Controle de Potência por Ângulo de Disparo

Código:

```
#define optoPin 4           // Pino que controla o PC817
const int analogPin = A0;   // Entrada analógica (sensor ou potenciômetro)
const int threshold = 900;  // Limite para acionar a carga
volatile int power = 100;   // Potência inicial (0 a 256)
int analogValue = 0;        // Valor lido do sensor

void setup() {
  Serial.begin(9600);
  pinMode(optoPin, OUTPUT);
  digitalWrite(optoPin, LOW); // Garante que a carga começa desligada
}

void loop() {
  analogValue = analogRead(analogPin); // Lê o sensor
  Serial.println(analogValue);

  if (analogValue > threshold) {
    // Ajusta a potência com base no sensor
    power = map(analogValue, threshold, 1023, 0, 256);

    // Calcula o tempo de condução em microssegundos
    int conductionTime = map(power, 0, 256, 0, 8333); // Até 8.33ms (meio ciclo AC)
    digitalWrite(optoPin, HIGH);                    // Liga a carga
    delayMicroseconds(conductionTime);              // Tempo proporcional à potência
    digitalWrite(optoPin, LOW);                     // Desliga a carga
  } else {
    digitalWrite(optoPin, LOW); // Mantém a carga desligada
  }

  delay(10); // Atraso para estabilidade
}
```

O código ajustado controla o **tempo de condução do optoacoplador PC817** em resposta a um valor analógico lido no pino `A0` do Arduino. Isso permite regular a potência média fornecida à lâmpada ou carga conectada. Aqui está uma análise do que ele faz:

1. Leitura do sensor ou potenciômetro

No início do `loop()`, o código lê o valor analógico de um sensor ou potenciômetro conectado ao pino `A0`:

```
analogValue = analogRead(analogPin); // Lê o sensor (0 a 1023)
Serial.println(analogValue);        // Exibe o valor no monitor serial
```

Esse valor é usado para determinar o comportamento do sistema:

- **Se for maior que o `threshold` (900)**: A carga será acionada.
- **Se for menor que o `threshold`**: A carga será mantida desligada

2. Ajuste da potência

Quando o valor lido do sensor é maior que o limite (`threshold`), o código ajusta o tempo de condução do optoacoplador. Isso regula a quantidade de energia entregue à carga:

```
power = map(analogValue, threshold, 1023, 0, 256);

int conductionTime = map(power, 0, 256, 0, 8333); // Tempo em
microssegundos
```

- A função `map()` converte o valor do sensor (900 a 1023) em uma escala de potência (0 a 256).
- A potência é convertida em um **tempo de condução** para o optoacoplador, variando de 0 até **8.33 ms** (meio ciclo da rede AC a 60 Hz).

3. Controle do optoacoplador

Com base no tempo de condução calculado, o optoacoplador é acionado:

```
digitalWrite(optoPin, HIGH); // Liga a carga

delayMicroseconds(conductionTime); // Mantém a carga ligada por um tempo proporcional

digitalWrite(optoPin, LOW);  // Desliga a carga
```

- Quanto maior o valor de `conductionTime`, mais tempo a carga ficará ligada, resultando em maior potência.
- Quando o valor lido do sensor é menor que o limite, o optoacoplador é mantido desligado:

```
digitalWrite(optoPin, LOW); // Garante que a carga está desligada
```

4. Finalidade do código

Esse controle é útil para:

- **Regular a intensidade luminosa de uma lâmpada**.
- **Controlar a potência média aplicada a uma carga**.
- **Simular um dimmer** usando o PC817, ajustando o tempo em que a carga recebe energia dentro de cada ciclo de tensão AC.

Resumo

O código lê um sensor, verifica se o valor é suficiente para ligar a carga e ajusta o tempo em que o optoacoplador conduz. Isso regula a potência média entregue à carga de forma proporcional ao valor do sensor.