Licenciatura Engenharia Informática e Multimédia

Instituto Superior de Engenharia de Lisboa

Ano letivo 2022/2023

Sensores e Atuadores

Relatório: Trabalho Lab04

Turma: 11N

Grupo: 0

Nome: Diogo Rodrigues	Número: 50776
Nome: Rodrigo Coelho	Número: 50251
Nome: Tatiana Damaya	Número: 50299

Data: 18 de Novembro 2022

Preparação teórica (anterior ao lab)

Verificar o bom funcionamento através do código exemplo ‘Blink’.

Pelo qual foi verificado corretamente por alterar os valores dos delays no qual produziu diferentes resultados.

Circuito 1: Potenciometro

Aqui devido há falta de um objeto viável para rodar o potenciómetro resolvemos por usar o Tinkercard para este circuito em questão no qual produzimos os seguintes valores.

Time(ms)	tensão(v)	pos(x)	pressed(0/1)	brilho(%)
32070	1.50	108.04	0	1.00
33073	1.50	108.04	0	1.00
34075	1.50	108.04	0	1.00
35077	1.50	108.04	0	1.00
36079	2.60	187.21	0	3.03
37082	4.00	287.86	0	14.39
38084	4.00	287.86	0	14.39
39087	4.00	287.86	0	14.39

Verifica-se a variância e correlação entre a posição de e a tensão em que nos casos com valores baixos no angulo a tensão também é baixa e vice-versa.

Circuito 2: Botão e pullup

Time(ms)	tensão(v)	pos(x)	pressed(0/1)	brilho(%)
2001	5.00	360.00	0	INF
3003	5.00	360.00	0	INF
4004	5.00	360.00	0	INF
5006	5.00	360.00	0	INF
6007	5.00	360.00	0	INF
7009	5.00	360.00	0	INF
8010	5.00	360.00	0	INF
9012	0.00	0.00	1	0.00
10012	0.00	0.00	1	0.00
11014	0.00	0.00	1	0.00
12014	0.00	0.00	1	0.00
13016	0.00	0.00	1	0.00
14017	0.00	0.00	1	0.00
15017	0.00	0.00	1	0.00

Aqui verificamos um valor curioso no brilho que é o INF (infinito), onde o seu valor máximo para quando o botão está solto (0) é infinito pois na sua prática é um valor bastante acima dos 100%, infinitamente superior a isso na verdade. Quando pressionamos o botão também verifica-se que a tensão passa imediatamente para 0 em V3 e por consequência a posição referida também passa para 0.

Circuito 3: Botão

Time(ms)	tensão(v)	pos(x)	pressed(0/1)	brilho(%)
10015	1.30	93.96	1	0.79
11017	1.29	92.90	1	0.79
12019	1.46	105.92	1	0.99
13021	1.43	101.70	1	0.89
14022	1.29	92.55	1	0.78

Aqui é de notar que quando o botão é premido a comunicação de entre as "pernas" do botão é interrompida, logo por sua vez a corrente também é cancelado, que por consequência o loop cessa.

Circuito 4: LDR

time(ms)	tensão(v)	posição(x)	pressed(0/1)	brilho(%)
493728	0.06	3.17	1	0.01
494730	0.04	2.82	1	0.01
495732	0.23	16.89	1	0.07
496734	0.63	45.40	1	0.28
497735	0.56	40.12	1	0.23
498737	0.43	31.32	1	0.17
499739	1.72	123.87	1	1.28
500741	2.82	203.05	1	3.75
501742	4.34	312.49	1	26.15
502744	4.44	319.53	1	32.15
503746	4.34	312.49	1	25.89
504748	4.36	314.25	1	27.54
505749	4.44	319.53	1	32.53
506750	4.48	322.70	1	36.28

Verificamos claramente que em casos de maior tensão traduz-se em maior brilho que era influenciado pelas lanternas dos nossos telemóveis que baixava a sua própria resistência para passar mais tensão (mais luz) e aumentava para passar menos tensão (menos luz).

Função para controlar o LED com base na frequência

```
// 3(a) Frequency value controls by a given tension D1 brightness
void controlLedByFrequency(float v) {
  float half_T = (500.0/0.5 - 500.0/50.0)/(-5.0) * v + 500/0.5;
  digitalWrite(PINLED1, HIGH);
  delay(half_T);
  digitalWrite(PINLED1, LOW);
  delay(half_T);
}
```

Esta função permite controlar o piscar de um LED 1 com base na frequência sua frequência em que esta mesmo varia consoante uma dada tensão (float v).
Primeiro fazemos uma regra de três simples para determinar o valor de metade do período (half_T), ou seja, o tempo que passa tanto no seu máximo ou no seu mínimo.
500 representa meio segundo em mil segundos, 0.5 é a sua mínima frequência, 50 a máxima frequência e por fim os 5 representam a tensão do circuito.
E depois de obtermos o valor de half_T passamos então o seu valor no delay para regular o período de blink do led em questão com o auxilio do digitalWrite que alterna entre HIGH e LOW por cada loop.

Função que regula o brilho em percentagem

```
// 3(b) test LED2 for it's brightness between 0 and 255, (0% - 100%)
void controlLedByPercentage(int percentage) {
  float brightness = (percentage * 100.0) / 255.0;
  analogWrite(PINLED2, brightness);
}
```

Nesta função controlamos o brilho do LED 2 com base num valor entre 0 e 100 e convertemos-lho num valor entre 0 e 255 para passar no segundo parâmetro do analogWrite que regula a luminosidade do componente.

Função para controlar o LED com o potenciómetro

```
// 4(a) Pot regulates the value of led D1
void controlLedByPot() {
  digitalWrite(PINLED1, HIGH);
  delay(voltage(PINPOT));
  digitalWrite(PINLED1, LOW);
  delay(voltage(PINPOT));
}
```

Usando o delay lógica para controlar o piscar do LED 1, passamos então a tensão (voltage) do ponteciometro (PINPOT) para extender ou encurtar o tempo de blink do componente.

Função de controlo do brilho do LED com o uso do input do user

```
// 4(b) Values from the console controls D2
void controlLedByConsole() {
  if(Serial.available()) {
    int input = Serial.parseInt();
    luminosidade = input * 2.55;
  }
  analogWrite(PINLED2, luminosidade);
}
```

Neste método verificamos primeiro se existe algum input do usuário (Serial.available()) e se sim então chamamos o Serial.parseInt() para obtermos o inteiro inserido compreendido entre 0 e 100, para então convertermo-lo num valor entre 0 e 255 (luminosidade) com o intuito de passarmos o valor para um analogWrite que irá alterar a potência luminosa no próximo loop.

Função que incrementa um valor quando o botão está premido

```
// 4(c) Increments by pressing a button
int incrementPressed() {
  int btnState = digitalRead(PINBTN);
  if (btnState == 1) {
    count++;
  }
  delay(1000);
  return count;
}
```

Começamos por verificar o estado do botão (btnState), se premido ou solto (0 ou 1). E se este estiver premido, ou seja, se tiver o valor 1, então incrementa-mos a variável inteira count, por fim fazemos um delay de 1000 para que este só ocorra a cada segundo, finalizando com a entrega do valor de count.