Sensor de Velocidade com Controle de RPM no Arduino

Objetivo: Este projeto simula o controle de RPM de um motor utilizando um LED como indicador de esforço. O sistema incorpora perturbações externas via potenciômetro e sensor touch, e aplica controle proporcional para manter o RPM próximo de um valor alvo

Componentes Utilizados: Arduino Uno, LED, Sensor Touch, Potenciometro, Resistor, Capacitor, Stepper Motor, Sensor DHT11, Push Button.

Esquema do Circuito:

O circuito está montado conforme a imagem anexada. A conexão dos principais componentes é:

- **LED** conectado ao pino **D5** do Arduino (PWM).
- **Sensor Touch** conectado ao pino **D4** (digital).
- **Potenciômetro** conectado ao pino **A0** (analógico).
- **Resistor** em série com o LED para limitar corrente.
- **Capacitor** próximo ao potenciômetro para filtragem de ruído.
- **Sensor DHT11**, botão e motor de passo estão presentes fisicamente, mas não são utilizados neste código.

Setup:

```
pinMode(PINO_PWM, OUTPUT);

pinMode(PINO_TOUCH, INPUT);

Serial.begin(9600);
```

Define os pinos do LED e sensor touch e inicia a comunicação serial para monitoramento.

 Loop principal

A cada 1000 ms (`DELTA_PULSO`), o sistema realiza:

1. **Leitura do Potenciômetro**:
   - Mapeia o valor de 0–1023 para uma perturbação de -30% a +30%.
2. **Leitura do Sensor Touch**:
   - Se ativado, aplica uma perturbação extrema de -50%.

Define os pinos do LED e sensor touch.

- Inicia a comunicação serial para monitoramento.

Loop principal

A cada 1000 ms (`DELTA_PULSO`), o sistema realiza:

1. **Leitura do Potenciômetro**:
    - Mapeia o valor de 0–1023 para uma perturbação de -30% a +30%.
2. **Leitura do Sensor Touch**:
    - Se ativado, aplica uma perturbação extrema de -50%.
3. **Simulação de RPM**:

    - Calcula o RPM como função do PWM e da perturbação:

```
rpm_simulado = pwm * FATOR_RPM * (1.0 - P / 100.0);
```

4. **Controle Proporcional (P)**:

    - Calcula o erro entre o RPM alvo e o simulado.
    - Aplica ajuste proporcional com ganho `KP`.

5. **Impacto da Perturbação**:

    - Aplica um impacto fixo baseado na perturbação

```
int impacto = P * 0.3;
```

6. **Estabilização:**

- Se o erro for pequeno, reduz o PWM para evitar overshoot.

7. **Saturação e Escrita PWM**:

    - Garante que o PWM fique entre 0 e 255.
    - Escreve o valor no pino do LED.

8. **Monitor Serial**:

- Exibe RPM, PWM, perturbação e erro no terminal.

Exemplo de saída da Serial:

```
RPM: 1425.00 | PWM: 95 | Perturbação: -5% | Erro: 75
```

Observações

- O LED pisca com intensidade proporcional ao esforço do motor simulado.
- O sensor touch simula uma falha ou interferência extrema.
- O potenciômetro permite testar variações suaves no sistema.
- O motor de passo e o sensor DHT11 estão presentes fisicamente, mas não são utilizados neste sketch — podem ser integrados futuramente.