

Física Moderna

Difração de Múltiplas Fendas

Luiz Fernando Gomes de Oliveira- 10/46969

Lucas Severo Alves- 10/0034772

Leonardo- xx/yyyyy

Resumo

Este experimento tem como objetivo apresentar o experimento de Thomas Young - experimento da fenda dupla. Este experimento demonstra como a luz em específico pode se comportar como partícula e onda ao mesmo tempo, o que gera uma grande dificuldade de compreensão de sua composição.

This experiment aims to present the experiment of Thomas Young - double slit experiment. This experiment demonstrates how light can behave as specific particle and wave at the same time, which creates a great difficulty to understand its composition.

Index Terms

Física Moderna, Arduino, Difração de Múltiplas Fendas, Thomas Young, Double-slit experiment, ondas, partículas



SUMÁRIO

1	Criação do Circuito	1
1.1	Ponte de Wheatstone	1
1.2	Abstração da ponte para o circuito final	1
2	Levantando o programa	2
2.1	Shield Comum	2
	Referências	2
	Apêndice A: Códigos Fontes	2
	Apêndice B: Anexos	3

LISTA DE FIGURAS

1	Ponte de Wheatstone	1
2	Circuito do LDR	1
3	Circuito proposto do LDR	2
4	Idealização do circuito do LDR	3
5	Fenda dupla	4
6	Difração em fenda simples	4
7	Difração em fenda dupla	5

LISTA DE TABELAS

LISTINGS

1	main.c	2
---	------------------	---

1 CRIAÇÃO DO CIRCUITO

PARA a criação do circuito, precisamos inicialmente entender como funciona uma ponte de *Wheatstone*.

1.1 Ponte de Wheatstone

A ponte de *Wheatstone* é utilizada quando queremos balancear um circuito com uma resistência desconhecida. A ponte é comumente caracterizada pelo desenho na figura 1, onde pode-se obter uma corrente no amperímetro igual a zero quando a seguinte equação é respeitada:

$$R_1 \cdot R_x = R_3 \cdot R_2 \quad (1)$$

Quando a equação 1 é respeitada, as tensões entre os pontos **A** e **B** é a mesma, de forma que não existe corrente nestes pontos

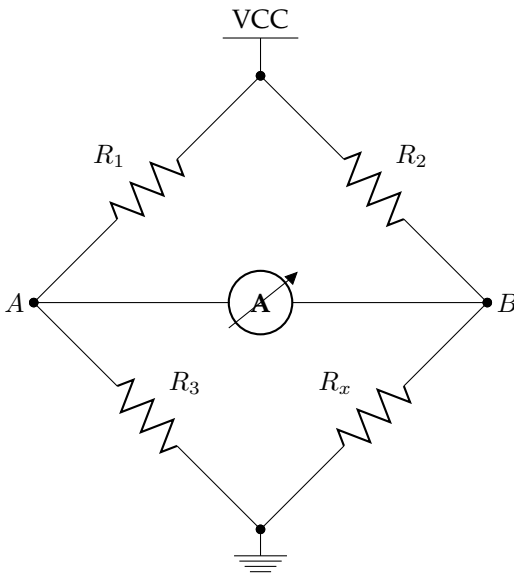


Figura 1: Ponte de Wheatstone

1.2 Abstração da ponte para o circuito final

Com base no conhecimento obtido na sessão 1.1, podemos inferir que no circuito da figura 2 teremos equilíbrio se:

$$R_1 \cdot R_x = R_3 \cdot R_2, \text{ sendo}$$

$$R_x = \frac{(R_3 + LDR) \cdot R_4}{(R_3 + LDR) + R_4}$$

Para uma maior facilidade em montar o circuito, podemos considerar que $R_1 = R_2 = R_3 = R$. Outra fator importante é considerar o range da resistência que o *LDR* adota de acordo com a intensidade de luz.

Normalmente, um *LDR* possui resistência em torno de 100Ω quando exposto a luz e $10^6\Omega$ quando em completa escuridão.

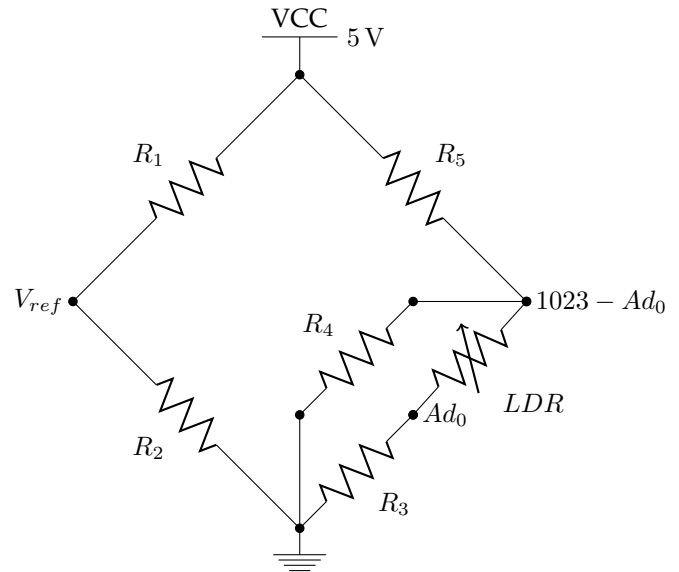


Figura 2: Circuito do LDR

Com base neste range e, adotando um valor para $r = 10K\Omega$, podemos chegar as seguintes equações:

$$R \cdot R_x = R \cdot R$$

$$R_x = R$$

$$R_x = \frac{(R_3 + LDR) \cdot R_4}{(R_3 + LDR) + R_4}$$

$$R_x = \frac{(R_3 + LDR) \cdot R_4}{(R_3 + LDR) + R_4} = 10^3$$

$$10^3 = \frac{(R_3 + LDR) \cdot R_4}{(R_3 + LDR) + R_4}$$

Tal que: $10^2\Omega < LDR < 10^6\Omega$

Podemos observar que assumindo $R_3 = 8.2 \cdot 10^3\Omega$, temos o valor de R_4 muito próximo para ambos os casos (10^3 e $1.2 \cdot 10^3$). Assim, uma possibilidade para o circuito final que abranja quase todo o range de resistência do *LDR* é o seguinte circuito:

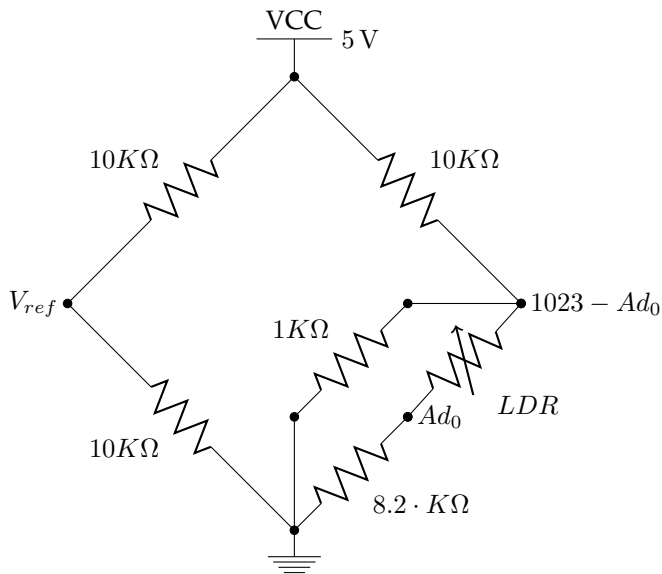


Figura 3: Circuito proposto do LDR

2 LEVANTANDO O PROGRAMA

Inicialmente, é importante frisar que o kit do Arduino nada mais é que uma camada de abstração criada acima do microcontrolador AVR. Desta forma, podemos programar nos kits do Arduino de duas formas sob o sistema Linux:

2.1 Shield Comum

A forma mais simples de utilizar um kit do Arduino é utilizando a própria IDE oferecida para o kit. A sua instalação é bem simples e fácil de ser feita. Basta entrar com o seguinte comando no terminal:

```
1 user@DESKTOP: sudo apt-get install arduino
```

REFERÊNCIAS

APÊNDICE A CÓDIGOS FONTES

Listing 1: Código principal

```

1 /**
2  * Test model for Arduino - UNO
3  */
4
5 #include <Arduino.h>
6 #include <stdio.h>
7 #include <stdlib.h>
8
9 /**
10 *
11 * The setup() method runs once, when the
12 * sketch starts
13 * This is just starting the UART in 9600
14 * and setting the reference for the analog
15 * as External
16 *
17 */
18 void setup()
19 {
20     // wait for 10ms
21     delay(10);
22     //! initialize the default debug output:
23     Serial.begin(9600);
24     //! Setting the analog reference as
25     //    external
26     analogReference(EXTERNAL);
27     // wait for 100ms to stabilize
28     delay(100);
29 }
30 /**
31 *
32 * The loop() method runs over and over again
33 * as long as the Arduino has power
34 *
35 */
36 void loop()
37 {
38     static int i=0;
39     char buf[20];
40
41     //! raw values
42     sprintf(buf, "%d %d\n", i++, analogRead(0));
43     Serial.print(buf);
44
45     delay(1); // wait for lms
46 }
47
48 /**
49 * This is the standard main function. On
50 * Arduino, it's NEVER changes.
51 * @return the standard returns that the main
52 * should return.
53 */
54 int main(void)
55 {
56     //! Initialize the pins I/O and standard
57     //    configures.
58     init();
59     //! Call the user's setup function.
60     setup();
61
62     //! The eternal loop
63     while(true) {
64         loop();
65     }
66
67     return 0;
68 }
```

APÊNDICE B

ANEXOS

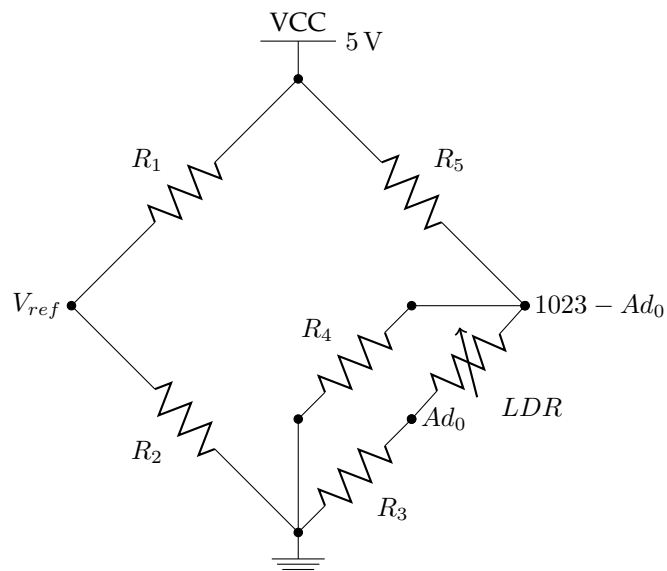
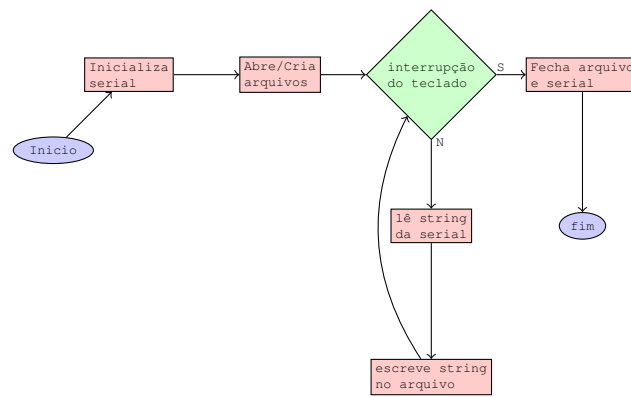
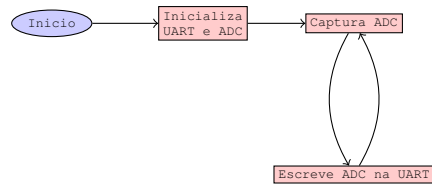


Figura 4: Idealização do circuito do LDR

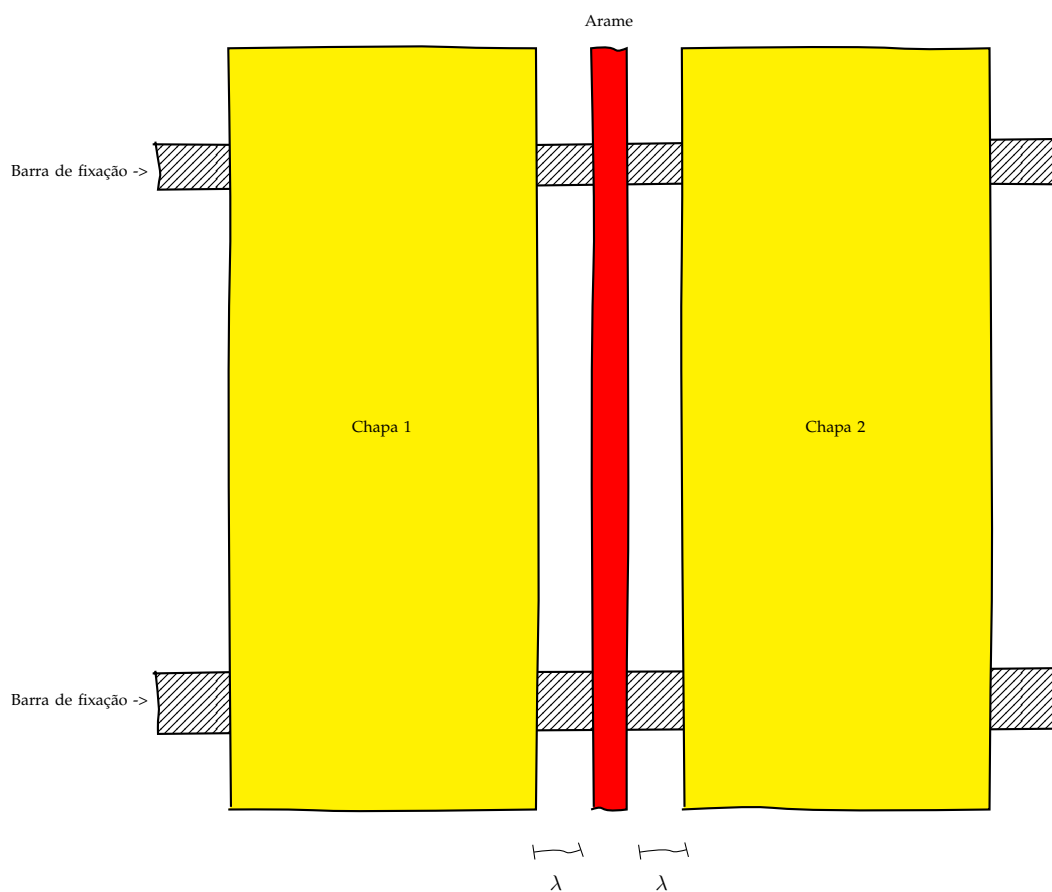


Figura 5: Fenda dupla

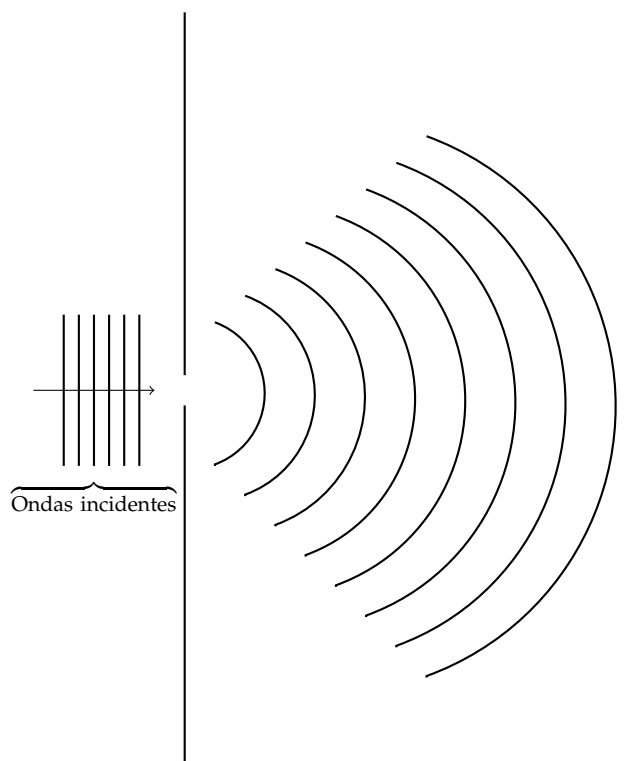


Figura 6: Difração em fenda simples

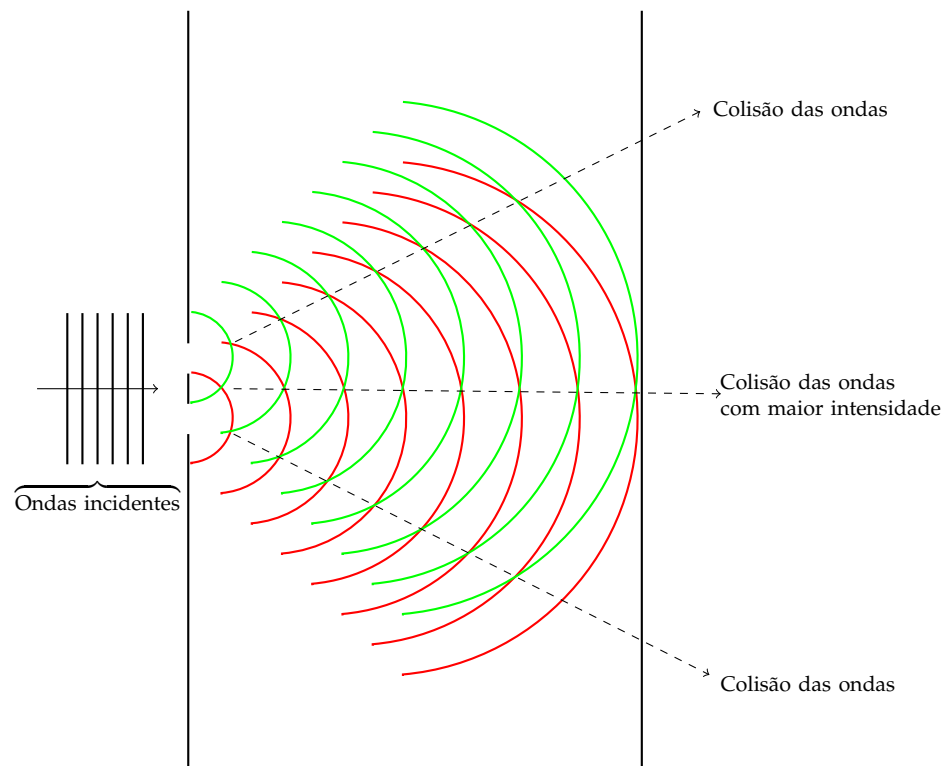


Figura 7: Difração em fenda dupla