AULA 2
Desenvolvendo a Lógica PID

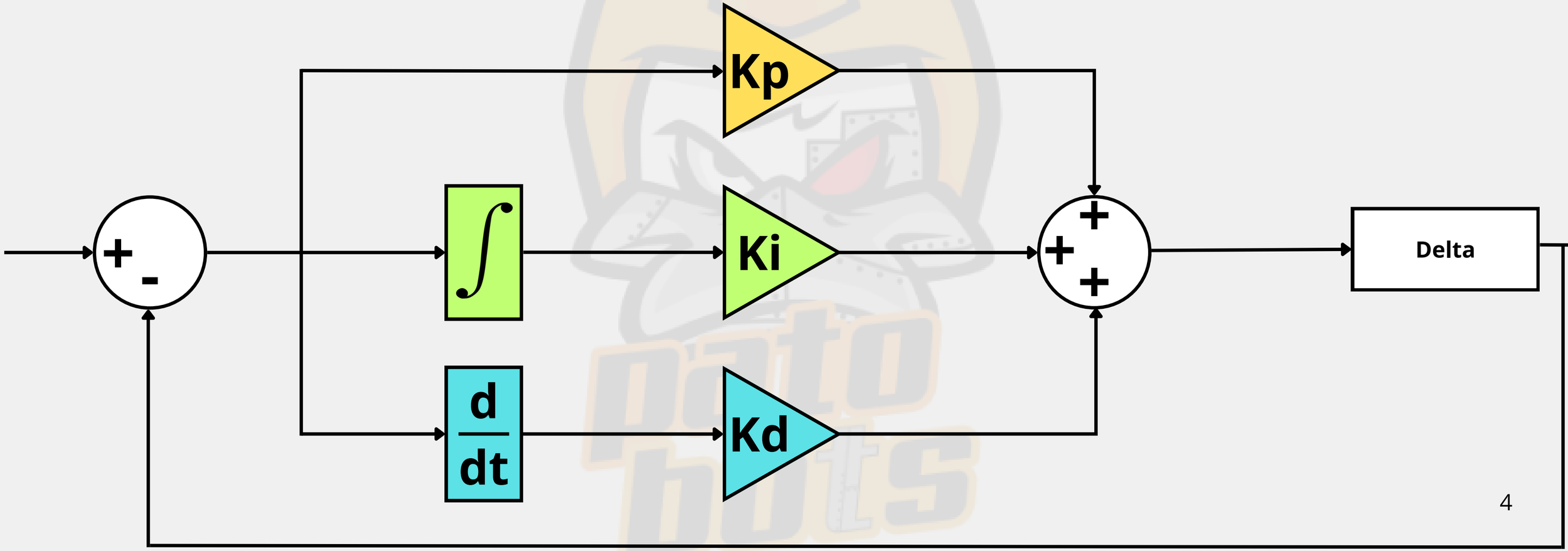
Patobots - UTFPR

Cronograma:

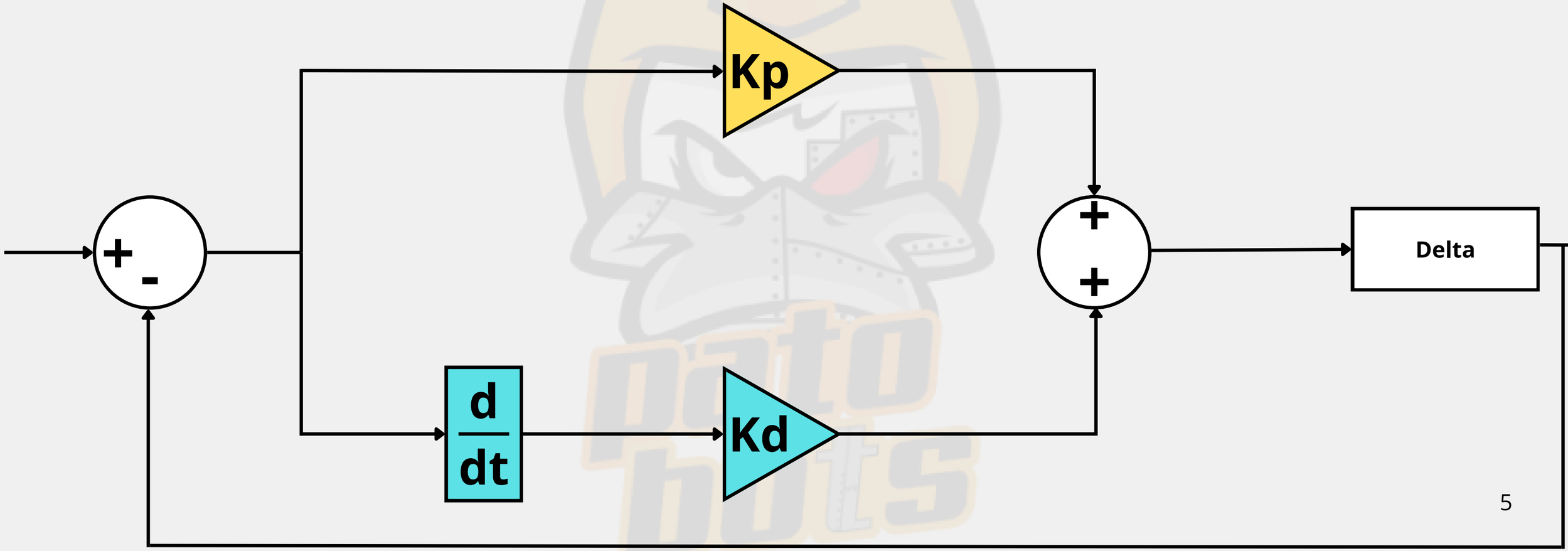
- Controle PID (Proporcional e Taxa de variação).
- Teste com apenas o Controle Proporcional.
- Teste com Controle Derivativo e Proporcional.
- Desafio.

CONTROLE PID

Controle PID
(Proporcional, Integral e Derivativo)



Controle PID
(Proporcional, Integral e Derivativo)



Controle PID

Controle PID para Robôs:

- **Estabilização:** O Controle ajuda o robô a manter equilíbrio e seguir o caminho correto.
- **Seguidor de Linha:** Ideal para robôs que precisam seguir trajetórias marcadas no chão.

Controle PID

Fórmula de Controle P e D:

Ajusta a movimentação do robô com base no erro de posição:

- **P (Proporcional):** Reage ao erro atual.
- **D (Derivativo):** Considera a variação do erro para corrigir a velocidade.

- **Fórmula:**

  Delta = Kp * Erro Atual + Kd * (Erro Atual - Erro Anterior)

CÓDIGO DE CONTROLE DO SEGUIDOR (Proporcional)

```
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 (A1 a A6)
19  int pinoSensores[NUM_SEN] = {A1, A2, A3, A4, A5, A6};
20
21  // Variáveis para calibração
22  int minSensor[NUM_SEN];
23  int maxSensor[NUM_SEN];
24  int valorSensores[NUM_SEN];
25
26  // Variáveis para controle PID
27  float Kp = 0.0;    // Constante proporcional
28  float erro = 0;
```

```
30  // Velocidades dos motores
31  int velBase = 40;      // Velocidade base (0-255)
32  int velMax = 255;       // Velocidade máxima
33  int velMin = 0;         // Velocidade mínima
34
35  // Estados do programa
36  enum Estado {
37    ESPERANDO,
38    CALIBRANDO,
39    CORRENDO
40  };
41
42  Estado estadoAtual = ESPERANDO;
43  bool botaoAtual = false;
44  bool botaoEstadoAnteior = false;
45
```

```
48  void setup() {
49    Serial.begin(9600);
50
51    // Configuração dos pinos
52    pinMode(BUTTON_PIN, INPUT);
53
54    // Configuração dos pinos do motor
55    pinMode(PWMA, OUTPUT);
56    pinMode(AI1, OUTPUT);
57    pinMode(AI2, OUTPUT);
58    pinMode(PWMB, OUTPUT);
59    pinMode(BI1, OUTPUT);
60    pinMode(BI2, OUTPUT);
61
62    // Parar motores inicialmente
63    pararMotores();
64
65    // Inicializar valores de calibração
66    for (int i = 0; i < NUM_SEN; i++) {
67      minSensor[i] = 1023;
68      maxSensor[i] = 0;
69    }
70
71    Serial.println("Sistema iniciado. Pressione o botão para calibrar.");
72  }
```

```
75  void loop() {
76    // Leitura do botão
77    bool estadoBotao = digitalRead(BUTTON_PIN);
78
79    // Detecção de borda de subida do botão
80    if (estadoBotao && !botaoEstadoAnteior) {
81      botaoAtual = true;
82      delay(50); // Debounce
83    }
84    botaoEstadoAnteior = estadoBotao;
```

```
86    // Máquina de estados
87    switch (estadoAtual) {
88      case ESPERANDO:
89        if (botaoAtual) {
90          estadoAtual = CALIBRANDO;
91          botaoAtual = false;
92          Serial.println("Iniciando calibração...");
93          delay(500);
94          calibrarSensores();
95        }
96        break;
```

```
 98        case CALIBRANDO:
 99          estadoAtual = CORRENDO;
100          Serial.println("Calibração concluída. Iniciando corrida!");
101          delay(3000); // Depois de 3s inicia a corrida
102          break;
103
104        case CORRENDO:
105          if (botaoAtual) {
106            estadoAtual = ESPERANDO;
107            botaoAtual = false;
108            pararMotores();
109            Serial.println("Parando robô. Pressione o botão novamente para r
110          } else {
111            seguirLinha();
112          }
113          break;
114      }
115  }
```

```
114  void calibrarSensores() {
115    Serial.println("Movendo robô para calibração...");
116
117    // Calibração por 3 segundos girando o robô
118    unsigned long tempoInicial = millis();
119
120    while (millis() - tempoInicial < 3000) {
121      // Girar o robô lentamente para a direita
122      //acelerar(50, -50);
123
124      // Ler sensores e atualizar valores min/max
125      for (int i = 0; i < NUM_SEN; i++) {
126        int faixa = analogRead(pinoSensores[i]);
127        if (faixa < minSensor[i]) {
128          minSensor[i] = faixa;
129        }
130        if (faixa > maxSensor[i]) {
131          maxSensor[i] = faixa;
132        }
133      }
134      delay(10);
135    }
```

```
137   // Girar para o outro lado
138   tempoInicial = millis();
139   while (millis() - tempoInicial < 3000) {
140     // Girar o robô lentamente para a esquerda
141     //acelerar(-50, 50);
142
143     // Ler sensores e atualizar valores min/max
144     for (int i = 0; i < NUM_SEN; i++) {
145       int faixa = analogRead(pinoSensores[i]);
146       if (faixa < minSensor[i]) {
147         minSensor[i] = faixa;
148       }
149       if (faixa > maxSensor[i]) {
150         maxSensor[i] = faixa;
151       }
152     }
153     delay(10);
154   }
155
156   pararMotores();
```

```
172  void seguirLinha() {
173    // Ler e normalizar sensores
174    lerSensores();
175
176    // Calcular posição da linha
177    float posicao = controleLinha();
178
179    // Calcular erro (0 = centro, negativo = esquerda, positivo = direita)
180    erro = posicao - 2.5; // Centro dos 6 sensores (0-5)
181
182    // Controle PID
183    float pid = Kp * erro;
184
185    // Calcular velocidades dos motores
186    int velEsquerda = pid;
187    int velDireita = -pid;
188
189    // Limitar velocidades
190    velEsquerda = constrain(velEsquerda, velMin, velMax);
191    velDireita = constrain(velDireita, velMin, velMax);
192
193    // Aplicar velocidades aos motores
194    acelerar(velEsquerda, velDireita);
```

```
95  void lerSensores() {
96    for (int i = 0; i < NUM_SEN; i++) {
97      int leitura = analogRead(pinoSensores[i]);
98      // Normalizar entre 0 e 1000
99      valorSensores[i] = map(leitura, minSensor[i], maxSensor[i], 0, 1000);
100     valorSensores[i] = constrain(valorSensores[i], 0, 1000);
101   }
102 }
```

```
220  float controleLinha() {
221    long num = 0;
222    long den = 0;
223
224    for (int i = 0; i < NUM_SEN; i++) {
225      num += (long)valorSensores[i] * i * 1000;
226      den += valorSensores[i];
227    }
228
229    if (den == 0) {
230      return 2.5; // Retorna centro se nenhum sensor detectar linha
231    }
232
233    return (float)num / den / 1000.0;
234  }
```

```
120  void acelerar(int velEsquerda, int velDireita) {
121    // Motor esquerdo (canal A)
122    if (velEsquerda >= 0) {
123      digitalWrite(AI1, HIGH);
124      digitalWrite(AI2, LOW);
125      analogWrite(PWMA, velEsquerda);
126    } else {
127      digitalWrite(AI1, LOW);
128      digitalWrite(AI2, HIGH);
129      analogWrite(PWMA, -velEsquerda);
130    }
131
132    // Motor direito (canal B)
133    if (velDireita >= 0) {
134      digitalWrite(BI1, HIGH);
135      digitalWrite(BI2, LOW);
136      analogWrite(PWMB, velDireita);
137    } else {
138      digitalWrite(BI1, LOW);
139      digitalWrite(BI2, HIGH);
140      analogWrite(PWMB, -velDireita);
141    }
142  }
```

```
144  void pararMotores() {
145    analogWrite(PWMA, 0);
146    analogWrite(PWMB, 0);
147    digitalWrite(AI1, LOW);
148    digitalWrite(AI2, LOW);
149    digitalWrite(BI1, LOW);
150    digitalWrite(BI2, LOW);
151  }
```

CÓDIGO DE CONTROLE DO SEGUIDOR (Derivativo)

```
26  // Variáveis para controle PID
27  float Kp = 0.0;     // Constante proporcional
28  float Kd = 0.0;     // Constante diferencial
29  float erro = 0;
30  float erroAnterior = 0;
31
```

```
174  void seguirLinha() {
175    // Ler e normalizar sensores
176    lerSensores();
177
178    // Calcular posição da linha
179    float posicao = controleLinha();
180
181    // Calcular erro (0 = centro, negativo = esquerda, positivo = direita)
182    erro = posicao - 2.5; // Centro dos 6 sensores (0-5)
183
184    // Controle PID
185    float derivativo = erro - erroAnterior;
186    float pid = Kp * erro + Kd * derivativo;
187    erroAnterior = erro;
188
189    // Calcular velocidades dos motores
190    int velEsquerda = pid;
191    int velDireita = -pid;
192
193    // Limitar velocidades
194    velEsquerda = constrain(velEsquerda, velMin, velMax);
195    velDireita = constrain(velDireita, velMin, velMax);
196
197    // Aplicar velocidades aos motores
198    acelerar(velEsquerda, velDireita);
```

DESAFIO
Andar com o robô na pista