CENTRO PAULA SOUZA
ETEC PAULINO BOTELHO
Técnico em Mecatrônica

Bianca Gabrielli da Silva
Everton Boaventura
Fabio Montanari
José Benedito Gomes

ROBÔ ESTEIRA

São Carlos
2024

Bianca Gabrielli da Silva

Everton Boaventura

Fabio Montanari

José Benedito Gomes

ROBÔ ESTEIRA

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Mecatrônica da Etec Paulino Botelho, orientado pelo Prof. Gabriel Luiz Bacha Junho, como requisito parcial para obtenção do título de técnico em mecatrônica.

São Carlos

2024

RESUMO

Desenvolvemos um robô esteira, para realização de trabalhos emergenciais e com possível risco à vida.
O projeto do robô tem o intuito de ajudar a verificar lugares confinados, lugares com possíveis riscos de vazamentos tóxicos, aproximação de objetos não identificados com potencial de risco, como por exemplo: uma bomba. Já que o próprio robô será controlado remotamente, ou seja, mantendo o operador á distancia de qualquer pessoa e perigo.

Palavras chaves: Robô Esteira; ESP32-CAM; Câmera robô;

ABSTRACT

We have developed a tracked robot for emergency response and high-risk situations. The robot's design is intended to help inspect confined spaces, areas with potential toxic leaks, and approach unidentified objects posing a risk, such as bombs. Since the robot will be controlled remotely, the operator will remain at a safe distance from any person or danger.

Keywords: *Tracked Robot; ESP32-CAM; Robot camera.*

LISTA DE FIGURAS

SUMÁRIO

1. Introdução

O projeto foi desenvolvido através de pesquisas de alguns planos já existentes, identificando as necessidades do robô e suas atuações.

Ao encontrar modelos de vários tipos de robôs esteira que atuam em áreas como:

Exemplos: Industriais, militares, civis etc.

A decisão de desenvolver esse projeto se justifica pela sua grande área de aplicações.

Na pesquisa foi utilizado ferramentas como: Google, YouTube, em canais especializados na área, imagens, entre outros.

2. DESENVOLVIMENTO

2.1 FUNDAMENTAÇÃO TEORICA

Especificações técnicas dos materiais utilizados (mecânica)

- Chassi: tubo de metalon 20x20
- 4 Coroas de bicicletas simples
- 4 cubos simples com rolamentos e eixo
- 8 molas para amortecedores
- Parafusos allen, porcas e arruelas
- Corrente moto
- 2 pinhões de moto
- Peças eletroeletrônicas
- 2 motores de vidro elétrico de automóvel 12v
- Arduino ESP com câmera
- Ponte H dupla de 35 amperes
- Cabos
- Led’s
- Bateria 12 V – 7 Ah.

2.1.1 Ponte H

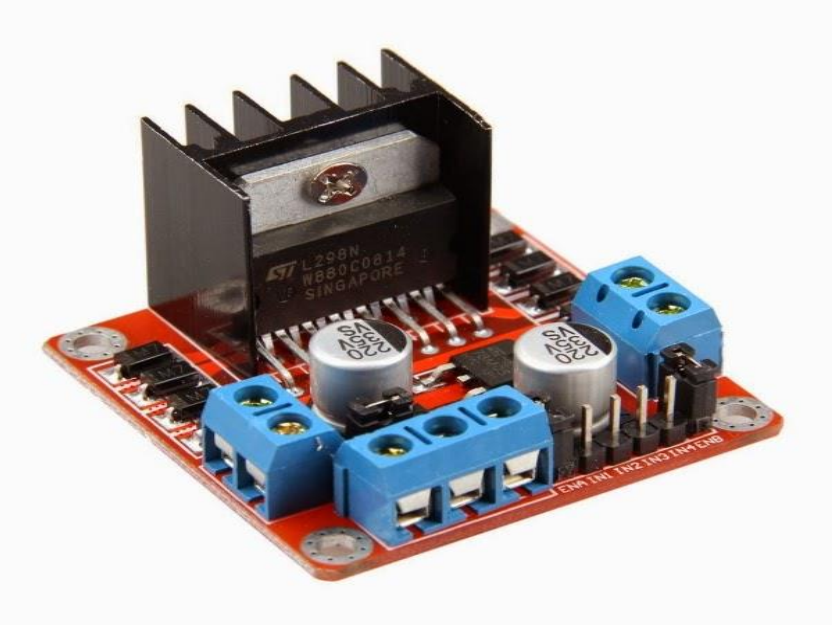
Uma ponte H é um circuito eletrônico que permite controlar a direção de rotação de um motor de corrente contínua (DC). A principal função de uma ponte H é permitir que um motor DC gire no sentido anti-horário, além de permitir o controle de velocidade do motor, se usado em conjunto com um controle de modulação por largura de pulso (PWM).

Função da Ponte H:

Controle de Direção: Permite que o motor DC mude de direção, possibilitando a rotação tanto para frente, quanto para trás.

Controle de Velocidade: Em combinação com PWM, pode controlar a velocidade do motor ajustando a largura dos pulsos enviados.

Figura 1 EXEMPLO DE PONTE H

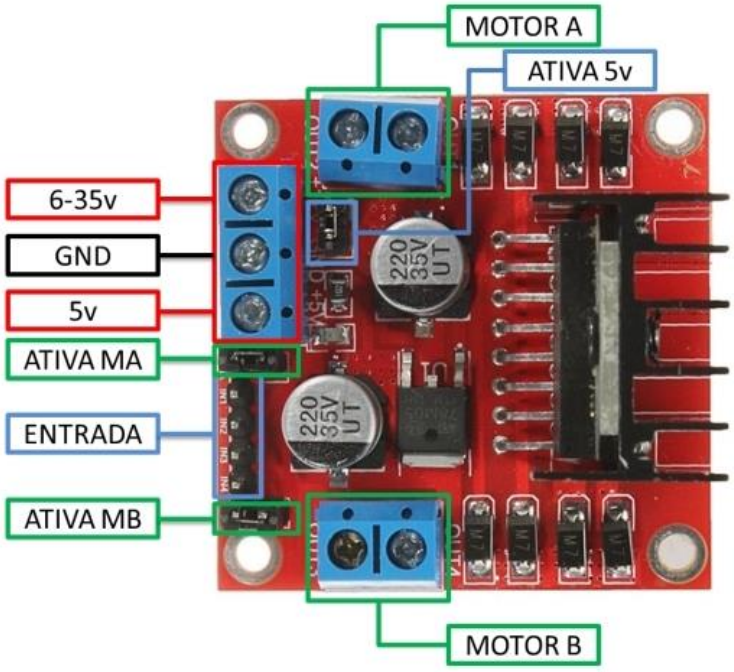


Fonte: https//www.autocorerobotica.com.br

2.1.2 Circuito Integrado L298N:

O circuito integrado L298N é muito utilizado para controlar motores. Uma das vantagens do uso deste circuito é o menor
espaço ocupado, a baixa complexidade do circuito e o fato de ele já possuir dois circuitos H, podendo assim, controlar dois motores. Outra vantagem do L298N é a resposta a sinais de PWM. Se forem utilizados sinais de PWM, é possível regular a tensão de saída, e dessa forma regular a velocidade dos motores. Conhecendo o funcionamento...

Figura 2 EXEMPLO DE CIRCUITO L298N



Fonte: ***https//www.autocorerobotica.com.br***

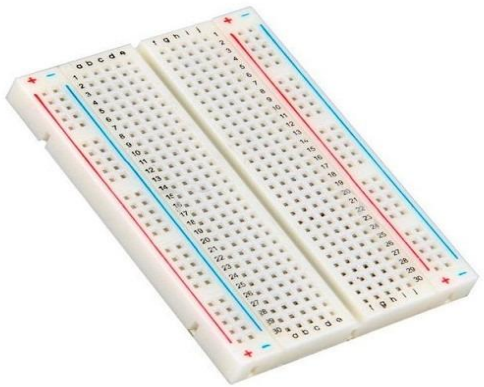
2.1.3 Motor A e Motor B:

Borne para conexão dos motores DC.
VCC: Bornes responsáveis pela alimentação dos motores.
GND: É o GND da placa, que deverá ser o mesmo GND do Arduino. Por isso, é importante interligar os GND's sempre com fios.
5V: Em casos de não haver fonte de alimentação com mais de 6V podemos alimentar a placa com 5V por esta porta.
(Ativa MA) e (Ativa MB): são os pinos responsáveis pelo controle PWM dos motores A e B. Se estiver com jumper, não haverá controle de velocidade.
(IN1 e IN2) e (IN3 e IN4): São utilizados para controlar o sentido dos motores.
Ativa 5V: É o Jumper responsável por regular a alimentação da Ponte H, caso o jumper esteja conectado, a alimentação deve ser de 6 a 12V. Se o jumper está desconectado, é preciso utilizar a porta que fornece 5V do ESP, porém é necessário se atentar a tensão do motor utilizado e a corrente, caso contrário o ESP pode queimar.

2.1.4 Protoboard

Nada mais é do que uma placa com furos e conexões condutoras utilizada para a montagem de protótipos e projetos que estão na fase inicial. A vantagem de se utilizar uma Protoboard é que podemos montar os componentes direto nele, assim eliminamos a necessidade de utilizar a solda.

Figura 3 EXEMPLO DE PROTOBOARD



Fonte: ***https//www.autocorerobotica.com.br***

2.1.5 Jumpers

Os Jumpers nada mais são do que condutores para ligar um ponto a outro.

Figura 4 EXEMPLO DE JUMPERS

Fonte: ***https//www.autocorerobotica.com.br***

2.2. ESPECIFICAÇÕES TÉCNICAS (IMAGEM)

Câmera ESP32: O modelo ESP32-CAM é um microcontrolador com uma câmera integrada que permite capturar imagens e transmiti-las através de WI-FI.

Função da Câmera ESP32:

Captura de Imagens: A câmera captura imagens em vários formatos e resoluções.

Transmissão de Imagens: Utilizando as capacidades WI-FI do ESP32, as imagens podem ser transmitidas por um servidor, uma página web ou um aplicativo.

Figura 5 EXEMPLO DE CÂMERA ESP32

Fonte: Google Imagens

2.3 PROGRAMAÇÃO ESP32

Antes de iniciarmos as ligações dos componentes no ESPCAM, precisamos programá-lo.

Para ser possível enviar a programação para a placa, vamos precisar de um módulo adaptador Esp32 Cam MB, apresentado na figura abaixo.

Figura 6 PLACA DE PROGRAMAÇÃO ESP32

Fonte: ***https//www.autocorerobotica.com.br***

Para esse robô, deve-se escolher entre dois programas, as diferenças entre eles são:

2.3.1 Primeiro programa:

No primeiro programa, a
placa irá criar um sinal de Wi-Fi, onde deverá conectar com um computador ou celular e acessar o site para realizar o controle do robô.

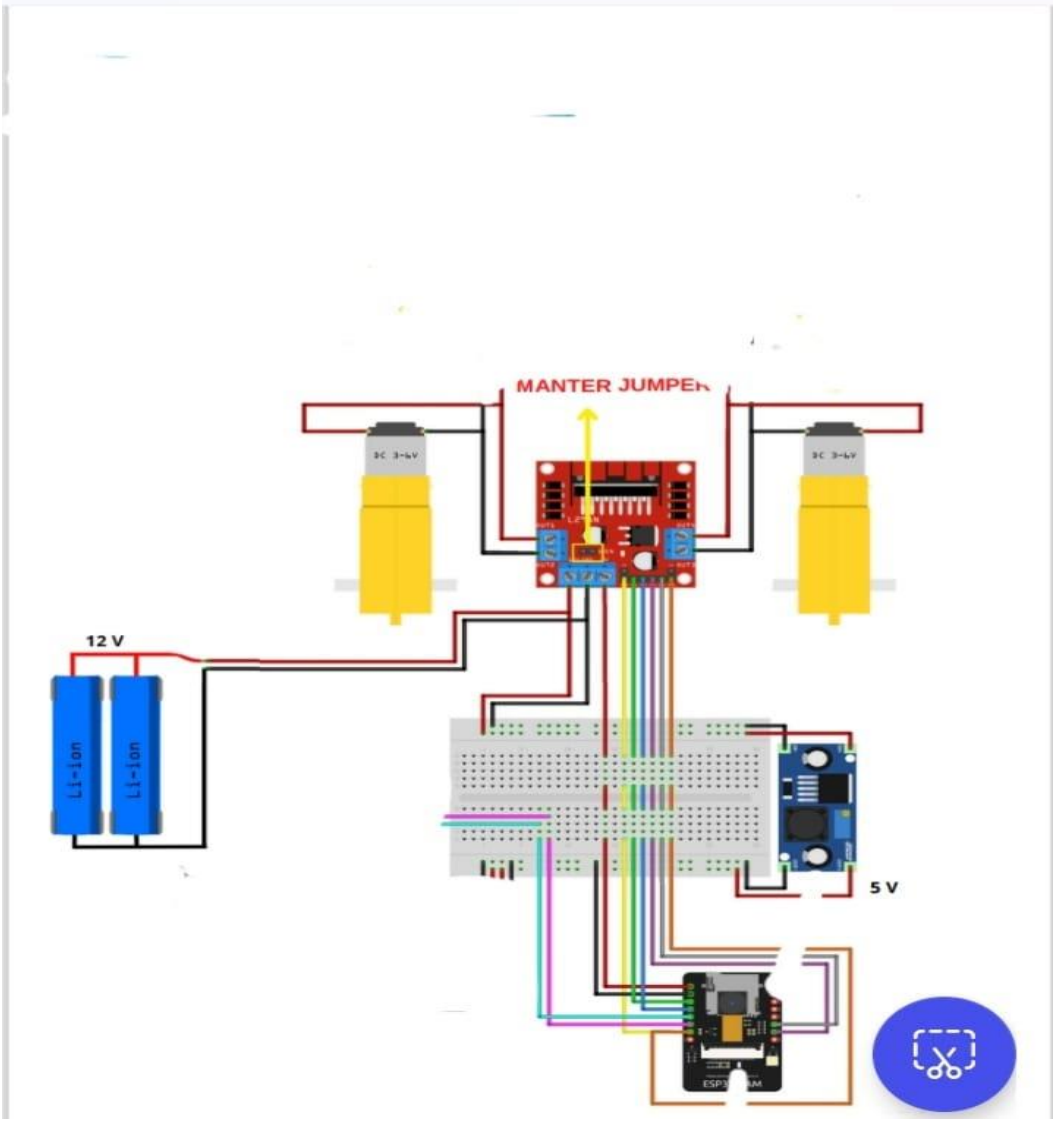
2.3.2 Segundo programa:

No segundo programa, a placa irá se conectar a um sinal de Wi-Fi já existente, essa forma será necessária informar a senha e a rede na programação. Para controlar o robô, algum dispositivo (celular ou computador) deve estar conectado a mesma rede que a placa.

JUSTIFICATIVA: No primeiro programa, a placa emitirá um sinal Wi-Fi que deve ser conectado a algum dispositivo, já no segundo, a placa se conectará em uma rede Wi-Fi já existente, para ser possível fazer uma conexão entre o celular e o robô.

2.4 ESQUEMA ELÉTRICO

Figura 7 ESQUEMA ELÉTRICO



Fonte: simulador tinkercad

2.5 PROGRAMAÇÃO

```
#include "esp_camera.h"
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <ESP32Servo.h>  // Inclua a biblioteca ESP32Servo para controle dos
servos

// Configuração da câmera
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

// Configuração da rede Wi-Fi
const char* ssid = "robô esteira etec 2024";    // Nome da rede Wi-Fi
const char* password = "12345678"; // Senha da rede Wi-Fi

// Configuração do servidor
AsyncWebServer server(80);       // Cria um servidor web na porta 80
AsyncWebSocket wsCamera("/Camera"); // Cria um WebSocket para a câmera
uint32_t cameraClientId = 0;     // ID do cliente conectado ao WebSocket

// Definições dos pinos dos servos e da luz
#define SERVO_PAN_PIN 14        // Pino do servo para movimentação horizontal
(pan)
#define SERVO_TILT_PIN 15       // Pino do servo para movimentação vertical
(tilt)
#define LIGHT_PIN 4             // Pino para controlar a luz

// Definições dos pinos dos motores
#define RIGHT_MOTOR_EN  2       // Pino de habilitação do motor direito
```

```
#define RIGHT_MOTOR_IN1 12      // Pino de controle do motor direito (IN1)
#define RIGHT_MOTOR_IN2 13      // Pino de controle do motor direito (IN2)
#define LEFT_MOTOR_EN   2       // Pino de habilitação do motor esquerdo
#define LEFT_MOTOR_IN3  1       // Pino de controle do motor esquerdo (IN3)
#define LEFT_MOTOR_IN4  3       // Pino de controle do motor esquerdo (IN4)

// Inicialize os servos com a biblioteca ESP32Servo
Servo servoPan;   // Servo para movimentação horizontal
Servo servoTilt;  // Servo para movimentação vertical

bool lightState = false;  // Estado inicial da luz (apagada)
int motorSpeed = 127;       // Velocidade inicial dos motores (127)

void setupCamera() {
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 10;
  config.fb_count = 1;

  esp_err_t err = esp_camera_init(&config); // Inicializa a câmera com a
configuração
  if (err != ESP_OK) {                    // Verifica se houve erro na
inicialização
    Serial.printf("Falha na inicialização da câmera com erro 0x%x\n", err);
    return; // Adiciona retorno para evitar erros subsequentes
  }
  Serial.println("Câmera inicializada com sucesso"); // Mensagem de sucesso
```

```
}

void sendCameraPicture() {
  if (cameraClientId == 0) return; // Se não houver cliente conectado, não
envia imagem

  camera_fb_t *fb = esp_camera_fb_get(); // Captura um frame da câmera
  if (!fb) { // Verifica se a captura falhou
    Serial.println("Falha na captura da câmera");
    return;
  }

  wsCamera.binary(cameraClientId, fb->buf, fb->len); // Envia a imagem via
WebSocket
  esp_camera_fb_return(fb); // Libera o buffer da câmera
}

void setupMotors() {
  pinMode(RIGHT_MOTOR_EN, OUTPUT);    // Configura o pino do motor direito
como saída
  pinMode(RIGHT_MOTOR_IN1, OUTPUT);   // Configura o pino IN1 do motor direito
como saída
  pinMode(RIGHT_MOTOR_IN2, OUTPUT);   // Configura o pino IN2 do motor direito
como saída
  pinMode(LEFT_MOTOR_EN, OUTPUT);     // Configura o pino do motor esquerdo
como saída
  pinMode(LEFT_MOTOR_IN3, OUTPUT);    // Configura o pino IN3 do motor
esquerdo como saída
  pinMode(LEFT_MOTOR_IN4, OUTPUT);    // Configura o pino IN4 do motor
esquerdo como saída
}

void setup() {
  Serial.begin(115200);               // Inicializa a comunicação serial a
115200 bps

  pinMode(LIGHT_PIN, OUTPUT);         // Configura o pino da luz como saída
  digitalWrite(LIGHT_PIN, LOW);       // Inicialmente a luz está apagada

  setupMotors();                      // Configura os pinos dos motores

  WiFi.softAP(ssid, password);        // Inicializa o ponto de acesso Wi-Fi
  IPAddress IP = WiFi.softAPIP();     // Obtém o endereço IP do ponto de
acesso
  Serial.print("AP IP address: ");
  Serial.println(IP);
```

```
  setupCamera();                        // Inicializa a câmera


//Criação do site

  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
    request->send(200, "text/html", R"rawliteral(
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">
  <title>Robo Omega</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      display: flex;
      flex-direction: column;
      height: 100vh;
      background: linear-gradient(135deg, #1b1b1b, #003d3d, #004d4d, #005e5e);
      color: white;
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      overflow: hidden;
    }
    .container {
      display: flex;
      flex-direction: row;
      height: 100%;
      width: 100%;
    }
    .left, .right, .center {
      display: flex;
      align-items: center;
      justify-content: center;
      padding: 10px;
      border-radius: 10px;
      box-shadow: 0 0 15px rgba(0, 0, 0, 0.7);
    }
    .left {
      flex: 1;
      flex-direction: column;
      align-items: flex-start;
      border-right: 2px solid #00979C;
      background: rgba(0, 0, 0, 0.6);
      overflow-y: auto;
```

```
}
.right {
  flex: 1;
  flex-direction: column;
  align-items: center;
  border-left: 2px solid #00979C;
  background: rgba(0, 0, 0, 0.6);
  overflow-y: auto;
}
.center {
  flex: 2;
  background: rgba(0, 0, 0, 0.8);
  display: flex;
  align-items: center;
  justify-content: center;
}
.control-button {
  width: 40px;
  height: 40px;
  background-color: #00979C;
  color: white;
  border: none;
  border-radius: 5px;
  margin: 3px;
  font-size: 14px;
  cursor: pointer;
  transition: transform 0.1s ease-in-out, box-shadow 0.1s ease-in-out;
}
.control-button:active {
  background-color: #007373;
  transform: scale(0.95);
  box-shadow: 0 0 6px rgba(0, 0, 0, 0.5);
}
.slider-container, .button-container {
  background: rgba(41, 41, 41, 0.7);
  border-radius: 10px;
  padding: 8px;
  box-shadow: 0 0 10px #00979C;
  margin: 5px;
}
.noselect {
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
}
```

```
.slidecontainer {
  width: 100%;
  padding: 5px 0;
}
.slider {
  -webkit-appearance: none;
  width: 100%;
  height: 3px;
  border-radius: 5px;
  background: #666;
  outline: none;
  opacity: 0.8;
  transition: opacity .2s;
}
.slider:hover {
  opacity: 1;
}
.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 10px;
  height: 10px;
  border-radius: 50%;
  background: #00979C;
  cursor: pointer;
  box-shadow: 0 0 4px rgba(0, 0, 0, 0.4);
}
.slider::-moz-range-thumb {
  width: 10px;
  height: 10px;
  border-radius: 50%;
  background: #00979C;
  cursor: pointer;
  box-shadow: 0 0 4px rgba(0, 0, 0, 0.4);
}
td[style="text-align:left"] b {
  color: #e0e0e0;
  font-weight: 600;
}
#cameraImage {
  border: 2px solid #00979C;
  border-radius: 10px;
  max-width: 100%;
  max-height: 100%;
  object-fit: contain;
}
.dpad {
  display: grid;
```

```
      grid-template-columns: 40px 40px 40px;
      grid-template-rows: 40px 40px 40px;
      gap: 3px;
    }
    .dpad button {
      width: 40px;
      height: 40px;
    }
    .dpad #stop {
      grid-column: 2;
      grid-row: 2;
    }
    .dpad #forward {
      grid-column: 2;
      grid-row: 1;
    }
    .dpad #backward {
      grid-column: 2;
      grid-row: 3;
    }
    .dpad #left {
      grid-column: 1;
      grid-row: 2;
    }
    .dpad #right {
      grid-column: 3;
      grid-row: 2;
    }
  </style>
</head>
<body class="noselect">
  <div class="container">
    <div class="left">
      <div class="button-container">
        <table style="width:100%; table-layout:fixed; border-collapse:
separate; border-spacing: 4px;">
          <tr>
            <td style="text-align:left"><b>Luz:</b></td>
            <td colspan=2>
              <button id="lightToggle" class="control-button">ON</button>
            </td>
          </tr>
          <tr>
            <td style="text-align:left"><b>Vel:</b></td>
            <td colspan=2>
              <div class="slider-container">
                <input type="range" min="0" max="255" value="150"
class="slider" id="speed">
```

```
            </div>
          </td>
        </tr>
        <tr>
          <td style="text-align:left"><b>Pan:</b></td>
          <td colspan=2>
            <div class="slider-container">
              <input type="range" min="0" max="180" value="90"
class="slider" id="pan">
            </div>
          </td>
        </tr>
        <tr>
          <td style="text-align:left"><b>Tilt:</b></td>
          <td colspan=2>
            <div class="slider-container">
              <input type="range" min="0" max="180" value="90"
class="slider" id="tilt">
            </div>
          </td>
        </tr>
      </table>
    </div>
  </div>
  <div class="center">
    <img id="cameraImage" src="" alt="Camera">
  </div>
  <div class="right">
    <div class="button-container">
      <div class="dpad">
        <button id="forward" class="control-button">▲</button>
        <button id="left" class="control-button">◄</button>
        <button id="stop" class="control-button">■</button>
        <button id="right" class="control-button">►</button>
        <button id="backward" class="control-button">▼</button>
      </div>
    </div>
  </div>
</div>
<script>
  var websocketCamera = new WebSocket('ws://' + location.hostname +
'/Camera');
  websocketCamera.binaryType = 'blob';
  websocketCamera.onmessage = function(event) {
    document.getElementById('cameraImage').src =
URL.createObjectURL(event.data);
  };
```

```
    document.getElementById('pan').addEventListener('input', function(event) {
      var panValue = event.target.value;
      fetch('/pan?value=' + panValue);
    });

    document.getElementById('tilt').addEventListener('input', function(event)
{
      var tiltValue = event.target.value;
      fetch('/tilt?value=' + tiltValue);
    });

    document.getElementById('speed').addEventListener('input', function(event)
{
      var speedValue = event.target.value;
      fetch('/speed?value=' + speedValue);
    });

    document.getElementById('lightToggle').addEventListener('click',
function() {
      var button = document.getElementById('lightToggle');
      var currentState = button.textContent;
      var newState = currentState === 'ON' ? 'OFF' : 'ON';
      button.textContent = newState;
      fetch('/light?state=' + newState);
    });


    function move(direction) {
      fetch('/move?direction=' + direction);
    }

    function stopMovement() {
      fetch('/move?direction=stop');
    }

    ['forward', 'backward', 'left', 'right'].forEach(function(direction) {
      var button = document.getElementById(direction);
      button.addEventListener('mousedown', function() {
        move(direction);
      });
      button.addEventListener('mouseup', function() {
        stopMovement();
      });
      button.addEventListener('touchstart', function() {
        move(direction);
      });
      button.addEventListener('touchend', function() {
```

```
        stopMovement();
      });
    });

    var stopButton = document.createElement('button');
    stopButton.textContent = 'Stop';
    stopButton.className = 'control-button';
    stopButton.addEventListener('click', stopMovement);
    document.querySelector('main').appendChild(stopButton);
  </script>


</body>
</html>
)rawliteral");
  });

  // Manipula as requisições para ajustar o ângulo do servo Pan
  server.on("/pan", HTTP_GET, [](AsyncWebServerRequest *request) {
    String panValue = request->getParam("value")->value(); // Obtém o valor do
ângulo do parâmetro
    int panAngle = panValue.toInt(); // Converte o valor para inteiro
    servoPan.write(panAngle);  // Define o ângulo do servo Pan
    request->send(200, "text/plain", "OK"); // Envia uma resposta de sucesso
  });

  // Manipula as requisições para ajustar o ângulo do servo Tilt
  server.on("/tilt", HTTP_GET, [](AsyncWebServerRequest *request) {
    String tiltValue = request->getParam("value")->value(); // Obtém o valor
do ângulo do parâmetro
    int tiltAngle = tiltValue.toInt(); // Converte o valor para inteiro
    servoTilt.write(tiltAngle);  // Define o ângulo do servo Tilt
    request->send(200, "text/plain", "OK"); // Envia uma resposta de sucesso
  });

  // Manipula as requisições para ajustar a velocidade dos motores
  server.on("/speed", HTTP_GET, [](AsyncWebServerRequest *request) {
    String speedValue = request->getParam("value")->value(); // Obtém o valor
da velocidade do parâmetro
    motorSpeed = speedValue.toInt(); // Converte o valor para inteiro e
atualiza a velocidade dos motores
    request->send(200, "text/plain", "OK"); // Envia uma resposta de sucesso
  });

  // Manipula as requisições para alternar o estado da luz
  server.on("/light", HTTP_GET, [](AsyncWebServerRequest *request) {
    lightState = !lightState;  // Alterna o estado da luz
```

```
    digitalWrite(LIGHT_PIN, lightState ? HIGH : LOW);  // Atualiza o pino da
luz
    request->send(200, "text/plain", lightState ? "Light On" : "Light Off");
// Envia uma resposta com o novo estado da luz
  });

  // Manipula as requisições para movimentar o robô
server.on("/move", HTTP_GET, [](AsyncWebServerRequest *request) {
    String direction = request->getParam("direction")->value(); // Obtém a
direção do parâmetro
    if (direction == "forward") {
      // Movimento para frente
      analogWrite(RIGHT_MOTOR_EN, motorSpeed);
      analogWrite(LEFT_MOTOR_EN, motorSpeed);
      digitalWrite(RIGHT_MOTOR_IN1, HIGH);
      digitalWrite(RIGHT_MOTOR_IN2, LOW);
      digitalWrite(LEFT_MOTOR_IN3, HIGH);
      digitalWrite(LEFT_MOTOR_IN4, LOW);
    } else if (direction == "backward") {
      // Movimento para trás
      analogWrite(RIGHT_MOTOR_EN, motorSpeed);
      analogWrite(LEFT_MOTOR_EN, motorSpeed);
      digitalWrite(RIGHT_MOTOR_IN1, LOW);
      digitalWrite(RIGHT_MOTOR_IN2, HIGH);
      digitalWrite(LEFT_MOTOR_IN3, LOW);
      digitalWrite(LEFT_MOTOR_IN4, HIGH);
    } else if (direction == "left") {
      // Movimento para esquerda
      analogWrite(RIGHT_MOTOR_EN, motorSpeed);
      analogWrite(LEFT_MOTOR_EN, motorSpeed);
      digitalWrite(RIGHT_MOTOR_IN1, HIGH);
      digitalWrite(RIGHT_MOTOR_IN2, LOW);
      digitalWrite(LEFT_MOTOR_IN3, LOW);
      digitalWrite(LEFT_MOTOR_IN4, HIGH);
    } else if (direction == "right") {
      // Movimento para direita
      analogWrite(RIGHT_MOTOR_EN, motorSpeed);
      analogWrite(LEFT_MOTOR_EN, motorSpeed);
      digitalWrite(RIGHT_MOTOR_IN1, LOW);
      digitalWrite(RIGHT_MOTOR_IN2, HIGH);
      digitalWrite(LEFT_MOTOR_IN3, HIGH);
      digitalWrite(LEFT_MOTOR_IN4, LOW);
    } else if (direction == "stop") {
      // Parar todos os motores
      analogWrite(RIGHT_MOTOR_EN, 0);
      analogWrite(LEFT_MOTOR_EN, 0);
      digitalWrite(RIGHT_MOTOR_IN1, LOW);
      digitalWrite(RIGHT_MOTOR_IN2, LOW);
```

```
      digitalWrite(LEFT_MOTOR_IN3, LOW);
      digitalWrite(LEFT_MOTOR_IN4, LOW);
    }
    request->send(200, "text/plain", "OK"); // Envia uma resposta de sucesso
  });


  // Manipula eventos do WebSocket da câmera
  wsCamera.onEvent([](AsyncWebSocket *server, AsyncWebSocketClient *client,
AwsEventType type, void *arg, uint8_t *data, size_t len) {
    switch (type) {
      case WS_EVT_CONNECT:
        cameraClientId = client->id(); // Atualiza o ID do cliente quando
conectado
        break;
      case WS_EVT_DISCONNECT:
        cameraClientId = 0; // Reseta o ID do cliente quando desconectado
        break;
      case WS_EVT_DATA:
        // Não faz nada com os dados recebidos no WebSocket
        break;
      default:
        break;
    }
  });
  server.addHandler(&wsCamera); // Adiciona o manipulador de WebSocket ao
servidor

  server.begin(); // Inicia o servidor
  Serial.println("HTTP server started");

  // Inicializa os servos com a biblioteca ESP32Servo
  servoPan.attach(SERVO_PAN_PIN);  // Anexa o pino do servo Pan
  servoTilt.attach(SERVO_TILT_PIN); // Anexa o pino do servo Tilt
  servoPan.write(90);  // Define o ângulo inicial do servo Pan
  servoTilt.write(90); // Define o ângulo inicial do servo Tilt
}

void loop() {
  wsCamera.cleanupClients(); // Limpa clientes desconectados do WebSocket
  sendCameraPicture();       // Envia uma imagem da câmera para o cliente)
```

Fonte: YOUTUBE CANAL MARLON

2.6 CONCEITOS

Em suma, o robô esteira é uma importante ferramenta de uso de trabalho, tanto da área militar quanto na civil, um bom exemplo da sua execução pode ser dado na área de resgates.

Este é um modelo de robô esteira utilizado em resgates. Sendo essa a função do robô esteira, o auxílio em casos civis, sendo de resgate ou no verificar um local em que se torna de difícil acesso.

Com a capacidade ampla de entrar em lugares confinados, o robô esteira é amplamente visto e utilizado em várias funções não somente em resgates, mas com capacidade transportar medicamentos, equipamentos e outro suprimentos médicos dentro do hospitalares civis e militares. Podendo ter utilidade em casos de desativação de explosivos (bombas, e outros dispositivos), sendo equipado com câmera e braços robótico, manipulando materiais perigosos remotamente, protegendo soldados de explosões.

No reconhecimento de áreas perigosas ou desconhecidas, transmitindo imagens em tempo real para os operadores. Isso permite que as equipes militares identifiquem ameaças, localizem inimigos e planejem forças táticas mais seguras e precisas.

Em ambientes de combates, robôs esteiras podem ser usados para resgatar soldados feridos ou retirar vítimas em áreas de risco. Sendo extremamente útil em locais onde veículos convencionais não conseguem entrar devido ao terreno acidentado

Sendo utilizados também na área da saúde, os robôs esteiras obtêm as seguintes funções a de distribuição de medicamentos e suprimentos médicos, com a função de transportar medicamentos equipamentos e outros suprimentos médicos dentro de hospitais, reduzindo o esforço dos profissionais de saúde e otimizando o tempo.

Na desinfecção de áreas hospitalares, como equipamentos com sistemas de desinfecção, como luz ultravioletas, e podem desinfetar corredores, quartos e salas de cirurgias para realizar desinfecção de forma autônoma.

Na assistência em locais remotos, em áreas de difícil acesso, onde a presença médica é limitada, robôs esteira podem transportar equipamentos médicos para apoiar

procedimentos de telemedicina ou exames remotos, auxiliando profissionais que realizam diagnósticos e atendimentos a distância.

2.7 CONSIDERAÇÕES FINAIS

O presente trabalho apresentou um projeto mecatrônico que, embora tenha alcançado seus objetivos iniciais, demonstra um grande potencial para futuras melhorias. A plataforma desenvolvida se mostra aberta a diversas modificações, tanto no âmbito da programação quanto das componentes mecânicas e elétricas. A adição de novos sensores, como os de presença e ultrassônicos, pode ampliar significativamente a capacidade de interação do sistema com o ambiente. Além disso, a implementação de novas funcionalidades, como o controle de iluminação adaptativa ou a integração com outros dispositivos, pode enriquecer a aplicação e abrir caminho para novas áreas de pesquisa.

3. CONCLUSÃO

Em resumo, o robô esteira tem se mostrado uma ferramenta essencial tanto em operações militares quanto na área da saúde, onde sua aplicação varia de: desativação de explosivos e resgate em zonas de riscos até a distribuição de medicamentos e desinfecção de ambientes hospitalares.

Sua capacidade de operar em terrenos difíceis, realizar tarefas autônomas e reduzir a exposição humana a perigos, os tornam aliados valiosos na proteção e otimização de recursos humanos.

Esses robôs provêm segurança, eficiência e inovação, cumprindo funções críticas que melhoram a qualidade de operações militares e atendimentos médicos, além de aumentar a segurança e reduzir custos em setores de alto risco.

REFERÊNCIAS

- **Página Web (YOUTUBE):**
  https://youtu.be/E4oNbqolud0?si=18jRWRLzvbJ9NJrE
  https://youtu.be/PlqHS1CT17Q?si=JJ_kWQLJkiPvJDYn
- **Sites:**
  Arduíno **https://www.arduino.cc/**
  ThinkerCad **www.tinkercad.com**