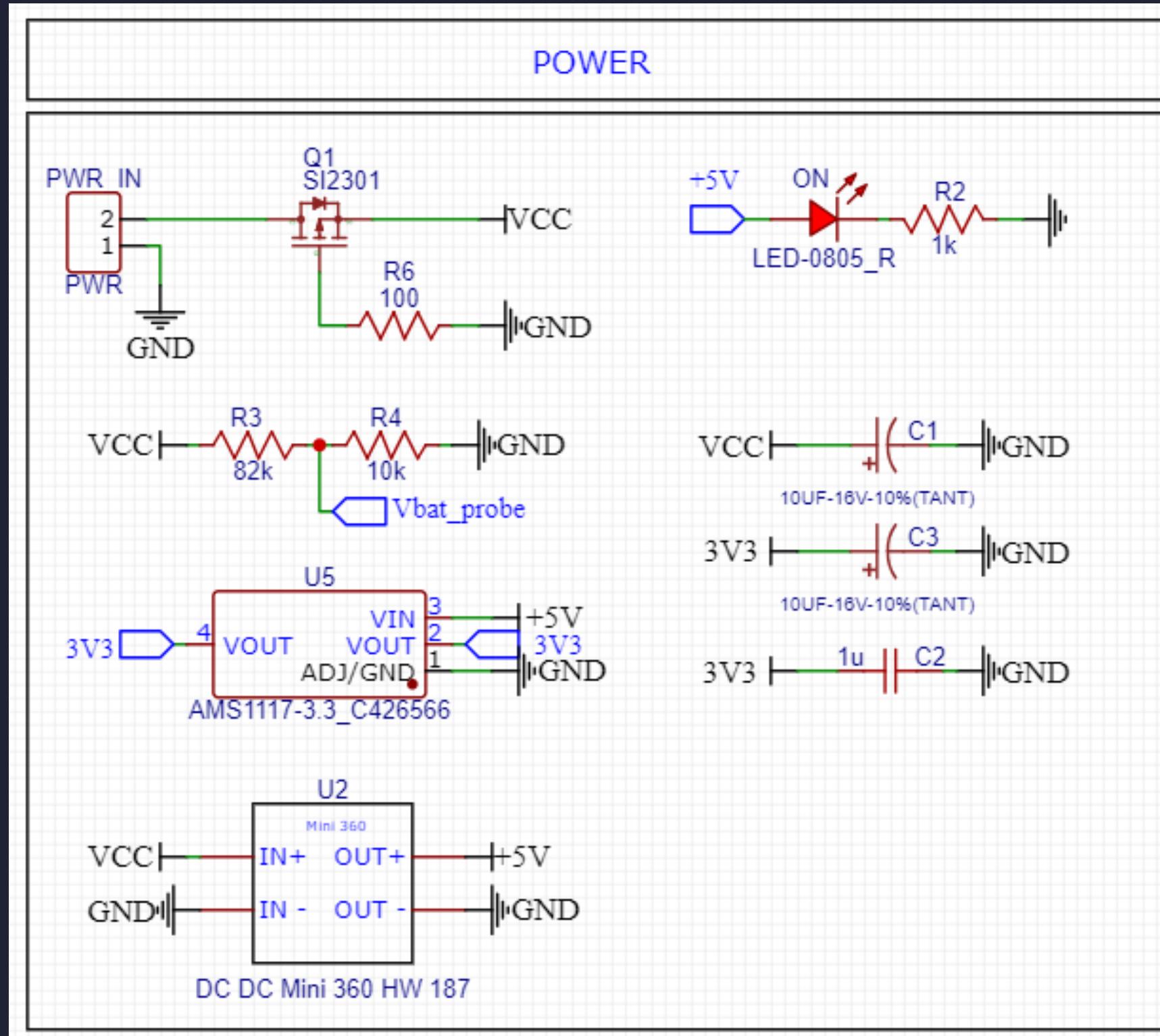
- Leve e pequena
- Componentes SMD (Surface Mounted Device)
- PCI fabricada.
- Montagem manual usando uma estação de solda e retrabalho.
- principais módulos e componentes:
  - ESP12 (ESP8266)
  - DRV8833
  - AMS1117 – 3V3
  - Conversor buck de tensão CC-CC MP2307



# Sistema Eletrônico - Abordagem 1, PCI



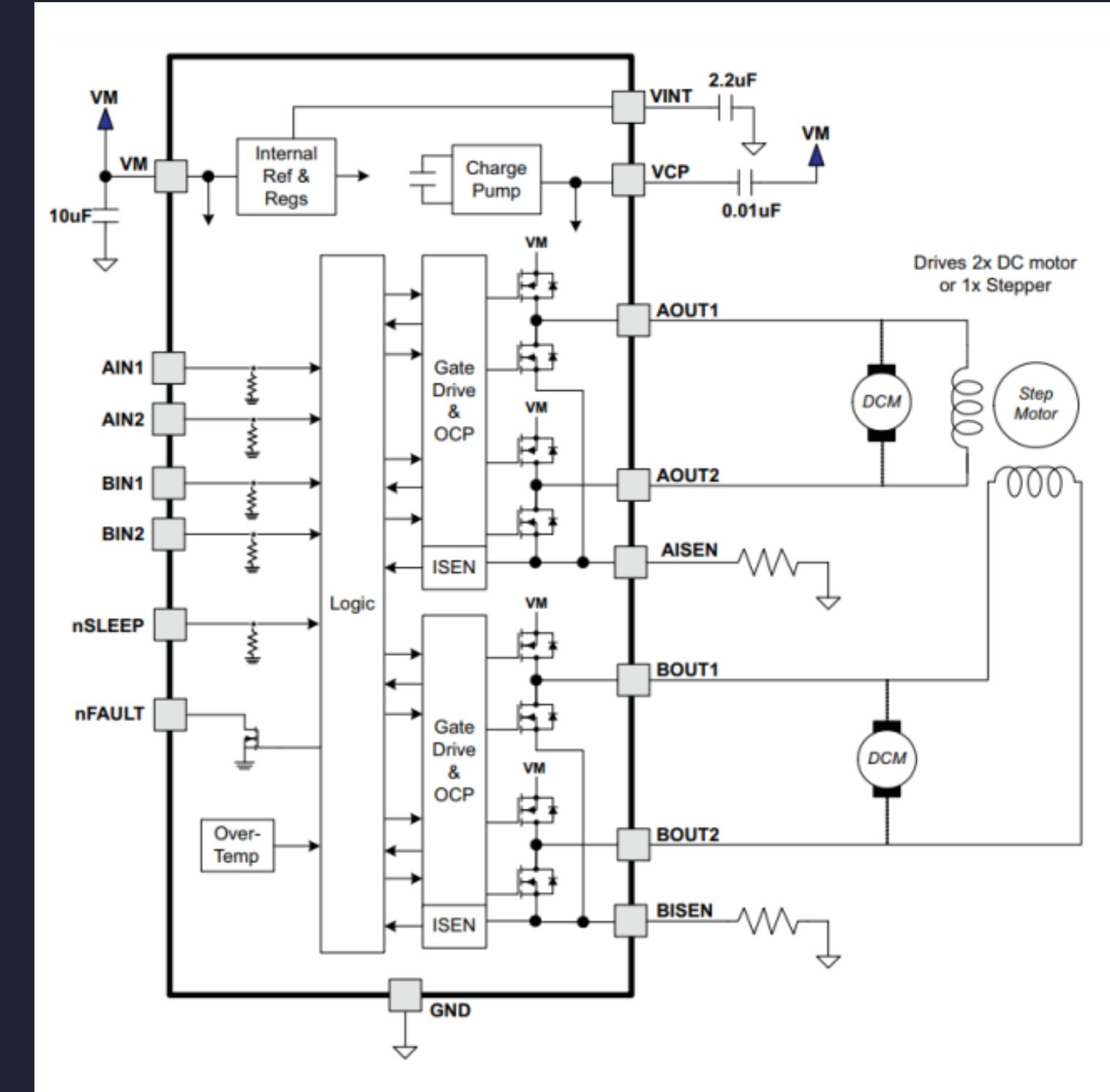
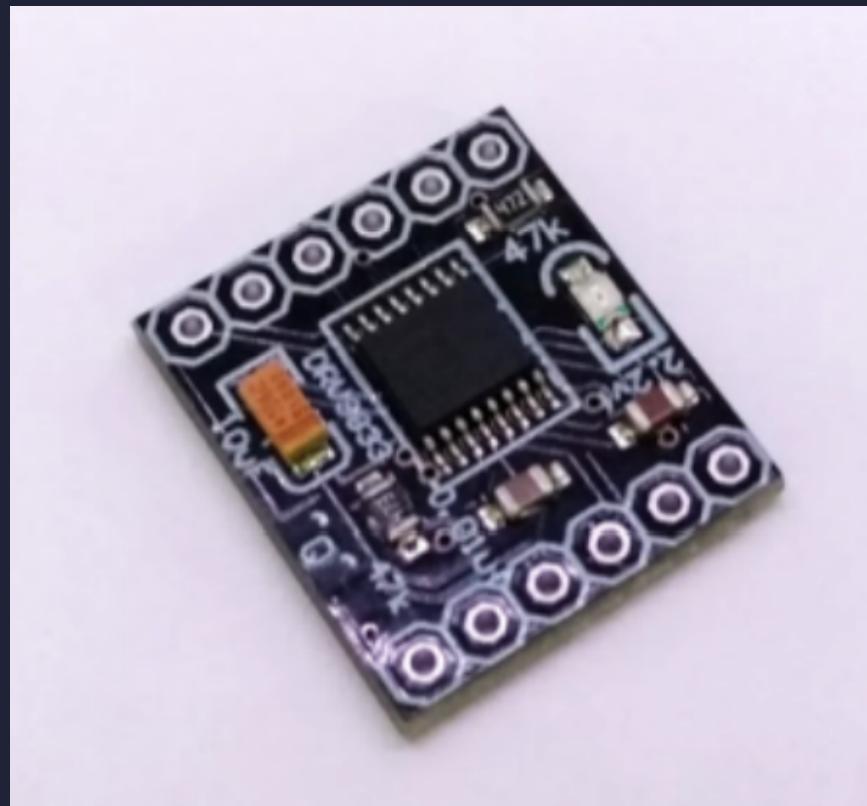
# Sistema Eletrônico - Abordagem 1, circuito de alimentação



- proteção contra polaridade reversa
- medição de tensão
- regulação para 5V e 3,3V

# Sistema Eletrônico – Abordagem 1, ponte H

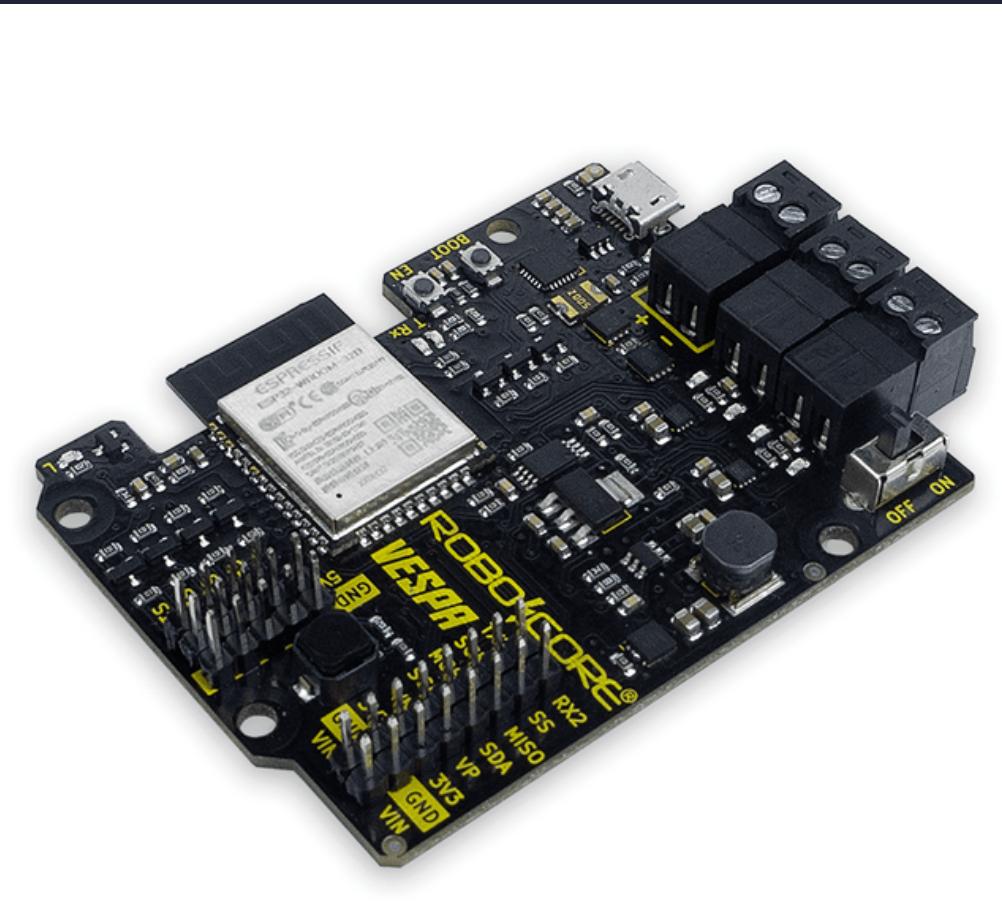
- Tensão de alimentação 2,7V até 10,8V;
- Baixo  $R_s(on)$ , 360mΩ no total (HS + LS);
- Corrente de pico de 2A por motor e 1,5A RMS;
- Proteção de contra sobre corrente e temperatura;
- Pinos de controle compatíveis com 3,3V ou 5V



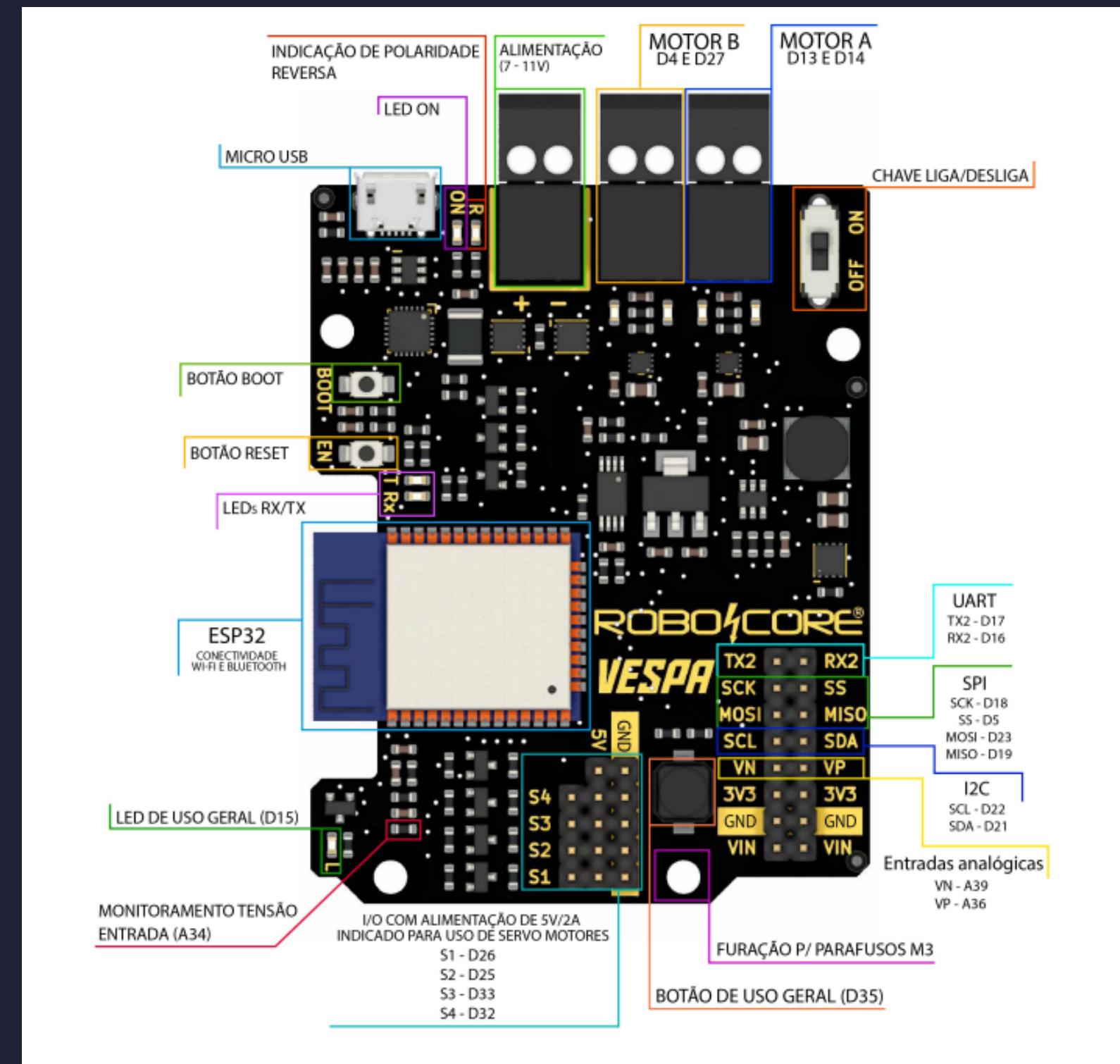
# Sistema Eletrônico – Abordagem 2

A placa Vespa é fabricada pela empresa Robocore, ela utiliza o microcontrolador ESP32 e possui formato compatível com a placa de desenvolvimento Arduino UNO Rev3. As principais características:

- possui duas pontes H DRV8837 (uma para cada motor) capaz de fornecer até 800mA contínuos
- Fonte DC-DC de alimentação capaz de fornecer até 2,5A e receber até 11V
- Capaz de medir a tensão da bateria
- Proteção contra polaridade reversa
- Pinos livres que permitem a conexão de uma IMU
- Microcontrolador ESP32



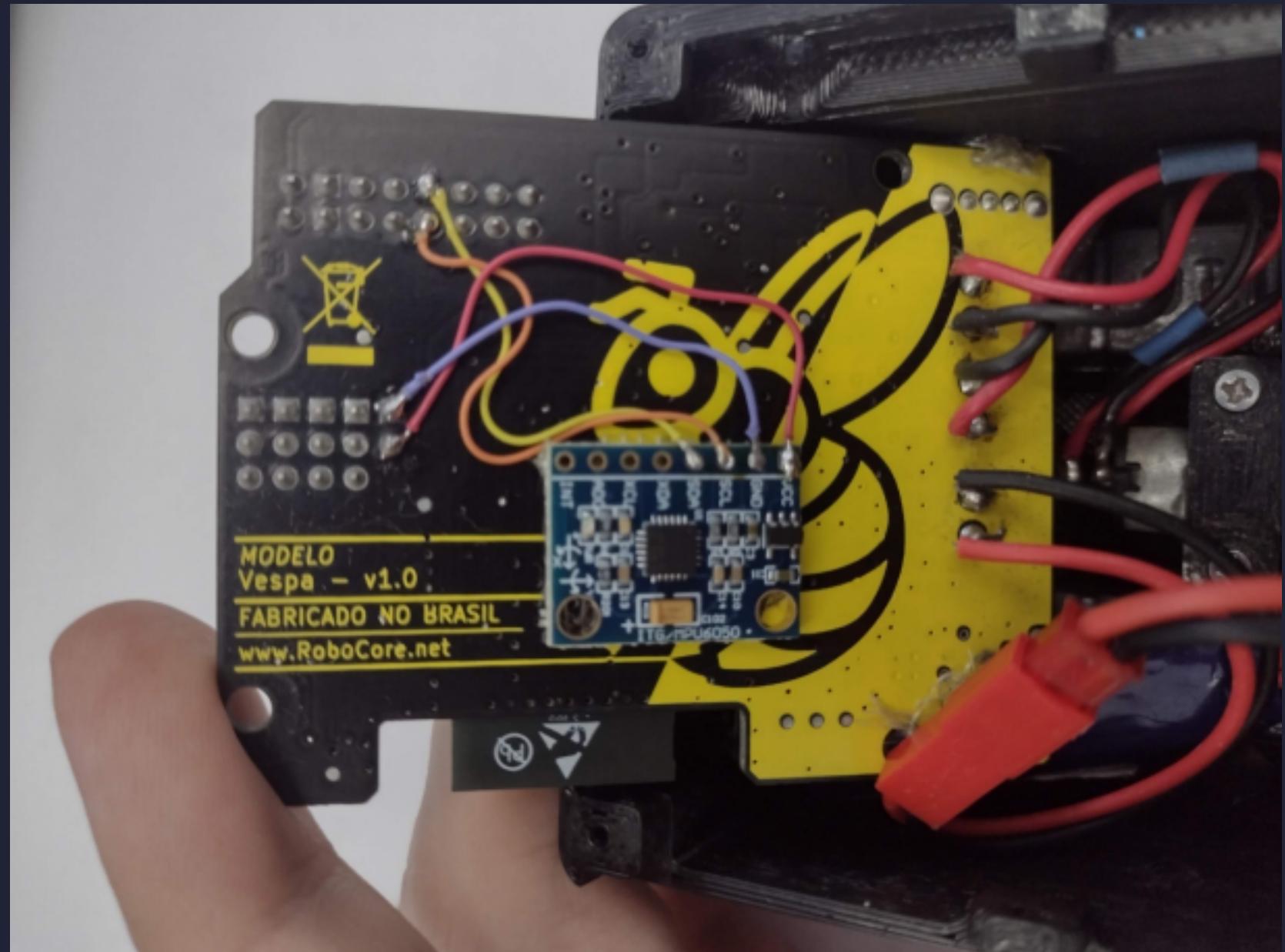
# Sistema Eletrônico - Abordagem 2



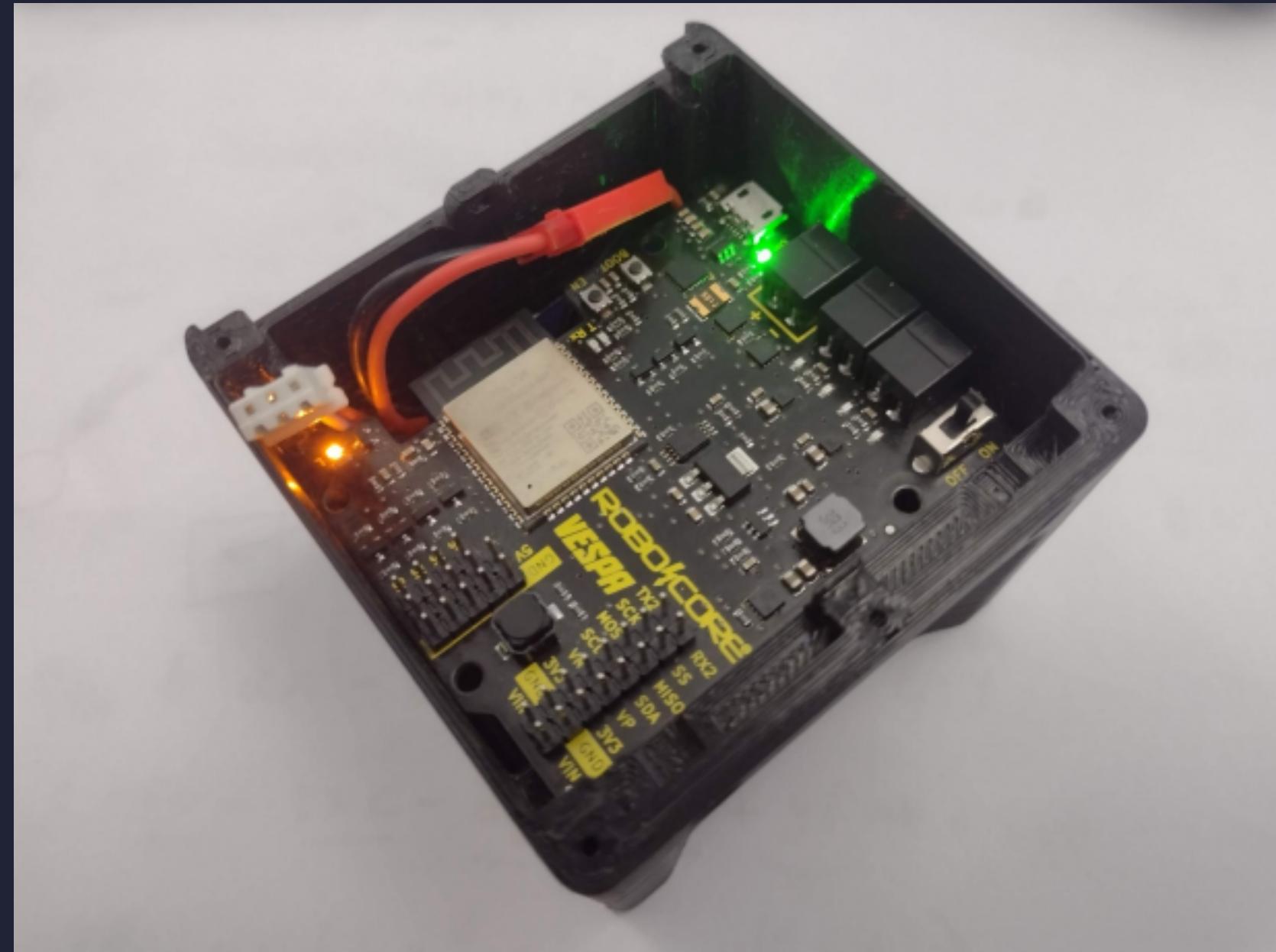
**Pinagem da Vespa**

# Sistema Eletrônico – Abordagem 2

A Vespa não possui IMU, por isso foi fixada com fita dupla-face a IMU MPU6050, que foi conectada com pequenos fios soldados diretamente nos pinos.



# Sistema Eletrônico – Abordagem 2



montagem final

# Alimentação dos Jogadores

A bateria é dimensionada para durar os dois tempos de jogo com folga e atender a velocidade máxima esperada para o robô.

- Bateria de LiPo
- 2S, possui duas células em série com tensão nominal de 3,7V, que em carga completa atinge 8,4V o conjunto
- Capacidade de 300mAh

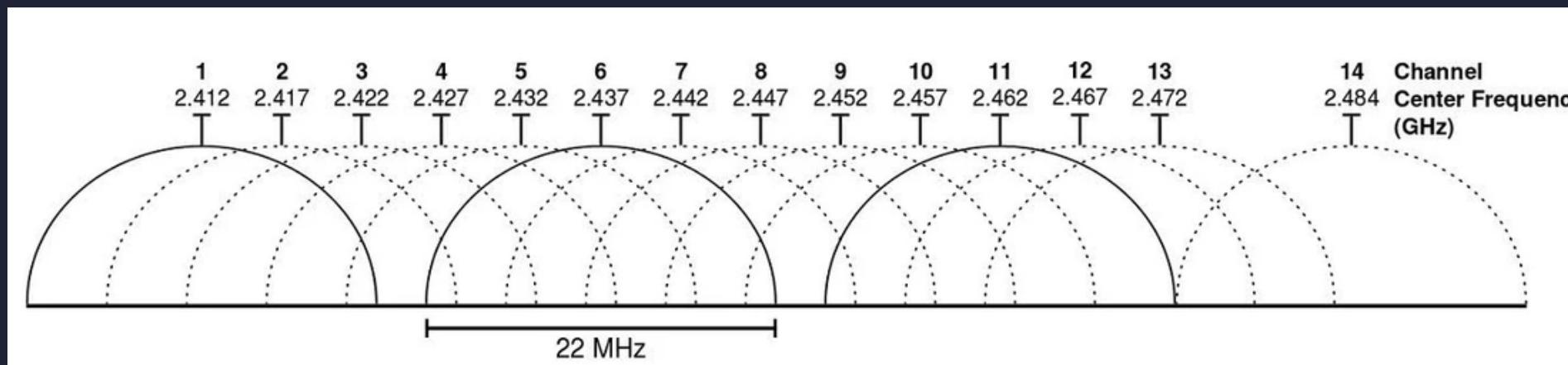


$$C = I_{medio} \Delta t$$

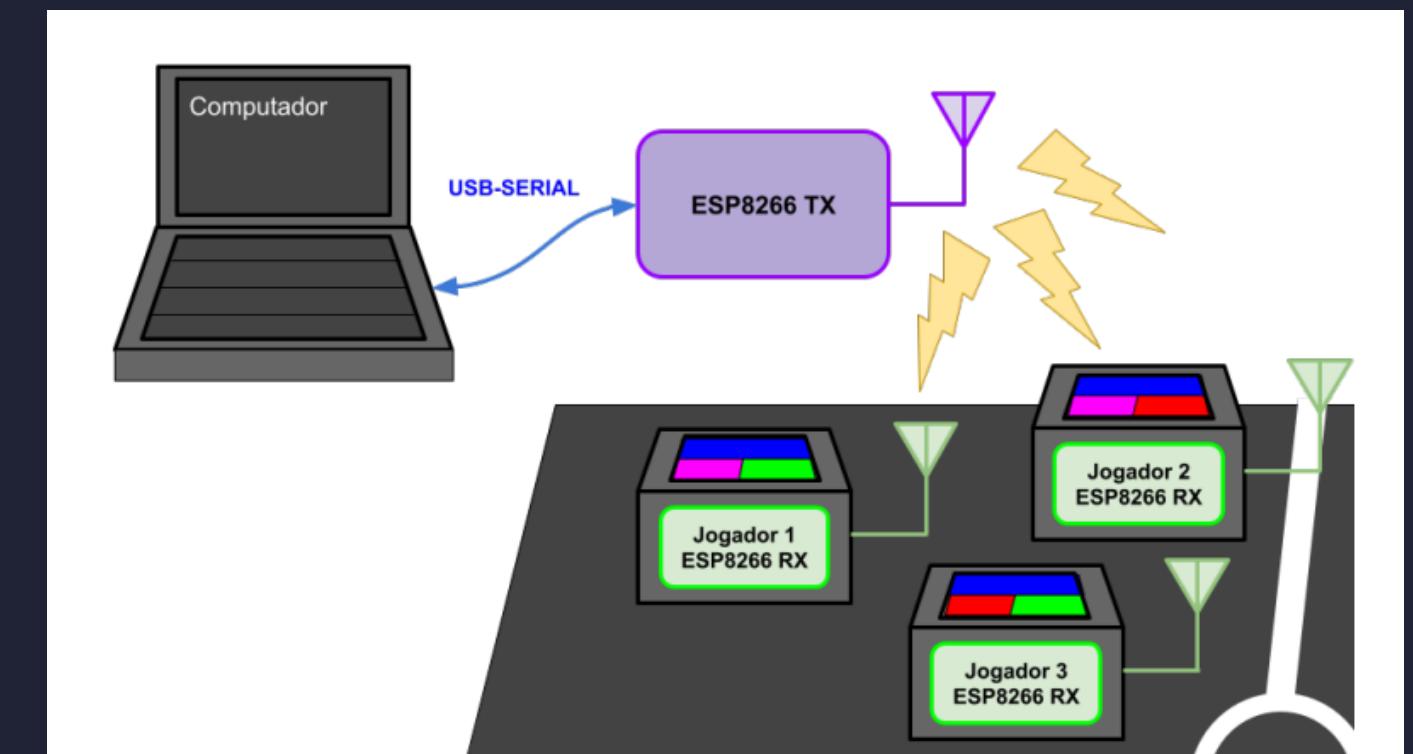
# Comunicação entre o computador e os robôs

É utilizado o protocolo de comunicação ESPNOW, este protocolo foi desenvolvido pela empresa Espressif Systems para seus microcontroladores. Ele permite o pareamento direto entre diversos ESP8266 e ou ESP32, permitindo enviar e receber dados. Motivações para seu uso:

- Facilidade de implementação, apenas um módulo ESP32 ou ESP8266;
- Baixa latência;
- Rápido reestabelecimento de conexão em casos de desligamento do transmissor ou receptor;
- Opera em 2,4GHz, podendo operar em até 13 canais.



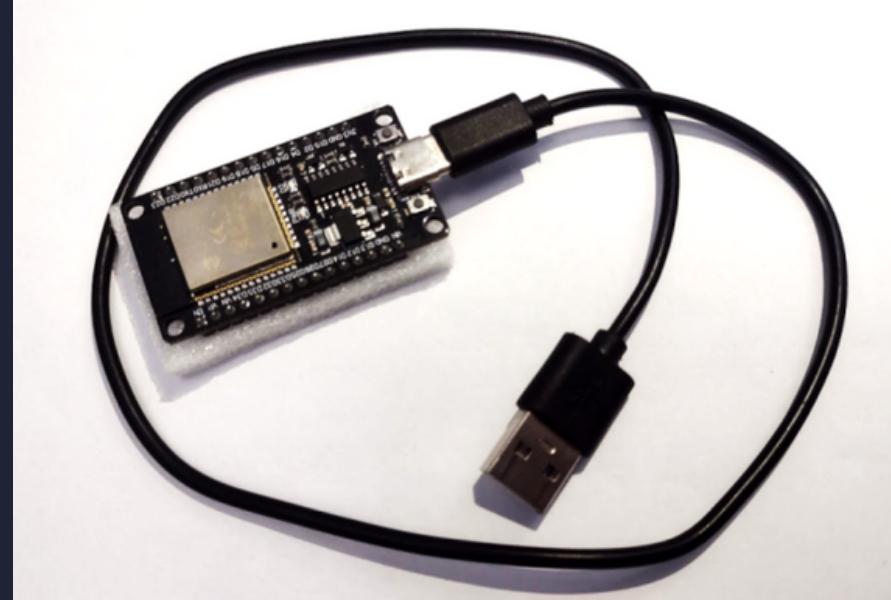
Canais 2.4GHz



# Comunicação entre o computador e os robôs

No computador é conectada uma placa de desenvolvimento com o microcontrolador ESP32, que recebe os comandos via conexão serial, em formato de texto, e repassa para os robôs.

## Principais comandos



comandos	funções
reset	reinicia o robô
bot.stop	freia o robô
bot.move <vel. esq.> <vel. dir.>	aplica as velocidade em cada roda
pid.on <estado>	liga ou desliga o controlador PID
pid.linear_speed <vel.>	altera a velocidade tangencial, valor de PWM
pid.angular_speed <vel.>	altera o valor da velocidade angular em rad/s
pid.kp <valor>	altera ou retorna o valor de kp
pid.ki <valor>	altera ou retorna o valor de ki
pid.kd <valor>	altera ou retorna o valor de kd
pid.plot <state>	habilita os desabilita o retorno de dados do PID
bot.bip	o robô faz um alerta sonoro
bot.voltage	retorna a tensão do robô

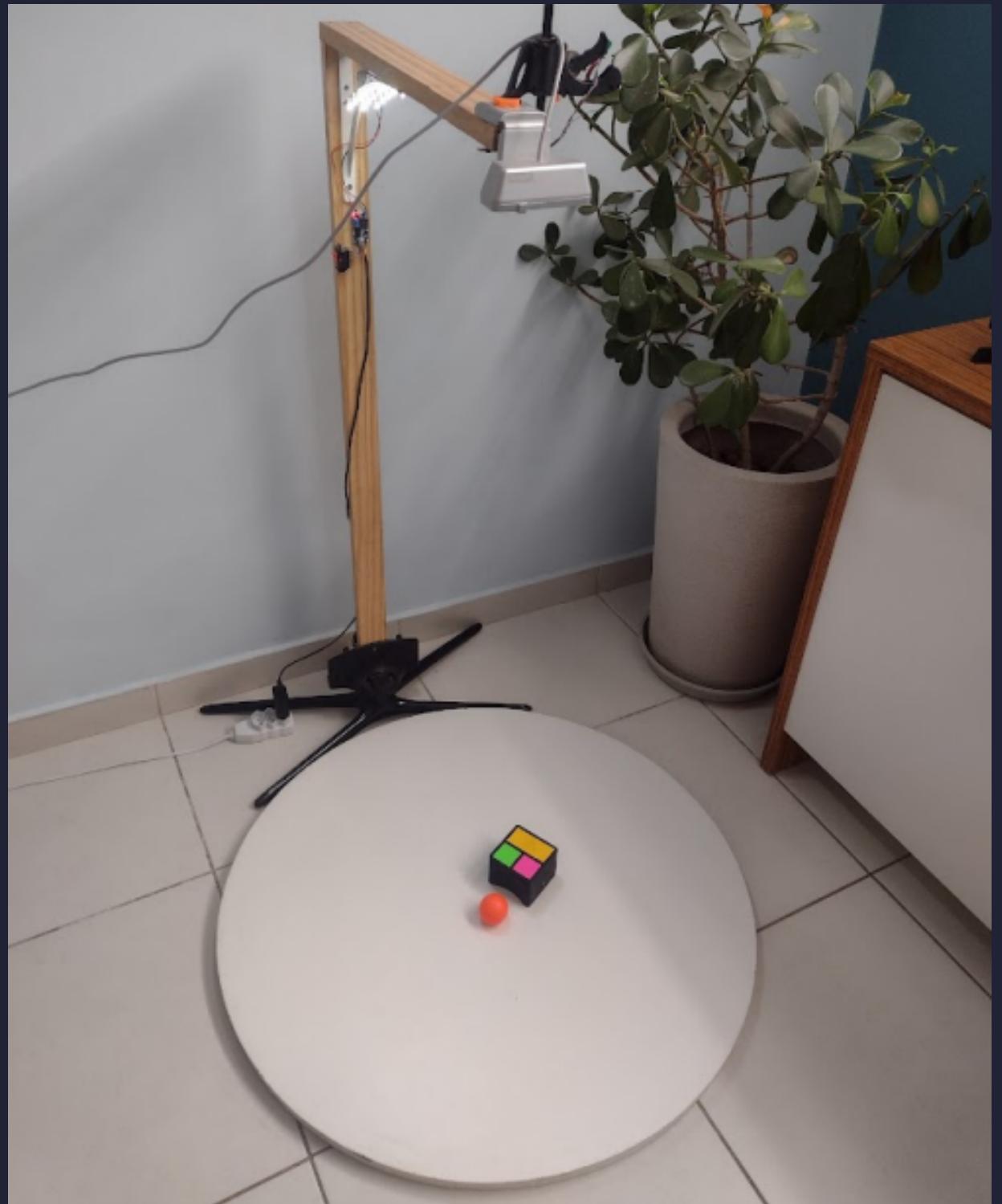
# Montagem do campo

A construção do campo foi realizada na oficina da UERJ, seguindo as regras oficiais da categoria IEEE VSSS 3x3.



# Material utilizado no sistema de visão

No sistema de visão é utilizado uma camera USB de 30 FPS fixada em um suporte com iluminação.



# Visão computacional e integração

[VOLTAR AO SUMÁRIO](#)

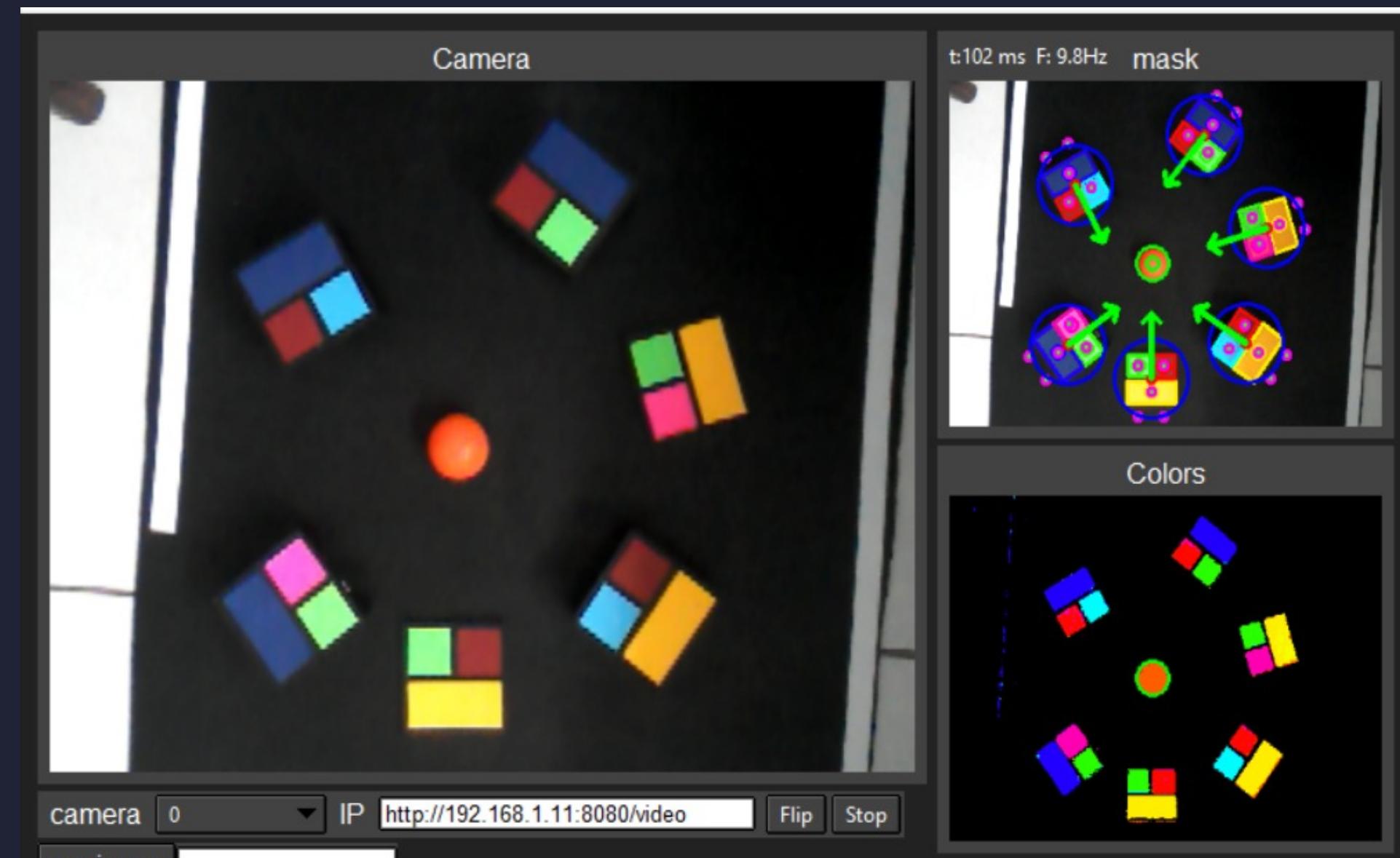


# Estrutura do código no computador



# Algoritmo de visão computacional

A visão computacional é o ramo da computação que busca extrair informações de imagens. No futebol de robôs ela informa ao algoritmo principal, a pose dos jogadores e da bola.



# Recursos computacionais

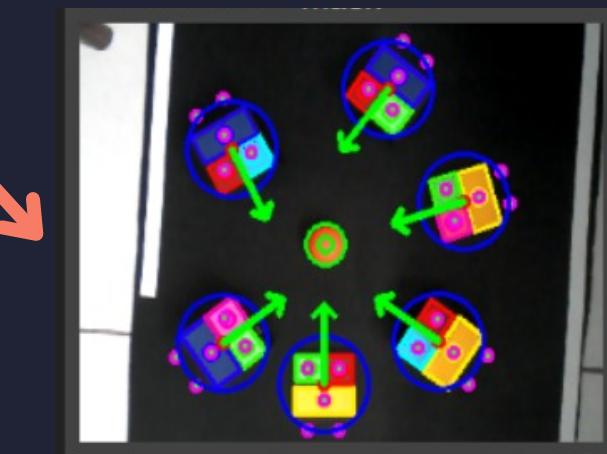
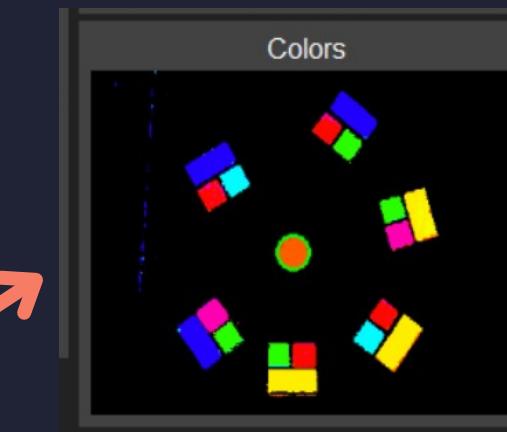
Todo o algoritmo do sistema de visão foi desenvolvido em Python, com a utilização, principalmente, de duas bibliotecas Numpy e OpenCV.

- **OpenCV** (*Open Source Computer Vision Library*): Biblioteca multi-linguagem amplamente utilizada na área de visão computacional. Ela possui as principais ferramentas para realizar o processamento de imagens;
- **Numpy**: Biblioteca de computação numérica para python, que utilizada pacotes compilados em c e c++, os quais proporcionam melhor performance de processamento.



# Algoritmo de visão computacional – passos

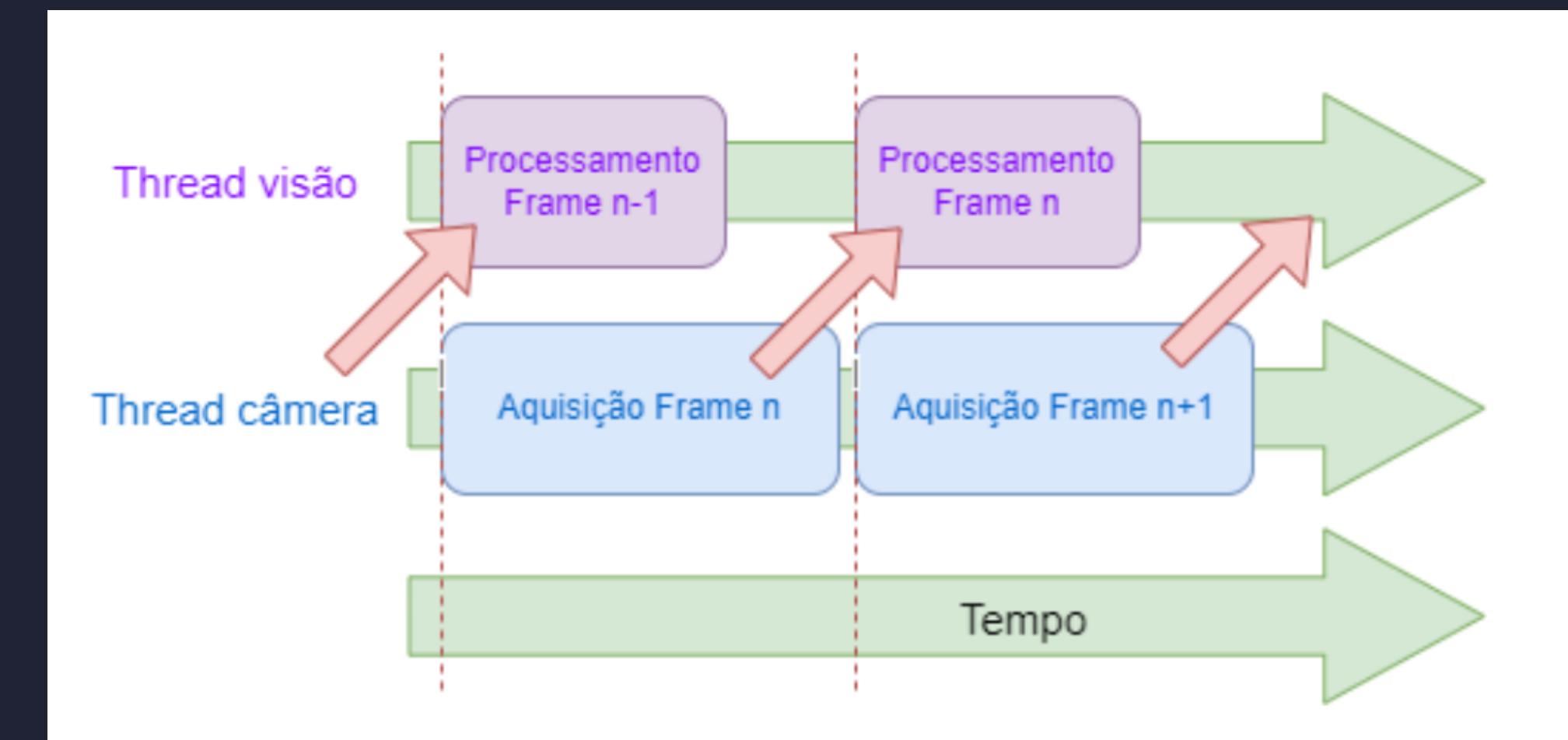
1. Captura dos quadros da câmera com maior frequência possível;
2. Detecção das cores, gera uma máscara para cada cor;
3. Detecção dos contornos das entidades;
4. Analise das informações dos contornos e das máscaras para identificar as etiquetas dos robôs e a bola
5. Ao final as informações são encapsuladas em um dicionário de python.



```
ball_detection = {  
    'ok': False,  
    'pos': 0,  
    'dimension': 0  
}  
  
bot_detection = {  
    'id': 0,  
    'pos': 0,  
    'orientation': 0,  
    'dimension': 0,  
    'vector': 0,  
    'colors': ['orange', 'orange']  
}  
  
game = {  
    'ball': ball_detection,  
    'team_blue': {},  
    'team_yellow': {}  
}
```

# Algoritmo de visão computacional – passos

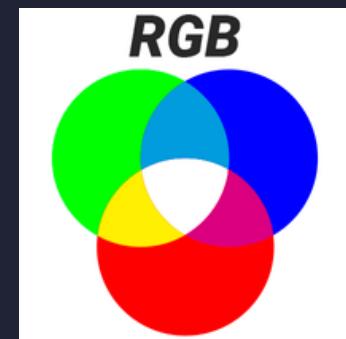
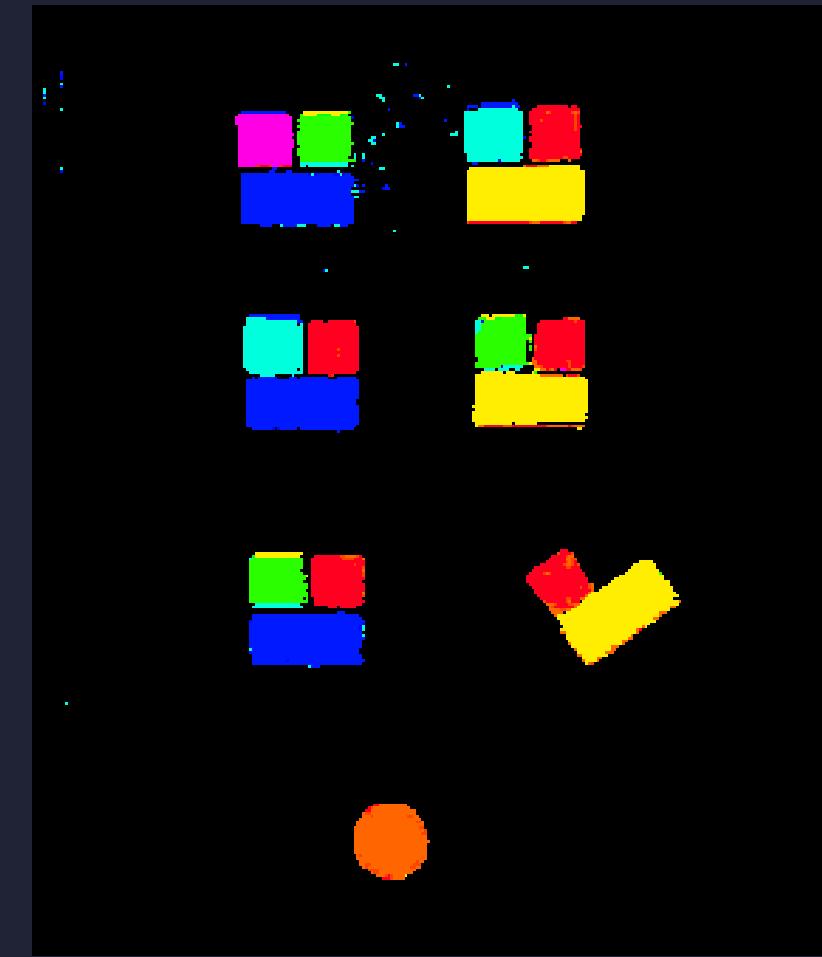
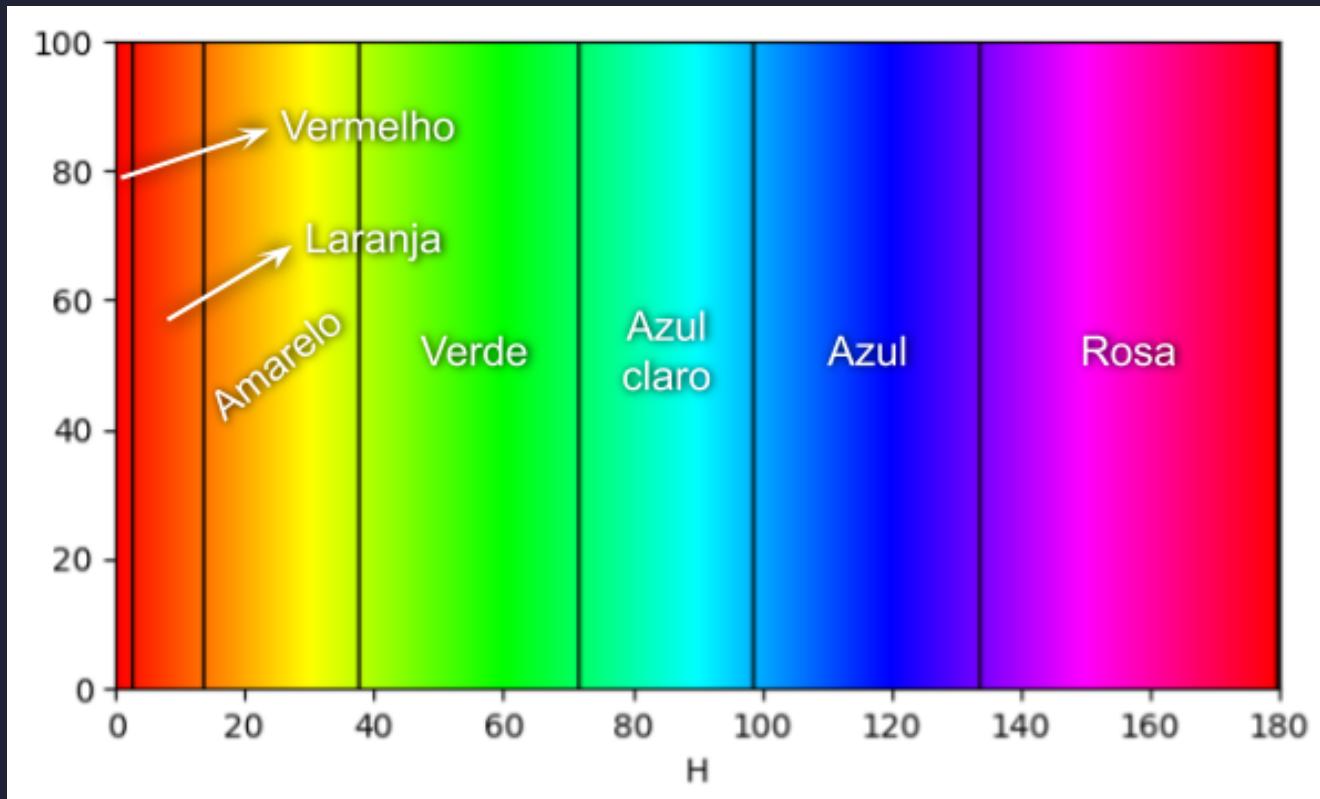
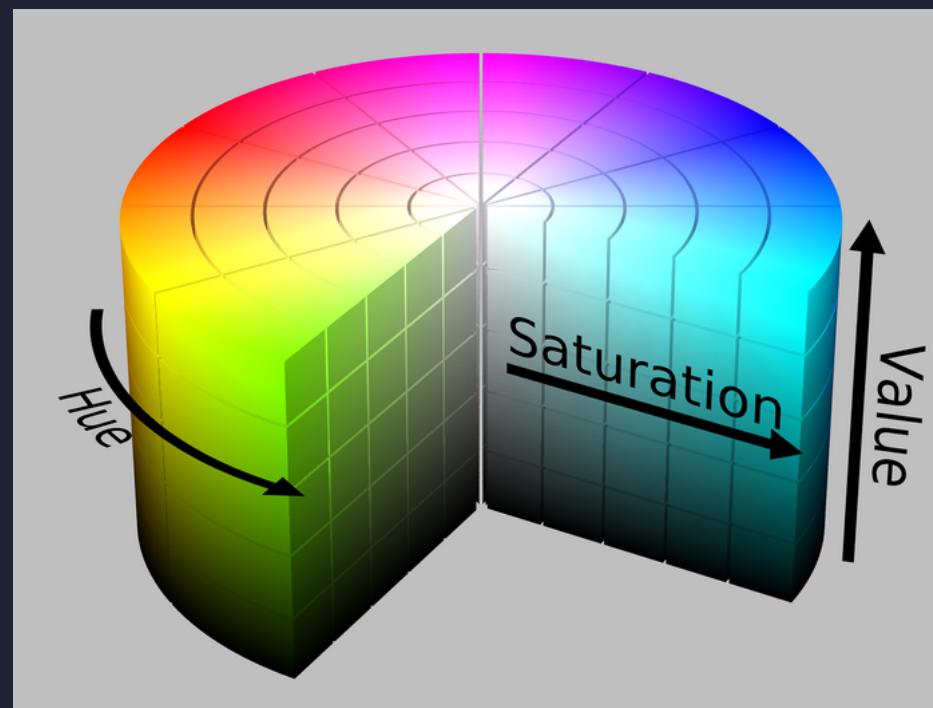
**Captura dos quadros da câmera com maior frequência possível**



# Algoritmo de visão computacional

**Detecção das cores, gera uma máscara para cada cor**

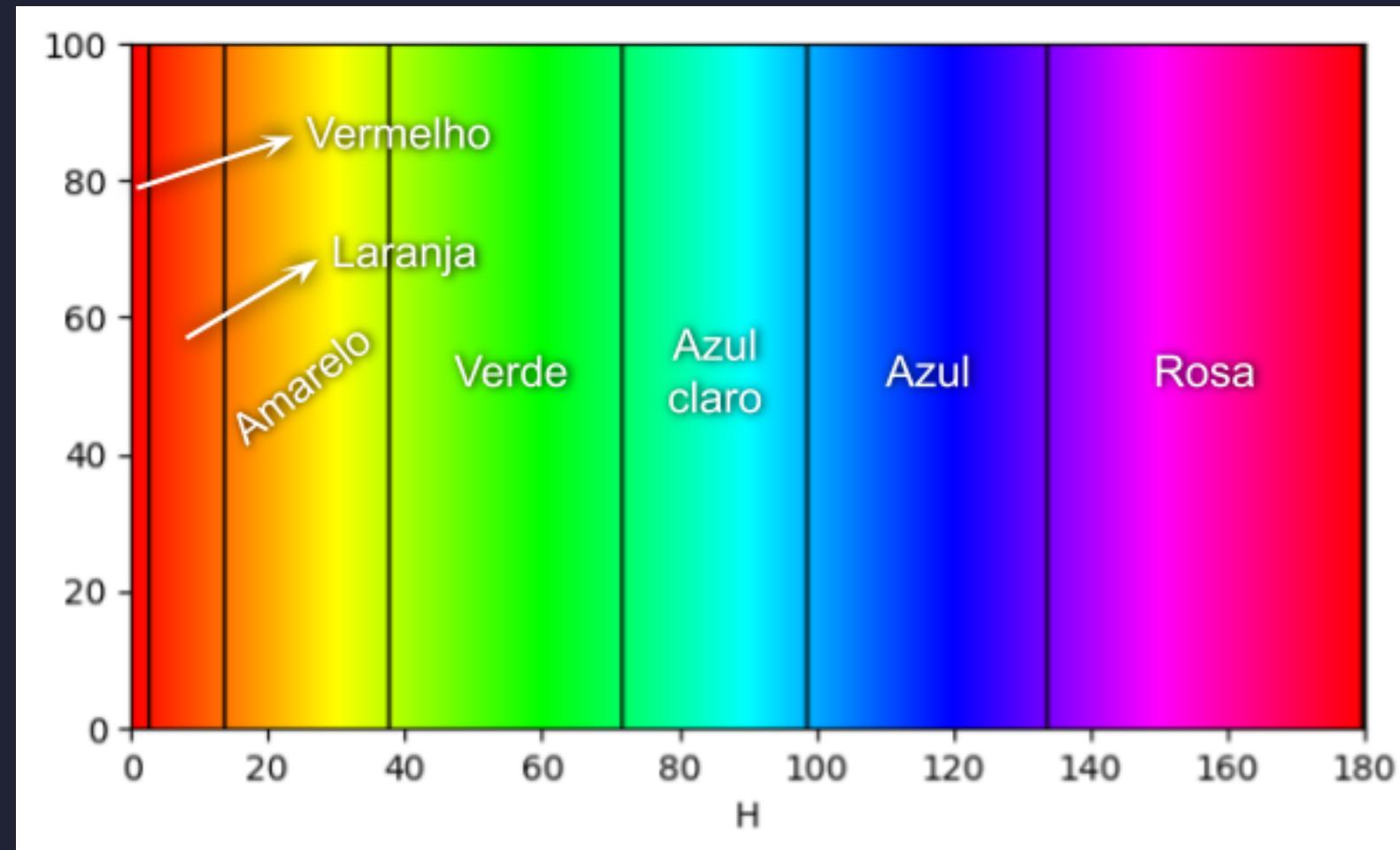
Espaço de cores HSV



$$(H_{min}, S_{min}, V_{min}) \text{ e } (H_{max}, S_{max}, V_{max})$$

# Algoritmo de visão computacional

## Detecção das cores

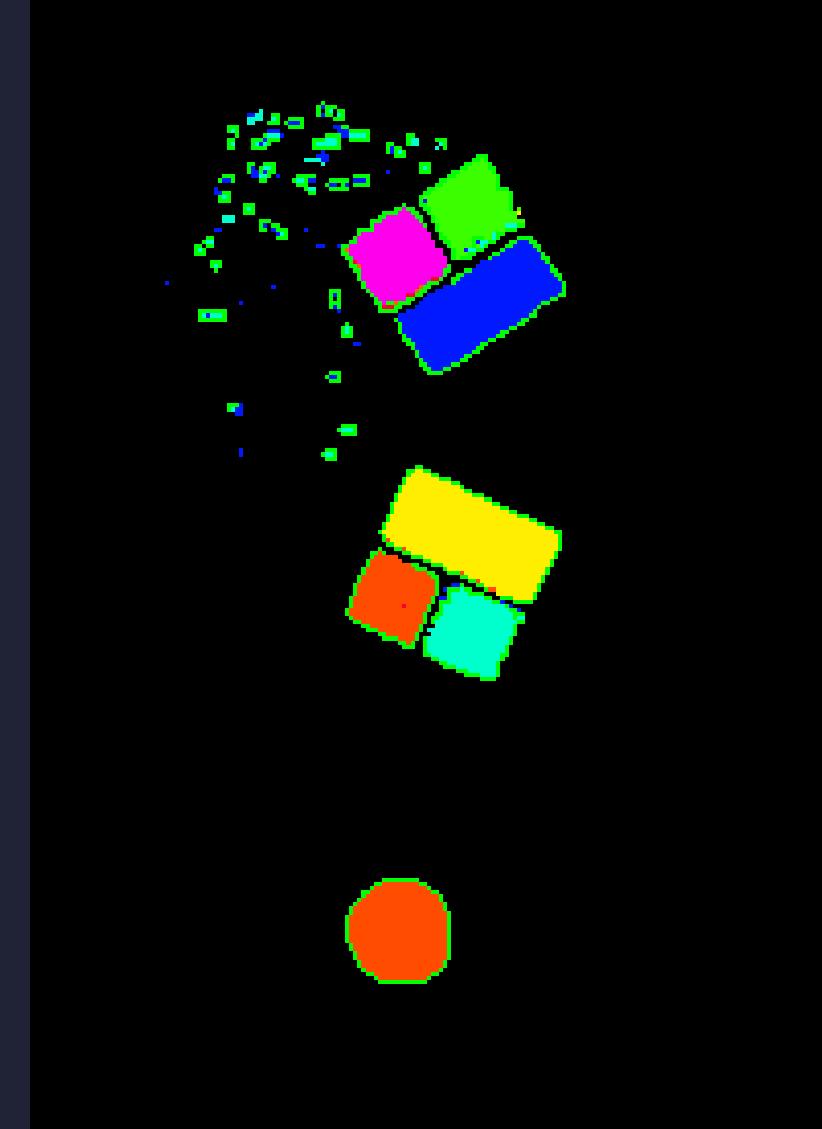
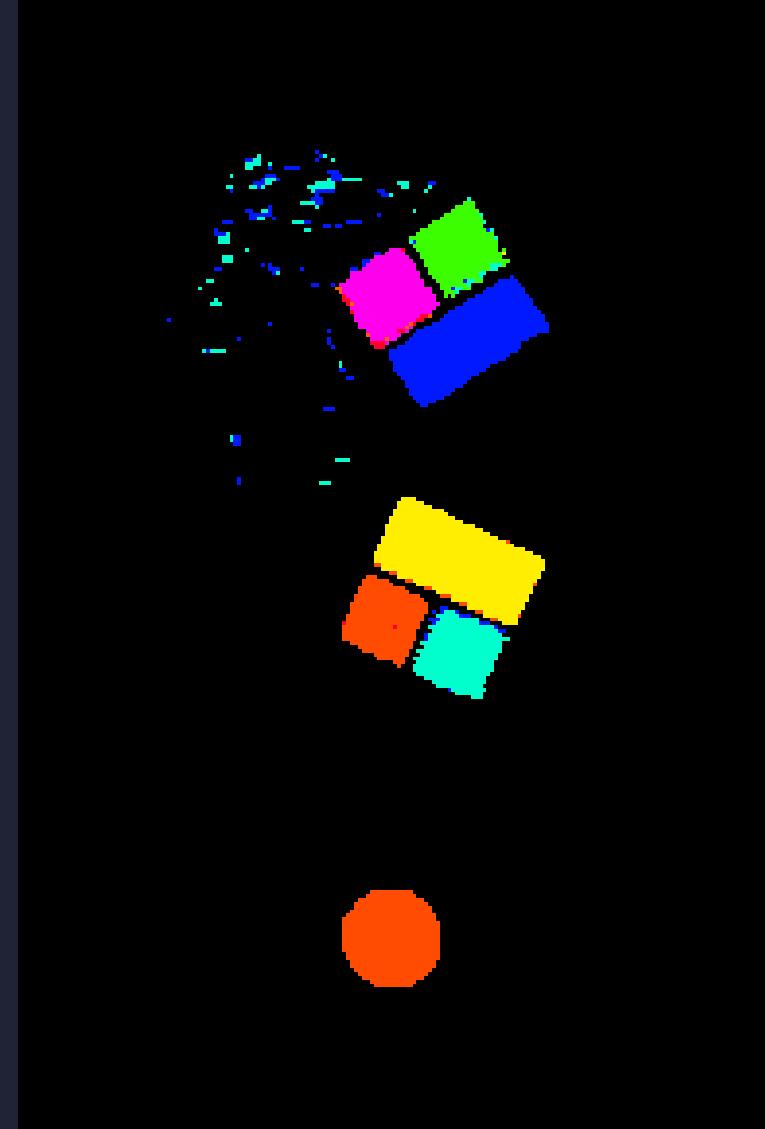


$$(H_{min}, S_{min}, V_{min}) \text{ e } (H_{max}, S_{max}, V_{max})$$

# Algoritmo de visão computacional

## Detecção dos contornos das entidades

- Filtragem dos contornos com pouca área.



*findContours*

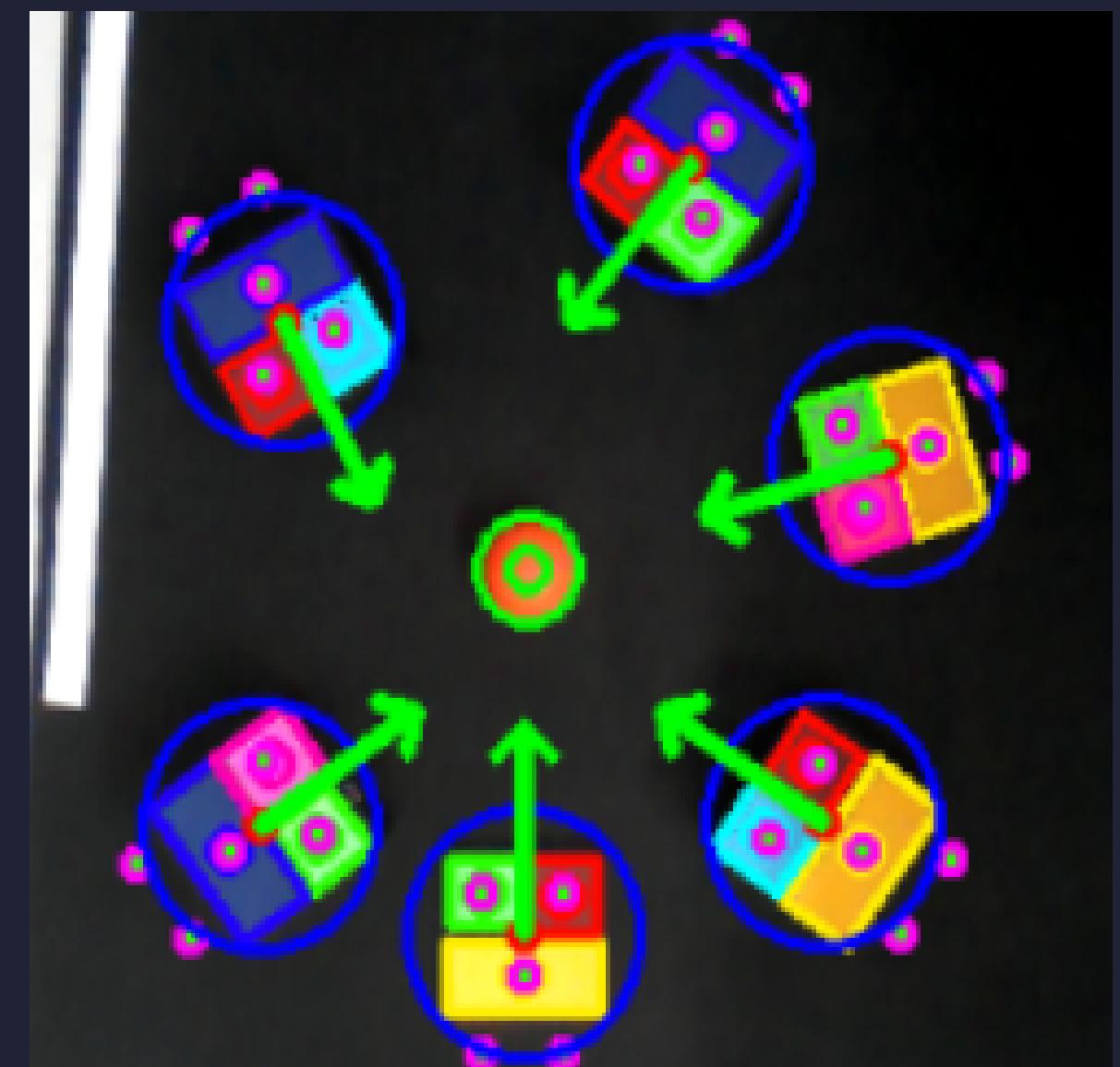
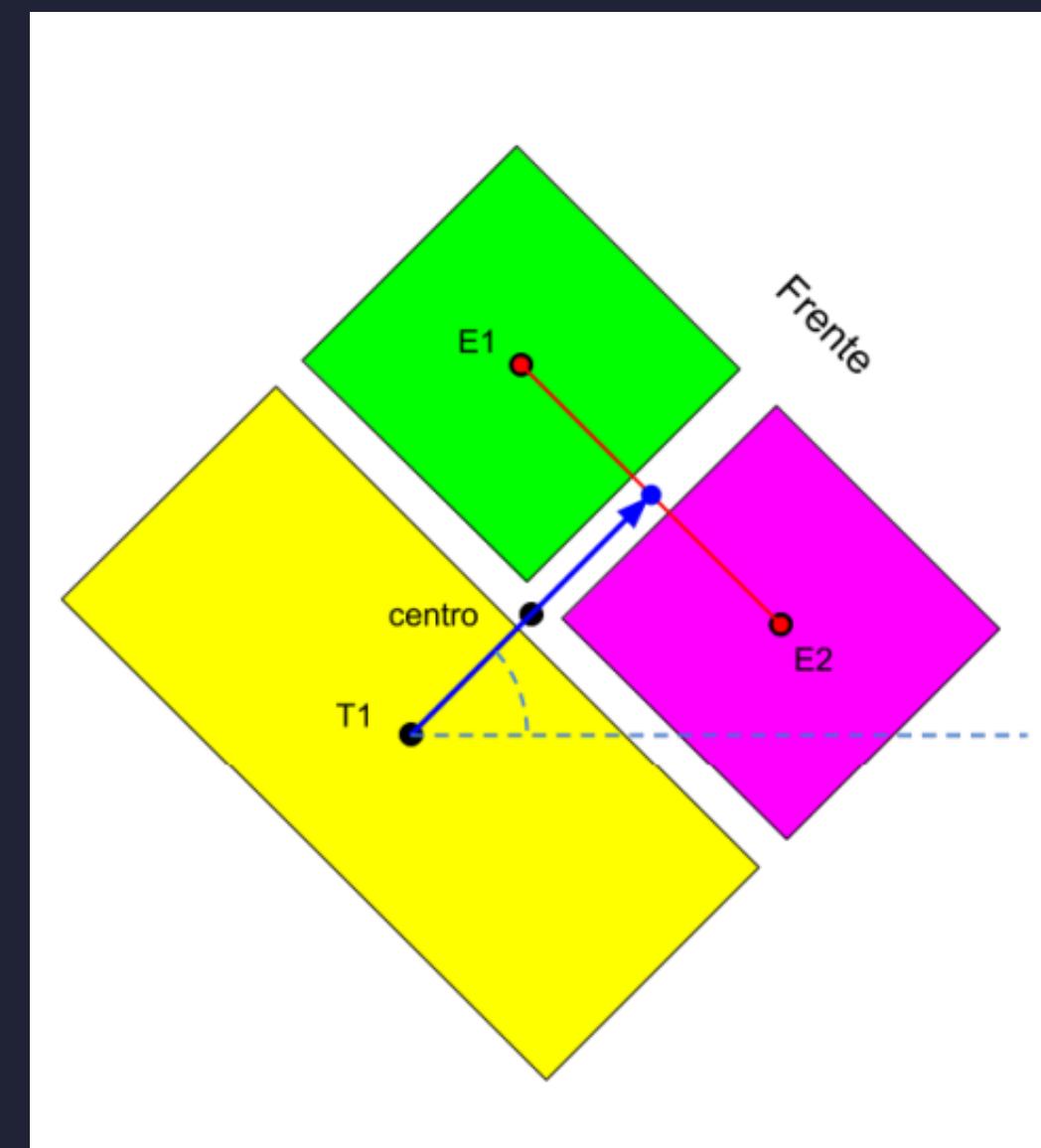
# Algoritmo de visão computacional – passos

**Analise das informações dos contornos e das máscaras para identificar as etiquetas dos robôs e a bola**

As figuras são aproximadas por retangulos e circunferencias

*minAreaRect*

*minEnclosingCircle,*



# Algoritmo de visão computacional – passos

Ao final as informações são encapsuladas em um dicionário de python

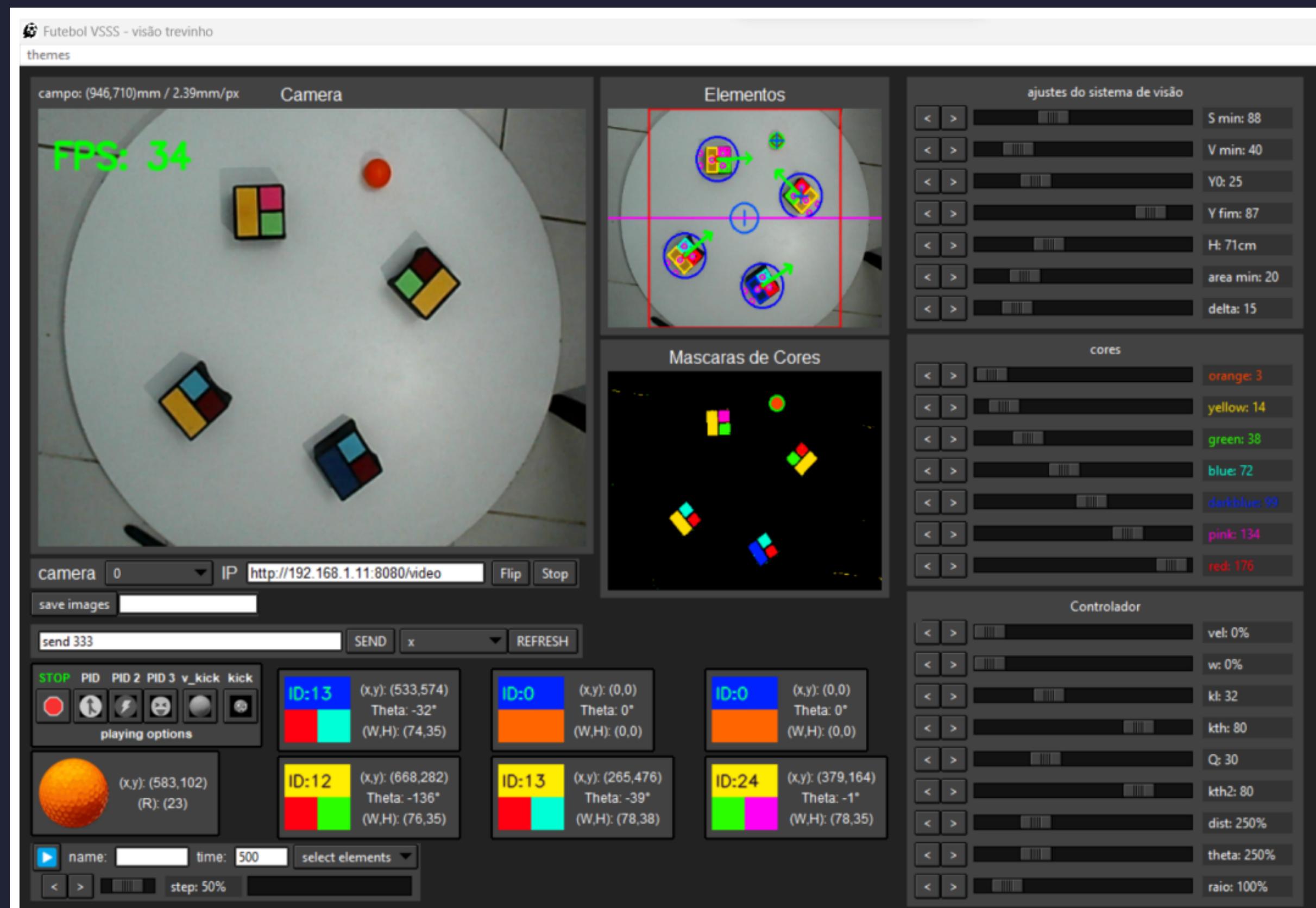
```
ball_detection = {  
    'ok': False,  
    'pos': 0,  
    'dimension': 0  
}
```

```
bot_detection = {  
    'id': 0,  
    'pos': 0,  
    'orientation': 0,  
    'dimension': 0,  
    'vector': 0,  
    'colors': ['orange', 'orange']  
}
```

```
game = {  
    'ball': ball_detection,  
    'team_blue': {},  
    'team_yellow': {}  
}
```

# Interface gráfica

1. Multiplataforma
2. Monitora o sistema de visão
3. Ajuste de parametros
4. Controle do estado de jogo
5. Testes de jogadas



# Modelagem Matemática

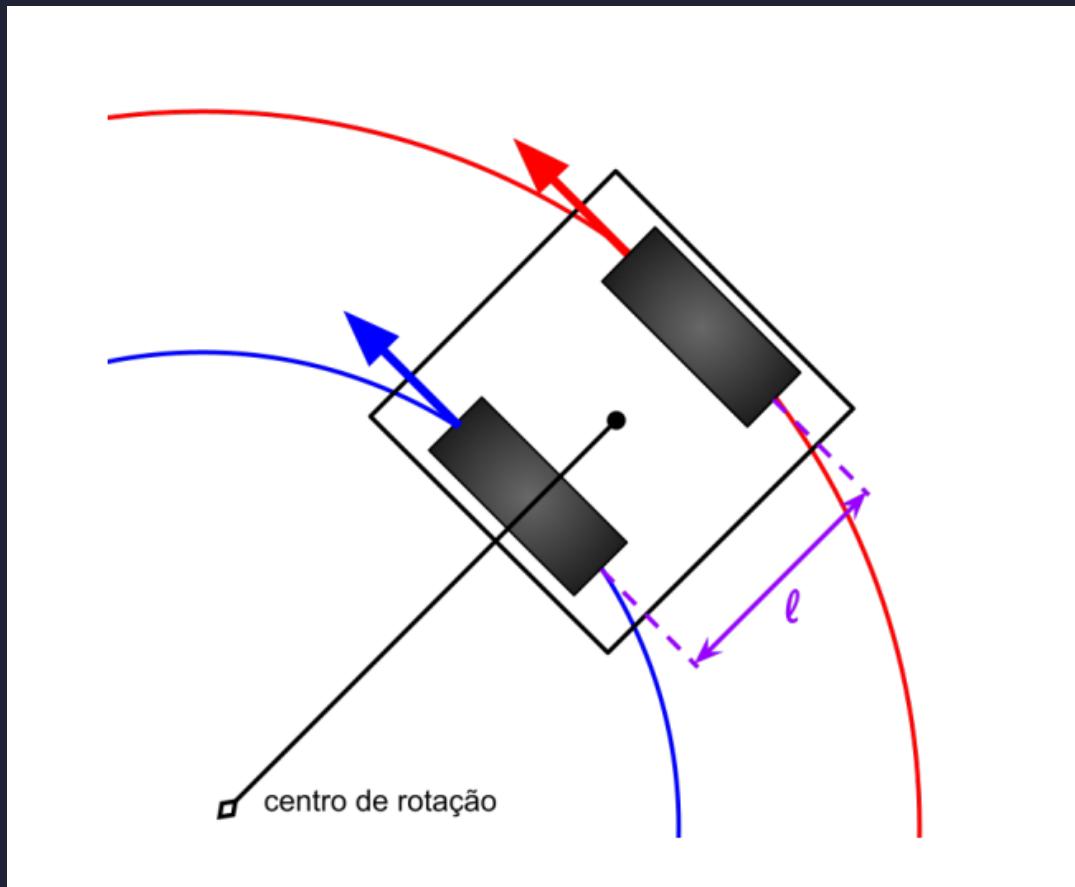
[VOLTAR AO SUMÁRIO](#)



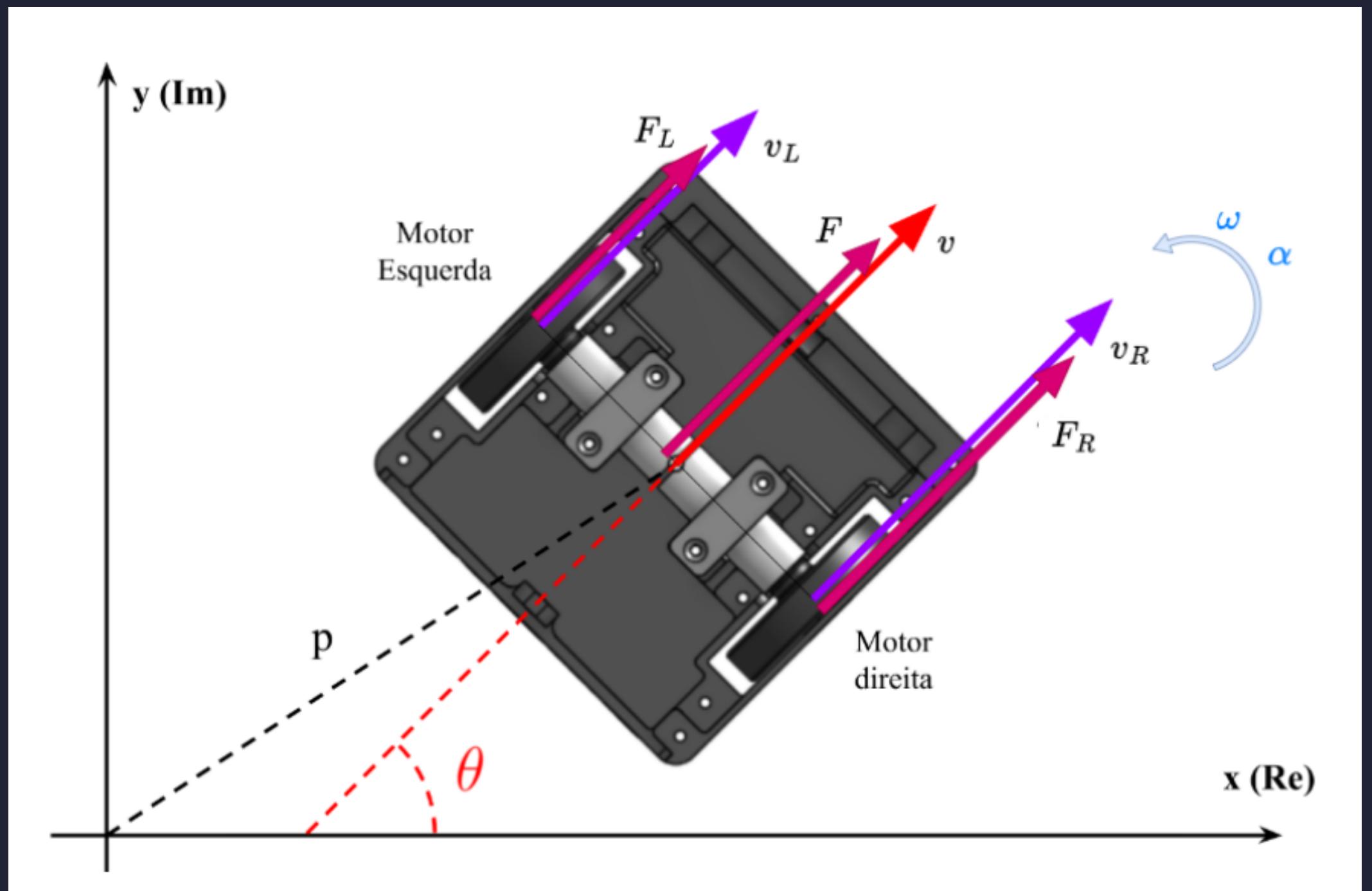
# Modelo do sistema de locomoção diferencial

A roda da direita possui velocidade  $v_r$  e a roda da esquerda possui velocidade  $v_l$ .

- Quando ambas as velocidades são iguais e positivas o robô desloca para frente em linha reta
- Quando as velocidades são diferentes ainda pode ocorrer deslocamento, porém com trajetórias circulares.
- A diferença de velocidade entre as rodas proporciona rotação.
- A média das velocidades resulta na velocidade tangencial.



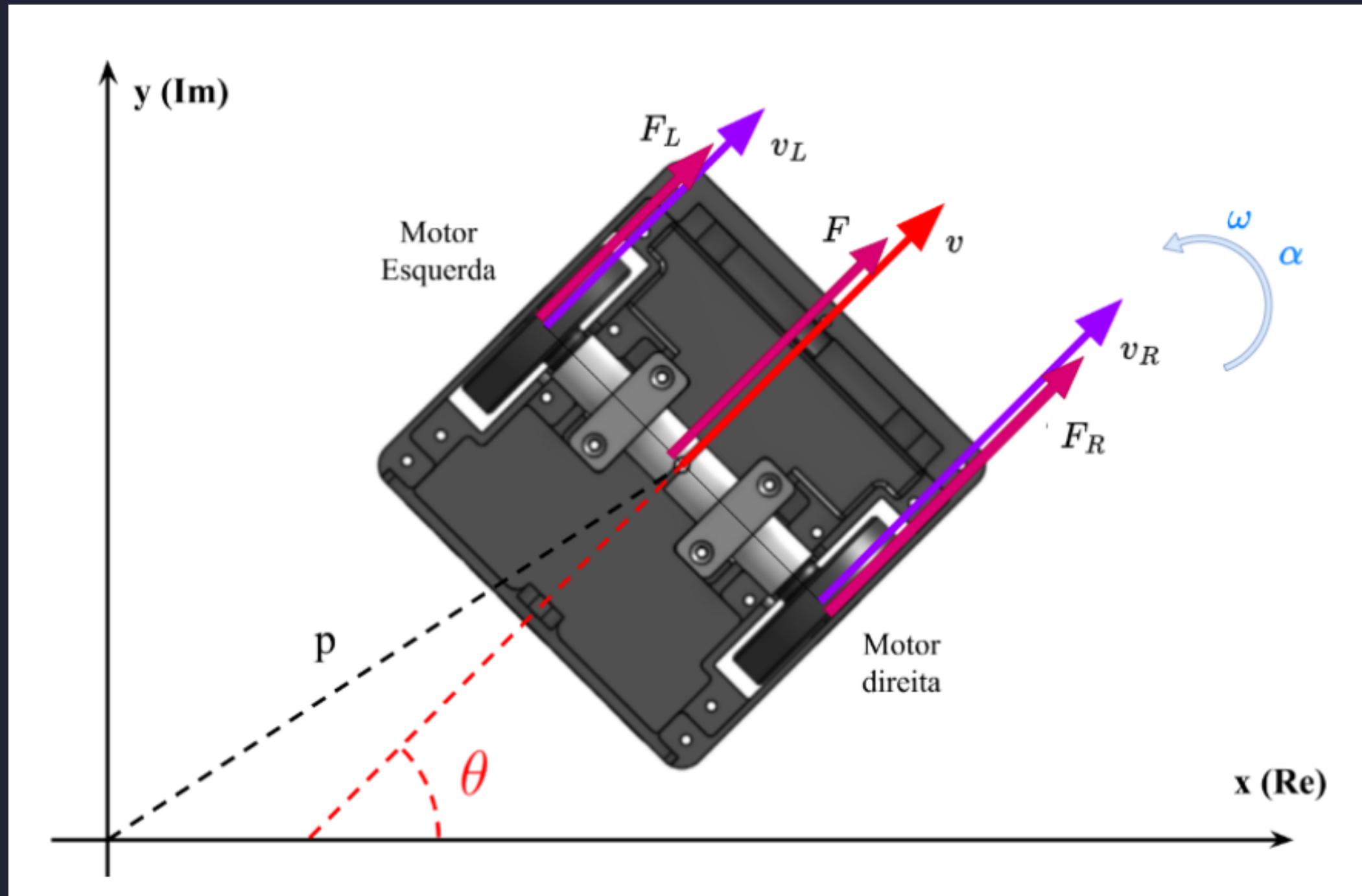
# Cinemática direta



$$v(t) = \frac{v_{dir}(t) + v_{esq}(t)}{2}$$

$$\omega(t) = \frac{v_{dir}(t) - v_{esq}(t)}{l}$$

# Cinemática Inversa



$$v_{dir}(t) = v(t) + \frac{l}{2}\omega(t)$$

$$v_{esq}(t) = v(t) - \frac{l}{2}\omega(t)$$

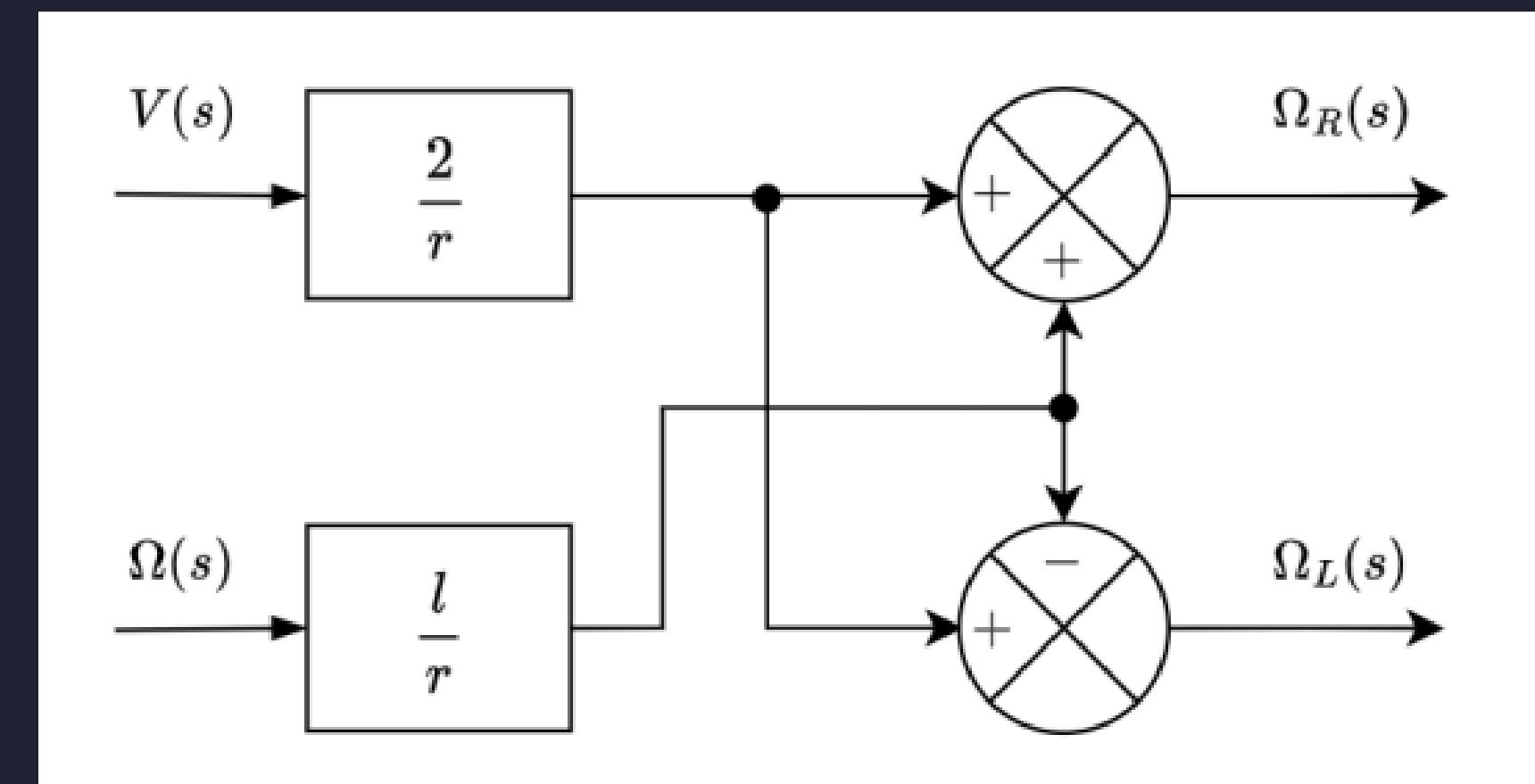
$$\Omega_{dir}(s) = \frac{1}{r}V(s) + \frac{l}{2r}\Omega(s)$$

$$\Omega_{esq}(s) = \frac{1}{r}V(s) - \frac{l}{2r}\Omega(s)$$

# Cinemática Inversa

$$\Omega_{dir}(s) = \frac{1}{r}V(s) + \frac{l}{2r}\Omega(s)$$

$$\Omega_{esq}(s) = \frac{1}{r}V(s) - \frac{l}{2r}\Omega(s)$$



Em forma de diagrama em blocos

# Modelagem dos motores

Os motores aplicados neste projeto, são pequenos motores de corrente contínua escovados de ímãs permanentes com uma micro-redução na saída. Motores CC podem ser descritos em forma de circuito ou pela s equações ao lado.

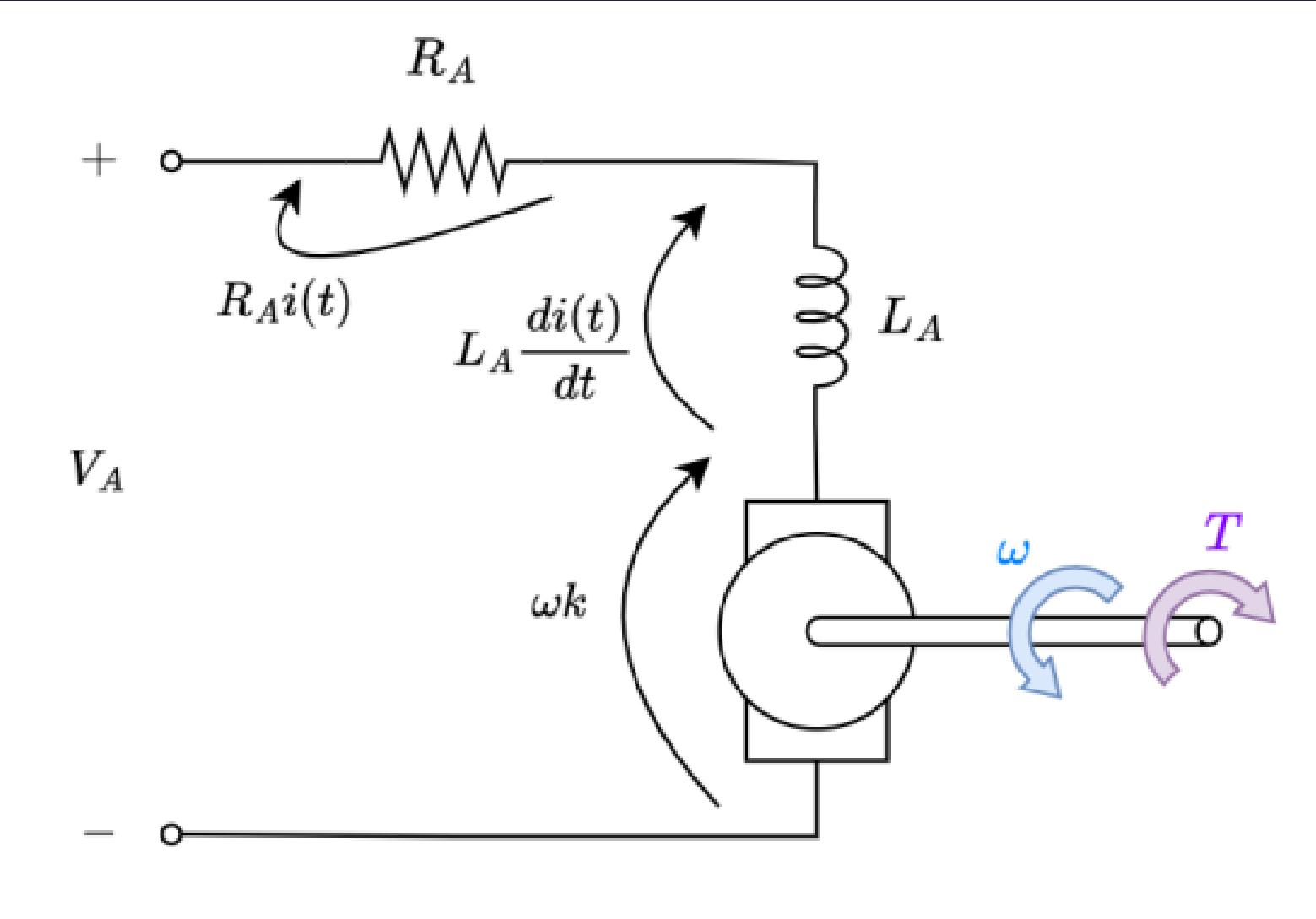
Parâmetros:

RA - perdas resistivas.

LA - Indutância de Armadura.

k - constante do motor

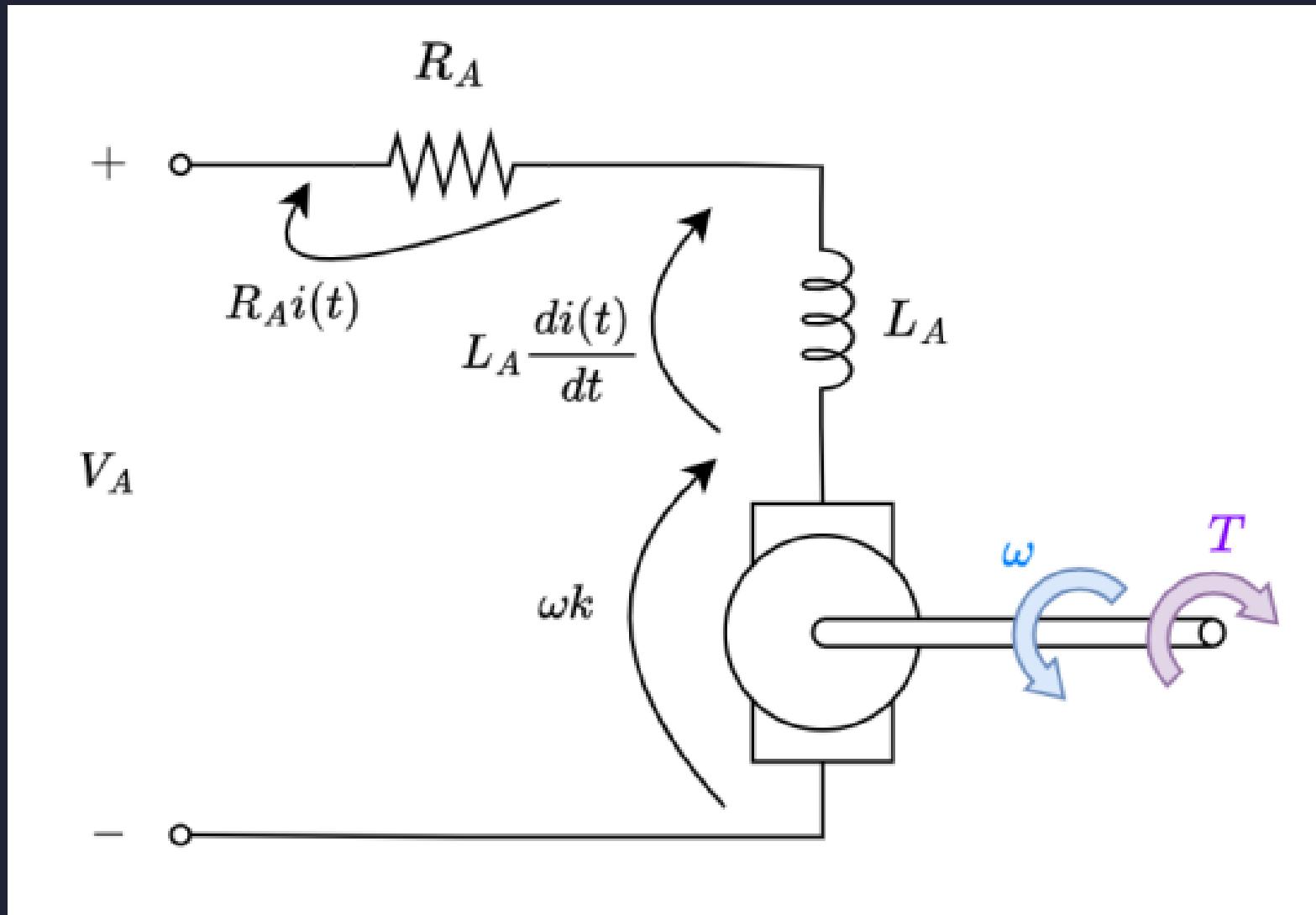
$$T(t) = ki(t)$$



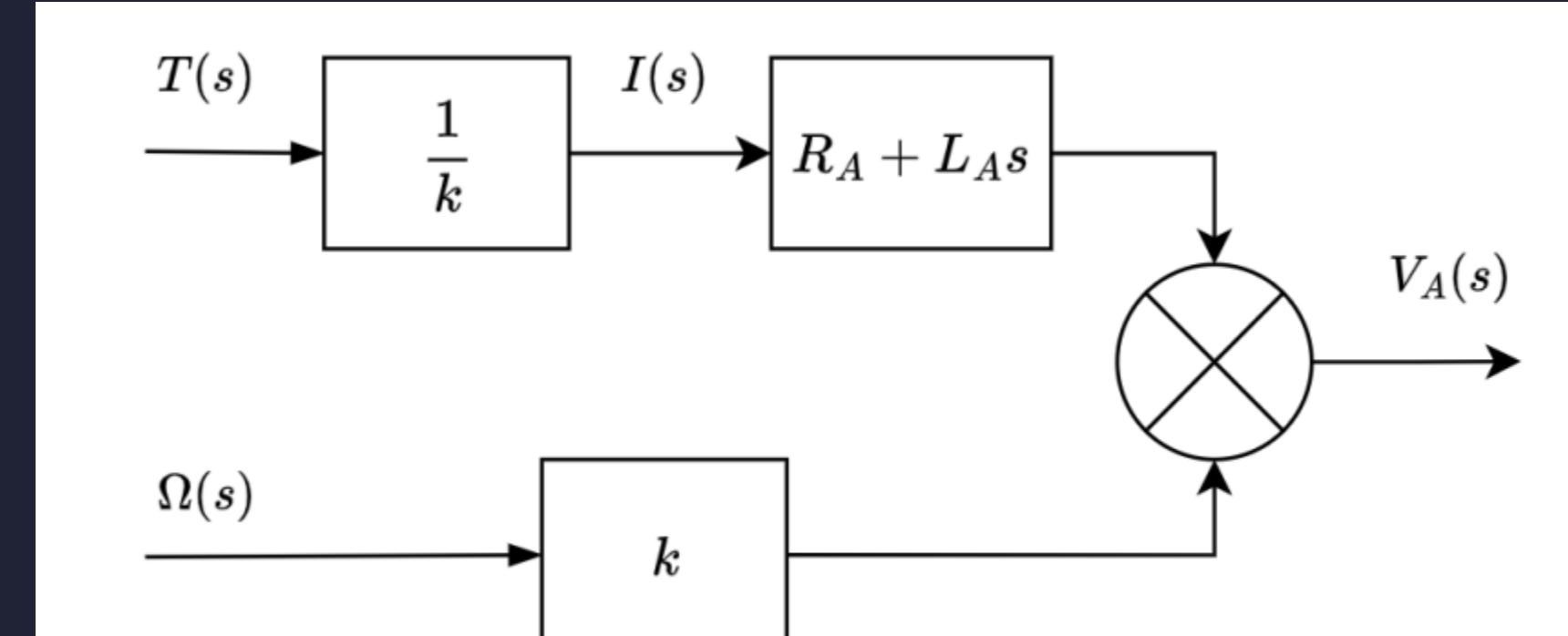
$$V_A(t) = R_A i(t) + L_A \frac{di(t)}{dt} + \omega(t)k$$

# Modelagem dos motores

$$V_A(s) = R_A I(s) + L_A s I(s) + \Omega(s) k$$

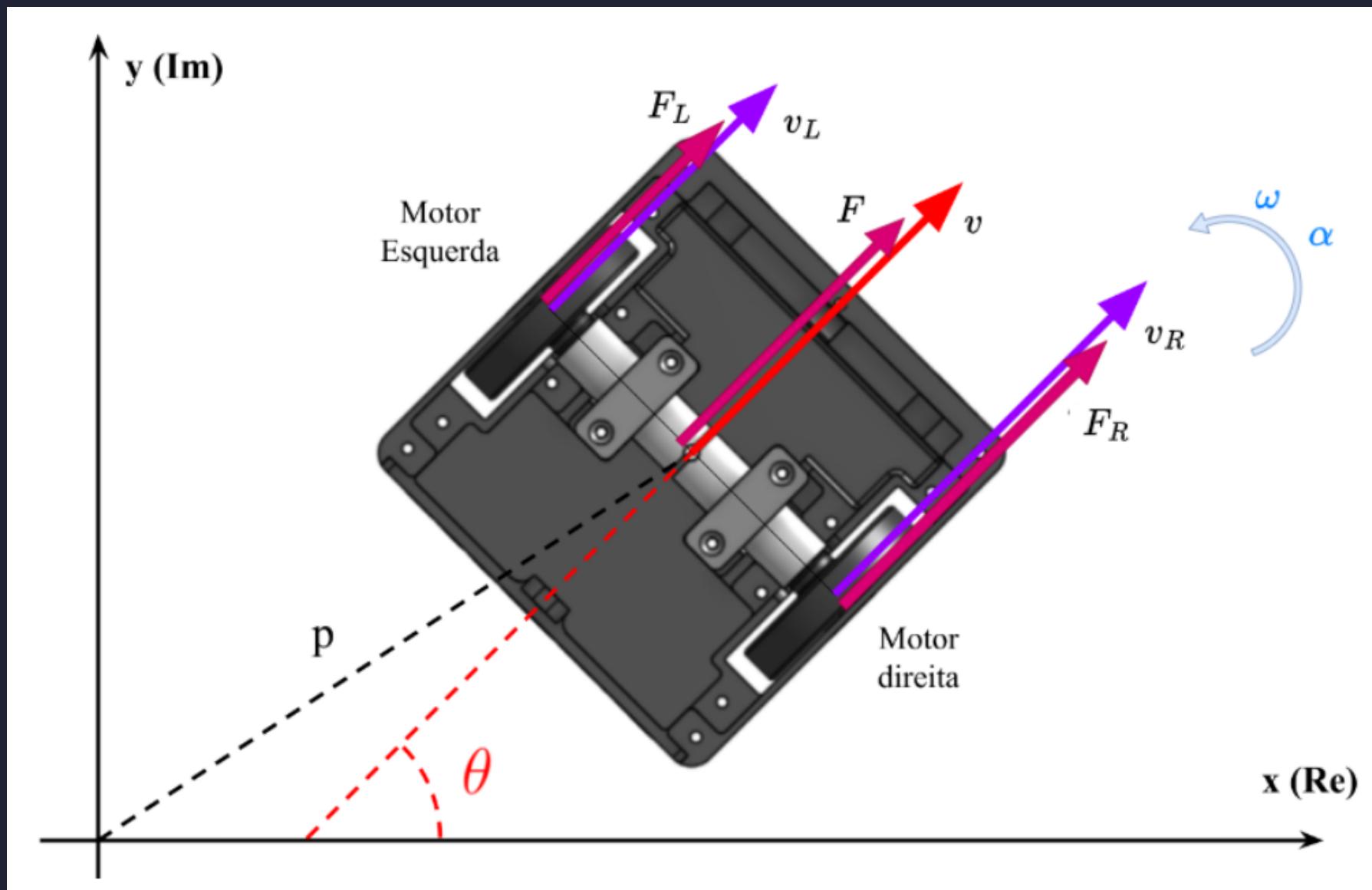


$$T(s) = k I(s)$$



Em forma de diagrama em blocos

# Modelo dinâmico do robô



Somatório de forças e momentos

$$ma = F_{dir} + F_{esq}$$

$$J\alpha = 2l(F_{dir} - F_{esq})$$

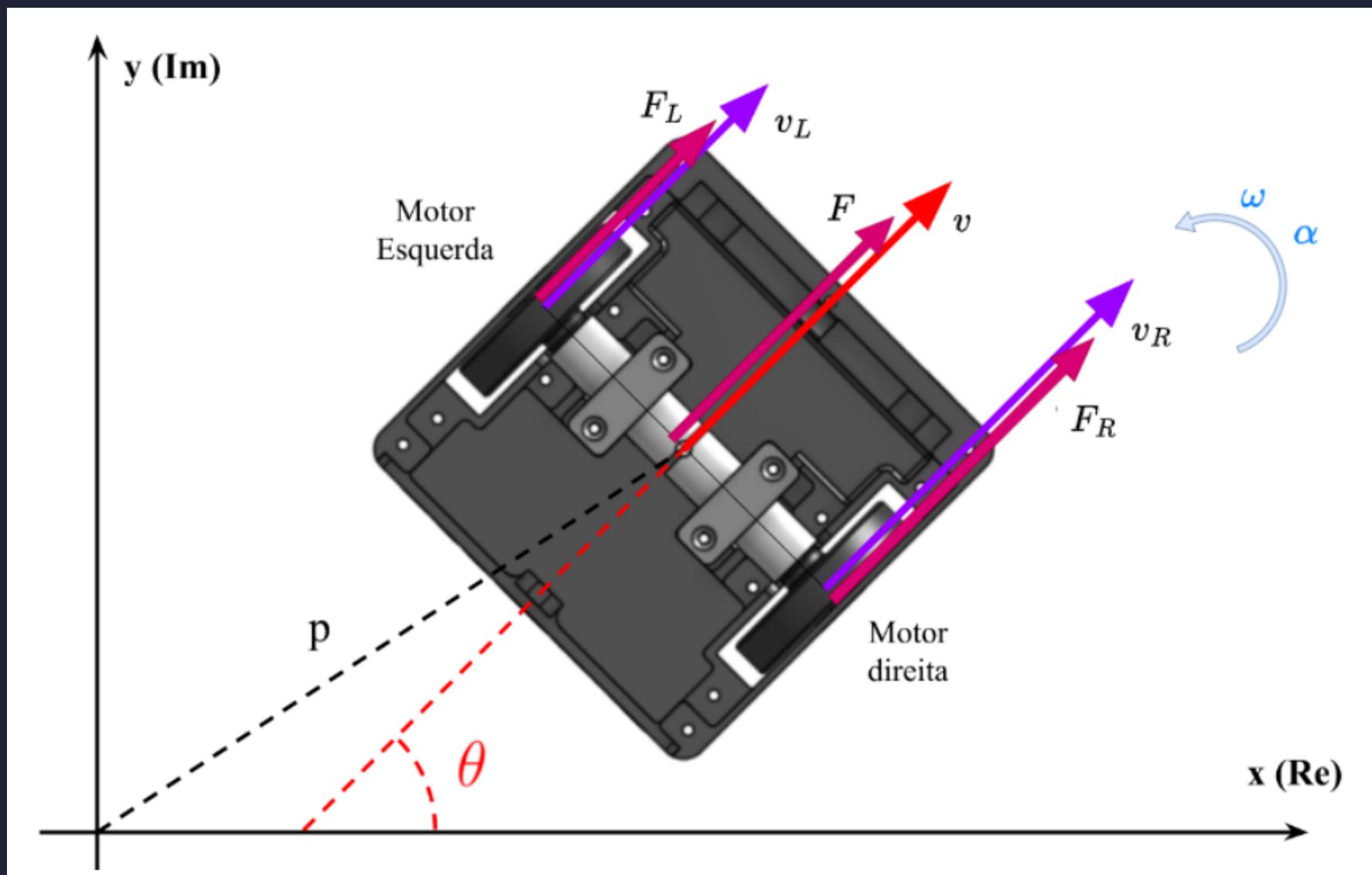
$$msV(s) = F_{dir}(s) + F_{esq}(s)$$

$$Js\Omega(s) = l(F_{dir}(s) - F_{esq}(s))$$

$$F_{dir}(s) = \frac{s(mV(s) + \frac{J}{l}\Omega(s))}{2}$$

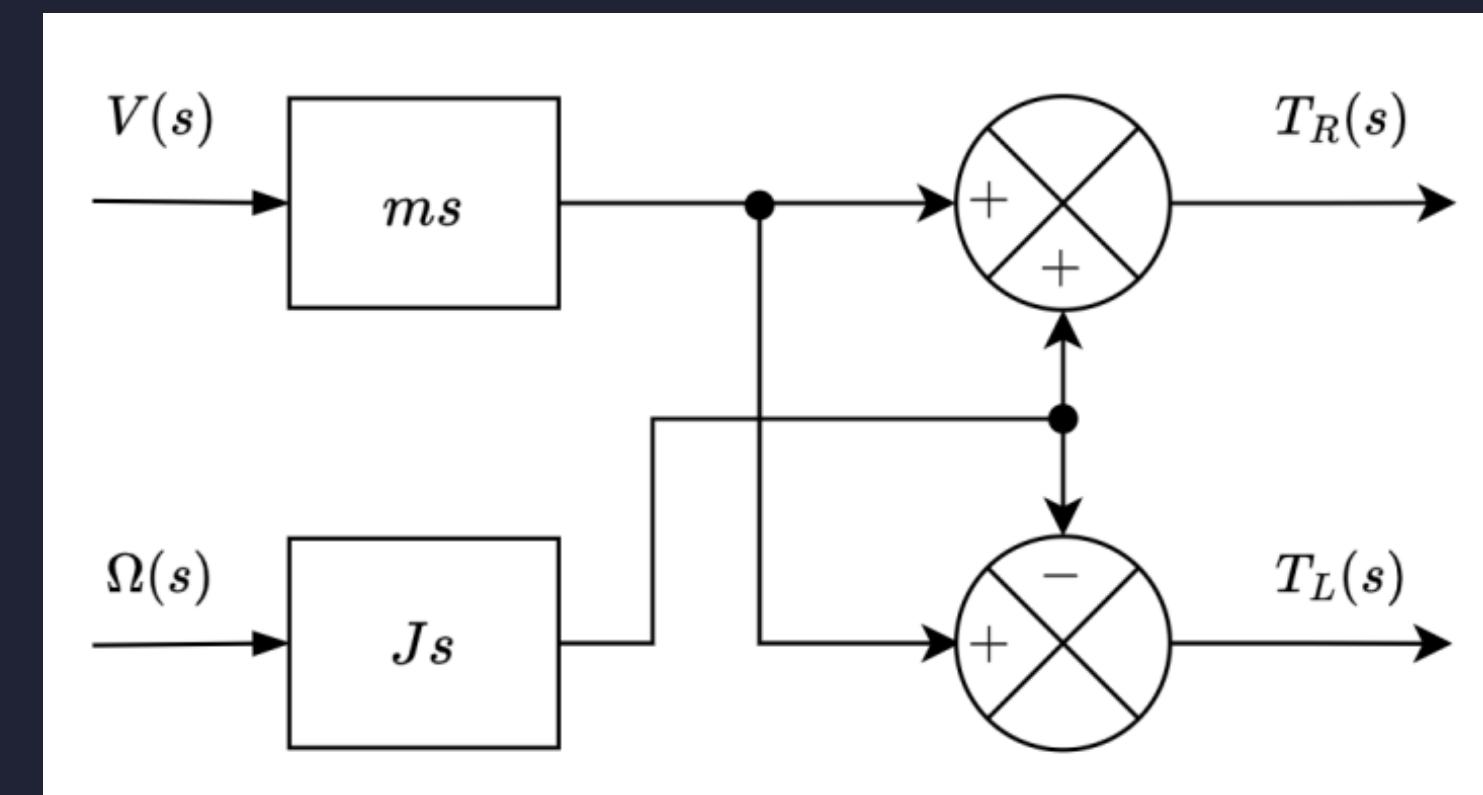
$$F_{esq}(s) = \frac{s(mV(s) - \frac{J}{l}\Omega(s))}{2}$$

# Modelo dinâmico do robô



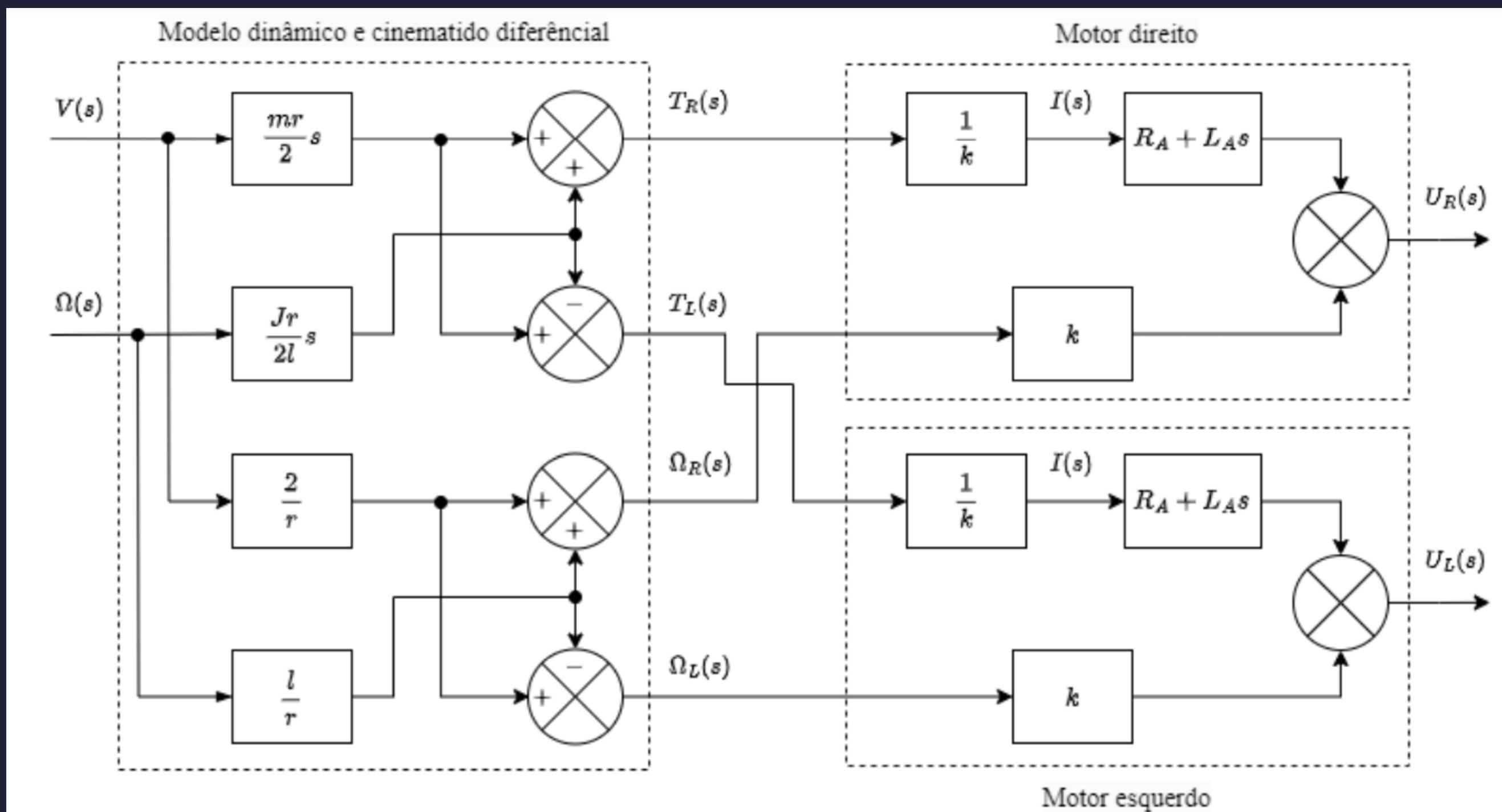
$$T_{dir}(s) = \frac{sr(mV(s) + \frac{J}{l}\Omega(s))}{2}$$

$$T_{esq}(s) = \frac{sr(mV(s) - \frac{J}{l}\Omega(s))}{2}$$



Em forma de diagrama em blocos

# Modelo dinâmico do robô



Unindo os diagramas

# Controlador de velocidade angular

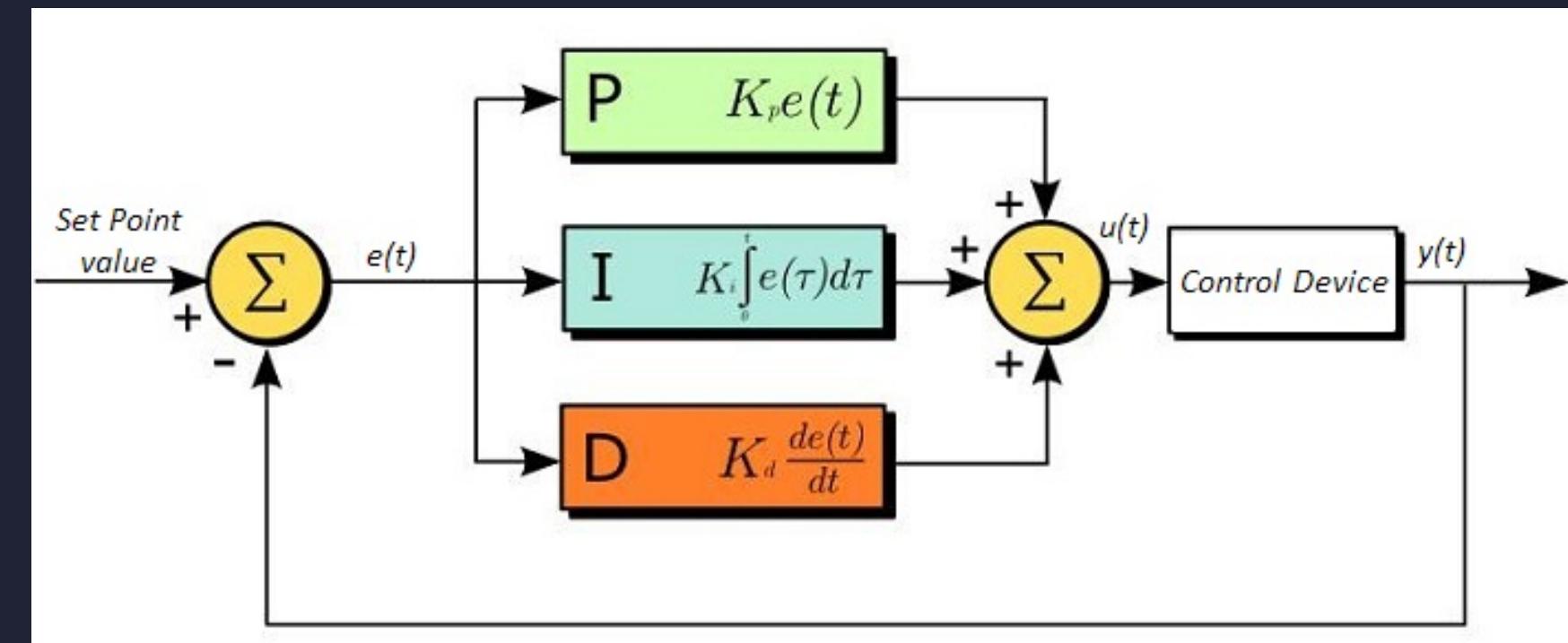
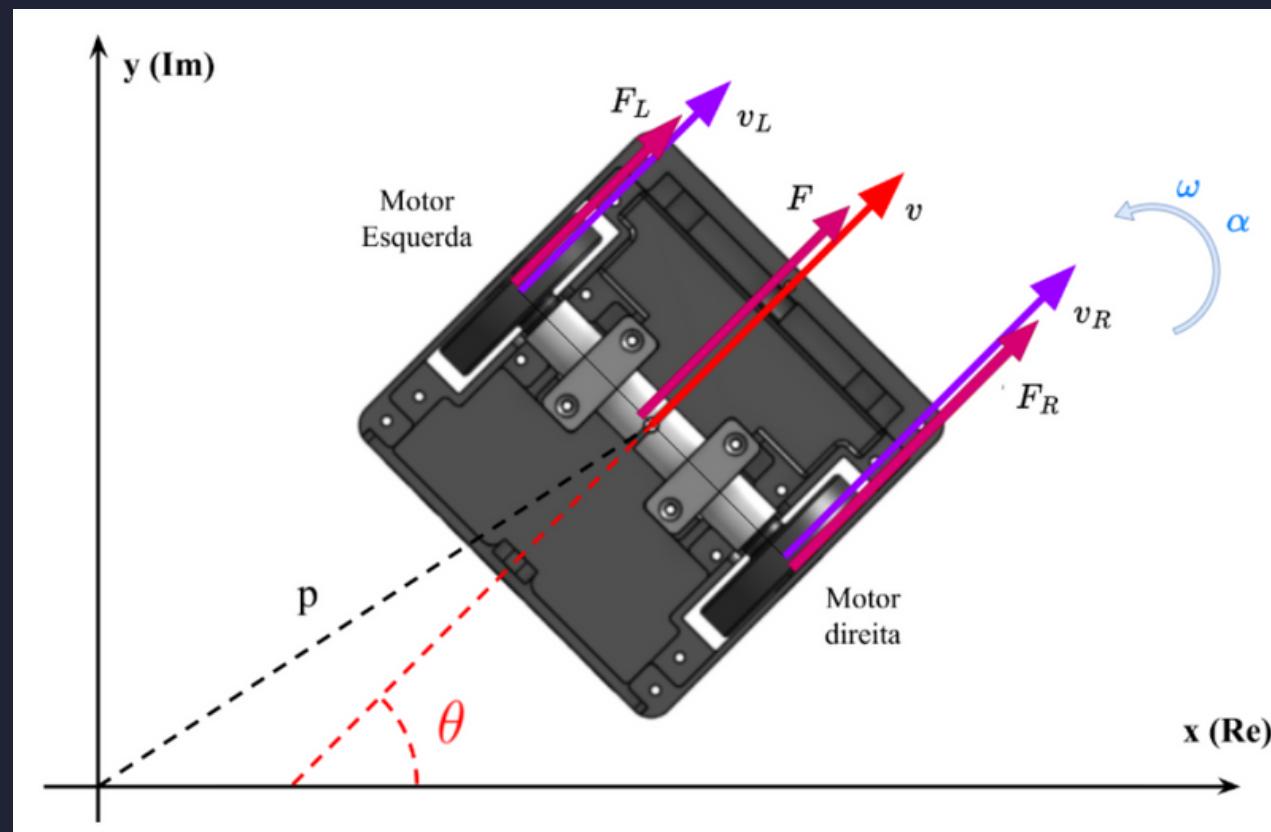
[VOLTAR AO SUMÁRIO](#)



# Controle de velocidade angular

Vários fatores podem acarretar discrepâncias entre as velocidades estimadas pelo modelo cinâmético e o obtido na prática. Esses efeitos podem comprometer totalmente o controlador de trajetória, ou piorar seu desempenho, uma vez que ele funciona com frequência limitada com base no sistema de visão.

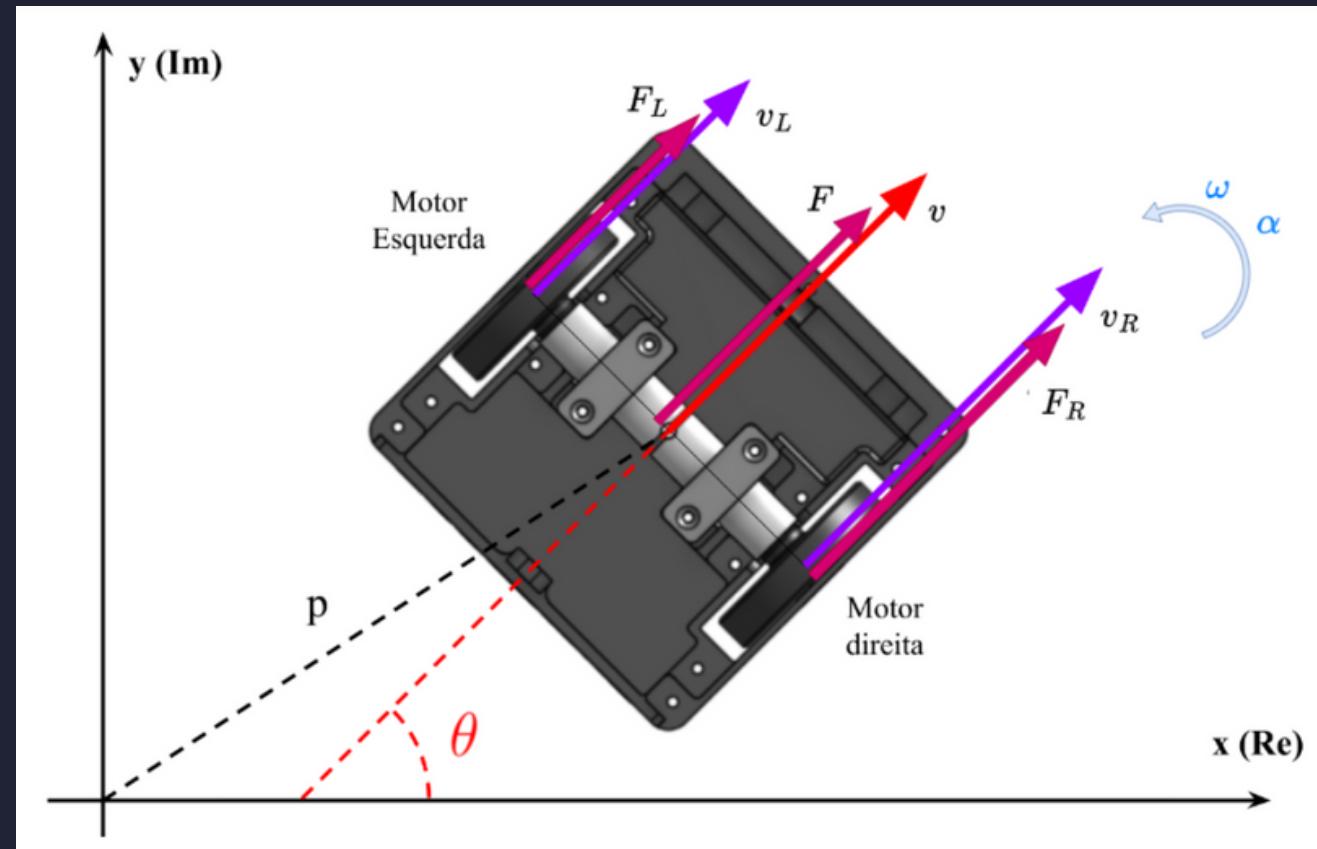
## Controle PID



$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

# Controlador PID discreto

O controlador PID utiliza a operação de derivada e integral, que devem ser realizadas via programação em um microcontrolador. Pra isso são aproximadas por funções numéricas de integração e derivação.



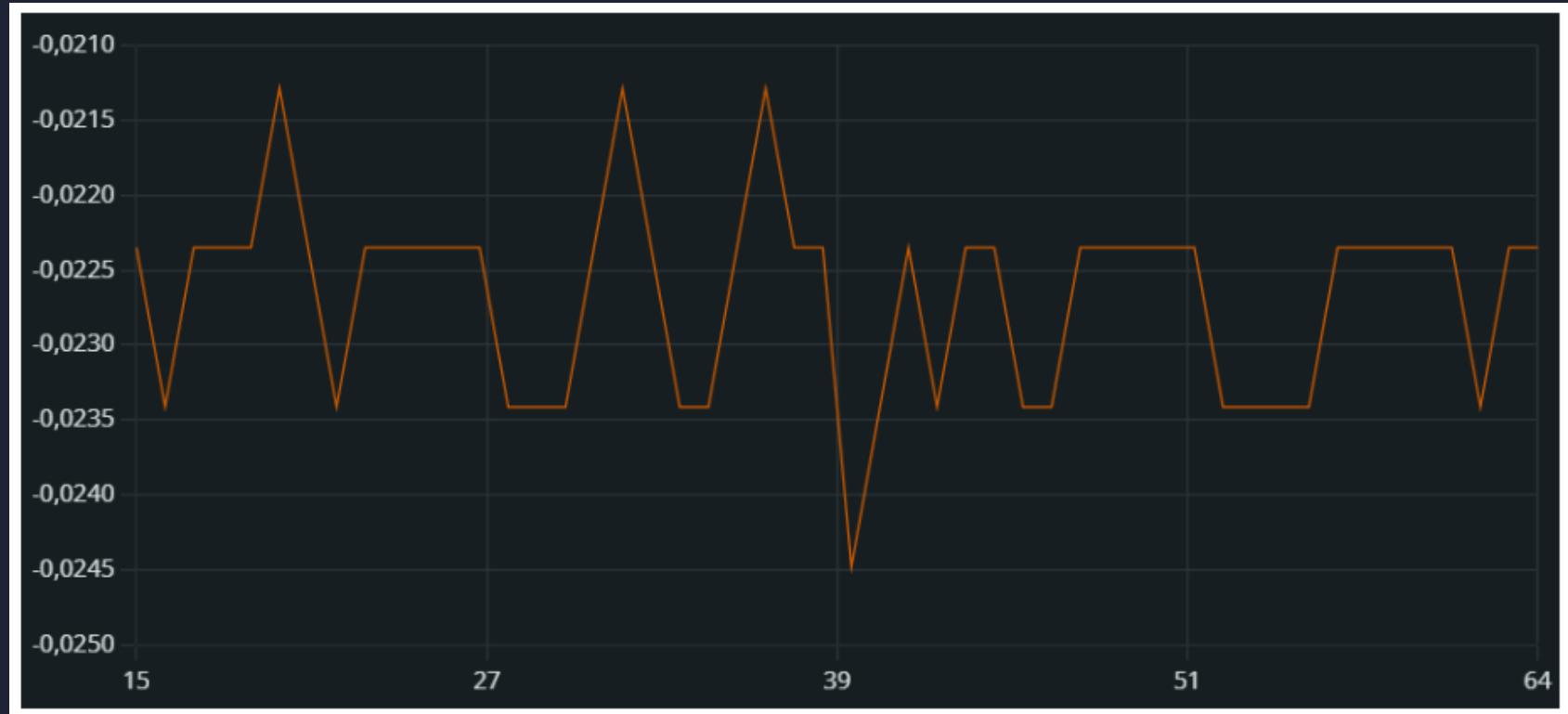
$$I[n] = \Delta t e[n] + I[n - 1]$$

$$D[n] = \frac{e[n] + e[n - 1]}{\Delta t}$$

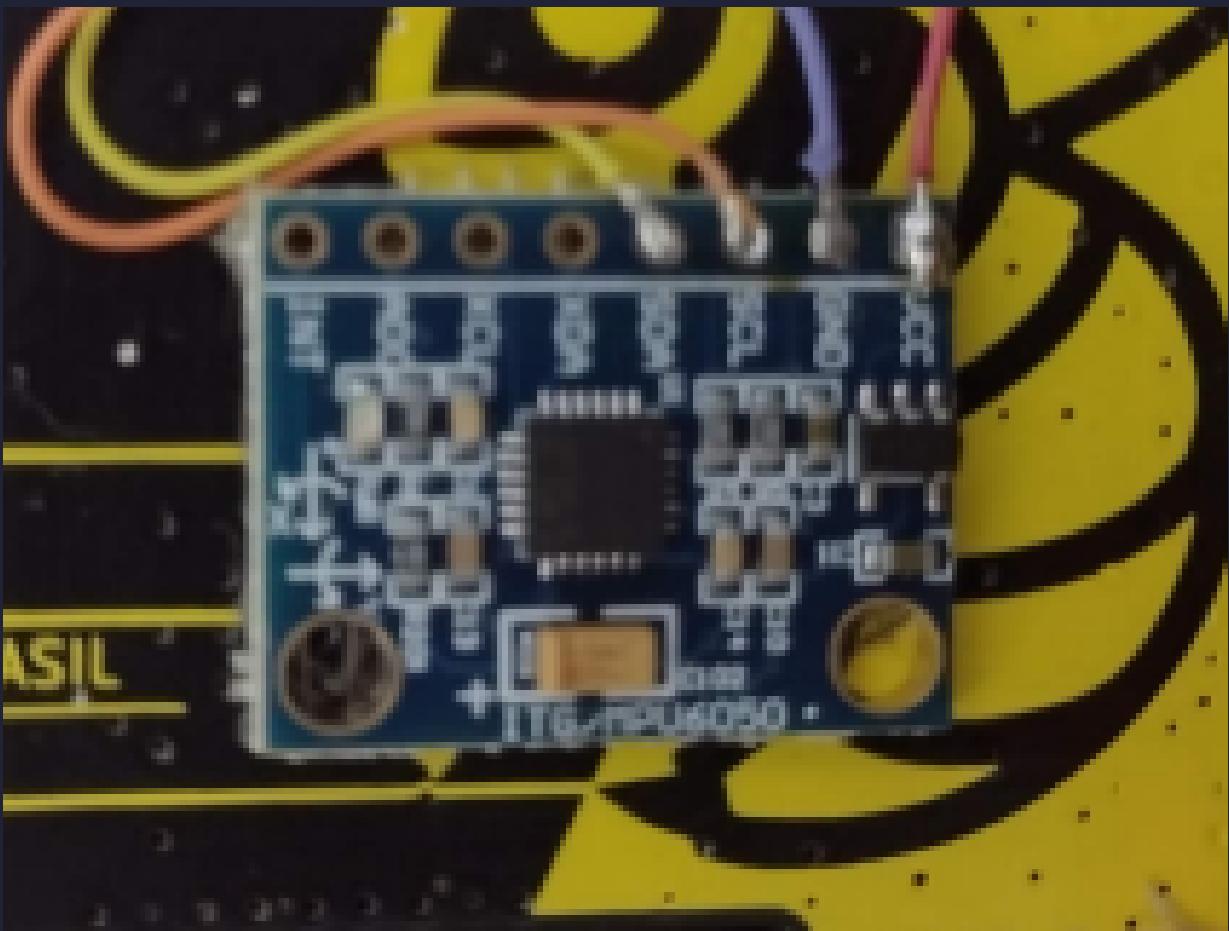
$$u[n] = k_p e[n] + k_i \Delta t e[n] + I[n - 1] + k_d \frac{e[n] + e[n - 1]}{\Delta t}$$

# Sensoriamento

É utilizado o giroscópio da IMU como sinal de realimentação do controlador. O giroscópio é calibrado antes das medições.



Leitura do giroscópio parado antes da calibração



# Sinal de controle

O sinal de controle é aplicado na diferença de velocidade dos motores. A média das velocidades é uma das estradas. Como isso é realizado em modulação PWM o sinal é um numero inteiro que varia de -1023 a 1023 onde o negativo representa a inversão de sentido.

$$PWM_{dir} = PWM_{medio} + 0,5PWM_{pid}$$

$$PWM_{esq} = PWM_{medio} - 0,5PWM_{pid}$$

# Movimentação autônoma

[VOLTAR AO SUMÁRIO](#)



# Movimentação autônoma

Existem dois controladores atuando na movimentação dos robôs:

- Controlador 1: Realizado internamente de cada robô. reduz o erro na velocidade angular do robô mais rapidamente que o segundo controlador
- Controlador 2: Atua no controle de trajetória de cada robô, utilizando o sistema de visão para fechar a malha com a pose atual de cada robô e da bola.

# Planejamento de trajetória

O planejamento de trajetória serve para definir um caminho para um robô sair de um ponto e chegar em outro sem colidir com outros obstáculos.

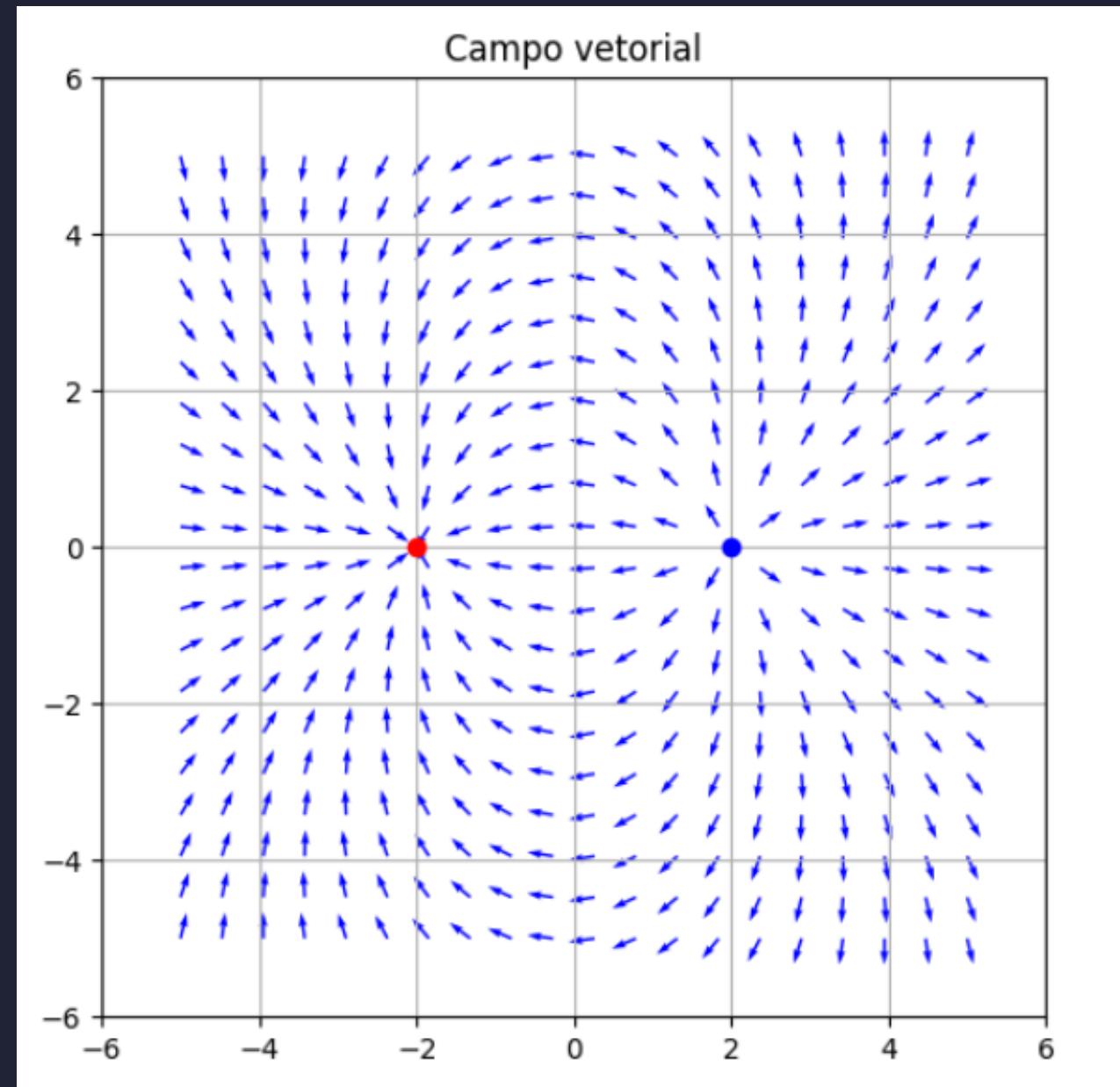
# Estrutura do código no computador



# Campos elétricos artificiais

A técnica de usar campos elétricos artificiais (ou campos vetoriais) permite definir boas trajetórias que desviam dos obstáculos com baixo custo computacional.

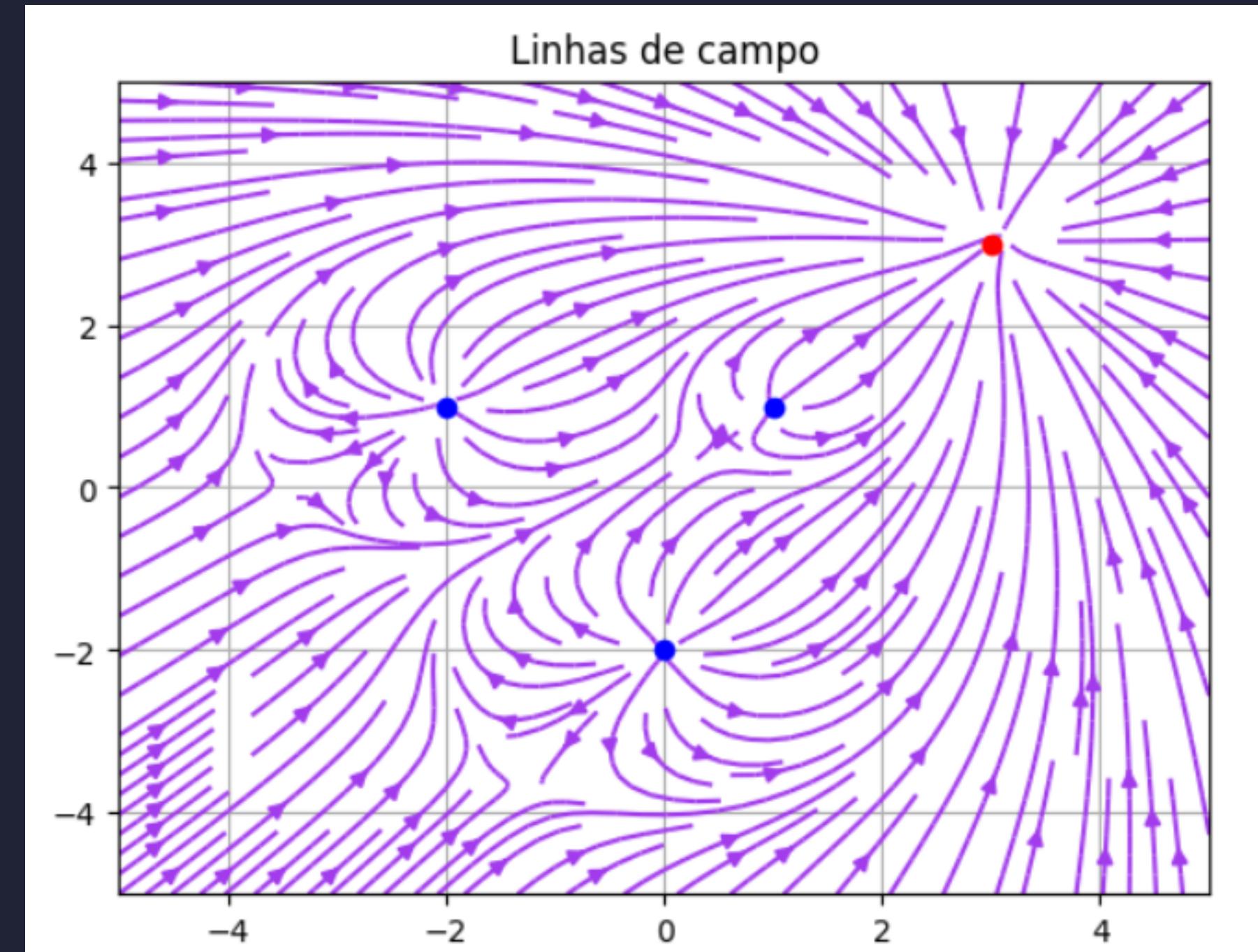
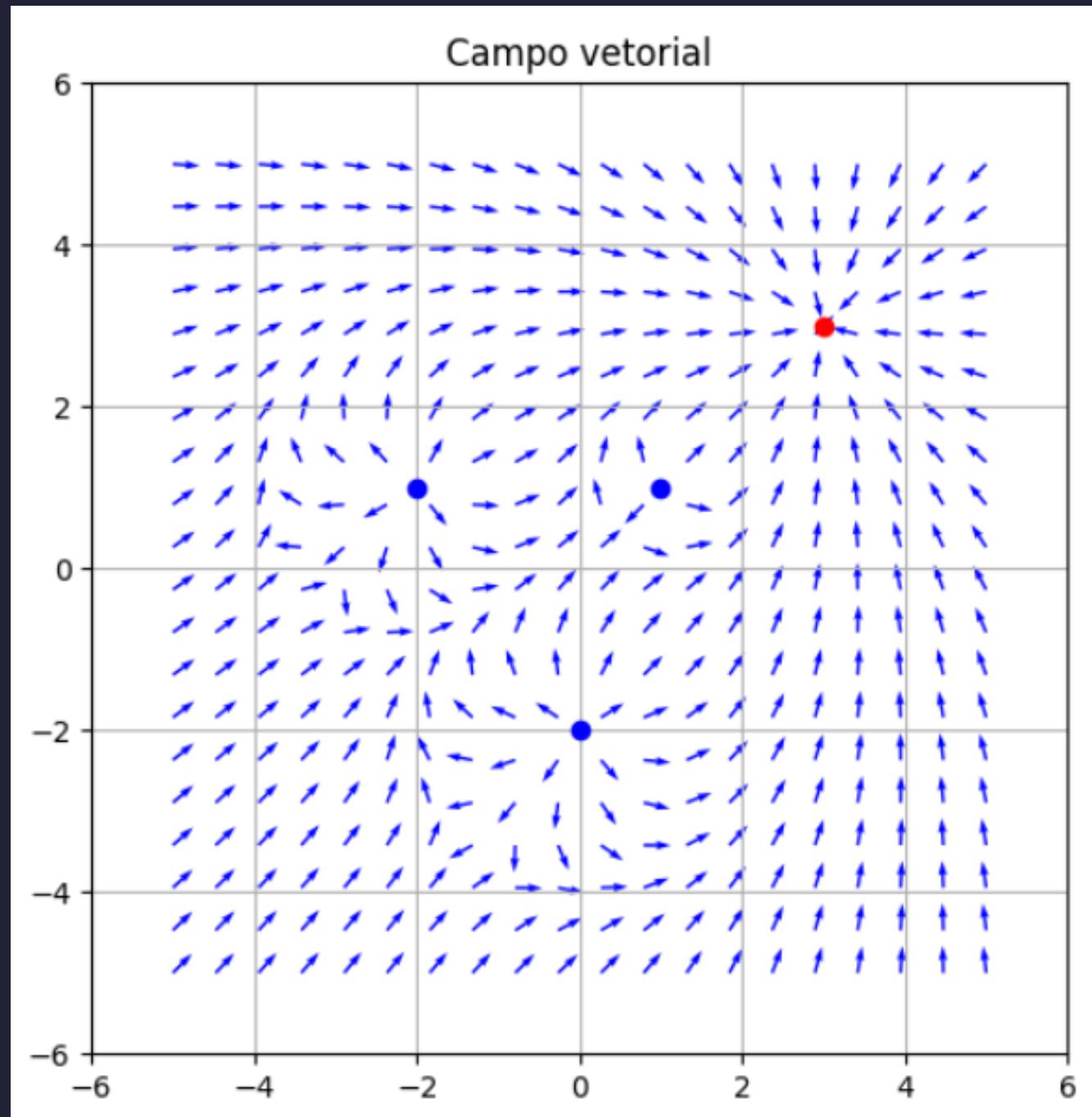
- Campos repulsivos em torno dos obstáculos
- Campo atrativos em torno do destino.



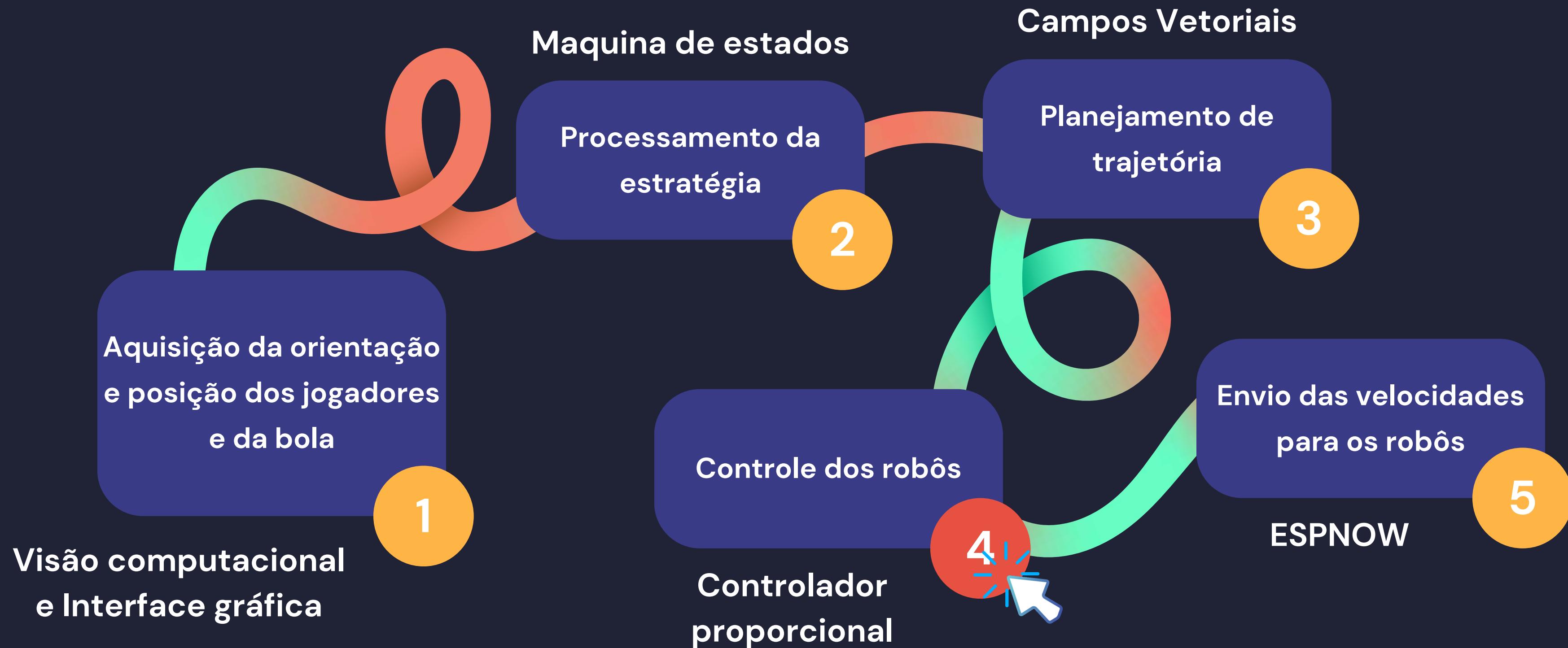
$$\vec{F}(x, y) = \sum_{cargas} \frac{Q_i}{(x_i - x)^2 + (y_i - y)^2}$$

$$\theta = \operatorname{arctg} \left( \frac{F_x(x, y)}{F_y(x, y)} \right)$$

# Simulação da implementação com campos vetoriais



# Estrutura do código no computador



# Controle de trajetória

Os campos vetoriais fornecem a orientação do robô em função de sua posição (x,y). Deste forma o controlador de trajetória regula a orientação e a velocidade tangencial do robô para que consiga percorrer o trajeto e atingir o alvo.

$$v = k_v erro_d$$

$$\omega = k_{ang} erro_{ang}$$

# Implementação do controlador de trajetória

---

## Algoritmo 1 Implementação do controlador de trajetória

---

```
Distancia = 1000 // valor alto
enquanto Distancia < 50 faça
    robo_vetor = vetor_robo_amarelo() // vetor diretor
    robo = pos_robo_amarelo() // posição (x,y)
    bola = pos_ball() // posição (x,y)
    azuis = lista_pos_robo_azul() // lista todas as posições (x,y)
    vetor = vetor_campo( robo, bola, azuis ) // calcula o vetor resultante
    erro_a = defasagem( robo_vetor, vetor ) // defasagem angular entre os vetores
    w = erro_a*kp_ang
    Distancia = dist(robo,bola)
    vel = Distancia*kp_vel + 300 // 300 é o valor mínimo para movimentar
    se mod(erro_angulo) > 1 então
        vel = 0
    fim se
    envia_robo( vel, w ) // envia velocidade tangencial e angular
fim enquanto
```

---

# Validação experimental

[VOLTAR AO SUMÁRIO](#)



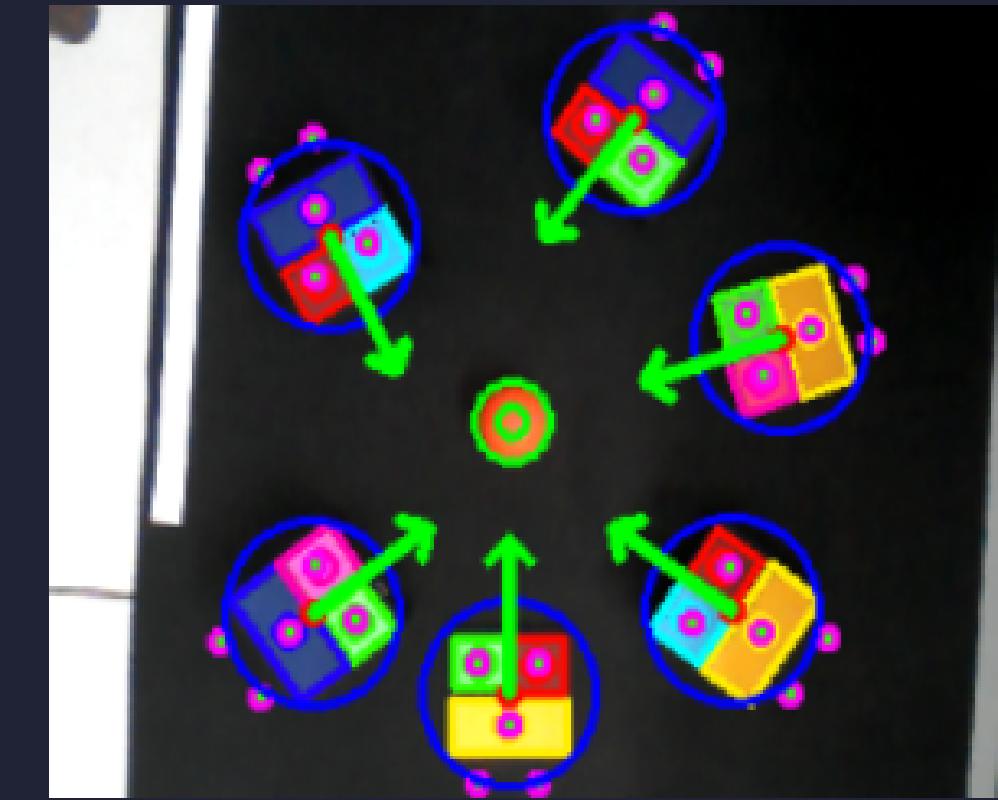
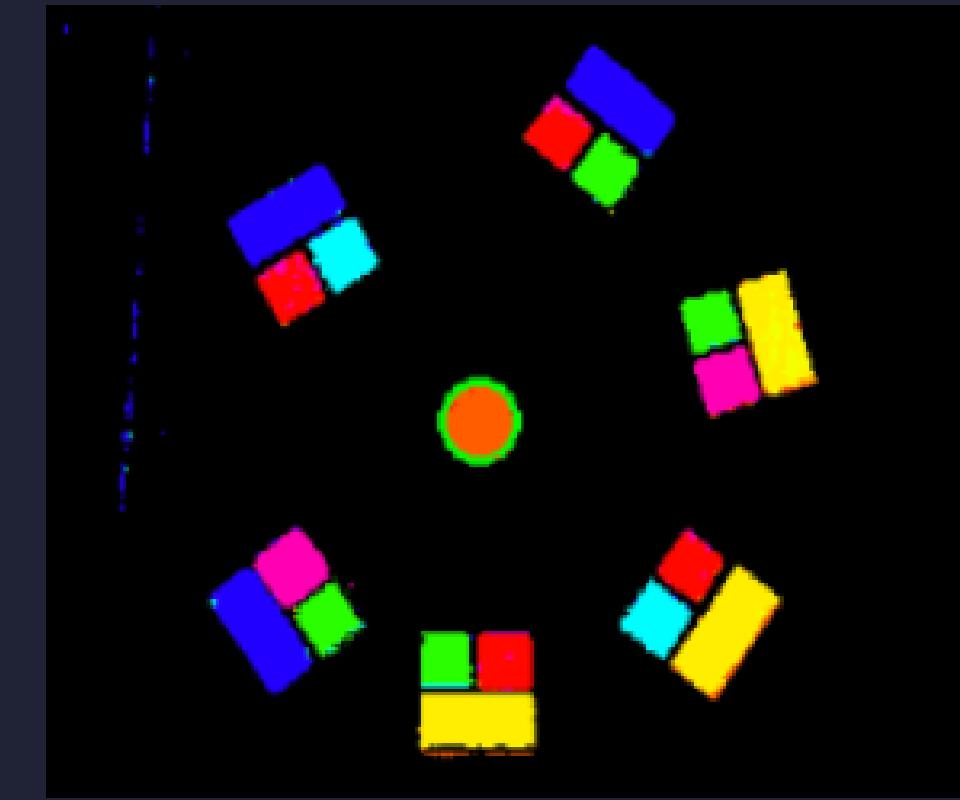
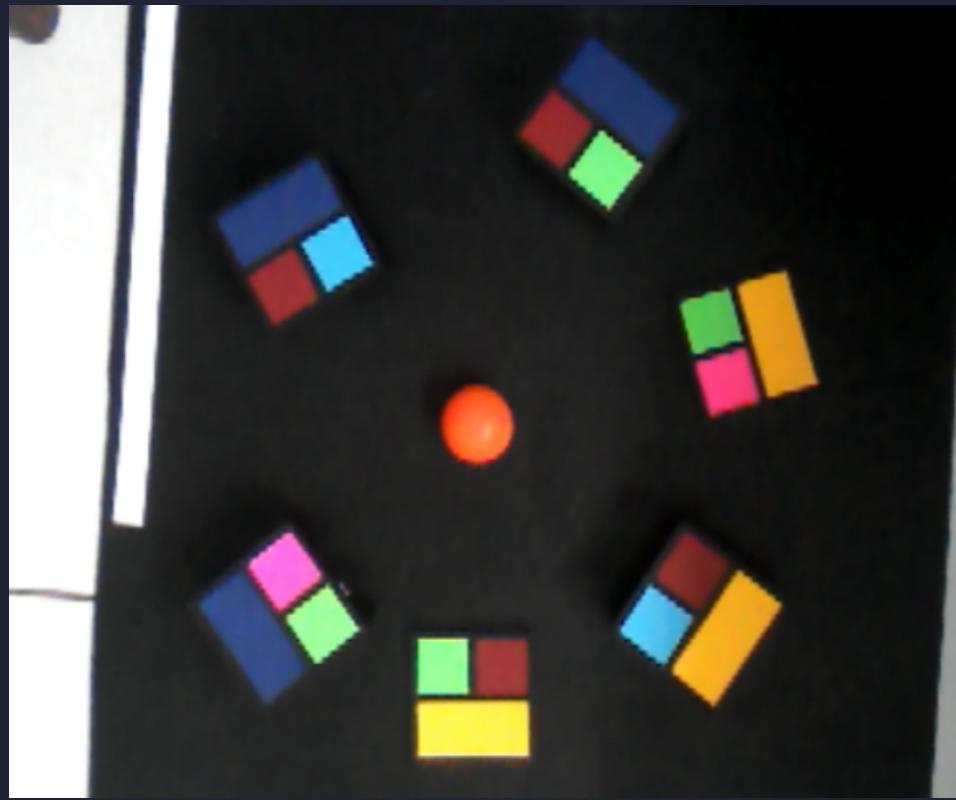
# Validação experimental do Sistema de visão computacional

Para validar o funcionamento do sistema de visão computacional foram realizadas 3 aferições em situações diferentes.

- 1.3 robôs de cada time e a bola parados;
- 2.Bola em movimento;
- 3.Robô em movimento.

# Validação experimental do Sistema de visão computacional

## Experimento 1 – 3 robôs de cada time parados



ID:12 (x,y): (66,13)  
Theta: 129°  
(W,H): (0,0)

ID:12 (x,y): (52,72)  
Theta: -90°  
(W,H): (0,0)

ID:13 (x,y): (32,25)  
Theta: 61°  
(W,H): (0,0)

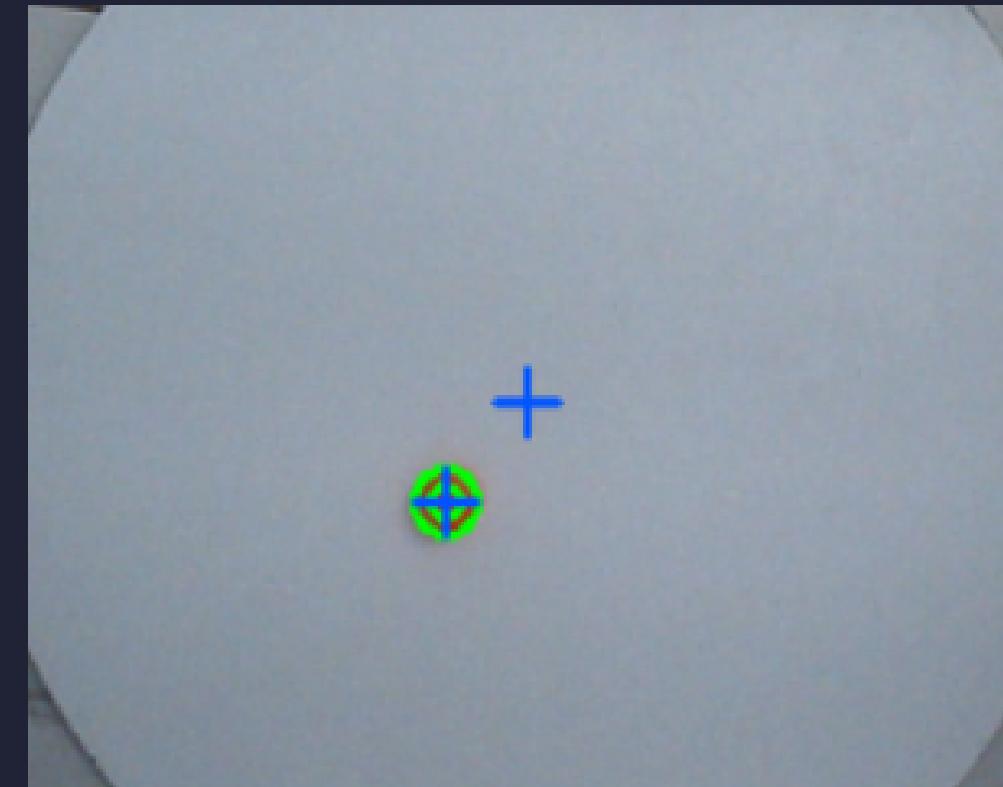
ID:13 (x,y): (77,64)  
Theta: -146°  
(W,H): (0,0)

ID:24 (x,y): (30,64)  
Theta: -35°  
(W,H): (0,0)

ID:24 (x,y): (82,35)  
Theta: 162°  
(W,H): (0,0)

# Validação experimental do Sistema de visão computacional

## Experimento 2 – bola em movimento



# Validação experimental do Sistema de visão computacional

## Experimento 3 – um robô em movimento



ID:24      (x,y): (265,330)  
(W,H): (69,39)      Theta: 108°

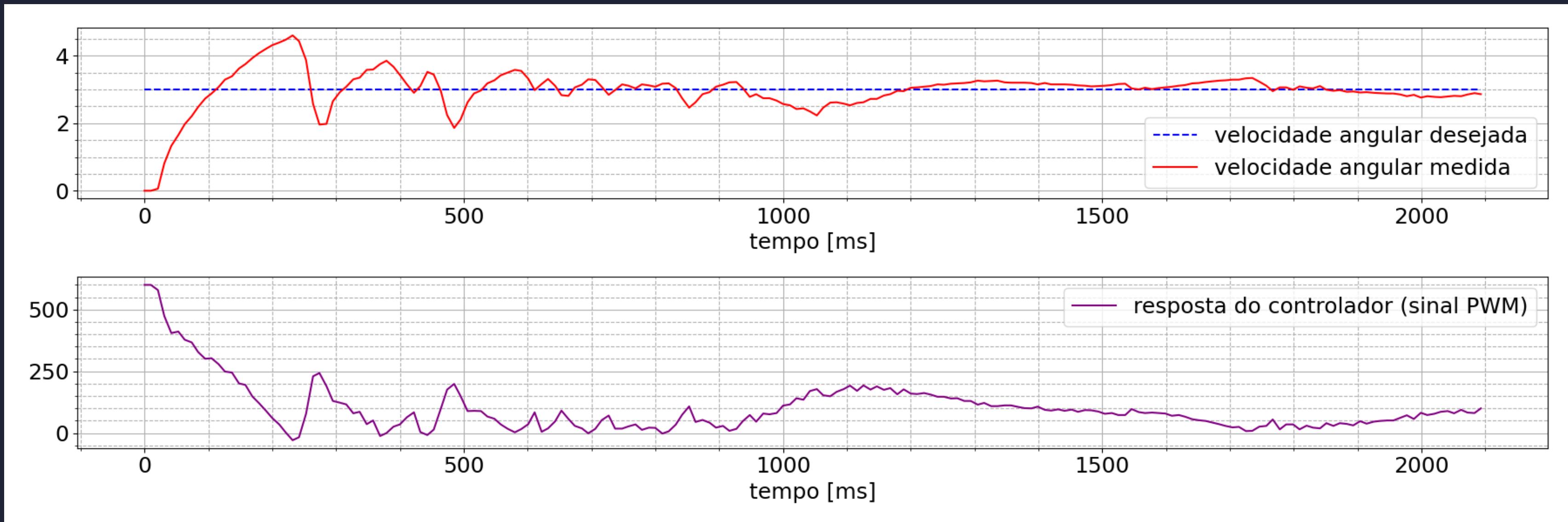
# Validação experimental do Controlador PID de velocidade angular

Foram realizadas medições para valores distintos de velocidade angular e valor médio de PWM. Onde, foram medidos e registrados os valores desejados e medidos de velocidade angular e sinal de controle.

$PWM_{medio}$	$\omega_{desejado} [rad/s]$	$Erro_{medio} [rad/s]$	$Erro_{medio} [\%]$
0	3.0	0.36	11.9%
0	5.0	0.75	15.04%
0	10.0	1.26	12.57%
0	20.0	1.86	9.3%
500	15.0	1.86	12.43%
600	3.0	0.41	13.56%
1000	0.0	0.35	inf %
1000	-4.0	0.6	14.9%
1000	4.0	0.66	16.53%
1000	10.0	0.67	6.68%

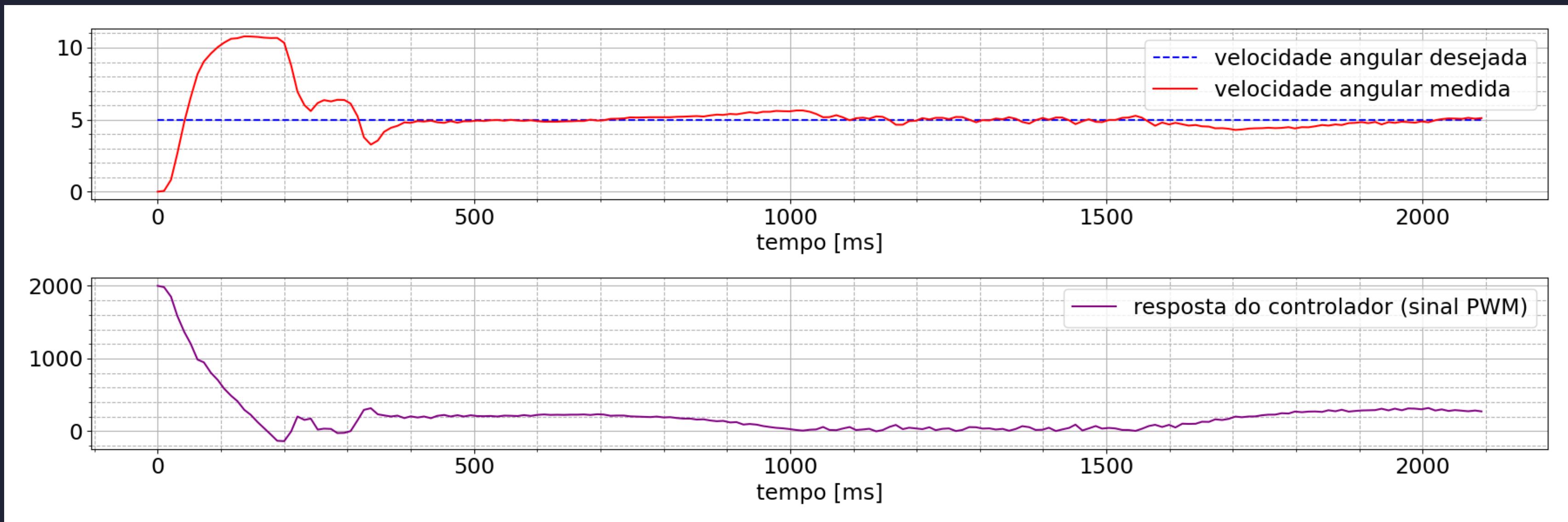
# Validação experimental do Controlador PID de velocidade angular

Experimento com PWM médio nulo e velocidade angular de 3 rad/s.



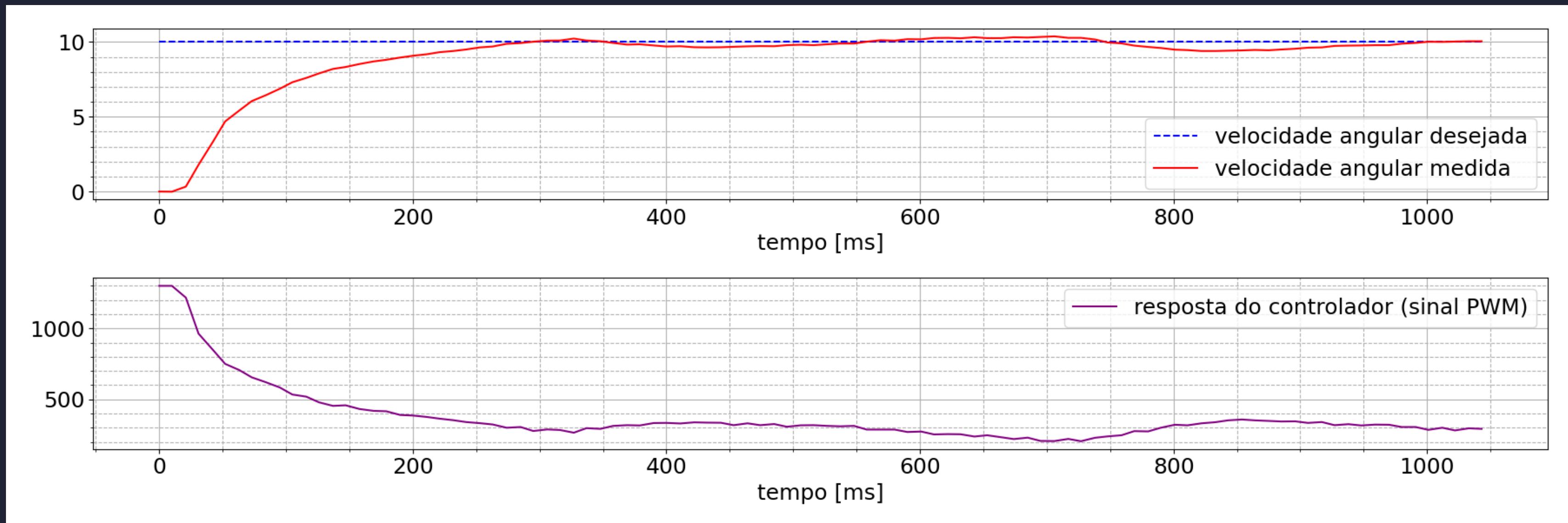
# Validação experimental do Controlador PID de velocidade angular

Experimento com PWM médio nulo e velocidade angular de 5 rad/s.



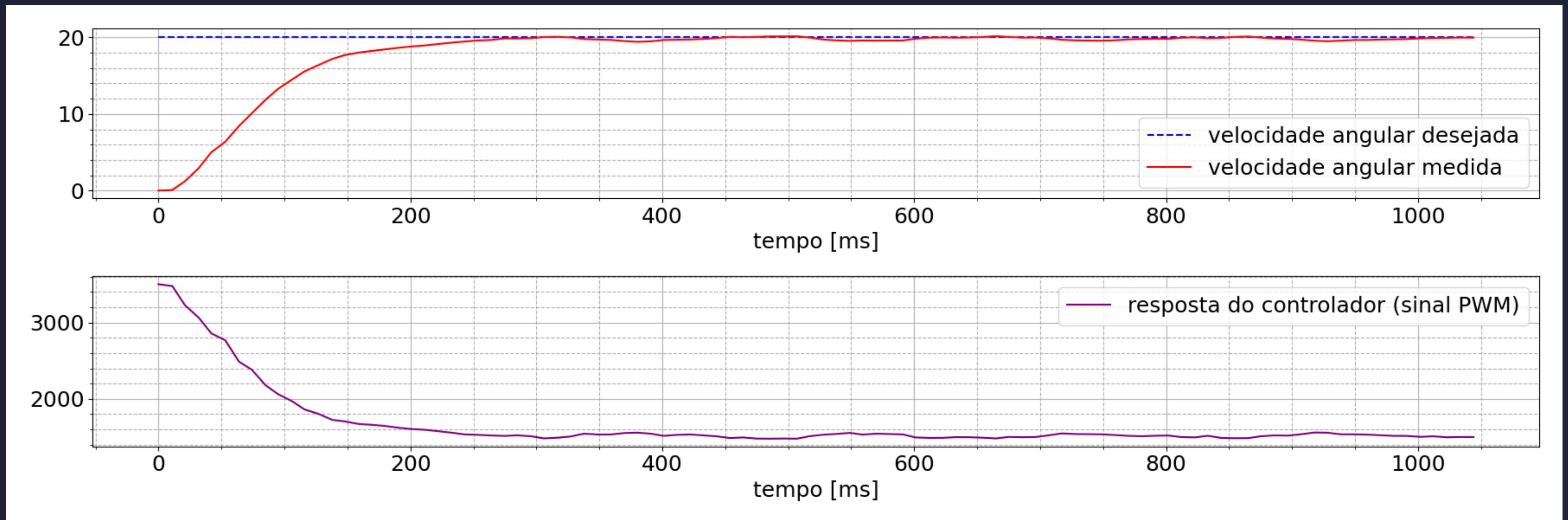
# Validação experimental do Controlador PID de velocidade angular

Experimento com PWM médio nulo e velocidade angular de 10 rad/s.



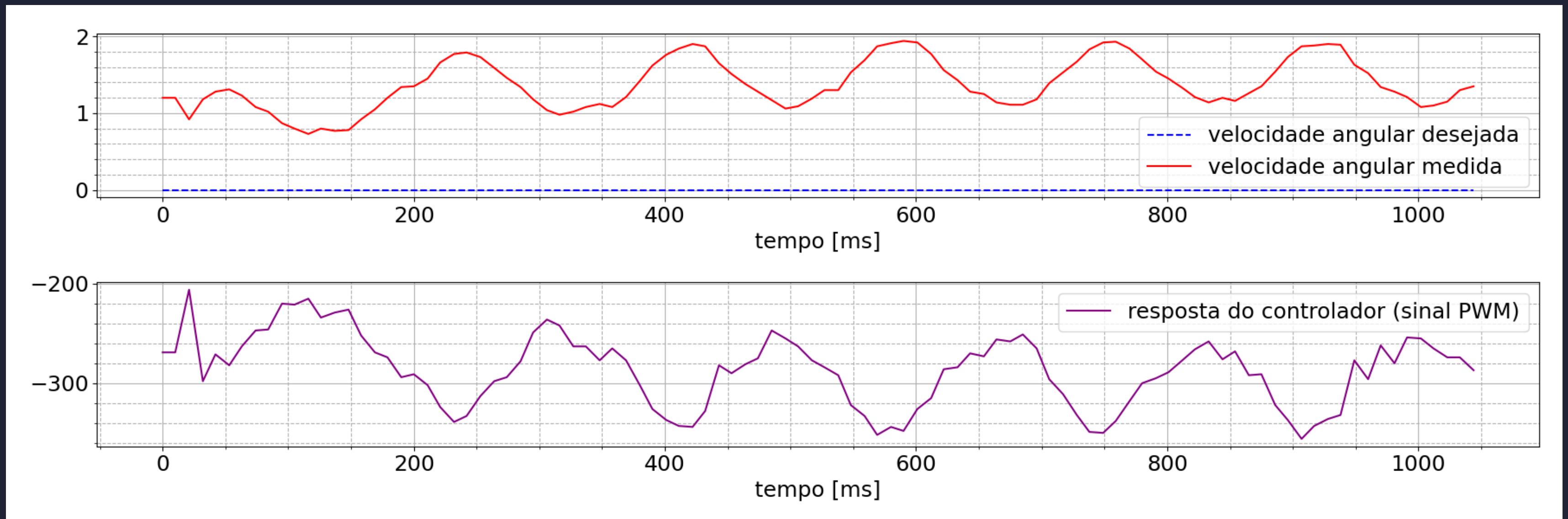
# Validação experimental do Controlador PID de velocidade angular

Experimento com PWM médio nulo e velocidade angular de 20 ras/s.



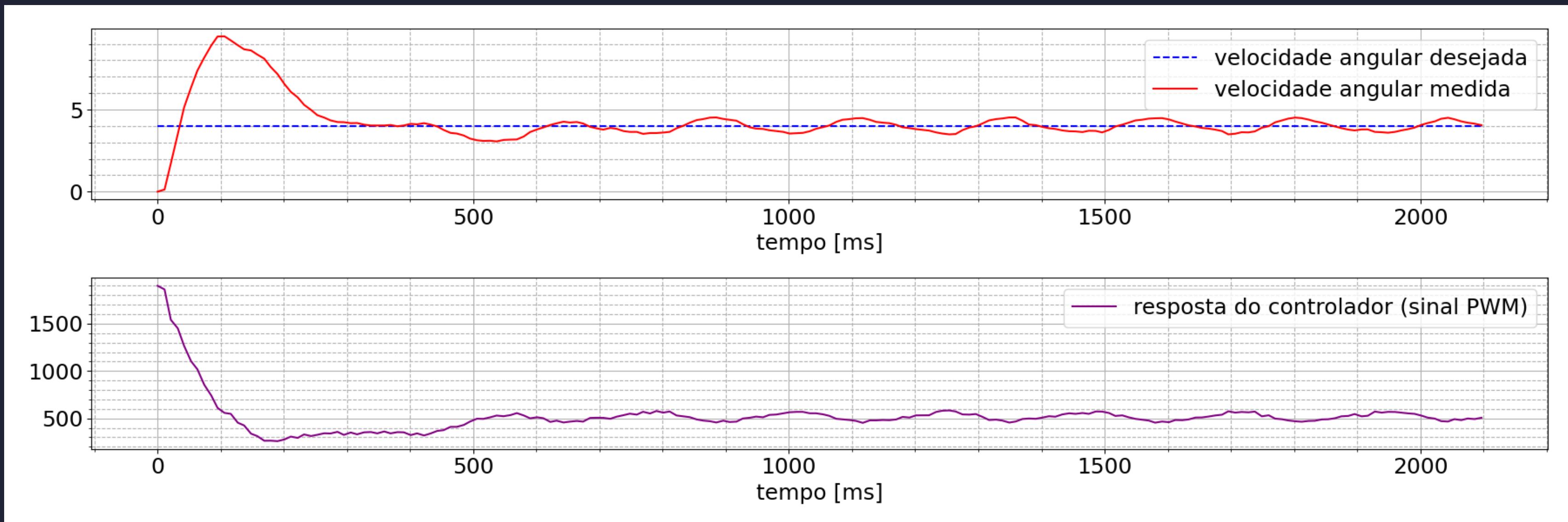
# Validação experimental do Controlador PID de velocidade angular

Experimento com PWM igual a 1000 e velocidade angular de 0 rad/s.



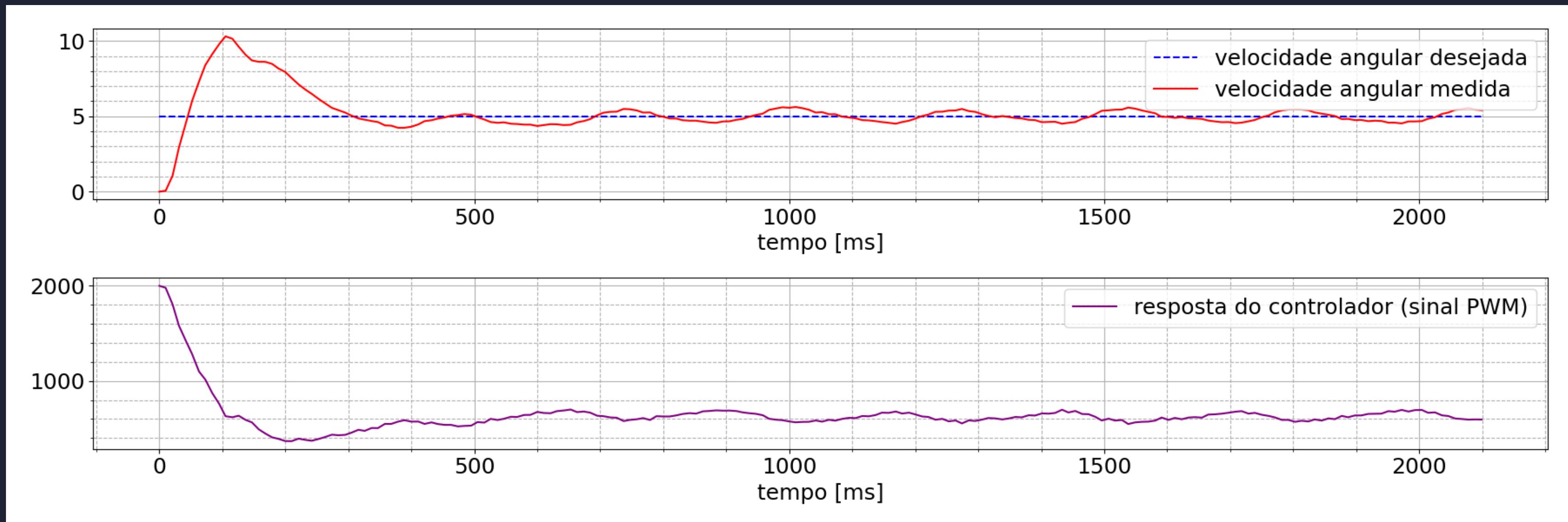
# Validação experimental do Controlador PID de velocidade angular

Experimento com PWM igual a 1000 e velocidade angular de 4 rad/s.



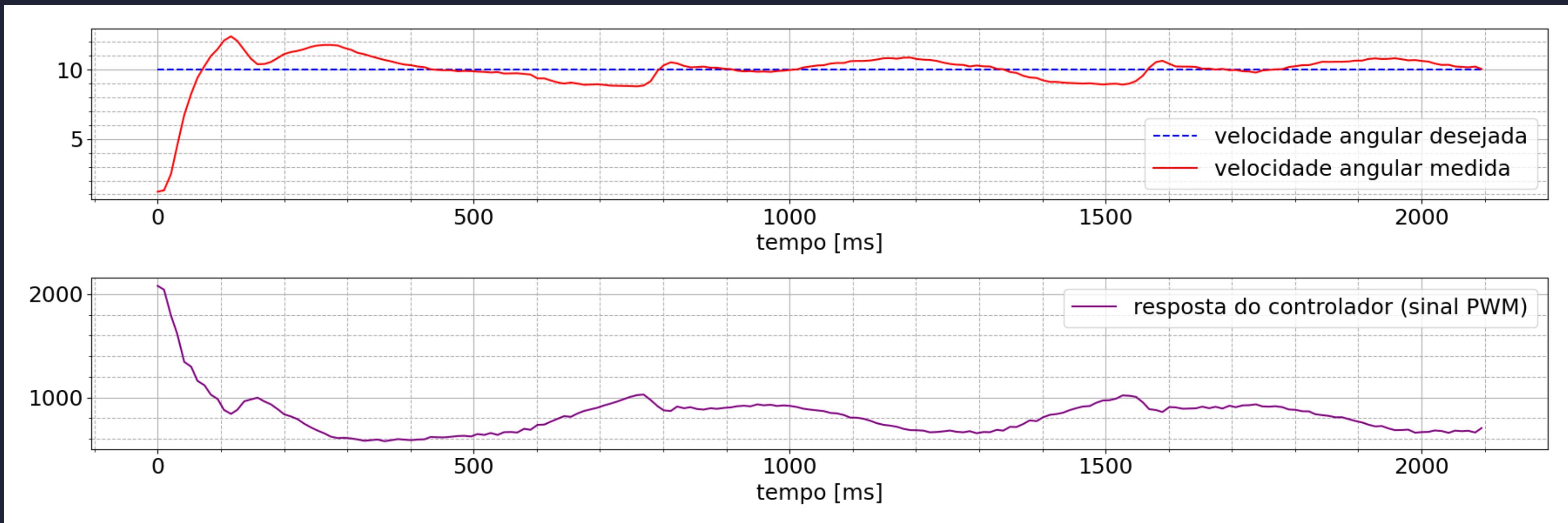
# Validação experimental do Controlador PID de velocidade angular

Experimento com PWM igual a 1000 e velocidade angular de 5 rad/s.



# Validação experimental do Controlador PID de velocidade angular

Experimento com PWM igual a 1000 e velocidade angular de 10 rad/s.



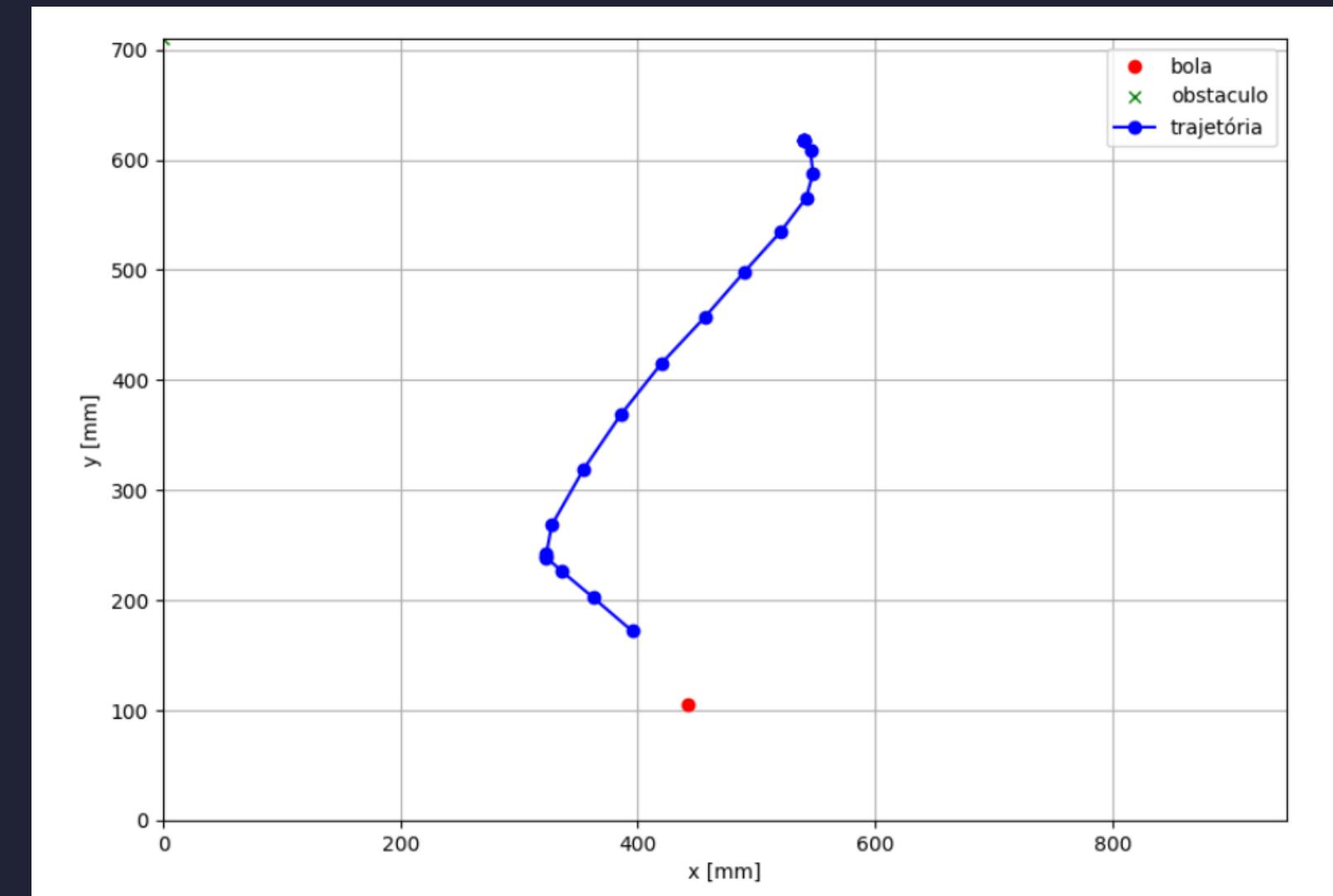
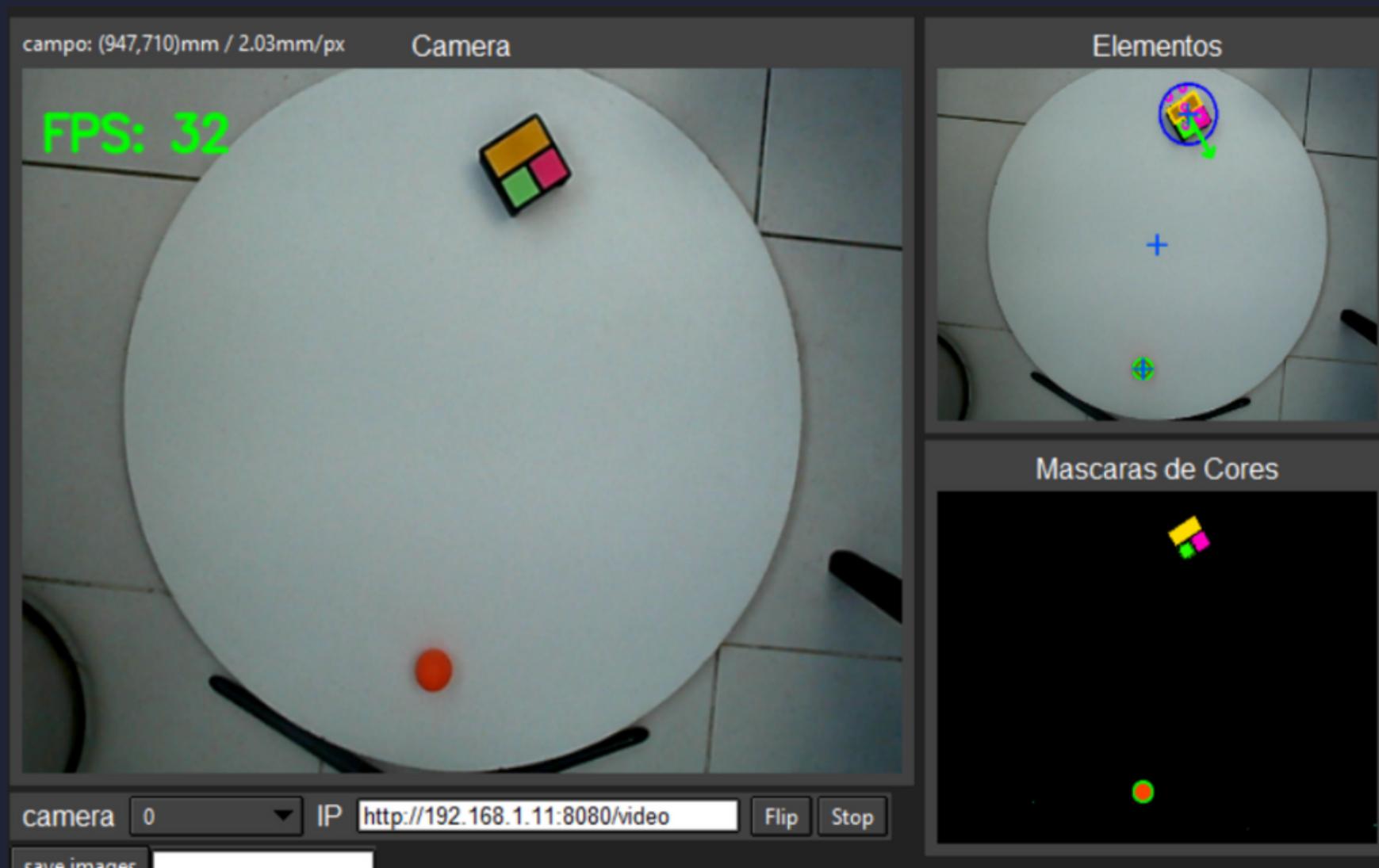
# Validação experimental do Controlador de trajetória

O controlador de trajetória desenvolvido é testado em duas situações nas quais o objetivo do jogador amarelo é alcançar a bola sem atingir o jogador azul.

- Experimento 1: São posicionados em campo apenas a bola e o jogador amarelo
- Experimento 2: É posicionado um jogador do time azul (um obstáculo para o jogador amarelo)

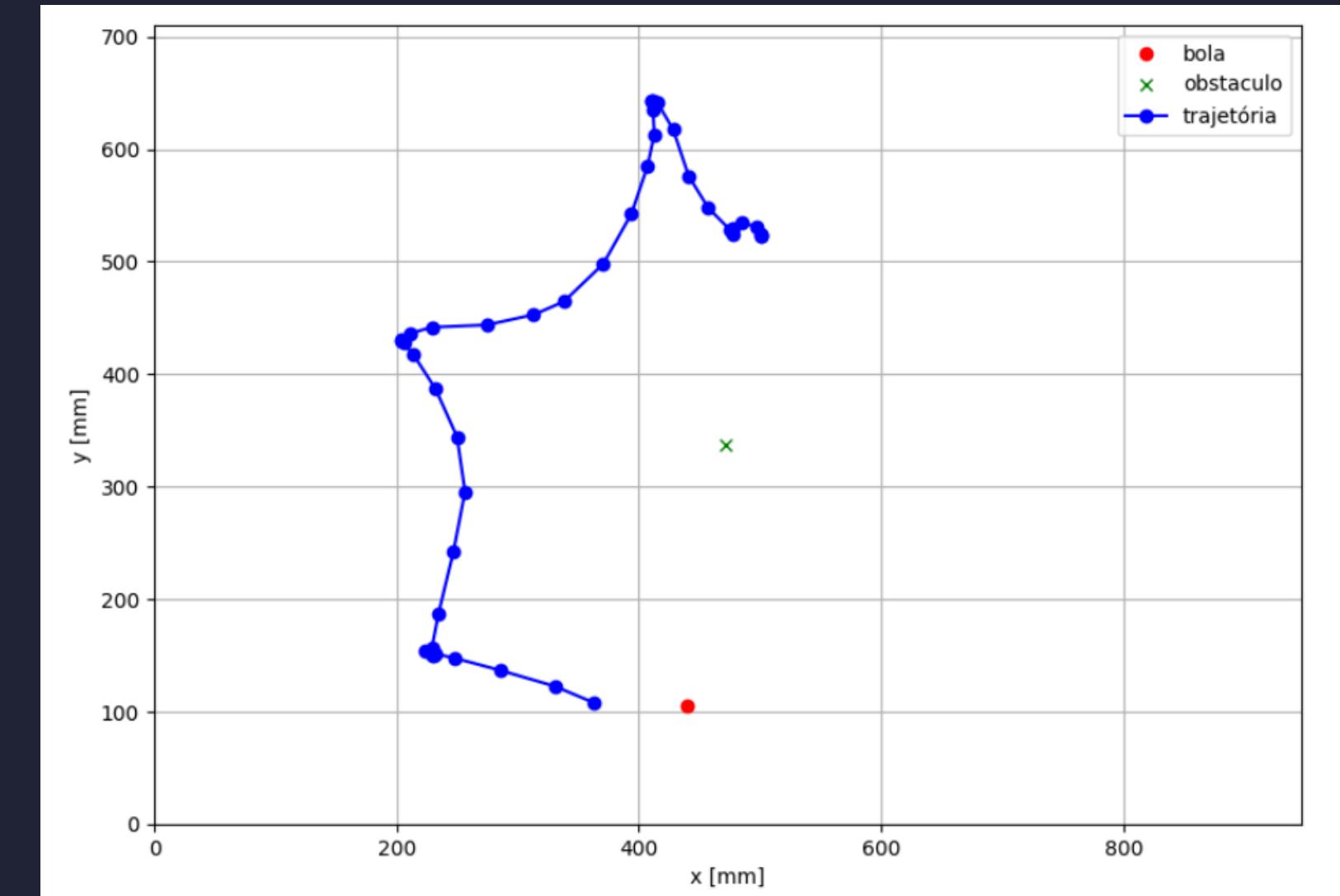
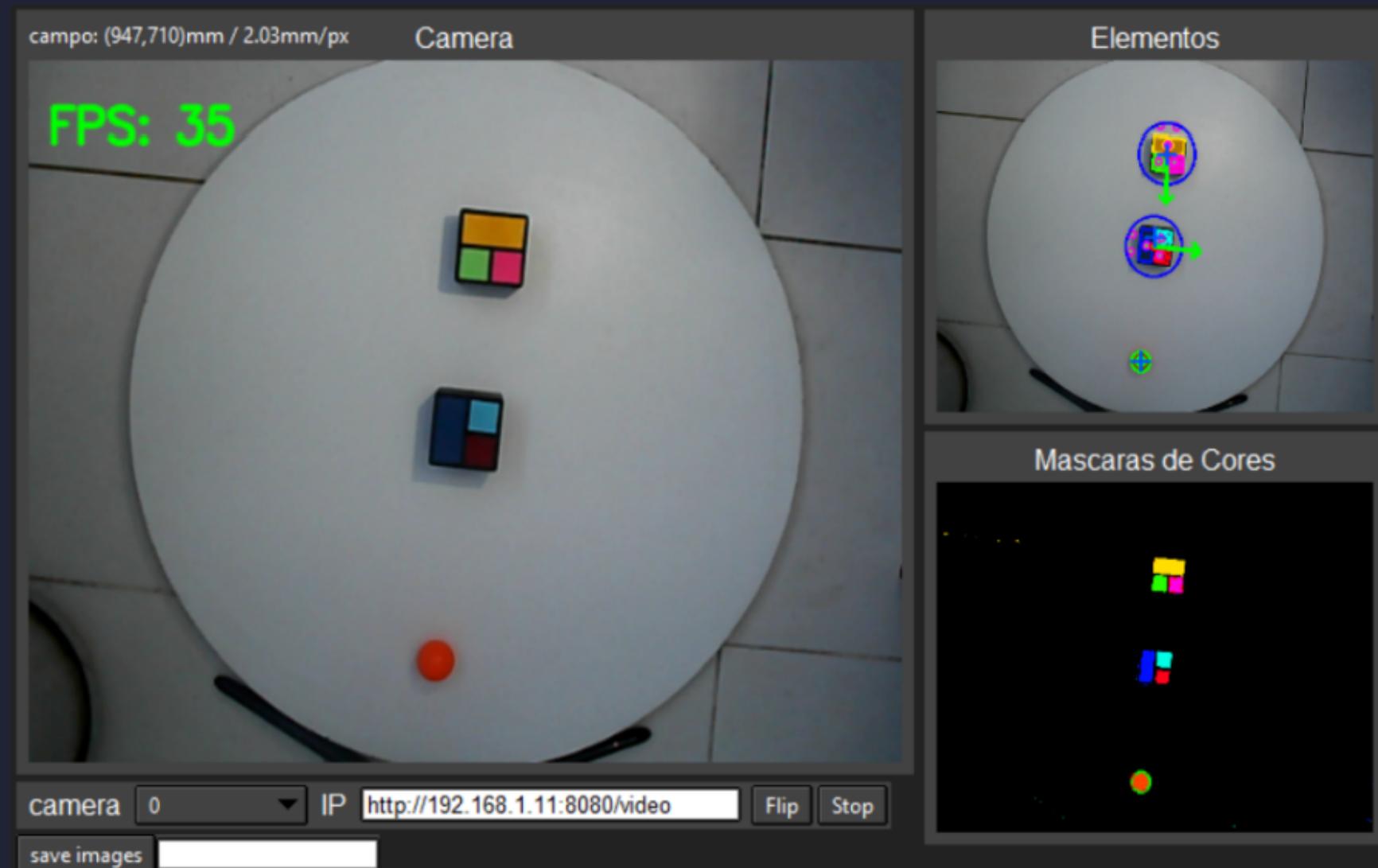
# Validação experimental do Controlador de trajetória

## Experimento 1 jogador amarelo e bola



# Validação experimental do Controlador de trajetória

## Experimento 1 jogadores amarelo e azul e bola



# Conclusão

[VOLTAR AO SUMÁRIO](#)

# Conclusão

Ao final deste trabalho, pode-se concluir, que ainda não é possível realizar de forma completa uma partida de futebol VSSS 3x3, no entanto, ele representa o primeiro passo no desenvolvimento de um time competitivo junto a equipe UERJBotz. Além disso, ele promoveu a construção da infraestrutura para futuros trabalhos.

# objetivos alcançados

1. Construção do campo principal com as medidas oficiais suporte menor de teste para câmera
2. Construção dos robôs: modelagem e impressão 3D das estruturas;
3. Desenvolvimento e fabricação do sistema eletrônico: projeto da placa de controle e modificações na placa vespa;
4. Desenvolvimento do algorítimo embarcado nos robôs, com capacidade de receber comandos do dispositivo Host e realizar a movimentação dos motores com controle em malha fechada de velocidade angular;
5. Desenvolvimento do sistema de visão computacional multiplataforma;
6. Desenvolvimento de interface gráfica multiplataforma, capaz de controlar os robôs durante jogos ou testes e monitorar e ajustar parâmetros do sistema de visão;
7. Desenvolvimento de algorítimo de controle de trajetória baseado em campos vetoriais.

# Conclusão

Desta forma, dentre os objetivos destacados na Introdução, todos foram cumpridos com exito, com exceção do ultimo objetivo ("Desenvolvimento de algoritmos de controle para movimentação autônoma dos robôs"). Apesar do controlador de velocidade angular funcionar de forma satisfatória, o controlador de trajetória ainda pode ser melhorado, uma vez que a movimentação apresenta muitos desvios de trajetória e oscilações até alcançar o alvo.

# Trabalhos Futuros

Ao final deste projeto foram levantadas varias questões que poderão ser estudadas em trabalhos futuros, apresentadas a seguir:

- Implementar arquivos de configuração no sistema de visão computacional, permitindo salvar as configuração sem a necessidade de alterar via linhas de código.
- Estudar a viabilidade do uso de redes neurais para automatizar o processo de detecção das máscaras de cores.
- Tornar o sistema de visão aberto para a comunidade e com boa documentação.
- Gerar uma versão executável do sistema de visão.
- O controlador embarcado no robô poderia funcionar melhor com *encoders* ao invés de uma *IMU*? Ou talvez com uma solução hibrida?
- Melhorar o algoritmo de planejamento de trajetória.
- Desenvolver estratégia de controle mais robustas, e com foco especial na movimentação concorrente dos robôs.

# Referências

[VOLTAR AO SUMÁRIO](#)

# Referências

- [1] RoboCIn. GitHub. Disponível em: <https://github.com/robocin/vss-vision>. Acesso em: [23 de julho de 2023].
- [2] A. Gottlieb, "CS W4733 NOTES – Differential Drive Robots," Columbia University, Fall 2017. Disponível em: [https://www.cs.columbia.edu/~allen/F17/NOTES\\_icckinematics.pdf](https://www.cs.columbia.edu/~allen/F17/NOTES_icckinematics.pdf). Acesso em: [23 de julho de 2023].
- [3] Regras IEEE Very Small Size Soccer (VSSS) – Serie A. ´ Chair: Adam Henrique Moreira Pinto. 2023. Disponível em: <https://www.crobotica.org/wp-content/uploads/2023/04/regrasVSS23.pdf>. Acesso em: [23 de julho de 2023].
- [4] Espressif Systems, "ESP8266EX Datasheet," Disponível em: [https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf). Acesso em: [23 de julho de 2023]. [5] Texas Instruments, "DRV8833 Datasheet," Disponível em: <https://www.ti.com/lit/ds/symlink/drv8833.pdf?ts=1690118666252>. Acesso em: [23 de julho de 2023].
- [6] Kim, J., Kim, D., Kim, Y. and Seow, K. (2004). Soccer Robotics. Springer Tracts in Advanced Robotics, Springer-Verlag.

# Agradecimentos

# Equipe UERJBotz

O desenvolvimento deste trabalho teve o apoio da equipe UERJBotz, a qual o autor é membro desde 2018. A UERJBotz é uma equipe dicente de robótica associada à Universidade do Estado do Rio de Janeiro (UERJ), comprometida desde 2013 com o progresso e realização de projetos em diversas categorias dentro do campo da robótica.



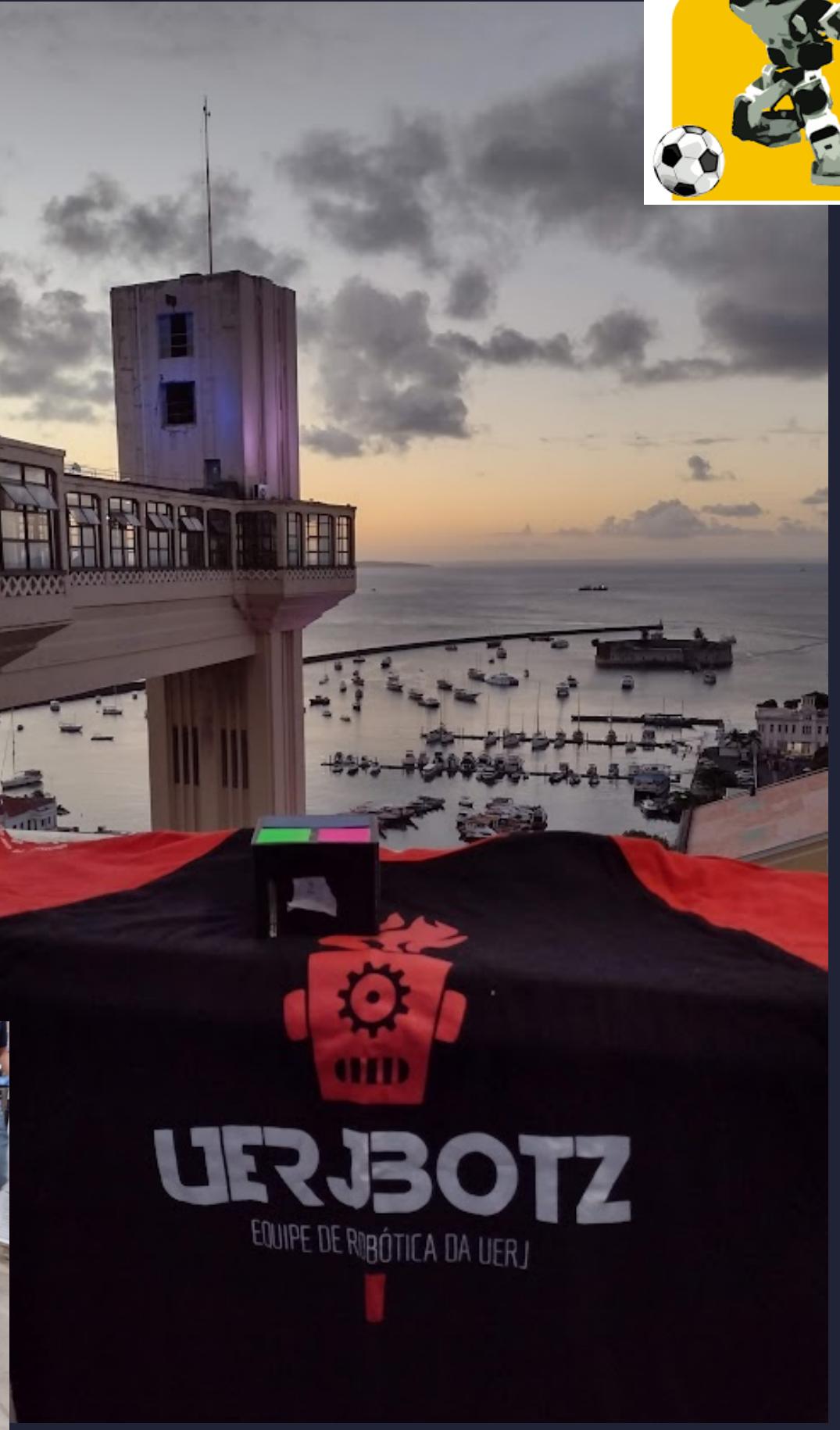
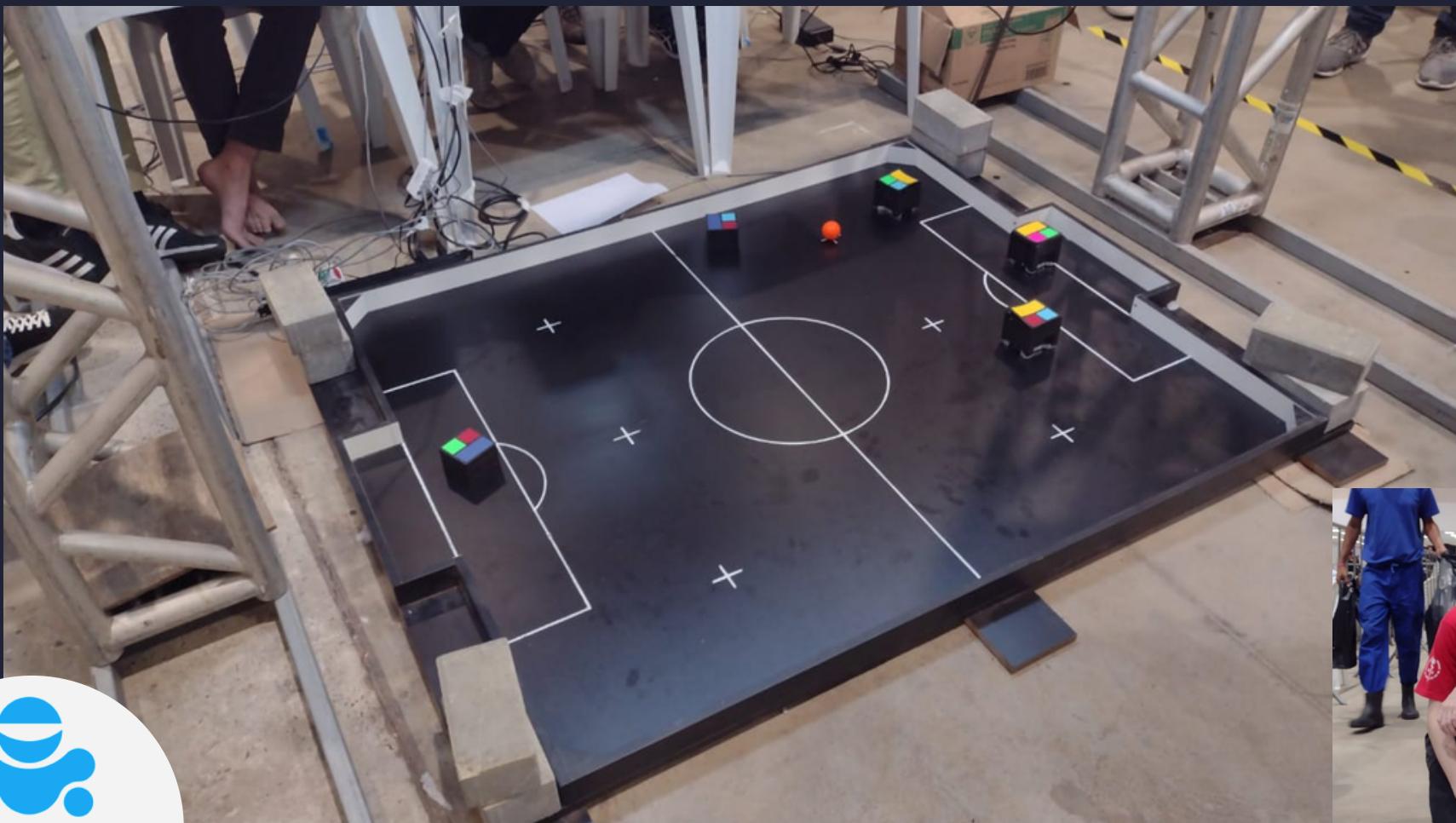
# FAPERJ

Este projeto teve apoio da FAPERJ através de edital voltado para competições educacionais.



# LARC / CBR 2023

Este ano junto a equipe UERJBotz participamos da LARC/CBR na categoria VSSS. Apesar do projeto ainda não estar 100% completo foi possível realizar partidas e aprender muito com as outras equipes.



The background features a dark navy blue gradient. Overlaid on it are several large, semi-transparent circles in orange, teal, and yellow-green. A single orange golf ball is positioned in the upper right quadrant.

# Demonstração prática!

# OBRIGADO!

PERGUNTAS?