



Sabrina Daiana Ferrari Krohn
Thiago Rafael Wendland

**IMPLEMENTAÇÃO DE CIRCUITO ELETRÔNICO E PROGRAMAÇÃO PARA
ROBÔ DE TELEPRESENÇA**

Horizontina - RS

2021

Sabrina Daiana Ferrari Krohn

Thiago Rafael Wendland

**IMPLEMENTAÇÃO DE CIRCUITO ELETRÔNICO E PROGRAMAÇÃO PARA
ROBÔ DE TELEPRESENÇA**

Trabalho Final de Curso apresentado como requisito parcial para a obtenção do título de bacharel em engenharia de Controle e Automação na Faculdade Horizontina, sob a orientação do Prof. Me. Paulo Marcos Flores.

Horizontina - RS

2021

FAHOR - FACULDADE HORIZONTINA
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

A Comissão Examinadora, abaixo assinada, aprova o trabalho final de curso

**“IMPLEMENTAÇÃO DE CIRCUITO ELETRÔNICO E PROGRAMAÇÃO PARA
ROBÔ DE TELEPRESENÇA”**

Elaborada por:
Sabrina Daiana Ferrari Krohn
Thiago Rafael Wendland

Como requisito parcial para a obtenção do grau de Bacharel em
Engenharia de Controle e Automação

Aprovado em: 02/07/2021
Pela Comissão Examinadora

Me. Paulo Flores
Presidente da Comissão Examinadora - Orientador

Dr. Geovane Webler
FAHOR – Faculdade Horizontina

Me. Rodrigo Bastos
FAHOR – Faculdade Horizontina

Horizontina - RS
2021

Às nossas famílias por todo auxílio prestado para que alcançássemos nosso objetivo. Aos nossos colegas e amigos que estiveram conosco durante todo esse período e a todos que direta ou indiretamente fizeram parte da nossa formação, o nosso muito obrigado.

Deixamos um agradecimento aos professores, pelas correções e ensinamentos que nos permitiram apresentar um melhor desempenho no nosso processo de formação profissional ao longo do curso, em especial ao nosso orientador pelo incentivo e dedicação do seu tempo ao nosso projeto.

“A única maneira de não cometer erros é não fazendo nada. Este, no entanto, é certamente um dos maiores erros que se poderia cometer em toda uma existência”.

(Confúcio)

RESUMO

A inserção de projetos tecnológicos nas mais vastas áreas está em constante crescimento, visto que há cada vez mais aplicações para dispositivos que antes tinham poucas finalidades. Devido a esta constatação, observou-se a seguinte situação-problema, o aumento dos riscos de transmissão de vírus e bactérias entre pessoas durante o pré-atendimento em uma clínica hospitalar. Para solucionar tal desgaste, é realizado o desenvolvimento de um protótipo de um robô de telepresença para auxiliar o paciente a encontrar o consultório desejado e, se necessário, obter informações através de uma videoconferência onde um profissional qualificado estará auxiliando por meio do robô de telepresença, isto através do desenvolvimento e implementação da programação e do circuito elétrico, obtendo assim um projeto eficaz. Espera-se que este estudo possa ser continuado e desenvolvido, contrapondo todas as adversidades, possibilitando também a redução de custos e assim ser aplicado nas empresas.

Palavras-chave: Microcontrolador, programação e robô.

LISTA DE FIGURAS

Figura 1 - Sensor ultrassônico.....	20
Figura 2 – Modelo de motor CC.....	22
Figura 3 – Ciclo percepção-planejamento-atuação.....	23
Figura 4 – Modelo de processador.....	25
Figura 5 – Modelo de microcontrolador ATmega328.....	25
Figura 6 – Comparativo entre as versões do Arduino.....	26
Figura 7 – ESP32 e suas conexões.....	27
Figura 8 – Buzzer.....	29
Figura 9 – Ponte H.....	30
Figura 10 – Relação entre C e C++.....	32
Figura 11 – Método Munari.....	35
Figura 12 – Protoboard.....	37
Figura 13 – Sensores ultrassônicos.....	38
Figura 14 – Buzzer de 12v utilizado no protótipo.....	38
Figura 15 – Driver Ponte H L298N.....	39
Figura 16 – Placa ESP32.....	39
Figura 17 – Botões de comando.....	40
Figura 18 – Motores de corrente contínua.....	40
Figura 19 – Bateria 12v.....	41
Figura 20 – Pares de rodas.....	41
Figura 21 – Esquema elétrico.....	43
Figura 22 – Circuito elétrico.....	43
Figura 23 – Tampa do robô.....	44
Figura 24 – Suporte do tablet e botões.....	45
Figura 25 – Circuito do ESP32 e Ponte H.....	45
Figura 26 – Circuito completo.....	46
Figura 27 – Robô simulando posição inicial.....	47
Figura 28 – Obstáculo adicionado para testar sensores.....	48
Figura 29 – Inclusão das bibliotecas na programação.....	48
Figura 30 – Definição das portas e variáveis dos motores.....	49
Figura 31 – Definição das portas dos botões.....	49
Figura 32 – Variável de estado dos botões.....	49

Figura 33 – Variável de distância dos sensores.....	49
Figura 34 – Definição das portas dos sensores.....	50
Figura 35 – Definindo o pino do Buzzer.....	50
Figura 36 – Programação que inicia os sensores.....	50
Figura 37 – Define os pinos como entrada ou saída.....	51
Figura 38 – Inicia a função de repetição.....	51
Figura 39 – Variável para ler o estado dos botões.....	51
Figura 40 – Função sensores sendo chamada.....	52
Figura 41 – Inicia a função lógica para o estado do botão.....	52
Figura 42 – Inicia a função lógica com as informações dos sensores.....	52
Figura 43 - Silenciar o Buzzer.....	52
Figura 44 – Chama a função do botão vermelho.....	53
Figura 45 – Emissão do sinal sonoro e parada dos motores.....	53
Figura 46 – Programação do botão B.....	54
Figura 47 – Comando que cria a função sensores.....	54
Figura 48 – Definição dos pinos Trigger dos sensores.....	54
Figura 49 – Comando que realiza uma pausa.....	55
Figura 50 – Definição dos pinos do sensor como alto.....	55
Figura 51 – Pinos dos sensores definidos como baixo.....	55
Figura 52 – Comando que obtém a distância.....	55
Figura 53 – Comando que cria a função do botão vermelho.....	56
Figura 54 – Comando para parar os motores durante 2 segundos.....	56
Figura 55 – Código para girar os motores para frente.....	56
Figura 56 – Comando para parar motores e ativar o sinal sonoro.....	57
Figura 57 – Programação para retornar ao posto inicial.....	57
Figura 58 – Robô retorna a posição inicial.....	58
Figura 59 – Função com a rota do botão B.....	59

LISTA DE QUADROS

Quadro 1 – Características do sensor ultrassônico.....	21
Quadro 2 - Comparação do ESP32 com Arduino UNO.....	28
Quadro 3 - Recursos utilizados no projeto.....	35
Quadro 4 – Quantidade e valores de materiais utilizados.....	42

LISTA DE TABELAS

Tabela 1 – Características da Ponte H L298N.....	30
--	----

LISTA DE ABREVIATURAS E/OU SIGLAS

MMU – Unidade de Gerenciamento de Memória

UC – Unidade de Controle

ULA – Unidade Lógica e Aritmética

IOT – Internet das Coisas

CC – Corrente Contínua

USB – Universal Serial Bus

GND – Graduated Neutral Density Filter

ANSI – American National Standards Institute

SDK – Software Development Kit

IDE – Ambiente de Desenvolvimento Integrado

CPU – Unidade Central de Processamento

SUMÁRIO

1. INTRODUÇÃO	14
1.1. TEMA	14
1.2. DELIMITAÇÃO DO TEMA	14
1.3. PROBLEMA DE PESQUISA	15
1.4. HIPÓTESES	15
1.5. JUSTIFICATIVA	16
1.6. OBJETIVOS	17
1.6.1. Objetivo Geral	17
1.6.2. Objetivos Específicos	17
2. REVISÃO DA LITERATURA	18
2.1. ROBÔ	18
2.1.1. Corporalidade	18
2.1.2. Sensoriamento	19
2.1.2.1. Classificação do sensor	19
2.1.2.2. Sensor ultrassônico hc-sr04.....	20
2.1.3. Efetuadores e atuadores	21
2.1.3.1. Motores elétricos.....	21
2.1.3.2. Motor de corrente contínua.....	22
2.1.4. controladores	23
2.1.5. Arquiteturas de controle	23
2.1.6. Microprocessadores e microcontroladores	24
2.1.6.1. Arduino.....	26
2.1.6.2. ESP32.....	27
2.1.6.3. <i>Buzzer</i>	28
2.1.6.4. Ponte h.....	29
2.1.7. Programação	30
2.1.7.1. Linguagem C.....	31
2.1.7.2. Linguagem C++	32
2.1.7.3. Programação esp32.....	33
3. METODOLOGIA	34
3.1. MÉTODOS E TÉCNICAS UTILIZADOS	35
3.2. MATERIAIS E EQUIPAMENTOS	35
4. APRESENTAÇÃO E ANÁLISE DOS RESULTADOS	37
4.1. PRINCIPAIS COMPONENTES UTILIZADOS	37
4.2. TESTE DE BANCADA.....	42
4.2.1. Suporte para o <i>tablet</i>	44
4.3. MONTAGEM DO PROTÓTIPO.....	45
4.3.1. Testes do funcionamento do robô	46
4.4. PROGRAMAÇÃO.....	48
CONCLUSÃO	60
REFERÊNCIAS	61
APÊNDICE A – PROGRAMA UTILIZADO	64

1. INTRODUÇÃO

Com a atual necessidade e alta demanda de serviços que permitam a realização de atividades de risco à distância, diversos dispositivos estão sendo estudados nos últimos anos e possuem como objetivo eliminar os riscos causados pelo contato direto entre pessoas. Em situações onde isso é dispensável, como durante o pré-atendimento em uma clínica hospitalar, onde é realizada a coleta de dados do paciente, encaminhamento para o consultório e que não há necessidade de contato físico.

Tendo em vista eliminar os riscos mencionados, a tecnologia pode ser utilizada com intuito de nos favorecer e assim fazer o possível para não haver propagação de contaminantes, isso através da elaboração de um robô da telepresença. O robô possui como principal função a locomoção de um *tablet*, onde o paciente poderá selecionar o local que deseja ir e caso necessite de ajuda poderá contatar um profissional qualificado que o atenderá via vídeo-chamada.

Neste contexto, o projeto tem como objetivo o desenvolvimento de uma base robótica através de um microcontrolador ESP32 que em conjunto com a elaboração da programação, dispensa a necessidade de interação humana presencial, eliminando os riscos de contaminação.

O projeto partirá da utilização da carcaça de uma base robótica já existente. Sendo desenvolvidos o circuito eletroeletrônico e a programação, além de realizar o estudo do microcontrolador ESP32. Esta base robótica será autônoma, isto é, não necessitará que alguém a controle, e contará com um suporte para o *tablet*, dispositivo este que permitirá a interação com as pessoas.

1.1. TEMA

Implementação do circuito eletrônico e desenvolvimento da programação de um robô de telepresença.

1.2. DELIMITAÇÃO DO TEMA

Este trabalho é delimitado através do levantamento de referencial teórico, partindo assim para a programação e implementação no protótipo, voltado para utilização no primeiro atendimento em serviços clínicos.

1.3. PROBLEMA DE PESQUISA

A disseminação de vírus e infecções no âmbito hospitalar depende de uma fonte do microrganismo, de um hospedeiro e de um meio de transmissão. O meio de transmissão pode ocorrer através do contato direto com o microrganismo, quando um paciente contaminado entra em contato com um indivíduo suscetível e, também, com contato indireto, que ocorre quando o indivíduo suscetível entra em contato com um material contaminado. (CHEHUEN NETO, et al. 2006).

Quando o paciente se dirige a uma consulta em uma clínica médica, primeiramente é atendido por um profissional que atua na recepção, este atende todas as pessoas que chegam à clínica, ou seja, se uma pessoa a infecta, ela pode, a partir dali, infectar.

O robô de telepresença apresenta a função de auxiliar pacientes que chegam à clínica, sem haver a necessidade de contato humano, dependendo apenas do caminho que o robô irá seguir, que ao selecionar o consultório desejado, o robô acompanhará o paciente até o local.

Com a introdução desse robô em ambientes hospitalares haverá a possibilidade de diminuir contaminações, pois as doenças que se alastram com facilidade podem ser contraídas dentro da própria clínica, que é o recurso procurado para tratar todos os tipos de enfermidades.

Com base nas informações expostas, o protótipo realizado para o controle do robô de telepresença será capaz de distinguir pessoas e objetos em sua frente através do circuito implementado e da programação desenvolvida, evitando assim ficar travado?

1.4. HIPÓTESES

Ao partir para a escolha dos componentes a serem utilizados, percebe-se uma grande quantidade de equipamentos disponíveis, dos quais, muitos, são capazes de suprir as necessidades deste projeto. Acredita-se que o principal ponto, e que exigirá maior cuidado, é a escolha da placa que irá receber todos os componentes e sensores. Dessa forma:

- a) Optar por um microcontrolador que seja capaz de receber todos os sensores necessários para montar a base de um robô e que ainda suporte as programações.

Construir uma placa do zero exigiria muito tempo e acabaria desfocando do objetivo que é elaborar o circuito completo e não somente a placa.

1.5. JUSTIFICATIVA

Em hospitais, clínicas e redes de atendimento hospitalar, por exemplo, doenças infecciosas e altamente transmissíveis são muito comuns, pois é o lugar onde as pessoas buscam tratamento e, com isso, acabam expondo outras ao contágio. Apesar das doenças leves, com fácil e rápido tratamento já causarem preocupação, os maiores problemas estão nas doenças graves e com difícil tratamento, ou até inexistente (CHEHUEN NETO, et al. 2006).

Proteger pacientes, funcionários e visitantes nesses locais tem sido desafiante. A tecnologia apresenta técnicas inovadoras de automação de processos de risco, visto que a contaminação de indivíduos suscetíveis nestes locais é alta. Diante disso, para minimizar os riscos provenientes de indivíduos e materiais contaminados, é possível encontrar práticas que não tornam necessário o contato direto entre pessoas durante o pré-atendimento.

É possível verificar uma vasta tecnologia atualmente, que vem crescendo dia após dia, e precisamos usá-la a nosso favor. Com esse objetivo, de evitar a interação humana nessas situações, a maneira mais apropriada é usufruir dos recursos tecnológicos existentes hoje, nesse caso com um robô de telepresença.

O robô é responsável por recepcionar o paciente, realizar o encaminhamento à consulta, mostrando o caminho do consultório ao paciente, sem haver interação humana e sem exposição a objetos infectados. Cada pessoa que for afastada do risco, também evitará que ela contamine outras pessoas, e assim sucessivamente, isso trará mais tranquilidade às pessoas que acessarem o hospital e mais segurança a todos.

Do ponto de vista econômico, é possível constatar que um robô, a longo prazo, acarretará um custo mais baixo ao proprietário, ao mesmo tempo que sabemos que um trabalho realizado por um ser humano agrega valor. Mas nesse caso, o que é necessário levar em conta é a saúde e a segurança das pessoas, do funcionário e de todos que entram em contato com ele.

Para esses casos um robô de serviço torna-se a melhor opção, além de afastar o ser humano do risco, os gastos que o proprietário terá após a compra será com a energia usada no carregamento da bateria e a manutenção do mesmo, que com a disseminação da tecnologia a tendência é que se torne cada vez mais barata, aumentando o custo benefício.

1.6. OBJETIVOS

O objetivo do trabalho em questão é desenvolver e programar um robô de telepresença equipado com um *tablet*, que atua no acompanhamento do paciente até o consultório, não havendo necessidade de contato humano.

1.6.1. Objetivo Geral

O objetivo deste trabalho em questão se dá através do desenvolvimento do robô de telepresença por meio da elaboração da programação e implantação do esquema eletroeletrônico. O intuito é auxiliar pacientes mostrando o caminho até o consultório desejado e, por meio do acionamento de um botão, o robô acompanhará até o destino e retornará até sua posição inicial.

1.6.2. Objetivos Específicos

- a) Definir os requisitos de componentes sensoriais do projeto;
- b) Programar e definir as rotas do robô;
- c) Realizar simulação de funcionamento do aplicativo via *software*;
- d) Efetuar testes no robô de telepresença com a programação definitiva.

2. REVISÃO DA LITERATURA

2.1. ROBÔ

Os primeiros estudos do que se tornaria um robô começaram em 1942, nos Estados Unidos da América, pela necessidade de manusear materiais radioativos para a fabricação de bombas atômicas por meio do Projeto Manhattan, onde o intuito era evitar qualquer proximidade humana com materiais como urânio e plutônio (ROMERO (Org.), et al. 2014).

Romero (Org.), et al. (2014) ainda descreve que assim surgiu o primeiro telemanipulador da história, possuía duas extremidades fixadas no teto, uma delas era manipulada pelo cientista enquanto a outra entrava em contato direto com o material. Está longe de ser um robô como os que conhecemos hoje, mas o seu sucesso fez com que, na época, a indústria se interessasse por tecnologia e novos estudos nesta área iniciaram.

Hoje um robô é um sistema autônomo, que através de um corpo físico com sensores, efetadores, atuadores e de um controlador é capaz de sentir o ambiente e tomar suas decisões em relação a ele, podendo receber informações dos seres humanos, mas não ser controlado por eles. O que diferencia um robô do outro e suas funções são o *hardware* e o *software* utilizados na sua criação (MATARIC, 2014).

A ISO 8373 (2012), que trata de robôs e dispositivos robóticos, define um robô como sendo um mecanismo atuado programável, capaz de se mover no local onde for inserido, com autonomia para realizar as funções programadas e destinadas a ele, permitindo alterações de seus movimentos e funções programados com ou sem alterações físicas.

2.1.1. Corporalidade

Segundo Mataric (2014), a corporalidade significa ter um corpo físico, que é a primeira exigência se tratando de um robô, isso lhe dará a capacidade de se mover, possibilitando que ele exerça algum trabalho no mundo físico, qualquer esfera computacional, por mais interativa que seja, se for somente virtual, não se enquadra nesse conceito por não partilhar desse mundo.

2.1.2. Sensoriamento

De acordo com Aguirre (2007), o sensoriamento de um robô permite que ele sinta o ambiente e tenha noção do lugar que ocupa, isso tornará possível a execução das tarefas atribuídas a ele com segurança, no caso de um robô que se locomove é muito importante o uso de sensores externos para que ele possa ir de um ponto ao outro detectando obstáculos no seu caminho, podendo evitá-los.

Os sensores também são chamados de transdutores, pois transformam uma energia de determinada medida em outra forma de energia, ou seja, o sinal de entrada (luz, som, pressão, entre outros) é convertido em sinal analógico ou digital (ROMERO (Org.), et al. 2014).

Segundo Bolton (2010) sensores são transdutores, mas transdutores não são, necessariamente, sensores. Ainda afirma que o termo sensor é utilizado quando tal componente produz um sinal concatenado à grandeza aferida, já transdutor são definidos como componentes que ao serem submetidos a uma variação física, entregam uma outra variação relacionada.

Existe uma vasta gama de sensores para robôs móveis, exercendo desde funções mais simples, como medir a velocidade rotacional dos motores e a temperatura interna da eletrônica, até funções mais aprimoradas como informações sobre o ambiente e, até mesmo, a sua posição global. Os sensores auxiliam, também, no movimento do robô e o prepara para imprevistos no caminho, sendo essa uma função crítica (SIEGWART, 2011).

2.1.2.1. Classificação do sensor

A capacidade sensorial de um robô afeta diretamente no seu desempenho: na realização de tarefas, na capacidade de reagir e de agir com inteligência (MATARIC, 2014).

Mataric (2014) aborda também que, normalmente, um robô possui dois tipos de sensores que o fazem sentir o ambiente:

1. Sensores propioceptivos: São responsáveis pelo sensoriamento interno do robô. Propriocepção significa sentir o próprio corpo, pode ser usado tanto para animais quanto para robôs. No caso dos robôs, são sensores que fazem com que ele tenha noção de si mesmo, saiba o que faz parte dele.

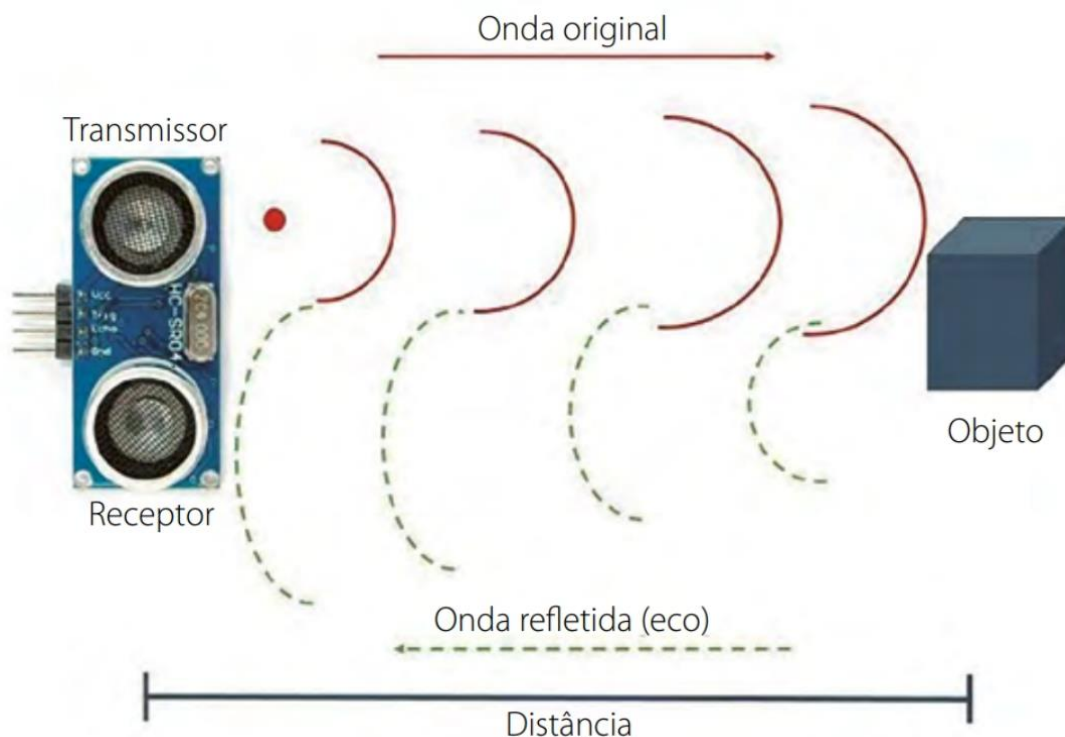
2. Sensores exteroceptivos: São responsáveis pela percepção externa do robô. Exterocepção se refere a sentir o mundo ao seu redor, ou seja, são sensores que mostram ao robô o lado de fora.

2.1.2.2. Sensor ultrassônico HC-SR04

O HC-SR04 é capaz de medir variáveis como distância e altura sem precisar entrar em contato com o objeto em questão, para isso, ele emite ondas sonoras acima de 40kHz que, ao entrar em contato com o obstáculo, retornam. Dessa forma é calculada a distância, de acordo com o tempo que o som leva para retornar a partir do momento que é emitido, como podemos observar na Figura 1 (PUHL, et.al. 2019).

Monk (2017) compartilha da mesma descrição e ilustra o seu funcionamento como podemos observar na Figura 1.

Figura 1 - Sensor ultrassônico



Fonte: Monk (2017).

Puhl et.al. (2019) ainda menciona que o seu funcionamento não é afetado por transparência, poeira, sujeira, vapores ou gases presentes no ambiente. O obstáculo só precisa ser capaz de refletir as ondas sonoras emitidas por ele.

De acordo com o manual do usuário *Cytron* (2013) ele detecta obstáculos de 2 cm a 400 cm e possui as características de acordo com o Quadro 1.

Quadro 1 – Características do sensor ultrassônico

Alimentação	+ 5V DC
Corrente de operação	< 2mA
Ângulo do sensor	< 15 graus
Resolução	3mm
Sinal de entrada trigger	10uS

Fonte: Manual do usuário *Cytron* (2013).

2.1.3. Efetadores e atuadores

Os efetadores e atuadores são os sinais de saída, e responsáveis por colocar em prática os comandos enviados pela programação, eles irão executar a tarefa predefinida (NORVIG, 2013).

Mataric (2014) descreve um efetador como um componente do robô que interage com o ambiente. Podem ser pernas, rodas, braços, dedos, entre muitos outros. O efetador recebe o comando do controlador e realiza a tarefa atribuída a ele na programação, fazem o trabalho físico e devem ser bem adaptados à sua função.

Atuador é um mecanismo que permite que o efetador execute a sua função. Fazendo uma relação com o corpo humano, os atuadores agem como os músculos e tendões permitindo a realização dos movimentos. Nos robôs os atuadores são os motores elétricos, dispositivos hidráulicos e pneumáticos, materiais sensíveis a temperatura e componentes químicos, são alguns exemplos, mas temos uma vasta gama de tecnologias (MATARIC, 2014).

2.1.3.1. Motores elétricos

Motores são atuadores, e, na área da robótica, são os mais comuns desse segmento. Eles exercem movimento de rotação e funcionam muito bem quando acoplados a efetadores, como por exemplo as rodas, são os motores que fazem com que elas girem (MATARIC, 2014). Existem vários tipos de motores elétricos, mas

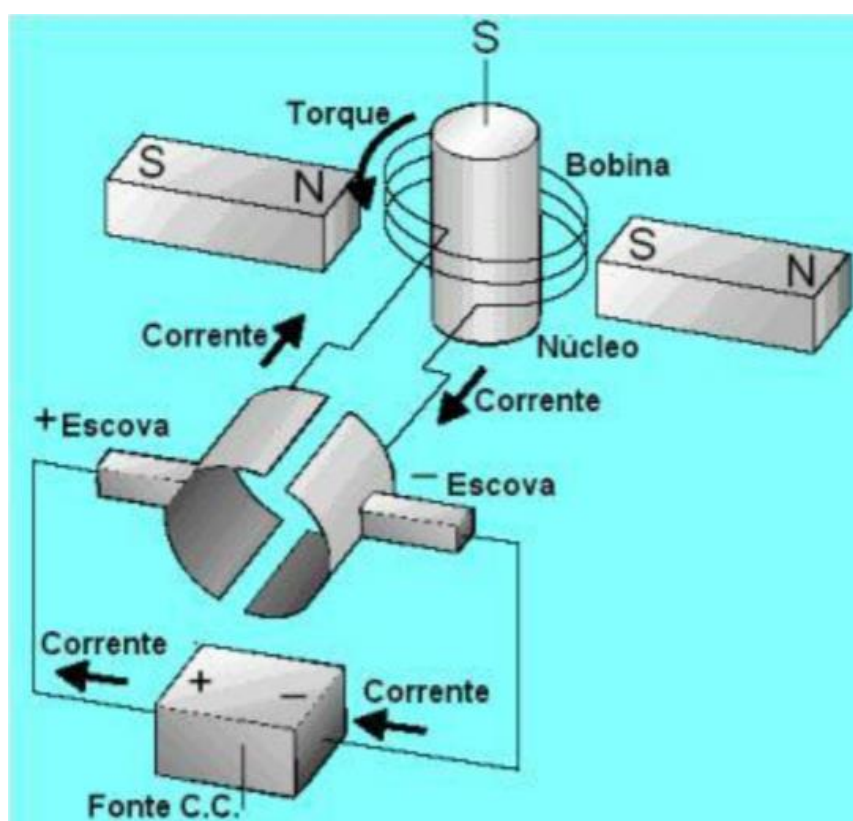
vamos abordar os mais comuns na robótica, que são os motores de passo e os motores de corrente contínua.

2.1.3.2. Motor de corrente contínua

Do ramo de efetadores, Mataric (2014) destaca que os motores de corrente contínua (CC) são os mais fáceis de encontrar, além de mais baratos e mais simples de usar. Assim como o motor de passo, os motores CC também convertem pulsos elétricos em energia mecânica, para isso utilizam ímãs, bobinas e correntes para gerar campos magnéticos que irão realizar o movimento de rotação do eixo do motor.

A escolha desse motor se dá em situações que exigem alto torque e ampla variação de velocidade, seu tamanho costuma ser maior do que os motores de passo e possuem maior necessidade de manutenção. A Figura 2 mostra um desenho esquemático bastante simplificado de um motor CC com apenas uma bobina, o comutador e as escovas (SIEMENS, 2006).

Figura 2 – Modelo de motor CC.



Fonte: SIEMENS (2006).

2.1.4. Controladores

Os controladores põem em prática a autonomia do robô fornecendo o *hardware* e/ou o *software*, com base nas informações vindas dos sensores, e de qualquer outra referência que esteja em sua memória, eles tomam as decisões e em seguida controlam os efetadores para que executem as funções escolhidas (MATARIC, 2014).

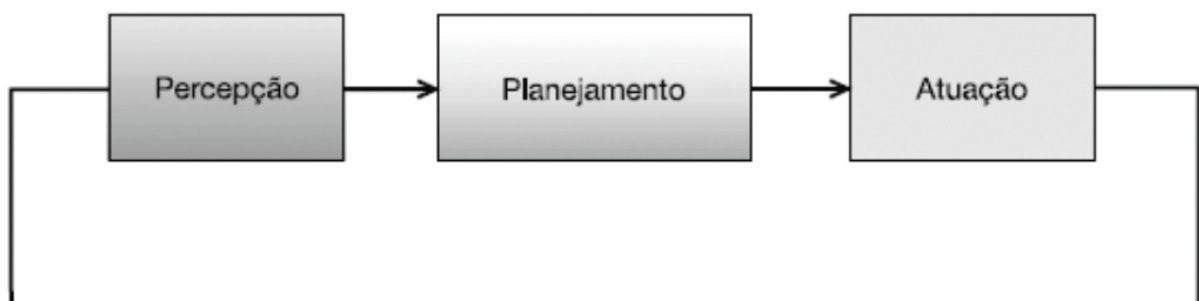
2.1.5. Arquiteturas de controle

Romero (Org.), et al. (2014) aponta que, de modo geral, os componentes básicos de um sistema de controle, que compõem uma arquitetura, são divididos em três grupos principais:

1. Percepção: responsável pela interpretação, modelagem de mundo e integração dos sensores;
2. Planejamento: planeja, sincroniza e monitora o cumprimento das tarefas do robô;
3. Atuação: responsável pela ação do robô, controla os atuadores para a realização dos movimentos.

Na Figura 3 podemos observar um dos possíveis ciclos compostos por esses grupos, como o ambiente pode ser modificado, ocorre uma repetição nessa série a cada período de tempo.

Figura 3 – Ciclo percepção-planejamento-atuação.



Fonte: Romero (Org.), et al (2014).

Segundo Grassi (2006), a arquitetura está mais ligada ao *software*, que são o conjunto de componentes lógicos do sistema, do que ao *hardware*, que se identificam

por ser o equipamento físico utilizado em computadores e eletrônicos, é a maneira que se constrói um software de controle inteligente, indicando os módulos que devem compor um sistema e como eles interagem entre si.

2.1.6. Microprocessadores e Microcontroladores

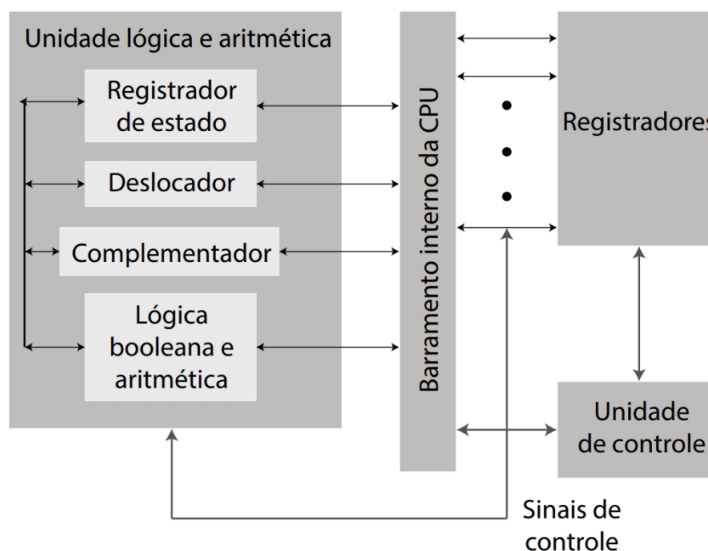
Com o avanço da tecnologia os sistemas digitais estão cada vez mais presentes no nosso dia a dia. Os pequenos computadores deixaram de ser apenas imaginário de ficção científica e viraram parte do nosso cotidiano. Estão nos carros, nos brinquedos, nos eletrodomésticos, em praticamente todos os lugares, tudo para realizarem funções que facilitam a nossa vida. Esses sistemas estão aptos a realizar inúmeras funções porque utilizam microprocessadores e microcontroladores que, com a evolução, se tornaram sistemas mais eficientes e baratos (PUHL, et.al. 2019).

Lenz (2019) complementa que os métodos e procedimentos aplicados à automação, controle e processamento evoluíram muito com a eletrônica digital. Com essa evolução, surgem os microprocessadores que, muito rapidamente, estão aumentando a sua potência e agilidade. Outros dispositivos, desse mesmo conceito, mas com menor capacidade, são os microcontroladores, que quando integrados a outros, permitem um desenvolvimento menos complexo.

Puhl (2019) aponta que os dispositivos que usamos, munido de inúmeros atributos, só possuem essas capacidades porque utilizam microprocessadores e microcontroladores, e suas características são:

- **Microprocessador:** também chamado apenas de processador, é encarregado de realizar todas as operações matemáticas e lógicas de um computador. Ele é um circuito integrado composto por uma unidade lógica e aritmética (ULA), uma unidade de controle (UC), registradores e uma unidade de gerenciamento de memória (MMU), como podemos observar na Figura 4.

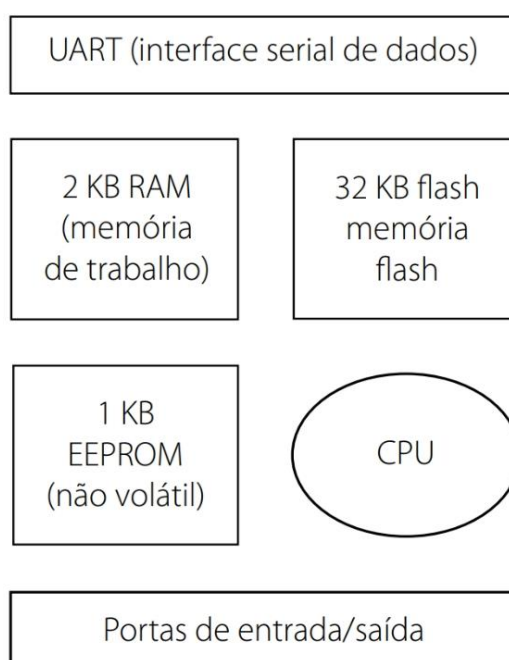
Figura 4 – Modelo de processador.



Fonte: Puhl, et.al (2019).

- **Microcontrolador:** como se fosse um mini computador existente em um circuito integrado. Possui uma unidade de processamento, memória e pinos de entrada e saída. São um tipo de tecnologia embarcada feitos especificamente para o dispositivo em que serão inseridos. Vemos na Figura 5 um modelo de microcontrolador utilizado em placas de Arduino Uno.

Figura 5 – Modelo de microcontrolador ATmega328.











Fonte: Puhl, et.al. (2019).

Apesar de parecidos, as diferenças entre um microprocessador e um microcontrolador são muitas, a começar pelo ramo da arquitetura e as atribuições que os dispositivos irão desempenhar. Microprocessadores são focados em computadores enquanto os microcontroladores são voltados para sistemas embarcados. Microprocessadores são capazes de receber programações em várias linguagens de alto nível, enquanto os microcontroladores, geralmente, operam apenas em linguagem C ou *Assembly* (PUHL, et.al. 2019).

2.1.6.1. Arduino

Arduino é uma plataforma de *hardware* com *software* livre baseada em microcontroladores da ATMEL, possui CPU, memória interna, comunicação serial, conversores analógico-digitais, dispositivos PWM, entre outros. Sua plataforma possui interface padrão, o que facilita no estudo e aprendizagem. Demonstrados na Figura 6 os tipos de Arduino e suas principais características (STEVAN JR., 2015).

Figura 6 – Comparativo entre as versões do Arduino.

	Arduino Uno	Arduino Mega2560	Arduino Leonardo	Arduino Due	Arduino ADK	Arduino Nano	Arduino Pro Mini	Arduino Esplora
								
Microcontrolador	ATmega328	ATmega2560	ATmega32u4	AT91SAM3X8E	ATmega2560	ATmega168 (versão 2.x) ou ATmega328 (versão 3.x)	ATmega168	ATmega32u4
Portas digitais	14	54	20	54	54	14	14	-
Portas PWM	6	15	7	12	15	6	6	-
Portas analógicas	6	16	12	12	16	8	8	-
Memória	32 K (0,5 K usado pelo bootloader)	256 K (8 K usados pelo bootloader)	32 K (4 K usados pelo bootloader)	512 K disponível para aplicações	256 K (8 K usados pelo bootloader)	16 K (ATmega168) ou 32K (ATmega328), 2 K usados pelo bootloader	16 K (2k usados pelo bootloader)	32 K (4 K usados pelo bootloader)
Clock	16 Mhz	16 Mhz	16 Mhz	84 Mhz	16 Mhz	16 Mhz	8 Mhz (modelo 3.3v) ou 16 Mhz (modelo 5v)	16 Mhz
Conexão	USB	USB	Micro USB	Micro USB	USB	USB Mini-B	Serial / Módulo USB externo	Micro USB
Conector para alimentação externa	Sim	Sim	Sim	Sim	Sim	Não	Não	Não
Tensão de operação	5v	5v	5v	3.3v	5v	5v	3.3v ou 5v, dependendo do modelo	5v
Corrente máxima portas E/S	40 mA	40 mA	40 mA	130 mA	40 mA	40 mA	40 mA	-
Alimentação	7 - 12 Vdc	7 - 12 Vdc	7 - 12 Vdc	7 - 12 Vdc	7 - 12 Vdc	7 - 12 Vdc	3.35 - 12 V (modelo 3.3v), ou 5 - 12 V (modelo 5v)	5v

Fonte: Adaptado de Arduino (2021).

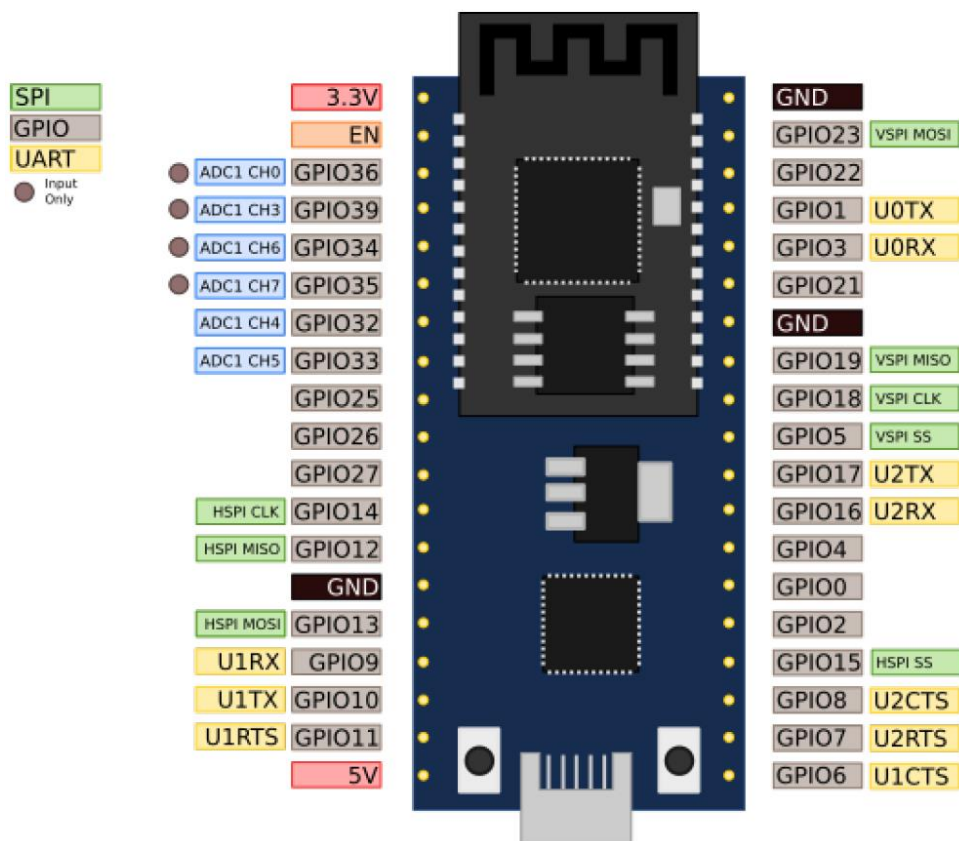
2.1.6.2. ESP32

O ESP32 é um microcontrolador desenvolvido pela *Espressif Systems*, lançado em 2016. Ele é um sistema dual-core que executa as instruções de *Xtensa LX6*, possui *Wi-Fi* e *Bluetooth* integrados e um bom consumo de energia, podendo consumir 260mA em potência total e 20uA em modo soneca (KOLBAN, 2017).

Ele possui duas CPUs, chamadas de “PRO_CPU” e “APP_CPU” (para protocolo e aplicação, respectivamente), ambas com 4 GB (32 bits), onde o mapeamento delas é simétrico, ou seja, elas usam os mesmos endereços para acessar a mesma memória e para a maioria das finalidades as duas CPUs são intercambiáveis (ESPRESSIF, 2021).

Juntamente com o lançamento do ESP32, a *Espressif* lançou sua própria placa para torná-lo mais atrativo aos consumidores, ela se chama ESP32-DevKit. Ela contém suporte para o ESP32, entrada micro USB e dois botões chamados EN, para habilitar, e *Boot*, para inicializar. Na Figura 7 estão apresentados com mais detalhes os seus componentes e a distribuição dos pinos (KOLBAN, 2017).

Figura 7 – ESP32 e suas conexões



Fonte: Kolban (2017).

Sua frequência de *clock* pode chegar a 240 Mhz e tem uma grande capacidade de armazenamento, se comparado com ATmega2560 pode chegar ao dobro, se levarmos em consideração a memória *flash*, mesmo o ESP32 tendo um custo mais baixo (IBRAHIM, 2019).

O principal diferencial do ESP32, e que contribui para que se tornasse um dos controladores mais indicados e utilizados em IOT, é a sua conectividade, apresentando dois módulos de integração com as redes sem fio, que são o *bluetooth* e o *Wi-Fi*, sendo um dos únicos de pequeno porte a reunir essas propriedades (KOLBAN, 2017).

No Quadro 2 temos a comparação do ESP32 com o Arduino UNO R3, que se encontra na mesma faixa de preço, podemos dessa forma perceber a sua superioridade.

Quadro 2 – Comparação do ESP32 com Arduino UNO

	ESP32	Arduino UNO R3
Corrente	220 mA	40 mA
Núcleo	2	1
Arquitetura	32 bits	8 bits
Clock	160 - 240 MHz	16 MHz
Wi-Fi	Sim	Não
Bluetooth	Sim	Não
RAM	520 KB	2 KB
FLASH	16 MB	32 KB
GPIO	22	12
DAC	2	0
ADC	18	6
Interfaces	SPI - I2C - UART - I2S - CAN	SPI - I2C - UART

Fonte: Adaptado de Espressif (2021) e Arduino (2021).

2.1.6.3. Buzzer

O dispositivo eletrônico chamado de *Buzzer* é utilizado na emissão de sinais sonoros por meio de uma tensão elétrica acionada em seus terminais e pode ser

encontrado em situações diárias como em campainhas e até mesmo em alarmes (STEVAN E ADAMSHUK, 2015).

O funcionamento do *Buzzer*, Figura 8, acontece através do efeito piezoelétrico reverso, que nada mais é do que o surgimento de uma tensão mediante a um esforço mecânico. Quando aplicada a tensão em seus terminais, a célula piezoelétrica começará a vibrar e assim, irá produzir o som, este, podendo variar de acordo com a frequência (STEVAN E ADAMSHUK, 2015).

Figura 8 – Buzzer

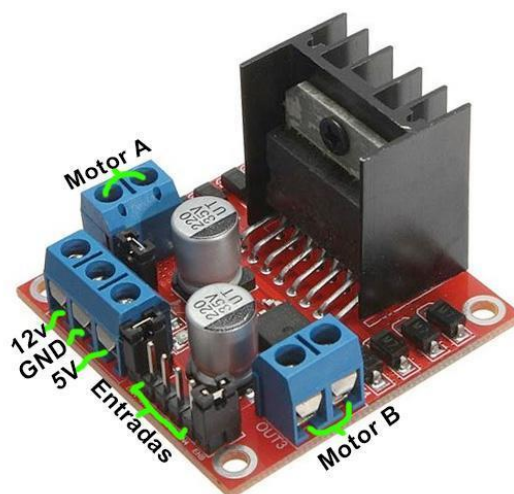


Fonte: Honytek (2018).

2.1.6.4. Ponte H

O módulo driver ponte H, de modelo L298N, é muito utilizado devido ao seu custo-benefício, e seu objetivo é controlar motores, seja de corrente contínua ou motor de passo, ele é capaz de trabalhar com dois motores simultaneamente, tornando assim possível o controle e direção do robô (STEVAN E ADAMSHUK, 2015).

Figura 9 – Ponte H



Fonte: Autores (2021).

Na Figura 9, podemos identificar na Ponte H as portas utilizadas para a realização do controle dos motores, são elas duas entradas (uma negativa e outra positiva) para cada motor (A e B), entrada de energia 12v que é conectada diretamente na bateria e também o GND (fio terra), e a saída de 5v usada para alimentar a placa ESP32. As portas de entradas são utilizadas diretamente na placa ESP32, onde são enviadas as informações de funcionamento dos motores.

Na Tabela 3, as características do módulo driver Ponte H L298N podem ser observadas de maneira mais clara:

Tabela 1 – Características da Ponte H L298N

Tensão de operação	De 4v a 35v
Corrente de operação máxima	2A
Tensão lógica	5V
Corrente lógica	0 a 36mA
Potência máxima	25W

Fonte: Eletrogate (2020).

2.1.7. Programação

A programação é o que torna um robô autônomo, para isso devem ser programados de acordo com a sua função. Esse programa deverá processar os dados

enviados pelos sensores contidos no robô e, através disso, enviar comandos para os atuadores. (AGUIRRE, 2007).

A base de uma programação são os algoritmos e a lógica de programação, um algoritmo representa uma sequência de passos que devem ser seguidos para a resolução de um problema, o estudo disso é indispensável para a criação de um *software*. Esse algoritmo para a criação de um *software* pode ser escrito em várias linguagens de programação como Pascal, C, Java e outras (SOUZA, et al. 2019).

A linguagem de programação permite que o usuário crie um programa a partir de uma sequência de ordens, dados e ações, executando assim o controle do comportamento lógico e físico do produto desenvolvido (AGUILAR, 2008).

2.1.7.1. Linguagem C

C é uma linguagem de alto nível muito usada para programar microprocessadores. É fácil de ser empregada e um mesmo programa pode ser usado em vários microprocessadores, basta utilizar o compilador específico de cada um, pois a linguagem C é padronizada pela ANSI (*American National Standards Institute*) (BOLTON, 2010).

A linguagem C é, atualmente, uma das mais populares do mercado, isso acontece devido às suas diversas vantagens apresentadas. Através dela é possível criar aplicativos para diversos dispositivos e sistemas operacionais, ou seja, possui um alto nível de portabilidade, além de ser confiável e gerar um código eficiente (DAMAS, 2007).

Esta linguagem é constituída por funções, e essas funções possuem uma rotina a qual é programada para realizar uma tarefa específica, e assim podendo também definir quais tipos de dados essas funções vão receber. Um exemplo é a obtenção de informações recebidas por um sensor ultrassônico, e utilizando essas informações, para realizar uma tarefa, como informar a distância que está um objeto em sua frente (DAMAS, 2007).

Uma das funções mais importantes da linguagem de programação C é a *main*, ela além de controlar o fluxo de chamada das demais funções, ela determina o quando o programa dará início (DAMAS, 2007).

Além das informações já mencionadas, os laços de repetição também são muito utilizados, como seu próprio nome diz, é executada um conjunto de instruções

enquanto uma condição é verdadeira. Como por exemplo, executar a função que faz ligar o motor enquanto o botão 'n' está acionado (DAMAS, 2007).

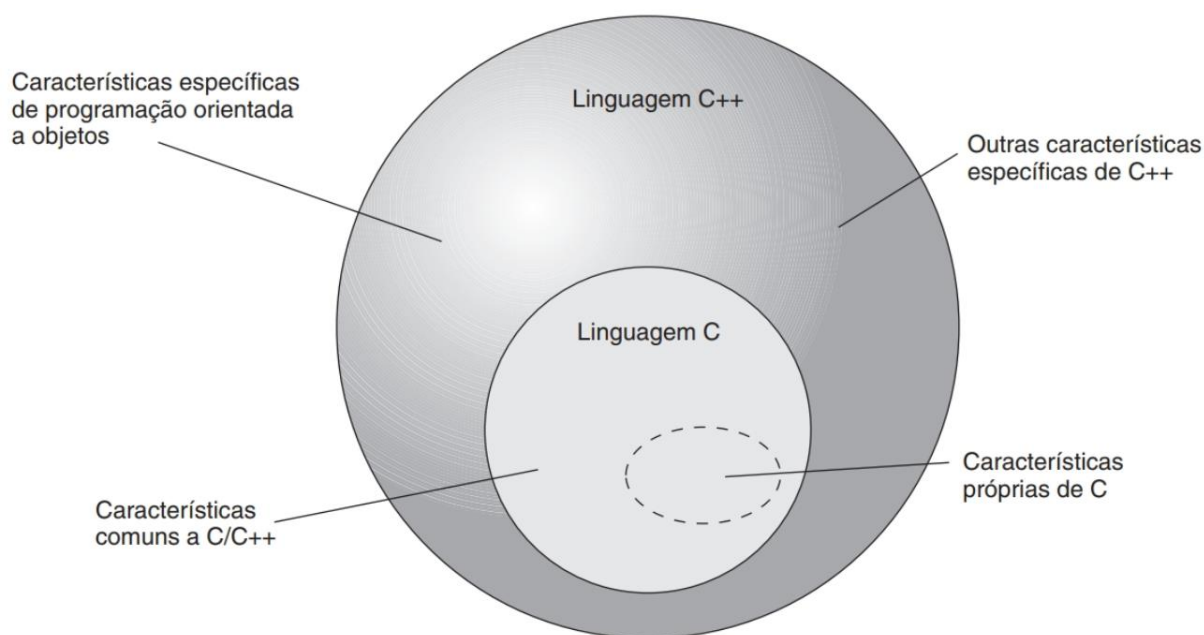
As bibliotecas que são comumente vistas em diversas linguagens de programação, também são utilizadas em C, elas possuem operações comuns, como o tratamento de dados das entradas e saídas (DAMAS, 2007).

2.1.7.2. Linguagem C++

C++ é uma extensão de C com atributos mais potentes, essa linguagem de programação utiliza sua própria biblioteca-padrão, mas também utiliza a biblioteca-padrão de C, definida pela ANSI. O padrão C agregou algumas propriedades que suportam alguns compiladores de C++, por esse motivo, quase todas as sentenças de C têm uma sentença correta em C++, mas o inverso não acontece (AGUILAR, 2011).

Aguilar (2011) ainda aponta que, C++ já foi chamada de "C com classes", pois é a linguagem C com o incremento de classes, objetos e programação orientada a objetos, e com uma melhora da entrada/saída (E/S). Na Figura 10 podemos observar a relação entre C e C++.

Figura 10 – Relação entre C e C++



Fonte: Aguilar (2011)

É possível fazer um programa em C++ semelhante a um em C, mas isso raramente acontece. Os programadores são estimulados, pela própria programação em C++, a utilizarem as suas características, mas é comum programadores utilizarem também as características específicas de C. Ter conhecimento da linguagem C facilita o aprendizado da linguagem C++, mas é possível iniciar a aprendizagem direto com a C++ (AGUILAR, 2011).

2.1.7.3. Programação ESP32

A linguagem utilizada para o ESP32 é a C/C++, e ela pode ser programada utilizando diversos *softwares* compatíveis, alguns exemplos são o *Software Development Kit* (SDK), que é fornecido por sua própria desenvolvedora e outro, também muito utilizado, é a IDE do Arduino (KOLBAN, 2017).

3. METODOLOGIA

Para a realização e execução do presente projeto se seguirá o método de pesquisa explicativa, que, segundo Gil (2008), foca em identificar as razões que indicam ou contribuem para o acontecimento do caso. Se trata de um estudo mais complexo, pois se aprofunda na realidade da pesquisa e explica a razão do ocorrido, expõe os fatos e os porquês.

O conhecimento científico se dá através da análise dos resultados, mesmo sendo exigido uma descrição detalhada do processo que o gerou. Por sua complexidade ela se torna delicada e com alto grau de dificuldade, já que o risco de cometer erros é maior ao pôr algo em prática (GIL, 2008).

Segundo Prodanov e Freitas (2013), o método de pesquisa é uma maneira para chegar a um objetivo, cuja finalidade dessa ciência é a busca do saber. Afirma-se então que este método se resume ao agrupamento de processos com a finalidade de alcançar o entendimento e o saber.

Para PMI (2013), ao possuir um projeto, este, pode ser dividido no número de fases necessárias, sendo que devem ser um conjunto de atividades relacionadas de maneira lógica, e que geralmente são sequenciais, e a sua quantidade é determinada pela necessidade de gerenciamento e controle do projeto.

A partir da metodologia apresentada, inicia-se a pesquisa acerca dos componentes que irão integrar o robô, partindo do princípio de que o corpo do mesmo será utilizado de um projeto anteriormente implementado, porém, passará por melhorias para adaptá-lo ao novo projeto, os próximos passos são a escolha da placa, dos sensores, efetadores e atuadores, com base nas referências bibliográficas pesquisadas.

Com todos os componentes definidos e adquiridos, com base nas pesquisas realizadas, inicia-se a construção da eletrônica que será inserida na base robótica, e com o intuito de obter uma melhor visualização do circuito eletrônico e antecipar possíveis falhas, foi elaborado o circuito em um *software* que permite a criação de projetos eletrônicos, incluindo a captura esquemática e simulação do circuito proposto.

Através da simulação em *software* do esquema elétrico, foi realizada a montagem do sistema, onde os componentes necessários foram conectados a placa ESP32, para assim ser possível a comunicação entre a programação elaborada e os *hardwares*.

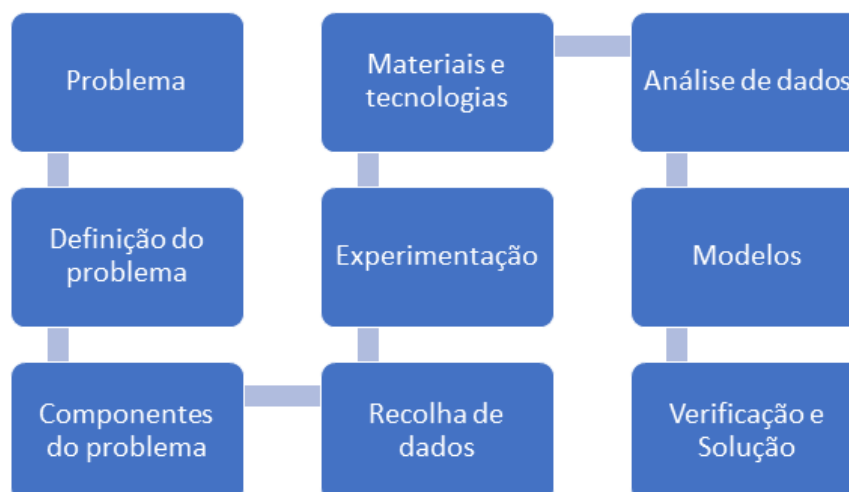
A próxima etapa é constituída pela montagem e fixação dos componentes na base, fazendo com que não haja falhas devido à vibração gerada quando o robô se mover, para então os testes serem executados e validar e identificar possíveis inexatidões, e assim aplicar as melhorias decorrentes no projeto.

3.1. MÉTODOS E TÉCNICAS UTILIZADOS

O método utilizado para a execução deste projeto é retratado por Bruno Munari no ano de 1981, e no livro *Das Coisas Nascem Coisas*. Segundo Munari (1981), o problema não é solucionado sozinho, mas traz também os segmentos utilizados para a sua solução, sabendo sua função e como utilizá-los.

Para a solução do problema, pode ser seguido os componentes da figura 11:

Figura 11 – Método Munari



Fonte: Autores (2021).

3.2. MATERIAIS E EQUIPAMENTOS

Neste item são apresentados os recursos utilizados para a elaboração e execução do projeto, detalhando assim os *softwares* e *hardwares* aplicados, conforme Quadro 3.

Quadro 3 – Recursos utilizados no projeto

Excel	É destinado a criação de planilhas eletrônicas e muito aplicado no gerenciamento de fluxos. Neste projeto, foi utilizado para o controle de componentes e administração de custos.
-------	--

Proteus Design Suite	<i>Software</i> utilizado para a criação de projetos eletrônicos, possibilita a captura esquemática e a simulação do circuito. Foi empregado para a simulação do projeto elétrico.
IDE Arduino	É um <i>software open-source</i> que possibilita um ambiente integrado para programação. Foi utilizado para realização da programação da placa ESP32.
Motor CC MR 210-60	O motor é uma máquina elétrica a qual converte energia de corrente contínua em energia mecânica, gerando o torque de saída. Foi empregado como dispositivo para locomoção e direcionamento do robô.
ESP32	É uma plataforma que possui um microcontrolador e suporte de entrada e saída embutido. Foi utilizado para a realização do controle do robô.
Sensor ultrassônico HC-SR04	<i>Hardware</i> projetado para medir distâncias de objetos através de ondas sonoras. Foi empregado para calcular a distância entre possíveis obstáculos.
Rodas	As rodas foram utilizadas para dar movimento ao robô e guiá-lo em seu percurso.
Botões	Os botões são usados para controlar processos, neste caso, foram empregados para dar início a uma função programada do robô.
Bateria	A bateria foi empregada com o intuito de entregar a energia necessária aos componentes elétricos do robô.
<i>Buzzer</i>	É um dispositivo de sinal sonoro, bastante utilizado em campanhas e dispositivos de alarme. Foi utilizado para ser acionado quando o robô ficar muito próximo de um obstáculo.
Ponte H L298	A ponte H é utilizada para controlar cargas indutivas como solenoides, relés, motores CC e motores de passo. Foi empregado no controle dos motores de corrente contínua.

Fonte: Autores (2021).

4. APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

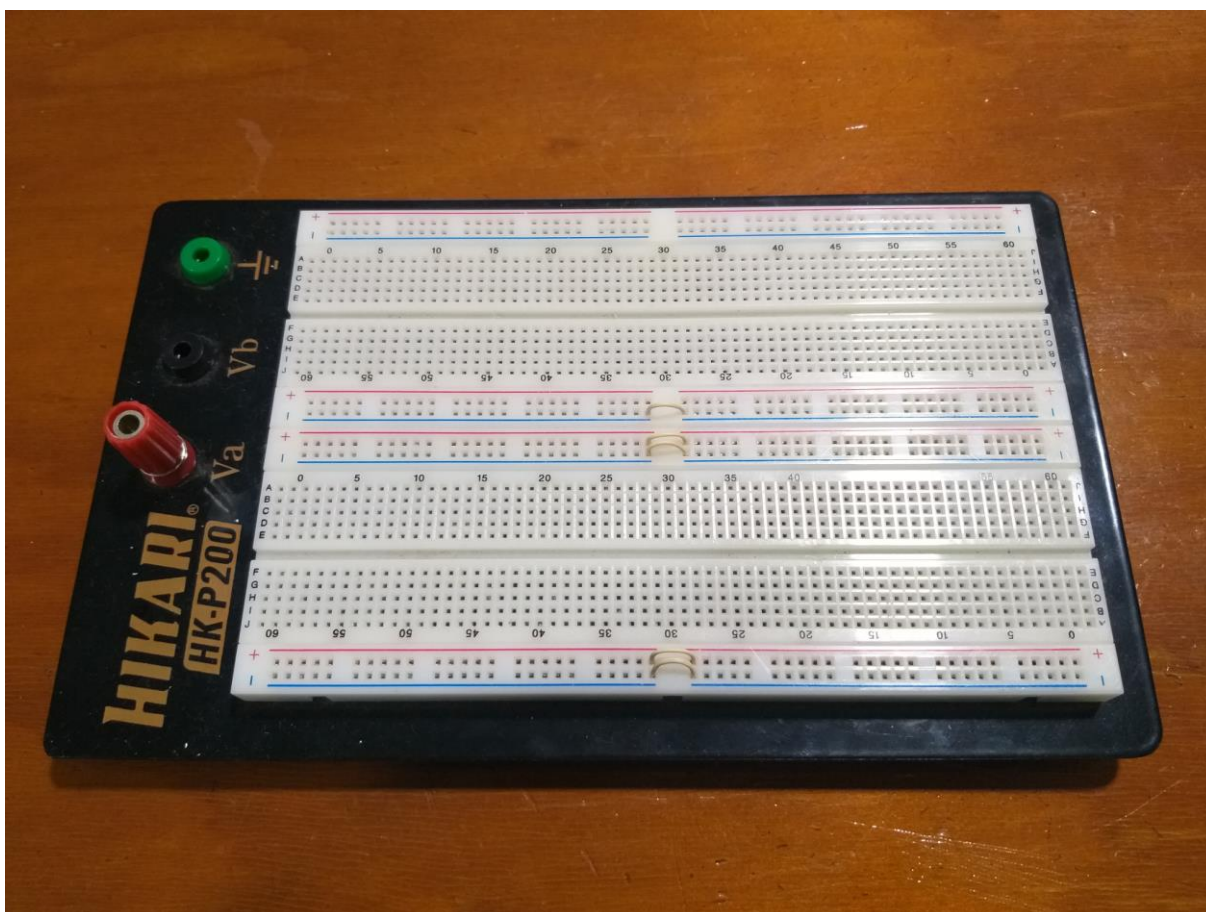
Neste tópico são expostos os resultados obtidos através da metodologia retratada no capítulo anterior. São apresentadas todas as etapas do projeto, exibindo seu passo a passo até ser realizado o protótipo final.

4.1. PRINCIPAIS COMPONENTES UTILIZADOS

Nesta seção são apresentados os principais componentes definidos para a realização do presente projeto. O que auxiliou na escolha de cada item foi a finalização do referencial teórico, onde analisou-se as características para as melhores escolhas dentre os citados.

Na Figura 12, temos uma *protoboard*, ela foi utilizada tanto nos testes de bancada quanto na versão final. Nela foram feitas todas as ligações eletrônicas, como a conexão dos sensores e motores ao ESP32.

Figura 12 – Protoboard



Fonte: Autores (2021).

Como podemos observar, na Figura 13 temos um sensor ultrassônico HC-SR04, utilizou-se três unidades deste componente. Como mencionado no item 2.1.2.2, uma de suas funções é detectar obstáculos que estão a uma determinada distância dele, e ele foi empregado no projeto com essa função.

Figura 13 – Sensores ultrassônicos



Fonte: Autores (2021).

O *buzzer*, apresentado na Figura 14, tem a função de emitir sinais sonoros. Neste projeto, quando o robô é barrado por algum obstáculo, o *buzzer* é acionado e permanece ativo por alguns segundos até a saída deste obstáculo em questão.

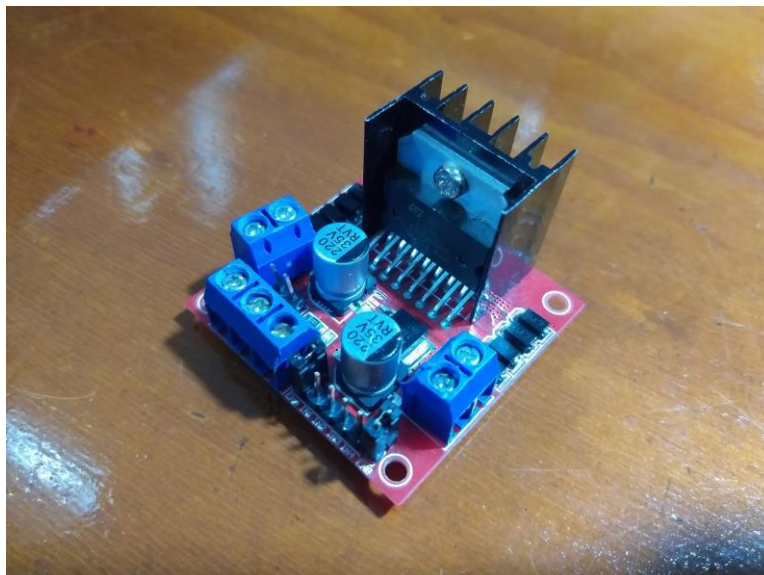
Figura 14 – Buzzer de 12v utilizado no protótipo



Fonte: Autores (2021).

Na Figura 15 tem-se a ponte H utilizada no projeto, e é responsável por regular a distribuição de energia que ela recebe, nesse caso, por meio de uma bateria de 12 V. É através dela que os motores e a placa são energizados.

Figura 15 – Driver Ponte H L298N



Fonte: Autores (2021)

O microcontrolador escolhido para gerenciar esse projeto é o ESP32, Figura 16, neste dispositivo são conectados os sensores, a ponte H, os botões e o *buzzer*. É ele que recebe a programação e se torna o responsável por enviar os comandos a todos os componentes conectados a ele, para que executem todas as funções incumbidas ao robô, comandos esses vindos da programação.

Figura 16 – Placa ESP32



Fonte: Autores (2021).

Botões, Figura 17, em geral, são feitos para acionar comandos, no caso do presente projeto, eles são usados para acionar rotas que o robô deve seguir, definidas na programação, são duas rotas disponíveis, uma para cada botão.

Figura 17 – Botões de comando



Fonte: Autores (2021).

O motor MR 210-60 é um motor de corrente contínua utilizado para realizar a movimentação das rodas do robô, deslocando-o para frente e para trás, e também para a execução das curvas, para a direita e para a esquerda. Foram utilizadas duas unidades deste item, conforme Figura 18.

Figura 18 – Motores de corrente contínua



Fonte: Autores (2021).

A bateria utilizada foi uma de 12 V, mostrada na Figura 19, ela é ligada somente na ponte H, para que ela faça a distribuição correta de energia necessária para cada componente.

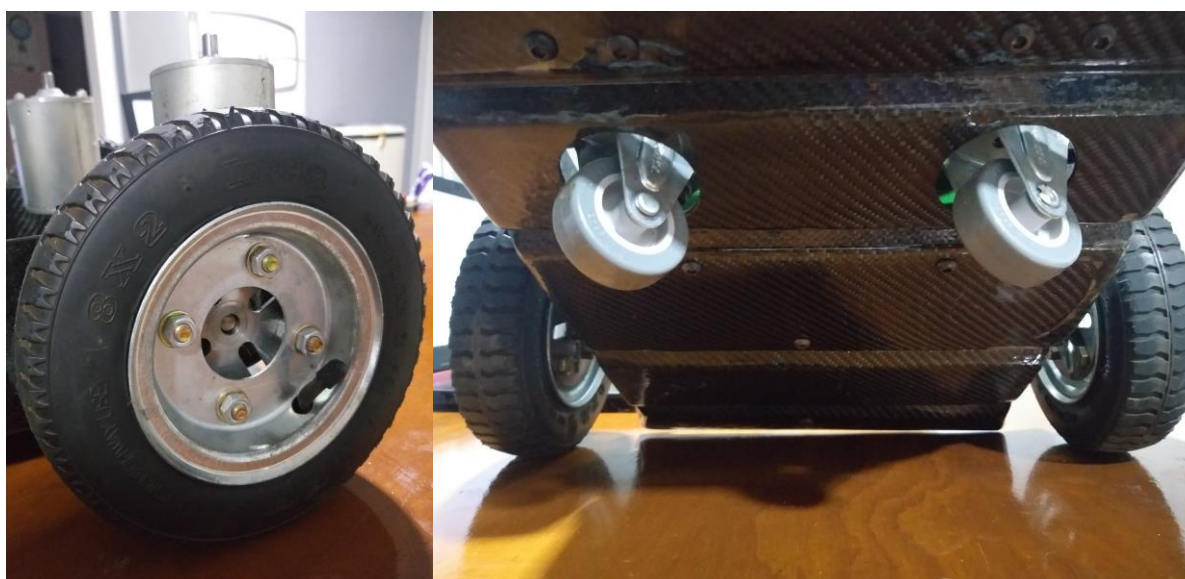
Figura 19 – Bateria 12v



Fonte: Autores (2021).

Foram utilizadas dois pares de rodas, Figura 20, um par delas é com pneu de borracha, para ter aderência ao solo, ficam localizadas na parte traseira do robô e são responsáveis pela movimentação do mesmo. O outro par é de plástico e fica localizado na parte dianteira do robô, dando suporte ao movimento. Os motores estão conectados apenas no par traseiro.

Figura 20 – Pares de rodas



Fonte: Autores (2021).

Após a definição dos componentes necessários para a realização deste projeto, foi realizada a compra destes materiais, podendo ser verificado na Quadro 4 os componentes adquiridos e seu custo.

Quadro 4 – Quantidade e valores de materiais utilizados

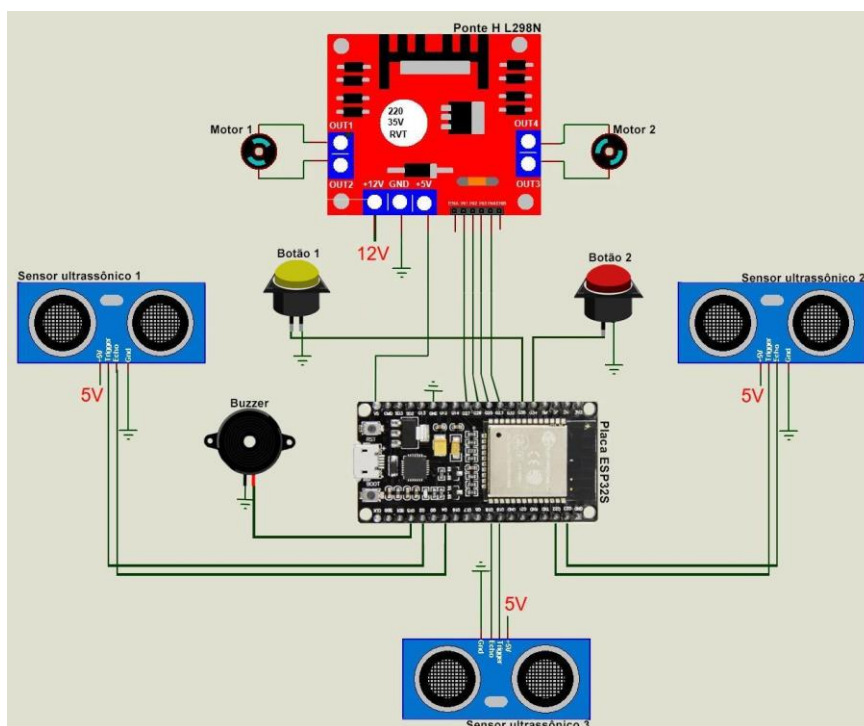
Componente	Quantidade	Valor total
Botão	2	R\$8,00
Placa ESP32	1	R\$78,38
Ponte H	1	R\$22,71
Protoboard	1	R\$99,90
Jumper	30	R\$29,90
Motor	2	R\$620,00
Roda frontal	2	R\$35,80
Roda traseira	2	R\$157,78
Bateria	1	R\$523,95
Sensor ultrassônico	3	R\$38,49
Total	-	R\$1614,91

Fonte: Autores (2021).

4.2. TESTE DE BANCADA

O teste de bancada foi iniciado a partir da montagem do circuito eletrônico utilizando os dispositivos elétricos, como os motores, a placa ESP32S, sensores e *buzzer*. Na Figura 21 pode ser observado o esquema elétrico.

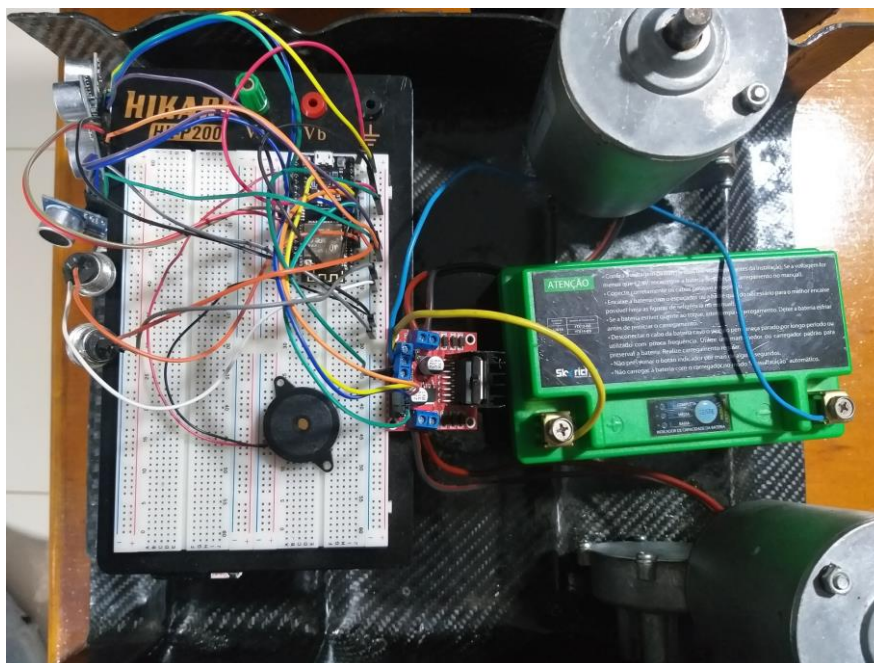
Figura 21 – Esquema elétrico.



Fonte: Autores (2021).

Através do esquema elétrico realizado, os testes no próprio *software* Proteus realizados e com resultado funcional, o teste físico de bancada é iniciado e montado seguindo este esquema, garantindo assim um funcionamento eficaz.

Figura 22 – Circuito elétrico



Fonte: Autores (2021).

Na Figura 22 observa-se de forma mais clara as conexões realizadas no sistema, onde a placa ESP32 está ligada aos sensores, botões, *buzzer* e na Ponte H, a qual é conectada aos motores.

4.2.1. SUPORTE PARA O *TABLET*

O robô possui uma estrutura onde são acoplados todos os componentes, e uma tampa superior (Figura 23), a qual possibilita a adição do suporte utilizado para segurar o *tablet*.

Figura 23 – Tampa do robô



Fonte: Autores (2021).

Este suporte, Figura 24, foi desenvolvido também para serem inseridos os botões de comando, onde quando pressionados irão realizar uma ação, então foi desenhado em *software*, onde há uma base fixadora para ser acoplada na tampa do robô, um suporte que deixa o *tablet* mais alto e torna mais fácil seu uso por uma pessoa que está de pé, e no seu topo o suporte para o *tablet* e botões.

Figura 24 – Suporte do *tablet* e botões.

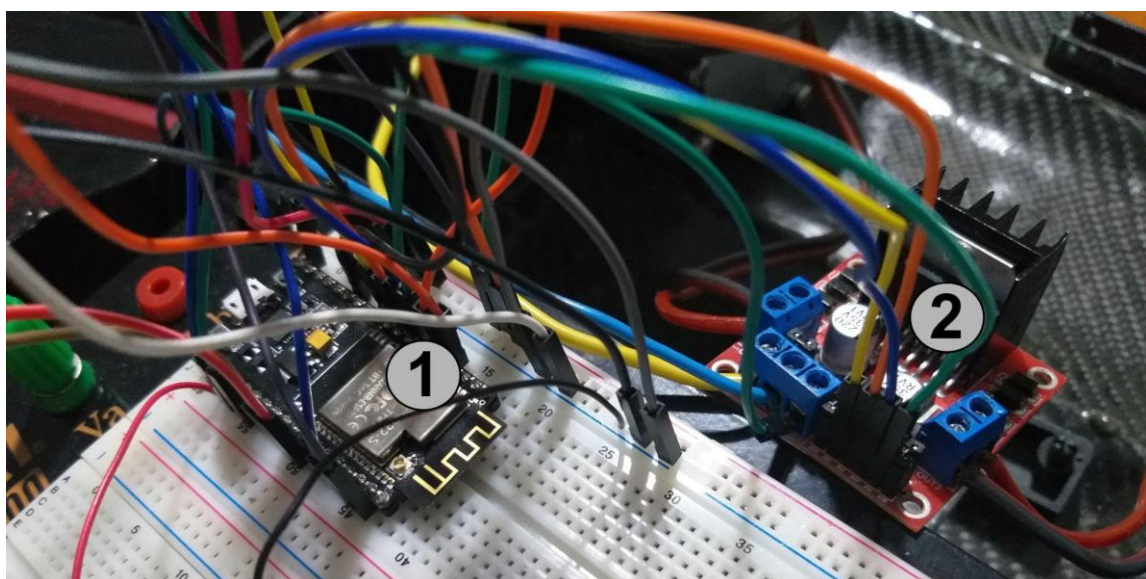


Fonte: Autores (2021).

4.3. MONTAGEM DO PROTÓTIPO

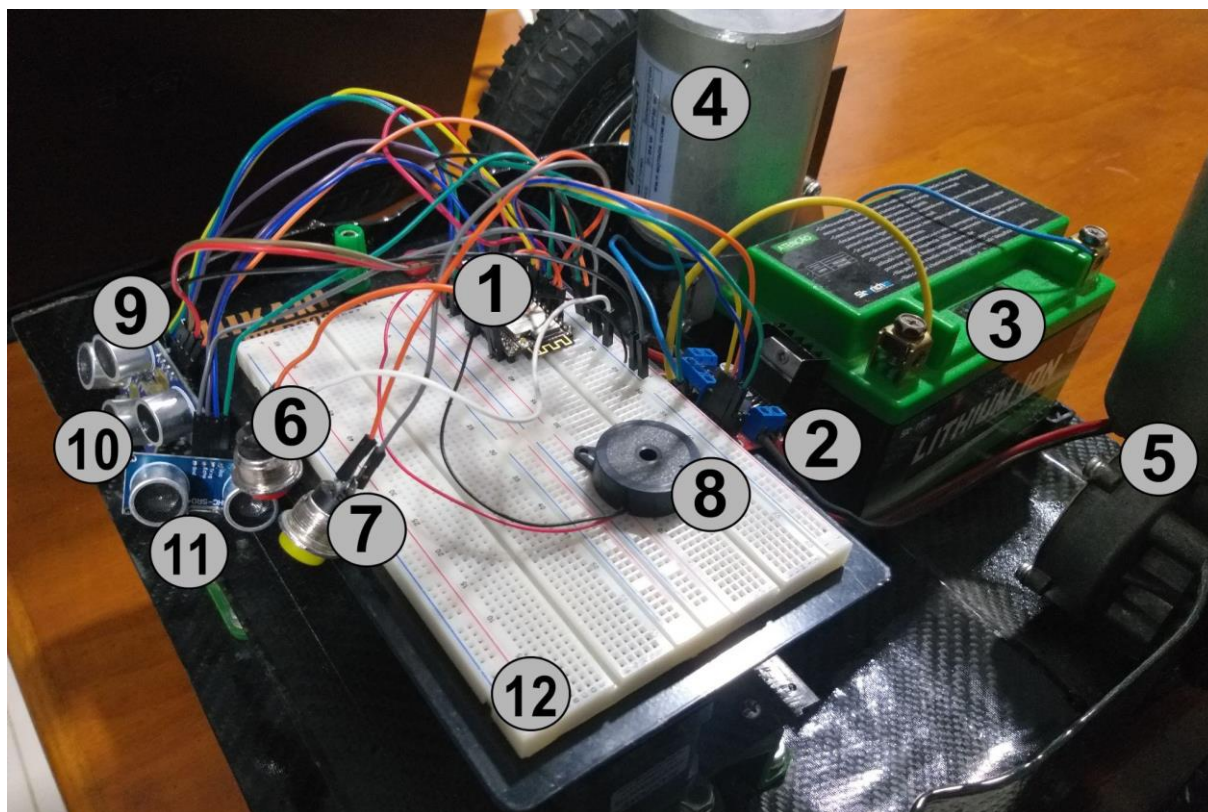
Para a realização da montagem do protótipo, o circuito eletrônico da figura 21 foi montado, e todos os componentes conectados, um a um, nas portas da placa, observam-se nas Figuras 25 e 26, na Figura 25 o foco está no ESP32 e na ponte H para melhor visualização das ligações do circuito.

Figura 25 – Circuito do ESP32 e Ponte H



Fonte: Autores (2021).

Figura 26 – Circuito completo.



Fonte: Autores (2021).

Na Figura 26, observa-se os componentes utilizados no robô, são eles que tornam o funcionamento possível, sendo a placa ESP32 o item 1, recebendo conexão de todos os outros componentes diretamente, com exceção da bateria e dos motores. A ponte H como item 2, recebe a energia de 12v diretamente da bateria, item 3, e alimenta todo o circuito, também na ponte H estão conectados os itens 4 e 5, que são os motores de corrente contínua. Os responsáveis por acionar comandos são os botões, itens 6 e 7, interligados também à placa, para transmitir a informação quando forem pressionados, e então, o *buzzer*, item 8, e os sensores ultrassônicos, como itens 9, 10 e 11. Para auxiliar na fixação dos componentes, foi utilizada também uma *protoboard*, item 12.

4.3.1. TESTES DO FUNCIONAMENTO DO ROBÔ

O robô de telepresença foi testado para seguir sua rota pré-programada e realizar a verificação do circuito e programação, sendo assim, foi colocado em um local para simular a porta de uma clínica, Figura 27, e no momento em que um dos

botões for acionado, este, irá se locomover até o consultório desejado e retornar até o seu local de origem.

Figura 27 – Robô simulando posição inicial



Fonte: Autores (2021).

A partir do momento em que o robô inicia seu trajeto, caso haja algum obstáculo em seu caminho, seja um objeto ou uma pessoa, os sensores irão detectar e o robô irá parar e acionará o sinal sonoro até o momento em que não haja mais obstáculos, e assim seguirá seu caminho. Na Figura 28, foi colocado um obstáculo na frente do robô por alguns segundos e então foi removido, obtendo sucesso no resultado, pois o robô parou, acionou o sinal sonoro e quando retirado o obstáculo, retornou a sua rota.

Figura 28 – Obstáculo adicionado para testar sensores



Fonte: Autores (2021).

4.4. PROGRAMAÇÃO

A programação do robô de telepresença foi efetuada na IDE do Arduino e inserida na placa ESP32 através do cabo USB. Este código permite que o robô seja acionado quando um dos botões for pressionado, guiando o usuário até o local selecionado. Caso seja barrado por algum obstáculo, o robô irá parar e acionar o sinal sonoro até que o caminho esteja livre para continuar seu trajeto. Quando o robô chegar, emitirá também um sinal sonoro para avisar a chegada e então retornar ao posto inicial.

O código inicial (Figura 29) adiciona as bibliotecas necessárias para o funcionamento do sensor ultrassônico e do *buzzer*.

Figura 29 – Inclusão das bibliotecas na programação

```
#include <Ultrasonic.h>  
#include <Buzzer.h>
```

Fonte: Autores (2021).

Define-se as portas e variáveis utilizadas para o controle dos motores, conforme Figura 30.

Figura 30 – Definição das portas e variáveis dos motores.

```
// Motor Direita
int motor1a = 27;
int motor1b = 26;

//Motor Esquerdo
int motor2a = 25;
int motor2b = 33;
```

Fonte: Autores (2021).

São definidas também as portas utilizadas nos botões, sendo elas constantes e inteiras, de acordo com a Figura 31.

Figura 31 – Definição das portas dos botões.

```
//Botões
const int ButtonA = 34;
const int ButtonB = 35;
```

Fonte: Autores (2021).

Para identificar se algum dos botões foi pressionado, uma variável para cada um é criada para receber esta informação, como na Figura 32.

Figura 32 – Variável de estado dos botões.

```
int StateA = 0;
int StateB = 0;
```

Fonte: Autores (2021).

Com o intuito de obter a distância a qual os sensores estão recebendo, uma variável é criada para salvar essa informação, de acordo com a Figura 33.

Figura 33 – Variável de distância dos sensores.

```
int distance = 0;
```

Fonte: Autores (2021).

Seguindo a Figura 34, também define-se as portas para cada sensor, sendo que cada um possui uma porta *Trigger* e uma *Echo*.

Figura 34 – Definição das portas dos sensores.

```
int ultrasonic1A = 2;
int ultrasonic1B = 4;
int ultrasonic2A = 23;
int ultrasonic2B = 22;
int ultrasonic3A = 19;
int ultrasonic3B = 18;
```

Fonte: Autores (2021).

O *buzzer* é utilizado para emitir um sinal sonoro quando solicitado, então definimos a porta deste, conforme Figura 35.

Figura 35 – Definindo o pino do *buzzer*.

```
const int pinoBuzzer = 15;
```

Fonte: Autores (2021).

O objetivo dos sensores ultrassônicos é emitir uma onda sonora de alta frequência e medir o tempo para a recepção do eco produzido no momento em que a onda se choca em um objeto, neste comando, os sensores são iniciados, como na Figura 36.

Figura 36 – Programação que inicia os sensores

```
Ultrasonic ultrasonic(ultrasonic1A, ultrasonic1B);
Ultrasonic ultrasonic(ultrasonic2A, ultrasonic2B);
Ultrasonic ultrasonic(ultrasonic3A, ultrasonic3B);
```

Fonte: Autores (2021).

A função *void setup* é executada quando o programa se inicia, é nela em que as entradas e saídas são configuradas de acordo com a função de cada variável definida anteriormente, sendo feitas pelo comando *pinMode*, segundo a Figura 37.

Figura 37 – Define os pinos como entrada ou saída.

```
void setup() {
  // seta os pinos como entrada ou saída
  pinMode(motor1a, OUTPUT);
  pinMode(motor1b, OUTPUT);
  pinMode(motor2a, OUTPUT);
  pinMode(motor2b, OUTPUT);
  pinMode(ButtonA, INPUT);
  pinMode(ButtonB, INPUT);
  pinMode(ultrasonic1A , INPUT);
  pinMode(ultrasonic1B , OUTPUT);
  pinMode(ultrasonic2A , INPUT);
  pinMode(ultrasonic2B , OUTPUT);
  pinMode(ultrasonic3A , INPUT);
  pinMode(ultrasonic3B , OUTPUT);
}
```

Fonte: Autores (2021).

Os comandos inseridos dentro da função *void loop* são executados infinitamente, ou seja, não há um tempo determinado para finalizar sua execução, deixando a programação sempre ativa na espera de um chamado, podendo verificar na Figura 38.

Figura 38 – Inicia a função de repetição.

```
void loop() {
```

Fonte: Autores (2021).

As variáveis de estado de cada botão se igualam ao valor lido, ou seja, a variável *StateA* é iniciada como 0, pois o botão não foi acionado, e caso seja pressionado, a leitura digital *digitalRead* encaminhará a informação e passará a ser 1, conforme mostrado na Figura 39.

Figura 39 – Variável para ler o estado dos botões.

```
StateA = digitalRead(ButtonA);
StateB = digitalRead(ButtonB);
```

Fonte: Autores (2021).

Uma função é uma sub-rotina que pode ser chamada durante a execução do programa, a função `sensores` é acionada para realizar as medições pelos sensores, segundo a Figura 40.

Figura 40 – Função `sensores` sendo chamada.

```
sensores ();
```

Fonte: Autores (2021).

O comando *if* altera o fluxo de execução do programa, essa função é realizada através de uma expressão lógica, sendo verdadeiro ou falso. No caso a seguir, a variável de estado do botão A é analisada para verificar se o botão foi acionado ou não, caso tenha sido e o estado da variável seja *High*, irá executar um comando, caso contrário executará outro, conforme Figura 41.

Figura 41 – Inicia a função lógica para o estado do botão.

```
if (StateA == HIGH) {
```

Fonte: Autores (2021).

Outro comando, visto na figura 42, de condição *if* é aplicado também para a verificação das medidas dos sensores, se a distância for maior que 30cm, será executado um código específico, caso contrário o *buzzer* será acionado e os motores irão parar.

Figura 42 – Inicia a função lógica com as informações dos sensores.

```
if(distance => 30) {
```

Fonte: Autores (2021).

Na Figura 43, define-se a ação de silenciar o *buzzer*.

Figura 43 - Silenciar o *buzzer*.

```
noTone (pinoBuzzer) ;
```

Fonte: Autores (2021).

A linha de código a seguir chama a função que irá ser executada quando o botão vermelho for pressionado, segundo a Figura 44.

Figura 44 – Chama a função do botão vermelho.

```
botaoVermelho ();
```

Fonte: Autores (2021).

Caso a distância do robô seja menor que 30cm, um sinal sonoro será emitido pelo *buzzer* durante 1,5 segundos e o robô irá parar até não haver mais obstáculos, como na Figura 45.

Figura 45 – Emissão do sinal sonoro e parada dos motores.

```
    }else{
    tone(pinoBuzzer,1500);
        digitalWrite(motor1a, LOW);
        digitalWrite(motor1b, LOW);
        digitalWrite(motor2a, LOW);
        digitalWrite(motor2b, LOW);
    } else {
        digitalWrite(motor1a, LOW);
        digitalWrite(motor1b, LOW);
        digitalWrite(motor2a, LOW);
        digitalWrite(motor2b, LOW);
    }
}
```

Fonte: Autores (2021).

O mesmo comando é realizado para o botão B, que fará a segunda rota, segundo a Figura 46.

Figura 46 – Programação do botão B.

```

if (StateB == HIGH) {
if(distance => 30){
noTone(pinoBuzzer);

botaoAmarelo();
}else{
tone(pinoBuzzer,1500);
digitalWrite(motor1a, LOW);
digitalWrite(motor1b, LOW);
digitalWrite(motor2a, LOW);
digitalWrite(motor2b, LOW);
} else {
digitalWrite(motor1a, LOW);
digitalWrite(motor1b, LOW);
digitalWrite(motor2a, LOW);
digitalWrite(motor2b, LOW);
}
}

```

Fonte: Autores (2021).

A função “sensores” realiza os comandos necessários para a verificação da distância de cada sensor ultrassônico, conforme Figura 47.

Figura 47 – Comando que cria a função sensores.

```

void sensores() {

```

Fonte: Autores (2021).

O código a seguir define os pinos *Trigger* de cada sensor como baixo (0), conforme mostrado na Figura 48.

Figura 48 – Definição dos pinos *Trigger* dos sensores.

```

digitalWrite(ultrasonic1A, LOW);
digitalWrite(ultrasonic2A, LOW);
digitalWrite(ultrasonic3A, LOW);

```

Fonte: Autores (2021).

Na Figura 49, adiciona-se uma pausa de 2 microssegundos.

Figura 49 – Comando que realiza uma pausa.

```
delayMicroseconds (2) ;
```

Fonte: Autores (2021).

Logo após, de acordo com a figura 50, os mesmos pinos são definidos como alto (1).

Figura 50 – Definição dos pinos do sensor como alto.

```
digitalWrite (ultrasonic1A, HIGH) ;
digitalWrite (ultrasonic2A, HIGH) ;
digitalWrite (ultrasonic3A, HIGH) ;
```

Fonte: Autores (2021).

E para finalizar a análise da distância, outra pausa é realizada e então os pinos são definidos como baixos novamente, como na Figura 51.

Figura 51 – Pinos dos sensores definidos como baixo.

```
delayMicroseconds (10) ;
digitalWrite (ultrasonic1A, LOW) ;
digitalWrite (ultrasonic2A, LOW) ;
digitalWrite (ultrasonic3A, LOW) ;
```

Fonte: Autores (2021).

Após a ativação dos sensores para obter as medidas, é chamada a função *ranging*, mostrada na Figura 52, para fazer a conversão do tempo de resposta em centímetros e armazenar na variável *distance*.

Figura 52 – Comando que obtém a distância.

```
distance = (ultrasonic.Ranging (CM)) ;
delay (500) ;
}
```

Fonte: Autores (2021).

A função já chamada anteriormente, `botaoVermelho`, quando executada, faz com que o robô siga o caminho pré-definido, conforme detalhado e de acordo com a Figura 53.

Figura 53 – Comando que cria a função do botão vermelho.

```
void botaoVermelho () {
```

Fonte: Autores (2021).

Possuindo uma pausa inicial de 2 segundos após o botão ser pressionado, para isso os motores estão em estado 0 (*low*), como na Figura 54.

Figura 54 – Comando para parar os motores durante 2 segundos.

```
digitalWrite (motor1a, LOW);
digitalWrite (motor1b, LOW);
digitalWrite (motor2a, LOW);
digitalWrite (motor2b, LOW);
delay (2000);
```

Fonte: Autores (2021).

O robô seguirá em frente durante 7 segundos até chegar em seu primeiro destino, tornando as variáveis `motor1b` e `motor2a` como alta (*high*), assim os motores irão girar para frente, seguindo a estrutura da Figura 55.

Figura 55 – Código para girar os motores para frente.

```
digitalWrite (motor1a, LOW);
digitalWrite (motor1b, HIGH);
digitalWrite (motor2a, HIGH);
digitalWrite (motor2b, LOW);
delay (7000);
```

Fonte: Autores (2021).

No momento em que o robô chegar em seu primeiro destino, os motores irão parar por 5 segundos e o *buzzer* emitirá um sinal sonoro de 1 segundo para informar que sua rota foi concluída, conforme mostrado na Figura 56.

Figura 56 – Comando para parar motores e ativar o sinal sonoro.

```
digitalWrite(motor1a, LOW);
digitalWrite(motor1b, LOW);
digitalWrite(motor2a, LOW);
digitalWrite(motor2b, LOW);
tone(pinoBuzzer, 1000);
delay(5000);
```

Fonte: Autores (2021).

Para retornar ao posto inicial, é realizada a volta em torno do seu próprio eixo e seguirá em frente, apresentado no código da Figura 57.

Figura 57 – Programação para retornar ao posto inicial.

```
//Faz a volta
digitalWrite(motor1a, LOW);
digitalWrite(motor1b, LOW);
digitalWrite(motor2a, HIGH);
digitalWrite(motor2b, LOW);
delay(3500);

//Frente
digitalWrite(motor1a, LOW);
digitalWrite(motor1b, HIGH);
digitalWrite(motor2a, HIGH);
digitalWrite(motor2b, LOW);
delay(7500);
```

Fonte: Autores (2021).

Com o intuito de se estabelecer na mesma posição em que saiu, o robô é programado para fazer a volta e parar, e assim aguardar um novo comando, como na Figura 58.

Figura 58 – Robô retorna à posição inicial.

```
//Posição inicial
digitalWrite(motor1a, LOW);
digitalWrite(motor1b, LOW);
digitalWrite(motor2a, HIGH);
digitalWrite(motor2b, LOW);
delay(3500);

//Parar
digitalWrite(motor1a, LOW);
digitalWrite(motor1b, LOW);
digitalWrite(motor2a, LOW);
digitalWrite(motor2b, LOW);
}
```

Fonte: Autores (2021).

Cada botão possui uma função para ser chamada e um código específico a ser executado, o segundo botão possui uma rota diferente, mas com códigos similares, segundo a Figura 59.

Figura 59 – Função com a rota do botão B.

```
//Local 2
digitalWrite (motor1a, LOW);
digitalWrite (motor1b, LOW);
digitalWrite (motor2a, LOW);
digitalWrite (motor2b, LOW);
delay (2000);

//Faz a volta
digitalWrite (motor1a, LOW);
digitalWrite (motor1b, LOW);
digitalWrite (motor2a, HIGH);
digitalWrite (motor2b, LOW);
delay (3500);

//Frente
digitalWrite (motor1a, LOW);
digitalWrite (motor1b, HIGH);
digitalWrite (motor2a, HIGH);
digitalWrite (motor2b, LOW);
delay (3000);

//Posição inicial
digitalWrite (motor1a, LOW);
digitalWrite (motor1b, LOW);
digitalWrite (motor2a, HIGH);
digitalWrite (motor2b, LOW);
delay (1750);

//Parar
digitalWrite (motor1a, LOW);
digitalWrite (motor1b, LOW);
digitalWrite (motor2a, LOW);
digitalWrite (motor2b, LOW);
}
```

CONCLUSÃO

No decorrer do desenvolvimento deste projeto, o intuito foi a implementação de um circuito eletrônico, tendo como componente principal de estudo o ESP32, e programação para um robô de telepresença com atuação em áreas hospitalares, com a finalidade de minimizar o contágio de doenças transmitidas por secreções contaminadas.

O ESP32 mostrou-se muito eficaz no objetivo proposto, juntamente com os demais componentes. Confirmou-se as informações passadas pelo fabricante, visto que ele apresentou um amplo poder de processamento, memória e conectividade.

Os procedimentos técnicos, implementação do circuito e programação, foram concluídos. Apresentou-se os principais componentes utilizados, como também a descrição das etapas que foram seguidas para o desenvolvimento e implementação do projeto.

De forma geral os objetivos citados foram alcançados, ele é capaz de se locomover, fazer curvas e detectar obstáculos, com sua finalização, fica como melhoria a substituição da *proto-board* por uma placa impressa, para minimizar riscos de mal contato dos fios, e a implementação do suporte para o *tablet*.

Melhorias essas que ficam como sugestões para trabalhos futuros, além da inclusão do *tablet* e implementação do aplicativo que será responsável pela interação do paciente com o robô, com todos os comandos essenciais para suprir as necessidades das clínicas e hospitais.

REFERÊNCIAS

AGUILAR, Luis Joyanes. **Fundamentos de programação. Algoritmos, estrutura de dados e objetos**. 3. ed. *E-book*. São Paulo: AMGH, 2011. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788521207733/cfi/4!/4/4@0.00:24.2>. Acesso em: 19 jun. 2021.

AGUILAR, Luis Joyanes. **Programação em C++: algoritmos, estruturas de dados e objetos**. 2. ed. *E-book*. Porto Alegre: AMGH, 2011. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788580550269/cfi/2!/4/4@0.00:40.2>. Acesso em: 19 jun. 2021.

AGUIRRE, Luis Antonio. **Enciclopédia de automática: controle e automação, volume III**. 1. ed. *E-book*. São Paulo: Blucher, 2007. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788521207733/cfi/4!/4/4@0.00:24.2>. Acesso em: 19 jun. 2021.

ARDUINO. Disponível em: <http://www.arduino.cc>. Acesso em: 19 jun. 2021.

BOLTON, William. **Mecatrônica: uma abordagem multidisciplinar**. 4. ed. Porto Alegre: Bookman, 2010.

CHEHUEN NETO, et al. **Exposição ocupacional a material biológico na área da saúde**. Minas Gerais: Rev Med, 2006. Disponível em: <http://rmmg.org/artigo/detalhes/1388>. Acesso em: 08 jul. 2021.

CYTRON TECHNOLOGIES. **HC-SR04 User's Manual**. 2013. Disponível em: <http://web.eece.maine.edu/~zhu/book/lab/HC-SR04%20User%20Manual.pdf>. Acesso em: 19 jun. 2021.

DAMAS, Luis. **Linguagem C** 10. ed. *E-book*. Rio de Janeiro: LTC, 2007. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788521207733/cfi/4!/4/4@0.00:24.2>. Acesso em: 19 jun. 2021.

ELETROGATE. **Guia definitivo de uso da Ponte H L298N**. Disponível em: <https://blog.eletrogate.com/guia-definitivo-de-uso-da-ponte-h-l298n>. Acesso em: 1 jun. 2020.

ESPRESSIF. Disponível em: <https://www.espressif.com/>. Acesso em: 19 jun. 2021.

GIL, Antonio Carlos. **Métodos e Técnicas de Pesquisa Social**. 11. ed. São Paulo: Atlas S. A., 2008.

GRASSI, Valdir Junior. **Arquitetura híbrida para robôs móveis baseada em funções de navegação com interação humana**. Tese de Ph.D., São Paulo: Escola Politécnica da Universidade de São Paulo, 2006. Disponível em: <https://teses.usp.br/teses/disponiveis/3/3132/tde-19092006145159/publico/TeseValdir2006.pdf>. Acesso em: 19 jun. 2021.

HONYTEK. **SMT9650 9.6*5.0 Piezo Buzzer Push Buzzer 12V Piezo**. Disponível em: <https://www.honytek.net/showroom/smt9650-9-6-5-0-piezo-buzzer-push-buzzer-12v-piezo.html>. Acesso em: 14 jun. 2021.

IBRAHIM, Dogan. **The Complete ESP32 Projects Guide**. 1. ed. Elektor International Media BV, 2019.

ISO. **ISO 8373. Standard 8373 - Robots and robotic devices - Vocabulary**. 2012.
KOLBAN, Neil. **Kolban's Book on ESP32**. Leanpub, 2017.

LENZ, M. L.; TORRES F. E. **Microprocessadores**. 1. ed. *E-book*. Porto Alegre: SAGAH, 2019. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788595029736/cfi/1!/4/4@0.00:26.4>. Acesso em: 19 jun. 2021.

MATARIC, Maja J. **Introdução à Robótica**. 1. ed. São Paulo: Unesp/Blucher, 2014.

MONK, S. **Programação com Arduino: começando com sketches**. 2. ed. Porto Alegre: Bookman, 2017.

MUNARI, Bruno. **Das coisas nascem coisas**. Tradução de José Manuel de Vasconcelos. Lisboa: Edições 70, 1981.

NORVIG, P; RUSSELL, S. **Inteligência Artificial**. 3. ed. *E-book*. Rio de Janeiro: Elsevier, 2013. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788595156104/cfi/6/8!/4/2/2/34/2@0:30.8>. Acesso em: 19 jun. 2021.

PMI. **Um Guia do Conhecimento em Gerenciamento de Projetos - Guia PMBOK 5ª Edição**. EUA: Project Management Institute, 2013.

PRODANOV E FREITAS. **Motodologia do trabalho científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico** [recurso eletônico]. Disponível em: <https://www.feevale.br/Comum/midias/0163c988-1f5d-496f-b118-a6e009a7a2f9/>. *E-book Metodologia do Trabalho Científico.pdf*. Acesso em: 18 junho. 2021.

PUHL, Flávio Luiz Junior, et al. **Robótica**. 1. ed. *E-book*. Porto Alegre: SAGAH, 2019. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788595029125/cfi/1!/4/4@0.00:28.8>. Acesso em: 19 jun. 2021.

ROMERO, Roseli Aparecida Francelin (Org.), et al. **Robótica Móvel**. 1. ed. *E-book*. Rio de Janeiro: LTC, 2017. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/978-85-216-2642-8/cfi/6/10!/4/2@0:0>. Acesso em: 19 jun. 2021.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to autonomous mobile robots**. 2. ed. Massachusetts: The MIT Press, 2011.

SIEMENS. **Motores de Corrente Contínua: Guia Rápido Para Uma Especificação Precisa**. 1. ed. *E-book*, 2006. Disponível em: http://www.marioloureiro.net/tecnica/electrif/Motores_CC_ind1.pdf. Acesso em: 19 jun. 2021.

SOUZA, Marco Antonio Furlan de, et al. **Algoritmos e lógica de programação: um texto introdutório para a engenharia**. 1. ed. *E-book*. São Paulo: Cengage, 2019. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788522128150/cfi/1!/4/4@0.00:25.6>. Acesso em: 19 jun. 2021.

STEVAN JR., S. L.; SILVA, R. A. **Automação e Instrumentação Industrial com Arduino: teoria e projetos**. 1. ed. *E-book*. São Paulo: Érica/Saraiva, 2015. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788536518152/cfi/2!/4/4@0.00:0.0>. Acesso em: 19 jun. 2021.

APÊNDICE A – PROGRAMA UTILIZADO

```
#include <Ultrasonic.h>
#include <Buzzer.h>

// Motor Direita
int motor1a = 27;
int motor1b = 26;

//Motor Esquerdo
int motor2a = 25;
int motor2b = 33;

//Botões
const int ButtonA = 34;
const int ButtonB = 35;
int StateA = 0;
int StateB = 0;

//Sensor ultrassônico
int distance = 0;
int ultrasonic1A = 2;
int ultrasonic1B = 4;
int ultrasonic2A = 23;
int ultrasonic2B = 22;
int ultrasonic3A = 19;
int ultrasonic3B = 18;

//Buzzer
const int pinoBuzzer = 15;

//Inicializa os sensores nos pinos definidos
Ultrasonic ultrasonic(ultrasonic1A, ultrasonic1B);
Ultrasonic ultrasonic(ultrasonic2A, ultrasonic2B);
```



```
Ultrasonic ultrasonic(ultrasonic3A, ultrasonic3B);
```

```
void setup() {  
  // seta os pinos como entrada ou saída  
  pinMode(motor1a, OUTPUT);  
  pinMode(motor1b, OUTPUT);  
  pinMode(motor2a, OUTPUT);  
  pinMode(motor2b, OUTPUT);  
  pinMode(ButtonA, INPUT);  
  pinMode(ButtonB, INPUT);  
  pinMode(ultrasonic1A , INPUT);  
  pinMode(ultrasonic1B , OUTPUT);  
  pinMode(ultrasonic2A , INPUT);  
  pinMode(ultrasonic2B , OUTPUT);  
  pinMode(ultrasonic3A , INPUT);  
  pinMode(ultrasonic3B , OUTPUT);  
}
```

```
void loop() {  
  StateA = digitalRead(ButtonA);  
  StateB = digitalRead(ButtonB);  
  sensores();
```

```
  if (StateA == HIGH) {  
    if(distance ==> 30){  
      noTone(pinoBuzzer);
```

```
      botaoVermelho();  
    }else{  
      tone(pinoBuzzer,1500);  
      digitalWrite(motor1a, LOW);  
      digitalWrite(motor1b, LOW);  
      digitalWrite(motor2a, LOW);  
      digitalWrite(motor2b, LOW);
```

```

    } else {
        digitalWrite(motor1a, LOW);
        digitalWrite(motor1b, LOW);
        digitalWrite(motor2a, LOW);
        digitalWrite(motor2b, LOW);
    }
}
if (StateB == HIGH) {
    if(distance ==> 30){
        noTone(pinoBuzzer);

        botaoAmarelo();
    }else{
        tone(pinoBuzzer,1500);
        digitalWrite(motor1a, LOW);
        digitalWrite(motor1b, LOW);
        digitalWrite(motor2a, LOW);
        digitalWrite(motor2b, LOW);
    } else {
        digitalWrite(motor1a, LOW);
        digitalWrite(motor1b, LOW);
        digitalWrite(motor2a, LOW);
        digitalWrite(motor2b, LOW);
    }
}
void sensores(){
    digitalWrite(ultrasonic1A, LOW);
    digitalWrite(ultrasonic2A, LOW);
    digitalWrite(ultrasonic3A, LOW);
    delayMicroseconds(2);
    digitalWrite(ultrasonic1A, HIGH);
    digitalWrite(ultrasonic2A, HIGH);
    digitalWrite(ultrasonic3A, HIGH);
    delayMicroseconds(10);
}

```

```
digitalWrite(ultrasonic1A, LOW);  
digitalWrite(ultrasonic2A, LOW);  
digitalWrite(ultrasonic3A, LOW);  
distance = (ultrasonic.Ranging(CM));  
delay(500);  
}
```

```
void botaoVermelho(){  
  //Delay inicial  
  digitalWrite(motor1a, LOW);  
  digitalWrite(motor1b, LOW);  
  digitalWrite(motor2a, LOW);  
  digitalWrite(motor2b, LOW);  
  delay(2000);
```

```
  //Frente  
  digitalWrite(motor1a, LOW);  
  digitalWrite(motor1b, HIGH);  
  digitalWrite(motor2a, HIGH);  
  digitalWrite(motor2b, LOW);  
  delay(7000);
```

```
  //Local 1  
  digitalWrite(motor1a, LOW);  
  digitalWrite(motor1b, LOW);  
  digitalWrite(motor2a, LOW);  
  digitalWrite(motor2b, LOW);  
  tone(pinoBuzzer,1000);  
  delay(5000);
```

```
  //Faz a volta  
  digitalWrite(motor1a, LOW);  
  digitalWrite(motor1b, LOW);  
  digitalWrite(motor2a, HIGH);
```

```
digitalWrite(motor2b, LOW);  
delay(3500);
```

```
//Frente
```

```
digitalWrite(motor1a, LOW);  
digitalWrite(motor1b, HIGH);  
digitalWrite(motor2a, HIGH);  
digitalWrite(motor2b, LOW);  
delay(7500);
```

```
//Posição inicial
```

```
digitalWrite(motor1a, LOW);  
digitalWrite(motor1b, LOW);  
digitalWrite(motor2a, HIGH);  
digitalWrite(motor2b, LOW);  
delay(3500);
```

```
//Parar
```

```
digitalWrite(motor1a, LOW);  
digitalWrite(motor1b, LOW);  
digitalWrite(motor2a, LOW);  
digitalWrite(motor2b, LOW);  
}
```

```
void botaoAmarelo(){
```

```
  //Delay inicial
```

```
  digitalWrite(motor1a, LOW);  
  digitalWrite(motor1b, LOW);  
  digitalWrite(motor2a, LOW);  
  digitalWrite(motor2b, LOW);  
  delay(2000);
```

```
  //Faz a volta
```

```
  digitalWrite(motor1a, LOW);
```

```
digitalWrite(motor1b, LOW);  
digitalWrite(motor2a, HIGH);  
digitalWrite(motor2b, LOW);  
delay(1750);
```

```
//Frente
```

```
digitalWrite(motor1a, LOW);  
digitalWrite(motor1b, HIGH);  
digitalWrite(motor2a, HIGH);  
digitalWrite(motor2b, LOW);  
delay(3000);
```

```
//Local 2
```

```
digitalWrite(motor1a, LOW);  
digitalWrite(motor1b, LOW);  
digitalWrite(motor2a, LOW);  
digitalWrite(motor2b, LOW);  
delay(2000);
```

```
//Faz a volta
```

```
digitalWrite(motor1a, LOW);  
digitalWrite(motor1b, LOW);  
digitalWrite(motor2a, HIGH);  
digitalWrite(motor2b, LOW);  
delay(3500);
```

```
//Frente
```

```
digitalWrite(motor1a, LOW);  
digitalWrite(motor1b, HIGH);  
digitalWrite(motor2a, HIGH);  
digitalWrite(motor2b, LOW);  
delay(3000);
```

```
//Posição inicial
```

```
digitalWrite(motor1a, LOW);  
digitalWrite(motor1b, LOW);  
digitalWrite(motor2a, HIGH);  
digitalWrite(motor2b, LOW);  
delay(1750);
```

```
//Parar  
digitalWrite(motor1a, LOW);  
digitalWrite(motor1b, LOW);  
digitalWrite(motor2a, LOW);  
digitalWrite(motor2b, LOW);  
}
```