Licenciatura Engenharia Informática e Multimédia
Instituto Superior de Engenharia de Lisboa
Ano letivo 2022/2023

Sensores e Atuadores

Relatório: Trabalho Lab04

Turma: 11D	Grupo: 0
Nome: Daniel Silva	Número: 50781
Nome: João Ramos	Número: 50730
Nome: Miguel Alcobia	Número: 50746

Data: 14 de Novembro 2022

Objetivo:

Esta experiência teve em vista os alunos saberem trabalhar com a placa Arduino e aprender as bases de C++ para futuros trabalhos.

Material:

- Breadboard
- Arduino Uno
- Resistências
- Botões
- LDR
- LEDs
- Cabos
- Osciloscópio
- Cabos

Preparação teórica:

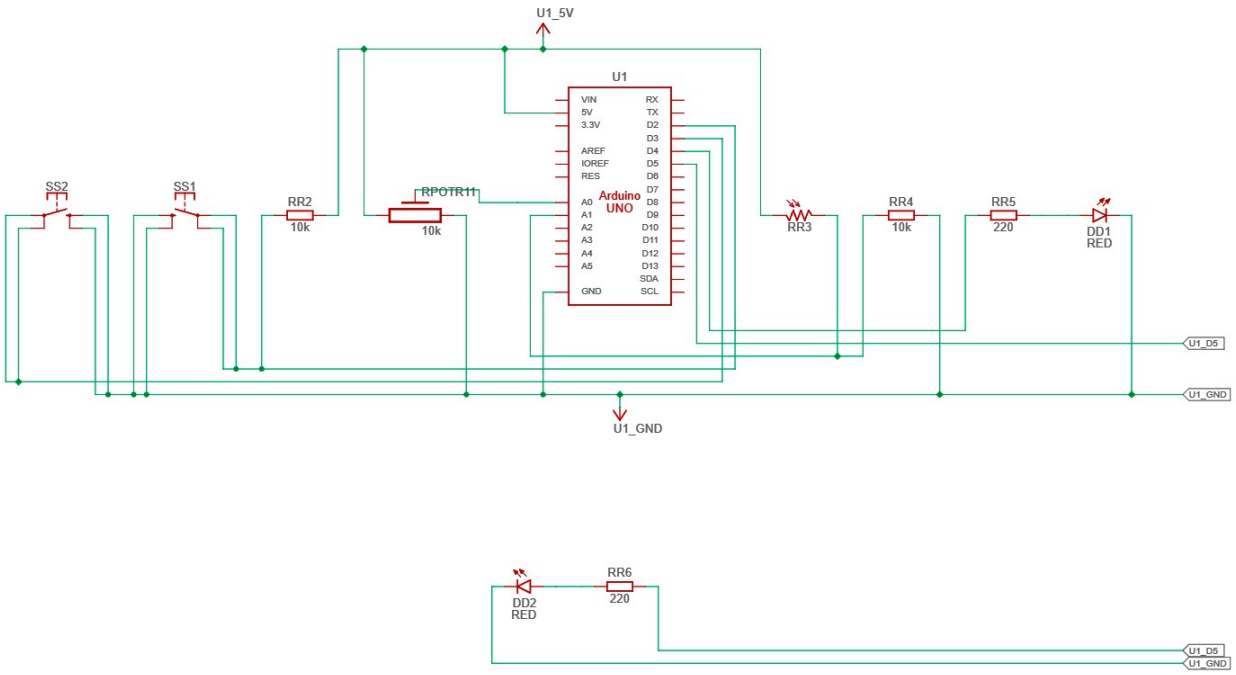


Figura 1 - Esquema feito no Tinkercad

2-

a) Leitura do V2:

Estando ligado ao pino A0, definido anteriormente como `PINPOT`, usou-se a função `analogRead()` para ler a tensão do pino do potenciómetro entre 0.00 e 5.00V. Como a função devolve o valor da tensão em valores inteiros entre 0 e 1023, recorreu-se a uma regra de 3 simples para devolver o valor em Volts. Por fim retornou-se o valor de `V2`.

```
float leituraV2(){
 float V2=analogRead(PINPOT)*5.00/1023;
 return V2;
}
```

Calcular o x:

Para calcular o valor de `x`, correspondente à posição do potenciómetro, declarou-se a variável `x`, do tipo `float` e atribuiu-se-lhe o valor do quociente entre a tensão de V2 (`fV2`) e 5.00V. No fim, retornou-se o valor de `x`.

```
float calcx(){
 float x=(fV2/5.00);
 return x;
}
```

Leitura de V3:

Estando ligado ao pino 2, definido anteriormente como PINS1, usou-se a função `digitalRead()` para ler o valor de S1 em `HIGH` ou `LOW` (1 ou 0, respetivamente). Porém como se deseja ter os valores ao contrário, colocou-se um sinal de negação (!). Como a função deveria retornar 0 ou 1, do género `True or False`, escolheu-se o tipo `bool` para função e para a variável retornada V3.

```
bool leituraV3(){
  bool V3=!digitalRead(PINS1);
  return V3;
}
```

Leitura V4

Estando ligado ao pino 3, definido anteriormente como PINS2, a função escolhida foi o digitalRead() para ler o valor de S2 em HIGH ou LOW (1 ou 0, respetivamente). Porém como se deseja ter os valores ao contrário, colocou-se um sinal de negação (!). Como a função deveria retornar 0 ou 1, do género True or False, escolheu-se o tipo bool para função e para a variável retornada V4.

```
bool leituraV4(){
  bool V4=!digitalRead(PINS2);
  return V4;
}
```

Leitura de V5:

Estando ligado ao pino A1, definido anteriormente como PINR3, usou-se a função analogRead() para ler o valor de R3(LDR) para ler a tensão do pino do LDR entre 0.00 e 5.00V. Como a função devolve o valor da tensão em valores inteiros entre 0 e 1023, recorreu-se a uma regra de 3 simples para devolver o valor em Volts. Por fim retornou-se o valor de V5

```
float leituraV5(){
  float V5= analogRead(PINR3)*5.00/1023;
  return V5;
}
```

Brilho do LDR:

Para efetuarmos a leitura do brilho do LDR, procedeu-se do seguinte modo: Atribuiu-se o valor de 10kΩ a R4, depois usou-se a fórmula $LDR = \frac{VDC*R4}{V5+R4}$, que traduzindo para o código fica: `LDR=(5.00*R4/leituraV5()-R4)`. Depois chegou-se ao brilho segundo a fórmula: $brilho = \sqrt[-0.837]{\frac{LDR}{23.48}}$ que em código fica: `pow((LDR/23.48), (-1/0.837)`. Por fim, retornou-se a variável do `brilho`.

```
float leituraLDR(){
  float R4 = 10;
  float LDR= (5.00*R4/(leituraV5()+R4));
  float brilho= pow((LDR/23.48), (-1/0.837);
  return brilho;
}
```

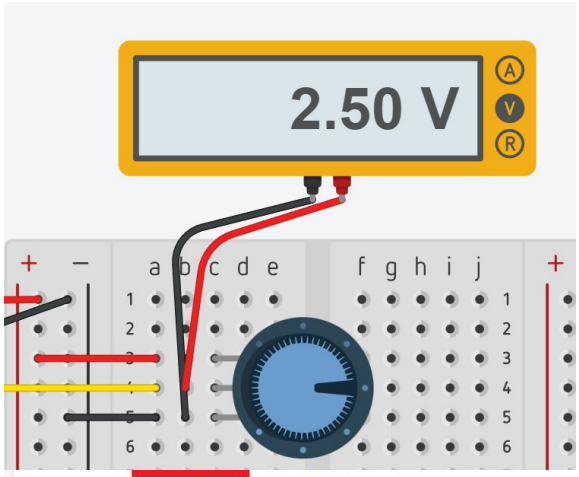
b) Para enviar os valores obtidos para a consola em formato CSV, criou-se a função do tipo `void`, `enviaconsola()`. Em seguida teve-se o seguinte raciocínio: Usaríamos `Serial.print()` nos valores que deveriam estar no *output* e separaríamos cada um com um *tab* usando `Serial.print('\t')`. As variáveis do *output* são as variáveis "finais" presentes no *loop,* cujo o valor é o resultado da função correspondente e desejada. Também se fez *print* do tempo de cada série de medições usando o `millis()`.

```
void enviaconsola(){
  Serial.print(millis());
  Serial.print('\t');
  Serial.print(fV2, 3);
  Serial.print('\t');
  Serial.print(fb, 3);
  Serial.print('\t');
  Serial.print(fV3);
  Serial.print('\t');
  Serial.print(fV4);
  Serial.print('\t');
  Serial.print(fV5, 3);
  Serial.print('\t');
  Serial.print(contador);
  Serial.print('\t');
  Serial.print(aux);
  Serial.print('\t');
  Serial.print(px, 3);
}
```

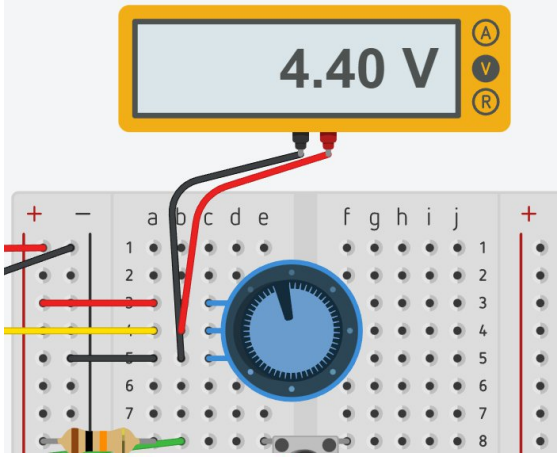
c) Para analisar o circuito duas vezes por segundo procedeu-se da seguinte forma: Criou-se a função `Ciclo2seg()` que recorre à função `enviaconsola()`e em seguida foi posto um `delay` de 500ms, pois é o resultado de 1000/2, que seria a metade de 1 segundo.

```
void Ciclo2seg(){
  enviaconsola();
  delay(500);
}
```

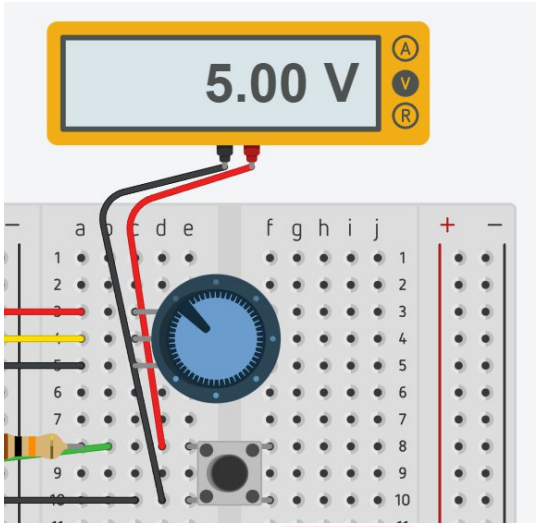
d)



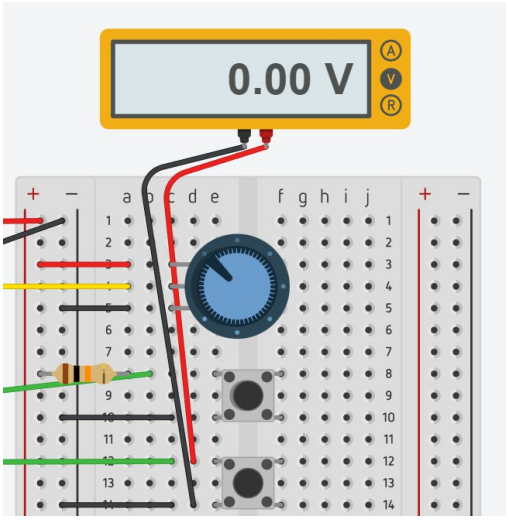
0	2.498	6.547	0	0	0.264	0	0.500
705	2.498	6.547	0	0	0.264	0	0.500
1411	2.498	6.547	0	0	0.264	0	0.500
2116	2.498	6.547	0	0	0.264	0	0.500
2822	2.498	6.547	0	0	0.264	0	0.500
3527	2.498	6.547	0	0	0.264	0	0.500
4233	2.498	6.547	0	0	0.264	0	0.500

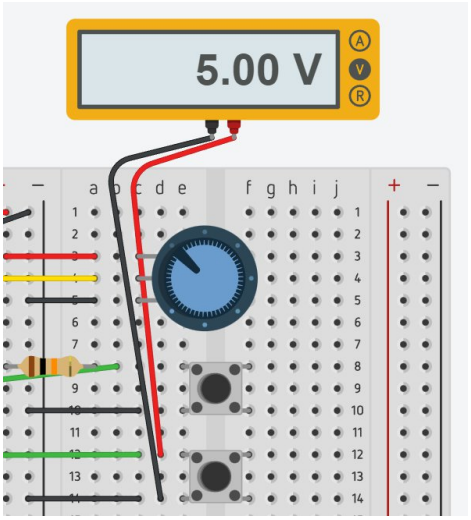


0	4.399	6.547	0	0	0.264	0	0.880
706	4.399	6.547	0	0	0.264	0	0.880
1411	4.399	6.547	0	0	0.264	0	0.880
2116	4.399	6.547	0	0	0.264	0	0.880
2822	4.399	6.547	0	0	0.264	0	0.880
3528	4.399	6.547	0	0	0.264	0	0.880
4234	4.399	6.547	0	0	0.264	0	0.880

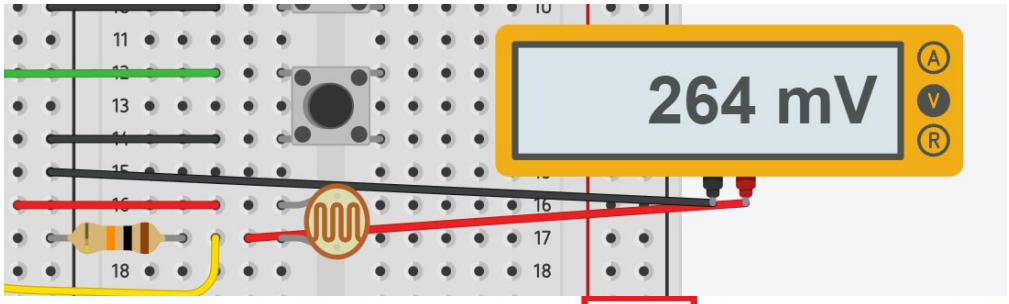


0	5.000	6.547	0	0	0.264	0	1.000
705	5.000	6.547	0	0	0.264	0	1.000
1411	5.000	6.547	0	0	0.264	0	1.000
2116	5.000	6.547	0	0	0.264	0	1.000
2822	5.000	6.547	0	0	0.264	0	1.000
3527	5.000	6.547	0	0	0.264	0	1.000
3117	5.000	6.547	1	0	0.264	1	1.000
4823	5.000	6.547	1	0	0.264	2	1.000
6529	5.000	6.547	1	0	0.264	3	1.000
8233	5.000	6.547	1	0	0.264	4	1.000
9939	5.000	6.547	1	0	0.264	5	1.000

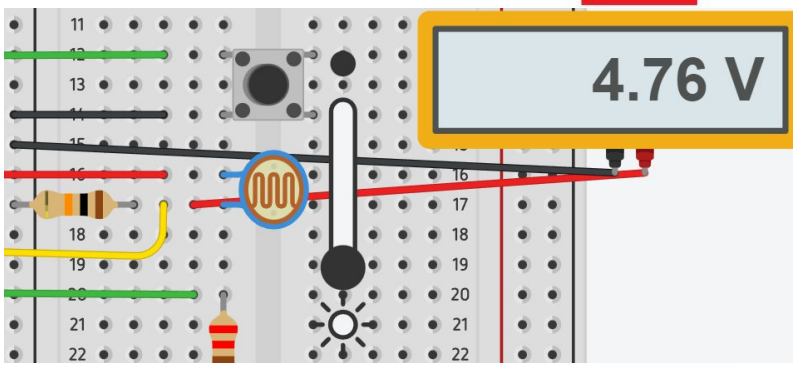




4528	5.000	6.547	0	1	0.264	1
6234	5.000	6.547	0	1	0.264	2
7940	5.000	6.547	0	1	0.264	3
9646	5.000	6.547	0	1	0.264	4
11351	5.000	6.547	0	1	0.264	5



0	5.000	6.547	0	0	0.264	0	1.000
705	5.000	6.547	0	0	0.264	0	1.000
1411	5.000	6.547	0	0	0.264	0	1.000
2116	5.000	6.547	0	0	0.264	0	1.000
2822	5.000	6.547	0	0	0.264	0	1.000
3527	5.000	6.547	0	0	0.264	0	1.000



6351	5.000	10.106	0	0	4.761	0	1.000
7057	5.000	10.106	0	0	4.761	0	1.000
7762	5.000	10.106	0	0	4.761	0	1.000
8468	5.000	10.106	0	0	4.761	0	1.000
9175	5.000	10.106	0	0	4.761	0	1.000
9880	5.000	10.106	0	0	4.761	0	1.000

3-

a) Para fazer o LED D1 piscar criou-se a função `piscaLed1()`, que recebe f (frequência) do tipo `float`. Em seguida, usou-se o `digitalWrite()` para se alternar o valor do LED entre máximo e mínimo. Como o período T é igual a $\frac{1}{f}$ e o período de T1=T2 e T1+T2=T, temos que T1=T2= $\frac{1}{2f}$ e como 1s=1000ms temos o resultado de `500/f`, valor do `delay` desejado.

```
void piscaLed1(float f){
  digitalWrite(PIND1,HIGH);
  delay(500/f);
  digitalWrite(PIND1,LOW);
  delay(500/f);
}
```

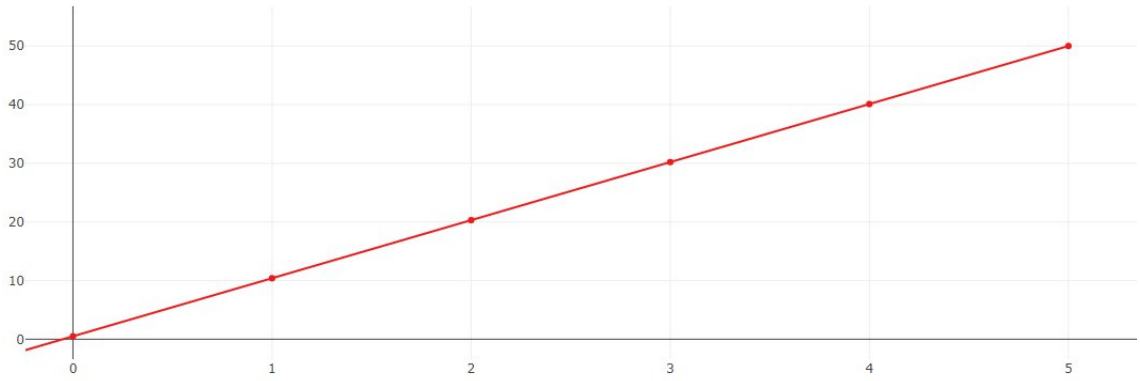
b) Fazer regular o brilho do LED D2 introduzindo o seu valor, fez-se o seguinte: Criou-se a função `regleituraD2()`, que recebe o valor inteiro do brilho num intervalo de 0 a 100. Depois criou-se uma variável do tipo `float`, `brilhoLed`, em que converte este valor para o intervalo de 0 a 255. Por fim, usou-se `analogWrite()` para gerar uma onda PWM.

```
float regleituraD2(int brilho){
  float brilhoLed= brilho*255/100;
  analogWrite(PIND2,brilhoLed);
}
```

4.

a) Para que o brilho de D2 fosse regularizado pelo valor enviado pela consola criou-se a função: `potLed1()`. A função segue a função do gráfico da frequência em função da tensão: $y = \frac{49.5}{5} * V2 + 0.5$. O valor de V2 é a variável enviada pela consola e a que regulariza o brilho de D2.

```
float potLed1(49.5/5*V2+0.5);
```



b) Para que o brilho de D2 fosse regularizado pelo valor enviado pela consola criou-se a função:

```
int lerserial(){
  if(Serial.available()>0){
    int aux=Serial.parseInt();
    if(aux>=0 && aux<=100){
      return aux;
    }
  }
  return -1;
}
```

c) Para incrementar um contador quando um dos botões foi premido, é necessário criar um `if` dentro do `void loop`, sendo que podemos lê-la da seguinte forma: se o interruptor S1 ou (||) interruptor S2 estiverem fechados (1) , uma variável que foi iniciada a 0 (contador) vai incrementando 1 num intervalo de 1 segundo devido ao delay de 1000 ms.

```
if((leituraV3()==1)||(leituraV4()==1)){
      contador=contador+1;
      delay(1000);
}
```

Conclusão:

O trabalho cumpriu o seu objetivo de introduzir os alunos ao trabalho com a placa Arduino. Contudo, esta revelou-se uma tarefa um pouco complicada, devido à inexperiência dos alunos.

Notas: As medições foram feitas no simulador Tinkercad, pois o grupo não as conseguiu realizar em laboratório. Houve problemas com o trabalhar com o osciloscópio, pois os alunos não sabiam trabalhar com ele.