Controle PID com ESP32 – Simulação no Wokwi

Autor: Labelle Candido

Data: Outubro de 2025

1. Introdução

Este projeto tem como objetivo demonstrar o funcionamento de um controlador PID (Proporcional, Integral e Derivativo) utilizando a placa ESP32. O controle PID é amplamente empregado em sistemas automáticos para corrigir erros entre um valor desejado (setpoint) e o valor medido (feedback), ajustando continuamente a saída do sistema até atingir a estabilidade.

2. Simulação

A simulação foi desenvolvida na plataforma Wokwi, utilizando um potenciômetro como sensor de posição. Dois LEDs (verde e vermelho) indicam o estado do sistema, mostrando quando o sistema está estável ou com erro elevado.

2.1 Componentes Utilizados (presentes no Wokwi)

- 1 × ESP32 — microcontrolador principal
- 1 × Potenciômetro (10kΩ) — simula o sensor de posição
- 2 × LEDs (vermelho e verde) — indicadores de estabilidade
- 2 × Resistores de 220Ω — limitam a corrente dos LEDs

3. Conexões do Circuito

- Potenciômetro VCC → 3V3 do ESP32
- Potenciômetro GND → GND do ESP32
- Potenciômetro Sinal → Pino 32
- LED Verde → Pino 2 (com resistor de 220Ω)
- LED Vermelho → Pino 4 (com resistor de 220Ω)
- LEDs → GND

4. Princípio de Funcionamento do PID

O controlador PID ajusta a saída de um sistema com base em três componentes matemáticos:

- Proporcional (Kp × erro): reage diretamente ao erro atual.
- Integral (Ki × soma dos erros): corrige desvios acumulados ao longo do tempo.
- Derivativo (Kd × variação do erro): reage à velocidade de mudança do erro.

A soma dessas três parcelas gera uma resposta equilibrada e adaptativa, permitindo ao sistema alcançar o setpoint de forma estável e eficiente.

5. Lógica do Projeto

1. O ESP32 lê o valor analógico do potenciômetro (0–4095).
2. Define-se o setpoint como 2048 (posição central).
3. Calcula-se o erro, o integral e o derivativo.
4. O valor PID é calculado pela soma ponderada das três parcelas.
5. O sistema acende o LED verde quando o erro é pequeno e o vermelho quando o erro é alto.

6. Código Utilizado

```
// ===== Controle PID Simples com ESP32 e Potenciômetro =====
#define POT_PIN 32
#define LED_OK 2
#define LED_FAIL 4

float Kp = 3.5;
float Ki = 0.6;
float Kd = 0.8;

float setpoint = 2048;  // Valor ideal (posição central)
float erro, erro_anterior = 0;
float integral = 0;
float derivativo = 0;
float saida = 0;
int leitura;

void setup() {
  Serial.begin(115200);
  pinMode(LED_OK, OUTPUT);
  pinMode(LED_FAIL, OUTPUT);
  Serial.println("=== Simulação PID no Wokwi ===");
}

void loop() {
  leitura = analogRead(POT_PIN);
  erro = setpoint - leitura;
```

```
  // Cálculo do PID
  integral += erro;
  derivativo = erro - erro_anterior;
  saida = (Kp * erro) + (Ki * integral) + (Kd * derivativo);

  erro_anterior = erro;

  // LEDs de estado
  if (abs(erro) < 100) { // Está estável
    digitalWrite(LED_OK, HIGH);
    digitalWrite(LED_FAIL, LOW);
  } else {
    digitalWrite(LED_OK, LOW);
    digitalWrite(LED_FAIL, HIGH);
  }

  // Exibição serial
  Serial.print("Leitura: ");
  Serial.print(leitura);
  Serial.print(" | Erro: ");
  Serial.print(erro);
  Serial.print(" | Saída PID: ");
  Serial.println(saida);

  delay(100);
}
```

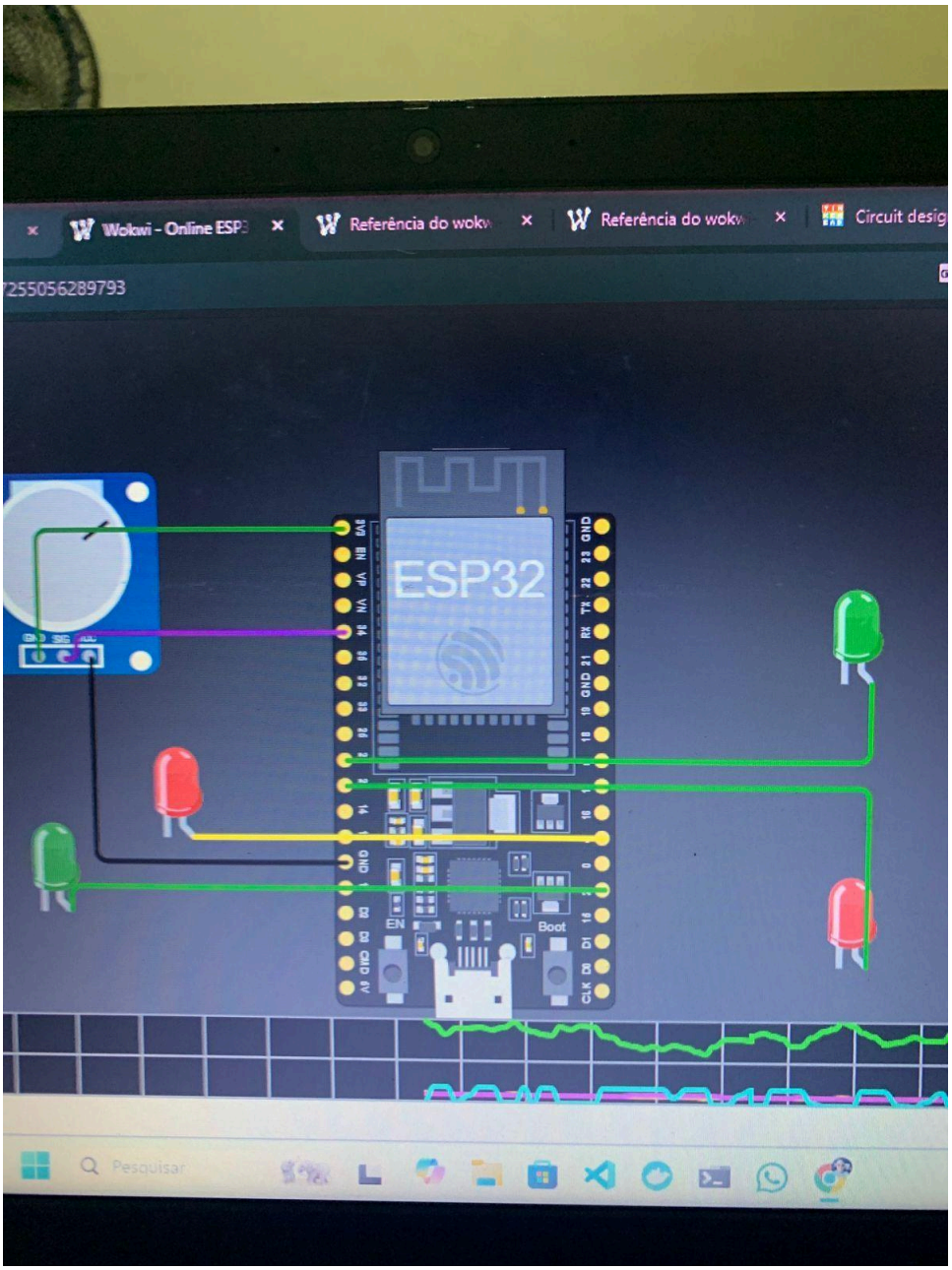
7. Resultados Observados

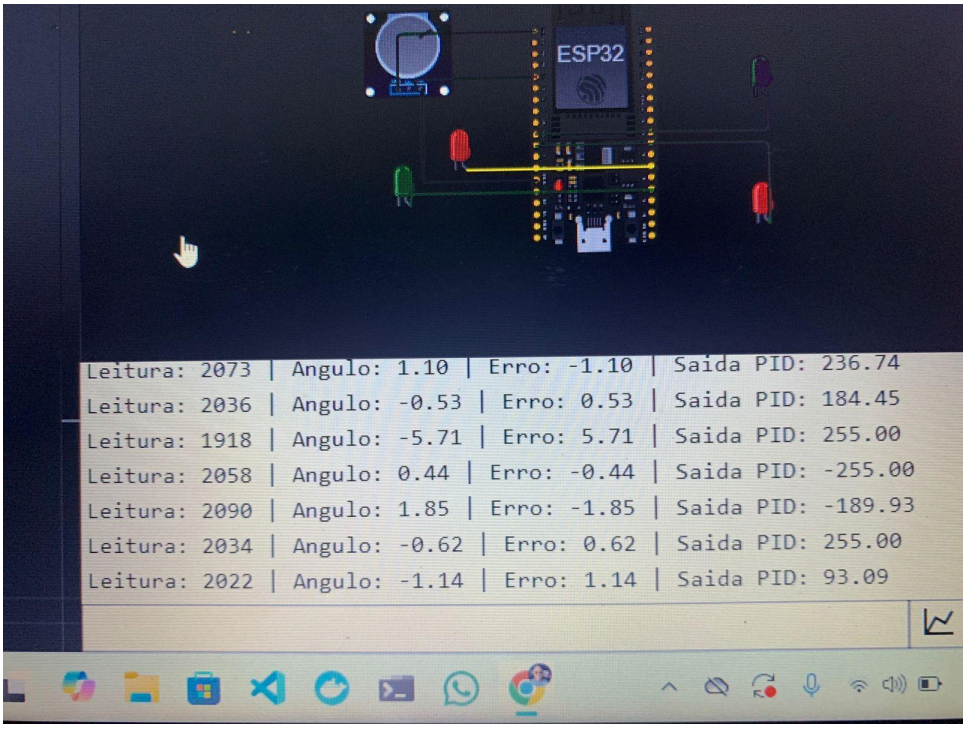
Durante a simulação no Wokwi, foram observados os seguintes resultados:

Leitura	Erro	Saída PID	Observação
2073	-1.1	236.74	LED verde aceso (estável)
2036	0.53	184.45	LED verde aceso

1918	5.71	255.0	LED vermelho aceso (erro alto)
2058	-0.44	-255.0	LED vermelho aceso
2090	-1.85	-189.93	LED vermelho aceso
2034	0.62	255.0	LED verde aceso
2022	1.14	93.09	LED verde aceso (quase estável)

As imagens a seguir mostram o circuito montado no simulador e o monitor serial exibindo os valores de leitura, erro e saída PID.


Wokwi - Online ESP
Referência do wokw
Referência do wokw
Circuit desig
7255056289793
ESP32
EN
Boot
Pesquisar



8. Conclusão

O projeto demonstrou de forma prática o funcionamento de um controlador PID, mostrando como o sistema reage ao erro e busca a estabilidade automaticamente. A variação dos valores do potenciômetro provoca mudanças no erro, e o algoritmo PID calcula a resposta de forma dinâmica. Esse tipo de controle pode ser aplicado em sistemas reais, como motores, robôs e controle de temperatura, entre outros.

9. Possíveis Extensões

- Adicionar servo motor para representar movimento físico real.
- Exibir gráficos em tempo real de erro e saída PID.
- Criar interface web com o ESP32 para visualização e ajuste dos parâmetros Kp, Ki e Kd.