**RELATÓRIO TÉCNICO: PROJETO INTEGRADOR – ROBÔ SUMÔ AUTÔNOMO**

Unidades Curriculares: Lógica de Programação + Fundamentos de Eletroeletrônica Aplicada

Instituição: SENAI

Professores: Ivanildo Gomes da Silva e Ícaro Vasconcelos Alvim

## 1. Identificação da Equipe

**Integrantes:**

- LUCAS AUGUSTO
- JUAN DA SILVA
- EUZEBIO ROCHA
- MARCOS VINICIUS
- RODRIGO MIRANDA

## 2. Objetivos do Projeto

O objetivo central deste projeto foi projetar e construir um robô autônomo competitivo para a modalidade "Sumô de Robôs". O projeto visou integrar as competências de eletrônica (montagem de circuitos, sensores e atuadores) com lógica de programação (algoritmos de decisão e controle), estimulando o trabalho em equipe e a capacidade de resolução de problemas técnicos sob pressão de prazos definidos.

## 3. Descrição Técnica e Funcionamento

O robô desenvolvido é um veículo autônomo com tração diferencial, projetado para localizar e empurrar um oponente para fora de uma arena circular preta com borda branca.

O sistema opera em "loop fechado", onde o microcontrolador toma decisões constantes baseadas em dois grupos de variáveis:

1. Detecção de Borda (Sobrevivência): Sensores infravermelhos voltados para o chão identificam a linha branca para evitar quedas e consequente derrota no round.
2. Detecção de Oponente (Ataque): Sensores ultrassônicos mapeiam o ambiente para localizar o adversário.

## 4. Hardware e Componentes Eletrônicos

A escolha dos componentes visou robustez e capacidade de processamento. A equipe optou por um upgrade técnico em relação ao kit básico sugerido, utilizando a permissão de customização do edital:

- Microcontrolador: Arduino Mega 2560.
    - *Justificativa:* Optamos pelo Mega devido ao maior número de pinos digitais e analógicos disponíveis, facilitando a conexão de múltiplos sensores (3 ultrassônicos + 3 de linha) sem necessidade de multiplexação e garantindo expansibilidade.
- Driver de Potência: Ponte H L298N (Controle bidirecional dos motores DC).
- Atuadores: 2 Motores DC 5V com caixa de redução (Torque para empurrar).
- Sensores de Distância: 3x HC-SR04 (Posicionados na Frente, Esquerda e Direita para cobertura ampla).
- Sensores de Linha: 3x Sensores Infravermelhos Analógicos.
- Alimentação: Duas baterias 9V alcalina.
- Estrutura: Chassi impresso em 3D (PLA), respeitando o limite de peso de 1 kg.

## 5. Esquemático e Conexões (Pinagem)

A integração entre hardware e software foi realizada conforme o mapeamento de pinos abaixo, verificado no código fonte e no esquemático elétrico:

| Componente | Função | Pino Arduino (Mega) |
|---|---|---|
| Motor Esquerdo | Controle Rotação | Pinos 2 e 3 |
| Motor Direito | Controle Rotação | Pinos 4 e 5 |
| Ultrassônico Frente | Trigger / Echo | 8 / 9 |
| Ultrassônico Esq. | Trigger / Echo | 10 / 11 |
| Ultrassônico Dir. | Trigger / Echo | 6 / 7 |
| Sensor Linha Dir. | Leitura Analógica | A0 |
| Sensor Linha Esq. | Leitura Analógica | A1 |
| Sensor Linha Trás | Leitura Analógica | A2 |

## 6. Lógica de Programação e Estratégia

O software foi desenvolvido em C++, utilizando uma estrutura de decisão hierárquica (Prioridade de Eventos). O algoritmo garante que a segurança do robô (não cair da arena) sempre sobreponha a vontade de atacar.

Foi implementada uma rotina de segurança no início (setup) com um atraso de 5 segundos, atendendo às normas oficiais da competição para permitir o afastamento seguro dos juízes e operadores antes do início do combate.

### 6.1. Pseudocódigo da Estratégia

1. Inicialização: Aguarda 5 segundos (delay(5000)).
2. Leitura dos Sensores:
    - Sensores de linha: Calibrados para detectar a borda branca com valores abaixo de 400.
    - Sensores ultrassônicos: Leitura com *timeout* de segurança (retorna 100cm se falhar) para evitar travamento do processador.
3. Verificação de Borda (Prioridade Máxima):
    - *SE* (Qualquer sensor de linha ler < 400): O robô para brevemente (parar()) para evitar inércia, e executa a manobra de retorno contrária ao sensor ativado.
4. Verificação de Combate (Prioridade Média):
    - *SE* (Inimigo à Frente < 60cm): ATACAR (Avança em direção ao oponente).
    - *SE* (Inimigo na Lateral < 60cm): Girar para o lado do inimigo.
5. Modo de Busca (Prioridade Baixa/Default):
    - *SE* (Nada detectado): Girar no próprio eixo (Direita) para escanear a arena.

### 6.2. Trecho Crítico do Código

A calibração dos sensores de linha foi ajustada para identificar a superfície preta (segura) com valores altos e a borda branca com valores baixos. O código inclui proteção contra leituras falsas.

C++

```
// Lógica de combate: SÓ EXECUTA SE ESTIVER SEGURO (Preto > 400)
if (valorEsq > 400 && valorDir > 400 && valorTra > 400) {
   if (distanciaFrente < 60) {
      frente(); // Ataque estendido para 60cm
```

```
    }
    // ... lógica de giro ...
}
// Lógica de Segurança (Borda Branca < 400)
else {
    if (valorEsq < 400) { // Borda detectada na esquerda
        parar();    // Estabiliza
        delay(50);
        direita();  // Foge da borda
        delay(300);
    }
    // ... lógica para os outros sensores ...
}
```

## 7. Montagem, Testes e Resolução de Problemas

Durante a fase de integração Hardware + Software, a equipe enfrentou desafios reais de engenharia que foram solucionados para garantir a operacionalidade do robô.

### 7.1. Desafio do Motor e Manutenção

Durante os testes iniciais de movimentação, identificamos que o robô não realizava curvas corretamente, girando de forma errática.

- Diagnóstico: Após testar cada um dos componentes, incluindo motores, e ver que estavam funcionando, concluímos que o problema provavelmente estaria na Ponte H.
- Solução: Foi realizada a substituição imediata da Ponte H e ressoldagem dos materiais. Após a troca, o robô voltou a ter seus movimentos de tração diferencial.

### 7.2. Organização e Calibração

Devido ao uso do Arduino Mega e múltiplos sensores, o volume de cabeamento foi alto. Utilizamos abraçadeiras para organizar os jumpers e evitar que fios soltos tocassem nas rodas.

Durante os testes finais, o código foi otimizado para incluir uma função de parar() antes das manobras de evasão. Isso foi necessário pois a inércia dos motores fazia o robô deslizar sobre a borda mesmo após detectá-la. Com a pausa de 50ms, a tração foi recuperada, garantindo que o robô permaneça na arena.

Também ajustamos o alcance de ataque para 60cm, permitindo que o robô detecte o oponente de uma distância maior e inicie a ofensiva mais cedo.

## 8. Conclusão

O projeto foi concluído com êxito. O robô cumpre todos os requisitos do edital: respeita o peso máximo, opera de forma totalmente autônoma e possui uma estratégia clara de combate e sobrevivência. A falha do motor serviu como aprendizado prático sobre diagnóstico de defeitos e manutenção rápida, fortalecendo a competência técnica da equipe e garantindo a prontidão para a competição.