UMA PROPOSTA DE PROGRAMAÇÃO COM O ARDUINO IDE E A MATEMÁTICA

Daniel da Silveira Guimarães[1]

RESUMO

As transformações tecnológicas trazem o desafio de repensar o processo de ensino e aprendizagem de Matemática, sendo necessário incorporar essas inovações. Nesse sentido, propomos a realização deste curso de Robótica Educacional, que busca desenvolver conteúdos de Matemática, a partir da construção e da programação de circuitos eletrônicos capazes de desenvolver ações do nosso cotidiano. Neste trabalho, iremos explorar a programação de microcontroladores justificando que o entendimento de ferramentas de Matemática, como por exemplo, o conceito de função, pode tornar mais simples a compreensão dos conceitos de programação, que se baseiam na Matemática, como tudo em tecnologia.

Palavras-chave: Robótica Educacional; Programação Arduino e a Matemática.

1 O ARDUINO

O Arduino é uma plataforma de prototipagem eletrônica de hardware livre e de placa única, que contém um microcontrolador. No caso do Arduino UNO, o microcontrolador é o Atmega328 da família da Atmega, veja na figura 1. Essa placa torna mais simples a inserção da programação e a execução de atividades que podem ser realizadas pelo microcontrolador Atmega328, pois possui, integrado a ela, o programador do microcontrolador.

Os microcontroladores precisam dos sensores, por exemplo, sensor de distância, fotoresistor, botões, microfones, entre outros, e dos atuadores como LED's, relés, motores, entre outros. Os sensores ouvem o mundo físico, transformam essas informações em corrente elétrica, tornando capaz a leitura do microcontrolador, que então executa alguma ação, usando a corrente elétrica, no mundo físico através dos atuadores.

[1] Universidade Federal de Catalão. E-mail danielguimaraes@ufcat.edu.br.

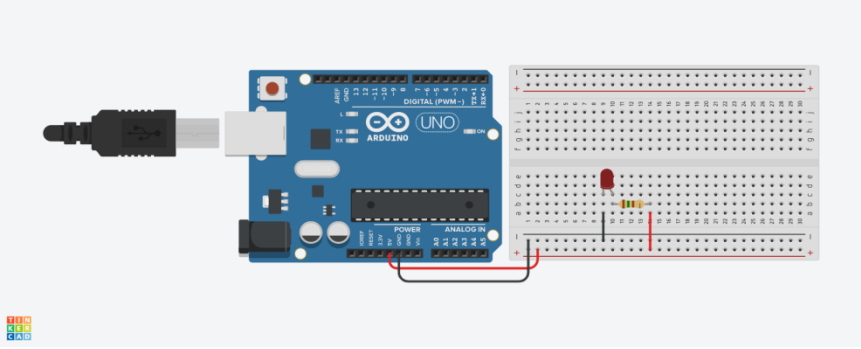
O Arduino possui 5 pinos analógicos, que trabalham com tensões que variam de 0 volt até 5 volts e 14 pinos digitais, que trabalham com o valor de 0 volt (desligado) ou 5 volts (ligado), dos quais 6 podem ser usados como saídas PWM. Além disso, possui pinos de 5 volts, GND (negativo) e 3,3 volts, entre outros, veja na figura 1.

2 A ELETRÔNICA

Iniciamos essa seção com uma breve explanação sobre as grandezas em um circuito eletrônico, que são: o diferencial de potencial entre dois pontos com uma diferença de elétrons, também denominada de tensão elétrica, é o que torna capaz o movimento de uma carga elétrica entre eles, cuja unidade de medida é o volts; a corrente elétrica, que é um fenômeno que ocorre nos condutores elétricos quando suas cargas se movem de forma ordenada devido a uma tensão gerada por uma diferença de potencial, cuja unidade de medida é em Amperes, e a resistência elétrica, cuja unidade de medida é Ohm, que é a capacidade de um corpo de se opor a passagem de corrente elétrica, mesmo quando existe uma diferença de potencial aplicada. É importante salientar que, convencionalmente, a corrente elétrica sempre percorre um circuito no sentido do positivo para o negativo, sendo fundamental a qualquer circuito se iniciar em um e terminar no outro, para ocorrer a passagem da corrente elétrica.

Após uma breve explanação dos componentes como protoboard, LED e resistor, é possível montar uma proposta de um circuito que inicia no pino de 5 volts, passa pelo resistor e LED e finaliza no GND, para acender o LED. Veja na figura abaixo a proposta simulada no Tinkercad.

Figura 1 – Circuito Arduino, LED e resistor de 150 ohm



Fonte: Autoria própria utilizando o software Tinkercad.

Porém, qual resistor utilizar para não danificar o LED, que opera com tensões entre 2 volts e 3 volts? A Matemática que nos permite definir esse resistor pode ser encontrada em Guimarães et al., 2020.

3 A PROGRAMAÇÃO NO ARDUINO IDE

Para converter um programa escrito e uma linguagem de alto nível para a linguagem de máquina, utilizamos os compiladores, sendo necessário utilizar o Arduino Integrated Development Environment (Arduino IDE), que é um aplicativo de computador que possui um compilador integrado, onde você pode escrever o seu programa, compila-lo e carregá-lo na placa do Arduino, para programarmos o Arduino. Veja figura 2. Podemos também utilizar o simulador Tinkercad para realizar as atividades propostas neste trabalho, sem a necessidade de obter o Arduino e os componentes.

Figura 2 – Arduino IDE

Fonte: Autoria própria.

O algoritmo ou simplesmente programa, é uma forma de dizer para um computador o que ele deve fazer. Os algoritmos normalmente são escritos em linguagens de programação de alto nível e seguem uma sequência lógica de comandos. A linguagem de programação utilizada no Arduino IDE é o C++, com pequenas modificações.

O Arduino IDE possui uma série de bibliotecas e funções já prontas para acessar as configurações do microcontrolador, facilitando a comunicação deste com os sensores e atuadores, encapsulando toda a complexidade do microcontrolador.

Todas as sequências lógicas e regras do C++ serão utilizadas, como por exemplo, a forma que declaramos qualquer nome utilizado em nosso programa, fazendo uso das variáveis, que podem ser, por exemplo, “int” para números inteiros. O significado de “int led;” é a declaração de um nome led que será um valor inteiro. Há um catálogo de tais variáveis, traremos aqui a que utilizaremos nessa proposta: boolean, quando a variável assume somente dois valores, sendo o valor verdadeiro (true em inglês) ou o valor falso (false em inglês), por exemplo “boolean botaoapertado = false;” double, quando a variável é um número real de precisão dupla (ponto flutuante) e void, quando a variável é do tipo vazio (não tem tipo).

Outro fato importante, a atribuição de certa variável ou a outros objetos da programação é dado pelo sinal de =. Por exemplo, ao informar que “led = 7;” estamos atribuindo a variável inteira led o valor 7. Temos também os operadores: + para adição, - para subtração, * para multiplicação e / para a divisão e os operadores, && para conjunção e, == para a igualdade, =! para atribuir um valor diferente do atual, entre outros.

Finalmente, o ponto e vírgula na programação é usado para encerrar um comando, ou seja, encerrar uma linha da programação. As duas barras // serve para realizar comentários que não serão considerados pelo compilador naquela linha a frente delas e, caso deseje comentar textos maiores, basta fazer a inserção do texto entre /* e */, exemplo “/*TEXTO*/”.

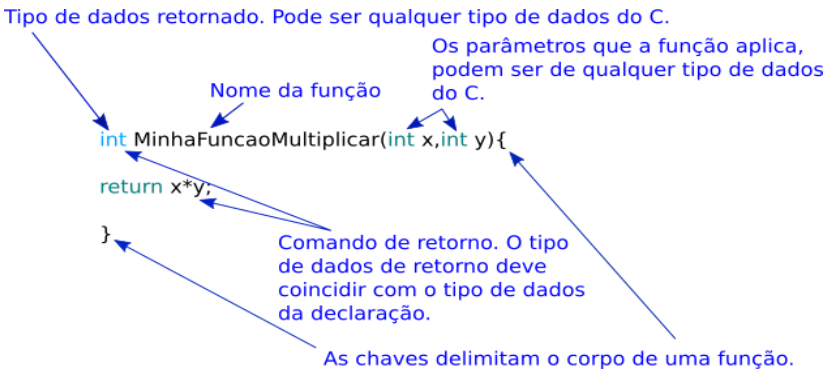
4 A MATEMÁTICA NA PROGRAMAÇÃO

Após as introduções e definições básicas, estamos aptos a verificar o quanto a Matemática se faz presente na programação. Um dos conceitos mais importantes em programação é o conceito de função, que herdado da Matemática, facilita todo o processo de comunicação com a máquina. Vejamos um exemplo de função em Matemática.

Exemplo: $f(x, y) = xy$. Essa função, aplicada em duas variáveis, retorna a multiplicação. Assim, $f(2{,}10) = 20$, $f(3{,}11) = 33$.

Em programação, os valores do domínio entram entre os parênteses () e a imagem, entre as chaves {}. Com essa regra, em Matemática, a definição da função supracitada seria $f(x, y)\{xy\}$.

Figura 3 – Anatomia de uma função em C



Fonte: Autoria própria.

Após a definição dessa função, podemos simplesmente utilizá-la na programação. Para o exemplo da Figura 3, faz-se isso com "MinhaFuncaoMultiplicar(2,10);" obtendo o valor de 20.

No caso do Arduino, temos duas funções nativas, a função setup e a função loop. A primeira é necessária para definir os pinos ou inserir alguma informação que será lida somente ao iniciar o microcontrolador. Já a segunda função, é importante pela forma que os microcontroladores trabalham, pois eles estão sempre esperando alguma ação externa, logo precisam estar sempre repetindo a leitura das ações programadas. O loop, que significa ciclo em inglês, faz essa leitura em ciclo. No caso do Arduino, as funções setup e loop se são inseridas automaticamente pelo Arduino IDE e são declaradas como void (tradução vazio), ou seja, sem retorno, veja Figura 5.

Quando for definir alguma outra função no Arduino, precisa realizar essa definição fora de void setup(){} e void loop(){} e chamá-la dentro de uma dessas funções, entre as chaves.

Além dessas duas funções, o que torna o Arduino IDE simples de se programar são as várias funções já declaradas em sua biblioteca, que facilita o processo. Vamos aqui falar de algumas funções já declaradas na memória do Arduino IDE, bastando somente serem chamadas na programação. Por exemplo, a função que define o modo do pino, chamada de pinMode, que aplicada em duas variáveis (pino,Modo), nos permitem definir um certo pino do microcontrolador como o modo de saída (OUTPUT em inglês) ou em modo de entrada (INPUT em inglês).

Em algum local da memória do Arduino IDE essa função foi declarada por: void pinMode(uint8_t pin, uint8_t mode){ /*A várias instruções do pinMode. */}. Para detalhes, veja Libretexts, 2021.

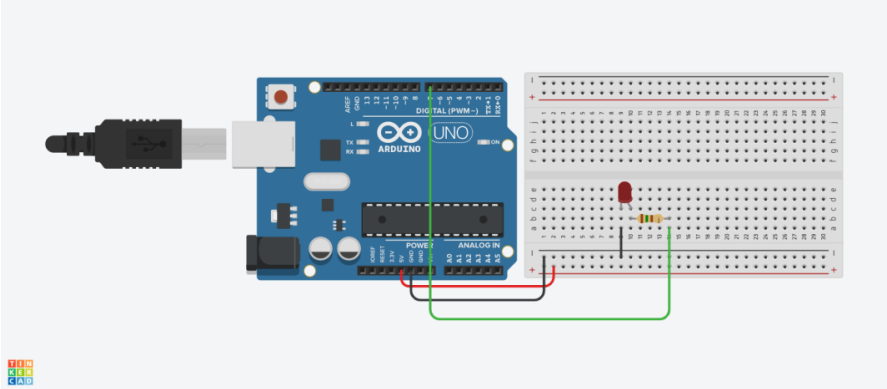
Como ela é void (tradução vazio), ela não tem retorno algum. Como já está declarada na memória, basta simplesmente usá-la na programação, sem precisar defini-la. Iremos inserir "pinMode(led,OUTPUT);" na função de inicialização setup para dizermos para o Arduino IDE que a porta led, que em nosso exemplo foi atribuído o valor de 7, é uma saída, ou seja, para essa porta enviar 0 volts ou 5 volts, conforme a programação. Outra função já declarada importante é a função digitalWrite de escrever na porta digital. Aplicada em (pino,valor), escolhemos o pino ou a variável que foi atribuída a um certo pino e o valor, ligado (HIGH, em inglês) ou desligado (LOW, em inglês). Exemplo, "digitalWrite(led,HIGH);" aqui estamos dizendo que é para escrever na porta digital led, que atribuímos 7 e informamos ser uma saída, o ligado. Novamente, não há a necessidade de declarar a função, simplesmente, utilizá-la. Para mais detalhes da definição dessa função, acesse Bauermeister, 2018.

Também temos a função já declarada delay, que permite gerar atrasos de milésimos de segundos. Exemplo, "delay(1000);" é para esperar 1000 milissegundos ou 1 segundos.

4.1 O CIRCUITO PISCAR O LED

Estamos prontos para propor o primeiro exemplo de circuito com o Arduino IDE. Caso não possua a placa Arduino, sugerimos utilizar o Tinkercad, como na figura 1. Observe que precisamos realizar a saída de 5 volts no pino 7 (pino que definimos como led e de saída), passar pelo resistor e LED e retornar para o GND, acendendo o LED conforme a programação do pino 7, além disso, como desejamos piscar o LED, precisamos alternar de 5 volts e 0 volts nesse pino. Veja o circuito na figura 4 e a programação na figura 5.

Figura 4 – Circuito Arduino, LED e resistor



Fonte: Autoria própria utilizando o software Tinkercad.

Figura 5 – Programação para piscar um LED

```
//Projeto 1 - Piscar um LED utilizando o Arduino
//Início da programação.

int led = 7; /*Aqui estamos introduzindo uma variável do tipo inteiro com o nome led
e atribuindo o valor 7.*/


void setup(){
  pinMode(led,OUTPUT); /*Estamos utilizando a função pinMode, para informar
  que o modo ou definição do pino led (ou 7) é OUTPUT (saída).*/
            }

void loop(){
    digitalWrite(led,HIGH); /*Estamos utilizando a função digitalWrite para escrever
    ligado no led (ou 7), ou seja, para enviar o valor alto 1 para a porta led.*/
    delay(1000); /*Espere por 1 segundo, ou seja, pausar a execução do
    código em 1000 ms.*/
    digitalWrite(led,LOW); /*É para escrever desligado no led (ou 7), ou seja,
    é para enviar o valor baixo 0 para a porta led.*/
    delay(1000); // Espere 1 segundo.
            } // Fim do loop e da programação.
```

Fonte: Autoria própria utilizando o software Tinkercad.

Para praticar a definição de função, sugerimos definir uma função que aplica em dois inteiros, sendo esses os dois delay's definidos na programação acima. Defina "void piscarled(int tempo1, int tempo2){ digitalWrite(led,HIGH); delay(tempo1); digitalWrite(led,LOW); delay(tempo2);}" e a utilize dentro da função loop, da seguinte maneira "void loop(){ piscarled(1000,500);}", gerando o piscar LED, ligando o por 1 segundo e desligando o por 0,5 segundos.

4.2 O CIRCUITO ACIONAR O LED UTILIZANDO UM PUSH BUTTON

Iremos agora adentrar com uma parte de funções que são de extrema importância para o C++, as chamadas estruturas de controle. Elas são blocos de instruções que alteram o fluxo de execução do código de um programa. Com elas é

possível realizar atividades como executar comandos diferentes conforme a condição ou repetir uma série de comandos várias vezes, por exemplo. Algumas dessas estruturas de controle são: if, if-else, while, for, entre outros. Diferente de uma função qualquer, esses caracteres if, if-else, while, for, entre outros, já estão declarados na memória dos compiladores C++.

O if (se em português) verifica uma condição, e apenas se ela for verdadeira, executa um conjunto de comandos adentrado na imagem da função entre as chaves { }. Caso a condição seja falsa, o programa prossegue sem executá-las. Exemplo:

if(condição) { … /*Aqui adentramos com o que se deve executar caso a condição seja verdadeira.*/ }

Antes de falarmos do if-else (se-senão em português), precisamos relembrar o que seja em Matemática uma função definida por partes. Por exemplo,

$$f(x) = \{x^2, se\ x > 0;\ x, caso\ contrário.$$

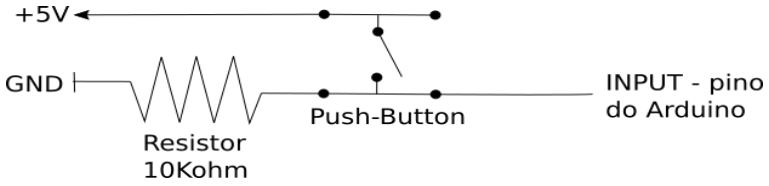
A função if-else desempenha na programação o papel de uma função definida por partes, ou seja, se a condição é verdadeira executa os comandos que estão entre o primeiro par de chaves {}, caso contrário, executa os que estão entre o segundo par de chaves {}. Definimos da seguinte maneira

if(condição) { … } else { … }

Finalmente, o while (enquanto em português) é uma estrutura que executa um conjunto de comandos repetidas vezes enquanto uma determinada condição for verdadeira. A sua definição será: while(condição) { … }.

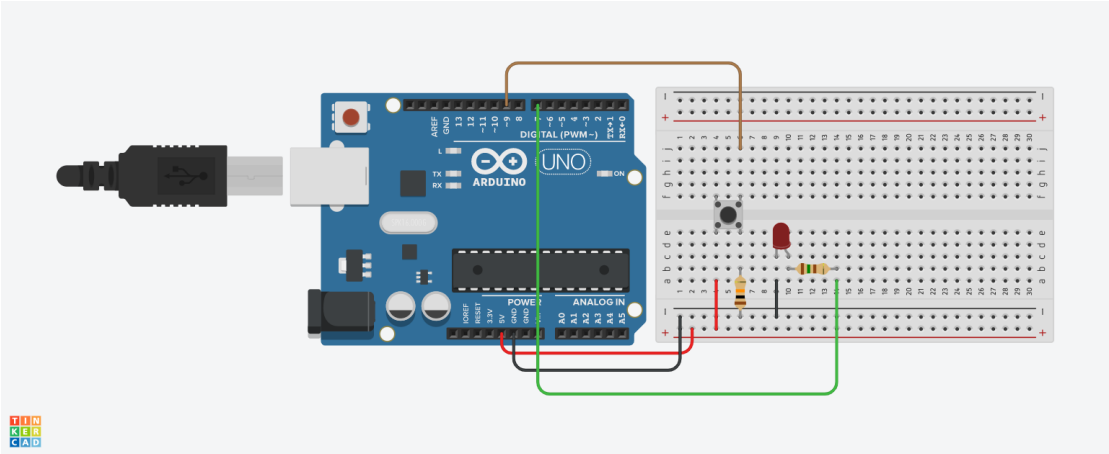
Estamos prontos para propor um segundo projeto, no qual acionaremos um LED a partir de um botão Push Button (botão de apertar em português). Esse botão, quando apertado, fecha uma chave, ligando o circuito que passa por ele. Como mostrado na figura abaixo, precisamos criar um circuito 5volts-Push Button-Resistor-GND. O resistor serve como pull down (puxar para baixo), ele reduz a corrente elétrica que passa pelo circuito, além de criar resistência para a corrente, forçando que flua pela entrada (INPUT, em inglês), que nesse caso estará ligado ao pino digital do Arduino.

Figura 6 – Circuito Push Button e Arduino



Fonte: Autoria própria.

Figura 7 – Circuito Arduino, Push Button, resistor 10 kohm, LED e resistor 150 ohm



Fonte: Autoria própria utilizando o software Tinkercad.

Figura 8 – Programação de acionar um LED com um botão

```
//Projeto 2 - Acionando um LED com um botão.
//Início da programação.

int led = 7;
int botao = 9; //Aqui estamos introduzindo uma variável do tipo inteiro com o nome botao e atribuindo o valor 9.
boolean botaoapertado = false; //A variável agora é um valor entre falso ou verdadeiro (0 ou 1).

void setup(){
  pinMode(led,OUTPUT);
  pinMode(botao,INPUT);/*Modo ou definição do pino botao (ou 9) é INPUT (entrada).
  O Arduino irá ler neste pino, corrente ou não corrente.*/
            }

void loop(){

  if(botaoapertado){ /*O if (se) funciona com o então, que está entre chaves. Nesse caso, somente se o
  botaoapertado for verdadeiro, irá executar o então. Na primeira leitura da programação, como
  botaoapertado é falso, o if não será executado.*/
  digitalWrite(led,HIGH); // É para escrever ligado no pino led.
                    }else{// Caso contrário a condição do if, executa o else.*/
                          digitalWrite(led,LOW); // É para escrever desligado no pino led.
                          }
  if(digitalRead(botao)){ // Aqui irá ler se o botão for pressionado. Se for, o if será executado.
    botaoapertado =! botaoapertado; // Se o botão for apertado, aqui irá inverter o estado do botaoapertado.
    while(digitalRead(botao)){} /*É para mudar o estado de botaoapertado somente uma vez enquanto o botão estiver
    pressionado.*/
                         } /*Ao retornar no início do loop, o botaoapertado está no estado diferente do anterior,
                         caso o botão tenha sido acionado.*/
           } //Fim do loop e da programação.
```

Fonte: Autoria própria utilizando o software Tinkercad.

4.3 O CIRCUITO AUTOMAÇÃO COM O SENSOR ULTRASSÔNICO E O BUZZER

Finalmente, iremos realizar uma programação de acender um LED (que pode ser substituído por um relé integrado a uma lâmpada) a partir de uma aproximação detectada pelo sensor de aproximação ultrassônico e assim criar uma automação.

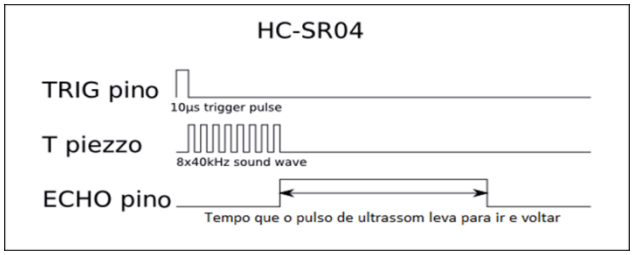
O sensor ultrassônico mede a distância através da medição do tempo que uma onda leva para sair do emissor, colidir com um obstáculo e ser refletido, para, finalmente, ser detectado pelo receptor. Desta forma, podemos notar que nosso sensor ultrassônico possui dois cilindros metálicos que se assemelham a olhos em sua placa. São, na realidade, dois alto falantes: um trabalha como o emissor, o T piezzo, do sinal ultrassônico e o outro como receptor, o Echo. Veja a figura abaixo

Figura 9 – Sensor ultrassônico

Fonte: Autoria própria utilizando o software Tinkercad.

Para iniciarmos uma medição, o pino Trig, que funciona como gatilho do nosso sensor ultrassônico, deve receber um pulso de 5V por pelo menos 10 microssegundos. Isso fará com que o sensor HC-SR04 emita 8 pulsos ultrassônicos em 40kHz pelo T piezzo e o pino Echo, que funcionará como nosso cronômetro, vai para 5V, iniciando assim a espera pelas ondas refletidas. Assim que uma onda refletida for detectada, o pino Echo, que estava em 5V, será alterado para 0V. Desta forma, o período que o pino Echo fica em 5V é igual ao tempo que a onda emitida leva para ir até o obstáculo e voltar. Para mais detalhes veja Silva e Barbosa, 2021.

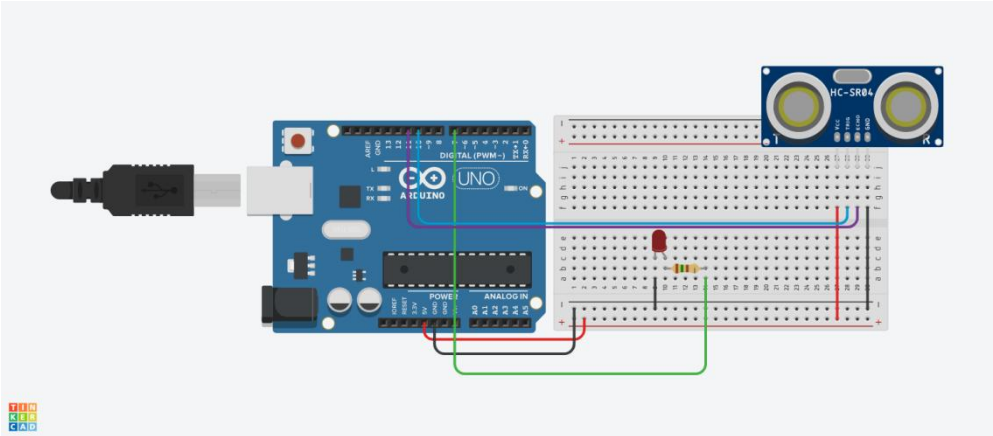
Figura 10 – Funcionamento do sensor ultrassônico



Fonte: Mota, 2017.

Em posse do tempo que o pino Echo ficou em 5 volts, sabendo que ele é o tempo de ida e volta da onda do sensor ultrassônico até o obstáculo, e, considerando a velocidade do som igual a 0.034029 centímetros por microssegundo, temos $\Delta s/\Delta t = 0.034029$ cm/µs. Porém, como $\Delta t = t/2$, temos $\Delta s/(t/2) = 0.034029 \Rightarrow \Delta s = 0.034029\, t/2$ cm.

Figura 11 – Circuito LED, resistor e sensor ultrassônico



Fonte: Autoria própria utilizando o software Tinkercad.

Figura 12 – Programação para acionar o LED com a aproximação

```
//Projeto 3 - Ligar um LED com a aproximação de objeto utilizando o sensor ultrassônico.
// Início da programação.

int led = 7;
int trigPin = 10;
int echoPin = 11;

void setup() {
  pinMode(led, OUTPUT);
  pinMode(trigPin, OUTPUT); // Configuração do pino Trig como saída.
  pinMode(echoPin, INPUT); //Configuração do pino Echo como entrada.
  digitalWrite(trigPin, LOW); //Desligar o pino Trig.
              }
void loop() {
  digitalWrite(trigPin, HIGH); /*Iniciar o envio das ondas para a leitura do Echo.
  Nesse momento o pino Echo fica em 5 volts*/
  delayMicroseconds(10);  /* Utilizar a função delayMicroseconds para esperar 10 microssegundos.*/
  digitalWrite(trigPin, LOW);  // Desligar o pino Trig para a leitura das ondas.
  double pulsetime = pulseIn(echoPin, HIGH); /*Mede o tempo que o pino de Echo ficou no estado
  ligado ou alto, ou seja, o tempo de propagação da onda. Lembrando que ele é desligado quando
  recebe o retorno da onda enviada pelo T piezzo.*/
  double distance = 0.034029 * pulsetime/2; /*A distância entre o sensor ultrassônico e o objeto será a
  velocidade do som = 0.034029 cm/µs em centímetros por microssegundo vezes a metade do tempo de
  propagação, pois a onda vai e volta.*/

  if(distance < 50){
    digitalWrite(led, HIGH); //Se a distância é menor do que 50 cm, então o LED é ligado.
                    }else{
    digitalWrite(led, LOW); /*Caso contrário, ou seja, a distância é maior do que 50 cm, então
    o LED é desligado.*/
                    }
    delay(100);
              } // Fim do loop e da programação.

```

Fonte: Autoria própria utilizando o software Tinkercad.

Para realizar a inserção do buzzer, primeiro crie o circuito GND-buzzer-resistor de 220ohm-pino4, e na programação, basta antes do void setup incluir "int buzzer = 4;", e no void setup informar "pinMode(buzzer,OUTPUT);". Finalmente, incluir as funções já declaradas "tone(buzzer,1500);" para acionar o buzzer e "noTone(buzzer);" para não acionar.

As propostas aqui apresentadas são para os iniciantes em Arduino que desejam explorar um pouco mais da Matemática nesta tecnologia. Os microcontroladores estão cada vez mais presentes na vida dos estudantes, por isso é essencial explorá-los como forma de ensino e aprendizagem de Matemática, possibilitando aos estudantes analisar e perceber o quanto a Matemática se faz necessária nas tecnologias.

REFERÊNCIAS

GUIMARÃES, D. da S.; SILVA, Élida A. da; BARBOSA, F. da C. Explorando a matemática e a física com o robô seguidor de linha na perspectiva da robótica livre. **Texto Livre**, Belo Horizonte-MG, v. 14, n. 1, 2020.

LIBRETEXTS. **PinMode().** Disponível em: <https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electronics/Embedded_Controllers_Using_C_and_Arduino_(Fiore)/21%3A_Bits_and_Pieces__pinMode()/21.1%3A_pinMode()>. Publicada em 7 set. 2021. Acesso em: 18 dez. 2022.

BAUERMEISTER, G. **Biblioteca Arduino; aprenda como criar a sua - Parte 1.** Disponível em: <https://www.filipeflop.com/blog/desenvolvendo-uma-biblioteca-arduino/>. Publicada em 10 abr. de 2018. Acesso em: 18 dez. 2022.

MOTA, A. **HC-SR04 - Sensor Ultrassônico de distância com Arduino.** Disponível em: <https://portal.vidadesilicio.com.br/hc-sr04-sensor-ultrassonico/>. Publicada em 17 maio de 2017. Acesso em: 18 dez. 2022.

SILVA, M. P. da; BARBOSA, F. da C. **Matemática e Física em experiências de Robótica Livre: explorando o sensor ultrassônico.** Texto Livre, Belo Horizonte-MG, v. 14, n. 3, p. e29629, 2021. DOI: 10.35699/1983-3652.2021.29629. Disponível em: https://periodicos.ufmg.br/index.php/textolivre/article/view/29629. Acesso em: 7 dez. 2023.