Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

IMPLEMENTAÇÃO DE UM ROBÔ USANDO O MICROCONTROLADOR ESP32

LUAN JUNIOR CHAVES

ORIENTADOR: PROF. DR. RUBENS BARBOSA FILHO

Dourados - MS
2022

IMPLEMENTAÇÃO DE UM ROBÔ USANDO O MICROCONTROLADOR ESP32

LUAN JUNIOR CHAVES

Este exemplar corresponde ao trabalho de conclusão de curso da disciplina Projeto Final de Curso, devidamente corrigida e defendida por Luan Junior Chaves e aprovada pela banca examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

ORIENTADOR: PROF. DR. RUBENS BARBOSA FILHO

DOURADOS - MS
202

C439i Chaves, Luan Junior
Implementação de um robô usando microcontrolador ESP32/ Luan Junior Chaves - Dourados, MS: UEMS, 2022.
43 p.

Monografia (graduação) - Ciência da Computação - Universidade estadual de Mato Grosso do Sul, 2022
Orientador: Prof. Dr. Rubens Barbosa Filho

1. Microcontrolador 2. Robótica 3. HTTP. I Barbosa Filho, Rubens II, Título.

CDD 23. ed. - 629.829

Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

IMPLEMENTAÇÃO DE UM ROBÔ USANDO O MICROCONTROLADOR ESP32

LUAN JUNIOR CHAVES
Novembro de 2022

Banca Examinadora:

Prof. Dr. Rubens Barbosa Filho (Orientador)
Área de computação - UEMS

Prof. MSc. André Chastel Lima
Área de computação - UEMS

MSc. Alfred Forster Junior
Área da computação - UEMS

Esse trabalho é dedicado aos meus pais, em especial ao meu pai que sempre sonhou em ver seu filho formado em uma universidade.

AGRADECIMENTOS

Agradeço primeiramente a minha família, pois sem o apoio deles não estaria entrando nesta reta final da conclusão da graduação.

Agradeço também ao meu orientador que me auxiliou durante o desenvolvimento deste trabalho, não só com assuntos técnicos, mas também conselhos que levarei para o resto da vida.

Agradeço à Professora Adriana Betânia de Paula Molgora por me proporcionar minha primeira iniciação científica e aos professores Nilton Cézar de Paula e Dalton Pedrozo de Queiroz por me proporcionarem experiências com os projetos de extensão.

RESUMO

Devido ao crescente aumento de soluções robóticas para os mais diversos tipos de problemas, torna-se por muitas vezes o estudo de tal tecnologia complexo e caro demais para estudantes conseguirem desenvolver essas soluções. Com esta problemática, empresas desenvolveram microcontroladores de baixo custo e fácil acesso para estudantes e pequenas empresas desenvolverem suas soluções. Este trabalho visa criar um pequeno robô utilizando o microcontrolador ESP32 com requisições HTTP (*Hyper Text Transfer Protocol*) a fim de explorar a capacidade deste tipo de dispositivo, mantendo um baixo custo e que possa ser facilmente escalado para outros problemas da área.

Palavras-chave: Microcontrolador, robótica, HTTP.

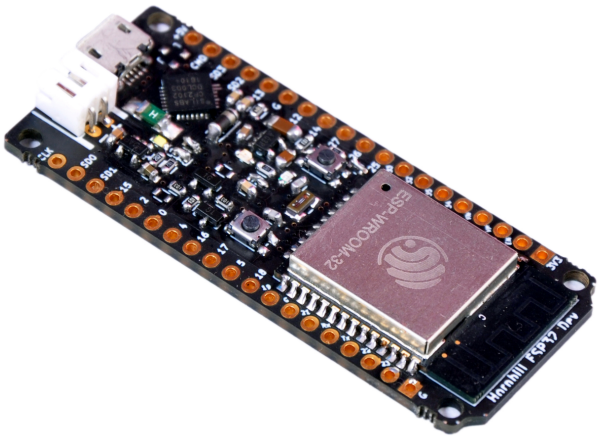
SUMÁRIO

LISTA DE ILUSTRAÇÕES

1 - Introdução.

Atualmente, temos milhares de soluções robóticas do mais alto nível para diversas aplicações, entretanto muitas vezes essas soluções tão avançadas acabam por se tornar inviáveis e muito caras para estudantes conseguirem compreender e aprender com tais tecnologias. Com isso, soluções vêm sendo criadas por empresas com o propósito de auxiliar a criação de protótipos de baixo custo e acessível para pequenas startups[1] e estudantes.

Figura 1 - Imagem demonstrativa de um ESP32



Fonte: *Explore Embedded*[2]

Microcontroladores, como o mostrado acima, permite que possam ser gerados protótipos de baixo custo para as mais diversas áreas, desde robótica até a automação industrial/residencial.

[1] O que é uma startup e como montar a sua, Disponível em <https://meunegocio.uol.com.br/blog/o-que-e-uma-startup-e-como-montar-a-sua/?gclid=Cj0KCQjwnP-ZBhDiARIsAH3FSRdSv-pYyso1fNh74Ti3UHjb15qfW0MgwfPbJMkonT8uKGd3YJFqzicaAtPyEALw_wcB> acessado em 07 de outubro de 2022

[2] Disponivel em: <https://www.exploreembedded.com/wiki/Overview_of_ESP32_features._What_do_they_practically_mean%3F>. Acesso em 07 de Outubro de 2022

1.1 - Objetivos.

O desenvolvimento deste trabalho tem como objetivo a criação de um pequeno robô de duas rodas que possa ser controlado através de comandos remotos utilizando a internet, se utilizando de peças de baixo custo e com programação acessível, desmistificando assim a complexidade criada em cima da programação de microcontroladores.

1.1.1 - Objetivos Específicos

Os objetivos deste trabalho são:

- Estudo e revisão bibliográfica de protótipos utilizando microcontroladores;
- Desenvolvimento de uma interface de comunicação com o robô que permite enviar comandos;
- Desenvolvimento de um sistema de monitoramento da carga da bateria;
- Desenvolvimento de um sistema de segurança que restringe o acesso de pessoas não autorizadas.

1.2 - Justificativa

O desenvolvimento desse projeto tem como objetivo a fomentação do conhecimento desses microcontroladores como um instrumento de estudo e criação de novas tecnologias que sejam acessíveis para toda a comunidade.

Soluções comerciais de automação, tanto residenciais como industriais, e até mesmo pequenos robôs tendem a ser muito caras, principalmente em países subdesenvolvidos como o Brasil, logo, o principal objetivo deste trabalho é despertar a curiosidade e mostrar a facilidade comercial da criação desses protótipos e assim, possivelmente, despertar o interesse na criação de novas soluções.

1.3 - Metodologia

Para a realização deste trabalho de forma teórica, foram realizadas buscas por trabalhos semelhantes na área acadêmica como artigos, livros e dissertações.

Para a realização deste trabalho de forma prática (mecânica) foi realizada a compra das peças em diversas plataformas de compras na internet. E para a

programação e injeção do código no microcontrolador foi utilizada a plataforma Arduino IDE 2.0

1.4 - Organização do Texto

O capítulo 2 apresenta o referencial teórico e a apresentação descritiva dos microcontroladores e componentes utilizados.

O capítulo 3 apresenta a montagem do robô e o desenvolvimento do código.

O capítulo 4 apresenta a versão final do robô e seu funcionamento prático.

O capítulo 5 apresenta as conclusões deste trabalho.

2 - Definições da problemática.

Para estudar as capacidades do ESP32 foi escolhido um protótipo de robô, pois um robô que seja controlado remotamente precisa de elementos como: Um grau de confiabilidade consideravelmente alto para ser controlado por qualquer sinal escolhido, realizar tarefas aproveitando da melhor forma possível a bateria disponível e capacidade, tanto em memória como em processamento para realizar as tarefas atribuídas.

2.1 - Definições de componentes.

Robôs possuem várias partes, desde sua estrutura física que podem ser feitas de polímeros quando exigem trabalhos mais delicados ou que não necessitam de tanta resistência física ou de metais quando exigem trabalhos mais pesados e ou resistência contra impactos, até seus sensores, que serão “os olhos e ouvidos” do robô, que serão necessários para enviar informações para todo o sistema elétrico para transformar informações em ações físicas.

Há também o sistema de alimentação, que é de vital importância para o funcionamento do robô. Um bom controle da energia que é consumida para o funcionamento preciso é de grande importância, visto que quanto menos energia gastar mais tempo de funcionamento sem interrupções terá. Este controle pode ser feito tanto na escolha dos componentes utilizados, quanto na própria programação, como veremos mais adiante.

2.2 - Microcontroladores e o Esp32

Muitas vezes confundido com microprocessador, um microcontrolador consiste em apenas um único circuito integrado, que reúne apenas um núcleo de processador juntamente de memórias voláteis e alguns periféricos para realizar entrada e saída de dados.

Uma plataforma de desenvolvimento muito famosa é fabricada pela empresa Arduino[3] utilizando os microcontroladores Atmel (adquirida pela Microchip Technology em 2016). Com o sucesso do Arduino pelos entusiastas da automação,

[3] ARDUINO, Disponível em <https://www.arduino.cc/en/about> Acesso em 06 de outubro de 2022

diversas outras plataformas surgiram no mercado, como o Esp32 e o Esp8266, ambas produzidos pela empresa Espressif Systems[4]

O ESP32 é um microcontrolador que vem ganhando cada vez mais espaço entre os entusiastas da robótica e automação, visto seu poder de processamento mais alto, comparado ao arduino e maior capacidade de armazenamento, tanto em relação ao código quando a memória ram para processamento de informações.

Figura 2 - Comparação entre o ESP8266, ESP32 e Arduino UNO

	ESP8266	ESP32	Arduino UNO
Corrente	197mA	220mA	40mA
Núcleo	1	2	1
Arquitetura	32 bits	32 bits	8 bits
Clock	80 – 160 MHz	160-240 MHz	16MHz
Bluetooth	Não	Clássico e BLE (Bluetooth Low Energy)	Não
WiFi	Sim	Sim	Não
RAM	160KB	520KB	2KB
FLASH	16Mb	16Mb	32KB
GPIO	13	34	14
DAC	0	2	0
ADC	1	18	6
Interfaces	SPI, I2C, UART e I2S	SPI, I2C, UART, I2S e CAN	SPI, I2C e UART

Fonte: Lobo da Robótica[5]

As vantagens do ESP32 em relação ao arduino são inúmeras quando considerado apenas valores relacionados a poder de processamento e capacidade de armazenamento, mas podemos destacar duas principais vantagens voltadas a aplicações que necessitem de conectividade com WiFi e bateria

1 - O ESP32 já vem com módulos de WiFi e bluetooth embutidos no próprio chip.

2 - O clock de processamento pode ser alternado dentro do próprio código, fazendo assim que protótipos que exigem pouco processamento possam gastar menos energia.

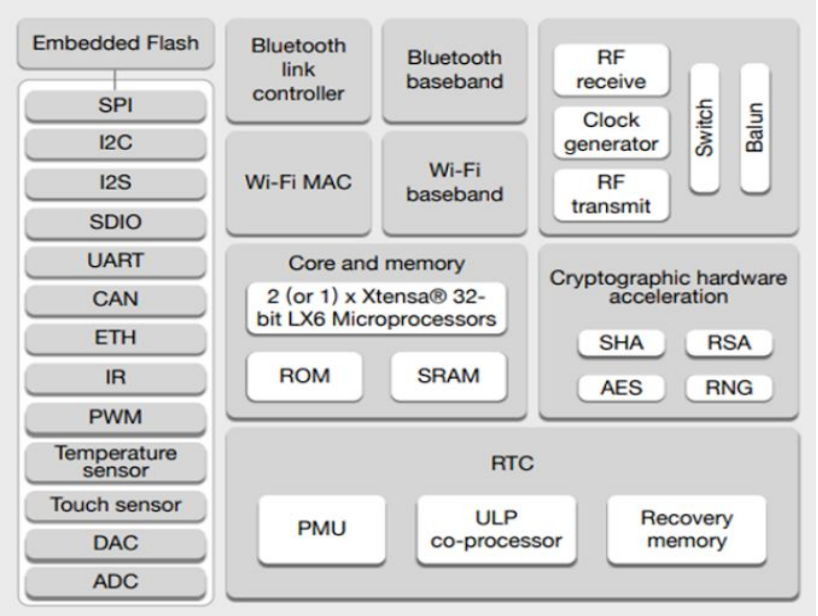
A vantagem de já se ter um módulo de WiFi embutido no próprio chip é que não se precisa comprar módulos externos para se utilizar o WiFi, pela proximidade

[4] Produtos Espressif Systems Disponivel em <https://www.espressif.com/en/products/socs> Acesso em 07 de outubro de 2022
[5] Disponível em: <https://lobodarobotica.com/blog/arduino-ou-esp-descubra-a-melhor-opcao/>. Acesso em: 08 de Outubro de 2022.

física entre os componentes o processamento de requisições feitas pelo WiFi se tornam mais rápidos.

Os dois núcleos que o ESP32 possui são denominados de Protocol CPU (PRO_CPU) e Application CPU (APP_CPU). Enquanto o PRO_CPU lida com WiFi, Bluetooth e outros periféricos internos o APP_CPU é destinado para o processamento do código da aplicação e para realizar a troca de informações entre os processadores é utilizado o freeRTOS[6]

Figura 3 - Diagrama de blocos do chip ESP32.



Fonte: *Explore Embedded*[7]

2.3 - Demais Componentes

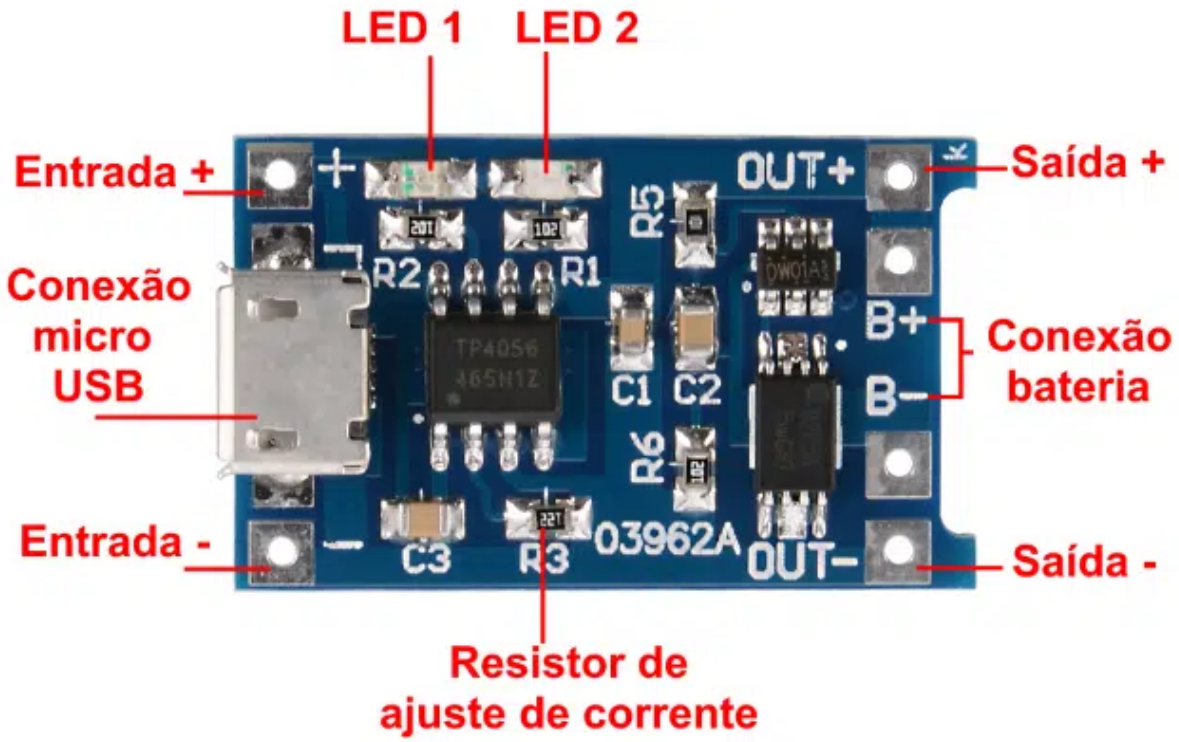
A seguir estão listados todos os demais componentes que compõem este robô, bem como uma breve análise sobre seu funcionamento para uma compreensão melhor de todo o circuito.

[6] FreeRTOS, Informação Técnica: Disponível em <https://www.freertos.org/>, Acesso em: 08 de Outubro de 2022.
[7] Disponível em: <https://www.exploreembedded.com/wiki/Overview_of_ESP32_features._What_do_they_practically_mean%3F>. Acesso em: 07 de Outubro de 2022.

2.3.1 - Módulo de carga TP4056.

O módulo TP4056 é um carregador de bateria de lítio muito utilizado em projetos por conta de sua facilidade de uso e de sua corrente de carga ser facilmente modificada para atender os mais diversos modelos de bateria.

figura 4 - Imagem Demonstrativa do módulo TP4056



Em conexão bateria (B+ e B-) temos as conexões que são ligadas diretamente na bateria, em saída e saída- temos as conexões que alimentam diretamente o circuito. O resistor R3 pode ser substituído a fim de modificar a corrente de carregamento da bateria, a tabela abaixo demonstra as substituições que podem ser feitas.

figura 5 - Tabela de substituição do resistor R3 do Módulo TP4056.

Resistor (K)	1.2	1.33	1.5	1.66	2	3	4	5	10
Corrente (mA)	1000	900	780	690	580	400	300	250	130

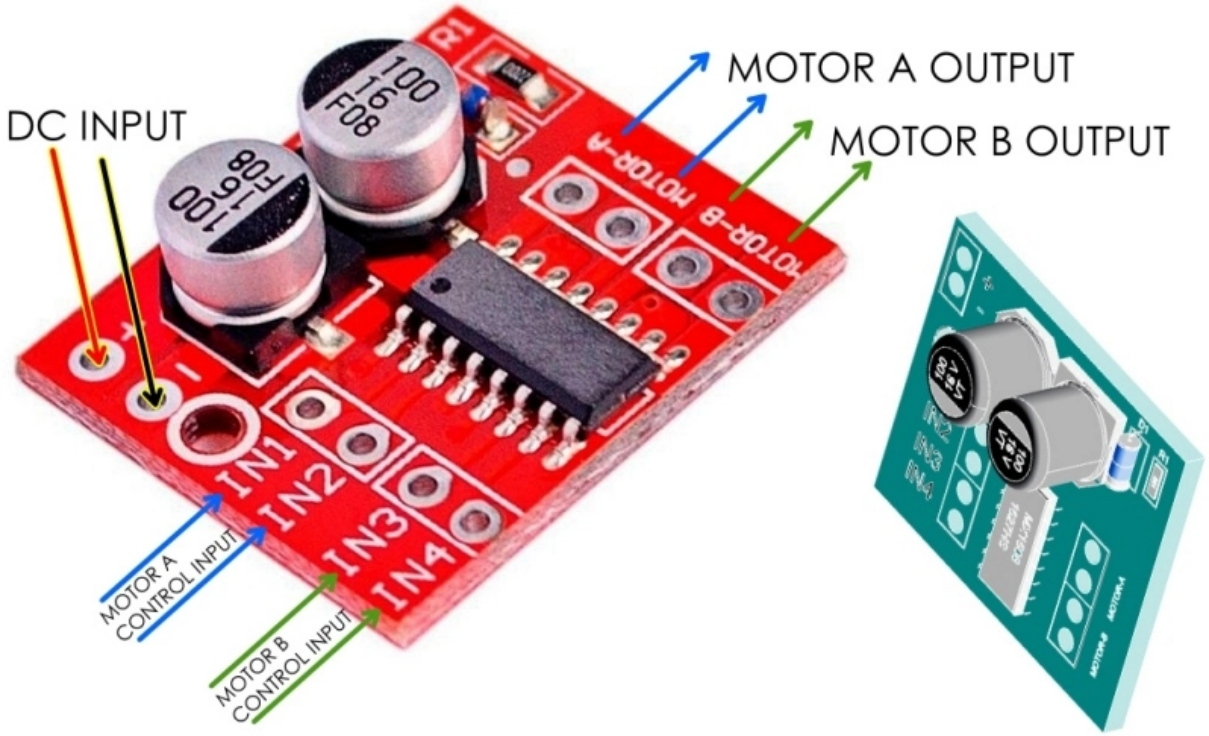
Fonte: Filipeflop[8]

[8] Disponível em: <https://www.filipeflop.com/blog/saiba-como-funciona-o-modulo-tp4056/>. Acesso em: 07 de outubro de 2022.

2.3.2 - Mini driver motor ponte H MX1508.

O funcionamento deste módulo é muito semelhante ao driver padrão ponte H L298n, porém com uma capacidade consideravelmente menor em questão de alimentação dos motores, operando entre voltagens entre 2V e 10V, o que é mais do que suficiente para o desenvolvimento deste trabalho. Um detalhe bastante interessante deste driver é que ele pode controlar até dois motores DC ao mesmo tempo.

Figura 6 - Imagem ilustrativa do mini driver motor ponte H MX1508



Fonte: Electro Schematics[9]

As conexões IN1, IN2, IN3 e IN4 são as conexões de entrada do driver que recebem os sinais de comando para os motores, enquanto as saídas motor-A e motor-B são as saídas de energia para o funcionamento dos motores.

[9] Disponivel em: <https://www.electroschematics.com/mx1508-motor-driver-module-quick-start/>. Acesso em 07 de Outubro de 2022.

2.3.3 - Motor DC 3V-6V com caixa de Redução

Este módulo é a base de movimento para o robô. Seu funcionamento é bem simples, sendo constituído de um pequeno motor DC simples, com alimentação de 3V a 6V, sendo que a diferença de tensão aumenta ou diminui a potência do motor.

figura 7 - Motor com caixa de redução

A caixa de redução acoplada a este motor tem a função transformar a alta quantidade de rotações em força de torque, ou seja, apesar de perdemos velocidade conseguimos ganhar mais força e consequentemente suportar mais peso.

Uma coisa que deve-se ficar muito atento ao ligar esses motores é que apesar de possuir pólos positivos e negativos, diferentemente dos outros

dispositivos listados neste trabalho, quando invertidos esses pólos ele não queima, apenas muda o sentido de rotação, que é basicamente o que o driver MX1508 faz para alternar o sentido de rotação das rodas e assim conseguir diferentes direções que o robô pode seguir.

3 - Desenvolvimento.

3.1 - Configurações Iniciais.

Antes de mais nada, para iniciar a codificação do ESP32 precisa-se configurar o ambiente de programação. Neste trabalho foi usado a IDE Arduino 2.0, e após baixada precisa-se fazer a instalação do gerenciamento da placa que permite injetar código na mesma, e para isso precisamos adicionar a URL de placa adicional dentro do arduino IDE, que encontra em *File* > *Preferences*

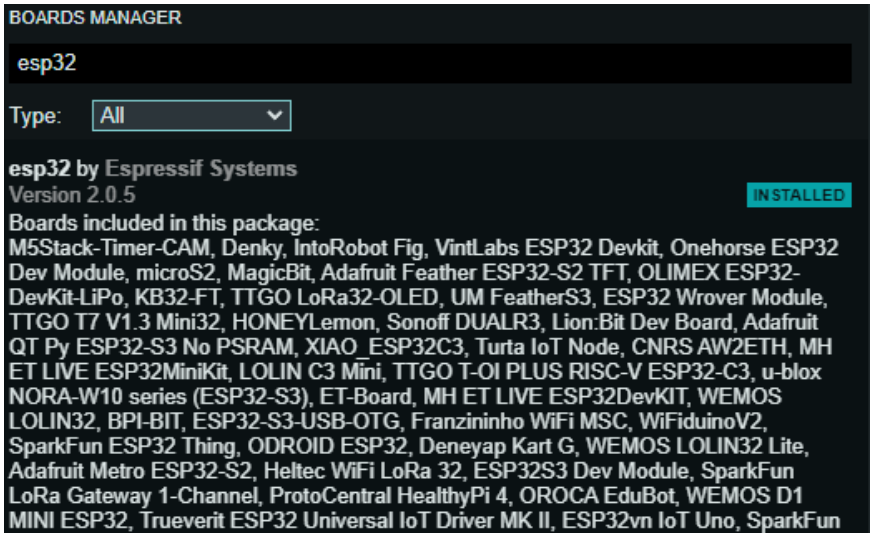
Figura 8 - Adição da URL[10] que permite a instalação da placa na IDE Arduino.

Fonte: Elaborado pelo autor.

Após adicionada a URL, a placa já se encontra disponível para instalação dentro da plataforma, clicando em *Tools* > *Boards* > *Boards Manager*:

[10] URL adicionada:
<https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json>

Figura 9 - Adição da biblioteca ESP32 disponibilizada pela *Espressif Systems*



Fonte: Elaborado pelo autor.

Dada essa configuração inicial do ambiente de desenvolvimento, pode-se começar o desenvolvimento do código.

3.2 - Funcionamento do ESP32 e envio de comandos.

Neste tópico, abordei de forma detalhada como acontece o envio de comandos para o ESP32 e como ele se comportou para transformar tais comandos em movimento.

Explicando de forma mais resumida, basicamente o ESP32 irá funcionar como um web server, este *web server* por sua vez armazena uma página que contém informações visuais de um controle para o robô, após esse envio de informações o ESP32 cuidará de interpretar esses comandos e transformar em movimento.

De maneira simplificada, um *web server* (servidor Web) é um sistema responsável pelo armazenamento, processamento e entrega de arquivos dos sites para os navegadores.

Os servidores Web basicamente seguem um modelo de cliente-Servidor. Neste modelo, um programa (cliente) solicita um serviço para outro programa (servidor), este por sua vez, processa essa solicitação e transforma em alguma ação pré estabelecida.

3.3 - Funcionamento do servidor web no ESP32.

Para realizar o tratamento do envio de informações para o ESP32, foi utilizada a capacidade do ESP32 de armazenar páginas HTTP e ser utilizado como um servidor web e ser acessado de qualquer dispositivo. Vale ressaltar que esse servidor web é disponível apenas localmente, ou seja, só será possível acessar o servidor de dentro da mesma rede em que o ESP32 está conectado.

Sendo assim, o ESP32 se conecta a uma rede wifi pré estabelecida dentro do código, e dentro desta rede ele fica aguardando outros dispositivos se conectarem para processar as solicitações de comando para o robô. O servidor é acessado através do número de IP que o modem de wifi disponibilizou para o seu funcionamento.

3.4 - Codificação

3.4.1 - Wifi e OLED

Para realizar a implementação do servidor web, vale ressaltar que foi utilizado uma pequena OLED na placa de desenvolvimento, esta por si só não altera o desenvolvimento do trabalho, serve apenas para gerar um feedback para otimização dos testes quando o robô foi alimentado pela alimentado pela bateria.

OLED significa diodo orgânico que emite luz (organic light-emitting diode) e atualmente é a tecnologia mais avançada para a fabricação de qualquer tipo de tela, seja para TVs, computadores, telefones celulares ou para seu videogame portátil favorito(Paulo Alves, 2012)

Primeiramente foram realizados os testes de conexão com wifi e seus modos de operação. Basicamente o ESP32 tem 3 modos de funcionamento quando nos referimos ao seu funcionamento como um servidor, mas vale ressaltar duas, são elas:

1 - *WIFI_STA* - Neste modo o ESP32 se conecta a um ponto de acesso

2 - *WIFI_AP* - Neste modo o ESP32 funciona como um ponte de acesso, ou seja, ele cria sua própria rede

Optei por utilizar o modo *WIFI_STA* para que o usuário conectado não perca sua conectividade com a internet, visto que no modo *WIFI_AP* teria que se

desconectar da rede wifi para se conectar na rede do ESP32, perdendo assim sua conexão com a internet.

Sendo assim, os primeiros testes se baseiam na conexão do ESP32 a rede wifi e a demonstração do endereço IP na tela OLED.

Figura 10 - endereço de IP na tela OLED após a conexão com o wifi.

Fonte: Elaborado pelo autor.

3.4.2 - Configuração do servidor e página HTML.

Após os testes com a conexão de wifi serem concluídos, podemos configurar o ESP32 para funcionar como um servidor. Primeiramente precisamos definir a porta que foi utilizado para receber as solicitações, que neste caso foi escolhida a porta 80 por ser quase que um padrão para servidores HTTP.

Para garantir a segurança do envio de requisições de comando, isto é, garantir que apenas o proprietário ou uma pessoa por vez acesse o controle e não haja duplicação ou concorrência de requisições, foi implementado um sistema de autenticação com *login* e senha.

Como o controle do robô não iria estar fora da rede, foi implementado em Base64. Base64 é um método de codificação de dados em formato de texto, é constituído por 64 caracteres ([A-Z], [a-z],[0-9], “/” e “+”).

Utilizando um usuário padrão denominado “admin” e senha padrão também denominado “admin”, obtemos o seguinte código em Base64:

YWRtaW46YWRtaW4=. Abaixo é demonstrado como é feita a codificação em Base64 para a palavra “BOLA” como exemplo.

Convertendo para binário esta palavra temos:

B(01000010) O(01001111) L(01001100) A(01000001)

Concatenando este conjunto e dividindo em grupos de 6 obtemos:

010000 100100 111101 001100 010000 01

Como no último conjunto faltam 4 caracteres para completar um conjunto de 6 elementos, adicionamos mais 4 dígitos de valor 0:

010000 100100 111101 001100 010000 010000

Agora é necessário transformar esses conjuntos de 6 *bits* para 8 *bits* e faremos isso adicionando dois zeros na frente de cada conjunto:

00010000 00100100 00111101 00001100 00010000 00010000

Consultando a tabela ASCII procuramos os binários relacionado e encontramos o número decimal correspondente a cada conjunto:

00010000(16) 00100100(36) 00111101(61) 00001100(12) 00010000(16) 00010000(16) = =

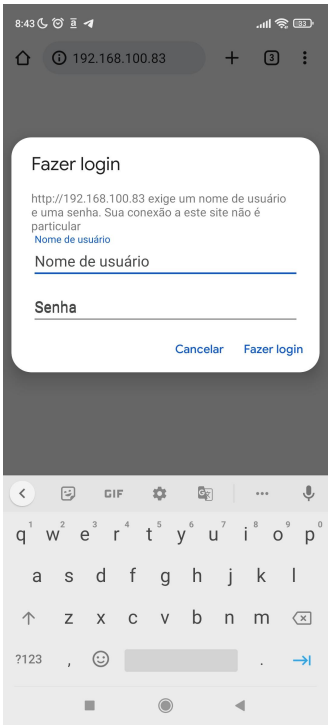
Sempre precisamos de grupos completos de 24 bits, por isso adicionamos dois marcadores “=” no final do código, esses marcadores têm valor nulo e tem apenas essa função dentro da codificação.

Procurando então a referência decimal na tabela de conversão do Base64 obtemos a seguinte conversão da palavra “BOLA”:

Qk9MQQ==

Assim, adicionamos uma página de login utilizando como parâmetro o código em Base64 para a autenticação do usuário.

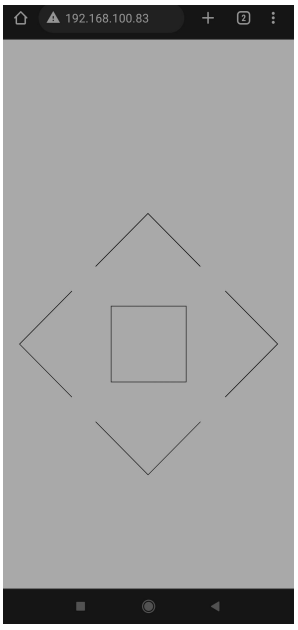
Figura 11 - Tela de solicitação de usuário e senha.



Fonte: Elaborado pelo autor.

Nesta etapa também definimos o que esse servidor armazena, que neste caso é uma página HTML que contém de forma visual um controle para o robô. Essa página também é responsável por armazenar os comandos de solicitação HTTP. Após codificado e ligado, usando o IP que o ESP32 recebeu do wifi, pode-se acessar a página apresentada na figura 12.

Figura 12 - Tela de controle do robô.



Fonte: Elaborado pelo autor.

Esta página contém os controles necessários para controlar o robô. As setas na borda são os controles de direção do robô, e o quadrado central é o comando de parada. Ao clicar em qualquer um dos botões mostrados na figura 12 uma requisição HTTP é solicitada ao servidor do ESP32, esse por sua vez processa essa solicitação e transforma em uma ação para o robô.

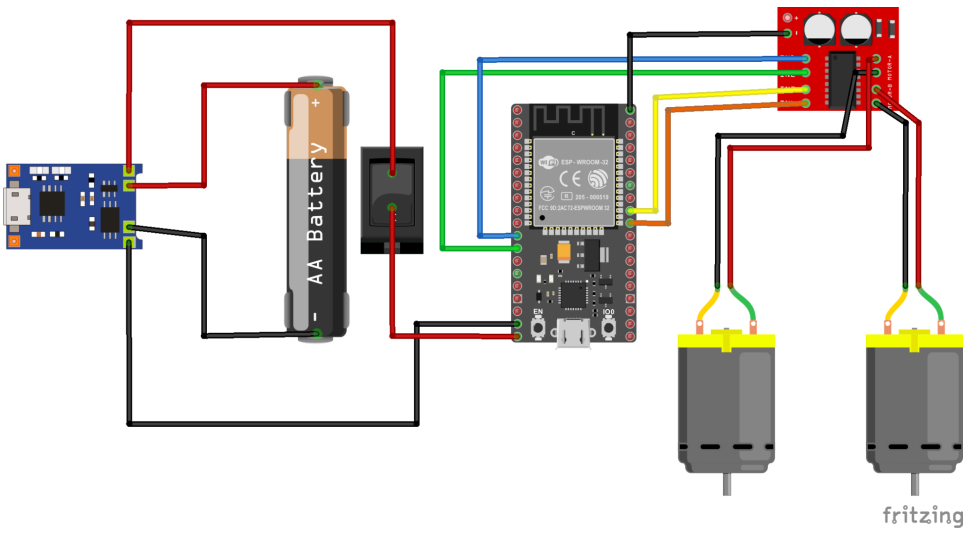
3.4.3 - Configurando o movimento das rodas.

Como este trabalho utiliza o *driver* MX1508, a programação para controlar o movimento das rodas é bem simples. Basicamente precisamos definir 4 pinos do ESP32 que são os responsáveis para o envio de sinais para o *driver*.

Definimos esses pinos como *OUTPUT*, ou seja, apenas como saída de sinal e em seguida ESP32 fica esperando as requisições de ação chegarem pelo servidor. Quando essas requisições chegam pelo servidor é feita uma análise para determinar quais dos 5 comandos disponíveis chegaram. Feita essa análise é enviado sinais para o driver e assim realizar o movimento das rodas.

O sentido para quais as rodas giram varia da forma como está ligada ao *driver*, neste trabalho elas seguem o esquema elétrico da figura 13.

Figura 13 - Esquema elétrico do robô.

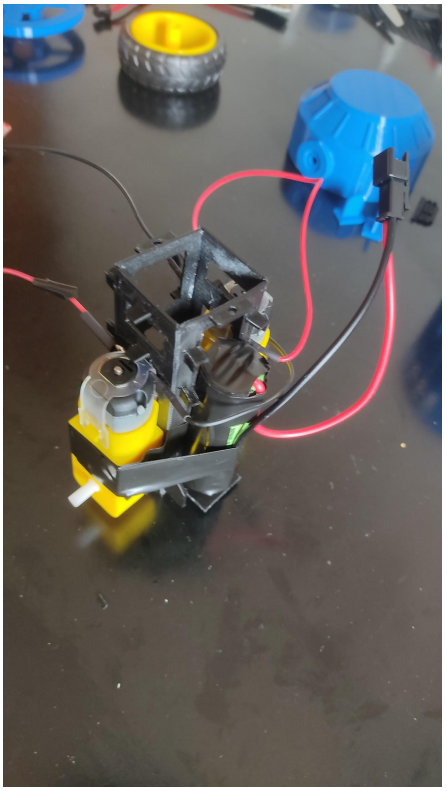


Fonte: Elaborado pelo autor.

4 - Montagem e testes finais.

Após todo o trabalho realizado do capítulo 3, chegou a hora de realizar a montagem de todo o circuito e acoplar a uma carcaça. O modelo de carcaça é ligeiramente simples, contendo uma estrutura central para a fixação dos componentes como a bateria e os motores, é basicamente o alicerce do robô.

Figura 14 - Montagem inicial da estrutura do robô.

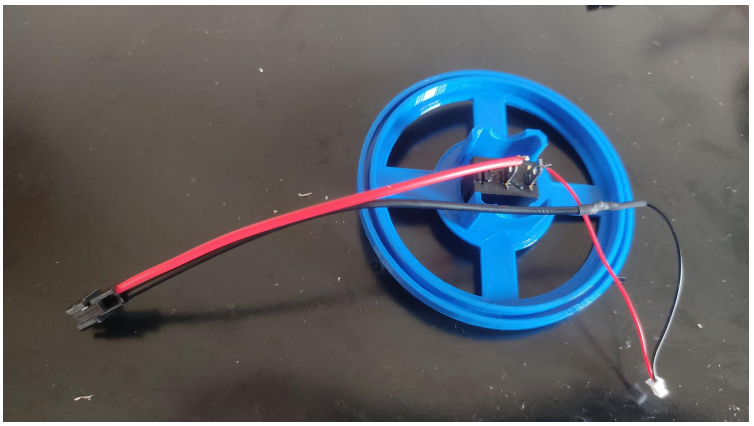


Fonte: Elaborado pelo autor.

Na figura 14 é apresentada a primeira parte da montagem, com a estrutura central da carcaça juntamente com os motores e a bateria já fixados.

Conectado os motores e bateria à estrutura central, montei a base e acoplei o interruptor, bem como fiz as soldagens necessárias. Após isso basta unir a base à estrutura central e está pronto a estrutura principal do robô.

Figura 15 - Montagem da base.

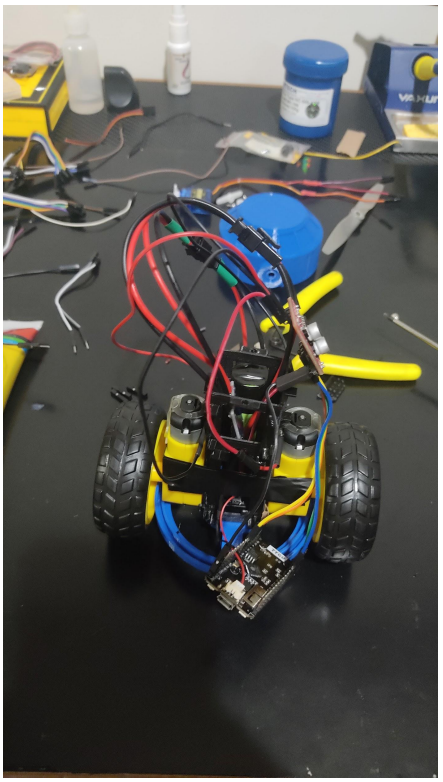


Fonte: Elaborado pelo autor.

Na figura 15 é apresentado a base onde a estrutura central será acoplada juntamente com o interruptor que é usado para ligar e desligar todo o circuito.

Após a última junção feita, já podemos acoplar todos os outros componentes a estrutura. Basicamente o robô já está pronto, a parte mais externa da carcaça tem função totalmente estética.

Figura 16 - Robô com todo o esquema elétrico montado.



Fonte: Elaborado pelo autor.

Na figura 16 é apresentada a união entre as duas partes apresentadas nas figuras 14 e 15, respectivamente.

5 - Conclusão

Hoje podemos observar que para desenvolver soluções de automação e robótica não precisamos de projetos extremamente complexos e acesso a tecnologia que até pouco tempo atrás eram disponíveis apenas para grandes empresas. Soluções como a que vimos neste trabalho estão cada vez mais acessíveis ao público e muitas das vezes usadas até como instrumento de ensino a robótica para crianças.

Através do estudo realizado no desenvolvimento deste trabalho, podemos observar o poder que os microcontroladores têm e a rápida curva de aprendizado que se tem, visto que com os módulos adicionais que desenvolveram para auxiliar a criação de proótipos não se precisa ter ideias mirabolantes para controlar os mais diversos tipos de componentes, muito menos ter grande conhecimento no mundo da eletrônica.

Este projeto pode ser facilmente adaptado para outros tipos de projetos, por exemplo, ao invés de controlar uma pequeno motor de 5V, podemos fazer o mesmo acionamento de um motor de passo para movimentar engrenagens que abrem e fecham portões automáticos.

Espero que este trabalho estimule o desenvolvimento de novas ideias utilizando microcontroladores e que desmistifique que o mundo da eletrônica juntamente com a computação seja complexa demais e de difícil acesso.

Referências Bibliográficas:

ESP32: Como aumentar a vida útil da bateria. Cap Sistema, 2021. Disponível em: <https://capsistema.com.br/index.php/2021/01/28/esp32-dicas-para-aumentar-a-vida-util-da-bateria/>. Acesso em: 07 de Outubro de 2022.

Overview of ESP32 features. What do they practically mean?. Explore Embedded, 2022. Disponível em <https://www.exploreembedded.com/wiki/Overview_of_ESP32_features._What_do_they_practically_mean%3F>. Acesso em: 07 de Outubro de 2022.

MX1508 Motor Driver Module - Quick Start. Electro Schematics, 2022. Disponível em: <https://www.electroschematics.com/mx1508-motor-driver-module-quick-start/>. Acesso em: 07 de Outubro de 2022.

Saiba como funciona o módulo TP4056. Adilson Thomsen, 2022. Disponível em: <https://www.filipeflop.com/blog/saiba-como-funciona-o-modulo-tp4056/>. Acesso em 07 de Outubro de 2022.

Web Server: O que é e Como Funciona?. Eduardo Weber, 2022. Disponível em: <https://www.hostinger.com.br/tutoriais/web-server>. Acesso em: 07 de Outubro de 2022.

Introdução ao Fritzing. Gedeane Kenshima, 2020. Disponível em: <https://www.filipeflop.com/blog/introducao-ao-fritzing/>. Acesso em: 07 de Outubro de 2022.

O que é Base64, para que serve e como funciona?. Henrique Marques Fernandes, 2020. Disponível em <https://marquesfernandes.com/self/o-que-e-base64-para-que-serve-e-como-funciona/>. Acesso em 07 de Outubro de 2022

O que é OLED?. Paulo Alvez, 2012. Disponível em <https://www.techtudo.com.br/noticias/2012/09/o-que-e-oled.ghtml>. Acesso em 29/11/2022

Apêndice A - Firmware desenvolvido para o robô

```
//Bibliotecas do display OLED
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//Bibliotecas para HTTP e Wifi
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>

//Pinos do Motor
#define motor1 5
#define motor2 18
#define motor3 14
#define motor4 27

//Pinos da Tela OLED
#define OLED_SDA 4
#define OLED_SCL 15
#define OLED_RST 16
#define SCREEN_WIDTH 128 // largura da tela, em pixels
#define SCREEN_HEIGHT 64 // Altura da tela, em pixels

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);

const char *ssid = "Nome_da_rede"; //nome da rede WIFI
const char *password = "password"; //senha da rede wifi

const char* base64Encoding = "YWRtaW46YWRtaW4="; //user=admin senha=admin

String header;

WiFiServer server(80); //Porta que o servidor irá usar

void setup() {
  //Inicializa o serial Monitor
  Serial.begin(115200);

  //reset OLED display via software
  pinMode(OLED_RST, OUTPUT);
  digitalWrite(OLED_RST, LOW);
  delay(20);
 digitalWrite(OLED_RST, HIGH);
```

```
  //inicialização do OLED
  Wire.begin(OLED_SDA, OLED_SCL);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x3C para
128x32
          Serial.println(F("SSD1306 allocation failed"));
          for(;;);
  }

  WiFi.mode(WIFI_STA); //Definição do modo do WIFI
  WiFi.begin(ssid, password);

  // esperando a conexão do wifi
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  //Printa na OLED
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0,0);
  display.println("connected");
  display.print("IP: ");
  display.println(WiFi.localIP());
  display.display();
  /////

  //inicializa motor
  pinMode(motor1, OUTPUT);
  pinMode(motor2, OUTPUT);
  pinMode(motor3, OUTPUT);
  pinMode(motor4, OUTPUT);

  //Definição de estado das saidas
  digitalWrite(motor1, LOW);
  digitalWrite(motor2, LOW);
  digitalWrite(motor3, LOW);
  digitalWrite(motor4, LOW);

  server.begin();//inicia o servidor HTTP
}

void loop() {
 WiFiClient client = server.available();
  if (client) {
    Serial.println("New Client.");
```

```
   String currentLine = "";
   while (client.connected()) {
     if (client.available()) {
       char c = client.read();
       Serial.write(c);
       header += c;
       if (c == '\n') {
         if (currentLine.length() == 0) {
           ///////Inicio HTML
           if(header.indexOf(base64Encoding)>=0){
             client.println("HTTP/1.1 200 OK");
           client.println("Content-type:text/html");
           client.println("Connection: close");
           client.println();

           client.print("<html>");
           client.print("<head><title> ""CONTROLE ROBO"" </title>");
             client.print("<style>");
             client.print("body {padding: 200px 0 0; background: white; display: flex;
flex-direction: row; justify-content: center; align-items: center;}");
             client.print(".frente,");
             client.print(".direita,");
             client.print(".esquerda,");
             client.print(".voltar {width: 250px; height: 250px; transition: .5s; float: left;
box-shadow: -2px 2px 0 black;}");
             client.print(".frente {transform: rotate(135deg);}");
             client.print(".voltar {transform: rotate(-45deg);}");
             client.print(".esquerda {transform: rotate(45deg);}");
             client.print(".direita{transform: rotate(-135deg);}");
             client.print(".cabdireita{transform: rotate(45deg);}");
             client.print(".cabesquerda{transform: rotate(-135deg);}");
             client.print(".parar{width: 250px;height: 250px;border: 2px solid black; float: left;}");
             client.print("</style></head>");
             client.print("<body>");
             client.print("<table>");
             client.print("<tr><td></td><td><a href=\"/FRENTE\"
class=\"frente\"></a></td><td></td></tr>");
             client.print("<tr>");
             client.print("<td><a href=\"/ESQUERDA\" class=\"esquerda\"></a></td>");
             client.print("<td><a href=\"/PARAR\" class=\"parar\"></a></td>");
             client.print("<td><a href=\"/DIREITA\" class=\"direita\"></a></td>");
             client.print("</tr>");
             client.print("<tr><td></td><td><a href=\"/VOLTAR\"
class=\"voltar\"></a></td><td></td></tr>");
             client.print("</table>");
             client.print("</body>");
           client.print("</html>");
           break;
```

```
          }else{
            client.println("HTTP/1.1 401 Unauthorized");
            client.println("WWW-Authenticate: Basic realm=\"Secure\"");
            client.println("Content-Type: text/html");
            client.println();
            client.println("<html>Authentication failed</html>");
///////FIM HTML
          }

         } else {
          currentLine = "";
         }
       } else if (c != '\r') {
         currentLine += c;
       }
       if (currentLine.endsWith("GET /FRENTE")) {
         digitalWrite(motor1, LOW);
         digitalWrite(motor2, HIGH);
         digitalWrite(motor3, HIGH);
         digitalWrite(motor4, LOW);
       } else if (currentLine.endsWith("GET /VOLTAR")) {
         digitalWrite(motor1, HIGH);//GPIO 5
         digitalWrite(motor2, LOW);//GPIO 18
         digitalWrite(motor3, LOW);
         digitalWrite(motor4, HIGH);
       }else if (currentLine.endsWith("GET /DIREITA")) {
         digitalWrite(motor1, LOW);
         digitalWrite(motor2, HIGH);
         digitalWrite(motor3, LOW);
         digitalWrite(motor4, LOW);
       }else if (currentLine.endsWith("GET /ESQUERDA")) {
         digitalWrite(motor1, LOW);
         digitalWrite(motor2, LOW);
         digitalWrite(motor3, HIGH);
         digitalWrite(motor4, LOW);
       }else if (currentLine.endsWith("GET /PARAR")) {
         digitalWrite(motor1, LOW);
         digitalWrite(motor2, LOW);
         digitalWrite(motor3, LOW);
         digitalWrite(motor4, LOW);
       }

     }
    }
  }
  client.stop();
}
```