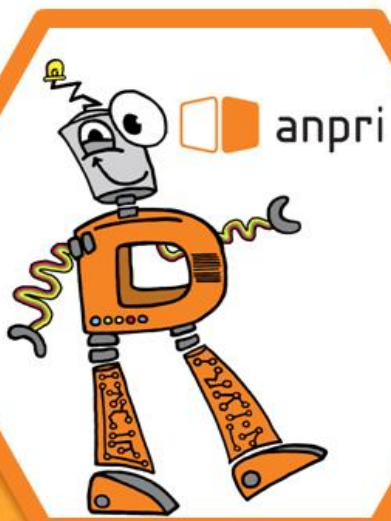
Ação: Programação com Arduinos

2ª sessão - Introdução à programação com Arduino

Abril, 2021



Atividade 2 – Led a piscar (Blink)

Descrição: Nesta atividade pretende-se construir um circuito que permita colocar um led a piscar.

Comece por selecionar os componentes de acordo com a lista de materiais. Monte o circuito conforme representado na figura seguinte. O resultado esperado é ver-se o led vermelho, sobre a placa de ensaio, a piscar a uma cadência de 1 segundo.

Nas suas montagens de hardware certifique-se que o Arduino está desligado (desligue o cabo USB entre o computador e o Arduino).

Pode construir o circuito em qualquer ponto da placa desde que respeite as disposições das linhas verticais e horizontais da mesma. A introdução dos componentes nos buracos da placa deve ser feita com algum cuidado, principalmente se for nova. É normal nessas condições ter que fazer alguma pressão aquando da introdução de um fio ou ponta de componente, num buraco.

O led tem polaridade, só emite luz se corretamente ligado. Certifique-se que a linha da placa de ensaio onde está ligada a perna mais longa do led (ânodo) está ligada a um pino digital (no exemplo está ligada ao pino 4) do Arduino. A perna mais curta (cátodo) do led deve ser ligada à resistência de 220 Ω e daí o circuito fecha no terminal GND (terra). Quando estiver certo da correção das ligações pode ligar o Arduino ao PC, através do cabo USB.

COMO DECIDIR QUAL O VALOR DA RESISTÊNCIA A USAR?

Os pinos digitais do Arduino trabalham com tensões entre 0V (OFF) e 5V (ON) e correntes até 40mA. No datasheet dos leds é especificado que produzem uma queda de tensão (forward voltage) de 1,8V e uma corrente máxima de 20mA para se obter o brilho máximo.

Precisamos, portanto, de reduzir a corrente dos 40mA para 20mA. Para tal, precisamos de colocar uma resistência em série com o LED.

O cálculo do valor da resistência é feito usando a lei de Ohm: I = V/ R, onde I é a corrente, V é a tensão e R a resistência. Para calcular a resistência usa-se a fórmula equivalente R = V / I. Sabe-se que na resistência a corrente deve ser limitada a 20 mA = 0,02A. A resistência então será calculada da seguinte forma:

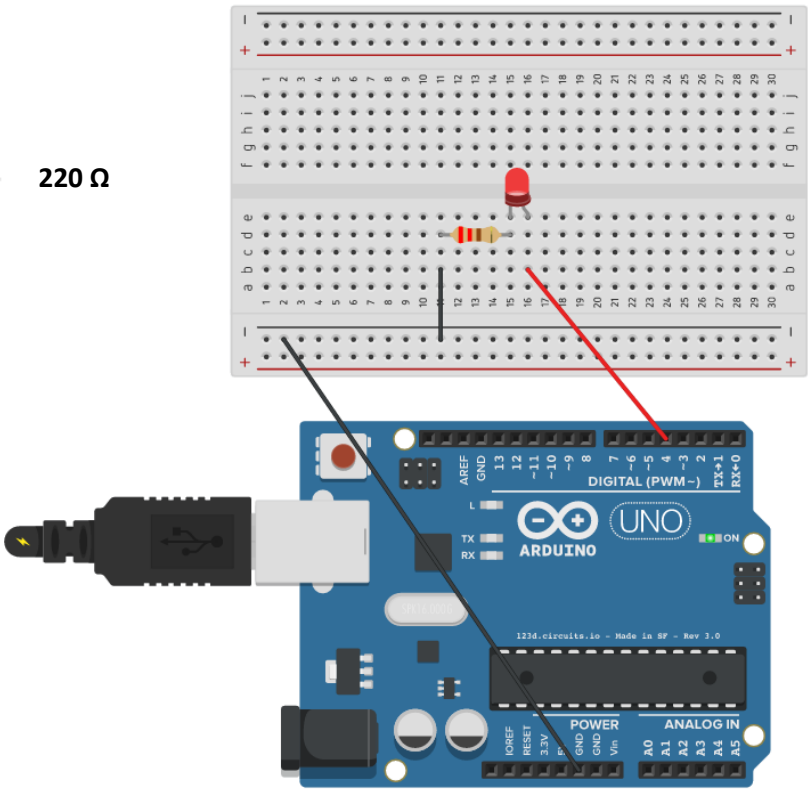
$$\boldsymbol{R = \frac{5 - 1,8}{0,02} = 160\Omega}$$

Caso não tenham uma resistência com este valor, então deverão escolher uma resistência com o valor superior mais aproximado do calculado. Escolhi a resistência de 220 Ω por ser a mais comum nos kits de Arduino.

Material necessário:

- **1 Led** 
- **1 Resistência**  **220 Ω**
- **Fios** 

Esquema do circuito:



Programação:

Dentro da IDE do Arduino iremos programar usando a linguagem de programação do Arduino, baseada em Wiring.
No decorrer do código é muito útil poder-se comentar ideias e algoritmos. As instruções escritas após o símbolo // ou entre /* o meu comentário */ são ignoradas pelo compilador.

1º - Vamos definir uma variável (constante) para o pino que vamos utilizar, no caso o pino "4", que vai permitir identificar, ao longo do código, o pino ao qual se vai ligar o LED.

```
#define ledPin 4
```

2º - A função **"setup"** permite configurar o modo de funcionamento do microcontrolador. É através desta função que o programa irá identificar corretamente os dispositivos que serão usados. Só será executada uma vez, quando se energiza a placa ou quando se prime o botão RESET.
Assim, dentro da função **"setup"** vamos escrever o código que permite configurar o pino do microcontrolador, onde está ligado o led, como pino de saída (OUTPUT).

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

3º - A função **"loop"** estará em execução contínua depois da execução da função **"setup"**. É esta a razão da ilusão de que o LED estará a piscar, dado que nesta função se troca o estado do pino ledPin entre os valores HIGH e LOW com uma pausa de um segundo entre cada passagem.

Assim, na função **"loop"** vamos dar a indicação ao led para se ligar, esperar um segundo, desligar e voltar a desligar-se:

```
digitalWrite(ledPin, HIGH);     //Acender o led
delay(1000);          //Esperar 1 segundo
digitalWrite(ledPin, LOW);     //Apagar o led
delay(1000);          //Esperar 1 segundo
```

Código Final:

```
#define ledPin 4     //definir o pino a utilizar, neste caso o pino 4

void setup() {
  // Função que permite definir se uma porta digital é de INPUT ou OUPUT
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);     //Acender o led
  delay(1000);          //Esperar 1 segundo
  digitalWrite(ledPin, LOW);     //Apagar o led
  delay(1000);          //Esperar 1 segundo
}
```

Atividade 3 – Semáforo

Descrição: Nesta atividade pretende-se construir um circuito que permita simular um semáforo. 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.

Regras de funcionamento do semáforo: Só pode estar um led aceso de cada vez. Cada vez que um led acende terá de se garantir que os outros leds estão desligados.

Sequência de cores:

Acende **Verde** (5 segundos e passa a **Amarelo**)

Acende **Amarelo** (1 segundo e passa a **Vermelho**)

Acende **Vermelho** (5 segundos e passa a **Verde**)

Material necessário:

- **3 Leds** 
- **3 Resistências**  **220 Ω**
- **Fios** 

Esquema do circuito:

Programação:

1º - Será necessário definir quais os pinos correspondentes a cada led: **#define**

2º - Configurar os pinos dos leds como pinos de saída: **pinMode**

3º - Utilizar a função **digitalWrite** para acender/apagar os leds e a função **delay** para determinar o tempo de espera entre as transições do semáforo.

Código Final:

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

void setup() {
  // Definir as portas digitais dos leds como OUTPUT
  pinMode(ledPinG, OUTPUT);
  pinMode(ledPinY, OUTPUT);
  pinMode(ledPinR, OUTPUT);
}
void loop() {
  // Ligar o led verde e desligar os outros leds
  digitalWrite(ledPinG, HIGH);
  digitalWrite(ledPinY, LOW);
  digitalWrite(ledPinR, LOW);
  delay(5000);    //Esperar 5 segundos
  // Ligar o led amarelo e desligar os outros leds
  digitalWrite(ledPinG, LOW);
  digitalWrite(ledPinY, HIGH);
  digitalWrite(ledPinR, LOW);
  delay(1000);    //Esperar 1 segundo
  // Ligar o led vermelho e desligar os outros leds
  digitalWrite(ledPinG, LOW);
  digitalWrite(ledPinY, LOW);
  digitalWrite(ledPinR, HIGH);
  delay(5000);
}
```

Atividade 4 – Arduino Knight Rider

Descrição: Nesta atividade pretende-se construir uma sequência luminosa com LEDs, semelhante ao que existia no famoso carro K.I.T.T da série de televisão o Justiceiro (Knight Rider). Com este projeto pretende-se introduzir o conceito de array.

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.

Confirme cuidadosamente todas as ligações antes de ligar o circuito. Ao montar corretamente o circuito, os LEDs irão criar um efeito de movimento entre cada extremidade da fila de leds.

Este circuito é um pouco mais trabalhoso, por isso, para minimizarmos os erros de montagem iremos reduzir o número de fios do mesmo. Sempre que possível utilize as cores sugeridas. Os LEDs estão ligados aos pinos digitais 4 a 13.

Tenha atenção à posição das pernas dos LEDs. Os cátodos ligam à terra (GND) (a linha assinalada a azul na placa e daí ligam-se ao GND) e os ânodos ligam cada ao respetivo pino digital em série com uma resistência de 220 Ω.

Material necessário:

- **10 Leds** 
- **10 Resistências**  **220 Ω**
- **Fios** 

Esquema do circuito:

Programação:

Como estamos a utilizar 10 leds, teríamos de declarar 10 variáveis para representar as respetivas portas digitais. Como são muitos leds e estes têm todos a mesma função, vamos optar por criar uma estrutura de dados que permita armazenar os números das portas digitais de todos os leds: o array. Um array é uma lista de valores de um mesmo tipo de dados que se podem posteriormente alterar ou ler utilizando um índice. Os índices, correspondem a cada posição do array e começam no valor zero.

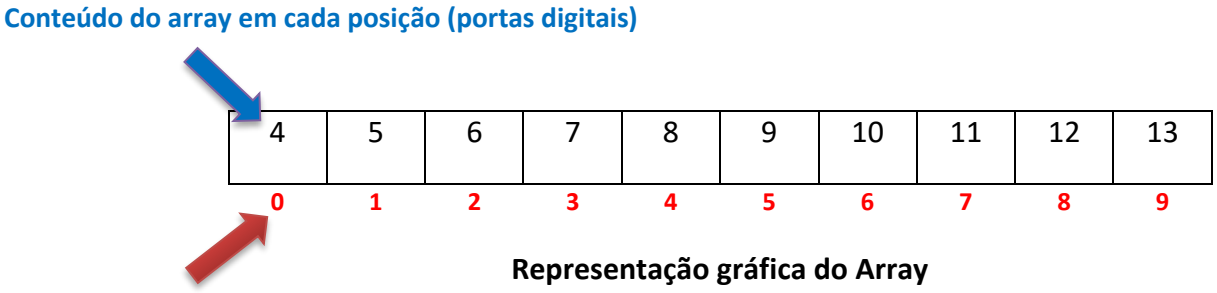
A intenção será criar uma estrutura única que permita manipular os valores correspondentes aos pinos dos vários leds.
Assim, vamos optar por declarar uma variável constante de um tipo especial: array de inteiros.

```
const int ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
```

A expressão **const** significa que a variável (array) será constante, ou seja, não poderá ser alterada ao longo da execução do programa.

O tipo de dados **int** permite representar números inteiros negativos ou positivos entre -32768 e 32767.

A expressão **ledPin[]** identifica o nome do array e os valores dentro das chavetas permitem inicializar e atribuir o número das portas digitais a cada posição do array.



Representação gráfica do Array

Na função setup() é necessário configurar os pinos digitais correspondentes a cada um dos leds, como pinos de saída (OUTPUT). Para evitar termos de repetir 10 vezes a instrução pinMode, podemos recorrer a um ciclo **for**. A instrução **for** é utilizada para repetir um bloco de expressões de código. Pode ser utilizado para percorrermos cada uma das posições dos arrays e executar operações repetitivas em conjuntos de dados ou de pinos. Normalmente, define-se uma variável e o seu respetivo valor inicial, a condição de paragem do ciclo e a atualização da variável após cada iteração do ciclo.

Função setup:

```
int i;

void setup() {
  for (i = 0; i < 10; i++) {
    pinMode(ledPin[i], OUTPUT);
  }
}
```

Neste exemplo, começamos por declarar uma variável inteira i, necessária para podermos percorrer cada posição do array:

```
int i;
```

Depois inicializamos a variável a zero, uma vez que a primeira posição do array é o índice 0:

```
for (i = 0;
```

Definimos a condição de paragem do ciclo:

```
i < 10;
```

Finalmente, é necessário indicar de que forma se atualiza a variável i, após cada iteração. Neste caso pretendemos que a variável seja incrementada numa unidade, pelo que usamos a seguinte expressão:

```
i++
```

Para se poder aceder aos valores de cada posição do array usamos a seguinte expressão:

```
ledPin[i]
```

Função loop:

Para se conseguir o efeito luminoso semelhante ao K.I.T.T, os leds começam por piscar da direita para a esquerda, do pino 4 até ao 13 fazendo o primeiro varrimento e de seguida do 13 até ao 4 efetuando o segundo varrimento. Toda esta sequência volta-se a repetir sucessivamente, pelo que necessitamos de um ciclo **for**. Em cada instante, estarão acesos apenas três leds. Entre cada iteração do ciclo vamos colocar um intervalo de 100ms.

Primeiro varrimento:

```
for (i = 0; i < 8; i++) {
  digitalWrite(ledPin[i], HIGH);
  digitalWrite(ledPin[i + 1], HIGH);
  digitalWrite(ledPin[i + 2], HIGH);
  digitalWrite(ledPin[i - 1], LOW);
  delay(100);
}
```

Começamos por criar um ciclo **for**, onde é inicializada a variável i a zero, que corresponde à posição inicial do array. Como em cada instante, estarão acesos três leds, definimos a condição de paragem i < 8, uma vez que apenas serão necessárias apenas oito iterações deste ciclo para

se conseguir acender todos os leds. Neste caso pretendemos que a variável i seja incrementada numa unidade, de forma a percorrer o array para a direita.

As seguintes instruções permitem acender três leds de cada vez:

```
digitalWrite(ledPin[i], HIGH);
digitalWrite(ledPin[i + 1], HIGH);
digitalWrite(ledPin[i + 2], HIGH);
```

A seguinte instrução permite apagar os leds à medida que o varrimento se desloca para a esquerda:

```
digitalWrite(ledPin[i - 1], LOW);
```

A instrução **delay** permite estabelecer um intervalo de 100ms entre cada iteração.

Segundo varrimento:

```
for (i = 9; i > 1; i--) {
  digitalWrite(ledPin[i], HIGH);
  digitalWrite(ledPin[i - 1], HIGH);
  digitalWrite(ledPin[i - 2], HIGH);
  digitalWrite(ledPin[i + 1], LOW);
  delay(100);
}
```

Começamos por criar um ciclo **for**, onde é inicializada a variável i a nove, que corresponde à posição final do array. Como em cada instante, estarão acesos três leds, definimos a condição de paragem i > 1, uma vez que apenas serão necessárias apenas oito iterações deste ciclo para se conseguir acender todos os leds. Neste caso pretendemos que a variável i seja decrementada numa unidade, de forma a percorrer o array para a esquerda.

As seguintes instruções permitem acender três leds de cada vez:

```
digitalWrite(ledPin[i], HIGH);
digitalWrite(ledPin[i - 1], HIGH);
digitalWrite(ledPin[i - 2], HIGH);
```

A seguinte instrução permite apagar os leds à medida que o varrimento se desloca para a direita:

```
digitalWrite(ledPin[i + 1], LOW);
```

A instrução **delay** permite estabelecer um intervalo de 100ms entre cada iteração.

Código Final:

```
const int ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};

int i;

void setup() {
  for (i = 0; i < 10; i++) {
    pinMode(ledPin[i], OUTPUT);
  }
}

void loop() {
  // put your main code here, to run repeatedly:
  for (i = 0; i < 8; i++) {
    digitalWrite(ledPin[i], HIGH);
    digitalWrite(ledPin[i + 1], HIGH);
    digitalWrite(ledPin[i + 2], HIGH);
    digitalWrite(ledPin[i - 1], LOW);
    delay(100);
  }

  for (i = 9; i > 1; i--) {
    digitalWrite(ledPin[i], HIGH);
    digitalWrite(ledPin[i - 1], HIGH);
    digitalWrite(ledPin[i - 2], HIGH);
    digitalWrite(ledPin[i + 1], LOW);
    delay(100);
  }
}
```