CYPHER – Robô de Rescue Maze da Escola de Robótica e Mecatrônica ABC Marcelo Salles

Felipe Catapano Emrich Melo, André Luiz Buzacarini Schulman, Gabriel Mari Salles, Marcelo Ribeiro Salles

Abstract – This article presents the development of an Arduino-controlled autonomous junior robot with emphasis on RoboCup 2019 and Latin American Robotics Competition 2019. All key components and critic design choices, as well as their purposes are detailed in this document.

I. INTRODUÇÃO

Com um nível mais amplo de experiência na categoria Rescue Maze e com um projeto pela primeira vez desenvolvido com todas as etapas, nosso robô competidor para a LARC de 2018 (edição ocorrida em João Pessoa, PB) mostrou-se apto a resolução de todas as tarefas propostas pela categoria, conquistando assim o título de primeiro lugar no evento. No entanto, percebemos de imediato a necessidade de um novo projeto, a fim de obter um alto rendimento na RoboCup 2019, campeonato mundial para o qual nos classificamos (este ocorrerá em Julho deste ano, na cidade de Sydney, Australia), e para competir novamente na LARC com um robô um pouco mais avançado, também montado a partir do zero. Este robô recebeu o nome de Cypher.

Isto resulta na melhoria geral de diversos aspectos do projeto, seja na parte de software, mecânica ou eletroeletrônica, voltados para uma arena com configurações de maior complexidade em relação às da LARC. São esses avanços e conceitos que pretendemos enfatizar neste artigo.

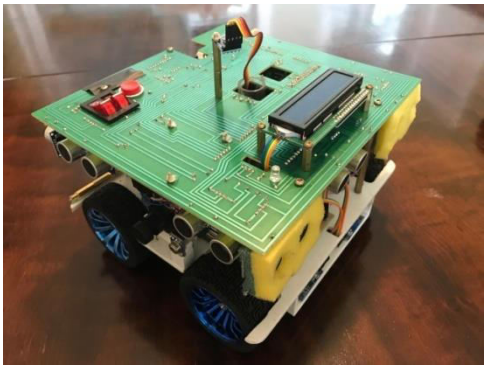


Imagem 1 - ENNARD, o robô de 2018

II. ESTRUTURA E CIRCUITOS

Para este projeto, obtivemos acesso total a uma impressora 3D UPBOX+, o que nos permitiu conseguir peças mais precisas e duráveis do que nunca, a um preço acessível. O chassi e suportes deste robô foram desenhados com os softwares AutoCAD e Fusion 360, ambos da Autodesk, e impressos em PLA branco ou vermelho. O chassi é dividido em uma parte superior e outra inferior. O tubo e kits do nosso depositador de resgate, no entanto, foram feitos em acrílico para que não houvesse qualquer rebarba (são identicos aos que estão presente em Ennard).

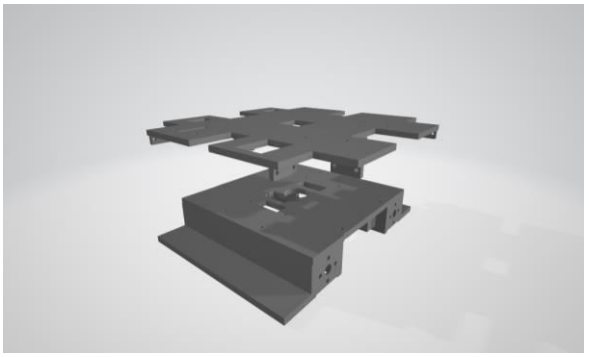


Imagem 2 – Chassis (Superior+Inferior)

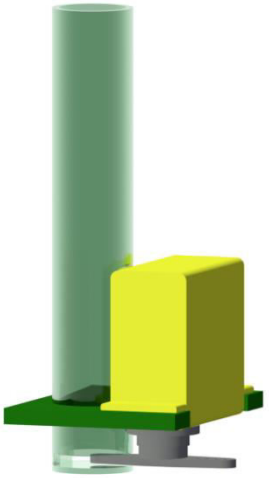


Imagem 3 – Kit Dropper

Duas placas de circuito integrado/impresso, desenhadas com AutoCAD, reúnem os fios dos três andares do robô (inferior, mediano e superior), que são inter-conectados por flat cables feitos sob medida. Isto permite um design mais compacto como um todo, e uma eletrônica mais confiável. Diferentemente de Ennard, o circuito superior não está mais exposto (está protegido por plástico), algo que julgamos ser uma mudança necessária para evitar novos problemas.

Levando isso em conta, o campo de visão de cada sensor voltado para fora foi também simulado para garantir que nenhuma peça haveria de ser necessariamente realocada por obstrução (e também para manter uma maior eficiência de cada sensor).

Em termos de tamanho absoluto, Cypher possui 19,5cm de largura, 19cm de comprimento e 14,5cm de altura.

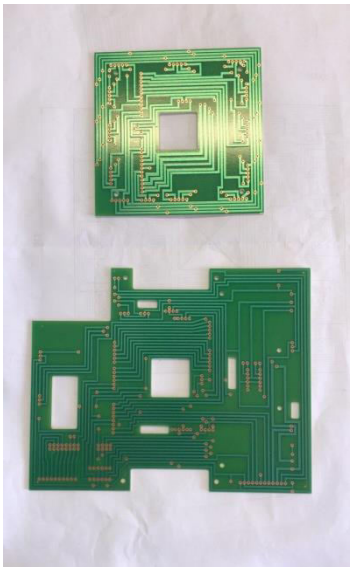


Imagem 4 – PCIs (primeiros protótipos)

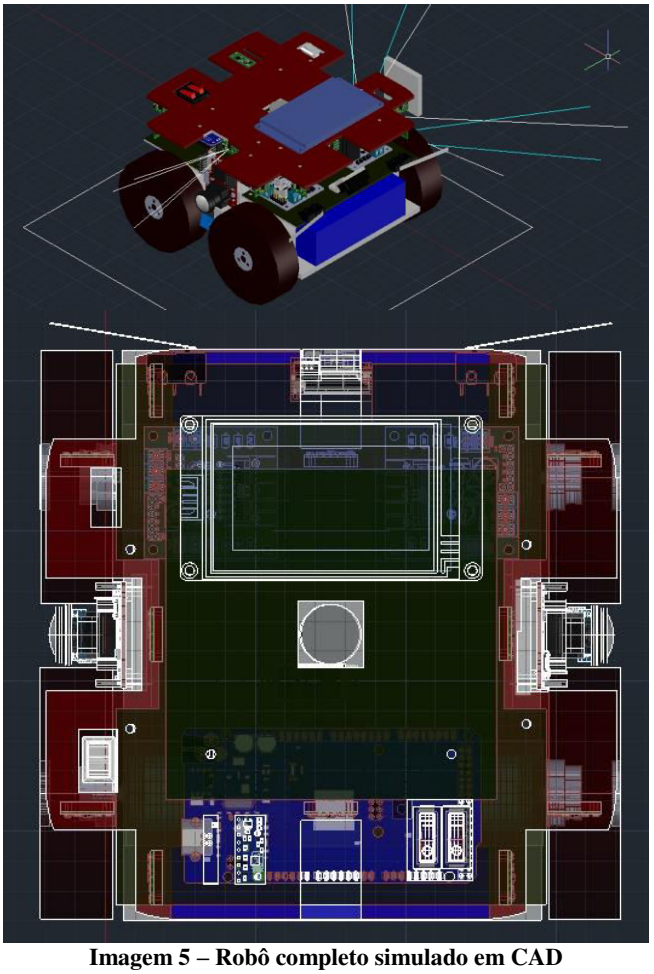


Imagem 5 – Robô completo simulado em CAD

III. CONTROLE E ATUAÇÃO

Decidimos utilizar uma embarcada da Arduino Mega como plataforma central escolhida pela simplicidade de uso, e garantia de funcionamento correto: A Blackboard Mega 2560 R3, da Robocore. Fabricada no Brasil, esta placa funciona de forma idêntica ao arduino convencional, porèm possui uma maior capacidade de corrente por pino e uma proteção contra inversão de polaridade. Os motores (ligados a dois drivers L298N) e a Blackboard com sensores (ligados juntamente com um regulador de tensão visto a grande quantidade destes) formam dois circuitos separados, cada um ligado por uma bateria LiPo de 12 Volts com 2.2 Amperes.

Quatro motores DC de 12V da Pololu controlam as rodas, dessa vez contendo uma redução de alta força e precisão: 99 para 1.

Um servo motor genérico SM-S2309S (presente no kit Arduino Multi-language Starter Kit) foi utilizado para o depósito dos kits de resgate pelo baixo peso e custo para seu tamanho.

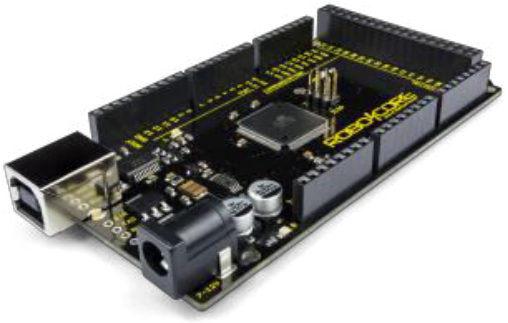


Imagem 6 – BlackBoard Mega 2560 R3

Imagem 7 – Motor com encoder

Como este robô estará sujeito a ter de passar por obstáculos maiores, precisamos de rodas bastante robustas, voltadas para o offroad. Escolhemos um conjunto de rodas de borracha com diâmetro total de 73mm (8mm maior em relação às de 65mm de Ennard), feitas para automodelos 1/10.

Imagem 8 – Rodas OffRoad 73mm

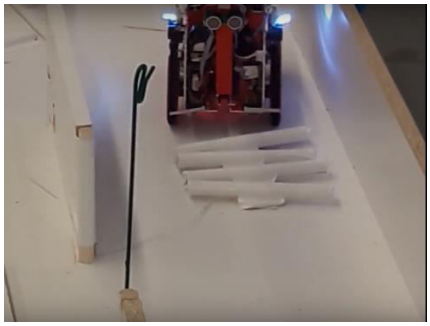


Imagem 9 – Exemplo de obstáculos fixos (RoboCup 2018)

IV. SENSORES E SINALIZAÇÃO

Para possibilitar a identificação das vítimas visuais, decidimos manter as duas pequenas placas programáveis OpenMV M7. Alguns parâmetros como exposição ou equilíbrio de branco podem ser ajustados de maneira automática, o que facilita bastante o trabalho do programador em algumas situações. Por possuírem conexão externa e uma IDE própria, calibrar as placas pode ser feito de maneira direta e é possível retirá-las do robô facilmente caso necessário. As OpenMVs comunicam-se com a Blackboard por serial UART.

Agora nossas câmeras possuem lentes grandes angulares além de estarem um pouco mais recuadas, fazendo o campo de visão horizontal de cada uma subir de aproximadamente 70 para 100 graus.

Seu software também está bem mais refinado, com melhores filtros de tamanho, proporção e posição das letras.

Imagem 10 – OpenMV M7 com lente grande angular

Doze sensores de distância com tecnologia LiDAR estão presentes, e interligadas por SMBus. Percebemos que esta tecnologia é bastante superior a sensores mais populares como sonares ou sharps, pela sua precisão, alcance mínimo e máximo e pelo seu tamanho. No caso, o módulo que usamos foi o Pololu VL53L0X (detecta até 200cm).

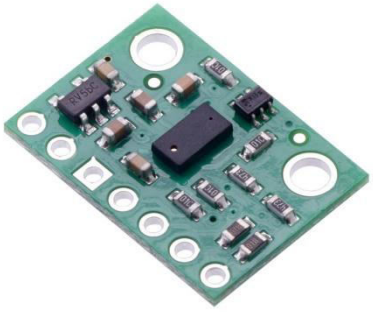


Imagem 11 – LiDAR ToF VL53L0X

A medição inercial é realizada por um BNO055: Módulo que além de reunir dados de 3 sensores diferentes formando um sistema 9DOF, possuir um processador ARM integrado cujo reúne e filtra os dados coletados em tempo real e devolvendo apenas os vetores finais que buscamos para cada eixo de rotação. O sensor também consegue informar dados em forma de quaterniões, característica que não utilizamos atualmente mas pretendemos explorar em breve.

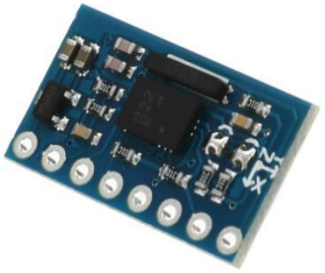


Imagem 12 – IMU Bosch BNO055

Além dos encoders presentes nos motores, três outros tipos de módulos de sensores completam o sensoriamento de Cypher: Dois dos já clássicos MLX90614 (de temperatura infravermelha), dois TCRT5000 (de refletância ativa infravermelha), e dois sensores de toque frontais.

E em questão de sinalização, uma tela gráfica Nextion de 3.5 polegadas facilita ao juíz e a nós mesmos o feedback de algumas ações-chave que dão retorno em pontos (como identificação de vítimas ou travessia de rampas) . A ideia inicial desta tela era a de possibilitar a visualização em tempo real do mapeamento realizado pelo robô, idéia que acabou sendo descartada pelas limitações técnicas dela. Decidimos mantê-la no robô mesmo assim.

Imagem 13 – Display Nextion

V. SOFTWARE

O software principal do robô, compilado com a IDE da própria Arduino, é responsável pela junção de todos os dados coletados em tempo real pelos sensores, controle de movimentação, e *path planning*. A alta velocidaade de leitura dos sensores permitiu que esse código fosse desenvolvido com menos limites para ações temporalmente críticas em mente, então aperfeiçoamos a utilização de alguns filtros exponenciais para filtragem de dados, como o *Low Pass*. Não sentimos necessidade de algoritmos mais avançados de filtragem, porém gostaríamos de estudar melhor a respeito do *Filtro de Kalman*

anteriormente à competição.

O robô consegue manter-se constantemente em linha reta sem a necessidade de parar, por conta de um algoritmo de controle Proporcional Integral Derivativo (PID). A partir de uma estimação da odometria espacial, incluindo dados de leituras acima dos 30cm disponíveis por quadrado da arena, ele minimizará os riscos de uma eventual perda de orientação caso ele tenha uma tendência a se deslocar de maneira errônea em um determinado trecho.

O mapeamento é uma extensão do algoritmo FloodFill, com algumas características criadas a partir do zero por nós. Aspectos da arena e do percurso desenvolvido pelo robô são armazenados em arrays estáticos, e todos os cálculos se resumem a simples comparações de valores em iteração, a partir de um sistema de quadrantes e coordenadas interno. Como esse algoritmo no geral se comporta de forma mais previsível, simples e não possui risco de falhas de memória em relação a algoritmos desenvolvidos para busca por grafos (como DFS ou A-Star), acreditamos ter desenvolvido uma ótima estratégia, dado o fato de que agora o robô é capaz de explorar completamente a arena.

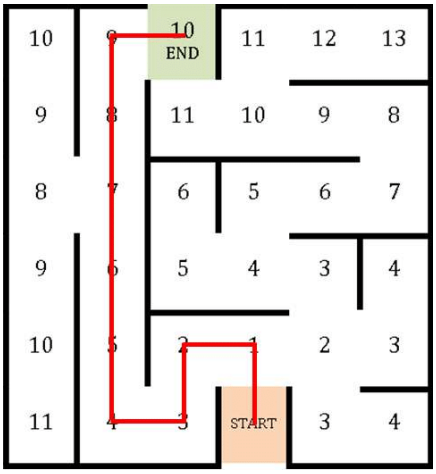
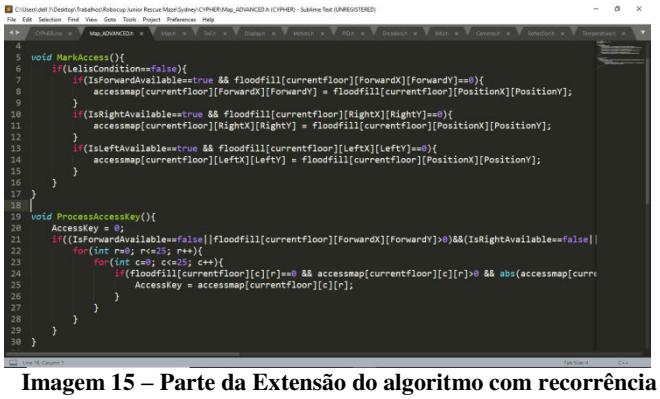


Imagem 14 – Exemplo de FloodFill resolvendo um labirinto com saída



```
void MarkAccess(){
    if(LelisCondition==false){
        if(IsForwardAvailable==true && floodfill[currentfloor][ForwardX][ForwardY]==0){
            accessmap[currentfloor][ForwardX][ForwardY] = floodfill[currentfloor][PositionX][PositionY];
        }
        if(IsRightAvailable==true && floodfill[currentfloor][RightX][RightY]==0){
            accessmap[currentfloor][RightX][RightY] = floodfill[currentfloor][PositionX][PositionY];
        }
        if(IsLeftAvailable==true && floodfill[currentfloor][LeftX][LeftY]==0){
            accessmap[currentfloor][LeftX][LeftY] = floodfill[currentfloor][PositionX][PositionY];
        }
    }
}

void ProcessAccessKey(){
    AccessKey = 0;
    if((IsForwardAvailable==false||floodfill[currentfloor][ForwardX][ForwardY]>0)&&(IsRightAvailable==false||
        for(int r=0; r<=25; r++){
            for(int c=0; c<=25; c++){
                if(floodfill[currentfloor][c][r]==0 && accessmap[currentfloor][c][r]>0 && abs(accessmap[curr
                    AccessKey = accessmap[currentfloor][c][r];
                }
            }
        }
    }
}
```

Imagem 15 – Parte da Extensão do algoritmo com recorrência

O código presente em cada OpenMV (que roda localmente) consiste na busca, a cada frame capturado, de *blobs* de cor preta, selecionados após passarem por filtros e parâmetros de normalização do contraste de cores. Quando um é identificado, ele é utilizado como input para uma pequena rede neural convolucional genérica(treinada para animais, veículos, etc.) para ajudar a classificá-lo como um H, S ou U.

VI. CONCLUSÃO

A partir do conhecimento empírico adquirido desde o início do projeto, tanto em âmbito nacional quanto internacional, é notável o quanto aprendemos. Ao nos colocarmos diante do desafio de produzir um robô mais rapidamente e colocando em prática novas tecnologias mais complexas em todas as áreas do agente, sentimos que somos capazes de operar uma gama bem maior de componentes para um melhor cumprimento das missões propostas. Criamos vínculos inesquecíveis com alguns especialistas, além de outros competidores dedicados de outros países no processo, o que sempre consideramos a parte mais importante de nossa convivência competitiva como um todo.

Mais do que nunca, estamos dispostos a compartilhar os erros e acertos pelos quais passamos ou eventualmente passaremos, em benefício do interesse pela engenharia mecatrônica e de computação.

VII. AGRADECIMENTOS

A equipe novamente gostaria muito de agradecer à Escola de Robótica, a seus professores e envolvidos, e aos nossos pais por nos propiciar esta incrível oportunidade e ferramentas para trabalhar com o Resgate, e aos ex-competidores brasileiros Leonardo Santander da Silva e Ivan Seidel Gomes pelas experiências compartilhadas e por serem de grande fonte de inspiração.

VIII. REFERÊNCIAS

[1] TDP do robô desenvolvido pela equipe em 2018. Disponível em: http://sistemaolimpo.org/midias/uploads/c69104bb463a03f8f2f8c199f249e4cd.pdf

[2] Escola de Robótica e Mecatrônica ABC Marcelo Salles. Disponível em: http://www.roboticaabc.com.br

[3] RoboCup Junior. Disponível em: http://junior.robocup.org

[4] Arduino platform. Disponível em: https://www.arduino.cc

[5] RoboCore BlackBoard Mega. Disponível em: https://www.robocore.net/loja/arduino/blackboard-mega

[6] OpenMV platform. Disponível em: https://openmv.io

[7] MicroPython. Disponível em: https://micropython.org

[8] Impressão 3D. Disponível em: https://pt.wikipedia.org/wiki/Impressão_3D

[9] Impressora 3D UPBOX+. Disponível em: http://www.up3d.com.br/produtos/impressoras-3d/tiertime/impressora-3d-modelo-up-box

[10] AutoCAD. Disponível em: https://www.autodesk.com.br/products/autocad/overview

[11] Sublime Text. Disponível em: https://www.sublimetext.com

[12] Driver Ponte H. Disponível em: https://pt.wikipedia.org/wiki/Ponte_H

[13] Motores Pololu de 25Dmm. Disponível em: https://www.pololu.com/category/115/25d-mm-metal-gearmotors

[14] Servo Motor. Disponível em: https://pt.wikipedia.org/wiki/Servomotor

[15] Servo SM-S2309S. Disponível em: https://www.arduino.cc/documents/datasheets/servoMotor.PDF

[16] US-015 vs HC-SR04. Disponível em: https://www.youtube.com/watch?v=aLkkAsrSibo&t=219s

[17] Display de Cristal Líquido (LCD) . Disponível em:

https://pt.wikipedia.org/wiki/LCD
[18] IMU. Disponível em: https://en.wikipedia.org/wiki/Inertial_measurement_unit
[19] Ultrassônico. Disponível em: https://pt.wikipedia.org/wiki/Ultrassom
[20] Temperatura Infravermelha. Disponível em: http://guias.oxigenio.com/como-funciona-um-termometro-infravermelho
[21] Luz Infravermelha. Disponível em: https://pt.wikipedia.org/wiki/Radia%C3%A7%C3%A3o_infravermelha
[22] Algoritmo Floodfill. Disponível em: https://en.wikipedia.org/wiki/Flood_fill
[23] Material para arena de testes. Disponível em: http://www.leomadeiras.com.br