Microcontrolador Arduino UNO

Raul Ramires[1]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – PR – Brazil

ra82293@uem.br

1. Introdução

O arduino é uma placa microcontroladora baseada no microcontrolador ATMega e pode ser integrada com uma grande diversidade de componentes e sensores. O projeto a ser desenvolvido propõe a configuração do ambiente de trabalho e a realização de diversos experimentos com diferentes sensores.

2. Ambiente de Trabalho

O arduino possui uma IDE *Integrated Development Environment* chamada de "Arduino IDE", que é onde serão escritos os códigos para que depois sejam enviados ao arduino e executado.

2.1. Download da IDE

Para fazer o download da IDE basta ir ao site do arduino e escolher a versão de acordo com o sistema operacional utilizado, estando disponível para Windows, Linux e MacOS.

3. Programação

A programação do arduino é feita na linguagem C++ e possui uma grande quantidade de bibliotecas prontas para a utilização de sensores.

3.1. Estrutura do código

A Figura 1 mostra a estrutura básica de um código para arduino. Esse código é dividido em duas funções: a função *setup* e a função *loop*.

```
void setup() {
  //Executa apenas uma vez
}

void loop() {
  //Executa indefinidamente
}
```

Figura 1. Estrutura do código.

A função *setup* será executada apenas uma vez no início do programa, geralmente é aqui que ficam as definições de pinos de entradas e saídas e algumas variáveis globais. A função *loop* será executada infinitamente enquanto o arduino estiver ligado, é aqui que serão feitas as leituras de sensores e decisões de ações com bases nos dados lidos dos sensores.

3.2. Upload do código

Para realizar o *upload* do código para a placa do arduino, basta clicar no botão “carregar”, então o código será compilado e carregado para o arduino e então começar a execução.

4. Experimentos

Nessa seção serão realizados vários experimentos com o arduino, utilizando uma grande diversidade de componentes e sensores.

Todos os experimentos realizados nesse trabalho estão disponíveis no repositório no *GitHub* e os esquemáticos dos circuitos estão disponíveis no *Tinkercad*.

4.1. Piscar um LED

Para realizar esse experimentos será necessário um LED, uma *protoboard* e dois *jumpers*.

A *protoboard* é construída como uma matriz, onde cada coluna possui 5 pontos de contato que são interligados entre si, porém uma coluna é isolada de sua vizinha, sendo necessário a utilização de um *jumper* para interligar duas colunas.

Alguns modelos de *protoboard* possuem dois barramentos laterais, um negativo e um positivo que percorrem a *protoboard* inteira. O padrão de utilização desses barramentos é ligar o 5V no barramento vermelho e o GND no barramento preto, ficando assim mais simples de realizar a alimentação dos componentes utilizados.

A Figura 2 mostra o esquema de uma *protoboard* de 360 pontos.

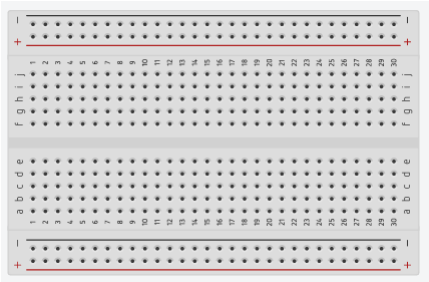

Figura 2. *Protoboard* de 360 pontos

A Figura 3 mostra o esquema do circuito para piscar um LED, utilizando o arduino.

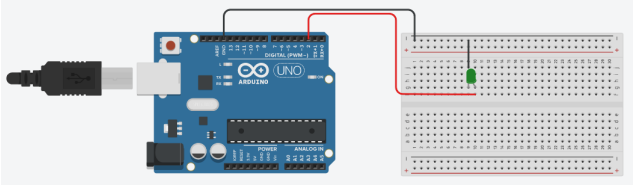

Figura 3. Esquemático do circuito do experimento 1.

Nesse circuito o pino digital 2 do arduino está ligado na coluna 10 da protoboard, onde, nessa mesma coluna está ligado o terminal positivo do LED. O terminal negativo do LED está ligado na coluna 9, que possui um *jumper* para o barramento preto, que por sua

vez está ligado ao GND do arduino. Sendo assim o LED irá acender quando o arduino enviar o sinal de 5V no pino 2.

A Figura 4 mostra o código que fará com que o arduino pisque o LED.

```
1  //Definição do pino positivo que será ligado ao LED
2  int pinoLED = 2;
3
4  void setup() {
5    //Executa apenas uma vez
6
7    //Definição do pino pinoLED como saída
8    pinMode(pinoLED, OUTPUT);
9  }
10
11 void loop() {
12   //Executa indefinidamente
13
14   //Envia um sinal alto (5V) para o pinoLED
15   digitalWrite(pinoLED, HIGH);
16
17   //Espera 1 segundo
18   delay(1000);
19
20   //Envia um sinal baixo (0V) para o pinoLED
21   digitalWrite(pinoLED, LOW);
22
23   //Espera 1 segundo
24   delay(1000);
25 }
```

Figura 4. Código do experimento 1.

4.2. Mudar o brilho de um LED

Para realizar esse experimento será necessário um LED, um potenciômetro e um resistor de 220Ω.

O potenciômetro é um resistor capaz de variar sua resistencia, para esse experimento, o arduino irá ler esse valor de resistência e associar a um valor de tensão e enviar ao LED, variando assim o brilho do LED. Como esse valor é variável, é necessário conectar o potenciômetro a uma porta analógica do arduino.

A Figura 5 mostra o esquemático do circuito do experimento.

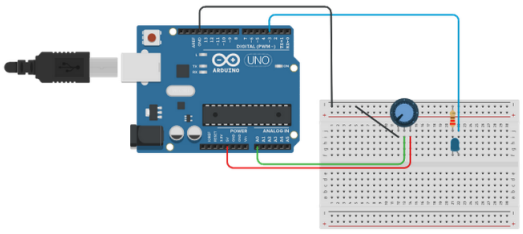

Figura 5. Esquemático do circuito do experimento 2.

O arduino faz a leitura do potenciômetro em um valor de 0 a 1023. Para usar esse valor para acender o LED, é necessário fazer uma conversão para um valor de 0 a 255. Para isso é utilizado a função *map*, que é responsável por fazer esse mapeamento de um intervalo para outro.

A figura 6 mostra o código do experimento 2.

```ino
//Definição dos pinos
int pinoPOT = A0; //Pino analogico do potenciometro
int pinoLED = 3; //Pino PWM do LED

float luminosidade;

float valorPOT;

void setup() {
  pinMode(pinoPOT, INPUT); //Pino de entrada
  pinMode(pinoLED, OUTPUT); //Pino de saida

}

void loop() {
  valorPOT = analogRead(pinoPOT); //Le o valor do potenciometro

  luminosidade = map(valorPOT, 0, 1023, 0, 255); //faz o mapeamento do valor do potenciometro para 0-255

  analogWrite(pinoLED,luminosidade); //envia o sinal para o LED
}
```

Figura 6. Código do experimento 2.

4.3. Sensor de luz

Esse experimento tem como objetivo controlar uma barra de LEDs a partir de um LDR (*Light Dependent Resistor*). O LDR é um resistor que varia sua resistência de acordo com a luz que incide sobre ele, quanto maior a quantidade de luz, menor a resistência do LDR. Como o LDR é um componente que possui um valor de resistência variável, é necessário conectá-lo a uma porta analógica do arduino.

Para esse experimento, quanto maior a luz incidente sobre o LDR, menos LEDs irão acender.

A Figura 7 mostra o esquemático do circuito do experimento 3.

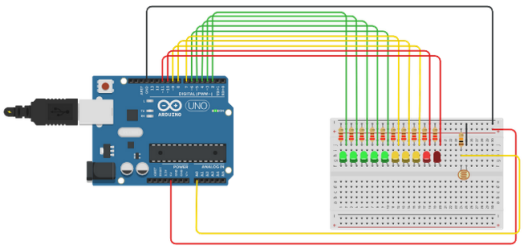

Figura 7. Esquemático do circuito do experimento 3.

A Figura 8 mostra o código do experimento 3.

```ino
int pinLDR = A0; //pino do LDR
int ledCount = 10; //quantidade de LEDs

int ledPins[] = {
    2,3,4,5,6,7,8,9,10,11
}; //array de pinos onde estao conectados os LEDs

void setup(){
    int led;
    for(led = 0; led < ledCount; led++){
        pinMode(ledPins[led], OUTPUT); //Define os pinos dos LEDs como saida
    }
}

void loop(){
    //leitura do valor do LDR
    int estadoLDR = analogRead(pinLDR);

    //mapeia o valor do LDR para uma quantidade entre 0 e ledCount
    int ledLevel = map(estadoLDR, 0, 1023, 0, ledCount);

    int led;

    //Itera sobre os leds
    for(led = 0; led < ledCount; led++){
        //se o indice for menor que ledLevel
        if(led < ledLevel){
            //acende o LED
            digitalWrite(ledPins[led], HIGH);
        }
        //se o indice for maior que ledLevel
        else{
            //apaga o led
            digitalWrite(ledPins[led],LOW);
        }
    }
}
```

Figura 8. Código do experimento 3.

4.4. Display de 7 Segmentos

Esse experimento faz o controle de um display de 7 segmentos utilizando um botão para exibir os dígitos de 0 a 9. O display incia exibindo o dígito 0 e incrementa um dígito quando o botão é pressionado. Os segmentos do display são nomeados de acordo com a Figura 9 [Pattabiraman 2017].

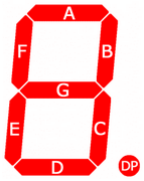

Figura 9. Segmentos do display.

Para acender um segmento o arduino deve enviar um sinal alto para o segmento, e para apagar deve enviar um sinal baixo.

A Figura 10 mostra a pinagem do display. Para esse experimento não será utilizado o ponto decimal.

Figura 10. Pinagem do display de 7 segmentos.

O botão é definido como *INPUT_PULLUP* para utilizar um resistor interno do arduino e simplificar o circuito, por conta disso, o arduino interpreta o botão como estado alto quando não pressionado e estado baixo quando pressionado. Para detectar o acionamento do botão é necessário considerar o estado atual do botão e o estado anterior, pois o arduino faz a leitura do botão um grande número de vezes por segundo, sendo assim é necessário esse tratamento para evitar que o arduino interprete que o botão foi acionado diversas vezes em apenas um acionamento.

A Figura 11 mostra o trecho de código que faz a leitura do botão.

```
54 void loop(){
55     int i;
56     //faz a leitura do pino do botao
57     //por conta da utilizacao do PullUP o botao fica no estado
58     //HIGH quando nao for apertado
59     //LOW quando for apertado
60     estadoAtual = digitalRead(pinBotao);
61 
62     //detecta se o botao foi apertado
63     if(estadoAtual == LOW && estadoAnterior == HIGH){
64         //incrementa o digito
65         digito++;
66     }
67 
68     estadoAnterior = estadoAtual;
```

Figura 11. Código que detecta o acionamento do botão.

Referências

[Pattabiraman 2017] Pattabiraman, K. (2017). How to set up 7-segment displays on the arduino. Acessado em: 15 de junho de 2019.