



# **Sensor de ecolocalização rotacional: Uma experiência com Arduino**

**Produto Educacional**

**Jonathan Sardo**





# Conteúdo

<b>I</b>	<b>Introdução</b>	
<b>1</b>	<b>Apresentação</b>	<b>7</b>
<b>2</b>	<b>Conceitos prévios</b>	<b>9</b>
<b>2.1</b>	<b>O que é Arduino?</b>	<b>9</b>
<b>2.2</b>	<b>O que eu posso conectar no Arduino?</b>	<b>11</b>
2.2.1	Sensor ultrassônico	12
2.2.2	Servo motor	13
2.2.3	Circuito elétrico	15
<b>2.3</b>	<b>E como é feita a comunicação?</b>	<b>16</b>
<b>2.4</b>	<b>TinkerCAD</b>	<b>18</b>
<b>II</b>	<b>O dispositivo</b>	
<b>3</b>	<b>Hardware</b>	<b>23</b>
<b>3.1</b>	<b>Materiais</b>	<b>23</b>
<b>3.2</b>	<b>Montagem</b>	<b>23</b>
<b>3.3</b>	<b>Armazenamento</b>	<b>24</b>
<b>4</b>	<b>Software</b>	<b>27</b>
<b>4.1</b>	<b>Arduino IDE</b>	<b>27</b>
<b>4.2</b>	<b>Código do dispositivo</b>	<b>28</b>
<b>4.3</b>	<b>Definições e pacotes</b>	<b>29</b>
<b>4.4</b>	<b>Função <code>setup()</code></b>	<b>31</b>
<b>4.5</b>	<b>Função <code>loop()</code></b>	<b>31</b>

4.6	Código completo	33
5	Teste .....	35

### III

## Atividades

6	Sugestão de atividades .....	39
6.1	Programação e prototipagem	39
6.2	Geometria analítica e trigonometria	39

### IV

## Considerações finais


7	Conclusão .....	43
	Bibliografia .....	45



# Introdução

<b>1</b>	<b>Apresentação .....</b>	<b>7</b>
<b>2</b>	<b>Conceitos prévios .....</b>	<b>9</b>
2.1	O que é Arduino?	
2.2	O que eu posso conectar no Arduino?	
2.3	E como é feita a comunicação?	
2.4	TinkerCAD	





# 1. Apresentação

Caro colega professor(a),

Esse produto educacional intitulado "Sensor de ecolocalização rotacional: Uma experiência com Arduino", é resultado de uma dissertação desenvolvida junto ao Programa de Pós-Graduação Mestrado Profissional em Matemática em Rede Nacional – PROFMAT, no Centro de Ciências Tecnológicas da Universidade Estadual de Santa Catarina, sob orientação do professor Dr. Fernando Deeke Sasse.

Durante meus experimentos com a prototipagem rápida, especificamente Arduino, e seus sensores, identifiquei ao longo dos testes a possibilidade de integração da prototipagem ao ensino de matemática, uma vez que não só a comunicação do microcontrolador para com sensores e acessórios são realizadas através da lógica de programação, quanto a interpretação de leituras obtidas no microcontrolador permite aplicar e as vezes ressignificar conteúdos matemáticos conectando estes a elementos reais.

O objetivo deste produto educacional é proporcionar a você professor(a) a elaboração de um protótipo. Uma experiência que permita contextualizar o ensino de matemática através da construção de um dispositivo e da manipulação de componentes eletrônicos, de modo a propiciar ao estudante durante as aulas regulares a verificação prática dos conceitos matemáticos na interpretação de sensores e na operação componentes.

O protótipo, baseado no trabalho de Criollo-Sánchez et al. (2018), consiste em posicionar um sensor ultrassônico para ecolocalização sobre um servomotor, de forma tal que ao girar o motor, com auxílio do Arduino, seja possível identificar a distância de elementos captados pelo sensor e o do ângulo estipulado pelo passo do servomotor. A proposta então é construir um equipamento que se comporte de forma similar ao que ocorre no georreferenciamento obtido em radares de navios, submarinos etc.

Dessa forma, esse guia de experimento seguirá em três partes:

- Primeira parte: Conceitos básicos; aqui falaremos brevemente sobre o Arduino como plataforma de prototipagem rápida, seu funcionamento e as componentes utilizadas nesse trabalho.
- Segunda parte: Hardware e software; mostraremos nesse capítulo os materiais necessários para a construção física do protótipo, como ele são realizadas as conexões necessárias para

seu funcionamento e verificaremos como é realizada a programação dos componentes através da interface em um computador.

- Terceira parte: Sugestão de atividades; aqui exploraremos algumas possibilidades de atividades que podem ser realizadas com auxílio do protótipo desenvolvido.

Espera-se então, que esse produto possa contribuir com a atividade docente e que desperte em você, professor(a), as potencialidades que a prototipagem rápida pode proporcionar no ensino de matemática.

Atenciosamente,  
Professor Jonathan Sardo.



## 2. Conceitos prévios

### 2.1 O que é Arduino?

O Arduino é uma plataforma de prototipagem rápida de baixo custo e de código aberto, que permite através de programação de seu microcontrolador, a leitura e ações envolvendo sensores, botões, LEDs, motores e outros componentes diversos. Criado na Itália pela Ivrea Interaction Design Institute, a principal proposta do Arduino é levar às pessoas uma ferramenta fácil de manipular e que permita desenvolver um protótipo de um produto para pessoas que não possuam conhecimento avançado em eletrônica e programação (ARDUINO, 2021).

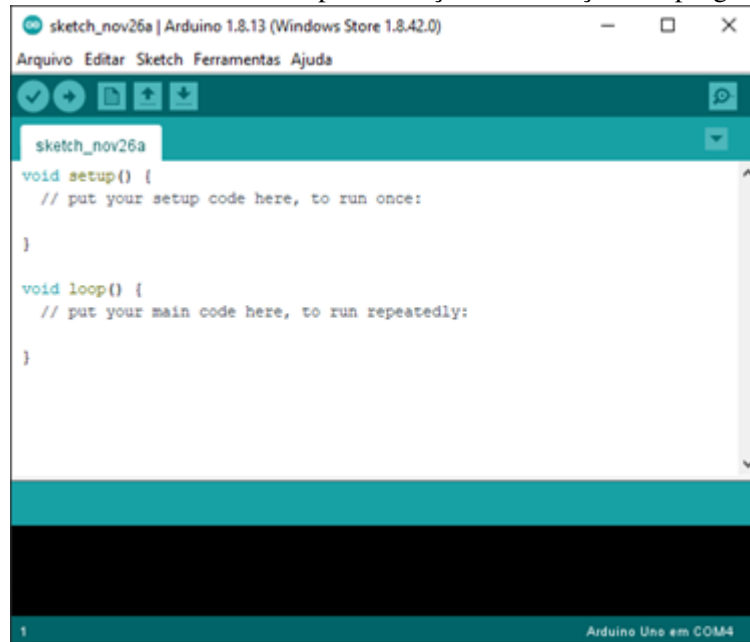
Existem diversos modelos de placas Arduino disponíveis no site oficial da plataforma adaptados à realidade de cada projeto no entanto, o modelo mais utilizado e comumente disponibilizado junto aos kits introdutórios de prototipagem é o modelo Arduino UNO da Figura 2.1. Através de software de desenvolvimento integrado à plataforma - IDE, ver Figura 2.2, o usuário pode instruir o microcontrolador da placa Arduino uma série de comandos a fim de executar uma determinada atividade através coleta, processamento e envio de informações.

Figura 2.1: Placa Arduino Uno.



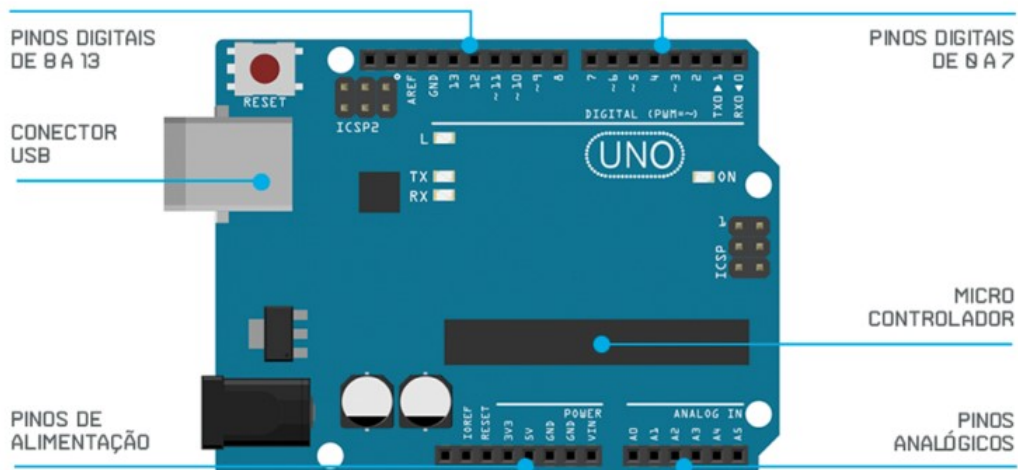
As conexões presentes na placa Arduino UNO podem ser verificadas na Figura 2.3. A comunicação entre a placa Arduino e o computador para registro do código no microcontrolador é

Figura 2.2: Interface de usuário para inserção de instruções de programação.



realizada através de alimentação USB, por onde também é possível alimentar a placa. Há também outra entrada de alimentação externa para que a placa mantenha o funcionamento sem que esteja necessariamente conectada a um computador, isto é, uma vez que o código seja registrado no microprocessador o computador não se faz necessário para o funcionamento do dispositivo. A codificação do microcontrolador se dá via código de programação na linguagem C/C++.

Figura 2.3: Placa Arduino UNO e seus principais elementos.



O Arduino possui uma série de pinos onde são conectados, jumpers, sensores e componentes diversos. Cada pino pode exercer o papel de entrada ou saída, denotado por I/O - de IN/OUT, e o papel que determinado pino realizará na programação do microcontrolador deverá ser categorizado para cada pino, isto é, se a componente que irá ser conectada no pino recebe - IN - ou envia informação - OUT. Além disso, conforme verificado na Figura 2.3, os pinos podem trabalhar com

informações digitais, digitais PWM<sup>1</sup> ou analógicas.

Para o bom funcionamento tanto da placa Arduino, como dos componentes que nela serão conectados, é necessário conhecimento básico em funcionamento de uma malha elétrica - relação entre tensão, corrente e resistência - para elaboração do protótipo como um todo, logo, alguns aspectos das especificações técnicas do Arduino disponibilizados na tabela 2.1 são necessários e relevantes.

Tabela 2.1: Especificações técnicas da placa Arduino UNO.

Arduino UNO	
Microcontrolador	ATmega328P
Tensão de operação	5V
Tensão de alimentação recomendada	7-12V
Pinos digitais I/O	14 (das quais 6 são saídas PWM)
Pinos analógicas	6
Corrente Contínua por pino I/O	20 mA
Corrente Contínua por pino de 3.3V	50 mA

**Observação!** O Arduino é uma plataforma de prototipagem de **código aberto** e assim, tanto a sua estrutura de hardware quanto de software podem ser moldadas a atividade que se propõe realizar.

*Código aberto? E o que isso significa?*

Significa que você pode comprar os componentes que compõem uma placa Arduino separados e montar a sua própria placa Arduino. Portanto, se você procurar por Arduino e encontrar vários modelos como Arduino Nano, Arduino Mega, Arduino Leonardo, e outros; saiba que ambos funcionam da mesma maneira, mas foram criados com dimensões específicas para atividades diferentes.

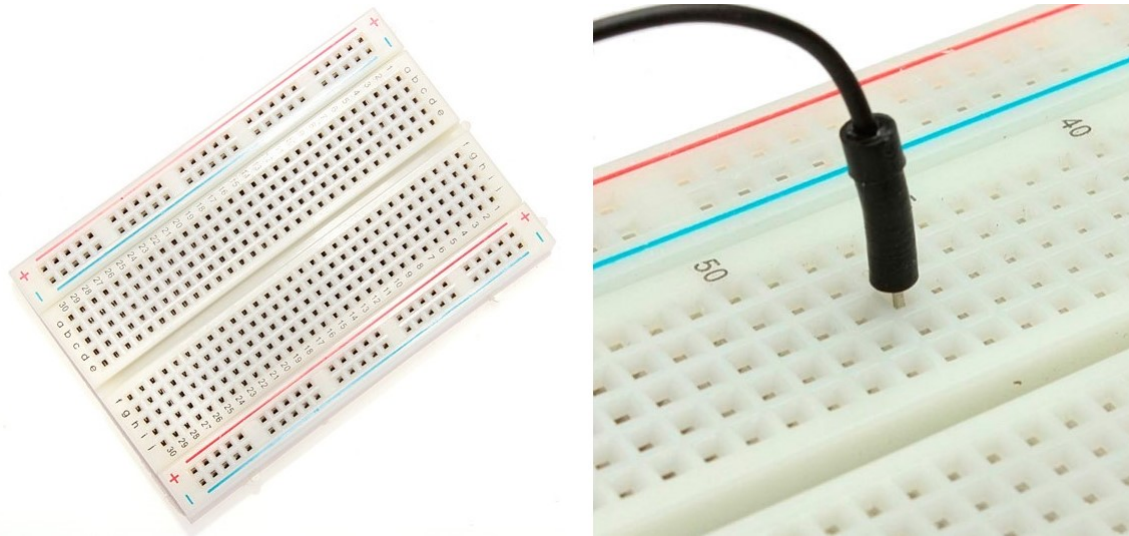
## 2.2 O que eu posso conectar no Arduino?

Existem diversos componentes, sensores e dispositivos que podem ser conectados ao Arduino e programados conforme a nossa vontade. Porém, inicialmente é necessário falar componentes básicos que para manipulação de um protótipo como os *jumpers* e a *proto-board*.

A conectividade entre componentes ao Arduino durante a fase protótipo é realizada através de *jumpers*, que são cabos conectores de fácil manipulação, ver Figura 2.5. Além desses é comumente utilizado durante a elaboração dos estudos placas de ensaio com furos interconectados, chamadas de *proto-boards*, que permitem que mais *jumpers* compartilhem a mesma conexão, ver Figura 2.4.

Particularmente no dispositivo em que esse trabalho foi proposto, dois componentes são essenciais para o funcionamento desse.

<sup>1</sup>PWM - Pulse Width Modulation é uma técnica para controle de tensão através da modulação pela largura de pulso digital em um circuito. Ao controlar o circuito através da abertura e fechamento de uma chave em determinada frequência, é possível dispor nesse circuito um percentual da tensão original, simulando o que ocorreria em um controle analógico

Figura 2.4: Placa de ensaio com furos interconectados - *protoboard*.Figura 2.5: Cabos conectores (*jumpers*) para conexão na *protoboard*.

### 2.2.1 Sensor ultrassônico

O sensor ultrassônico utilizado durante o trabalho foi o sensor HC-SR04, Figura 2.6 e especificação na tabela 2.2, que é um módulo com emissor e receptor acoplado que permite medir distâncias de 2cm a 4m.

Seus 4 pinos referem-se a: alimentação (VCC) e saída (GND), e seus pinos do meio são um ao responsável por enviar um sinal sonoro (Trigger) com frequência 40kHz, isto é inaudível ao ouvido humano, e o outro (ECHO) para receber o tempo que demorou para o retorno do sinal.

Para isso, é necessário que se mantenha o pino ECHO sempre ativo (HIGH) e o pino Trigger esteja ativo (HIGH) disparando o sinal por pelo menos 10 microssegundos.

Da física, temos que a velocidade é a razão da distância pelo tempo percorrido, isto é,

$$v = \frac{\Delta s}{\Delta t} \Rightarrow \Delta s = \Delta t \cdot v$$

Logo, é possível determinar a distância percorrida pela onda sonora, estabelecendo  $v$  como a velocidade do som  $340,29m/s$  e  $\Delta t$  o tempo, em microssegundos, capturado pelo sensor para que a onda fizesse o trajeto de ir até o objeto e retornar ao sensor. Desta forma, o valor definido por  $\delta s$  corresponderia ao dobro da distância, a qual denominaremos  $d$ , do objeto ao sensor é definido da

forma

$$d = \frac{\Delta s \cdot v}{2} \rightarrow d = \frac{t_{\text{ECHO}} \cdot V_{\text{som}}}{2}.$$

Estabelecendo centímetro para a distância coletada pelo sensor, e sabendo que o sensor HC-SR04 realiza suas operações em microssegundos, é necessário efetuar as devidas conversões para o funcionamento da equação. Isto é, adotando a velocidade do som de  $340,29\text{m/s}$  para  $0,03429\text{cm/s}$  temos que a distância entre o objeto e o sensor é

$$d = t_{\text{ECHO}} \cdot 0.0170145.$$

**Observação!** Durante a elaboração e aplicação de um protótipo em aula utilizando o Arduino como ferramenta educacional, é natural que diversas áreas de conhecimento se entrelacem e se comuniquem na busca pela solução de um problema. Logo, explorar o conceito físico da onda sonora, bem como seu comportamento periódico podem potencializar a prática como um todo.

Figura 2.6: Sensor Ultrassônico HC-SR04

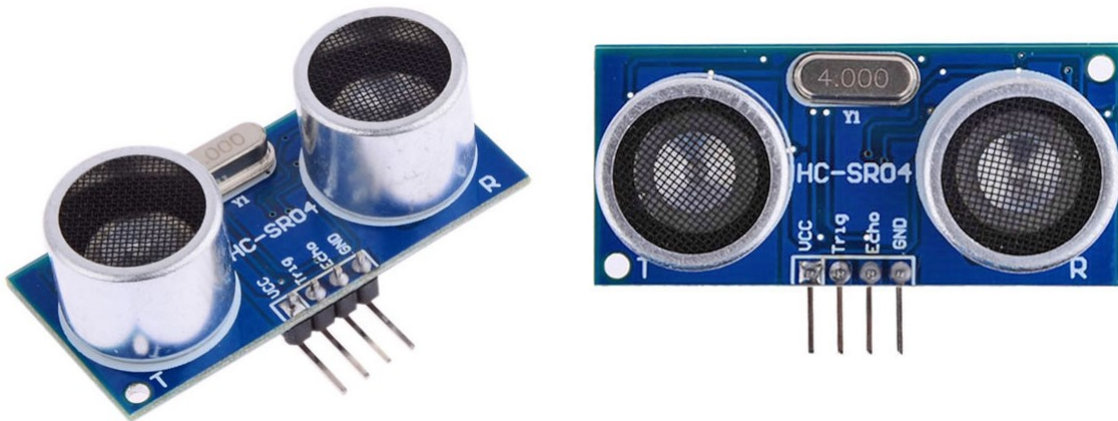


Tabela 2.2: Especificações técnicas do sensor ultrassônico HCSR04

HC-SR04	
Alimentação	5V DC
Corrente de Operação	2mA
Ângulo de efeito	15°
Alcance	2cm 4m
Precisão	3mm

### 2.2.2 Servo motor

O servo motor é um motor elétrico que possui internamente um encoder e um controlador. Esse encoder, responsável pelo controle de velocidade do motor, possui um sensor interno o qual fornece



a função de controle da velocidade e posicionamento do motor. Logo, através desse componente, é possível estabelecer a velocidade de giro do motor e posição angular que ele se encontra.

Para giro do sensor sônico, posicionaremos este sobre o Micro Servo 9G SG90, Figura 2.7 o qual possui como principais características para este trabalho, a amplitude de giro de 180 graus.

Figura 2.7: Micro Servo 9G SG90



Elaborada pelo autor (2021).

Tabela 2.3: Especificações técnicas do Micro Servo 9G SG90

Micro Servo 9G SG90	
Tensão de Operação	3,0 – 7,2V
Ângulo de rotação	180 graus
Velocidade	0,12 seg/60Graus (4,8V) sem carga
Torque	1,2 kg.cm (4,8V) e 1,6 kg.cm (6,0V)

Elaborada pelo autor (2021).

**Observação!** Os servos motores de forma geral são componentes amplamente utilizados em sistemas robóticos, de automação, aeromodelismo e diversas outras aplicações.

De forma geral, existem diversos componentes que podem ser utilizados na elaboração de um protótipo como: LEDs, visores LCD, módulos de conectividade com internet e outros dispositivos, e diversos outros sensores. Tal liberdade fica a critério do propósito ao qual o

protótipo foi elaborado. Portanto, demais acessórios não serão detalhados aqui, embora possam acrescentar funcionalidades ao dispositivo proposto neste trabalho.

### 2.2.3 Circuito elétrico

Para efetuar as conexões dos componentes é necessário estabelecer o comportamento físico que ocorre dentro da malha elétrica, suas principais unidades de medida e como se relacionam. Para isso, seguindo Johnson Hilburn (1994) e Gaspar (2013), define-se que circuito elétrico é uma coleção de elementos elétricos onde a proposição básica é mover ou transferir cargas através de um percurso especificado.

A movimentação das cargas dentro do circuito, gera o que chamamos de corrente elétrica. Com unidade de medida *Ampère* (A), a corrente elétrica ( $i$ ) em um circuito é a quantidade de carga ( $Q$ ) que trafega em um determinado período de tempo ( $t$ ). Isto é

$$i = \frac{dQ}{dt}.$$

Ao se estabelecer a quantidade de cargas que passam em um determinado tempo, naturalmente define-se um sentido para a corrente elétrica. Imaginou-se então que tal sentido para a corrente iria do polo positivo para o polo negativo. No entanto, isso ocorreu antes de se saber que são elétrons - cargas negativas, que se movimentam ao se desprender das órbitas dos átomos dos metais condutores. Dessa forma, contrário ao movimento das cargas, a corrente elétrica possui sentido partindo do polo positivo para o negativo.

Existem diferentes maneiras com que a corrente é trabalhada em um circuito, no entanto focaremos aqui a corrente contínua, que é utilizada em plataformas digitais como Arduino.

A movimentação dos elétrons de forma orientada, se dá devido a diferença dos polos que conectam o início ao fim do circuito elétrico. Como estudado em cursos de física de ensino médio, as cargas com sinais iguais se repelem e com sinais diferentes se atraem. Logo, proporcionar ao circuito essa diferença de polos é o que permite a movimentação ordenada dos elétrons. Define-se então tensão ou diferença de potencial, de unidade de medida *volt* (V), o trabalho realizado pela carga elétrica.

Diferentes componentes possuem características próprias para seu funcionamento, conforme verificado nas especificações das componentes das Tabelas 2.1, 2.2 e 2.3. Dessa forma, para que o circuito elétrico funcione corretamente, é necessário atender as especificidades dos componentes que estão presentes no circuito. Um LED por exemplo, possui especificidades conforme a Tabela 2.4, e não é possível conectá-lo diretamente ao Arduino que proporciona ao circuito em seus pinos digitais uma corrente de 40mA.

Tabela 2.4: Especificações técnicas de um LED

LED	
Diâmetro	5mm
Tensão de operação	1,9V 2,1V
Corrente de operação	20mA
Luminosidade	300 MCD

Elaborada pelo autor (2021).

Dessa forma, existem técnicas para controle de tensão e corrente em circuito elétrico que vão desde a forma em que são associados os componentes, série e paralelo. Algumas regras fundamentais e leis que regem a conectividade em circuitos elétricos temos:

- Lei de Ohm: Aborda a resistência elétrica de condutores ( $R$ ), determinando uma relação de proporcionalidade entre corrente elétrica ( $i$ ) e tensão ( $V$ ) ao qual um circuito foi submetido. Tal relação pode ser verificada na equação

$$V = R \cdot i.$$

- Lei de Kierchkoff para corrente: A soma algébrica da intensidade das correntes elétricas em um **nó** é nula.
- Lei de Kierchkoff para tensão: A soma algébrica das variações de potencial elétrico em uma **malha** é nula.

O aprofundamento das leis e utilização destas em circuito podem ser verificados em materiais de cursos de Física e eletrônica como Johnson Hilburn (1994) e Gaspar (2013) ou em apostilados de curso de Arduino, como Vidal (2018).

### 2.3 E como é feita a comunicação?

Uma vez realizada a conexão dos componentes, a programação do microcontrolador presente no Arduino é realizada através da interface IDE da Figura 2.2. Isto é, através da conexão USB do computador, conecta-se a placa Arduino ao computador e submete ao microcontrolador a programação desejada.

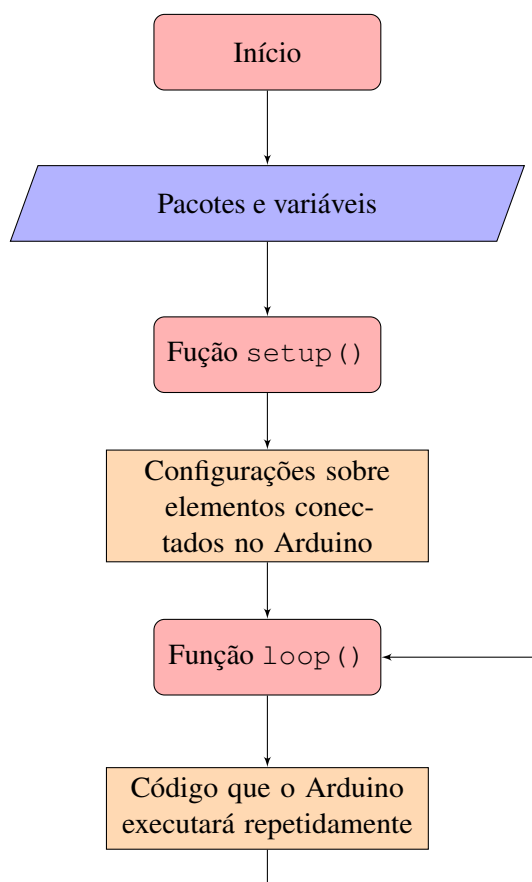
A estrutura de código no microcontrolador é dividido em dois grupos conforme verifica-se no diagrama da Figura 2.8. No primeiro bloco é realizada a configuração inicial do Arduino, definida pela função `setup()`, que é chamada assim que o dispositivo é ligado.

A função `setup()` é responsável por identificar quais são os componentes que estão conectados no dispositivo, em quais pinos e qual a configuração se faz nesses necessária nesses pinos - entrada (INPUT) ou saída (OUTPUT).

O segundo bloco é uma repetição sem limitação definida pela função `loop()` a qual executará o código nela restrito intermitentemente. Por exemplo, se configurarmos para acender um LED por 1 segundo e depois mantê-lo apagado por 1 segundo, o resultado que teríamos é um LED que pisca a cada 2 segundos. O código deste exemplo, pode ser verificado na Figura 2.9.



Figura 2.8: Processo de execução do microcontrolador.



Elaborada pelo autor (2021).

Figura 2.9: Exemplo de código para o Arduino - Acender e apagar um LED a cada segundo.

```
1 void setup()
2 {
3   pinMode(11, OUTPUT); // Definir o pino 11 como saída de
   informação.
4 }
5
6 void loop()
7 {
8   digitalWrite(11, HIGH); // Enviar ao pino 11, sinal digital
   HIGH, para que acenda o LED
9   delay(1000); // Aguardar 1000 milissegundos
10  digitalWrite(11, LOW); // Enviar ao pino 11, sinal digital LOW,
   para que desligue o LED
11  delay(1000); // Aguardar 1000 milissegundos
12 }
```

Elaborada pelo autor (2021).

**Observação!** Algumas considerações referentes aos códigos que apresentaremos ao longo do trabalho e especificamente no código da Figura 2.9:

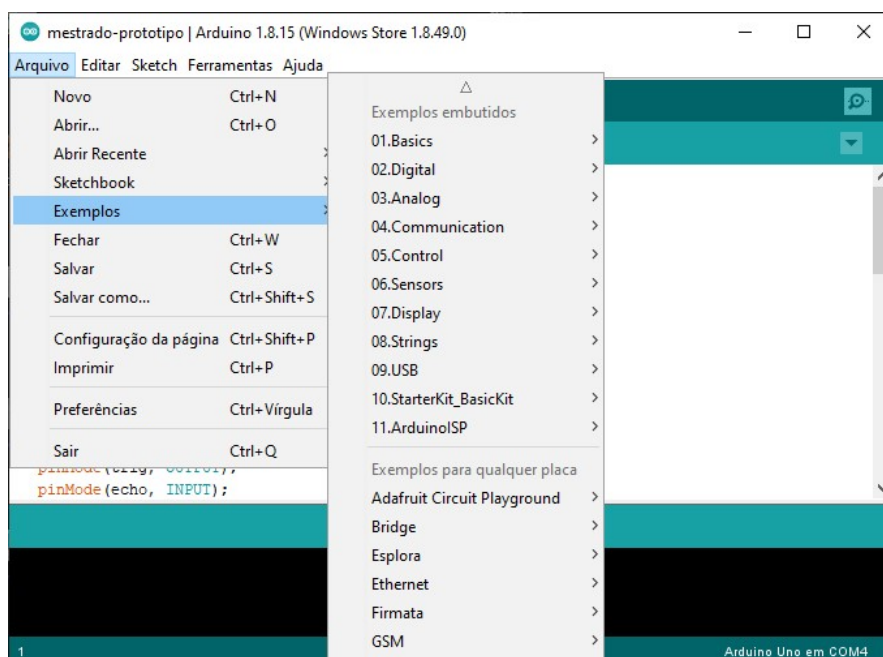
- Os comandos // presentes no código, simbolizam que o que estiver escrito após é um comentário, isto é, é executado pelo Arduino. Dessa forma, é comum a comunidade de desenvolvedores utilizarem tal prática para descrever o que determinada linha de comando irá realizar.
- O Arduino realiza seu código em estrutura de tempo de milissegundos, portanto quando se faz necessário aguardar determinado tempo em uma linha de comando, utiliza-se o comando `delay()` com o tempo já convertido.

Como a proposta de prototipagem rápida, consiste em elaborar produtos de forma que não seja necessário aprofundamento nos conhecimentos de eletrônica e programação, a própria interface IDE do Arduino traz uma série de exemplos de como manipular diversos acessórios e componentes, conforme é possível verificar na Figura 2.10.

## 2.4 TinkerCAD

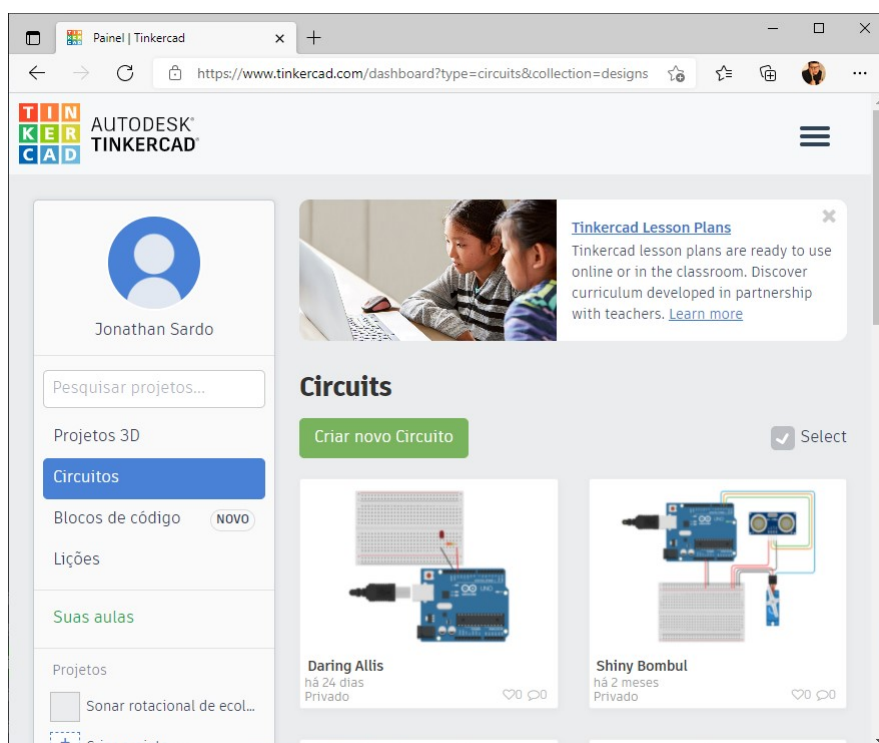
Visando praticidade na elaboração de protótipos, existem ferramentas que permitem que qualquer pessoa possa simular anteriormente os projetos em prototipagem além de compartilhar com a comunidade e entusiastas de prototipagem rápida. É o caso da Autodesk TinkerCAD, que segundo o próprio site, ver Figura 2.11, é um aplicativo gratuito e fácil de usar para projetos 3D, componentes eletrônicos e codificação. É usado por professores, crianças, amadores e projetistas para imaginar, projetar e fabricar qualquer coisa!

Figura 2.10: Exemplos no Arduino IDE



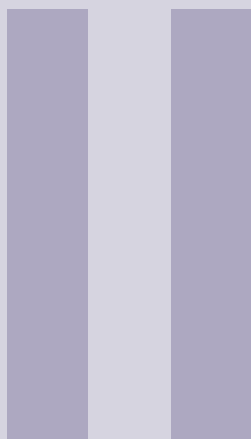
Elaborada pelo autor (2021).

Figura 2.11: Interface TinkerCAD



Elaborada pelo autor (2021).





# O dispositivo

<b>3</b>	<b>Hardware</b> .....	<b>23</b>
3.1	Materiais	
3.2	Montagem	
3.3	Armazenamento	
<b>4</b>	<b>Software</b> .....	<b>27</b>
4.1	Arduino IDE	
4.2	Código do dispositivo	
4.3	Definições e pacotes	
4.4	Função <code>setup()</code>	
4.5	Função <code>loop()</code>	
4.6	Código completo	
<b>5</b>	<b>Teste</b> .....	<b>35</b>





## 3. Hardware

### 3.1 Materiais

Para elaboração do dispositivo, são necessários os seguintes componentes:

- 1 placa Arduino UNO - Pode ser outro modelo, desde que tenha conectores o suficiente;
- 1 *Protoboard*;
- 1 Sensor de ecolocalização HC-SR04;
- 1 Microservo motor SG90G;
- *Jumpers* conectores;
- Materiais para construção física do dispositivo: papelão, palitos, sucatas, cola quente, parafusos etc.

### 3.2 Montagem

Primeiramente seguimos com a conexões dos componentes ao Arduino, seguindo o protótipo desenvolvido no TinkerCAD, conforme a Figura 3.1.

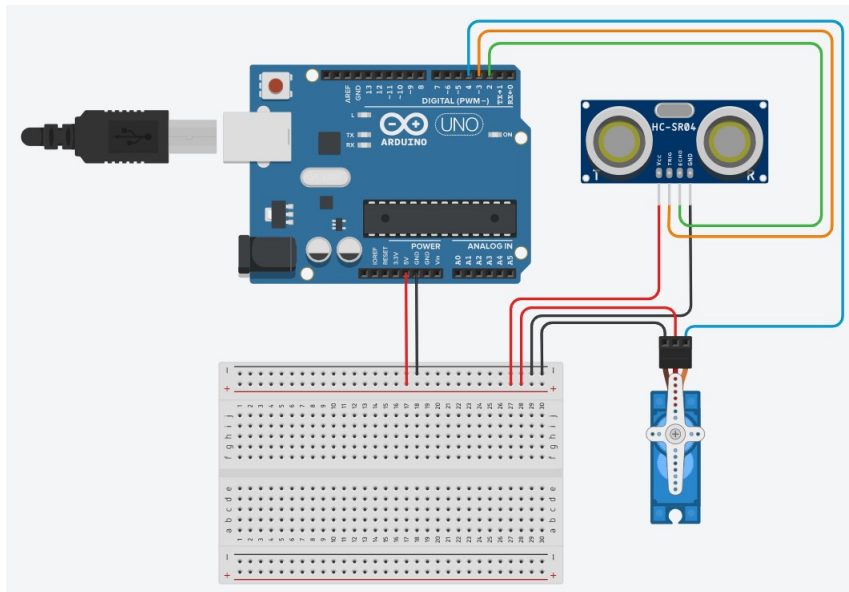
1. Energizamos a *protoboard*, conectando um *jumper* na saída 5V e na trilha positiva da *protoboard* e outro *jumper* na trilha negativa da *protoboard*.
2. A partir da *protoboard* energizada, alimentamos o servo motor e o sensor sonar ultrassônico.
3. Nos pinos digitais, conectamos ao pino 2 e 3 do Arduino as conexões ECHO e TRIG do sensor de sonar ultrassônico.
4. No pino digital 4, conectamos o servo motor.

Atente sempre aos pinos em que os dispositivos foram conectados, pois durante a configuração do microcontrolador do Arduino, os componentes serão referenciados pelas posições em que foram conectados.

Uma vez realizada as conexões, é preciso posicionar o sensor ultrassônico junto ao servo motor, o que pode ser feito de maneira artesanal através de papelão ou cartolina com auxílio de cola quente. Para um melhor acabamento, existem suportes elaborados especificamente para o posicionamento do sensor ultrassônico, que já possui os furos de modo a fixação sobre o eixo do servo motor. Esses suportes podem ser adquiridos em lojas de equipamentos eletrônicos e robótica.

**Observação!** A coloração dos *jumpers* na conexão dos elementos é uma boa prática para evitar possíveis conexões erradas. Note que na Figura 3.1 foi estabelecido *jumpers* vermelhos para o polo positivo dos conectores e *jumpers* pretos para o polo negativo, representado na placa Arduino pela sigla GND.

Figura 3.1: Conexão dos componentes elaborado previamente via TinkerCAD



Elaborada pelo autor (2021).

### 3.3 Armazenamento

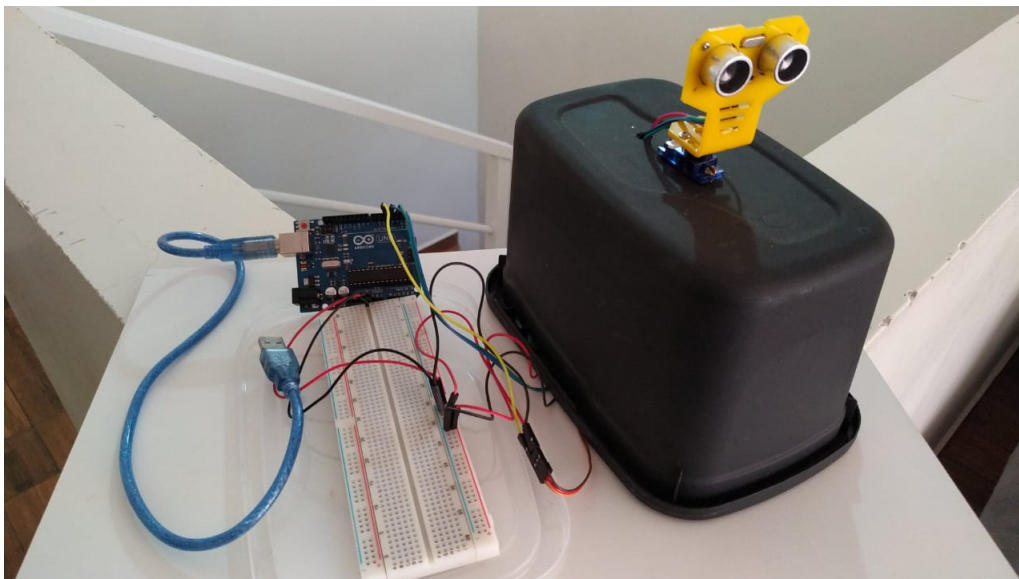
Após ter efetuado a construção física dos componentes no Arduino, tanto para manipulação do equipamento quanto para evitar possíveis acidentes, é interessante armazenar os acessórios de forma que seja possível transportá-los e guardá-los.

Para isso, uma saída de fácil operação é a alocação dos equipamentos dentro de um pote plástico, obtido de sucatas. Recomenda-se um pote que seja fácil de cortar, para a passagem de cabos; e que permita a alocação dos componentes sem que estes fiquem apertados dentro do recipiente. Para o dispositivo desenvolvido no projeto, foi utilizado um pote de sorvete devidamente pintado para fins estéticos, onde os equipamentos ficaram presos à tampa, de modo que o servo motor foi alocado ao fundo do pote, conforme Figura 3.2.

**Observação!** Na elaboração de um protótipo, nem sempre é interessante a elaboração de elementos complexos pois não se sabe da viabilidade do produto final. Portanto, a utilização de sucatas e materiais de baixo custo são extremamente interessantes.



Figura 3.2: Componentes alocados em pote de sorvete.



Elaborada pelo autor (2021).



```

9 Servo myServo; // Nome atribuido ao microservo
10
11 void setup() {
12     Serial.begin(9600); // Painel para exibir de dad
13     pinMode(trig, OUTPUT); // Configurar pino 2 como s
14     pinMode(echo, INPUT); // Configurar pino 3 como en
15     digitalWrite(trig, 0); // Atribuir ao pino 2 o valo
16     myServo.attach(servo); // Atrelar o servo ao pino 4
17     myServo.write(0); // Ordenar servo que va para
18 }
19
20 void loop() {
21     for (int i = 0; i < 180; i++) { // Repetir a variav
22                                     // i de 0 ate 180

```

## 4. Software

### 4.1 Arduino IDE

Para efetuar a programação do microcontrolador no Arduino, é necessário primeiramente instalar a interface de desenvolvimento - IDE, disponibilizada no site oficial da plataforma na aba Software conforme Figura 4.1.

Figura 4.1: Página oficial Arduino e aba para download da interface IDE.



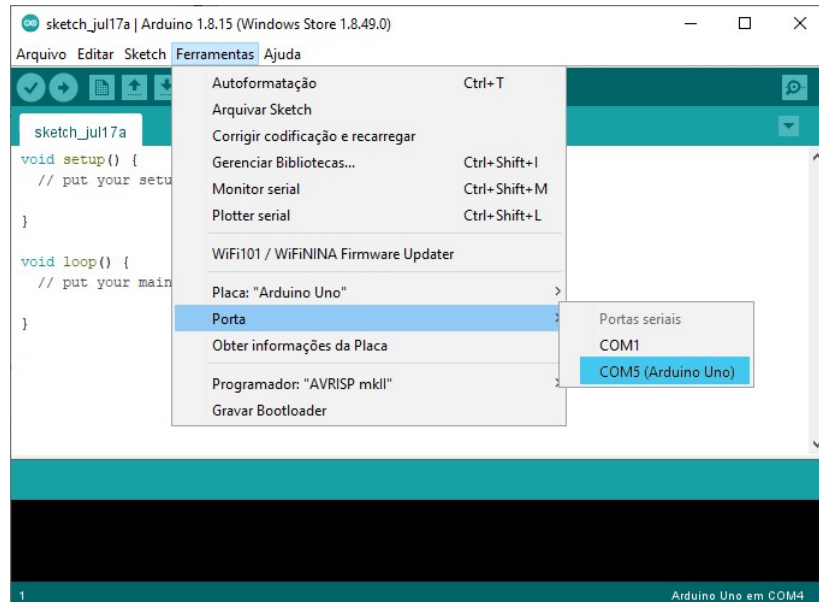
Elaborada pelo autor (2021).

Após prosseguir com a instalação, basta executar o aplicativo Arduino IDE para que sejamos conduzidos a tela inicial já apresentado por aqui na Figura 2.2.

Uma vez com o aplicativo aberto e a placa Arduino devidamente conectada em uma das portas USB, é necessário identificar se o computador está reconhecendo a placa Arduino. Isso ocorre

pois o computador possui diversas portas de comunicação com elementos USBs diversos. Assim, conforme a Figura 4.2, deve-se configurar o aplicativo para se comunicar com a porta correta.

Figura 4.2: Identificação do Arduino na porta USB correta.



Elaborada pelo autor (2021).

**Observação!** O passo anterior também é necessário quando se possui mais de uma placa Arduino conectada ao mesmo computador, uma vez que os códigos gerados Arduino IDE só podem ser submetidos a uma placa por vez.

Após a elaboração do código, a submissão deste para o microcontrolador é realizada em dois passos: Verificar e Carregar, cada um representado por um botão na interface IDE, conforme Figura 4.3. O botão de Verificar faz uma compilação prévia do código para identificar possíveis falhas na estrutura do código antes desse ser submetido através do botão Carregar.

## 4.2 Código do dispositivo

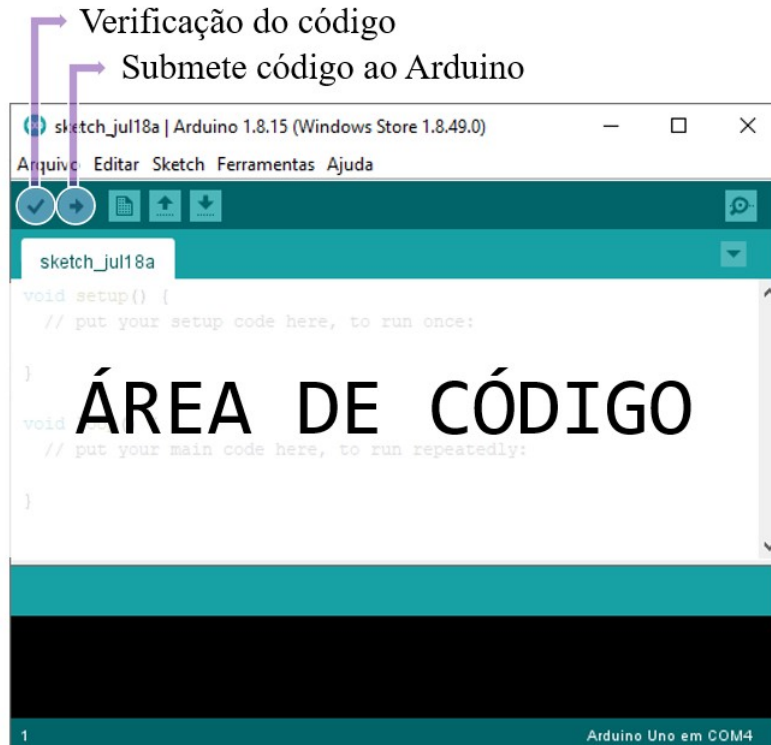
Para melhor compreender o que pretendemos desenvolver com o dispositivo, é conveniente elaborar uma estrutura via fluxograma do código que vamos desenvolver de forma a facilitar a escrita dele depois. Essa estrutura pode ser verificada na Figura 4.4

Conforme verificado no capítulo introdutório, o código utilizado na programação do Arduino é estruturado em linguagem de programação C. Apesar de no código que utilizaremos para esse dispositivo utilizar poucas estruturas de programação, é possível incrementar o código de forma a apresentar novas informações, características e comportamento.

**Observação!** Assim como a linguagem de programação C++, o código compreendido é *case sensitive*. Isto é, o código diferencia as letras maiúsculas de letras minúsculas.

Para apresentar o código utilizado na elaboração do dispositivo, distribuiremos o código em três etapas que devem ser unidas ao submeter ao Arduino, conforme código completo ao final do capítulo.

Figura 4.3: Área de código e botões para submeter o código ao microcontrolador do Arduino.



Elaborada pelo autor (2021).

### 4.3 Definições e pacotes

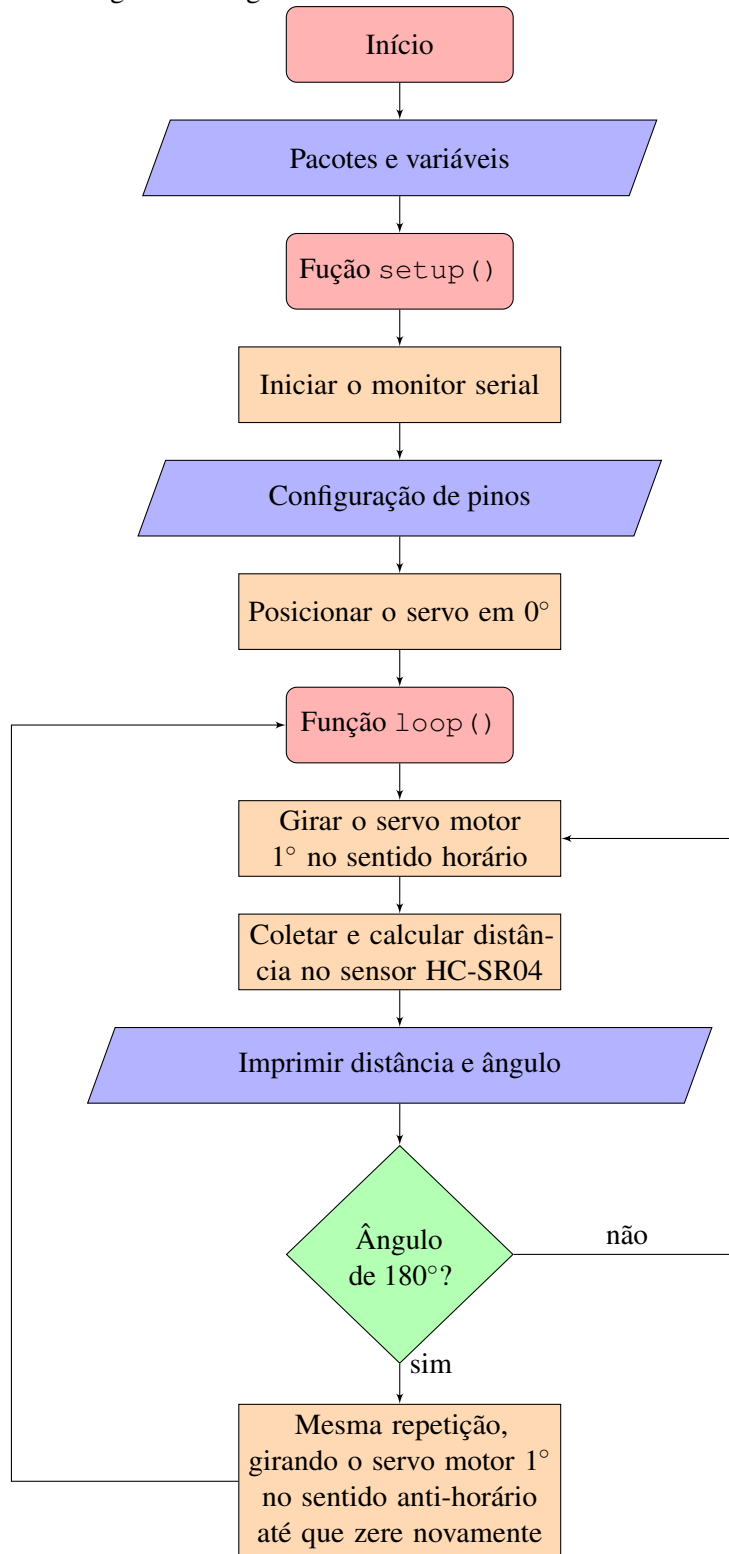
Iniciando a programação do código, ver Figura 4.5, temos que uma característica da prototipagem rápida é a utilização de pacotes pré desenvolvidos pela comunidade que permitem fácil manipulação de diferentes componentes. Como o dispositivo utiliza um servo motor, existe um pacote chamado `Servo.h` que permite controle do componente de forma eficiente e sem que o usuário precise se aprofundar no controle eletrônico do servo motor.

Através do pacote `Servo.h`, define-se um nome para o servo motor o qual será utilizado ao longo do código para definir o pino de controle que está conectado - através do comando `nome.attach(pino)`, bem como o ângulo que o servomotor gira `nome.write(ângulo)`.

Ainda nessa primeira etapa do código, algumas definições precisam ser feitas referente as variáveis que utilizaremos ao longo do código. São os casos da variável do tipo `float` para distância e uma variável do tipo `int` para contabilizar a quantidade de pulsos sonoros emitidos pelo sensor ultrassônico.

**Observação!** Uma prática comum para boa estrutura do código, é nomear as portas em que foram conectadas as componentes. No caso do dispositivo, as portas digitais 2, 3 e 4 foram utilizadas para as conexões do sensor ultrassônico - echo no pino 2 e trig no pino 3 - e o servo motor no pino 4, de forma que foi definido as palavras `trig`, `echo` e `servo` para essas portas. Assim, quando em algum momento o código chamar a palavra `trig`, o Arduino identificará que se trata do valor 2.

Figura 4.4: Fluxograma de algoritmo utilizado no desenvolvimento do protótipo.



Elaborada pelo autor (2021).

Figura 4.5: Início de código: Definições e pacotes

```
1 #include <Servo.h>
2
3 Servo myServo;
4
5 float distance;
6 int pulse;
7
8 #define trig 2
9 #define echo 3
10 #define servo 4
```

Elaborada pelo autor (2021).

#### 4.4 Função `setup()`

Na função `setup()`, ver Figura 4.6, iniciamos o monitor serial - `Serial.begin(9600)` - o qual é uma das formas de comunicação do Arduino com o usuário via janela na própria interface IDE pelo computador. Nesse monitor serial, será exibido as informações coletadas pelo sensor ultrassônico e em que ângulo se encontra o servo motor quando é coletada tal informação.

Aqui também classificamos os pinos digitais que foram utilizados pelo sensor ultrassônico. Conforme estabelecido anteriormente, a porta `trig`, referente à porta 2, está com característica de saída pois envia a onda sonora e a porta `echo`, referente à porta 3, está com característica de entrada, pois recebe o retorno da onda sonora.

O comando `digitalWrite` informa a porta digital um valor ligado ou desligado que em código é representado por 1 ou 0, ou `HIGH` ou `LOW`, respectivamente.

Por último, configura-se o servo motor com os comandos descritos na seção anterior. Atrémos a porta `servo`, referente à porta 4, o servo motor que foi nomeado de `myServo`, e instruímos que o servo motor vá para o ângulo zero ou posição inicial.

Figura 4.6: Função `setup()`: Configuração inicial de componentes

```
1 void setup() {
2     Serial.begin(9600);
3     pinMode(trig, OUTPUT);
4     pinMode(echo, INPUT);
5     digitalWrite(trig, 0);
6     myServo.attach(servo);
7     myServo.write(0);
8 }
```

Elaborada pelo autor (2021).

#### 4.5 Função `loop()`

Para a função `loop()`, ver Figura 4.7, temos que atribuir um laço de repetição pois, para cada ângulo percorrido pelo servo motor, queremos fazer a coleta de informações do sensor ultrassônico.

Dessa forma, estabelecemos no laço de repetição uma contagem  $i$  que começa em 0 e vai até 180, onde a cada ciclo de repetição  $i$  aumenta seu valor em uma unidade.

Figura 4.7: Função loop(): Etapa de execução

```

1  for (int i = 0; i < 180; i++) {
2      myServo.write(i);
3      digitalWrite(trig, 1);
4      delayMicroseconds(10);
5      digitalWrite(trig, 0);
6      pulse = pulseIn(echo, 1);
7      distance = (pulse * 0.017221195) / 100;
8      Serial.print("Distancia_em_unidades:_");
9      Serial.println(distance);
10     Serial.print(" ngulo _em_graus:_");
11     Serial.println(i);
12     Serial.println("_");
13     delay(300);
14 }

```

Elaborada pelo autor (2021).

Os valores atribuídos nessa contagem já serão a base do ângulo do servo motor, através do comando `myServo.write(i)`, o que nos garantirá o giro do motor a cada repetição a que submetemos a variável  $i$ , até que o servo motor percorra 180 graus.

Dentro de cada giro do motor, queremos que o sensor ultrassônico colete as informações de objetos que se encontrem na frente deste. Assim, liga-se o sensor através de sua porta `trig` por 10 microssegundos e desliga-o novamente. O comando `pulseIn` irá capturar a duração de um pulso no pino `echo` de forma que exporta a duração que esse pulso teve.

Essa duração, atribuída a variável `pulse` é a que será utilizada para determinar a distância de que o elemento que refletiu a onda sonora se encontra. A fórmula para determinação dessa distância é

$$d = \frac{t_{ECHO} \cdot V_{som}}{2}$$

que, conforme apresentado anteriormente, nos leva ao fator 0.0170145 para determinar uma distância em centímetros que o objeto se encontra.

Como a coleta das informações do dispositivo foi realizada em amplo espaço, optou-se pela unidade de distância em metros. Portanto, tomou-se a divisão por 100 do valor calculado.

Por fim, utilizou-se do monitor serial para exibir as informações coletadas, para cada ângulo registrado no servo motor, através dos comandos `Serial.print()` e `Serial.println()` para exibir textos e valores, respectivamente.

Como o Arduino realiza a leitura do código de maneira rápida, é necessário que seja ordenado que demore o suficiente para que seja possível acompanhar o giro do servo motor e a coleta das informações. Para isso, utiliza-se o comando `delay(tempo)` com o valor em milissegundos que gostaríamos que o Arduino atrasasse na execução do seu código.



**Observação!** Para o dispositivo elaborado, definiu-se 300 milissegundos de espera para cada passo da variável *i*, porém esse valor pode ser atribuído livremente. Caso os estudantes tenham dificuldades para conseguir coletar as informações, basta utilizar um valor maior como 1000ms, ou 1 segundo, por exemplo.

Note que o algoritmo apresentado na Figura 4.7, efetua apenas o giro do motor de 0 graus até 180 graus e portanto, é necessário que o servo motor faça o giro contrário para que retorne à posição de 0 graus. Dessa forma, basta repetir o mesmo código, dentro de um novo laço de repetição. Atribui-se a variável *i* o valor 180 e diminui em uma unidade para cada repetição, até que se chegue novamente à posição inicial, conforme verificaremos a seção a seguir.

## 4.6 Código completo

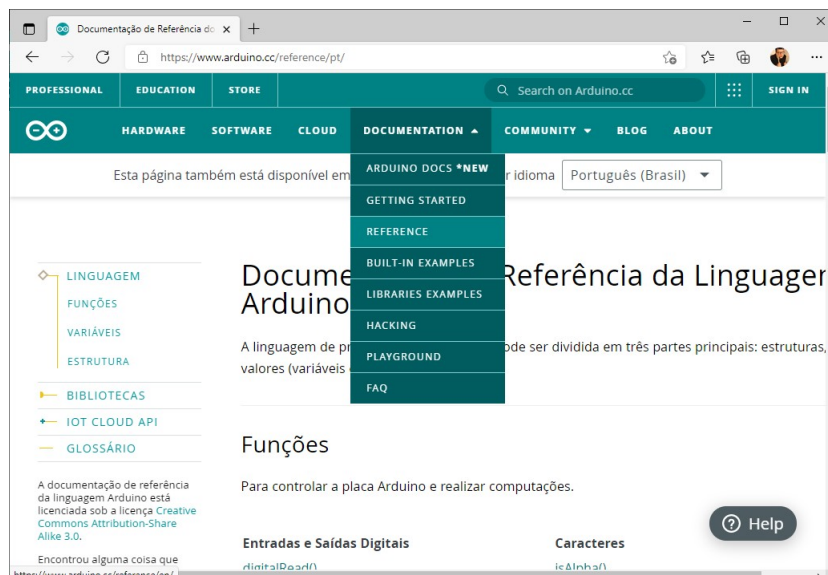
Estabelecidas as estruturas do código, basta então que essas sejam unidas para submeter, via Arduino IDE, à placa Arduino que está conectada ao computador.

No código completo da Figura 4.9, foi inserido alguns comentários sobre os comandos que foram utilizados. Os comentários são escritos após comando `"/ /"` e estes não são comandos executados pelo microcontrolador.

A utilização de seções de comentários em meio ao código não é obrigatória, porém é de boa prática. Quando se busca por exemplos ou explicações sobre determinados comandos, é comum tal prática pois permite que o usuário identifique rapidamente o que determinada linha de comando faz.

Todos os comandos utilizados na elaboração desse código, podem ser vistos com maiores detalhes na documentação disponibilizada página oficial da plataforma Arduino, conforme Figura 4.8.

Figura 4.8: Documentação disponibilizada na página oficial Arduino.



Elaborada pelo autor (2021).

Figura 4.9: Codificação do algoritmo da Figura 4.4 para a linguagem C++.

```
1 #include <Servo.h> // Pacote para controle do micro servo
2 #define trig 2     // Atribuir o numero 2 a palavra trig
3 #define echo 3    // Atribuir o numero 3 a palavra echo
4 #define servo 4   // Atribuir o numero 4 a palavra servo
5
6 int distance; // Definicao de variaveis
7 int pulse;
8
9 Servo myServo; // Nome atribuido ao microservo
10
11 void setup() {
12     Serial.begin(9600); // Painel para exibir de dados no IDE
13     pinMode(trig, OUTPUT); // Configurar pino 2 como saida
14     pinMode(echo, INPUT); // Configurar pino 3 como entrada
15     digitalWrite(trig, 0); // Atribuir ao pino 2 o valor 0 ou LOW
16     myServo.attach(servo); // Atrelar o servo ao pino 4
17     myServo.write(0); // Ordenar servo que va para posicao 0
18 }
19
20 void loop() {
21     for (int i = 0; i < 180; i++) { // Repetir a variavel inteira
22                                     // i de 0 ate 180
23         myServo.write(i); // Ordenar ao servo que va para posicao i
24
25         digitalWrite(trig, 1); // Processo descrito para
26         delayMicroseconds(10); // funcionamento do sensor
27         digitalWrite(trig, 0); // HC-SR04, no Capitulo 3.12
28         pulse = pulseIn(echo, 1);
29         distance = pulse * 0.0170145;
30
31         Serial.print("Distancia_em_cm:_"); // Impressao dos dados
32         Serial.println(distance); // obtidos e calculados
33         Serial.print("Angulo_em_graus:_"); // no monitor serial IDE
34         Serial.println(i);
35         Serial.println("_");
36         delay(300);
37     }
38
39     for (int i = 179; i >= 0; i--) {
40
41         // Mesmo codigo utilizado no primeiro laco de repeticao
42
43     }
44 }
```

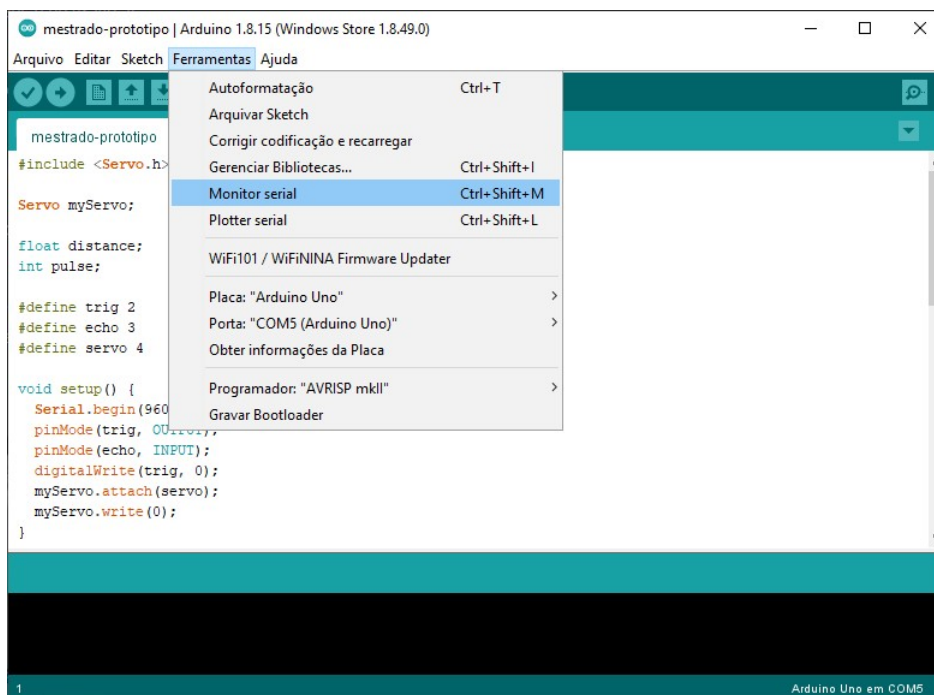
Elaborada pelo autor (2021).

## 5. Teste

Uma vez feita as conexões entre os componentes e a placa Arduino e tendo conectada ela ao computador via porta USB, você verificará que assim que a etapa de envio do código pelo Arduino IDE for concluída, o Arduino já iniciará a execução do código zerando a posição do servo motor, conforme estabelecido como etapa inicial do código, e já iniciará o giro do servo motor de grau em grau, respeitando o tempo definido para o atraso em cada passo da repetição.

A exibição dos dados coletados, no entanto, se dá somente quando abrimos o monitor serial do Arduino IDE, conforme Figura 5.1

Figura 5.1: Janela de exibição do monitor serial.

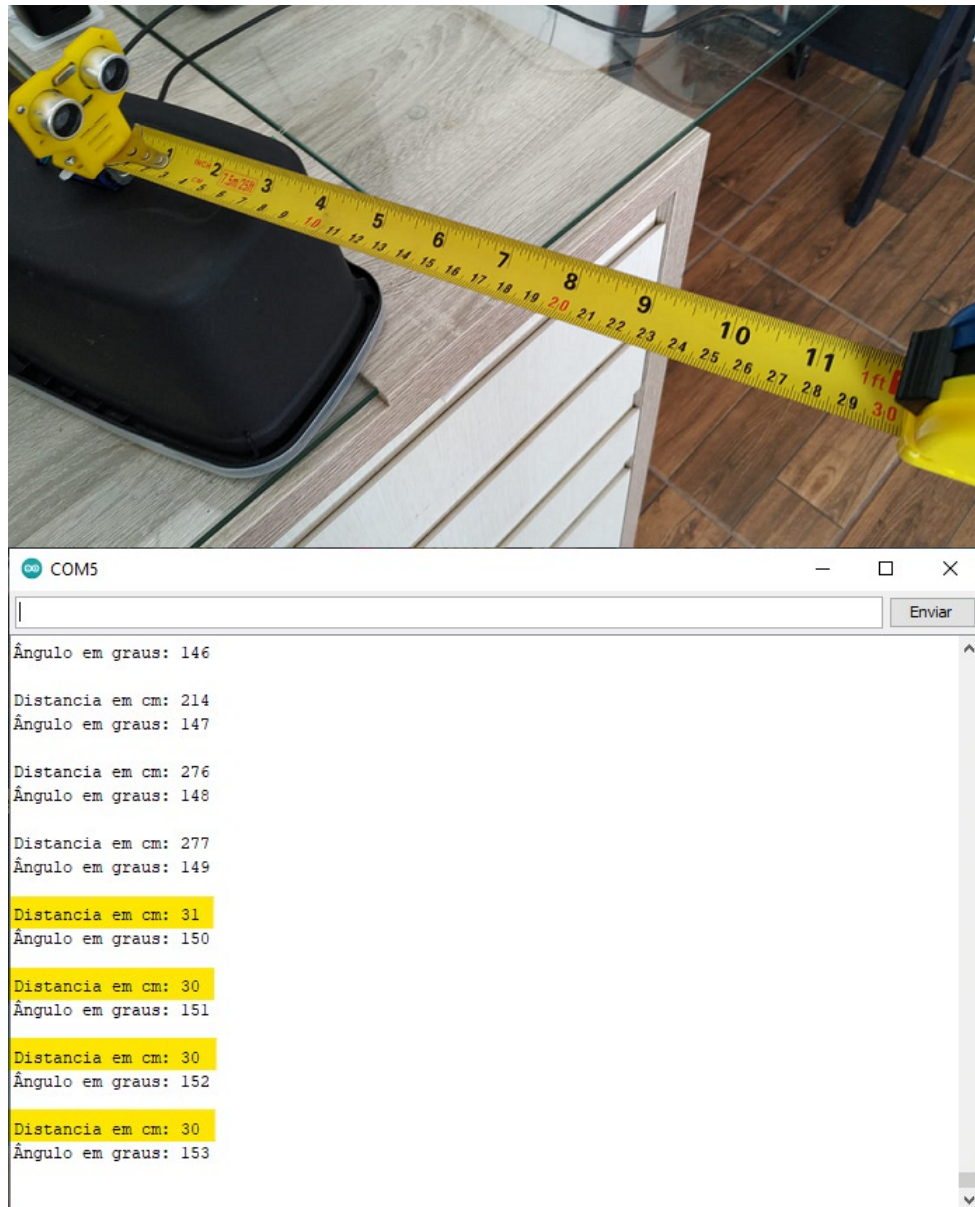


Elaborada pelo autor (2021).

A aferição do dispositivo pode ser realizada através do auxílio de algum material de medição

como régua, trena ou fita métrica. É possível então, verificar no monitor serial se o dispositivo capta corretamente a distância de algo posicionado a frente do sensor ultrassônico e também se a mudança do ângulo durante o giro do servo motor é exibido, conforme Figura 5.2.

Figura 5.2: Exibição do monitor serial: distância captada pelo sensor ultrassônico e ângulo no servo motor



Elaborada pelo autor (2021).

**Observação!** Como o sensor ultrassônico faz uma leitura em forma de cone, é necessário que haja auxílio visual na coleta do elemento que se posiciona na frente do sensor.



# Atividades

<b>6</b>	<b>Sugestão de atividades</b> .....	<b>39</b>
6.1	Programação e prototipagem	
6.2	Geometria analítica e trigonometria	



Elemento - ? - Distância - Ângulo  
CRL 498 60°

## 6. Sugestão de atividades

O leitor deve ter percebido que existem aqui dois caminhos para se seguir no campo de possíveis atividades para se trabalhar no ensino de matemática.

### 6.1 Programação e prototipagem

Devido à praticidade e facilidade de elaboração e execução de um protótipo utilizando Arduino, um dos caminhos é a elaboração não precisamente deste protótipo mas de outros produtos e experimentos envolvendo não só outros componentes.

Segundo a BNCC (2018), a habilidade específica EM13MAT406 nos apresenta "Utilizar os conceitos básicos de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática".

Dessa forma, utilizar o Arduino suas componentes e acessórios, pode proporcionar ao estudante não somente o raciocínio lógico por trás da elaboração de algoritmos.

Algumas sugestões de atividades dentro desse cenário:

**Atividade: 1** Acender LED via Arduino

**Atividade: 2** Controlar acendimento de vários LEDs via Arduino

**Atividade: 3** Acendimento de LEDs, condicionados a algum sensor

**Observação!** O Arduino possui uma comunidade bastante ativa que compartilha experimentos e atividades. Dessa forma é possível encontrar em diversos sites, materiais e listas de exercícios que complementam, enriquecem e possibilitam explorar novas potencialidades que a prototipagem rápida pode proporcionar no ensino.

Um canal com apostilas para aprofundar conhecimentos na área de prototipagem:  
<https://blog.eletrogate.com/apostilas/>

### 6.2 Geometria analítica e trigonometria

Outro caminho que se desdobra na utilização do protótipo desenvolvido é que este permita com que o estudante aplique e identifique a matemática como elemento fundamental na compreensão das informações que este equipamento proporciona.



Como os estudantes estão familiarizados com o georreferenciamento através do plano cartesiano, amplamente durante o curso de geometria analítica, o dispositivo elaborado possibilita que esse georreferenciamento seja obtido através de outro sistema de coordenadas.

Dessa forma, basta que seja posicionado elementos a frente do dispositivo, que ele determinará a distância que esse elemento se encontra do dispositivo e o ângulo aplicado no servo motor. Tais informações possibilitam ao estudante

- Compreender a trigonometria no triângulo retângulo como elemento fundamental para conversão de coordenadas polares para coordenadas cartesianas.
- Identificar que elementos podem ser posicionados em sistemas de coordenadas que sejam mais convenientes à aplicação em que se está inserido.

Portanto, o dispositivo permite que os mesmos exercícios visualizados nos livros de exercícios, possam ser simulados em ambiente real, como por exemplo;

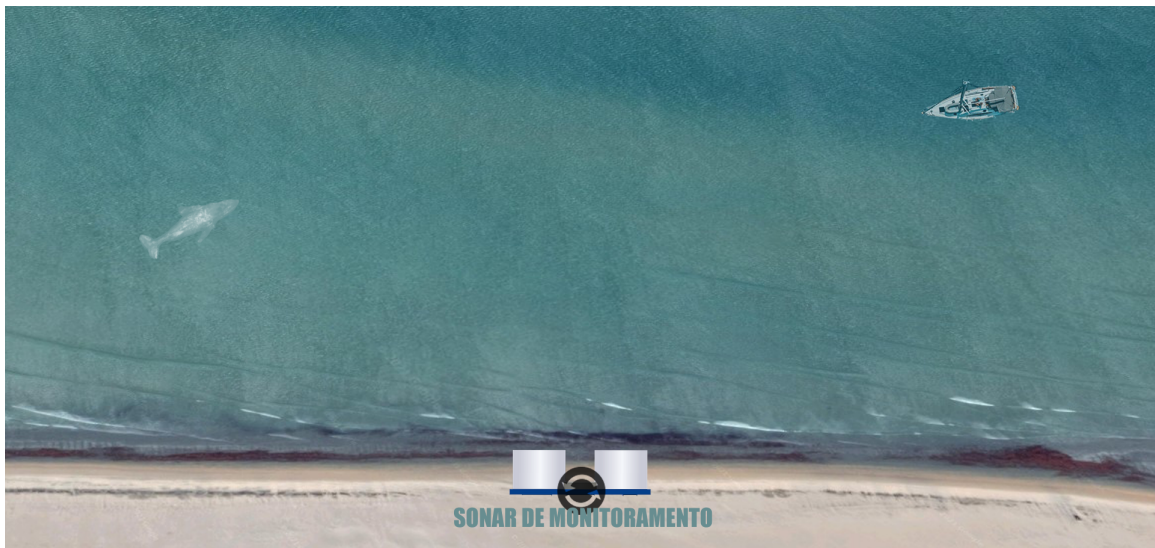
**Atividade: 4** Identificar a equação de reta que passa por dois objetos localizados na frente do dispositivo.

**Atividade: 5** Calcular a área entre três objetos em frente ao dispositivo.

**Atividade: 6** Verificar se um objeto se encontra dentro da área de cobertura radial de outro objeto em frente ao dispositivo.

**Observação!** Tais propostas podem ter ainda mais significado se forem explorados dentro de contextos reais, como por exemplo o monitoramento de uma região costeira. Com os objetos sendo elementos que ali trafegam: barcos, peixes, pedras etc. conforme Figura 6.1.

Figura 6.1: Proposta de monitoramento em costa marítima.



Elaborada pelo autor (2021).



# IV

## Considerações finais

<b>7</b>	<b>Conclusão .....</b>	<b>43</b>
	<b>Bibliografia .....</b>	<b>45</b>



Elemento - ? - Distancia - Ângulo  
CRL 498 60°

## 7. Conclusão

Esse trabalho se propôs à condução de construção de um protótipo que permita georreferenciar objetos em torno do dispositivo, de forma a permitir ao estudante compreender relações entre diferentes formas de coordenadas em atividades diversas, dentro do escopo de geometria analítica utilizando trigonometria.

Além disso, buscou-se apresentar as potencialidades da prototipagem rápida, via Arduino e materiais diversos, para com o ensino de matemática.

Este dispositivo foi desenvolvido de forma simples e objetiva ao que se propõe, no entanto fica o convite a cada colega professor que venha a desenvolver a atividade similar, que busque melhorias que possam contribuir com o ensino aprendizagem dos estudantes.



Elemento - ? - Distancia - Ângulo  
CRL 498 60°

## Bibliografia

- [1] Arduino. “Página Arduino”. Em: (2021). Acessado: 12 jun. 2021 (ver página 9).
- [2] Ministério da Educação BRASIL. “Base Nacional Comum Curricular”. Em: (2018) (ver página 39).
- [3] Fredy Criollo-Sánchez et al. “Implementation of a basic Sonar of Echolocation for Education in Telecommunications”. Em: *International Journal of Advanced Computer Science and Applications* 9 (2018). DOI: 10.14569/IJACSA.2018.091106 (ver página 7).
- [4] Johnny R. Hilburn David E. Johnson. “Fundamentos De Análise De Circuitos Elétricos”. Em: (1994) (ver páginas 15, 16).
- [5] Alberto Gaspar. “Eletromagnetismo e Física Moderna”. Em: *Compreendendo a Física: 3 3* (2013) (ver páginas 15, 16).
- [6] Vitor Vidal. “Apostila KIT Arduino Start”. Em: (2018). Disponível em: <https://blog.eletrogate.com/apostilas/>. Acessado: 12. jun. 2021 (ver página 16).

