AULA 3
Sensores Laterais e Estratégia de Trajetos

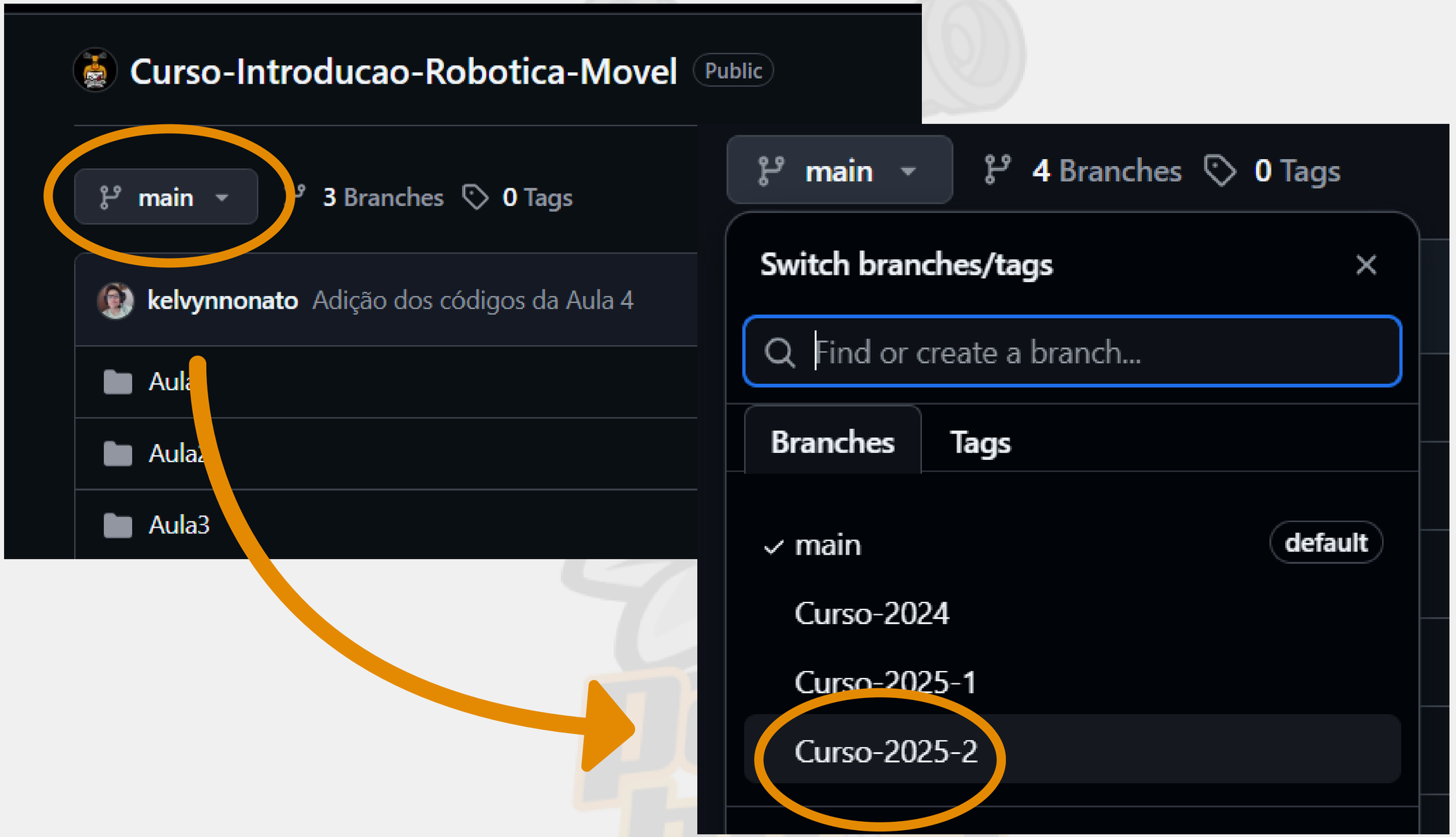
Patobots - UTFPR

Cronograma:

- Introdução das regras das faixas laterais.
- Teoria da estratégia de trajetos no Seguidor de Linha.
- Implementação dos sensores laterais no código.
- Completar uma volta na pista no menor tempo.

GITHUB:

https://github.com/PatoBots/Curso-Introducao-Robotica-Movel/tree/Curso-2025-2


Curso-Introducao-Robotica-Movel
Public
main
3 Branches
0 Tags
kelvynnonato Adição dos códigos da Aula 4
Aula
Aula2
Aula3
main
4 Branches
0 Tags
Switch branches/tags
Find or create a branch...
Branches
Tags
main
default
Curso-2024
Curso-2025-1
Curso-2025-2

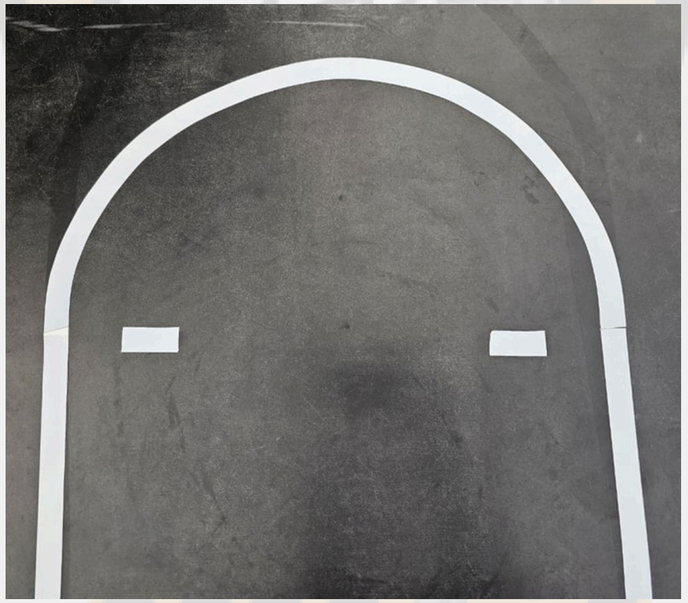
FAIXAS LATERAIS

Faixas laterais

- **Faixa lateral direita:** Serve para indicar o início e fim da pista.

Faixas laterais

- **Faixa lateral esquerda:** Serve para indicar o início e fim de uma curva.



Faixas laterais

- **Cruzamentos:** Também contam como faixas laterais, mas indicam uma interseção de linhas na pista.
- O Seguidor de linha identifica o cruzamento como uma faixa lateral esquerda e direita, ele deve ser levado em conta na contagem final das faixas laterais da pista.

ESTRATÉGIA DE TRAJETOS

Estratégia de Trajetos

- Utilizaremos essa estratégia para definir a velocidade do robô em cada trajeto da pista, podendo aumentar sua velocidade em retas e diminuir em curvas.

- A cada vez que uma faixa esquerda for identificada, iremos alterar ou manter a velocidade do robô.

- É ideal para mantermos um controle total durante todo o percurso e garantir o melhor tempo possível.

Estratégia de Trajetos

- **Como funciona:** Definimos um vetor com valores de velocidade no código. A cada faixa esquerda detectada, incrementamos um contador que vai acessar a velocidade no vetor em seu respectivo trajeto.
- **O que é um trajeto:** Entre cada faixa esquerda existe um trajeto, sendo uma reta ou uma curva. Ao identificar cada trajeto, podemos definir uma velocidade ideal para ele.

Estratégia de Trajetos

- Também é possível controlar o Kp e o Kd por cada trajeto, porém torna a estratégia mais complexa e demorada de ser aplicada.
- É um excelente desafio para Seguidores de Linha que não dependem de Sucção ou Ventoinhas.

CÓDIGO DO SEGUIDOR COM OS SENSORES LATERAIS

```
1   // Robô Seguidor de Linha Repouso com Controle Proporcional e Derivativo - Arduino Nano
2   // Sensores: A1-A6 (6 sensores analógicos)
3   // Botão: D12 (pull-down)
4   // Motor Driver TB6612FNG: PWMA(D5), AI1(D9), AI2(D4), PWMB(D6), BI1(D7), BI2(D8)
5
6   // Definições dos pinos
7   #define NUM_SEN 6
8   #define BUTTON_PIN 12
9
10  // Pinos do motor driver TB6612FNG
11  #define PWMA 5   // PWM Motor A (esquerda)
12  #define AI1 9    // Direção Motor A
13  #define AI2 4    // Direção Motor A
14  #define PWMB 6   // PWM Motor B (direita)
15  #define BI1 7    // Direção Motor B
16  #define BI2 8    // Direção Motor B
17
18  // Pinos dos sensores frontais (A1 a A6)
19  int pinoSensores[NUM_SEN] = {A1, A2, A3, A4, A5, A6};
20
21  // Pinos dos sensores lateriais (A0 e A7)
22  int sensorDir = A0;
23  int sensorEsq = A7;
24
```

```
25  // Variáveis para calibração
26  int minSensor[NUM_SEN];
27  int maxSensor[NUM_SEN];
28  int valorSensores[NUM_SEN];
29
30  int minDir;
31  int maxDir;
32  int valorDir;
33  int N_Dir = 0;
34
35  int minEsq;
36  int maxEsq;
37  int valorEsq;
38  int N_Esq = 0;
39
40  // Variáveis para controle PID
41  float Kp = 0.0;    // Constante proporcional
42  float Kd = 0.0;    // Constante diferencial
43  float erro = 0;
44  float erroAnterior = 0;
45
46  // Velocidades dos motores
47  int velBase = 40;     // Velocidade base (0-255)
48  int velMax = 255;      // Velocidade máxima
49  int velMin = 0;        // Velocidade mínima
50
```

```
51  //Variáveis de Tempo
52  unsigned long ultimoTempoDir = 0;
53  const unsigned long intervaloDir = 50; // 50 milissegundos
54
55  unsigned long ultimoTempoEsq = 0;
56  const unsigned long intervaloEsq = 100; // 100 milissegundos
57
58  // Definir a velocidade pelos trajetos
59  #define N_FAIXA 0 // Coloque a quantidade de faixas da esquerda + 1
60  int velTrajeto[N_FAIXA] = {}; // Coloque as velocidades para cada um dos trajetos
61
```

```
90  // Inicializar valores de calibração
91  for (int i = 0; i < NUM_SEN; i++) {
92    minSensor[i] = 1023;
93    maxSensor[i] = 0;
94  }
95
96  minEsq = 1023;
97  maxEsq = 0;
98  minDir = 1023;
99  maxDir = 0;
100
```

```
133     case CORRENDO:
134       if (botaoAtual) {
135         estadoAtual = ESPERANDO;
136         botaoAtual = false;
137         N_Dir = 0;
138         N_Esq = 0;
139         pararMotores();
140         Serial.println("Parando robô. Pressione o botão novamente para recalibrar.");
141       } else {
142         seguirLinha();
143         Serial.println(N_Dir);
144         Serial.println(N_Esq);
145         if(N_Dir == 4){
146           botaoAtual = true;
147         }
148       }
149       break;
```

```
163     // Ler sensores e atualizar valores min/max
164     for (int i = 0; i < NUM_SEN; i++) {
165       int faixa = analogRead(pinoSensores[i]);
166       if (faixa < minSensor[i]) {
167         minSensor[i] = faixa;
168       }
169       if (faixa > maxSensor[i]) {
170         maxSensor[i] = faixa;
171       }
172     }
173
174     int faixaEsq = analogRead(sensorEsq);
175     if (faixaEsq < minEsq) minEsq = faixaEsq;
176     if (faixaEsq > maxEsq) maxEsq = faixaEsq;
177
178     int faixaDir = analogRead(sensorDir);
179     if (faixaDir < minDir) minDir = faixaDir;
180     if (faixaDir > maxDir) maxDir = faixaDir;
181
182     delay(10);
```

```
241  void seguirLinha() {
242    // Ler e normalizar sensores
243    lerSensores();
244    lerSensorLateral(sensorEsq);
245    lerSensorLateral(sensorDir);
246
247    unsigned long agora = millis();
248
249    Serial.println(valorDir);
250    if (valorDir < 230 && (agora - ultimoTempoDir >= intervaloDir)) {
251      N_Dir++;
252      ultimoTempoDir = agora; // Atualiza o tempo da última contagem
253    }
254
255    Serial.println(valorEsq);
256    if(valorEsq < 230 && (agora - ultimoTempoEsq >= intervaloEsq)){
257      N_Esq++;
258      ultimoTempoEsq = agora; // Atualiza o tempo da ultima contagem
259    }
260
```

```
267     // Controle PID
268     float derivativo = erro - erroAnterior;
269     float pid = Kp * erro + Kd * derivativo;
270     erroAnterior = erro;
271
272     // Calcular velocidades dos motores
273     int velEsquerda = velTrajeto[N_Esq] + pid;
274     int velDireita = velTrajeto[N_Esq] - pid;
275
276     // Limitar velocidades
277     velEsquerda = constrain(velEsquerda, velMin, velMax);
278     velDireita = constrain(velDireita, velMin, velMax);
279
280     // Aplicar velocidades aos motores
281     acelerar(velEsquerda, velDireita);
282
```

```
307  void lerSensorLateral(int sensorLateral){
308    int leitura = analogRead(sensorLateral);
309
310    // Determinar qual sensor está sendo lido para usar calibração correta
311    if (sensorLateral == sensorEsq) {
312      // Normalizar entre 0 e 1000
313      valorEsq = map(leitura, minEsq, maxEsq, 0, 1000);
314      valorEsq = constrain(valorEsq, 0, 1000);
315    } else {
316      // Normalizar entre 0 e 1000
317      valorDir = map(leitura, minDir, maxDir, 0, 1000);
318      valorDir = constrain(valorDir, 0, 1000);
319    }
320  }
```