Desafios da Construção de um Time de Futebol de Robôs

Johnathan Fercher da Rosa

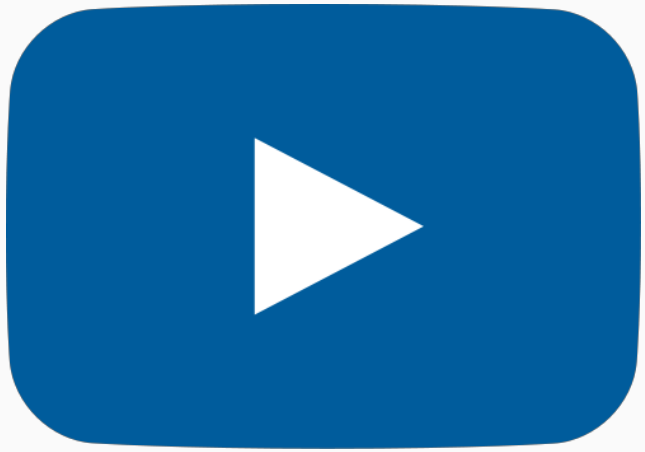
Sumário

Introdução

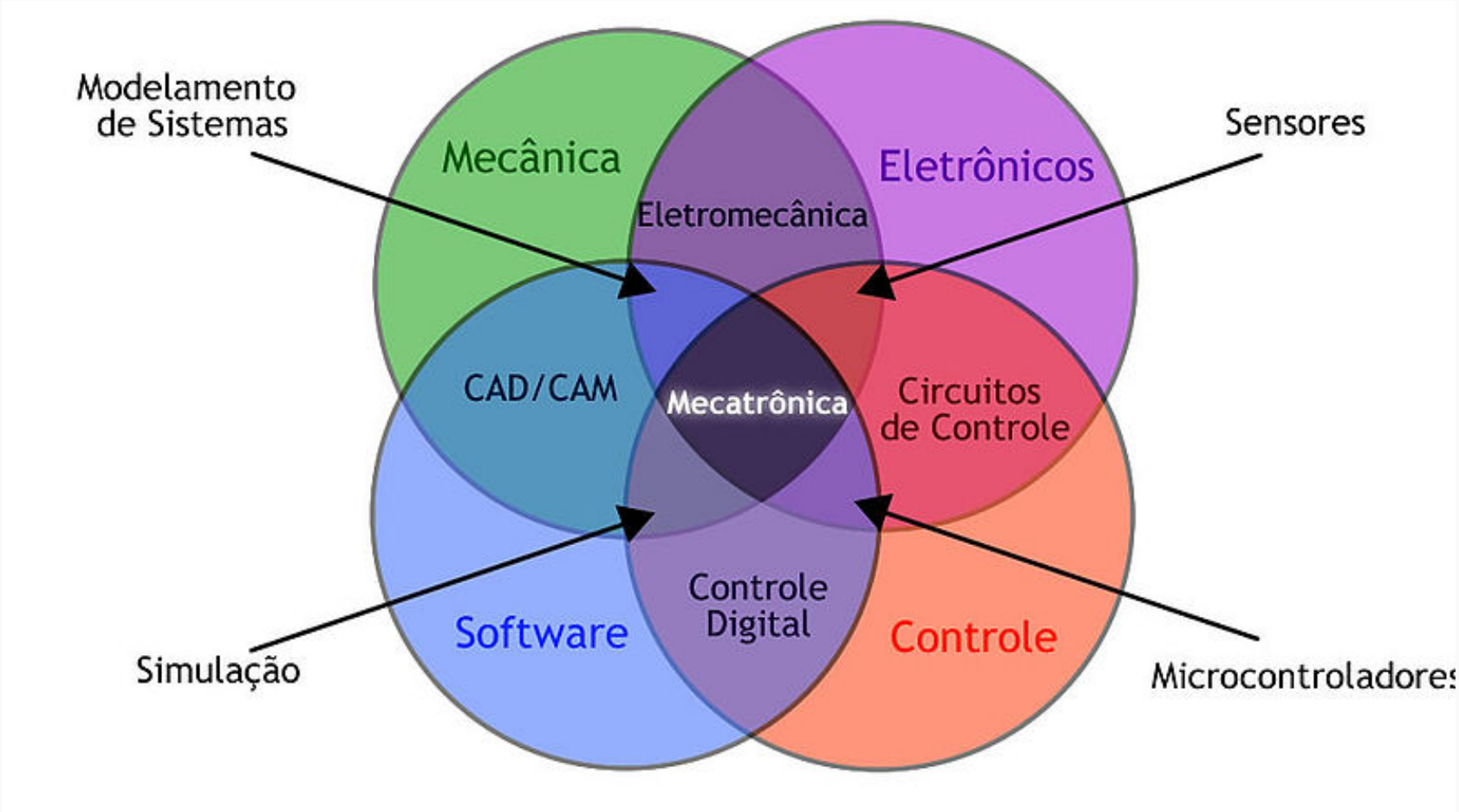
Objetivo

- Fornecer uma visão geral dos problemas envolvidos na construção de um time de futebol de robôs (IEEE Very Small Size);

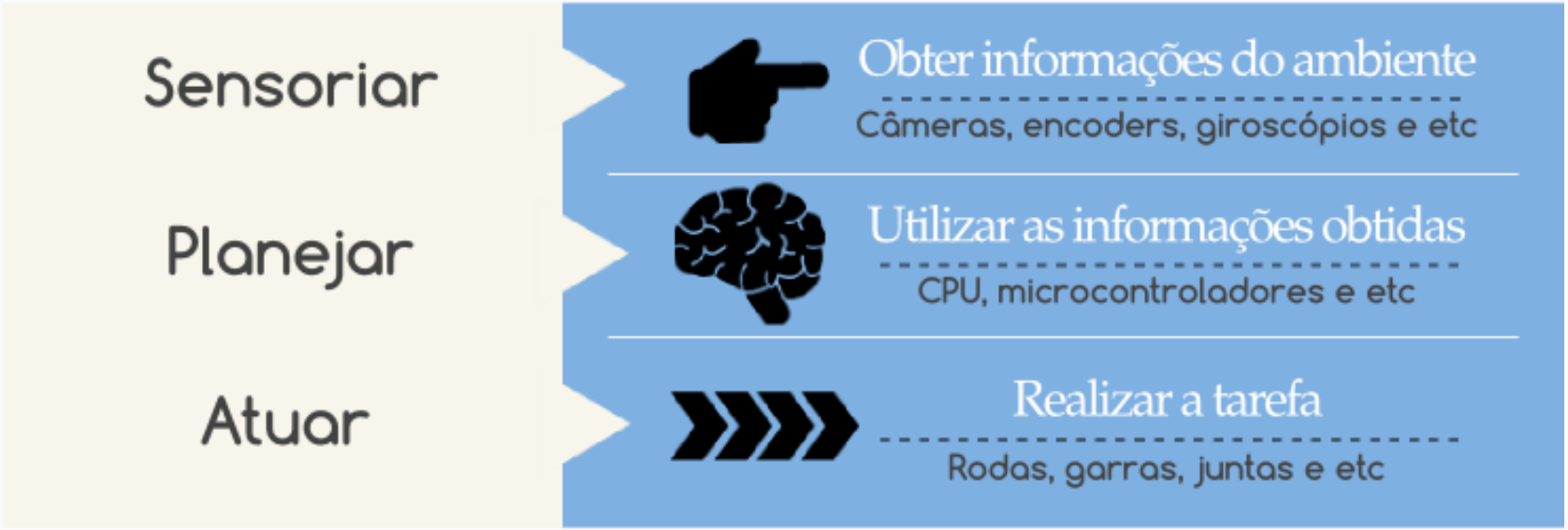
Final 2017



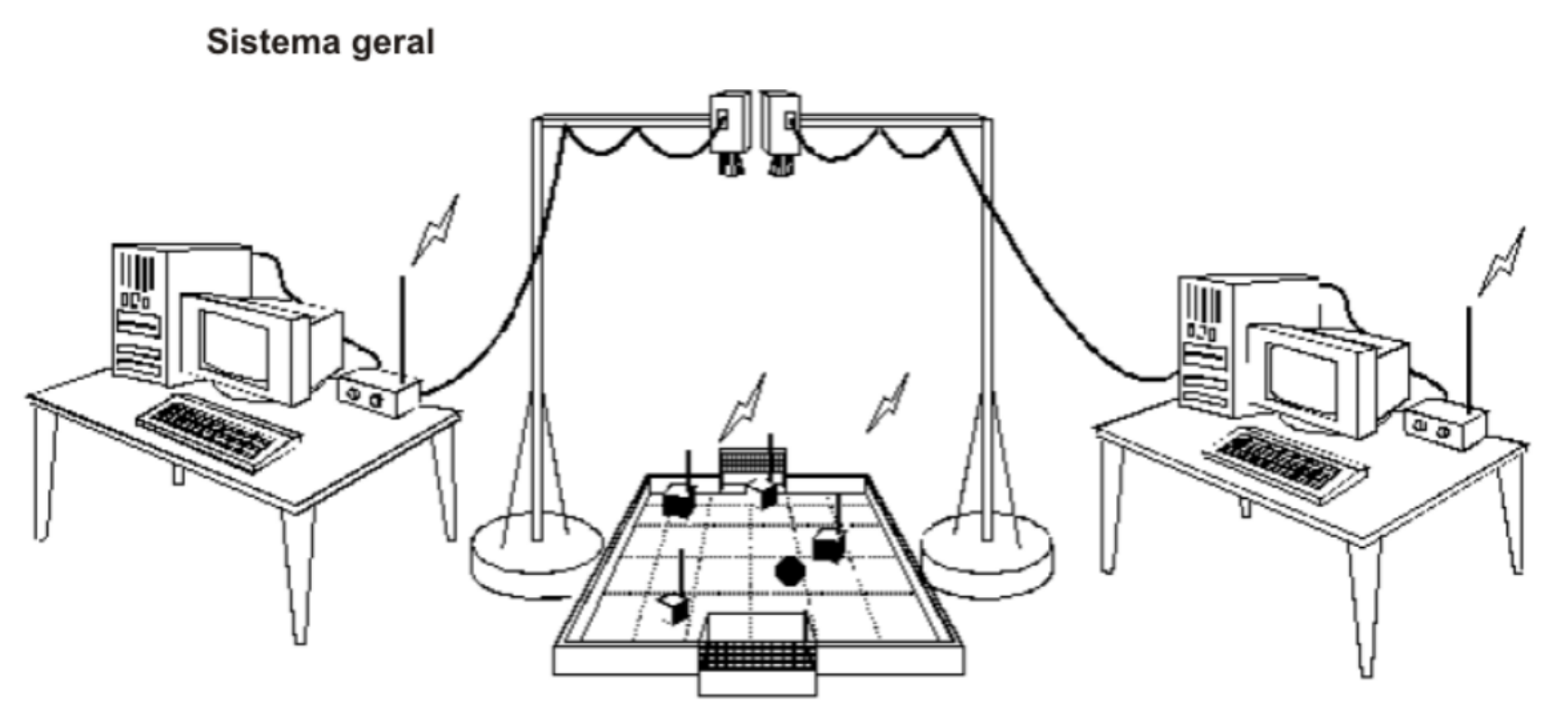
Visão Geral


Modelamento
de Sistemas
Sensores
Mecânica
Eletrônicos
Eletromecânica
CAD/CAM
Mecatrônica
Circuitos
de Controle
Software
Controle
Digital
Controle
Simulação
Microcontroladores

Fluxo Principal de um Robô

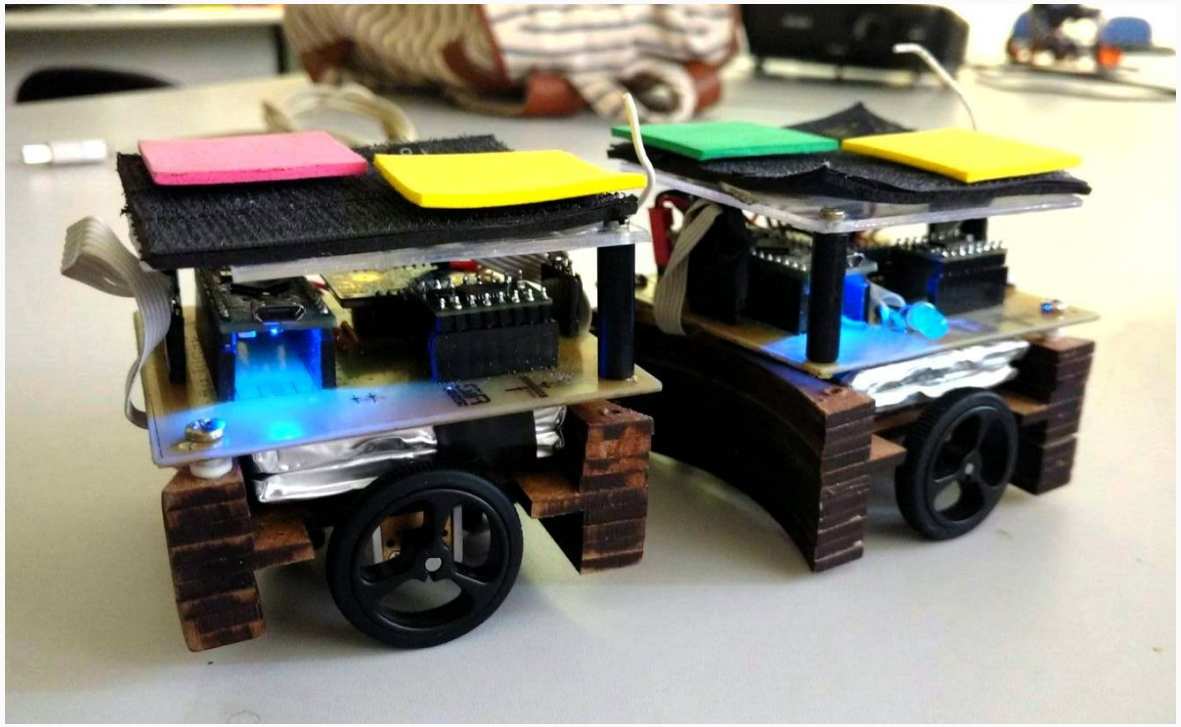


IEEE Very Small Size Soccer



- Software de inteligência em servidores;
- Localização utilizando câmeras;
- Comunicação via radio, bluetooth ou wi-fi;

Robôs



- Robôs com dimensões restritas a **8cm x 8cm x 8cm**;
- Padrões de cores definidos por regra;

Por onde começar?

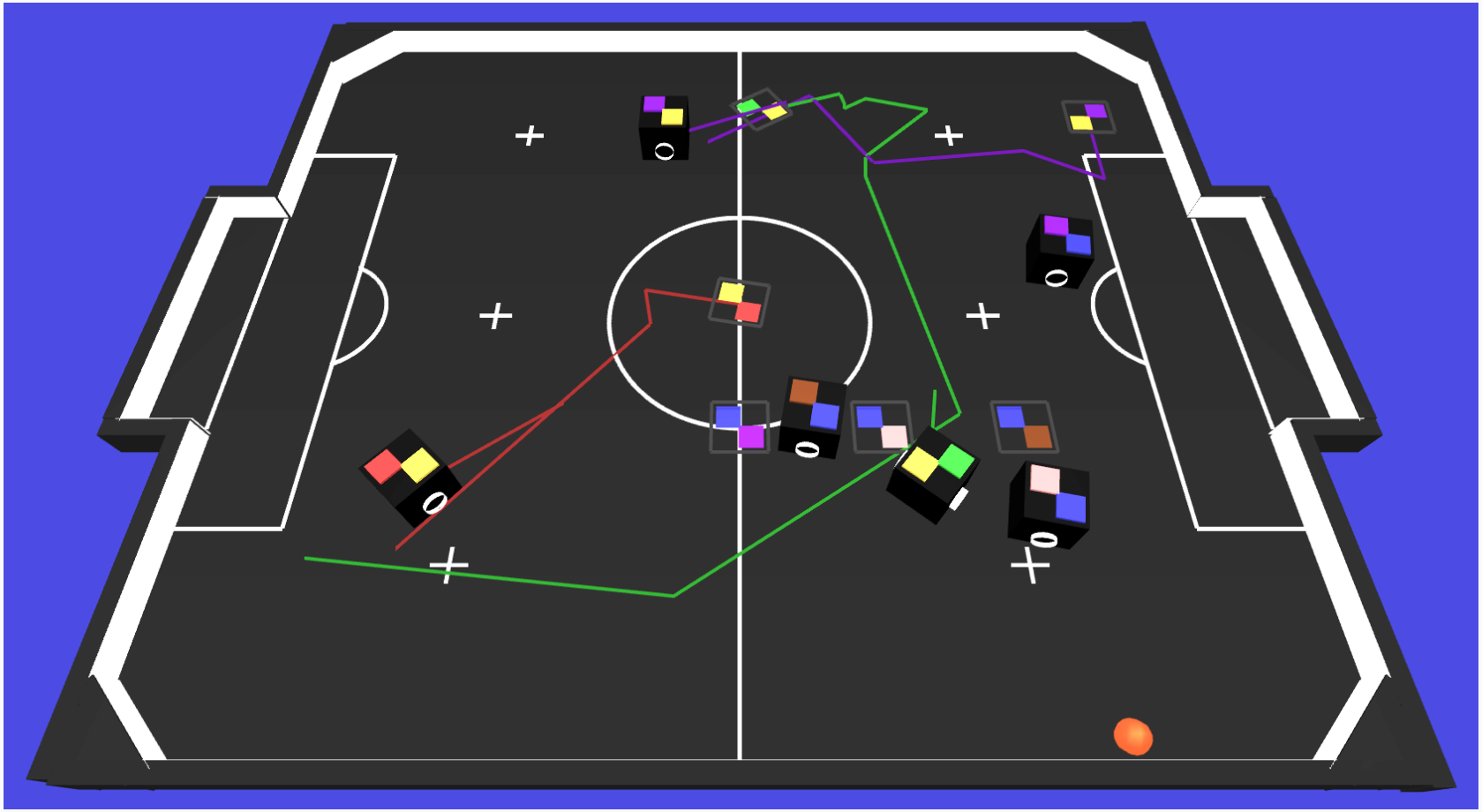
Don't Panic

- Robôs são caros;
- Muitas variáveis correlacionadas;
- Existem partes do problema que são mais importantes que outras;

Primeiro faça simulações

- Qual a velocidade mínima aceitável?
- Qual a precisão mínima aceitável?
- Em um mundo perfeito, o que é possível fazer?
- Existe um simulador?

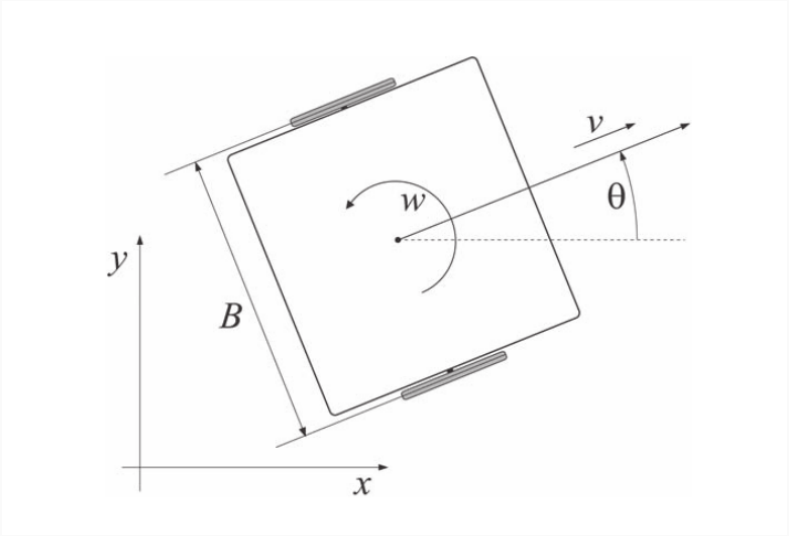
VSS-Simulator



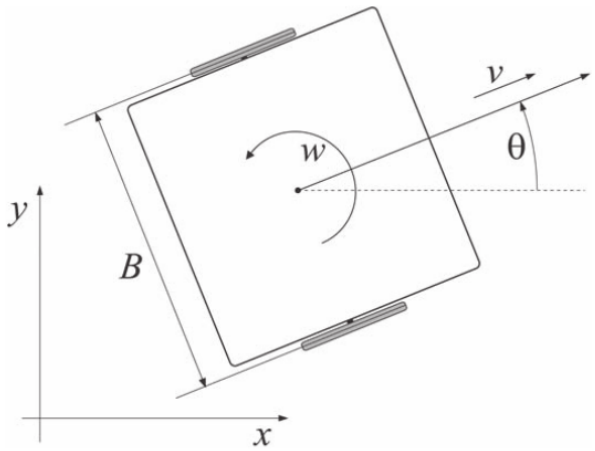
Agora que tenho uma ideia do problema

- Como posso deixar o robô mais ágil?
- Como um goleiro deve se comportar?
- Tem que existir um goleiro?
- Como construir uma estratégia?

Movimentação



Modelo Cinemático

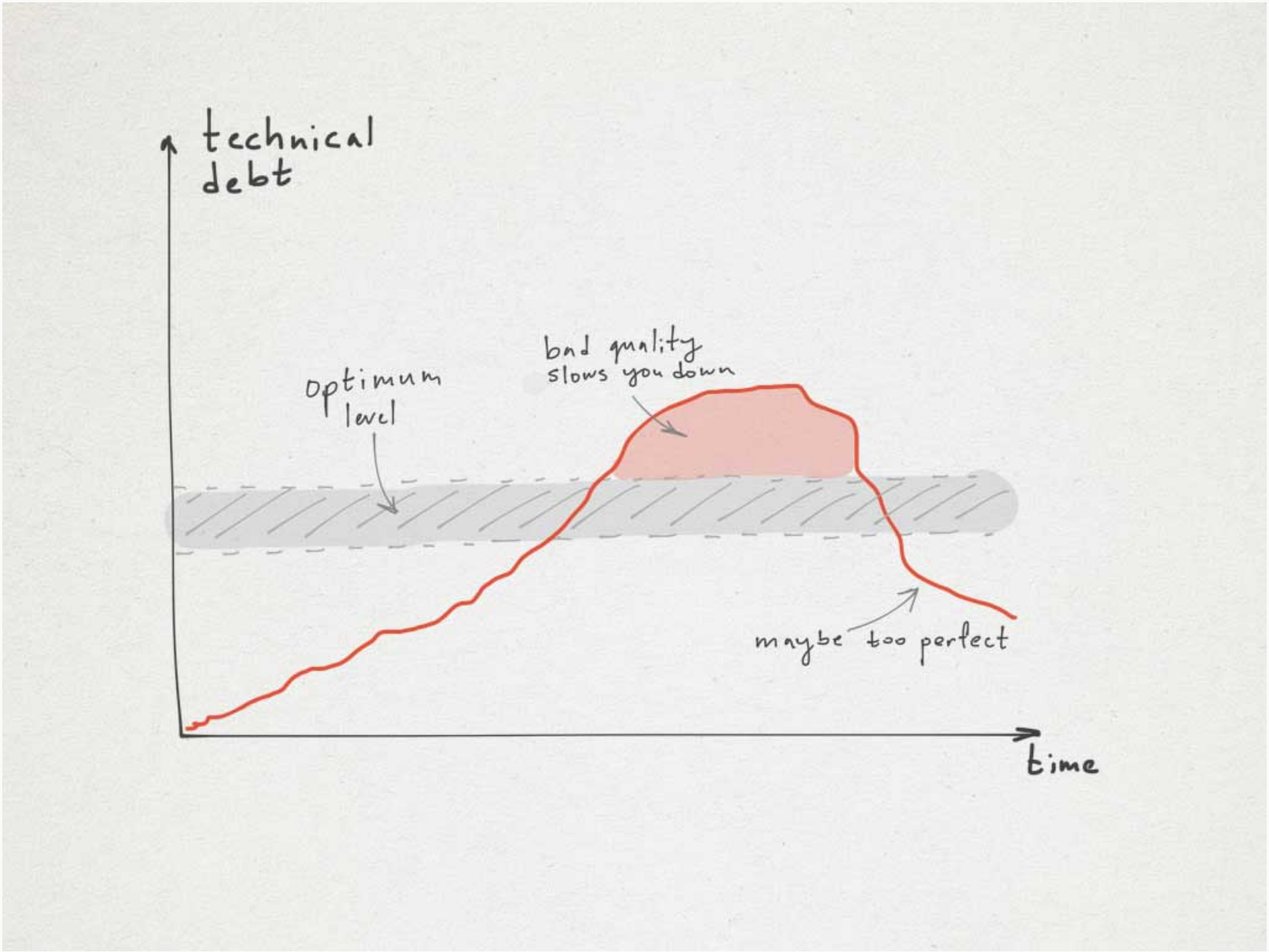


$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}$$

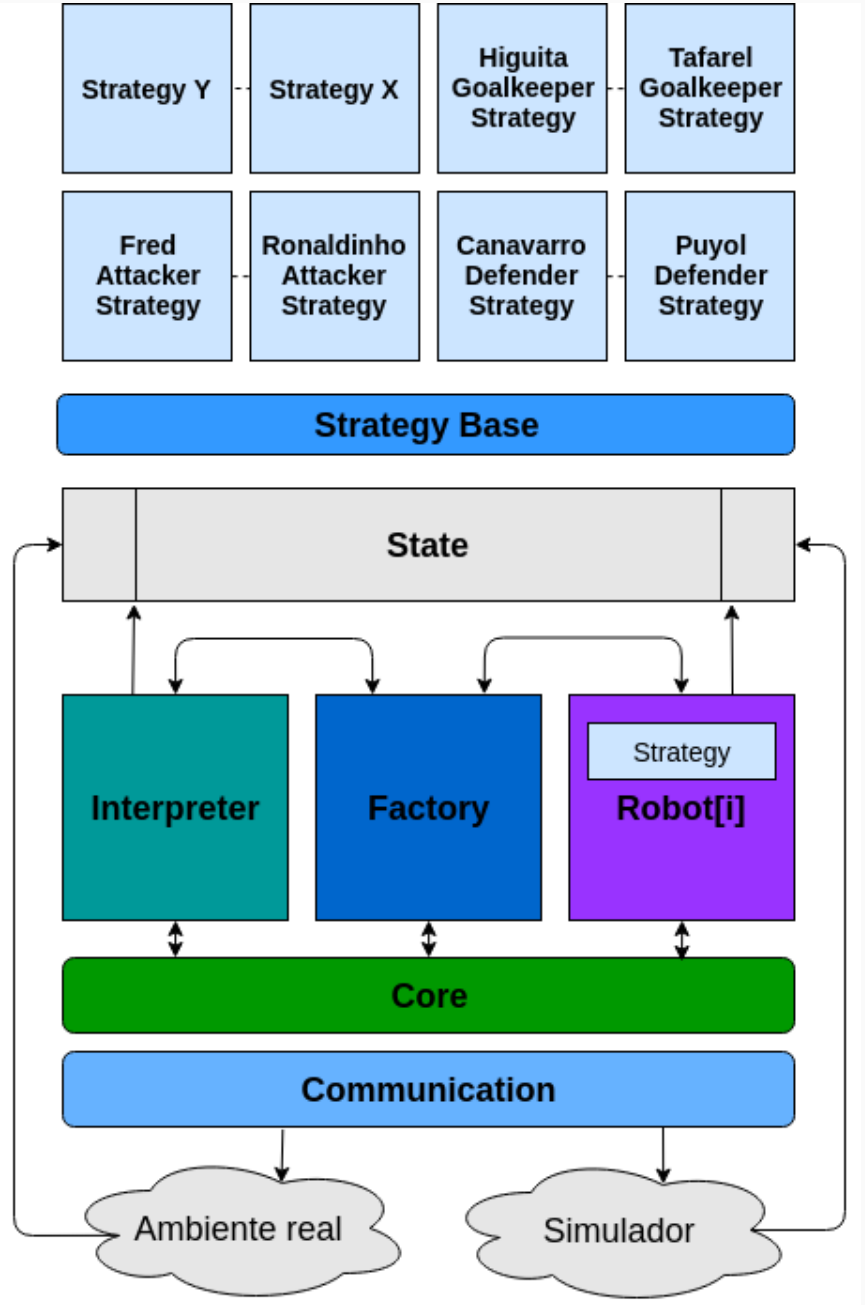
$$v_R = v + \frac{wB}{2} \quad \text{and} \quad v_L = v - \frac{wB}{2}$$

Estratégia/Inteligência

Débito Técnico

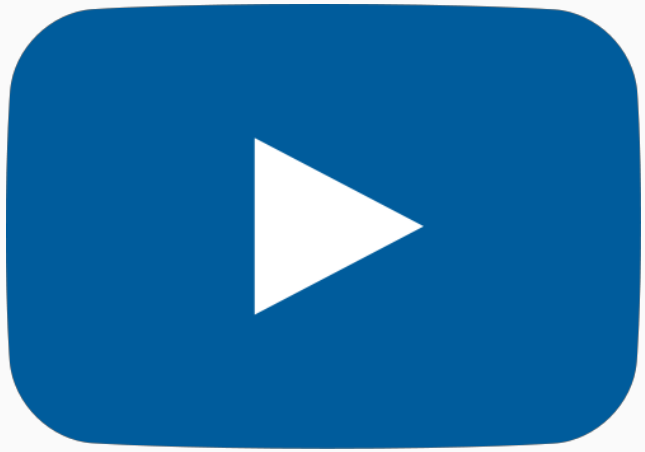


Design Patterns



Prova de Conceito

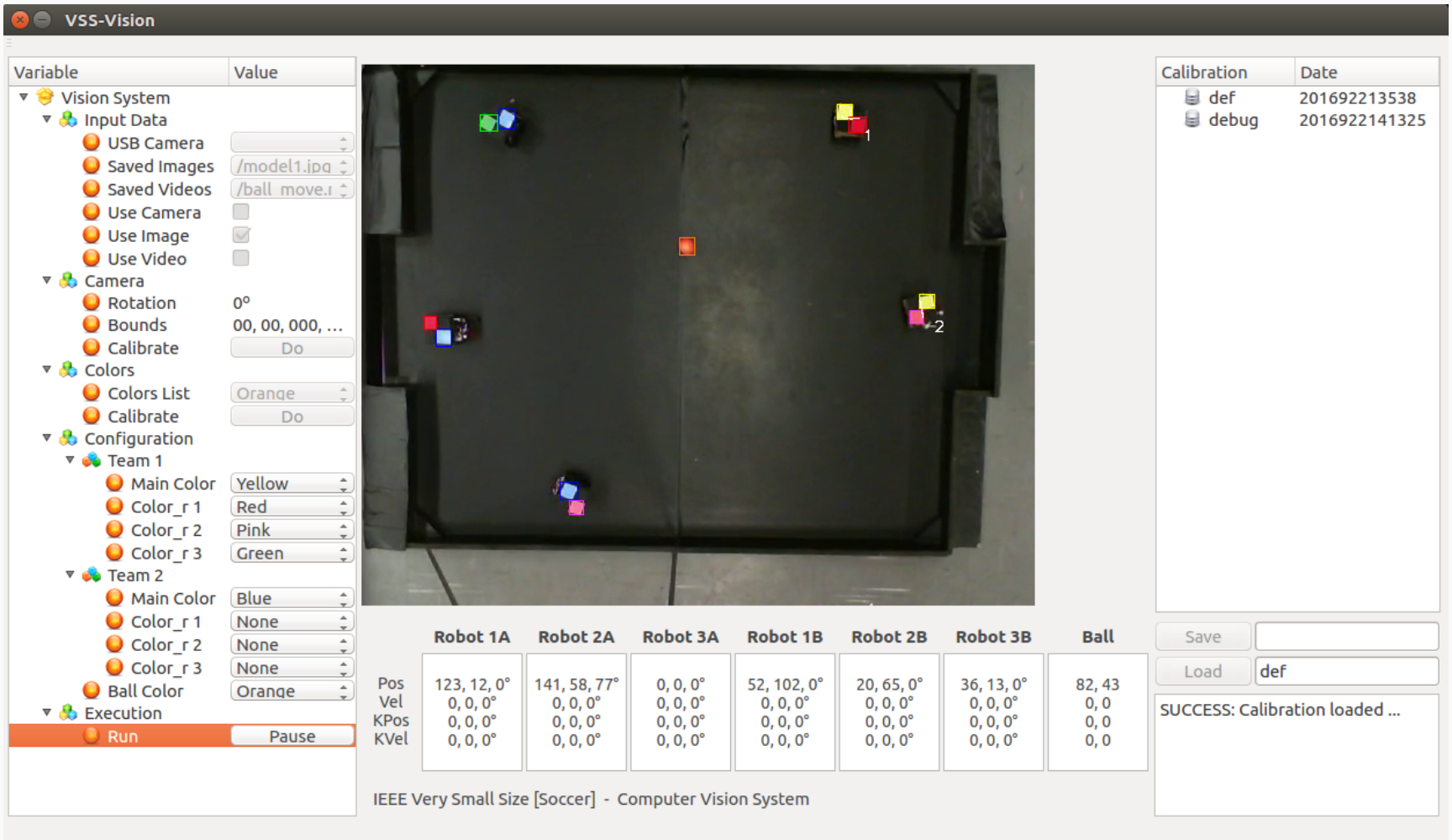
Prova de conceito de estratégia



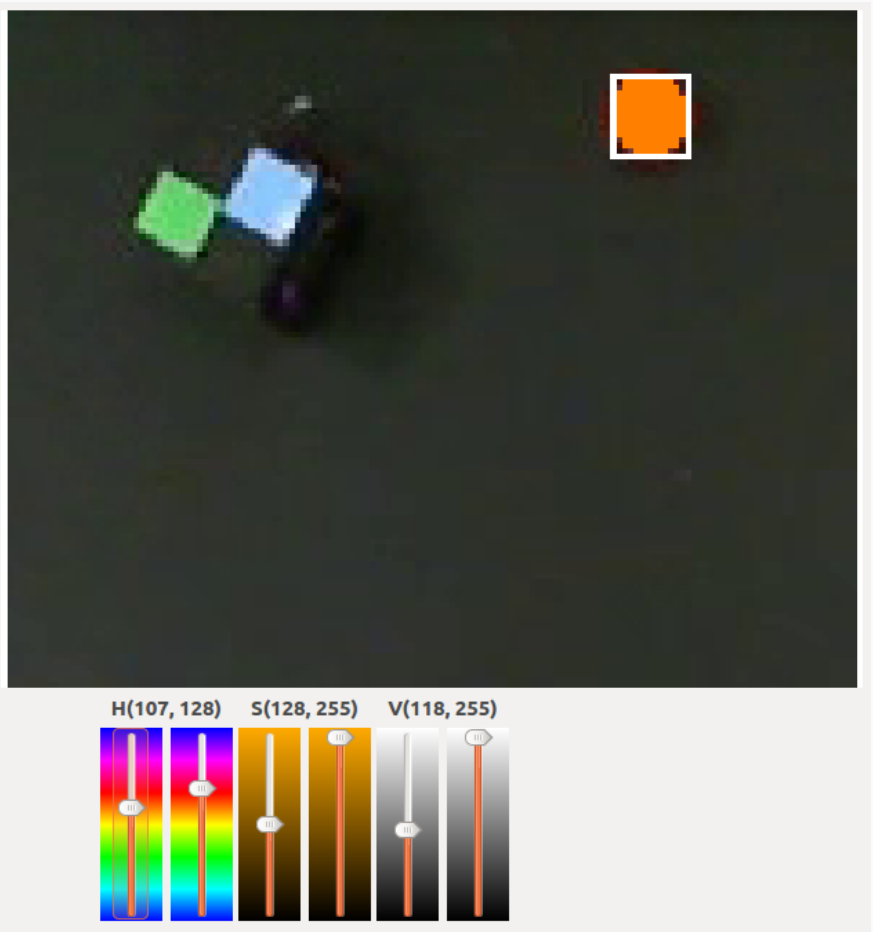
Agora que tenho uma POC

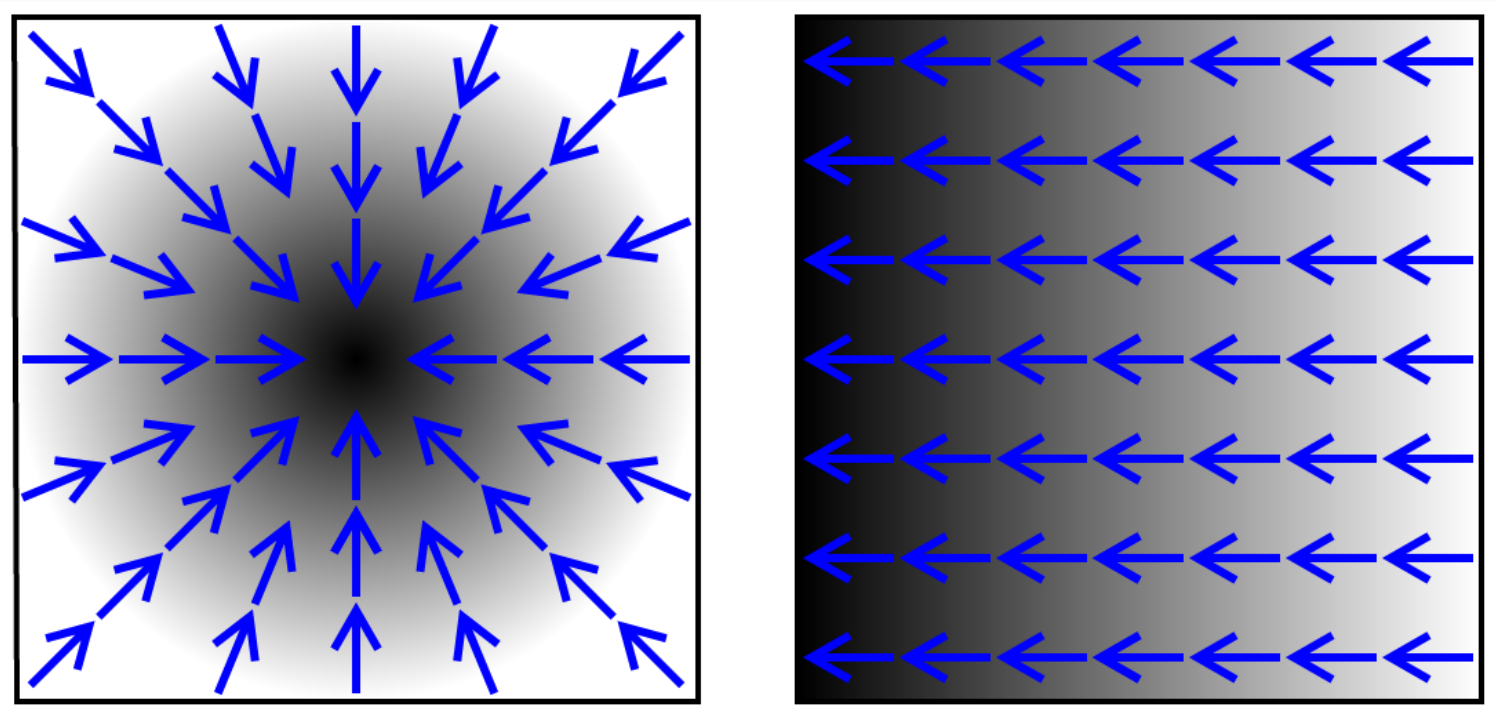
- Como posso levar minha solução para o mundo real?
- Como um robô sabe onde ele está?
- Existe algo que possa me auxiliar agora?

VSS-Vision

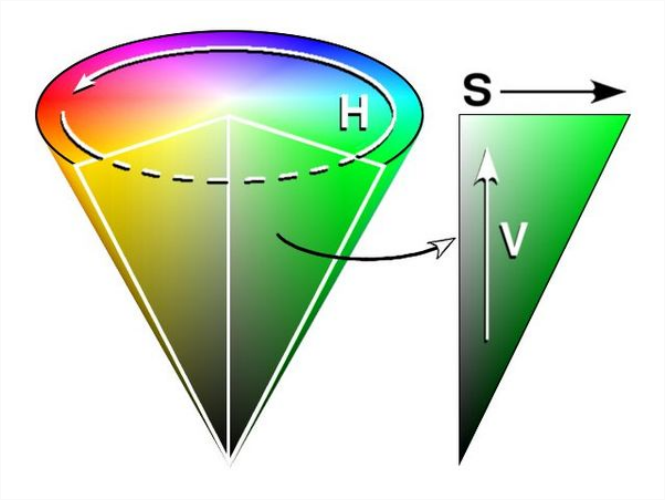

VSS-Vision
Variable
Value
Vision System
Input Data
USB Camera
Saved Images
/model1.jpg
Saved Videos
/ball_move.r
Use Camera
Use Image
Use Video
Camera
Rotation
0°
Bounds
00, 00, 000, ...
Calibrate
Do
Colors
Colors List
Orange
Calibrate
Do
Configuration
Team 1
Main Color
Yellow
Color_r 1
Red
Color_r 2
Pink
Color_r 3
Green
Team 2
Main Color
Blue
Color_r 1
None
Color_r 2
None
Color_r 3
None
Ball Color
Orange
Execution
Run
Pause
Calibration
Date
def
201692213538
debug
2016922141325
Robot 1A
Robot 2A
Robot 3A
Robot 1B
Robot 2B
Robot 3B
Ball
Pos
Vel
KPos
KVel
123, 12, 0°
0, 0, 0°
0, 0, 0°
0, 0, 0°
141, 58, 77°
0, 0, 0°
0, 0, 0°
0, 0, 0°
0, 0, 0°
0, 0, 0°
0, 0, 0°
0, 0, 0°
52, 102, 0°
0, 0, 0°
0, 0, 0°
0, 0, 0°
20, 65, 0°
0, 0, 0°
0, 0, 0°
0, 0, 0°
36, 13, 0°
0, 0, 0°
0, 0, 0°
0, 0, 0°
82, 43
0, 0
0, 0
0, 0
Save
Load
def
SUCCESS: Calibration loaded ...
IEEE Very Small Size [Soccer] - Computer Vision System

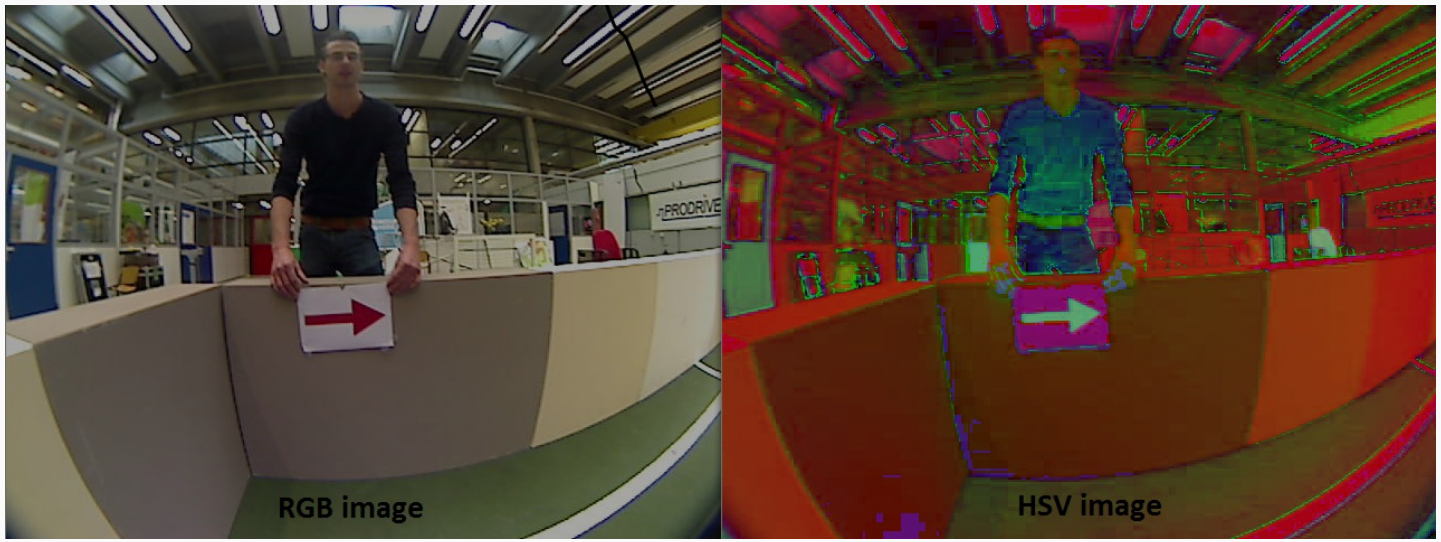
Visão Computacional



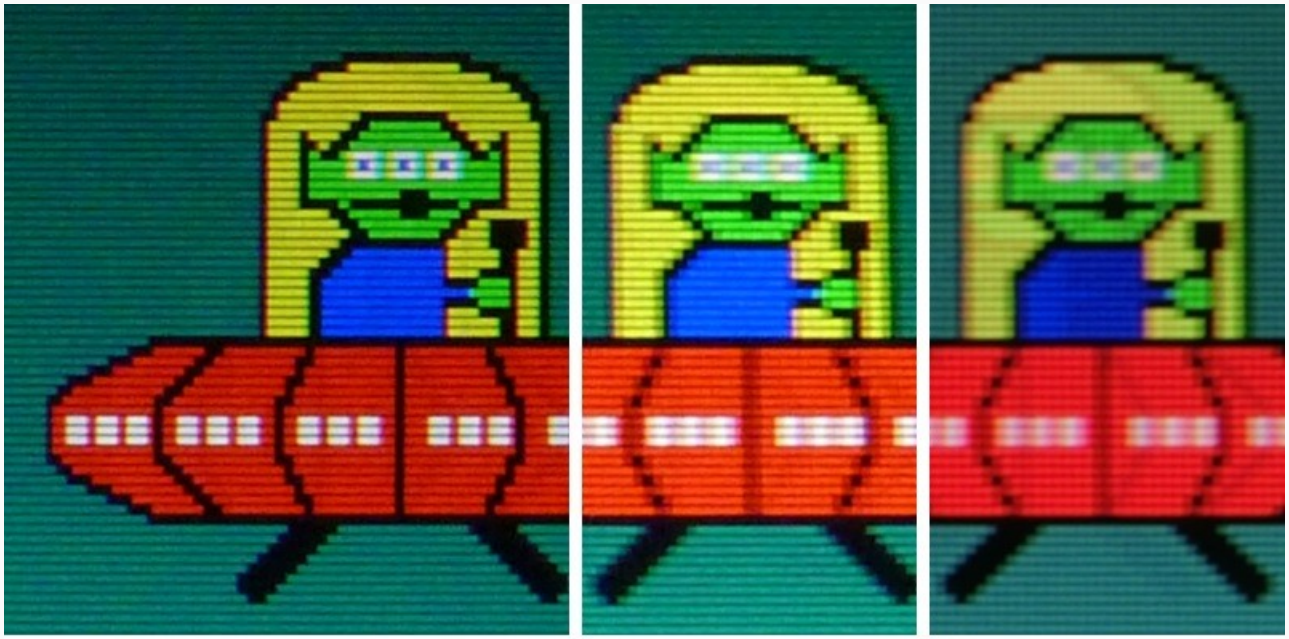


- Gradiente de luminosidade;
- Exemplos de vermelho: RGB(255, 0, 0), RGB(229, 43, 80), RGB(204, 0, 0) e RGB(195, 33, 72);





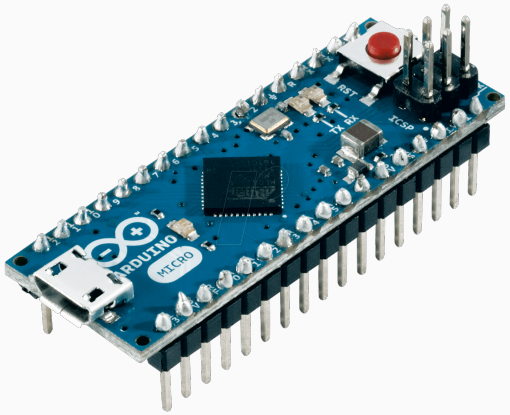
- Espaço de cores HSV;

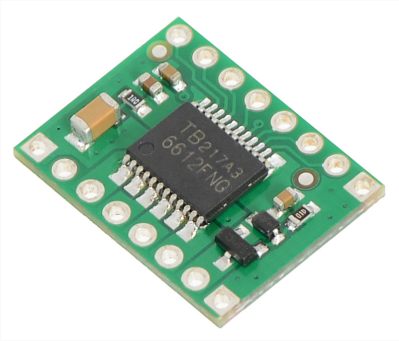


- Borrão de velocidade;

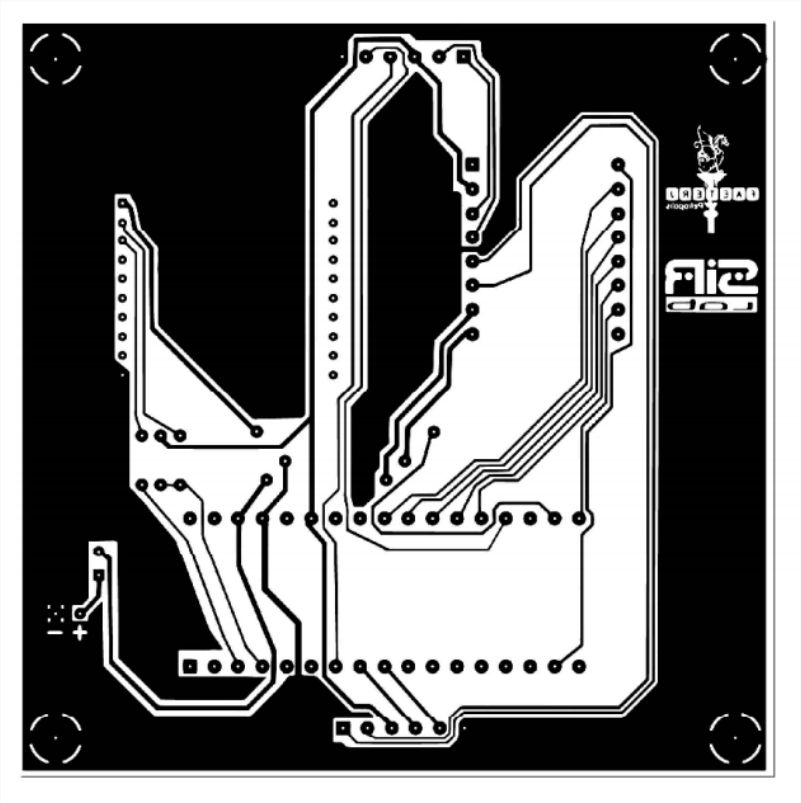
- **Uma câmera com atraso de 100ms**;
- **Um robô andando a 2m/s em linha reta:**
  - Erro de 5 centímetros;
- **Um robô girando 2 vezes por segundo:**
  - Erro de 72 graus;

Componentes

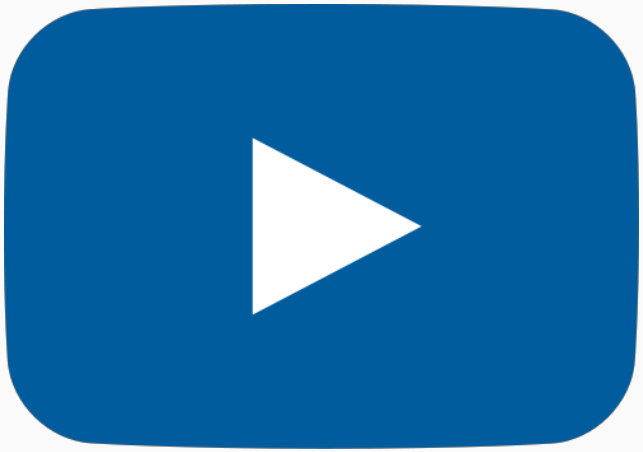




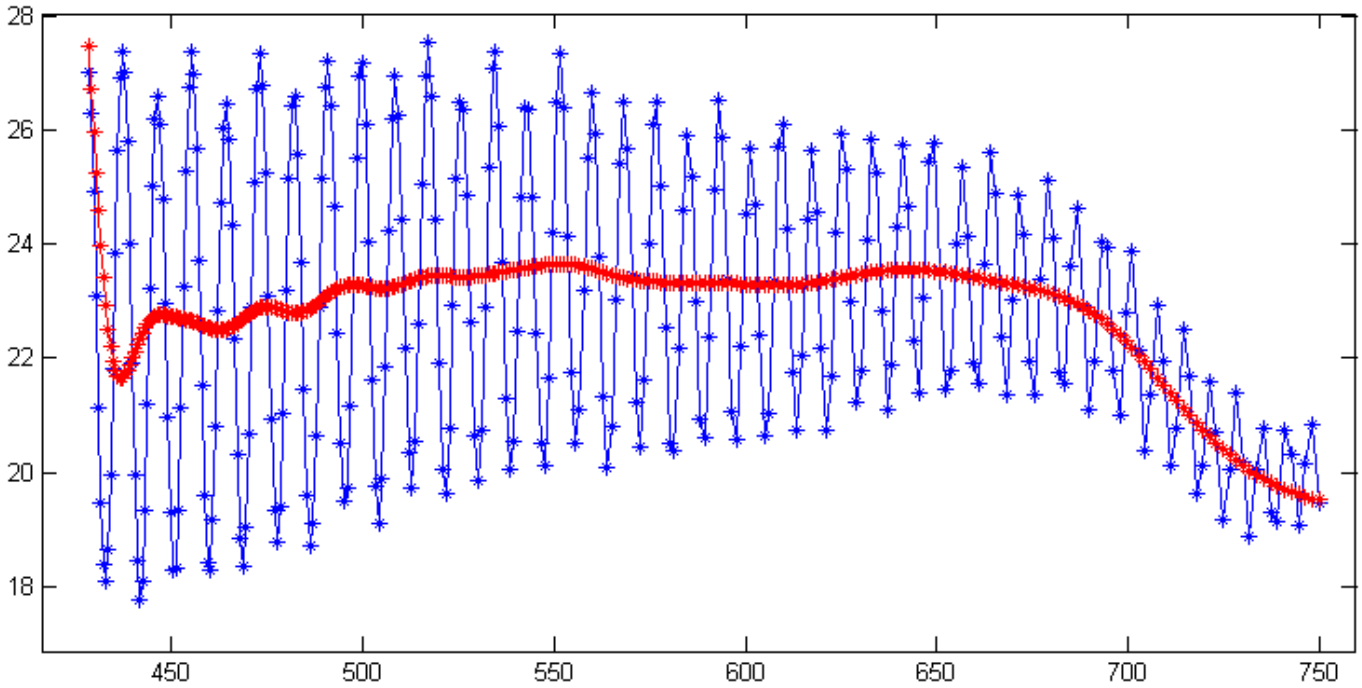
Componentes



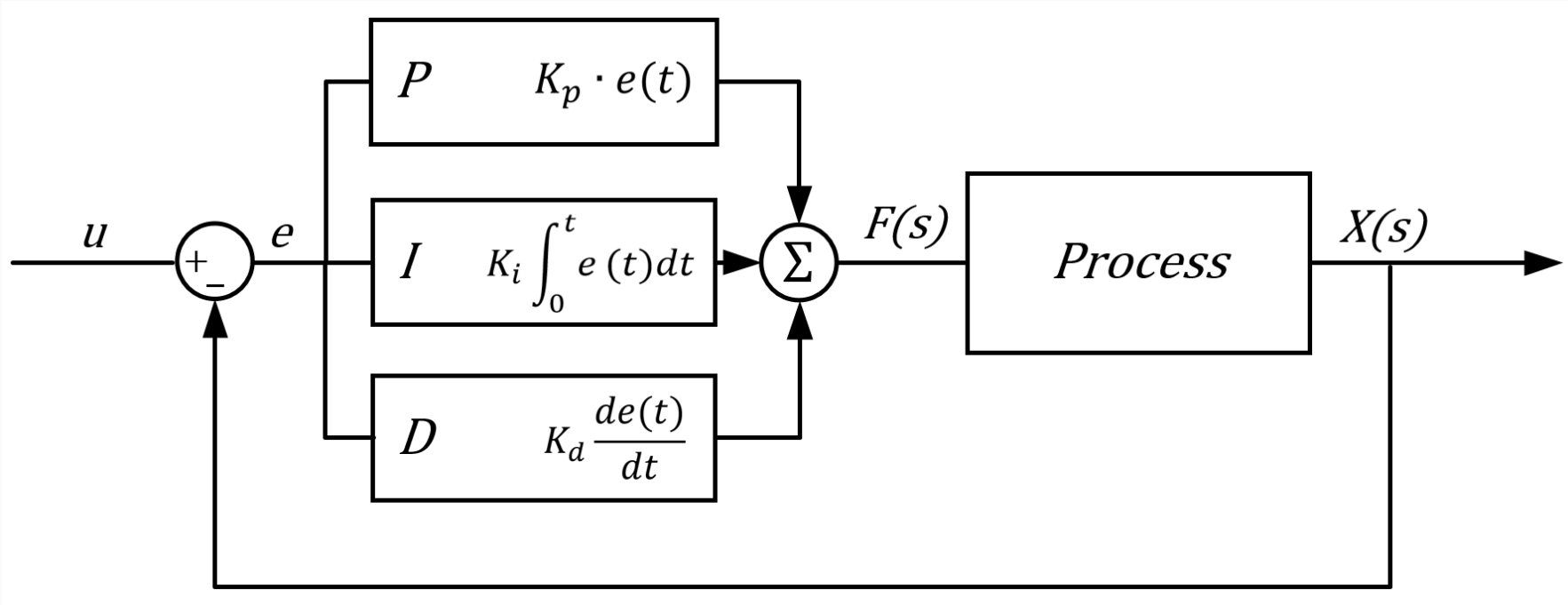
Fails



Filtragem de Sinais



Controle PID



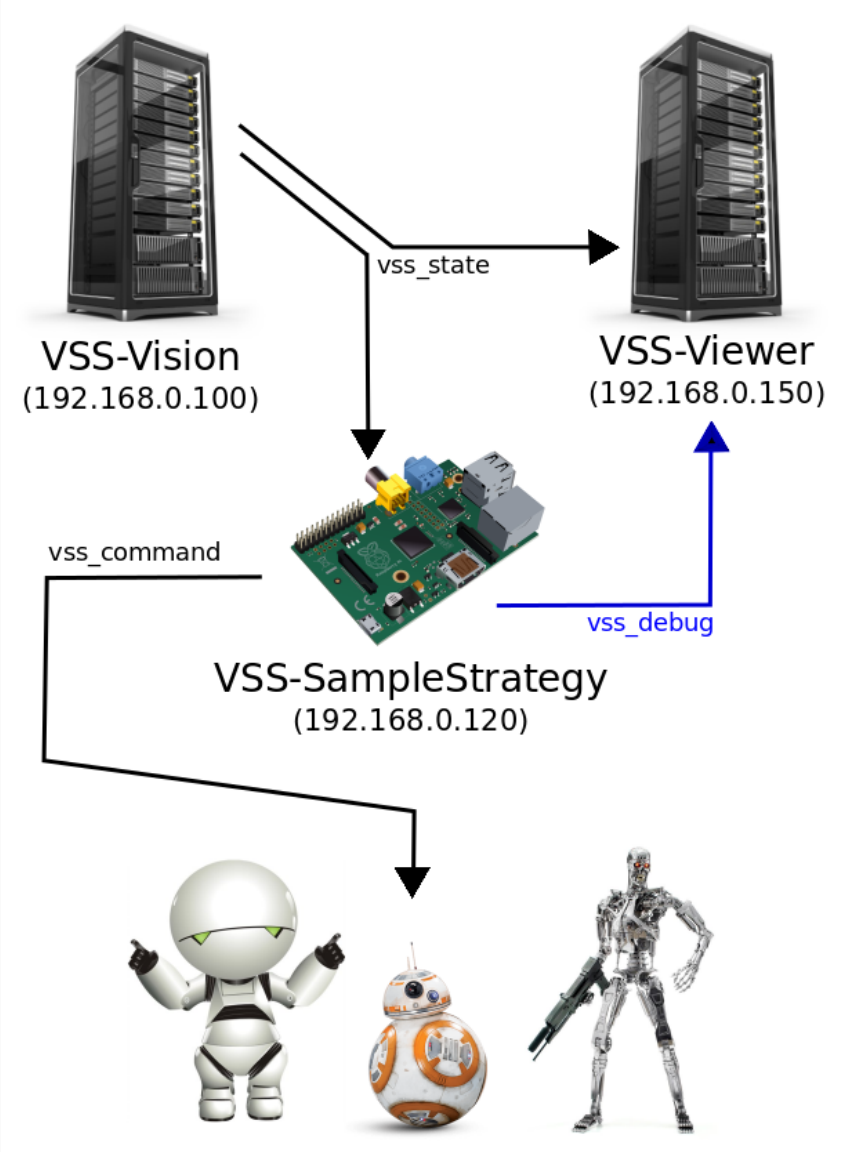
$$e(t) = set(t) - act(t) \tag{1}$$

VSS-SDK

VSS-SDK

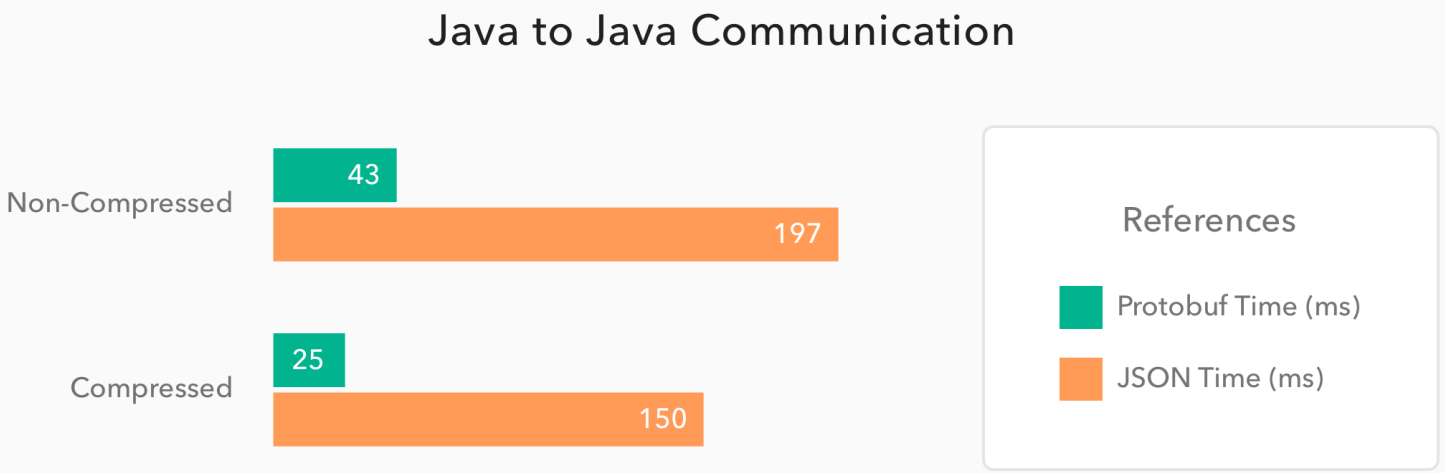
- Sistema de visão computacional;
- Simulador de partidas;
- Ambiente visual 3D;
- Controle via joyticks USB;

Distribuído

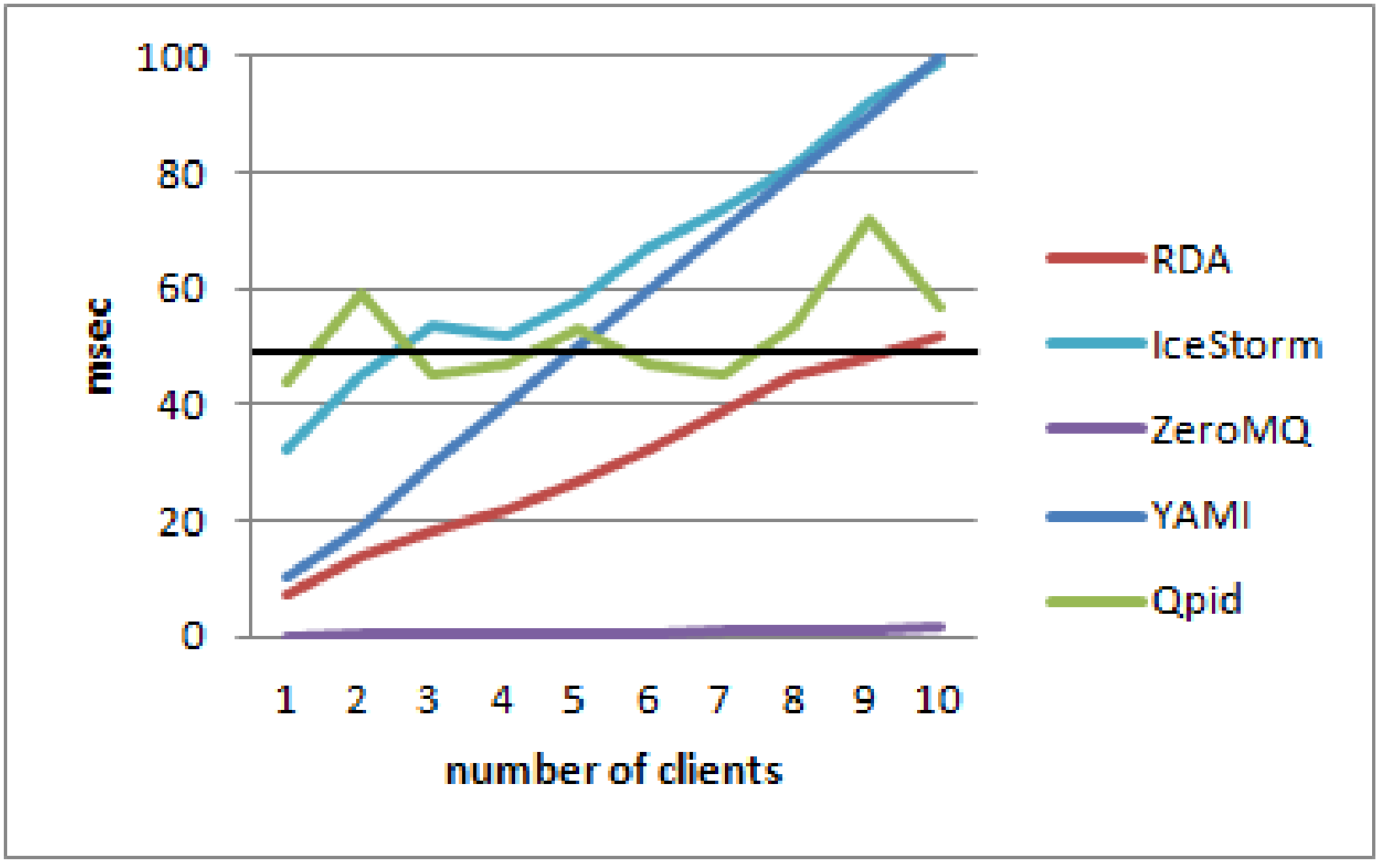


Serialização e Comunicação

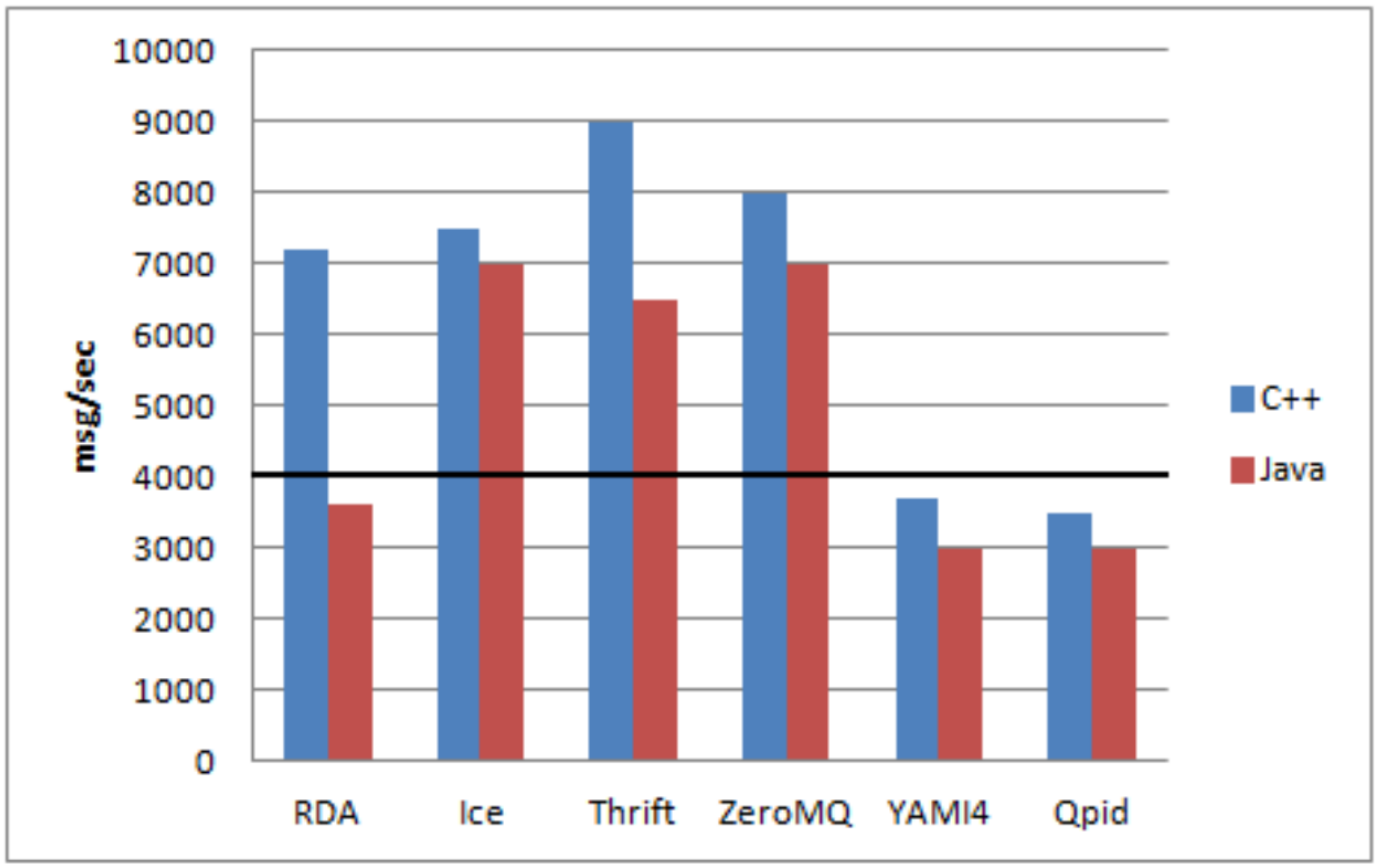
Benchmark Protobuf x JSON



Benchmark ZMQ



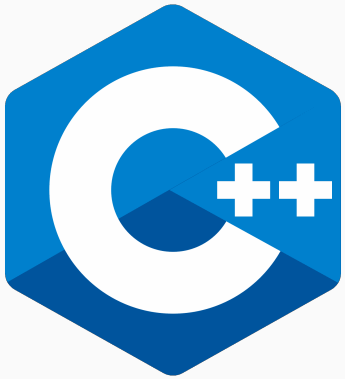
Benchmark ZMQ

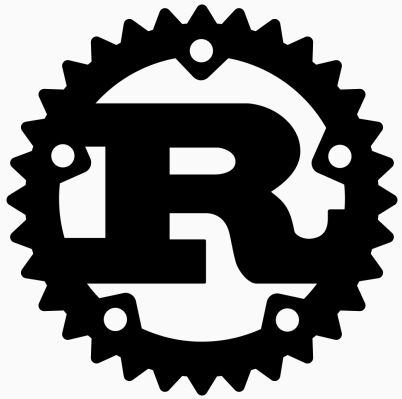


Serialização e Comunicação

- **Delay em localhost**: 0.00035 segundos (350 microsegundos)

Biblioteca de integração





Equipes que utilizam

Backlog items

- Refactor do sistema de visão computacional;
- Aumentar a cobertura de testes dos projetos;
- Configurar integração contínua com docker;
- Publicar no repositório oficial do Ubuntu/Debian;
- Refactor do simulador para utilizar o Gazebo;

Obrigado