

Ação: Programação com Arduinos

3ª sessão - Introdução à programação com Arduino



Abril, 2021



Atividade 5 – Led ativado por um botão de pressão (Push-Button)

Descrição: Nesta atividade pretende-se o controlar um LED recorrendo a um botão de pressão. A finalidade desta atividade traduz-se na manipulação de pinos digitais de forma a resolver o desafio de ligar/desligar um LED, em função do estado de um botão de pressão.

Comece por seleccionar os componentes de acordo com a lista de materiais, monte o circuito conforme representado na figura seguinte e, por fim, efetue a respetiva programação.

Após completar todos os passos, montagem do hardware, desenvolvimento da programação e envio do respetivo código para o Arduino, o LED deverá estar inicialmente apagado, ao pressionar o botão de pressão, o LED deverá acender e se o soltar, deverá desligar-se novamente

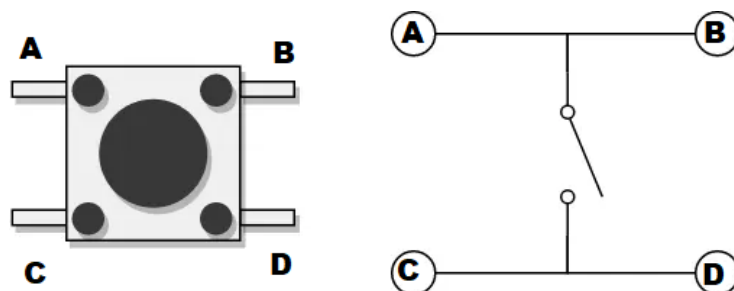
Resistências Pull-up e Pull-down

As resistências de input **Pull-up** (conectadas aos +5v) ou input **Pull-down** (conectadas à terra) são resistências adicionadas aos inputs do Arduino que definem o valor de input, no caso de não haver um input definido. São importantes porque estabilizam o circuito, caso contrário obteríamos um comportamento errático.






Imagine que um pino do microcontrolador está definido como input. Se não existe nada conectado a esse pino e o programa tentar ler o estado do mesmo, qual será o valor a atribuir? HIGH (ligado a VCC) ou LOW (ligado à terra)? É difícil dizer. Este fenómeno designa-se por floating e é por isso que se utilizam resistências no circuito pull-up ou pull-down. Habitualmente usam-se com botões de pressão ou interruptores.

Botão de pressão

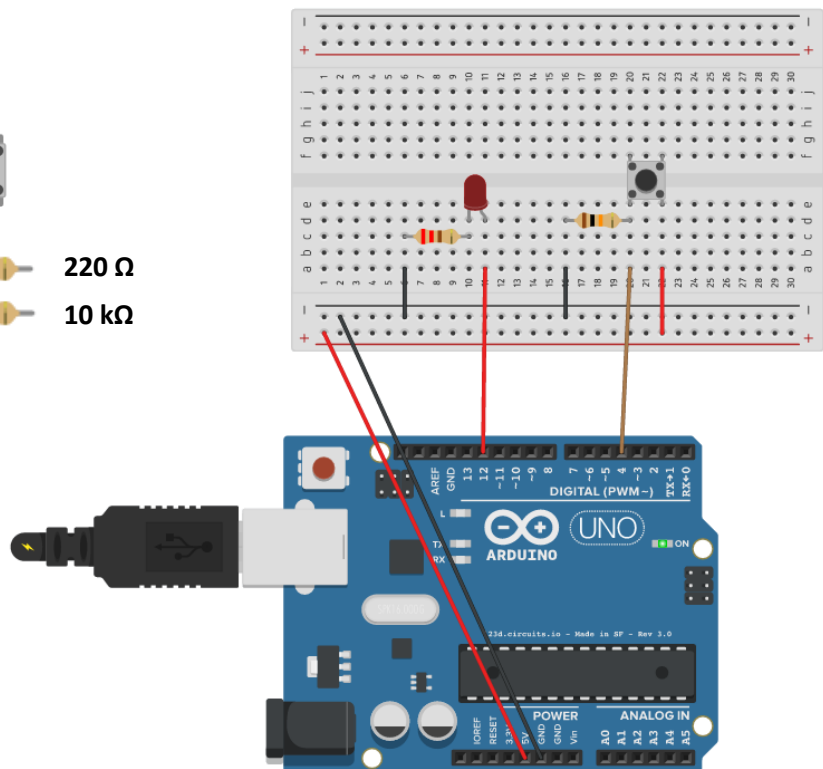
O push button (ou botão de pressão) é um componente que conecta dois pontos de um circuito quando é pressionado. Tem o mesmo funcionamento elétrico que um interruptor, fechando ou abrindo o circuito elétrico. Observe através da figura seguinte, que o pino A está ligado diretamente ao pino B, e o pino C ao pino D.



Material necessário:

- 1 Led 
- 1 Push-Button 
- 2 Resistências  220 Ω  10 k Ω
- Fios 

Esquema do circuito:



Programação:

1º - É necessário definir duas variáveis constantes que caracterizam os pinos digitais, onde se ligará o LED e o botão de pressão, portas 12 e 4 respectivamente.

```
#define ledPin 12    //Porta digital led
#define btnPin 4     //Porta digital botão
```

Para além disso, será necessário declarar uma variável para se poder guardar o estado do botão:

```
int botao;          //Variável que armazena o estado do botão
```

2º - Na função “**setup**” será necessário escrever o código que permita configurar o pino onde está ligado o led, como pino de saída (OUTPUT) e o pino do botão de pressão, como pino de entrada (INPUT):

```
pinMode(ledPin, OUTPUT);
```

3º - Deve utilizar a função **digitalRead** para obter o estado do botão e a função **digitalWrite** para acender ou apagar o led. Através do estado do botão (premido 1; não premido 0) e utilizando a estrutura condicional **if** é possível determinar se o led irá acender ou apagar:

```
botao = digitalRead(btnPin); //Ler o estado do botão
//Se o estado do botão for igual a 1 (botão pressionado)
if (botao == 1) {
    digitalWrite(ledPin, HIGH); //Liga o led
} else { //Senão
    digitalWrite(ledPin, LOW); //Desliga o led
}
```

Código Final:

```
#define ledPin 12    //Porta digital led
#define btnPin 4     //Porta digital botão

int botao;          //Variável que armazena o estado do botão

void setup() {

    // Definir a porta digital do led como OUTPUT
    pinMode(ledPin, OUTPUT);
    // Definir a porta digital do botão como INPUT
    pinMode(btnPin, INPUT);

}

void loop() {

    botao = digitalRead(btnPin); //Ler o estado do botão
    //Se o estado do botão for igual a 1 (botão pressionado)
    if (botao == 1) {
        digitalWrite(ledPin, HIGH); //Liga o led
    } else { //Senão
        digitalWrite(ledPin, LOW); //Desliga o led
    }
}
```

Desafio:

Efetue uma alteração ao seu código de forma que o botão se comporte como um interruptor:

- se o LED estiver desligado quando se pressiona o botão, então este deverá acender;
- se o LED estiver ligado quando se pressiona o botão, então este deverá apagar.

Atividade 6 – Leds controlados por uma letra

Descrição: Nesta atividade pretende-se construir um circuito que permita controlar três leds através da utilização do Monitor Série.

Para cada led é atribuída uma letra. Quando o utilizador digitar, no monitor série, a letra correspondente ao led, este acenderá.

R – Red

Y – Yellow

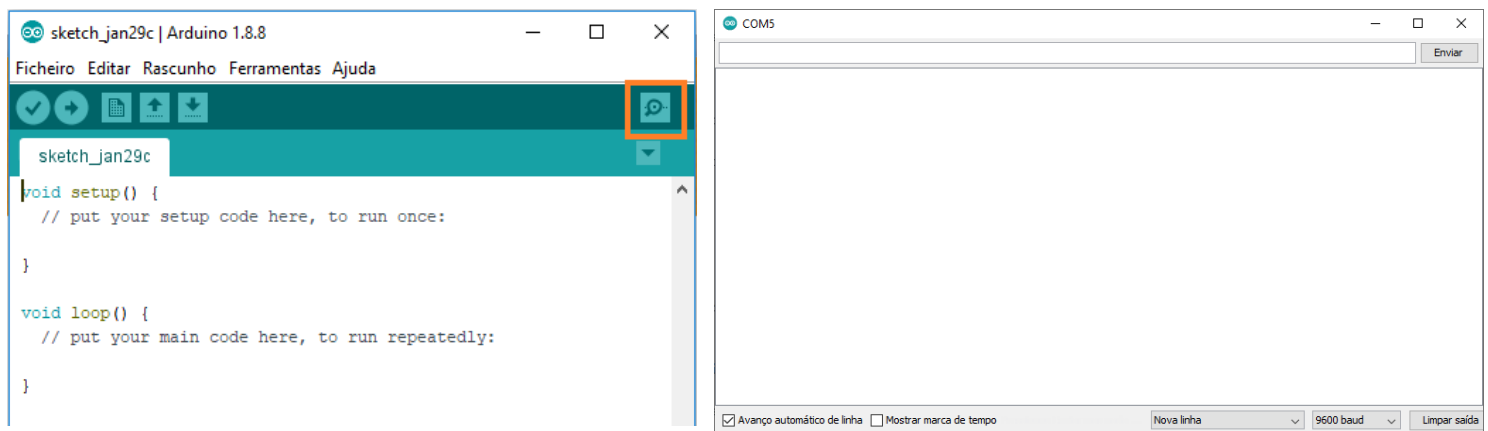
G – Green

Se o utilizador digitar a letra A, todos os leds se apagam.

Se o utilizador digitar a letra L, todos os leds se ligam.

O Monitor Série é uma ferramenta muito útil, especialmente para se poder efetuar debug (depuração) do código. O monitor mostra os dados transmitidos pelo Arduino através da porta USB ou porta série. Este permite:

- a comunicação entre o Arduino e a porta série;
- visualizar valores lidos pelos sensores;
- mostrar textos;
- enviar dados para o Arduino (como se fossem ordens).



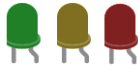
Para se poder utilizar o monitor série é necessário inicializá-lo na função Setup, através da instrução `Serial.begin(x)`, em que x é a taxa de transferência em bits por segundo: 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, ou 115200.

Para que não surjam caracteres ilegíveis durante a transferência de dados, deverá ser selecionada a opção 9600. O que significa que os dados serão transmitidos, através do cabo USB, a uma taxa de transferência de 9600 bits por segundo.

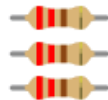
Comece por selecionar os componentes de acordo com a lista de materiais. Monte o circuito conforme representado na figura seguinte e efetue a respetiva programação. Sempre que possível utilize as cores sugeridas para os fios.

Material necessário:

- 3 Leds



- 3 Resistências

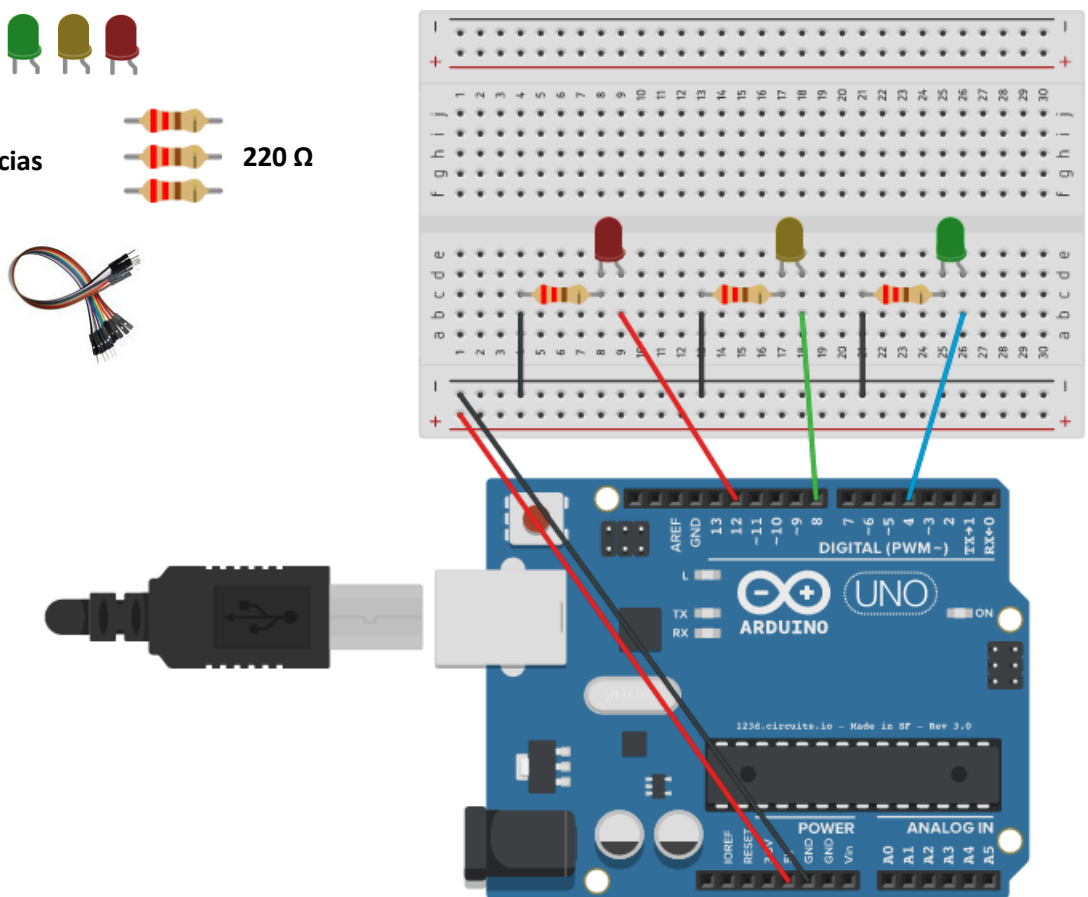


220 Ω

- Fios



Esquema do circuito:



Programação:

1º - Será necessário definir três variáveis constantes correspondentes aos pinos de cada led:

```
#define ledPinG 4 //Porta digital led verde
#define ledPinY 8 //Porta digital led amarelo
#define ledPinR 12 //Porta digital led vermelho
```

2º - Declarar uma variável que irá guardar a letra escrita pelo utilizador:

```
//variável que armazena o caracter enviado pelo monitor série
char letra;
```

3º - Na função “**setup**” será necessário escrever o código que permita configurar os pinos onde estão ligados o led, como pinos de saída (OUTPUT):

```
pinMode(ledPinG, OUTPUT);
pinMode(ledPinY, OUTPUT);
pinMode(ledPinR, OUTPUT);
```

Projeto ArdRobotica

4º - Na função “**setup**” adicionar a instrução que permite estabelecer a velocidade de comunicação entre o monitor série e o Arduino:

```
Serial.begin(9600);
```

5º - Na função “**loop**” obter as letras escritas pelo utilizador no monitor série caso a comunicação esteja disponível. Utilizar a estrutura condicional [if](#):

```
if (Serial.available() > 0) {  
    letra = Serial.read();  
    ...  
}
```

6º - Analisar a variável letra e de acordo com a respetiva letra associada, acender o led. Utilizar a estrutura condicional [switch](#):

```
switch (letra) {  
    case 'R':  
        digitalWrite(ledPinR, HIGH);  
        break;  
    case 'Y':  
        digitalWrite(ledPinY, HIGH);  
        break;  
    case 'G':  
        digitalWrite(ledPinG, HIGH);  
        break;  
    case 'A':  
        digitalWrite(ledPinR, HIGH);  
        digitalWrite(ledPinY, HIGH);  
        digitalWrite(ledPinG, HIGH);  
        break;  
    case 'L':  
        digitalWrite(ledPinR, LOW);  
        digitalWrite(ledPinY, LOW);  
        digitalWrite(ledPinG, LOW);  
        break;  
}
```


Código Final:

```
#define ledPinG 4 //Porta digital led verde
#define ledPinY 8 //Porta digital led amarelo
#define ledPinR 12 //Porta digital led vermelho
char letra; //variável que armazena o caracter enviado pelo monitor série

void setup() {
    // Definir as portas digitais dos leds como OUTPUT
    pinMode(ledPinG, OUTPUT);
    pinMode(ledPinY, OUTPUT);
    pinMode(ledPinR, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    if (Serial.available() > 0) {
        letra = Serial.read();
        switch (letra) {
            case 'R':
                digitalWrite(ledPinR, HIGH);
                break;
            case 'Y':
                digitalWrite(ledPinY, HIGH);
                break;
            case 'G':
                digitalWrite(ledPinG, HIGH);
                break;
            case 'A':
                digitalWrite(ledPinR, HIGH);
                digitalWrite(ledPinY, HIGH);
                digitalWrite(ledPinG, HIGH);
                break;
            case 'L':
                digitalWrite(ledPinR, LOW);
                digitalWrite(ledPinY, LOW);
                digitalWrite(ledPinG, LOW);
                break;
        }
    }
}
```


Atividade 7 – Led com Fade

Descrição: Nesta atividade pretende-se construir um circuito que permita controlar a luminosidade de um led, através de um potenciômetro.

Potenciômetro

O potenciômetro é uma resistência ajustável a uma gama de valores entre 0 e um valor definido que se encontra inscrito no potenciômetro. Uma das aplicações mais comuns dos potenciômetros é como divisor de tensão, uma vez que permite obter uma tensão de saída variável no pino central, a partir de um diferencial de tensão definido entre as duas extremidades do mesmo. Um terminal é ligado à terra, outro aos 5V e o último (o central) é ligado a um pino analógico do Arduino. O potenciômetro pode ser utilizado para controlar o volume de uma aparelhagem, a luminosidade de uma lâmpada, etc.

PWM - Pulse-Width Modulation

As entradas analógicas permitem distinguir qual a tensão que está na porta, ou seja, permitem “dividir” este valor (0V-5V) em 1024 níveis (1024 bits), sendo que o valor máximo é 1023 (4.955V) e o valor mínimo é 0 (0V). As entradas digitais não permitem ver a tensão que está na porta tão detalhadamente, visto o valor desta só poder ser 1 (5V) ou 0 (0V).

Nas saídas digitais podemos variar a tensão entre 0 e 1 (sendo que 1=5V e 0=0V). Algumas saídas digitais têm também um tipo de saída específico, uma saída digital PWM. A sigla PWM provém do inglês “Pulse Width Modulation” e representa uma técnica que permite por via de um sinal digital, emular um resultado analógico.

Neste tipo de saída podemos controlar a tensão que sai das portas, ou seja, podemos controlar a velocidade de um motor, ou a luminosidade de um LED.





Como o Arduino só tem pinos de Input Analógico (A0...A5), o Output Analógico só pode ser conseguido com PWM, que correspondem aos pinos digitais sinalizados com ~ no Arduino.

O Arduino Uno disponibiliza os pinos 3, 5, 6, 9, 10 e 11 com função PWM. Nestes pinos além da função digitalWrite() que permite escrever os valores 0 ou 1, está também disponível a função analogWrite() que permite definir valores numa escala de 0 a 255.

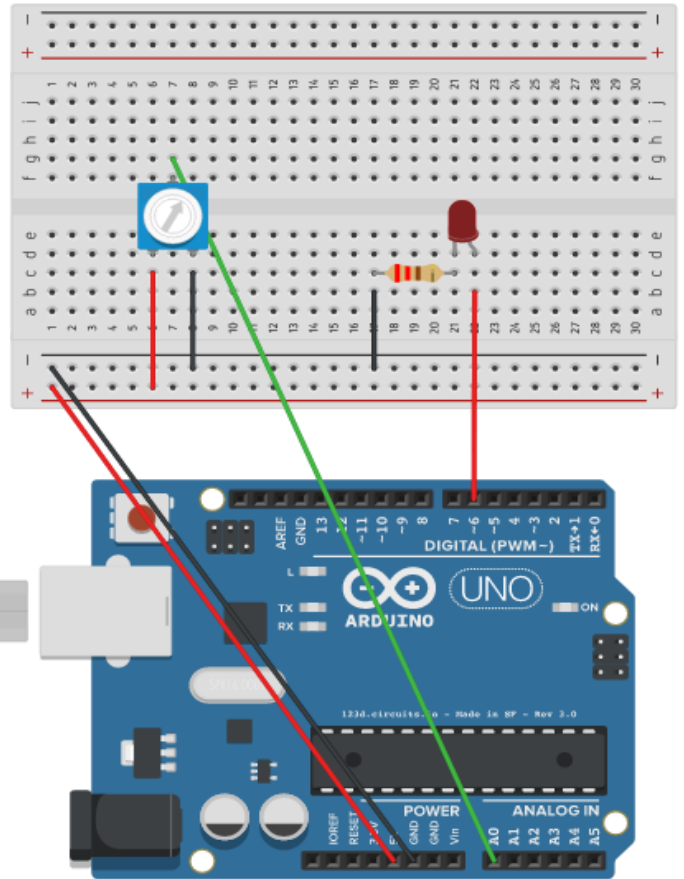
Projeto ArdRobotica

Comece por selecionar os componentes de acordo com a lista de materiais, monte o circuito conforme representado na figura seguinte e, por fim, efetue a respectiva programação.

Material necessário:

- 1 Led 
- 1 Potenciômetro 
- 1 Resistência 220 Ω 
- Fios 

Esquema do circuito:



Programação:

1º - Será necessário definir duas variáveis constantes relativas aos pinos correspondentes ao led e ao potenciômetro:

```
#define ledPin 6    //Porta digital led
#define potPin A0   //Porta analógica do potenciômetro
```

2º - Declarar as variáveis associadas ao valor obtido no potenciômetro e à luminosidade do led:

```
int potenciometro, luminosidade;
```

3º - Configurar o pino do led como pino de saída (OUTPUT) e o pino do potenciômetro como pino de entrada (INPUT):

```
pinMode(ledPin, OUTPUT);
pinMode(potPin, INPUT);
```

3º - Utilizar a função **analogRead** para obter os valores do potenciômetro:

```
potenciometro = analogRead(potPin);
```

4º - Mapear os valores registados pelo potenciômetro (0-1023) com o intervalo de valores possível numa porta digital PWM (0 - 255); Utilizar a função **map**.

```
luminosidade = map(potenciometro, 0, 1023, 0, 255);
```

5º - Utilizar a função **analogWrite** para definir a luminosidade do led numa porta **PWM**.

```
analogWrite(ledPin, luminosidade);
```

Código Final:

```
#define ledPin 6      //Porta digital led
#define potPin A0    //Porta analógica do potenciômetro

//potenciometro - variável que armazena os valores obtidos através do potenciômetro
//luminosidade - variável que armazena os valores de luminosidade
int potenciometro, luminosidade;

void setup() {
    // Definir a porta digital do led como OUTPUT
    pinMode(ledPin, OUTPUT);
    pinMode(potPin, INPUT);
}

void loop() {
    //Efetua a leitura dos valores do potenciômetro (0 - 1023);
    potenciometro = analogRead(potPin);
    //Mapear os valores do potenciômetro para o intervalo de valores 0 - 255
    luminosidade = map(potenciometro, 0, 1023, 0, 255);
    //Atribuir o valor da luminosidade ao led
    analogWrite(ledPin, luminosidade);
    //Aguardar 25ms
    delay(25);
}
```