SENAI
Serviço Nacional de Aprendizagem Industrial
PELO FUTURO DO TRABALHO

SENAI CFP Alvimar Carneiro Rezende
TÉCNICO EM ELETRÔNICA

Relatório Técnico - AlviMAR

Fellipe Parreiras
Geovanna Lopes
Isabela Pontes
Pedro Diniz
Yasmin Tomaz

Contagem
Abril de 2023

Relatório Técnico - AlviMAR

Fellipe Parreiras
Geovanna Lopes
Isabela Pontes
Pedro Diniz
Yasmin Tomaz

Relatório do projeto do Trabalho de Conclusão de Curso (TCC) do curso Técnico em Eletrônica do CFP SENAI Alvimar Carneiro Rezende.

Contagem
Abril de 2023

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

O trabalho de conclusão de curso é uma etapa importante na vida acadêmica de todo estudante que se dedica a um curso técnico ou superior. É um momento em que o aluno deve aplicar todos os conhecimentos adquiridos durante o período de estudo e apresentar um projeto inovador e relevante para a área em que se especializou.

No caso do SENAI, os estudantes têm a opção de desenvolver um projeto próprio ou escolher uma demanda da indústria no site Saga SENAI de Inovação e oferecer uma solução. Esse modelo de trabalho estimula a criatividade e a capacidade de resolver problemas dos estudantes, além de proporcionar a eles uma oportunidade de se aproximar do mercado de trabalho.

Em nosso grupo, decidimos desenvolver um projeto próprio durante o curso. Nossa ideia foi criar um carrinho utilizando os componentes eletrônicos disponíveis nas faculdades do SENAI. Buscamos desenvolver um projeto simples, barato e viável, mas que ao mesmo tempo fosse interessante, atraente e inovador.

Para isso, optamos por utilizar um microcontrolador (Arduino UNO) para controlar o carrinho. O microcontrolador é uma placa programável que permite aos desenvolvedores criarem sistemas eletrônicos inteligentes e interativos. Com ele, conseguimos programar o carrinho para seguir uma rotina de ações com a finalidade principal de apresentar um local.

Nosso projeto contou com diversas etapas, desde a concepção da ideia até a realização da montagem. Foram necessárias muitas horas de estudo e trabalho em equipe para chegar ao resultado final. Durante todo o processo, enfrentamos desafios e imprevistos, o que nos exigiu muita dedicação e persistência.

No entanto, ao final do projeto, sentimos uma grande satisfação por termos conseguido colocar em prática tudo o que aprendemos durante o curso e por termos desenvolvido um projeto inovador e relevante para a área em que atuamos.

Por meio deste relatório técnico, buscamos apresentar de forma clara e objetiva todo o processo de desenvolvimento do nosso projeto, desde a concepção da ideia até a realização da montagem do carrinho. Esperamos que este trabalho possa servir de inspiração para outros estudantes que buscam desenvolver projetos inovadores e relevantes para a sua área de atuação.

2 Desenvolvimento do projeto

Para desenvolver o AlviMAR, a equipe enfrentou diversos desafios, principalmente relacionados ao desenvolvimento do código e à lógica de programação. Foram necessários vários testes para identificar e corrigir os bugs que surgiam.

Em relação à construção do corpo externo do carrinho, a equipe optou por utilizar uma impressora 3D e criou o arquivo .stl no software CAD SketchUp. Essa escolha permitiu que o projeto fosse desenvolvido de maneira mais precisa e rápida, além de possibilitar a criação de um design mais interessante e personalizado.

A parte eletrônica do AlviMAR foi feita com um Arduino UNO, um servo motor, um sensor ultrassônico, 4 motores DC brushless para as rodas e uma ponte H para controlar as rodas. Para a estrutura interna, a equipe também utilizou a impressora 3D, inicialmente, iriamos fazer uma parte externa com a impressão em 3D, porém devido à problemas com o tempo disponível para o desenvolvimento do projeto essa ideia foi descartada. Caso alguém decida contribuir com o projeto futuramente, esse é um ponto importante que deve ser trabalhado.

O display gráfico utilizado no AlviMAR possui funcionalidade estética, mas desempenha um papel importante na apresentação do carrinho. Posicionado como se fosse a cabeça do Alvi, ele exibe diferentes telas dependendo do que está sendo falado. Quando não está falando, exibe um sprite (imagens em sequencia que dão a impressão de movimento, como se fosse um vídeo curto) de olhos fofinhos piscando, e quando está esperando os visitantes, exibe o tempo restante que o Alvi vai esperar até seguir com a apresentação. Essas são algumas das funções do display.

Em suma, o AlviMAR é um projeto inovador que utiliza tecnologias avançadas de eletrônica e impressão 3D para criar um carrinho simpático e fofinho que serve para apresentar um local. A equipe enfrentou diversos desafios durante o desenvolvimento, mas conseguiu superá-los e criar um projeto simples, barato e viável, porém interessante, atraente e inovador.

2.1 Desenvolvimento do código

Nesta seção, vamos descrever brevemente como funciona a ideia básica do código. O arquivo está disponível para visualização e download no nosso GitHub.

Para ser um guia tour, é necessário saber por onde transitar e o que dizer aos visitantes. Simplificando este problema, robô precisa ter em sua memória as distâncias, as esquinas, as falas e as expressões faciais que deve percorrer, virar, reproduzir e exibir, respectivamente. Combinando estas ações, temos um robozinho que pode fazer apresentações. Para programar estas funções, utilizamos um *array* de *strings* que contém as ações que o robô deve realizar. Embora pudéssemos utilizar um array de números para representar estas ações, optamos por strings para tornar o código mais legível e intuitivo para outras pessoas que desejem utilizá-lo (vale ressaltar que é vedado o uso da classe *String* na programação em C++ com o Arduino, isso se deve ao fato desta classe utilizar a memória de forma ineficiente, e pode acarretar em certos problemas. Porém, nosso código não tem como objetivo a eficiência de memória, é apenas um protótipo e uma demonstração de viabilidade). O array de ações contém cinco tipos de ações: *"move"*, *"rotate"*, *"wait"*, *"speak"* e *"display"*. Uma função supervisora *"EfetuarRotinaDeTour"* reconhece o tipo de ação por

meio de um *loop for* e busca pelo valor correspondente.

2.1.1 Movimentação do robô

Também temos um array de duas dimensões com as distâncias e ângulos a serem percorridos. Os valores de distância possuem como segunda dimensão a velocidade que o robô deve ter ao percorrer a distância associada. Isso permite que o robô ande mais lentamente em um corredor, por exemplo, dando aos visitantes tempo para observar os pontos de interesse enquanto o robô os apresenta. Os valores de ângulo têm como segunda coordenada o número 0 para que possamos identificá-los no código (já que uma distância associada à uma velocidade "0"não faz sentido).

```
// Distâncias e ângulos de rotação
// (Ângulos tem segundo valor "0")
const float valoresSegundoAndar [][2] = {
    {9.27, velocidadeMaxima},
    {90, 0},
    {43.45, velocidadeMaxima},
    {180, 0},
    {9.4, velocidadeMaxima},
    {8.68, velocidadeMaxima},
    {26.20, velocidadeMaxima},
    {8.84, velocidadeMaxima},
    {6.58, velocidadeMaxima}
};
```

Figura 1: Array bidimensional.

Neste array por exemplo, os elementos de index 1 e 3 são instruções para rotação de, respectivamente, 90 e 180 graus. Quando a função supervisora é executada, ela declara uma variável para determinar o primeiro index de um valor de movimento e de rotação.

Quando a função identifica a ação *"move"* no código, ela busca pelo primeiro índice do array bidimensional de valores que ainda não foi lido e que possui um valor de distância e velocidade. A função ignora os valores que têm 0 como valor da segunda coordenada. Da mesma forma, quando a função encontra a ação "rotate", busca pelo primeiro índice não lido contendo uma coordenada com o segundo valor 0, caracterizando assim um valor de ângulo.

```
void EfetuarRotinaDeTour(String arrayAcoes[], float valoresCaminho[][2], int temposDeEspera[], int indexComeco = 0) {
    int indexDistancia, indexAngulo, indexEspera;
    // Definir primeiro index de valor de distância
    while (valoresCaminho[indexDistancia][1] < 0) {
        indexDistancia++;
        if (indexDistancia == sizeof(valoresCaminho)) {
            break; // Não existem movimentos
        }
    }
    // Definir primeiro index de valor de rotação
    while (valoresCaminho[indexAngulo][1] != 0) {
        indexAngulo++;
        if (indexAngulo == sizeof(valoresCaminho)) {
            break; // Não existem rotações
        }
    }

    for (int i = indexComeco; i < sizeof(arrayAcoes); i++) { // Efetuar todas as ações em <arrayAcoes>
        if (arrayAcoes[i] == "move") {
            MoverRobo(valoresCaminho[indexDistancia][0], valoresCaminho[indexDistancia][1]);
                indexDistancia++;
                while (valoresCaminho[indexDistancia][1] == 0) { // Ignorar ângulos, definir <indexDistancia> para próxima valor de distância
                    indexDistancia++;
                }
        } else if (arrayAcoes[i] == "rotate") {
            RotacionarRobo(valoresCaminho[indexAngulo][1]);
                indexAngulo++;
                while (valoresCaminho[indexAngulo][1] != 0) { // Ignorar distâncias, definir <indexAngulo> para próxima valor de rotação
                    indexAngulo++;
                    if (indexAngulo == sizeof(valoresCaminho)) {
                        break;
                    }
                }
        } else if (arrayAcoes[i] == "wait") {
            ExibirFace(8, temposDeEspera[indexEspera]);
            indexEspera++;
        } else if (arrayAcoes[i] == "speak") {
            // bluetooth.print("lx");
        }
    }
}
```

Figura 2: Função supervisora.

2.1.2 A função MoverRobo()

A função *"MoverRobo"* é responsável por controlar as rodas do robô por meio da ponte-h. Ela recebe dois valores: um *float* representando a distância a ser percorrida e outro *float* representando a velocidade que o robô deve ter ao percorrer essa distância. Por padrão, a velocidade máxima é utilizada. Se a distância fornecida for negativa, o robô se move para trás; se for positiva, o robô se move para frente.

A ponte-h é acionada pelos pinos EN1, EN2, IN1, IN2, IN3 e IN4. Quando os pinos EN1 ou EN2 são acionados com *pulse width modulation (PWM)*, é possível controlar a velocidade das rodas. Se os pinos EN são acionados com um nível lógico alto (*duty cycle* de 100%), as rodas giram na velocidade máxima. Se o *duty cycle* for 50%, as rodas devem girar na metade da velocidade, conforme uma hipótese que ainda precisa ser testada empiricamente.[1]

[1]Não tivemos tempo de testar essa afirmação empiricamente, porém imaginamos que se esta hipótese estiver incorreta, seria necessária uma correção linear simples ($||\vec{v}(D_c)|| = D_c + kD_c$).

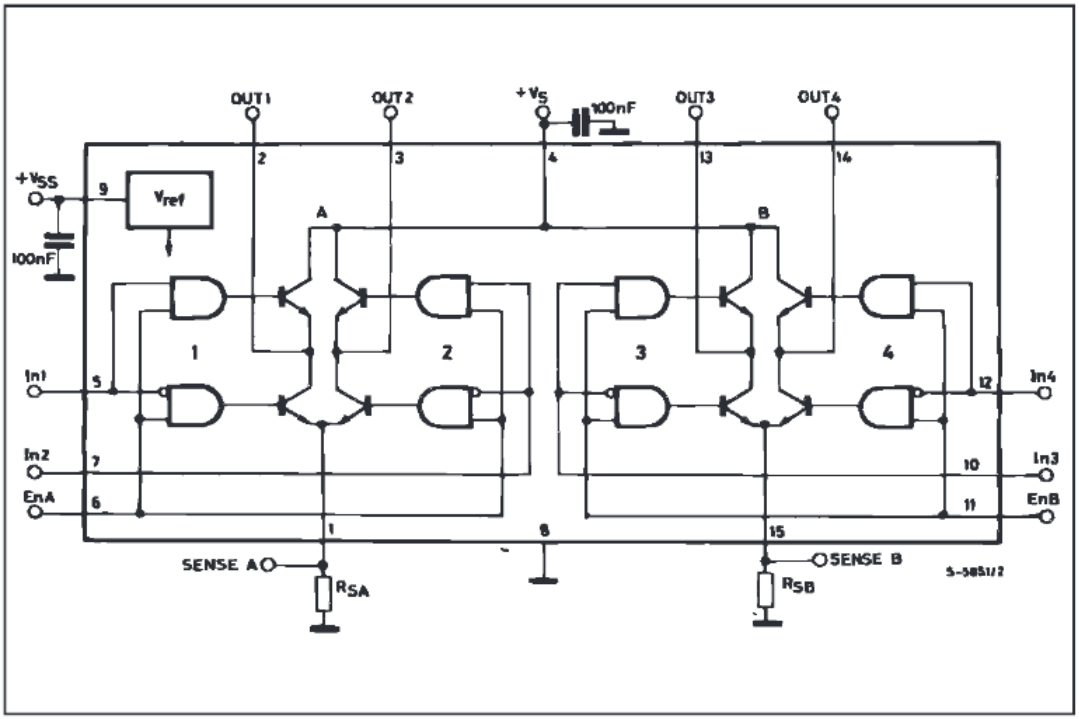

Figura 3: Diagrama em blocos do driver L298N (ponte-h).

Pino	EN1	EN2	IN1	IN2	IN3	IN4
distância ≥ 0	$\sim$	$\sim$	1	0	1	0
distância < 0	$\sim$	$\sim$	0	1	0	1

Tabela 1: Acionamento da ponte-h
O símbolo $\sim$ denota pinos acionados com PWM.

Para percorrer a distância especificada, é necessário calcular a distância percorrida a cada iteração do *loop*. Isso é feito acrescentando a velocidade passada como argumento, dividida por mil, a uma variável *float* declarada na função. A cada iteração, a função *delay* é utilizada para aguardar um milissegundo. Essa divisão permite simular uma velocidade instantânea e verificar se há algum sinal enviado pelo bluetooth a cada iteração do *loop*, bem como verificar se há algo obstruindo a passagem do robô, evitando uma colisão.

Outra forma de controlar o movimento do robô seria calcular o tempo que ele levaria para percorrer a distância na velocidade declarada, acionar as rodas e aguardar o tempo calculado utilizando a função *delay*. Entretanto, esse método faria com que o robô não respondesse a nenhuma ação nem freasse com a presença de um obstáculo, até atingir o tempo necessário para percorrer a distância requisitada.

```
void MoverRobo(float distancia, float velocidade) {
    float distanciaPercorrida = 0;

    analogWrite(enable_1, map(velocidade, 0, velocidadeMaxima, 0, 255));
    analogWrite(enable_2, map(velocidade, 0, velocidadeMaxima, 0, 255));

    if (distancia < 0) {
        digitalWrite(pinoIn_1, LOW);
        digitalWrite(pinoIn_2, HIGH);
        digitalWrite(pinoIn_3, LOW);
        digitalWrite(pinoIn_4, HIGH);
    } else {
        delay(3000);
        digitalWrite(pinoIn_1, HIGH);
        digitalWrite(pinoIn_2, LOW);
        digitalWrite(pinoIn_3, HIGH);
        digitalWrite(pinoIn_4, LOW);
    }

    int caminhoDesobstruido = 1; // Booleana para parar de calcular a distância percorrida caso o caminho esteja obstruído
    while (toInt(round(distanciaPercorrida*100)) < toInt(round(distancia*100))) {
        distanciaPercorrida += velocidade/1000 * caminhoDesobstruido;
        delay(1);

        if (toInt(DistanciaUltrassonico()) <= toInt(disparoSensorUltrassonico)) {
            digitalWrite(enable_1, LOW);
            digitalWrite(enable_2, LOW);
            caminhoDesobstruido = 0;
        } else {
            analogWrite(enable_1, map(velocidade, 0, velocidadeMaxima, 0, 255));
            analogWrite(enable_2, map(velocidade, 0, velocidadeMaxima, 0, 255));
            caminhoDesobstruido = 1;
        }

        if (bluetooth.available() > 0) {
            digitalWrite(enable_1, LOW);
            digitalWrite(enable_2, LOW);
            return;
        }
    }
}
```

Figura 4: Função *MoverRobo()*.

2.1.3 A função RotacionarRobo()

O funcionamento desta função é análogo ao da função *MoverRobo*. A diferença é que o robô não confere o sensor ultrassônico, e o acionamento dos pinos é feito da seguinte forma:

Pino	EN1	EN2	IN1	IN2	IN3	IN4
ângulo ≥ 0	$\sim$	$\sim$	1	0	0	1
ângulo < 0	$\sim$	$\sim$	0	1	1	0

Tabela 2: Acionamento da ponte-h

```
void RotacionarRobo(int angulo) {
    float anguloAtual = 0;

    if (angulo < 0) {
        digitalWrite(pinoIn_1, HIGH);
        digitalWrite(pinoIn_2, LOW);
        digitalWrite(pinoIn_3, LOW);
        digitalWrite(pinoIn_4, HIGH);
    } else {
        digitalWrite(pinoIn_1, LOW);
        digitalWrite(pinoIn_2, HIGH);
        digitalWrite(pinoIn_3, HIGH);
        digitalWrite(pinoIn_4, LOW);
    }
    analogWrite(enable_1, HIGH);
    analogWrite(enable_2, HIGH);

    while (toInt(round(anguloAtual*100)) <= toInt(round(abs(angulo)*100))) {
        anguloAtual += velocidadeAngular/1000;
        delay(1);

        if (bluetooth.available() > 0) {
            digitalWrite(enable_1, LOW);
            digitalWrite(enable_2, LOW);
            return;
        }
    }

    digitalWrite(enable_1, LOW);
    digitalWrite(enable_2, LOW);
}
```

Figura 5: Função *RotacionarRobo()*.

2.1.4 Outros comandos

Para o comando *"wait"*, existe um array de inteiros com os valores que o robô deve aguardar. Quando a função *"EfetuarRotinaDeTour"* recebe essa ação, ela busca pelo próximo índice a ser lido. Da mesma forma, para a ação *"speak"*, existe um array de strings que são códigos para serem enviados para um telefone via Bluetooth. O telefone, conectado ao Arduino via Bluetooth, recebe esses dados e os decodifica, identificando o arquivo de áudio que deve ser reproduzido. A ação *"display"* é similar à ação *"wait"*, mas serve para escolher a face que o robô deve exibir no display gráfico.

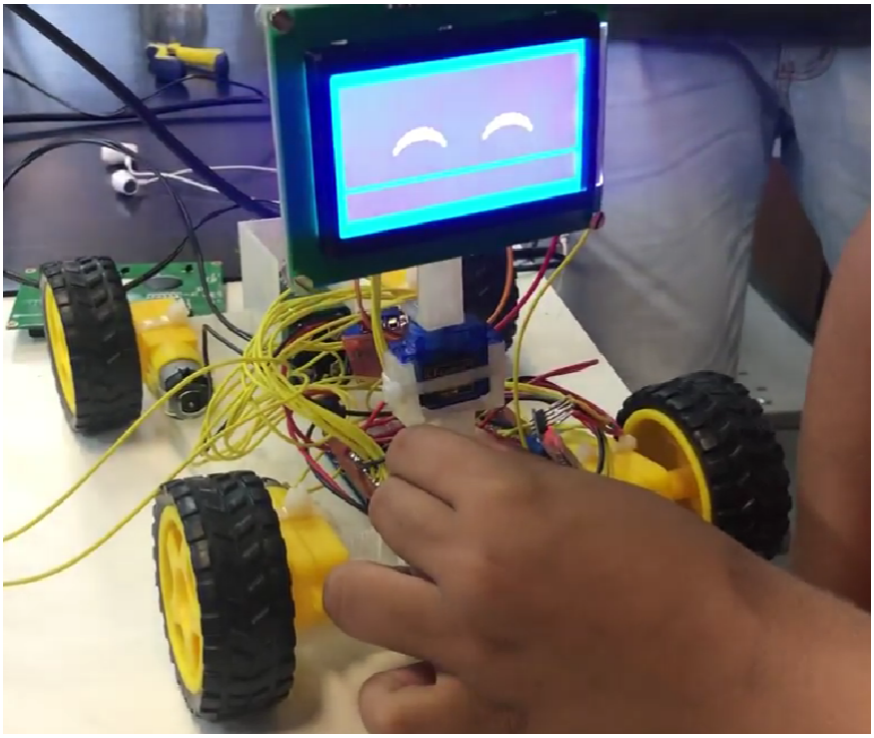

Figura 6: Protótipo montado.
Nesta imagem podemos ver a função display exibindo a face do Alvi.

2.1.5 Comunicação Arduino-App

O aplicativo que desenvolvemos funciona como um controle remoto, e serve para testar as funções e calibrar as distâncias à serem percorridas. A comunicação é simples; o aplicativo *"AlviMAR"* envia um código, como por exemplo *"<h180> "* (comando para girar o servo motor do pescoço para 180 graus), e o módulo bluetooth o recebe e envia a informação por meio de comunicação serial para o Arduino. O Arduino recebe os caracteres um-por-um, portanto é necessária uma função caso queiramos receber a mensagem completa. Com a mensagem completa recebida, o primeiro caractere define a ação que o robô deve efetuar. Os códigos são *"h"*, *"f"*, *"m"*, *"c"*, *"r"*, *"s"*, *"a"* [2] e designam, respectivamente, a ação de girar o servo motor do pescoço, exibir uma imagem no display, mover por uma distância definida, mover indefinidamente, rotacionar o robô, definir a velocidade do robô e requisitar o array de ações da rotina.

Um detalhe importante. No código do Arduino, incluímos a biblioteca *"SoftwareSerial"*, que serve para receber informações seriais pelos pinos digitais. O Arduino possuí dois pinos para comunicação serial, os pinos 0 e 1, porém a utilização desses pinos interfere na comunicação serial USB. Caso o módulo bluetooth fosse conectado aos pinos 0 e 1, não seria possível fazer qualquer tipo de comunicação Arduino-Computador com o módulo bluetooth ligado, isso iria nos atrapalhar muito no desenvolvimento.

[2] A escolha dessas abreviações se deve à descrição das ações em inglês. Em ordem: "head", "face", "move", "(move) continuously", "rotate", "speed", "actions".

2.2 Desafios no desenvolvimento

Inicialmente, tínhamos a ideia de imprimir tanto a estrutura interna quanto a externa do robô, mas decidimos priorizar a estrutura interna para iniciar os testes o quanto antes possível. O primeiro arquivo .stl que produzimos era grande demais para a impressora, o que foi facilmente resolvido fracionando a impressão em duas partes. No entanto, um dos arquivos teve uma falha que fez com que a impressora tentasse imprimir filamentos sem suporte, resultando em uma perda de impressão.

Na segunda tentativa, aprendemos com os erros da primeira e reduzimos o tamanho da impressão para o tamanho da impressora. Imprimimos a estrutura principal do robô e, em outra impressão, fizemos o suporte do servo motor do pescoço e o suporte do LCD. Contudo, diversos erros críticos foram cometidos, como a ponte-H que bloqueava o conector de alimentação e o conector USB do Arduino, os furos nos lugares errados nos suportes das rodas e o sensor ultrassônico que não encaixava no buraco designado. Felizmente, conseguimos retificar esses erros utilizando uma micro retífica para desgastar as partes necessárias e prosseguir com o projeto.

Já na parte do projeto do sistema eletrônico, tivemos problemas como fios de conexão muito grandes, o terminal de conexão para as baterias de alimentação da ponte-H localizado do lado contrário às baterias e a alimentação do Arduino que não fornecia corrente suficiente, entre outros.

Esses erros foram causados pela falta de comunicação entre os integrantes encarregados do projeto da estrutura impressa em 3D e os encarregados da parte eletrônica. Aprendemos, assim, uma importante lição sobre a vital importância da comunicação nos mínimos detalhes entre os integrantes do grupo para o correto andamento de um projeto.

3 Considerações finais

O projeto de construção de um robô autônomo foi uma experiência extremamente enriquecedora, que nos permitiu aplicar muitos dos conceitos teóricos que aprendemos em sala de aula. Através desse projeto, pudemos ver na prática como é desafiador coordenar equipes com habilidades e conhecimentos distintos, além de lidar com imprevistos ao longo do processo de construção.

A escolha da plataforma Arduino como base para o robô nos permitiu uma grande flexibilidade na programação e controle de diversos componentes eletrônicos. Além disso, a impressão 3D foi uma técnica crucial para a construção da estrutura física do robô, permitindo a criação de peças personalizadas de forma rápida e eficiente.

Durante o processo, tivemos que superar diversos desafios, como a falta de comunicação entre as equipes responsáveis pelo projeto da estrutura impressa em 3D e pelo projeto da parte eletrônica. Como resultado, houve a necessidade de retrabalho e correções ao longo do processo. Aprendemos que a comunicação é fundamental em um projeto desse tipo, mesmo nos mínimos detalhes.

Outro desafio foi a medição dos corredores para que o robô pudesse andar as distâncias certas. Foi necessário uma atenção especial para garantir que as medidas estivessem corretas, e para ajustar a velocidade do robô para que se movesse de forma precisa e controlada.

Figura 7: Medição dos corredores com laser.

Figura 8: Resultado da medição.

Apesar dos desafios, estamos muito satisfeitos com o resultado final do projeto, que permitiu a aplicação prática de muitos conceitos teóricos aprendidos ao longo do curso. Este projeto nos ajudou a desenvolver habilidades de trabalho em equipe, resolução de problemas, planejamento e execução, e aprimorou nossa compreensão sobre robótica e sistemas embarcados.

É importante ressaltar que este projeto está aberto para outros estudantes que desejarem aprimorá-lo. O código e o aplicativo estão disponíveis no GitHub. Esperamos que esta iniciativa possa incentivar outros a explorarem as diversas disciplinas necessárias para o desenvolvimento de um projeto como este. Cálculo, física, programação, lógica, eletrônica... vários campos de conhecimentos foram necessários para idealizar e concretizar o protótipo. Um projeto final seria ainda mais desafiador. Construir um robô autônomo foi uma jornada emocionante e enriquecedora, e estamos animados para ver como este projeto pode evoluir com a contribuição de outros estudantes.

4 Agradecimentos

Gostaríamos de agradecer ao nosso instrutor, o Professor Charles Neves, pelo auxílio que ofereceu à todos os alunos da nossa turma, e ao instrutor Lucas Taylon por nos acompanhar em várias etapas do processo de desenvolvimento, desde a etapa criativa, até a apresentação final. Agradecemos também à toda a equipe da unidade CFP SENAI Alvimar Carneiro Rezende.

"Agradeço aos meus amigos e amigas Isabela Pontes, Isabela Miranda, Pedro, Yasmin, Samuel, Geovanna e Luiz, que sempre proporcionaram um ambiente descontraído nas aulas. Foi ótimo conhecê-los, e desejo muito sucesso na vida de vocês!"

- Fellipe Parreiras

5 Referências

Sending and Receiving Data with HC-05 - MIT App Inventor. Robo India, 2023. Disponível em: <https://roboindia.com/tutorials/sending-receiving-with-hc05-mit-app-inventor/>. Acesso em: 15/04/2023.

I want to see only Bluetooth name not address. MIT App Inventor, 2021. Disponível em: <https://community.appinventor.mit.edu/t/i-want-to-see-only-bluetooth-name-not-address/43879/2>. Acesso em: 15/04/2023.

Adding Bluetooth to Your Arduino Project with an HC-05 or HC-06 Bluetooth Module. YouTube, 2021. Disponível em: <https://www.youtube.com/watch?v=NXlyo0goBrU>. Acesso em: 15/04/2023.

U8GLIB and bitmap creation/display. Arduino, 2013. Disponível em: <https://forum.arduino.cc/t/u8glib-and-bitmap-creation-display/148125>. Acesso em: 15/04/2023.

Serial Input Basics - updated. Arduino, 2016. Disponível em: <https://forum.arduino.cc/t/serial-input-basics-updated/382007>. Acesso em: 15/04/2023.

How to create Sidebar in MIT App Inventor 2 | App Sidebar Design. YouTube, 2022. Disponível em: <https://www.youtube.com/watch?v=W7HTuJa4fZ0>. Acesso em: 15/04/2023.