Arduino Sensor Kit da Yahboom Technology: Tutorial para Experimentos

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.

O Arduino Sensor Kit da da Yahboom Technology, utilizado é um kit de componentes e sensores baseado no Arduino UNO R3. O objetivo do presente tutorial é o de apresentar os passos e explicações de como realizar experimentos com o referido kit.

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 https://www.arduino.cc/en/Main/Software e escolher a versão de acordo com o sistema operacional utilizado, estando disponível para Windows, Linux e MacOS.

A Figura 1 mostra a tela principal da IDE do arduino.

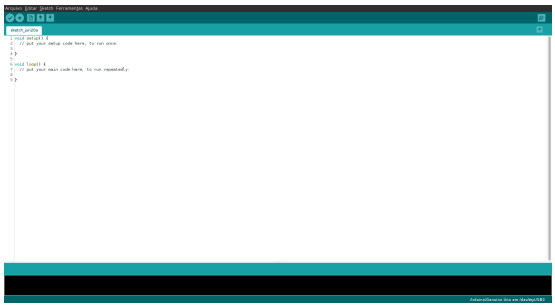

Figura 1. Tela principal da Arduino IDE.

2.2. Arduino Sensor Kit

O kit de sensores para arduino UNO da Yahboom contém os seguintes componentes:

Protoboard Placa de prototipação de 830 pontos;

Arduino UNO Microcontrolador Arduino UNO R3;
Bateria 9V Bateria para alimentação do arduino;
Motor de Passo Motor de passo alimentado por 5V;
Motor DC Motor de corrente contínua;
Servo Motor Micro Servo 9g;
Matriz de LED Matriz 8x8 de LEDs (1588BS);
Display 7 Segmentos Módulo com um display de 7 segmentos (5161AS);
4 Displays de 7 Segmentos Módulo com 4 displays de 7 segmentos (3461AS-1);
Hélice Hélice de plástico com 4 pás;
LDR Resistor dependente de luz;
Push Button Botões de pressionamento;
Sensor de Presença Sensor infravermelho de presença;
Sensor de Gás Sensor de gás *Flying-Fish*;
Sensor Gray Scale Sensor de escala de cinza;
Joystick Módulo joystick de 3 eixos;
Sensor de Som Módulo de sensor de som;
LED Flashlight Módulo composto por 2 LEDs RGB;
Sensor Ultrasônico Sensor ultrasônico de distância (HC-SR04);
Display LCD Display LCD 16x2;
Potenciômetro Resistor varável de 5kΩ;
Ponte H Módulo para controle de motores;
LEDs 15 LEDs coloridos;
Resistores Resistores de 220Ω, 1kΩ e 10kΩ;
Buzzer Buzzer ativo para emitir som.

A Figura 2 mostra os componentes descritos acima.

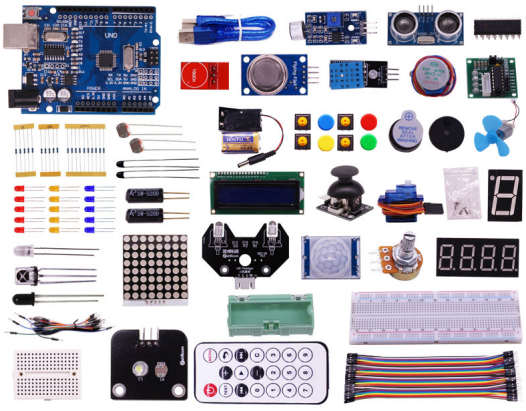

Figura 2. Sensores presentes no kit.

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 3 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 3. 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.

A Figura 4 mostra em destaque o botão para carregar o código para o arduino.

Figura 4. Destaque do botão de carregar.

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 5 mostra o esquema de uma *protoboard* de 360 pontos.

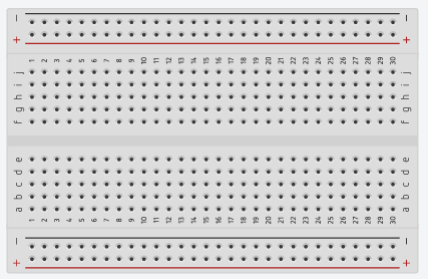

Figura 5. *Protoboard* de 360 pontos

A Figura 6 mostra o esquema do circuito para piscar um LED, utilizando o arduino.

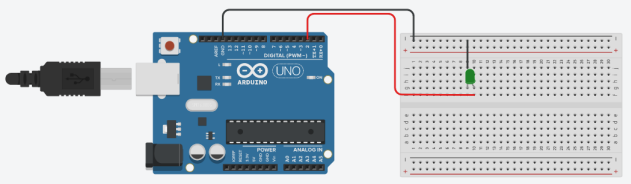

Figura 6. Esquemático do circuito do experimento 1.

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 7 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 saida
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 7. 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 8 mostra o esquemático do circuito do experimento.

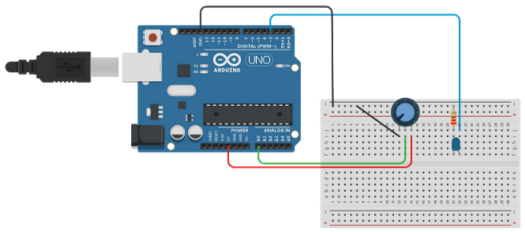

Figura 8. Esquemático do circuito do experimento 2.

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 9 mostra o código do experimento 2.

```
//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 9. 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 10 mostra o esquemático do circuito do experimento 3.

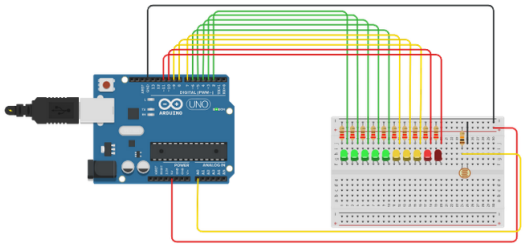

Figura 10. Esquemático do circuito do experimento 3.

A Figura 11 mostra o código do experimento 3.

```
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 11. 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 12 [Pattabiraman 2017].

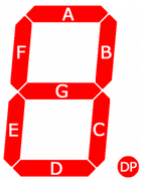

Figura 12. Segmentos do display.

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

A Figura 13 mostra a pinagem do display. Para esse experimento não será utilizado o ponto decimal.

Figura 13. 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 14 mostra o esquemático do circuito do experimento 4.

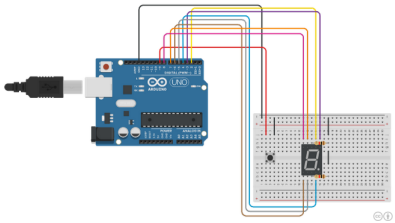

Figura 14. Esquemático do experimento 4.

A Figura 15 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 15. Código que detecta o acionamento do botão.

4.5. Matriz de LEDs 8x8

Esse experimento tem como objetivo percorrer os LEDs da matriz utilizando um joystick.

Como o *Tinkercad* não possui os módulos da matriz e do joystick, esse experimento não estará disponível para simulação.

A Figura 16 mostra a numeração de pinos da matriz de LEDs [Oliveira 2017].

Figura 16. Numeração de pinos da matriz.

A Tabela 1 mostra como fazer a conexão das linhas da matriz com o arduino.

Linha	Pino Matriz	Pino Arduino
1	9	2
2	14	3
3	8	4
4	12	5
5	1	6
6	7	7
7	2	8
8	5	9

Tabela 1. Conexão das linhas da matriz no arduino.

A Tabela 2 mostra como fazer a conexão das colunas da matriz com o arduino.

Coluna	Pino Matriz	Pino Arduino
1	13	10
2	3	11
3	4	12
4	10	13
5	6	A0
6	11	A1
7	15	A2
8	16	A3

Tabela 2. Conexão das colunas da matriz no arduino.

A Figura 17 mostra a numeração de pinos do módulo joystick.

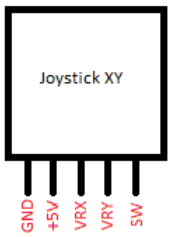

Figura 17. Numeração de pinos do joystick.

Para ligar o joystick ao arduino basta ligar o *GND* do módulo ao *GND* do arduino, o +5V do módulo ao +5V do arduino. Os pinos VRX e VRY são ligados aos pinos A4 e A5, respectivamente.

4.6. Sensor de Distância

Esse experimento consiste em utilizar o sensor ultrasônico de distância para exibir a distância medida no display LCD. Esse experimento possui uma medida de distância mínima que pode ser regulada utilizando um potenciômetro. Quando a distância medida for menor que a distância mínima será soado um alarme sonoro.

O circuito possui ainda um segundo potenciômetro necessário para a regulagem do brilho do display.

Para esse experimento será necessário a utilização de duas bibliotecas, a **LiquidCrystal.h** que é responsável por controlar o display e a biblioteca **Ultrasonic.h** que controla o sensor ultrasônico de distância.

A biblioteca **LiquidCrystal.h** já está contida na IDE do arduino. Já a biblioteca **Ultrasonic.h** deve ser baixada e instalada manualmente.

Originalmente a biblioteca **Ultrasonic.h** foi desenvolvida pelo site **FilipeFlop** [Thomsen 2011]. A biblioteca está disponível no repositório do *GitHub* para fácil acesso.

Para adicionar a biblioteca ao Arduino IDE basta clicar em *Sketch* $\rightarrow$ *Incluir Biblioteca* $\rightarrow$ *Adicionar Biblioteca .ZIP* e selecionar o arquivo *Ultrasonic.zip* presente na pasta Bibliotecas.

Como não é possível instalar bibliotecas adicionais no *Tinkercad*, esse experimento não está disponível para simulação, porém é possível ver o esquemático do circuito no site, como mostrado na Figura 18.

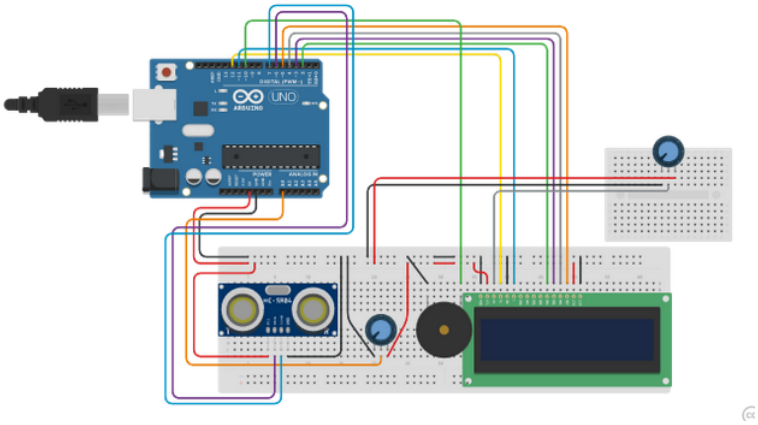

Figura 18. Esquemático do experimento 6.

Referências

[Oliveira 2017] Oliveira, E. (2017). Arduino – utilizando a matriz de led 5mm 8x8. Acessado em: 20 de junho de 2019.

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

[Thomsen 2011] Thomsen, A. (2011). Como conectar o sensor ultrassônico hc-sr04 ao arduino. Acessado em: 7 de julho de 2019.