Trabalhando com Entradas e Saídas Digitais

No Arduino, os pinos podem ser configurados como entradas ou saídas para interagir com o mundo físico, seja para receber dados (como a leitura de um botão) ou enviar sinais (como acender um LED).

Conceito de Entradas e Saídas Digitais

Conceito de Entradas e Saídas Digitais

No Arduino, os pinos podem ser configurados como **entradas** ou **saídas** para interagir com o mundo físico, seja para receber dados (como a leitura de um botão) ou enviar sinais (como acender um LED).

- **Entrada Digital**: Recebe um sinal de dispositivos como botões, sensores de presença ou interruptores. O valor lido será **HIGH** (5V) ou **LOW** (0V).
- **Saída Digital**: Envia sinais digitais para controlar dispositivos como LEDs, relés ou motores. O valor enviado será **HIGH** (5V) ou **LOW** (0V).

Configurando Entradas e Saídas

Configurando Entradas e Saídas

No Arduino, usamos a função `pinMode()` para definir o modo de um pino como **entrada** ou **saída**. O código é geralmente colocado na função `setup()`.

```
void setup() {
  pinMode(13, OUTPUT);  // Define o pino 13 como saída
  pinMode(7, INPUT);    // Define o pino 7 como entrada
}
```

Controlando Saídas Digitais com `digitalWrite()`

A função `digitalWrite()` é usada para enviar sinais digitais a um pino de saída. Ela pode ser configurada para `HIGH` (ligado) ou `LOW` (desligado).

Exemplo: Ligar e desligar um LED no pino 13.

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);  // Liga o LED
  delay(1000);             // Aguarda 1 segundo
  digitalWrite(13, LOW);   // Desliga o LED
  delay(1000);             // Aguarda 1 segundo
}
```

Leitura de Entradas Digitais com `digitalRead()`

A função `digitalRead()` permite ler o estado de um pino configurado como entrada. Ele retornará `HIGH` se o pino estiver recebendo 5V, ou `LOW` se estiver recebendo 0V.

Exemplo: Ler o estado de um botão conectado ao pino 7.

```
int buttonPin = 7;
int buttonState = 0;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(13, OUTPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);  // Lê o estado do botão
  if (buttonState == HIGH) {
    digitalWrite(13, HIGH);  // Liga o LED se o botão estiver pressionado
  } else {
    digitalWrite(13, LOW);   // Desliga o LED se o botão estiver solto
  }
}
```

Pull-up e Pull-down Resistors

Pull-up e Pull-down Resistors

Para garantir uma leitura precisa das entradas digitais, é importante usar resistores **pull-up** ou **pull-down**. Esses resistores "puxam" o valor da entrada para HIGH ou LOW quando o botão não está sendo pressionado, evitando leituras incorretas causadas por ruído elétrico.

- **Resistor Pull-down**: Mantém o valor LOW até que o botão seja pressionado (quando o valor passa para HIGH).
- **Resistor Pull-up**: Mantém o valor HIGH até que o botão seja pressionado, o que faz com que o valor caia para LOW. O Arduino tem resistores **pull-up internos**, que podem ser ativados com `pinMode(pino, INPUT_PULLUP)`.

Exemplo: Usar o resistor pull-up interno.

```
int buttonPin = 7;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);  // Ativa o resistor pull-up interno
  pinMode(13, OUTPUT);
}

void loop() {
  int buttonState = digitalRead(buttonPin);
  if (buttonState == LOW) {  // O botão está pressionado (LOW por causa do pull-up)
    digitalWrite(13, HIGH);  // Liga o LED
  } else {
    digitalWrite(13, LOW);   // Desliga o LED
  }
}
```