

UNIVERSIDADE FEDERAL DO PARANÁ

FLAVIA MARIA SCHUTZ GRR20195838

IZABELLA CALAIS FERNANDES ISO20190290

KAREN DA LUZ GUARNIERI GRR20195848

THIAGO BATISTA DOS SANTOS MARTINS GRR20195874

MEDINDO DISTÂNCIAS COM ARDUÍNO: MÓDULO HC-SR04 E LED'S

CIDADE

2019

RESUMO

Neste trabalho utilizamos uma placa de prototipagem eletrônica, chamada Arduíno, para desenvolver um dispositivo de índice de proximidade. Para isso utilizamos um sensor ultrassônico que medirá a distância de um objeto; esta será enviada para a placa, onde serão processadas as informações, através de um código, e determinarão as ações dos demais componentes.

Que dependendo do quão perto o objeto se encontra do sensor os parâmetros de saída de dados são alterados. Os Outputs por sua vez, são LED's e um Buzzer. Conforme a distância vai diminuindo estes têm frequências maiores, parando apenas quando o parâmetro de leitura é zero.

Foram utilizadas duas linguagens, a linguagem padrão da IDE do Arduíno C++, e a linguagem Python de maneira Ilustrativa, para uma melhor compreensão por parte dos nossos colegas de turma, pois a matéria em questão tem essa linguagem como referência.

LISTA DE FIGURAS

FIGURA 1 - ARDUÍNO UNO REV3.....	18
FIGURA 2 - MICROCONTROLADOR.....	19
FIGURA 3 - PINAGEM DA PLACA UNO.....	19
FIGURA 4 - CONECTORES DE ALIMENTAÇÃO.....	20
FIGURA 5 - SENSOR ULTRASSÔNICO	22
FIGURA 6 - LEITURA DO SENSOR	23
FIGURA 7 - SITE PARA DOWNLOAD.....	24
FIGURA 8 - DOWNLOAD DA BIBLIOTECA.....	25
FIGURA 9 - INSTALAÇÃO DA BIBLIOTECA.....	25
FIGURA 10 - DIRETORIO DE ARQUIVOS.....	26
FIGURA 11 - ULTRASONIC.....	26
FIGURA 12 - ULTRASONIC.TIMING	27
FIGURA 13 - ULTRASONIC.CONVERT	27
FIGURA 14 - DECLARAÇÃO DE VARIÁVEIS	27
FIGURA 15 - DEFININDO A <i>SERIAL</i>	28
FIGURA 16 - MONITOR SERIAL.....	29
FIGURA 17 - FUNÇÃO DIGITAL WRITE	29
FIGURA 18 – APLICAÇÃO DA FUNÇÃO I	30
FIGURA 19 - APLICAÇÃO DA FUNÇÃO II	30
FIGURA 20 - APLICAÇÃO DA FUNÇÃO III	31
FIGURA 21 - INÍCIO DO CÓDIGO.....	32
FIGURA 22 - CONFIGURAÇÃO NA PINMODE.....	33
FIGURA 23 - PARÂMETRO INEXISTENTE.....	33
FIGURA 24 - PARÂMETRO DUPLICADO	34
FIGURA 25 - SOLICITAÇÃO DE DISTÂNCIA.....	34
FIGURA 26 - SOLICITAÇÃO DA EXECUÇÃO.....	35
FIGURA 27 - EXECUÇÃO DA FUNÇÃO DIGITAL WRITE I	36
FIGURA 28 - EXECUÇÃO DA FUNÇÃO DIGITAL WRITE II	36
FIGURA 29 - FINAL DO CÓDIGO.....	37
FIGURA 30 - IDE ARDUÍNO	38
FIGURA 31 - BARRA DE FERRAMENTAS	39
FIGURA 32 - CONFIGURA PORTA.....	39

FIGURA 33 - ATIVAÇÃO DA PORTA	40
FIGURA 34 - SENSOR	42
FIGURA 35 - PINO GND	43
FIGURA 36 - PINO ECHO.....	43
FIGURA 37 - PINO TRIGGER.....	44
FIGURA 38 - PINO VCC	45
FIGURA 39 - INDICAÇÃO DE CONDUTORES LED	46
FIGURA 40 - LED AZUL	47
FIGURA 41 - LED VERDE	47
FIGURA 42 - LED AMARELO	48
FIGURA 43 - LED VERMELHO.....	49
FIGURA 44 - INDICAÇÃO DE CONDUTORES BUZZER	50
FIGURA 45 - BUZZER	50
FIGURA 46 - JUMPER GND	51
FIGURA 47 - MAQUETE	52
FIGURA 48 - CARRINHO.....	53

LISTA DE TABELAS

TABELA 1 - MATERIAIS DA MONTAGEM	41
--	----

SUMÁRIO

1 INTRODUÇÃO	16
1.1 JUSTIFICATIVA	16
1.2 OBJETIVOS	16
1.2.1 Objetivo geral	16
1.2.2 Objetivos específicos.....	16
ARDUÍNO.....	17
1.3 COMO FUNCIONA?.....	17
1.3.1 Hardware.....	18
1.3.1.1 Entradas e saídas da Placa UNO	19
1.3.1.2 Alimentação da placa	20
1.3.2 Software	21
1.3.2.1 Leitura do Código	21
1.4 MODULOS DO ARDUÍNO	22
1.4.1 Entendendo o Módulo: Sensor HC-SR04.....	22
1.4.1.1 Funcionamento e Medição	22
2 CÓDIGO EM C++.....	24
2.1 LINGUAGEM C++	24
2.2 BIBLIOTECAS UTILIZADAS	24
2.3 O CÓDIGO	27
3 CÓDIGO EM PYTHON.....	32
4 MATERIAL E MÉTODOS	38
4.1 SOFTWARE	38
4.2 HARDWARE	41
4.2.1 MANUAL de montagem.....	41
4.2.1.1 Sensor HC-SR04	42
4.2.1.2 LED's.....	46
4.2.1.3 BUZZER CLDZ.....	50
5 APRESENTAÇÃO DOS RESULTADOS	52
6 CONSIDERAÇÕES FINAIS	54
6.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS	54
REFERÊNCIAS.....	55

1 INTRODUÇÃO

Com a tecnologia desenvolvida por Massimo Banzi e outros colaboradores, emergiu então uma crescente das placas de prototipagem eletrônica, onde por possuir um código *Open-source* ganhou rapidamente visibilidade, e hoje em dia oferece uma área vasta de projetos e ideias a se desenvolver.

1.1 JUSTIFICATIVA

A fim de explorar mais áreas que a programação poderia nos oferecer, buscamos por algo que proporcionaria uma gama de conhecimentos. Assim, escolhemos trabalhar com as placas de prototipagem, em específico a Placa Arduino, pois conseguiríamos aprender não só com a parte de software, mas também um pouco sobre o hardware.

1.2 OBJETIVOS

Aprender mais sobre a parte técnica de hardwares e softwares, e desenvolver um projeto aplicável no dia a dia.

1.2.1 Objetivo geral

De modo geral, buscamos desenvolver algo aplicável no dia a dia, e que demonstre que a programação vai muito além de ambientes técnicos restritos a um computador, e que pelo contrário está presente em muitas coisas cotidianas.

1.2.2 Objetivos específicos

Desenvolver um dispositivo de índice de proximidade, utilizando um sensor ultrassônico que medirá a distância de um objeto; esta será enviada para a placa, onde serão processadas as informações, através de um código, e determinarão as ações dos demais componentes, estes que são LED's e um Buzzer.

ARDUÍNO

Arduíno ou uma placa Arduíno, é uma plataforma *Open-source* de prototipagem eletrônica com hardware e software flexíveis, destinada a pessoas de diversas áreas que estejam interessadas em criar objetos ou ambientes interativos.

Em outras palavras, e em termos gerais um Arduíno é uma “plaquinha” com capacidade de funcionamento de um pequeno computador, onde se é possível programar como seus *inputs* e *outputs* deverão se comportar junto a outros módulos e componentes externos, esses serão exemplificados e explicados no decorrer do trabalho, e podem ser acoplados na mesma.

Essa tecnologia foi desenvolvida em 2005 pelo italiano Massimo Banzi, e outros colaboradores; o projeto surgiu com o objetivo de facilitar o ensino de eletrônica para alunos de Design, pois na época não havia placas com um custo acessível disponíveis no mercado, então Banzi decidiu criar um dispositivo que fosse semelhante à estrutura de um computador, surgindo então a placa de baixo custo Arduíno.

Um dos motivos para o Arduíno ter feito tanto sucesso é sua plataforma *Open-source*, que é um termo em inglês que significa Código aberto, ou seja, o código fonte pode ser adaptado e alterado a escolha do usuário. Logo por não possuir custo de licença, ganhou uma comunidade rapidamente, onde são compartilhadas ferramentas, bibliotecas, criações de diversos desenvolvedores, e tudo sem restrições para qualquer um que deseje criar um novo projeto.

1.3 COMO FUNCIONA?

O funcionamento de um Arduíno é formado por dois processos, o Hardware, que seria a Placa e todos os Módulos e componentes dependendo do projeto, e a parte de Software, onde escrevemos o código por meio de um interpretador chamado IDE Arduíno.

1.3.1 Hardware

Existem diversos tipos de placas de prototipagem destinadas a projetos diferentes, neste usaremos o Arduino UNO, modelo: Placa UNO Rev3, este pode ser visualizado na FIGURA 1.

FIGURA 1 - ARDUÍNO UNO REV3

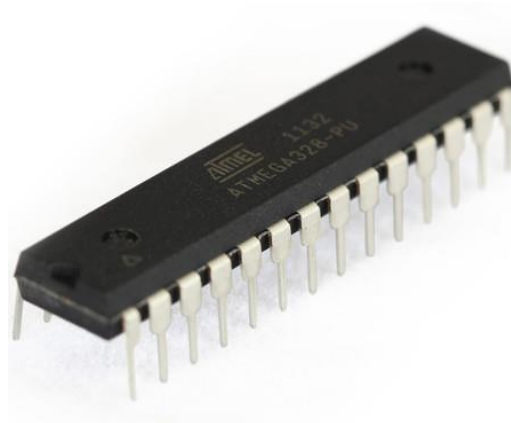


FONTE: Store Arduino UNO

O Arduino é composto de diversos componentes que auxiliam a comunicação com o mundo externo. Como já mencionado, o Arduino é comparado com um pequeno computador, isso porque possui a capacidade de execução semelhante a um computador.

O principal componente que faz essa semelhança é o Microcontrolador, este que é a peça responsável por fazer a leitura do código, e com auxílio de outros componentes tem a capacidade de processar o que foi desenvolvido pelo criador, e indicar para os demais circuitos sua execução. O Arduino UNO, por exemplo, usa o microcontrolador ATmega328, este pode ser visto na FIGURA 2.

FIGURA 2 - MICROCONTROLADOR



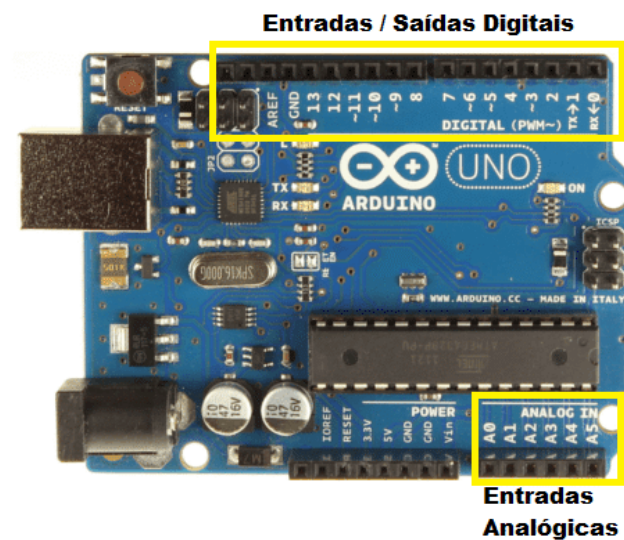
FONTE: Instituto Digital

Embora execute coisas como um computador, seu uso é limitado comparado a computadores de verdade, processos como o tamanho e processamento de memória RAM ainda ficam abaixo de um computador normal, mas tem baixo custo, consumo energético reduzido, e assim uma aplicação eficaz.

1.3.1.1 Entradas e saídas da Placa UNO

A placa Arduino UNO possui pinos, ou também chamadas de portas, de entrada e saída digitais e analógicas, e sua pinagem padrão segue o que indica a FIGURA 3.

FIGURA 3 - PINAGEM DA PLACA UNO



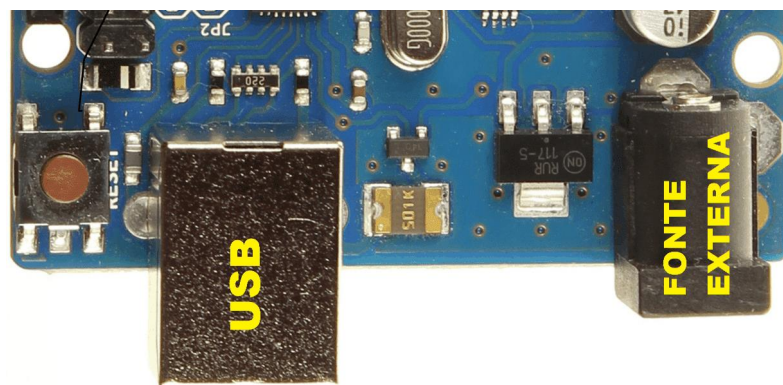
FONTE: Embarcados

Conforme exibido na figura, a placa Arduino UNO possui 14 pinos que podem ser usados como entrada ou saída digitais, e estas operam em 5 Volts cada; já para a parte analógica são 6 entradas onde cada uma tem uma resolução de 10 bits.

1.3.1.2 Alimentação da placa

Para conectar o Arduino ao computador é necessário um conector USB do tipo B, este cabo além de transportar a informação, é uma opção de alimentação de energia da placa, além disto, uma fonte de alimentação externa pode energizar a placa sem a necessidade de um cabo, por meio de uma bateria.

FIGURA 4 - CONECTORES DE ALIMENTAÇÃO



FONTE: Embarcados

Caso queira utilizar uma fonte externa de energia, deve-se usar o conector Power Jack, onde o limite de tensão é de 6V a 20V. Embora se a tensão estiver muito próxima desses extremos possa causar instabilidade ou superaquecer e danificar a placa. Sendo assim, é recomendado para tensões de fonte externa valores de 7V. A 12V.

Isso não é necessário no caso do uso do USB, já que a tensão é estabilizada pelo regulador de tensão. Sendo assim, a placa é alimentada diretamente pelo USB, que possui alguns componentes que protegem contra essas anormalidades de oscilação.

Essa alimentação externa ira servir para conectar a placa com os demais Shields e módulos externos. Deixando então os circuitos energizados. Alguns conectores de alimentação são: IOREF, RESET, 3,3V, 5V, GND e o VIN.

A placa Arduino não pode ser ligada ou desligada, caso queira deixar desativada, basta manter fora de uma alimentação.

1.3.2 Software

O Software do Arduino é uma IDE, ou seja, *Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado, que é executado em um computador por meio do programa próprio, o IDE Arduino.

Uma das vantagens do Arduino é que sua IDE foi desenvolvida como uma multiplataforma escrita em Java, sua interface é simples, e possui uma flexibilidade para atender todos os tipos de criadores e plataformas como Windows, MAC OS X, e Linux.

A IDE pode ser baixada gratuitamente no site oficial do Arduino, onde pode ser escolhida a melhor opção de download conforme plataforma utilizada.

1.3.2.1 Leitura do Código

Na IDE Arduino é onde o desenvolvedor irá fazer o Código, também chamado de Sketch, utilizando de suas bibliotecas e funções; esse sketch posteriormente será upado via Cabo USB diretamente para a placa.

O código irá ser direcionado a placa onde será compreendido e executado através do microcontrolador. Sua criação permite adaptações, porem a linguagem padrão é feita em C++.

A partir do momento que foi feito o *upload* o Arduino não precisa mais do computador: o Arduino executará o *sketch* criado, desde que seja ligado a uma fonte de energia.

Isso não significa que você terá de comprar uma nova placa a cada projeto que queira desenvolver, basta apertar o botão de Reset para que seu Arduino possa receber novos códigos.

1.4 MODULOS DO ARDUÍNO

A versatilidade das placas Arduino tem como fator determinante os módulos ou Shields. Estes são placas de circuitos, sensores, relês, plug's, entre outros componentes com determinações específicas, que expandem a quantidade de coisas que se pode desenvolver com Arduino.

1.4.1 Entendendo o Módulo: Sensor HC-SR04

O módulo utilizado neste trabalho é um Sensor Ultrassônico, este tem como papel medir através do tempo a distância de objetos a partir de sua posição, exemplificado na FIGURA 5.

FIGURA 5 - SENSOR ULTRASSÔNICO



Este componente tem a capacidade de ler distâncias de 2 cm a aproximadamente 4 metros, com uma precisão de 3 mm. Pode ser utilizado de varias maneiras e tem uma gama de opções de projetos para serem executados.

1.4.1.1 Funcionamento e Medição

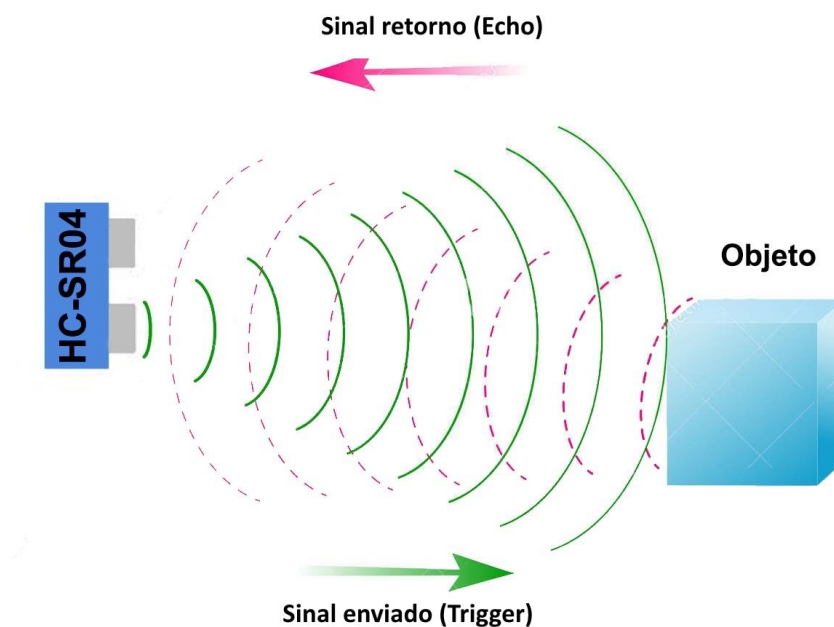
O HC-SR04, envia um pulso de ondas ultrassônicas a partir de um emissor, sendo este configurado no pino *Trigger* do módulo, e capta a onda novamente com o

pino de recebimento *Echo*, o sensor não envia nenhuma medida para o controlador, apenas sinaliza quando o pulso enviado é recebido.

Dessa forma, cabe ao Arduíno à responsabilidade de medir o tempo e calcular as distâncias, tomando como referência o tempo que a onda ultrassônica demora para atingir um objeto e voltar.

Quando medimos o pino *Echo* do sensor com auxílio do osciloscópio, que verifica a variação da largura do pulso, ou seja, o tempo em que levou para ocorrer à variação da distância, assim como apresentado na FIGURA 6.

FIGURA 6 - LEITURA DO SENSOR



FONTE: Portal Vida de Silício

E assim de acordo com o que foi estipulado no código que o Arduíno vai executar, criam-se parâmetros ou aplicações para as medidas que o sensor vai ler.

2 CÓDIGO EM C++

Como a linguagem padrão do Arduino é feita em C++, faremos uma breve introdução, e apresentaremos as bibliotecas, e o código utilizado para reger a parte de Hardware da Placa UNO.

2.1 LINGUAGEM C++

A linguagem C++ é uma mistura de linguagens de baixa e alta compreensão, tendo seu nível médio de entendimento. Como toda linguagem de programação o C++ necessita de informações para que o criador possa programar, no qual algumas delas são:

- Importação de bibliotecas;
- Declaração de variáveis, podendo ser números (int, long, float) ou textos;
- Funções;
- Condicionais, sendo **if** , **else if** ou **else**;
- Etc.

2.2 BIBLIOTECAS UTILIZADAS

Para o nosso código, em C++, utilizamos uma biblioteca específica para o módulo HC-SR04. Para obter a biblioteca, é só fazer o download nesse link:

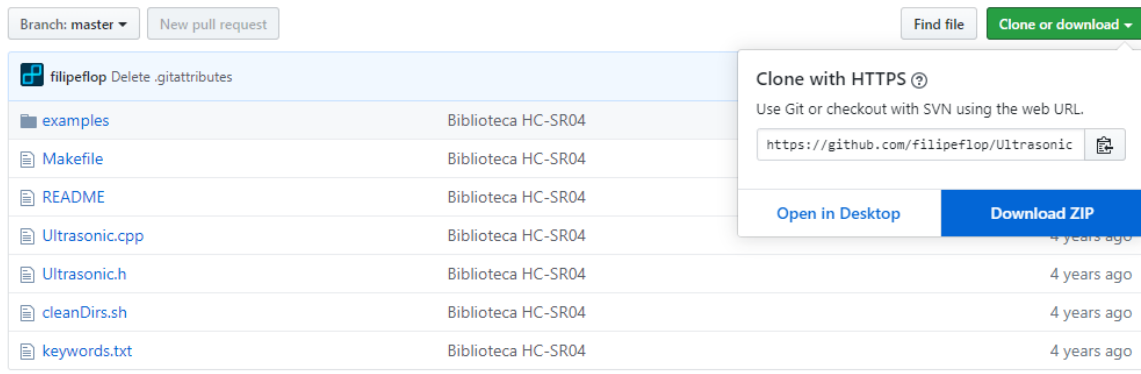
- <https://github.com/filipeflop/Ultrasonic>, ao clicar, você encontrará a pagina da FIGURA 7.

FIGURA 7 - SITE PARA DOWNLOAD

Branch: master ▾ New pull request		Find file Clone or download ▾	
filipeflop Delete .gitattributes		Latest commit 76a8a6e on 27 Jul 2015	
examples	Biblioteca HC-SR04	4 years ago	
Makefile	Biblioteca HC-SR04	4 years ago	
README	Biblioteca HC-SR04	4 years ago	
Ultrasonic.cpp	Biblioteca HC-SR04	4 years ago	
Ultrasonic.h	Biblioteca HC-SR04	4 years ago	
cleanDirs.sh	Biblioteca HC-SR04	4 years ago	
keywords.txt	Biblioteca HC-SR04	4 years ago	

Ao acessar o link, no canto superior direito, em verde, é possível baixar a biblioteca, assim como mostra a FIGURA 8

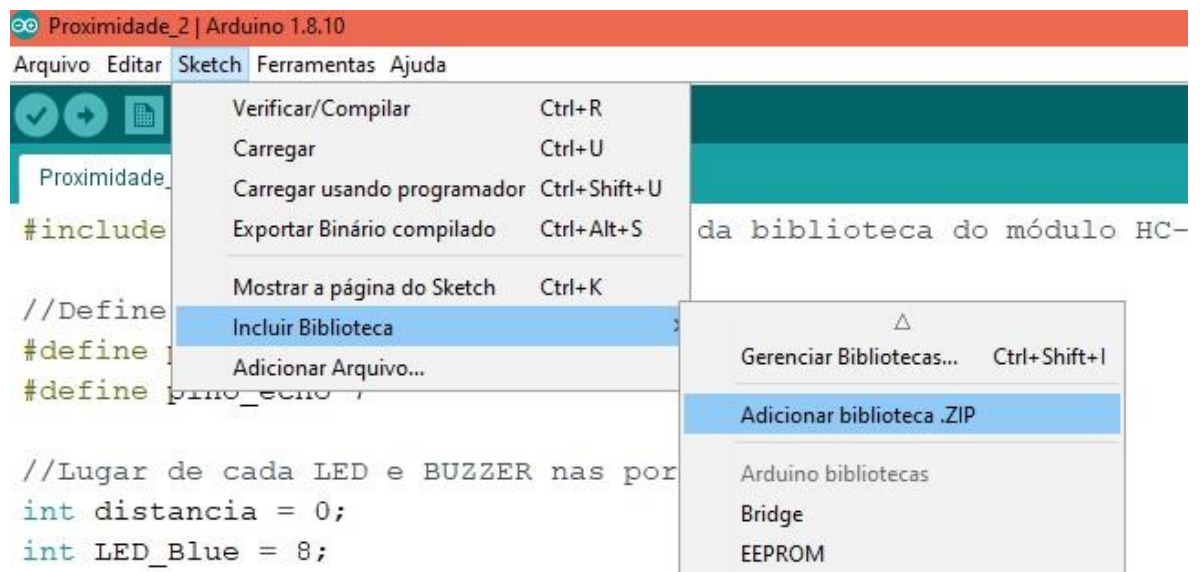
FIGURA 8 - DOWNLOAD DA BIBLIOTECA



FONTE: FILIPEFLOP

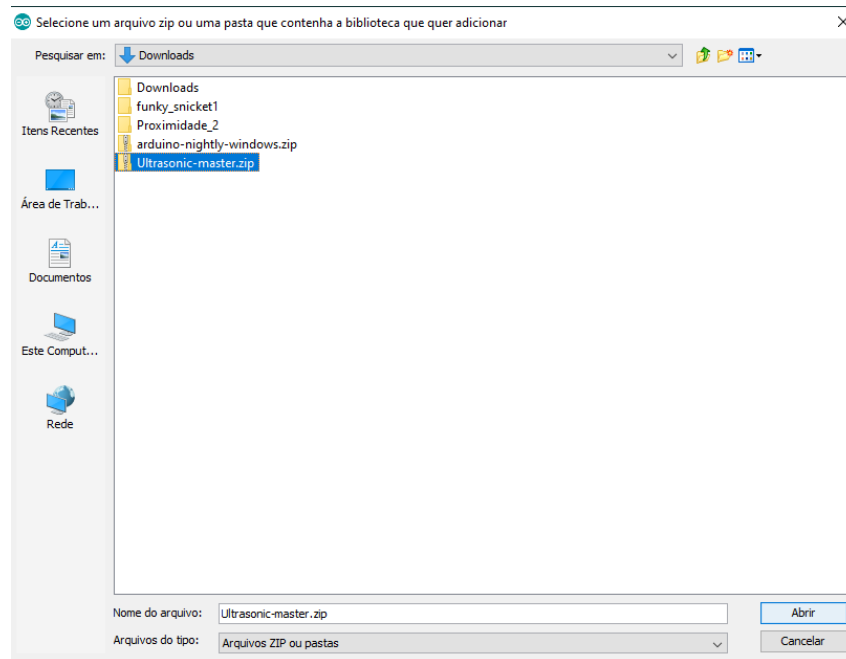
Após o download, abra a IDE do Arduino e busque na barra de ferramentas a opção “Sketch”, depois a opção “Incluir Biblioteca” aparecerá, e na nesta “Adicionar biblioteca.ZIP”, como indica a FIGURA 9

FIGURA 9 - INSTALAÇÃO DA BIBLIOTECA



Ao fazer isso, a biblioteca deve ser buscada no diretório em que foi baixada, e atribua a IDE do Arduino com a opção “Abrir”, como indica a FIGURA 10.

FIGURA 10 - DIRETORIO DE ARQUIVOS



Clicando em <Adicionar biblioteca.ZIP>, na janela que aparece, selecionamos o diretório onde foi baixada a biblioteca e clicamos em “abrir”, podemos tanto incluir um arquivo compactado (.ZIP), quanto também uma pasta descompactada. Feito o procedimento a biblioteca está pronta para uso.

Feito isto, podemos falar sobre as funções utilizadas.

Ultrasonic(); ultrasonic(“Porta do pino trigger”, “Porta do pino echo”); :
Responsável por informar para o programa as portas responsáveis pelos pinos, conforme a FIGURA 11.

FIGURA 11 - ULTRASONIC

```
#include <Ultrasonic.h> //Importação da biblioteca do módulo HC-SR04.

//Define os pinos para o trigger e echo.
#define pino_trigger 6
#define pino_echo 7
```

Ultrasonic.timing(); Responsável pelo cálculo do tempo entre a emissão e recepção da onda ultrassônica, indicado na FIGURA 12.

FIGURA 12 - ULTRASONIC.TIMING

```
long microsec = ultrasonic.timing();
```

Ultrasonic.convert(ultrasonic.timing, Ultrasonic::"medida"); Responsável pela conversão do tempo de envio e recepção da onda(determinado pela função ultrasonic.timing) para centímetros ou polegadas (substituir "medida" por: CM para centímetros ou IN para polegadas), conforme a FIGURA 13.

FIGURA 13 - ULTRASONIC.CONVERT

```
distancia = ultrasonic.convert(microsec, Ultrasonic::CM);
```

2.3 O CÓDIGO

Na primeira parte do código, fora das funções básicas do Arduino, incluímos a biblioteca utilizada, e declaramos as variáveis. Outra coisa que podemos fazer é chamar a biblioteca e definir quais vão ser os pinos de *input* do sensor, assim como na FIGURA 14.

FIGURA 14 - DECLARAÇÃO DE VARIÁVEIS

```
//Define os pinos para o trigger e echo.
#define pino_trigger 6
#define pino_echo 7

//Lugar de cada LED e BUZZER nas portas do Arduino.
int distancia = 0;
int LED_Blue = 8;
int LED_Green = 9;
int LED_Yellow = 10;
int LED_Red = 11;
int Buzzer = 12;
```

Nessa etapa a função *setup*, obrigatória para o funcionamento do Arduino, é responsável por fazer a uma declaração fixa do que vai ser usado e o que precisa ser iniciado.

Declaramos então para que servirão os pinos utilizados para ligar os LED's e o Buzzer no Arduino; um pino pode ser de *input*, ou seja, receber alguma informação vinda de algum sensor conectado ao pino, ou *output*, enviar informação para o pino. A informação é enviada, ou recebida na forma de impulso elétrico de tensão de 5 volts, como o objetivo é ligar os LED's e o Buzzer, os pinos utilizados são declarados como de *output*.

Para facilitar o entendimento das informações enviadas pelo sensor, também iniciamos a função *serial*, que imprime um resultado, este que é definido na programação mais a frente, no monitor serial da IDE do Arduino, definimos a taxa de transmissão como de 9600, assim como indica a FIGURA 15.

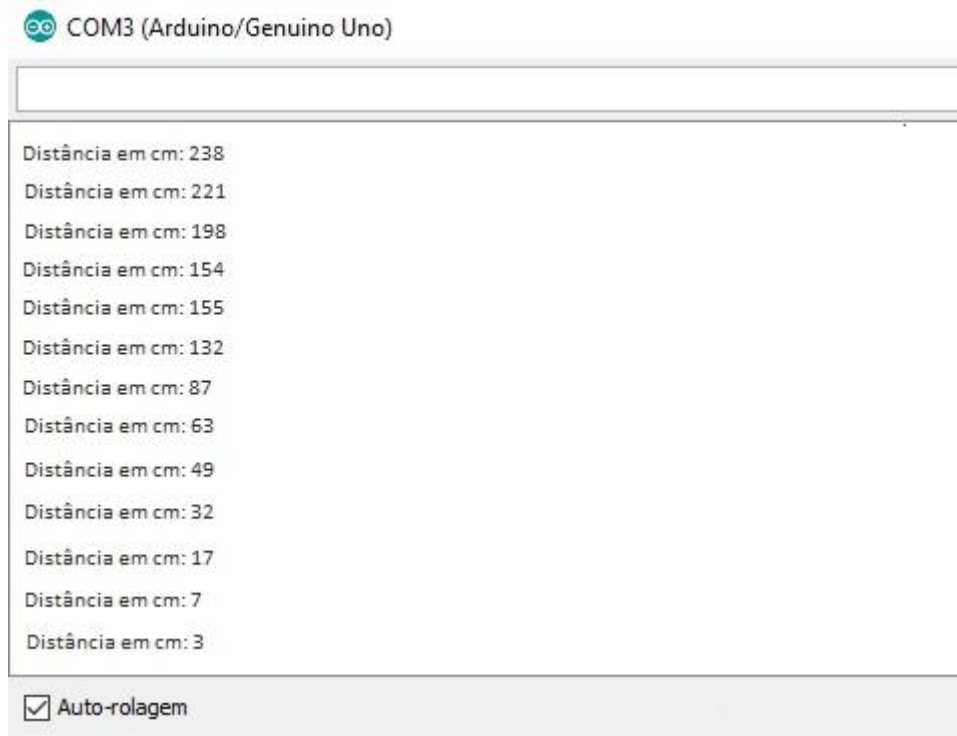
FIGURA 15 - DEFININDO A SERIAL

```
void setup()
{
  Serial.begin(9600); //Monitor para verificar a distância.
  //Função que configura a saída dos pinos(pinMode).
  pinMode(LED_Blue, OUTPUT);
  pinMode(LED_Green, OUTPUT);
  pinMode(LED_Yellow, OUTPUT);
  pinMode(LED_Red, OUTPUT);
  pinMode(Buzzer, OUTPUT);
}
```

A próxima função obrigatória funciona como um laço de repetição automático e sem fim, enquanto o Arduino estiver ligado, tudo que estiver dentro da função se repete.

Além da utilização das funções já explicadas e existentes na biblioteca do sensor, nessa etapa também utilizamos o *monitor serial* para fazer o acompanhamento da distância que é fornecida pelo sensor, essa informação é impressa constantemente e já convertida para centímetros, conforme FIGURA 16.

FIGURA 16 - MONITOR SERIAL



Após essa conversão, utilizamos condicionais começando da seguinte forma: Se a distância detectada pelo sensor for superior a “x” cm, o Arduino manda a informação para todos os pinos se apagarem. A função *digitalwrite()* é responsável por mandar a informação para o pino digital sendo: 0 para 0 volts ou 1 para 5 volts. Esta que pode ser verificada na FIGURA 17.

FIGURA 17 - FUNÇÃO DIGITAL WRITE

```
//Le as informações do sensor, em centímetros
int distancia;
long microsec = ultrasonic.timing();
distancia = ultrasonic.convert(microsec, Ultrasonic::CM);
Serial.print("Distancia em cm: ");
Serial.println(distancia);
```

A partir do momento que a distância informada é menor que “x” cm, o Arduino manda a informação para os LED’s correspondentes e o Buzzer; estes que permanecem acesos, ou apitando, respectivamente de acordo com um tempo determinado, explicitado com a função *delay*.

Esse tempo é medido em milissegundos, após a conclusão, e caso a condição de distância ainda seja satisfeita, os LED’s se apagam e o Buzzer para de

apitar, e permanecem dessa forma pela mesma quantidade de tempo e novamente se acendem após concluído o período. Este procedimento está indicado nas FIGURAS 18 e 19.

FIGURA 18 – APLICAÇÃO DA FUNÇÃO I

```
// 0 é quando o LED/Buzzer está desligado, e 1 para quando o LED/Buzzer está ligado.
if ( distancia > 200 )
{
//Função liga/desliga (digitalWrite) própria do arduino.
digitalWrite(LED_Blue, 0);
digitalWrite(LED_Green, 0);
digitalWrite(LED_Yellow, 0);
digitalWrite(LED_Red, 0);
digitalWrite(Buzzer, 0);
```

FIGURA 19 - APLICAÇÃO DA FUNÇÃO II

```
else if (distancia > 150)
{
digitalWrite(LED_Blue, 1);
digitalWrite(LED_Green, 0);
digitalWrite(LED_Yellow, 0);
digitalWrite(LED_Red, 0);
digitalWrite(Buzzer, 1);
delay (500); //500 microsegundos que permanece desligado.
digitalWrite(LED_Blue, 0);
digitalWrite(LED_Green, 0);
digitalWrite(LED_Yellow, 0);
digitalWrite(LED_Red, 0);
digitalWrite(Buzzer, 0);
delay (500);
}
else if ( distancia > 120 )
```

Conforme o objeto se aproxima, o *delay* vai diminuindo, fazendo os LED's e o Buzzer oscilarem seu funcionamento mais rapidamente. Ao fim, se o objeto se aproxima demais, todos os LED's e o Buzzer ligam e desligam em uma frequência, indicando ao usuário essa proximidade, conforme a FIGURA 20.

FIGURA 20 - APLICAÇÃO DA FUNÇÃO III

```
,  
else if ( distancia <= 8 )  
{  
digitalWrite(LED_Blue, 1);  
digitalWrite(LED_Green, 1);  
digitalWrite(LED_Yellow, 1);  
digitalWrite(LED_Red, 1);  
digitalWrite(Buzzer, 1);  
delay (50);  
digitalWrite(LED_Blue, 0);  
digitalWrite(LED_Green, 0);  
digitalWrite(LED_Yellow, 0);  
digitalWrite(LED_Red, 0);  
digitalWrite(Buzzer, 0);  
delay (50);  
}  
} //Fechamento da função loop
```

3 CÓDIGO EM PYTHON

Devido ao fato de que as bibliotecas em C++, que possibilitam a manipulação do Arduino, não serem compatíveis com as bibliotecas utilizadas na linguagem Python, para melhor visualização e compreensão, segue o código figurativo na linguagem Python.

A primeira ação do programa é perguntar ao usuário se ele deseja ligar, ou não o sensor. Se o usuário ligar, o programa seguirá pedindo as demais informações. Caso contrário, o programa insistirá que o usuário ligue o sensor, conforme a FIGURA 21.

FIGURA 21 - INÍCIO DO CÓDIGO

```
58  #INÍCIO DO CÓDIGO PARA O USUÁRIO
59
60  #Usuário informa se deseja ligar, ou não o sensor.
61  ard = input("(L)iga ou (D)esliga o Sensor:").upper()
62
63  #Enquanto o usuário informar "D", o sensor permanecerá desligado.
64  while ard != "L":
65      print("Sensor Desligado, ligue o Sensor.")
66      ard = input("Liga ou Desliga o Sensor:").upper()
```

Como no C++ a função `pinMode()` configura automaticamente as portas de entrada e saída, desde que indicadas no código, não é necessário pedir manualmente ao usuário este comando. Porém, no Python, ao ligar o sensor, o programa solicitará a configuração desejada, a dos LED's ou a do sensor, esta que será de saída de dados(*output*), ou de entrada de dados(*input*), essa foi exemplificada na FIGURA 22.

FIGURA 22 - CONFIGURAÇÃO NA PINMODE

```

1  #Leitura de distância em Arduinos
2
3  #Função responsável por configurar as saídas de luzes dos LED's
4  #E definir o emissor/captor de sinais no arduino.
5  def pinMode(parametro):
6      if parametro == "LED":
7          status = "Todos configurados como Output"
8      else:
9          status = "É Output/Input, Pino_echo é Entrada e Pino_trigger é Saída"
10     return status

```

Caso o usuário solicite algo diferente dos parâmetros disponíveis, o código retorna a mensagem “Parâmetro Inexistente”, e informará a mesma mensagem até atender um dos dois parâmetros, conforme indica a FIGURA 23.

FIGURA 23 - PARÂMETRO INEXISTENTE

```

68  #Quando o usuário informar que deseja ligar o sensor, o mesmo irá ligar.
69  if ard == "L":
70      print("Sensor ligado")
71      #Configure os LED's ou o sensor apenas uma vez
72      parametro = input("Deseja configurar LED ou Sensor?").upper()
73      while parametro != "LED" and parametro != "SENSOR":
74          #Caso algo diferente de LED e SENSOR seja digitado
75          print("Parâmetro Inexistente")
76          parametro = input("Deseja configurar LED ou Sensor?").upper()
77      #O while acima impossibilita que o parâmetro seja diferente de "LED" e "SENSOR".
78      #Se o parâmetro for "LED" o programa pedirá a próxima configuração, que só poderá ser SENSOR

```

Se o usuário tentar configurar mais de uma vez o mesmo parâmetro, o programa indicará a seguinte mensagem “LED já configurado!”, ou “SENSOR já configurado!”, pois não é possível configurar mais de uma vez a mesma entrada, assim como a FIGURA 24.

FIGURA 24 - PARÂMETRO DUPLICADO

```

81 #Caso contrário, o programa pedirá novamente o parâmetro.
82 if parametro == "LED":
83     print(pinMode(parametro))
84     parametro = input("Deseja configurar LED ou Sensor?").upper()
85     while parametro != "SENSOR":
86         if parametro == "LED":
87             print("LED já configurado!")
88         else:
89             print("Parametro Inexistente")
90             parametro = input("Deseja configurar LED ou Sensor?").upper()
91     print(pinMode(parametro))
92 #Se o parâmetro for "SENSOR" o programa pedirá a próxima configuração, que só poderá ser LED
93 #Caso contrário, o programa pedirá novamente o parâmetro.
94 else:
95     print(pinMode(parametro))
96     parametro = input("Deseja configurar LED ou Sensor?").upper()
97     while parametro != "LED":
98         if parametro == "SENSOR":
99             print("SENSOR já configurado!")
100        else:
101            print("Parametro Inexistente")
102            parametro = input("Deseja configurar LED ou Sensor?").upper()
103    print(pinMode(parametro))

```

Como já dito anteriormente as bibliotecas das linguagens utilizadas não são compatíveis, sendo assim, os *inputs* de distância simularão o sensor, informando a distância do objeto, como indicado na FIGURA 25.

FIGURA 25 - SOLICITAÇÃO DE DISTÂNCIA

```

105 #Estando os LED's e o SENSOR configurados, o usuário informará a distância
106 distancia = int(input("Digite sua distância em Cm do Sensor:"))

```

A distância será enviada para a função `digital_write()`, caso seja maior que 0, conforme FIGURA 26.

FIGURA 26 - SOLICITAÇÃO DA EXECUÇÃO

```
108      #Se a distancia for maior que 0, o programa fara os comandos enviados a função digital_write().
109      while distancia > 0:
110          status_LED, status_Buzzer = digital_write(distancia)
111          if distancia <= 8 and distancia > 0:
112              for i in range(len(status_LED)):
113                  print(status_LED[i])
114          else:
115              print(status_LED)
116              print(status_Buzzer)
117
118      distancia = int(input("Digite sua distância em Cm do Sensor:"))
```

Assim como na imagem acima, ao chamar a função, ela então executará os passos descritos na FIGURA 27, se estiverem menores que 200 cm e maiores que 50 cm, e apresentados na FIGURA 28, caso estejam menores que 50 e maiores que 0.

FIGURA 27 - EXECUÇÃO DA FUNÇÃO DIGITAL WRITE I

```

12  #Função responsável por analisar a distância do objeto ao sensor
13  # Acender os LED's e aumentar/diminuir a intensidade do som do Buzzer.
14  def digital_write(dist):
15      print("Sensor sempre é 1") #Esta sempre ligado.
16
17      #Para cada distância informada pelo usuário, uma ação será realizada.
18      #Distância sempre informada em cm.
19      if dist > 200:
20          status_LED = "Nenhum LED Ligado"
21          status_Buzzer = "Buzzer Desativado"
22
23      elif dist > 150 and dist <= 200:
24          status_LED = "LED Azul Ligado"
25          status_Buzzer = "Buzzer com 1/2 segundo"
26
27      elif dist <= 150 and dist > 120:
28          status_LED = "LED Verde Ligado"
29          status_Buzzer = "Buzzer com 1/3 segundo"
30
31      elif dist <= 120 and dist > 80:
32          status_LED = "LED Verde Ligado"
33          status_Buzzer = "Buzzer com 1/4 segundo"
34
35      elif dist <= 80 and dist > 50:
36          status_LED = "LED Amarelo Ligado"
37          status_Buzzer = "Buzzer com 1/5 segundo"

```

FIGURA 28 - EXECUÇÃO DA FUNÇÃO DIGITAL WRITE II

```

39      elif dist <= 50 and dist > 30:
40          status_LED = "LED Amarelo Ligado"
41          status_Buzzer = "Buzzer com 1/6 segundo"
42
43      elif dist <= 30 and dist > 15:
44          status_LED = "LED Vermelho Ligado"
45          status_Buzzer = "Buzzer com 1/7 segundo"
46
47      elif dist <= 15 and dist > 8:
48          status_LED = "LED Vermelho Ligado"
49          status_Buzzer = "Buzzer com 1/10 segundo"
50
51      elif dist <= 8 and dist > 0:
52          status_LED = ["LED Azul Ligado", "LED Verde Ligado", "LED Amarelo Ligado", "LED Vermelho Ligado"]
53          status_Buzzer = "Buzzer com 1/20 Segundo"
54
55      # Retorna o LED que será aceso e a intensidade de som do Buzzer.
56      return status_LED, status_Buzzer

```

Ao ser informada uma distância nula, ou menor que zero, o programa vai diretamente para outro condicional, e o código encerrará assim como na FIGURA 29.

FIGURA 29 - FINAL DO CÓDIGO

```
120      #Caso a distância seja menor ou igual a 0.  
121      #O programa entende que o usuário bateu no sensor, e automaticamente encerra.  
122      if distancia <= 0:  
123          print("Você Bateu, a gente avisou!!!")  
124          #PROGRAMA ENCERRADO
```

4 MATERIAL E MÉTODOS

Agora que já sabemos como funciona o Hardware e o Software do Arduino, e como o código foi programado podemos ir para os materiais utilizados na montagem do Hardware do projeto.

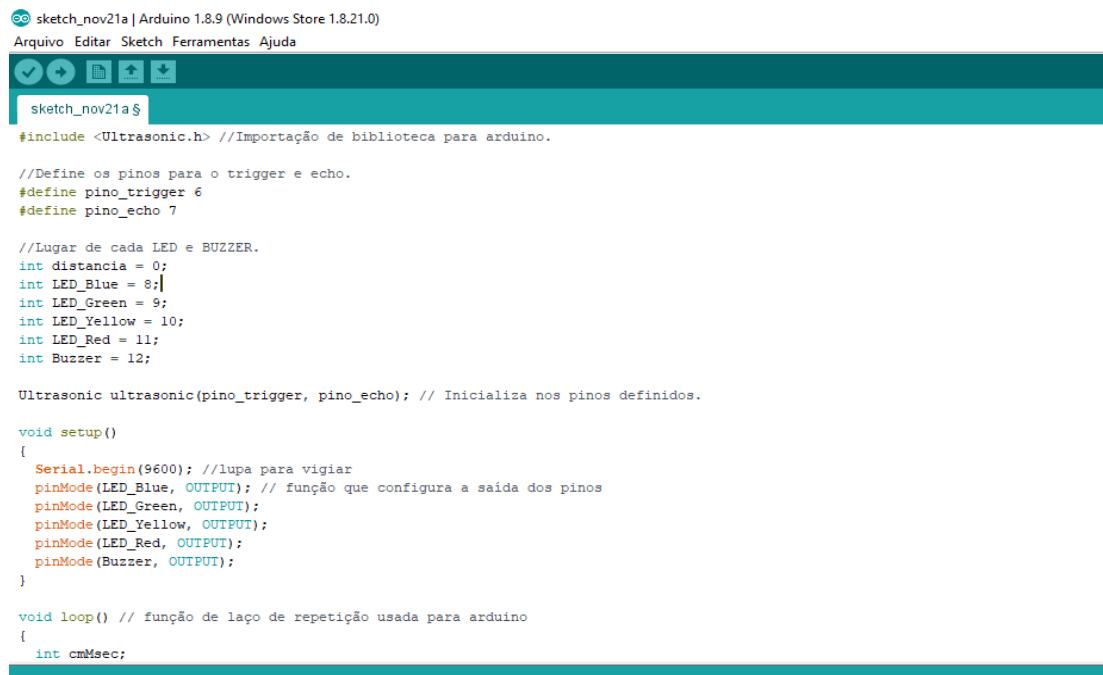
Tendo em vista que a parte do software pode ser executada em qualquer tipo de computador desde que instalada a IDE do Arduino, falaremos de como enviar esses dados para a placa.

4.1 SOFTWARE

O Arduino possui uma IDE própria, então basta instalá-la em seu computador para realizar os próximos passos.

Com a IDE Arduino já instalada, você irá programar, ou abrir o código na interface, como indica FIGURA 30.

FIGURA 30 - IDE ARDUÍNO



```
sketch_nov21a | Arduino 1.8.9 (Windows Store 1.8.21.0)
Arquivo Editar Sketch Ferramentas Ajuda

sketch_nov21a$

#include <Ultrasonic.h> //Importação de biblioteca para arduino.

//Define os pinos para o trigger e echo.
#define pino_trigger 6
#define pino_echo 7

//Lugar de cada LED e BUZZER.
int distancia = 0;
int LED_Blue = 8;
int LED_Green = 9;
int LED_Yellow = 10;
int LED_Red = 11;
int Buzzer = 12;

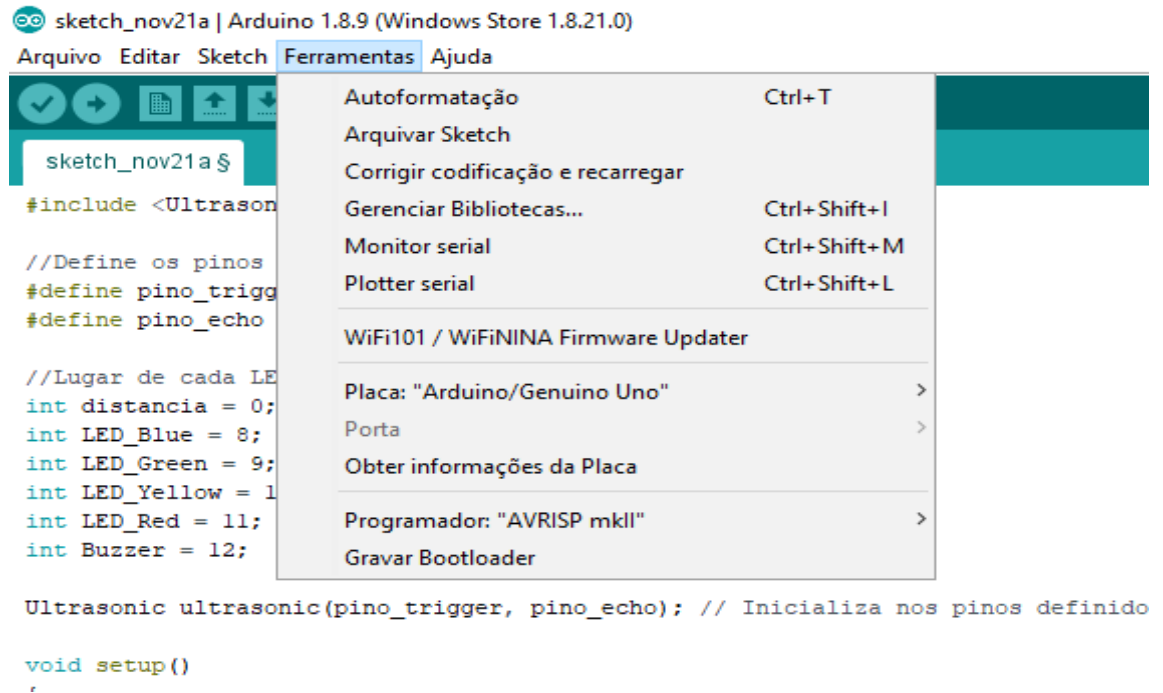
Ultrasonic ultrasonic(pino_trigger, pino_echo); // Inicializa nos pinos definidos.

void setup()
{
  Serial.begin(9600); //lupa para vigiar
  pinMode(LED_Blue, OUTPUT); // função que configura a saída dos pinos
  pinMode(LED_Green, OUTPUT);
  pinMode(LED_Yellow, OUTPUT);
  pinMode(LED_Red, OUTPUT);
  pinMode(Buzzer, OUTPUT);
}

void loop() // função de laço de repetição usada para arduino
{
  int cmMsec;
```

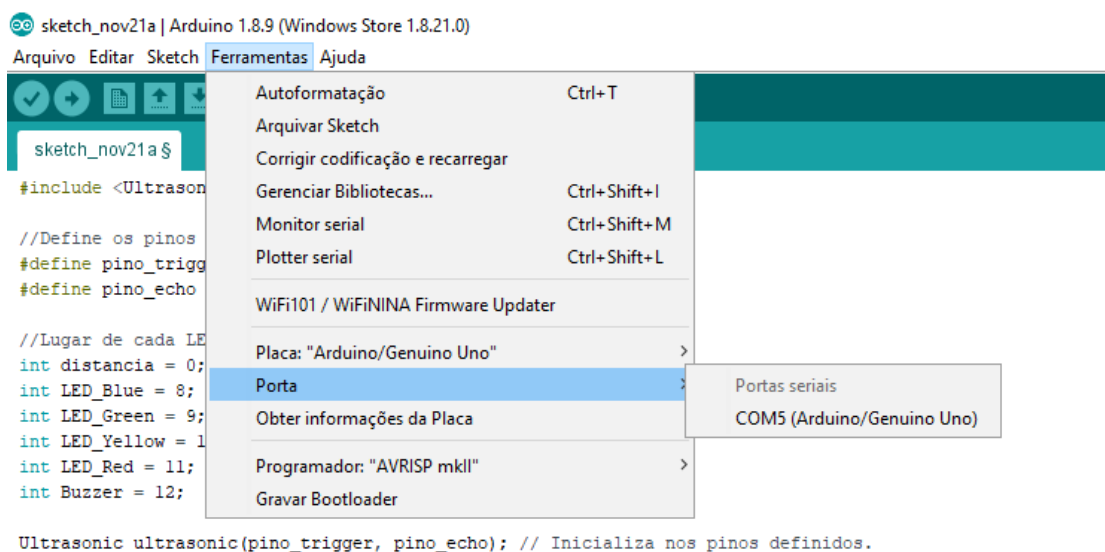
Para executar o código feito, você deve buscar na barra de ferramentas, pelo ícone “Ferramentas”, conforme a FIGURA 31.

FIGURA 31 - BARRA DE FERRAMENTAS



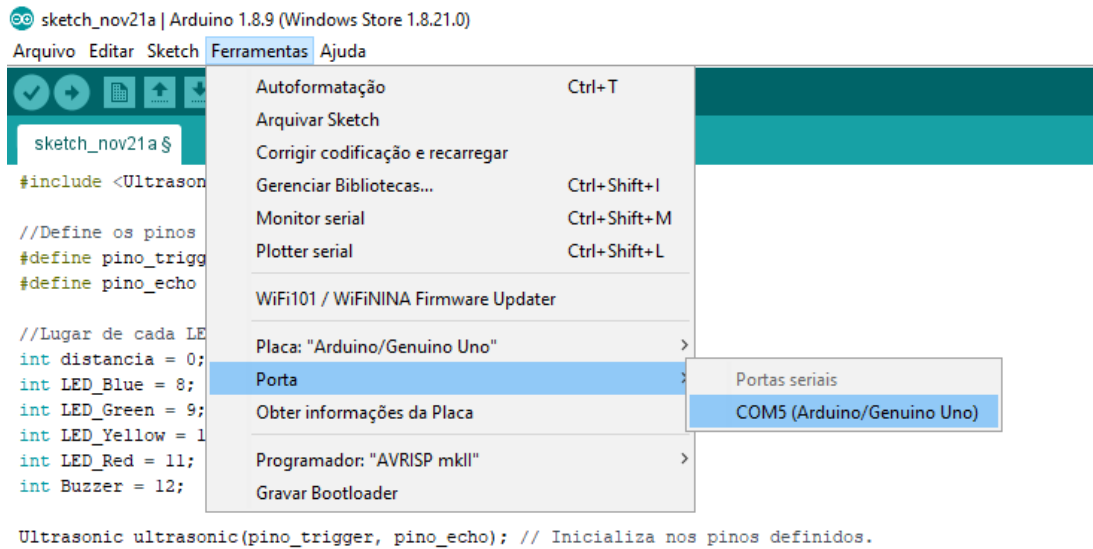
Com o Arduino conectado no computador a opção “porta” será ativada, então basta clicar nela para abrir o ícone seguinte, assim como indica a FIGURA 32.

FIGURA 32 - CONFIGURA PORTA



Na opção porta, aparecerá o nome do Arduino, no caso do Arduino utilizado no projeto “COM5”, conforme a FIGURA 33.

FIGURA 33 - ATIVAÇÃO DA PORTA



Para saber se já podemos prosseguir, basta esperar os LED's dos Microcontroladores, presentes na Placa Arduino pararem de piscar, e estabilizarem.

4.2 HARDWARE

Com o código já transferido para o Arduino, podemos desconectar o cabo USB da placa, e prosseguir para a montagem do Hardware.

Todos os materiais utilizados estão listados na tabela 1.

TABELA 1 - MATERIAIS DA MONTAGEM

Materiais Utilizados no Hardware	Qtda
Arduino UNO Rev3 + Cabo USB do tipo B	1
Protoboard	1
Sensor HC-SR04	1
LED Azul	1
LED Verde	1
LED Amarelo	1
LED Vermelho	1
Resistores de 330Ω	4
Jumper's Macho-Macho	14
Buzzer	1

4.2.1 MANUAL de montagem

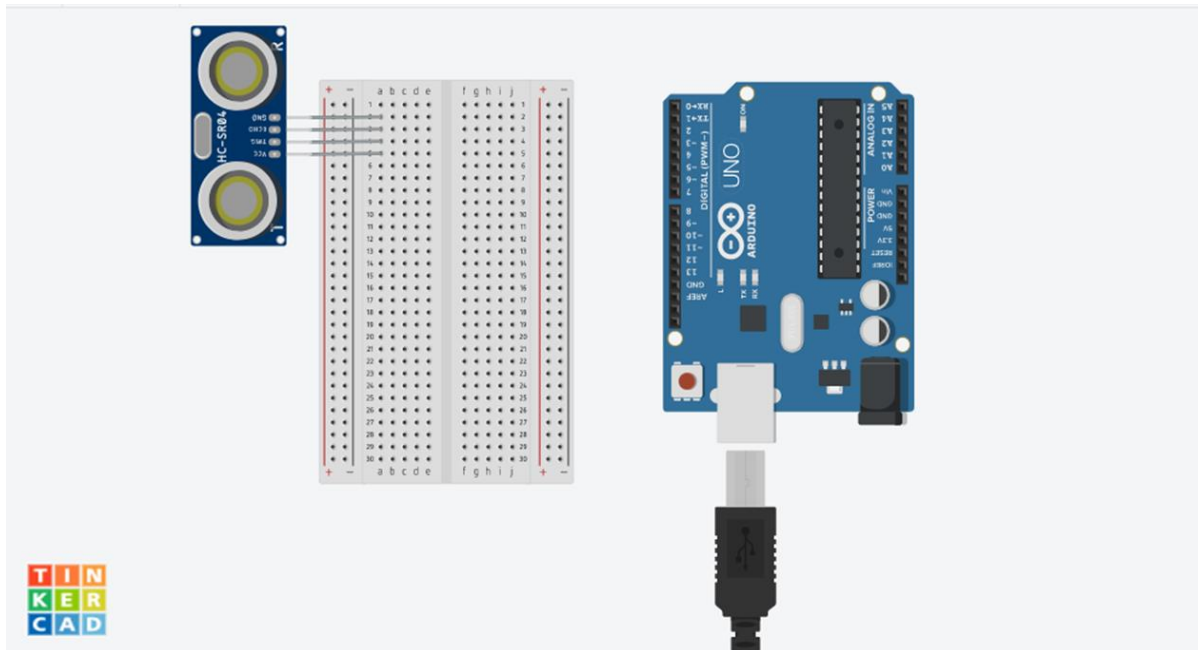
A montagem não necessariamente precisa seguir exatamente todas as entradas ou portas que usamos, fica ao seu critério seguir a risca, ou não a montagem, isso não ira afetar a execução do projeto desde que montado corretamente.

Para começar a montagem você vai precisar do Arduino, e a Protoboard, na posição positivo a esquerda e negativo a direita. A distribuição das entradas da Protoboard são denominadas da seguinte forma para este caso: Linhas na Vertical, indicada pelas letras, e colunas na horizontal, indicada por números. Assim como na posição indicada na FIGURA .

4.2.1.1 Sensor HC-SR04

Conecte o Sensor na protoboard, na linha “A”, e entradas 2,3,4,5, assim como indica a FIGURA 34.

FIGURA 34 - SENSOR

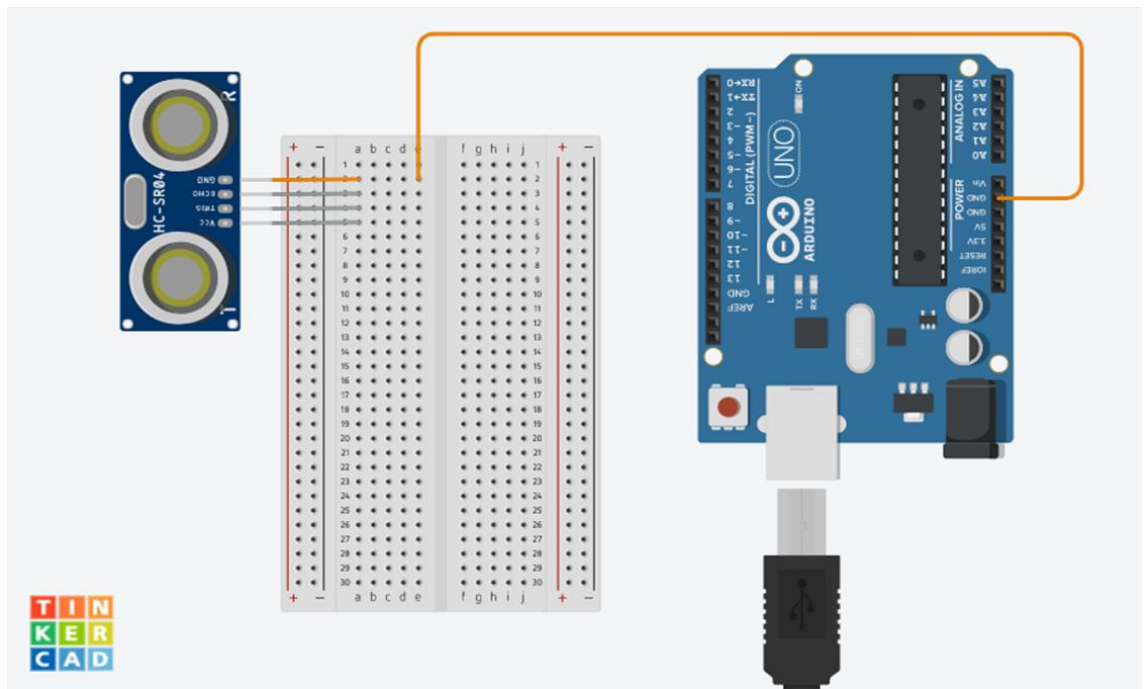


Para uma melhor visualização, os cabos dos procedimentos serão destacados em laranja.

Para o primeiro pino do sensor, o Pino GND, iremos conectar um Jumper da coluna do pino, e na linha “E” da protoboard ate uma das entradas GND do Arduino.

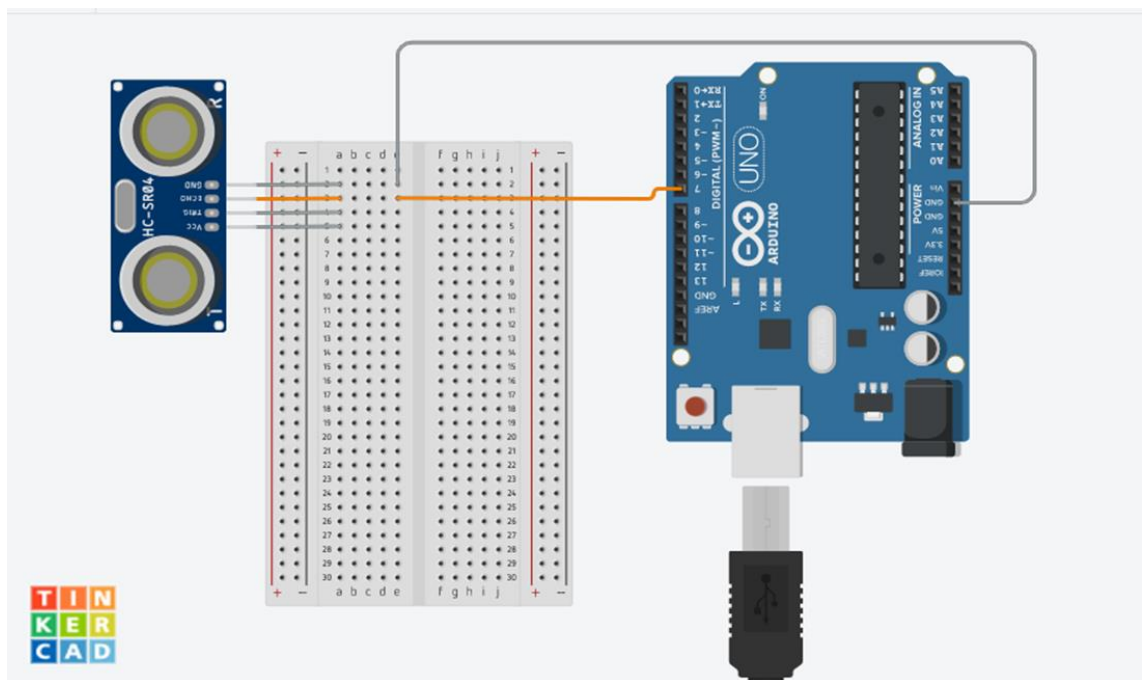
Como a UNO tem três entradas GND, fica da preferencia de quem esta montando qual das entradas irá utilizar. Para uma melhor visualização usaremos uma das entradas do lado direito, assim como indica a FIGURA 35.

FIGURA 35 - PINO GND



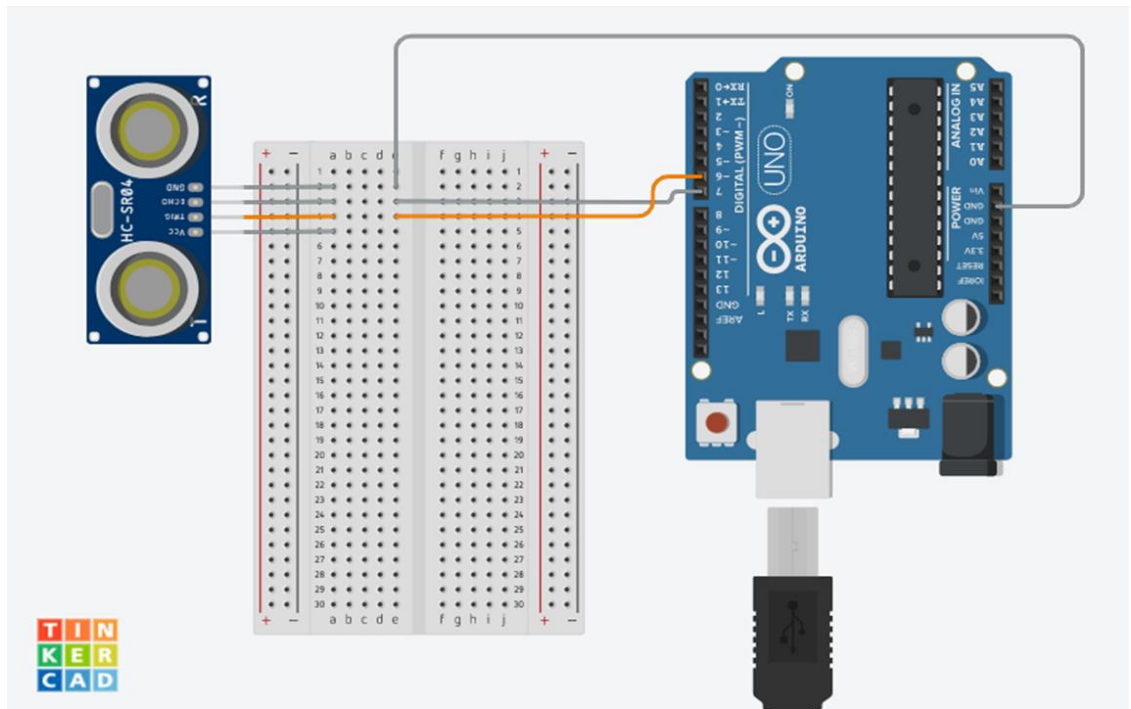
Para o segundo pino do sensor, o Pino Echo, iremos conectar um Jumper da coluna do pino, e na linha “E” da protoboard ate a porta 7 do Arduíno, pois essa foi a que estipulamos em nosso código para a entrada da leitura do sensor, conforme a FIGURA 36.

FIGURA 36 - PINO ECHO



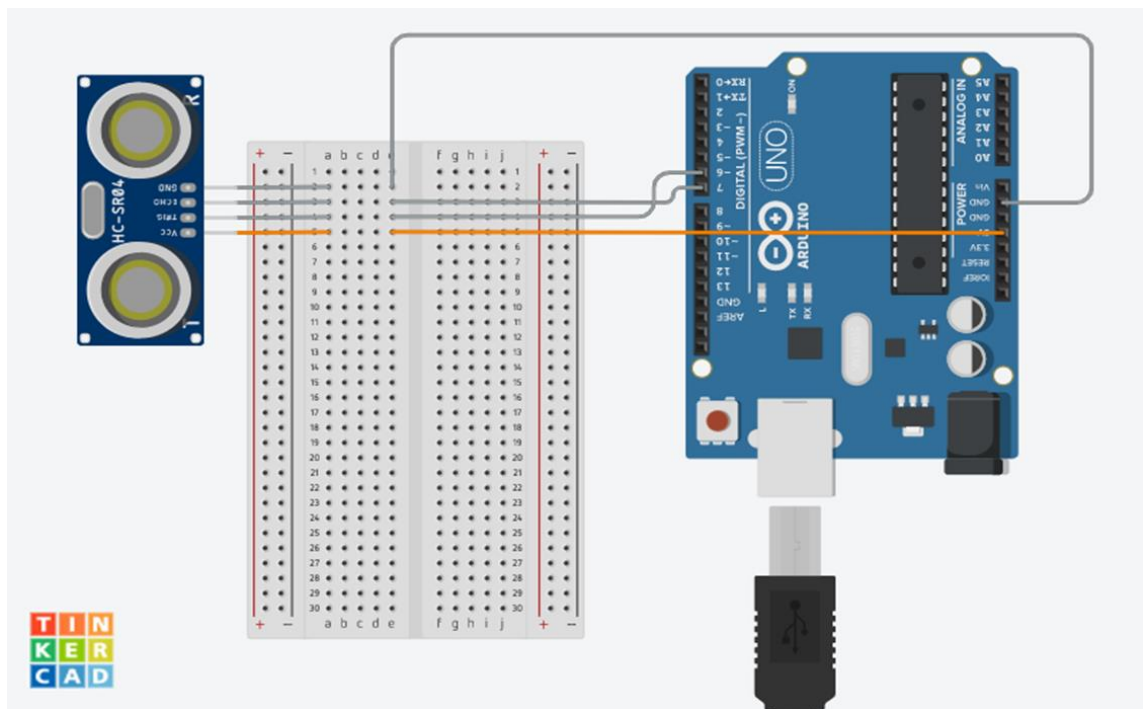
Para o terceiro pino do sensor, o Pino Trigger, iremos conectar um Jumper da coluna do pino, e na linha “E” da protoboard ate a porta 6 do Arduino, pois essa foi a que estipulamos em nosso código para a saída da leitura do sensor, como indicado na FIGURA 37.

FIGURA 37 - PINO TRIGGER



Para o quarto pino do sensor, o Pino VCC, iremos conectar um Jumper da coluna do pino, e na linha “E” da protoboard até a porta 5 Volts do Arduino, pois é ele que irá alimentar o módulo proveniente do USB, ou de uma fonte externa, conforme FIGURA 38.

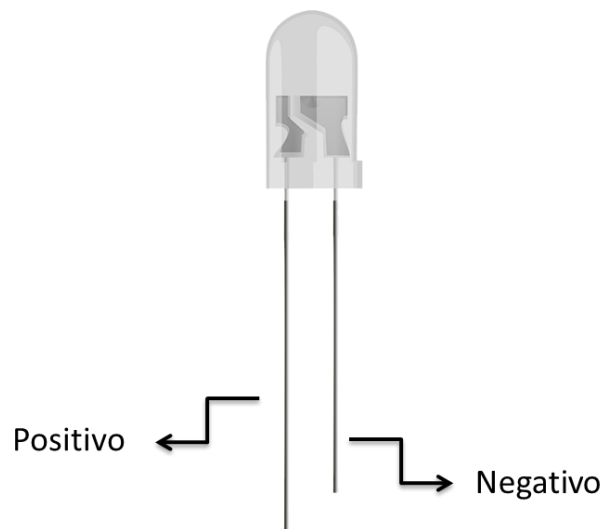
FIGURA 38 - PINO VCC



4.2.1.2 LED's

Para conectar os LED's na protoboard basta colocar a parte negativa do LED na linha negativa da protoboard, à esquerda, e a positiva na linha "B". Para saber qual é o lado negativo e o positivo do componente, confira a FIGURA 39.

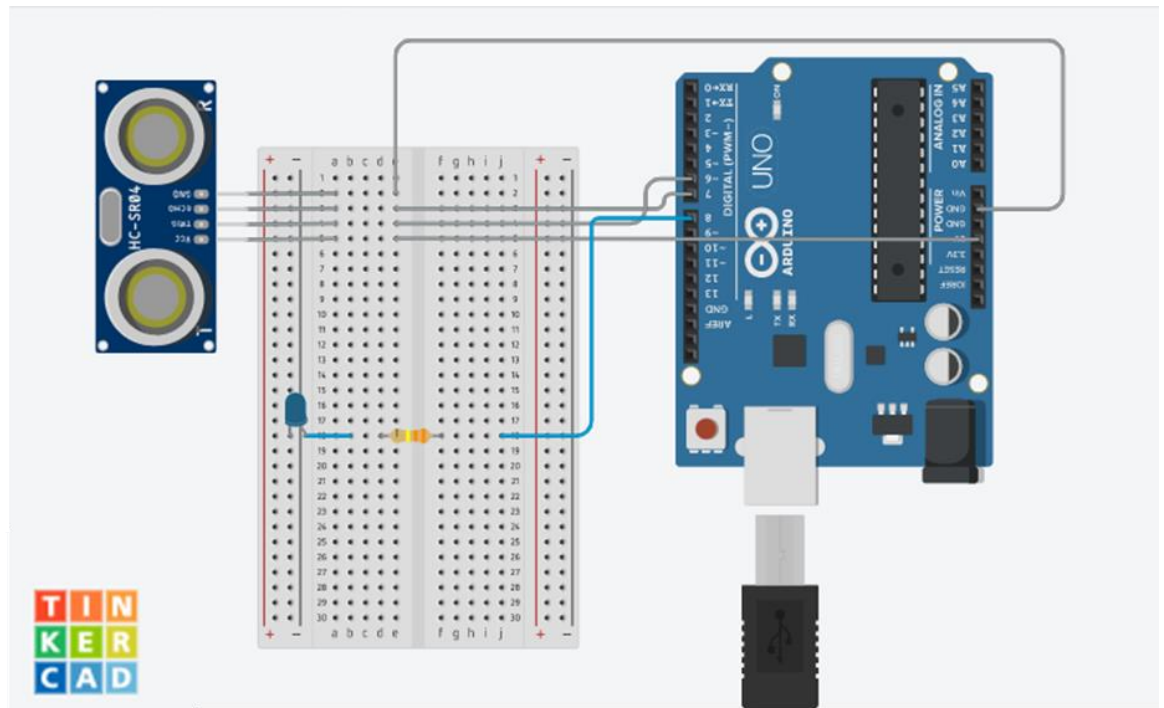
FIGURA 39 - INDICAÇÃO DE CONDUTORES LED



Na mesma coluna que for colocada a parte positiva do LED, (coluna 18), um resistor de 330Ω (Ohms) deve ser posicionado entre a linha "E" e "F", resistores são despolarizados, então não possuem lado correto para sua introdução no circuito.

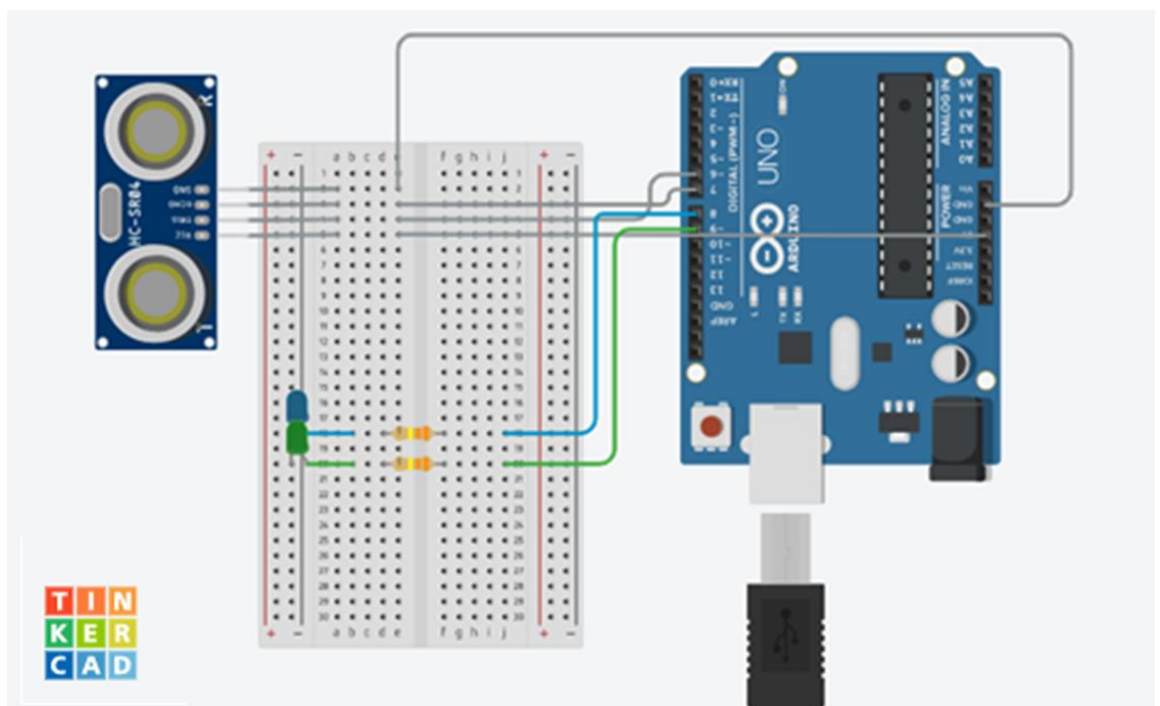
Assim que o resistor for posicionado um jumper na linha "J" liga o LED até a porta configurada no código, ou seja, na porta 8 do Arduino.

FIGURA 40 - LED AZUL



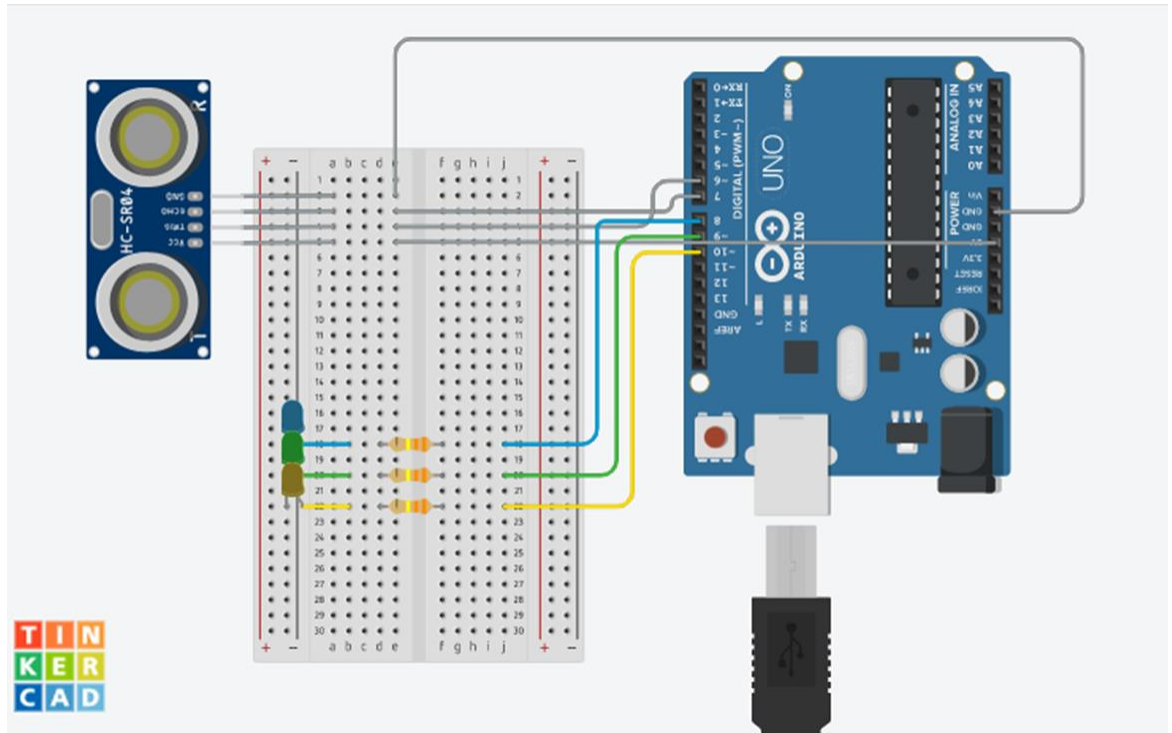
O Mesmo processo será feito para o LED verde, a única diferença é o posicionamento na coluna 20, e a entrada do Jumper na porta 9, conforme FIGURA 40.

FIGURA 41 - LED VERDE



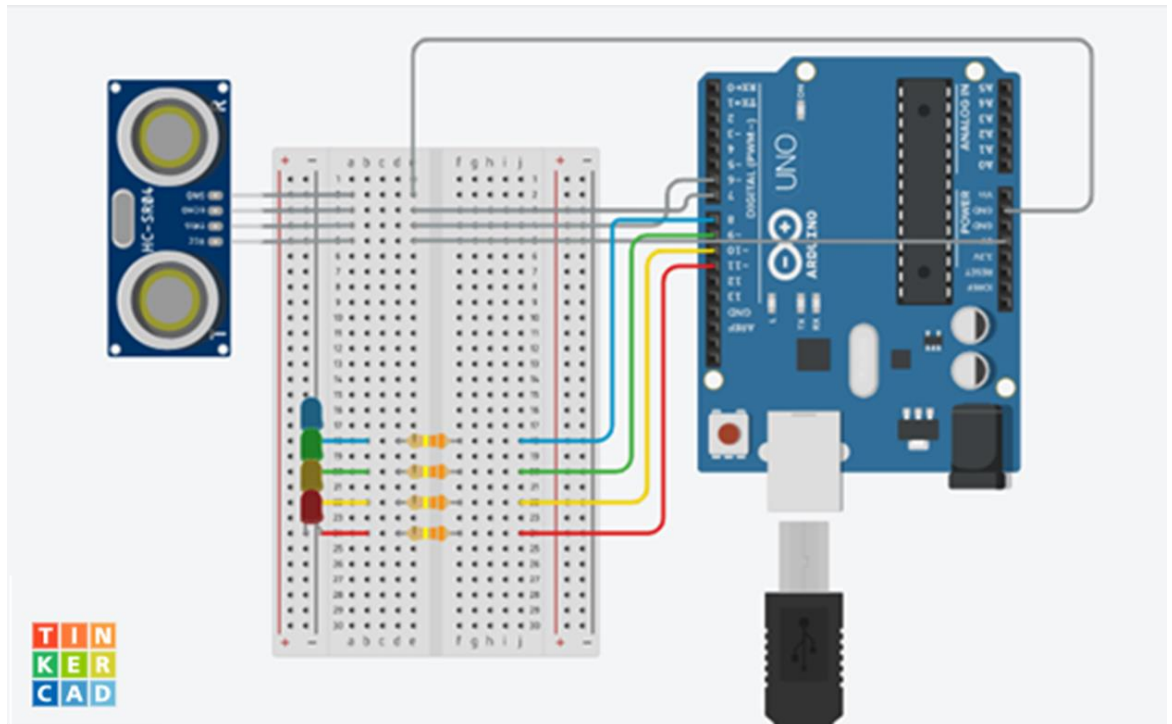
Repetindo o processo para o LED amarelo, a única diferença é o posicionamento na coluna 22, e a entrada do Jumper na porta 10, conforme a FIGURA 41.

FIGURA 42 - LED AMARELO



E por último, o processo para o LED vermelho, o posicionando na coluna 24, e a entrada do Jumper na porta 11, conforme a FIGURA 42.

FIGURA 43 - LED VERMELHO



E esse foi o procedimento para os LED's, lembrando que as portas para conexão com o Arduino são determinadas de acordo com o seu código.

4.2.1.3 BUZZER CLDZ

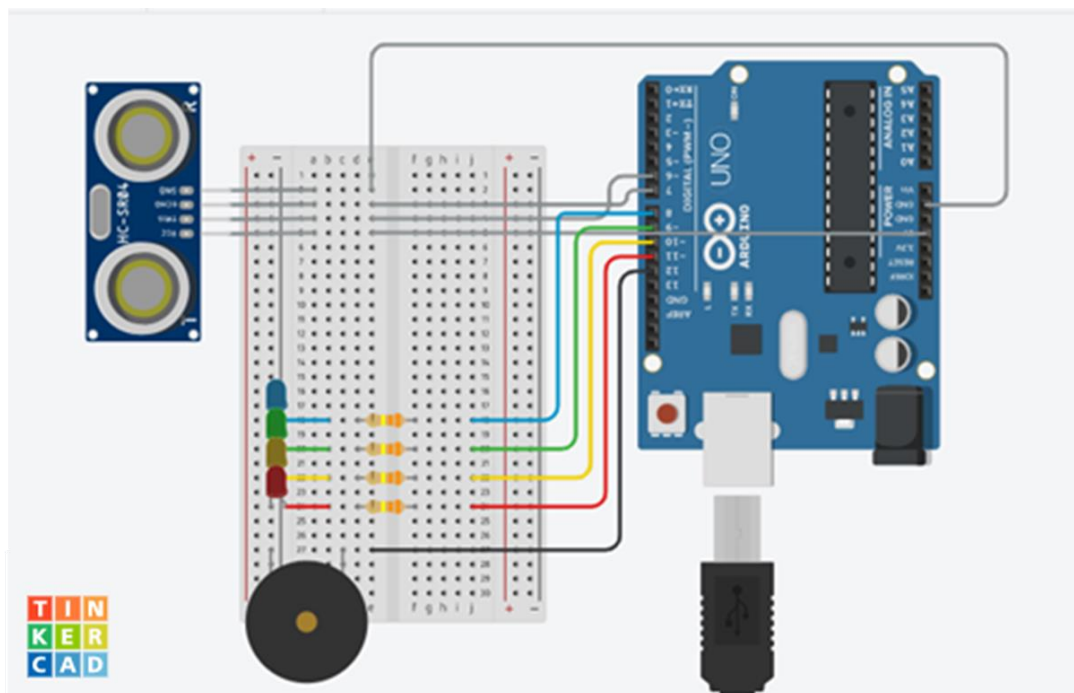
O Buzzer será conectado na protoboard na coluna 27, entre a mesma linha negativa em que posicionamos os LED's e linha "C"; o Buzzer tem uma indicação seja na parte superior, ou em alguns casos na inferior de qual é o conector positivo e qual é o negativo. É importante se atentar a esse detalhe para não queimar o componente, para não restar dúvidas, confira FIGURA 43.

FIGURA 44 - INDICAÇÃO DE CONDUTORES BUZZER



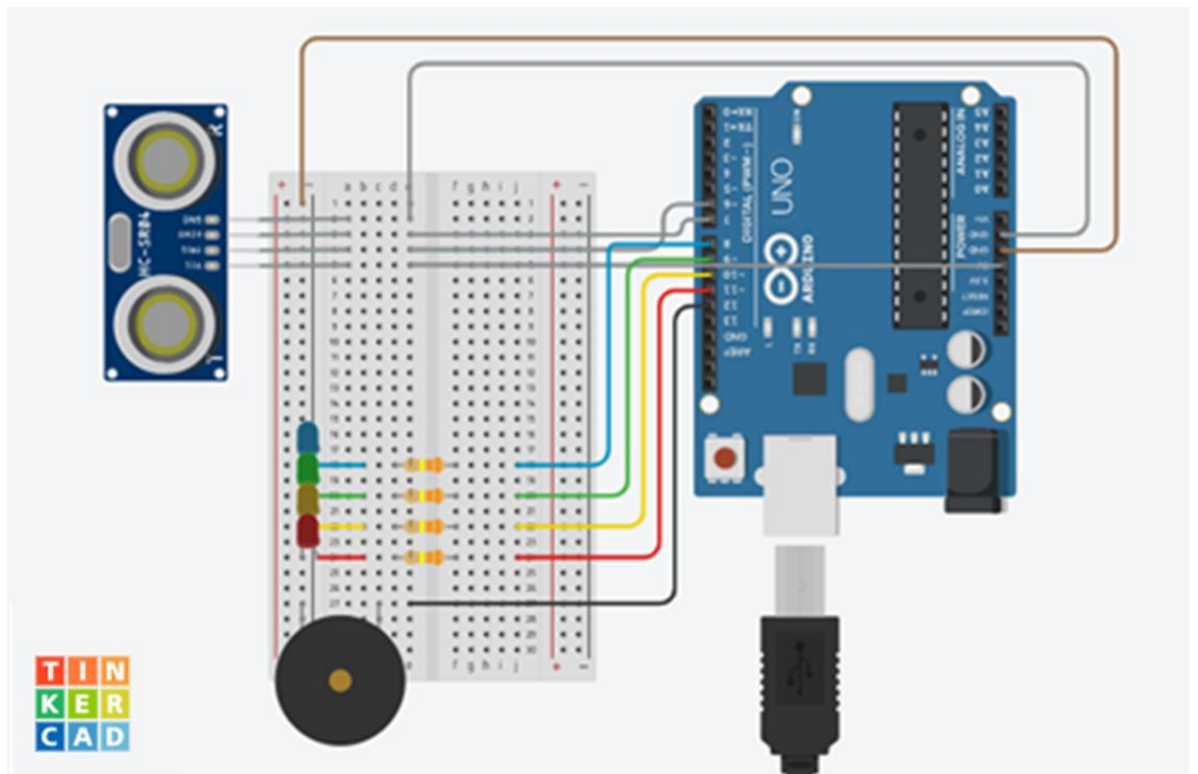
Com o Buzzer conectado, um jumper irá ligá-lo da linha "J" até a porta 12 do Arduino, confira a FIGURA 44.

FIGURA 45 - BUZZER



Assim chegamos ao último procedimento para montar a parte do Hardware do Arduino. É necessário conectar um jumper na mesma linha negativa em que os LED's e o Buzzer foram posicionados, de preferencia em uma das pontas, e liga-lo em uma das portas GND do Arduino, conforme a FIGURA 45.

FIGURA 46 - JUMPER GND



A porta GND, também conhecida como terra, ou em inglês *ground*, é responsável por fechar os circuitos, sendo assim necessária para o funcionamento do que acabamos de montar.

Com isso, finalizamos a montagem, e podemos executar o código conectando o cabo USB novamente para servir de fonte de energia.

5 APRESENTAÇÃO DOS RESULTADOS

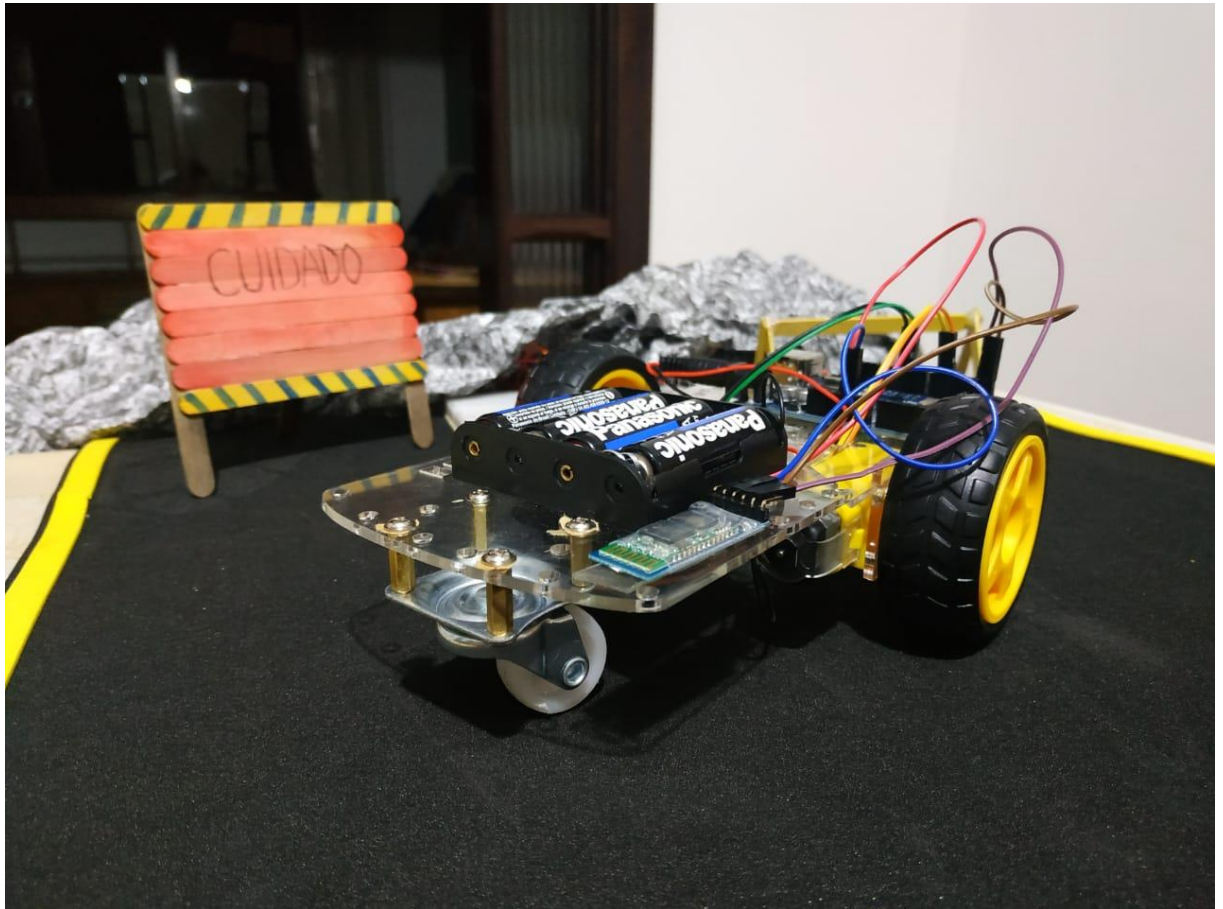
Com a finalidade de demonstrar a parte pratica, desenvolvemos uma maquete feita de Eva, esta que terá o sensor posicionado entre os pedregulhos conforme FIGURA 46.

FIGURA 47 - MAQUETE



E utilizamos também um carrinho que já está montado e programado em outro Arduíno, e através de um controle remoto vamos aproxima-lo do sensor. Este não foi desenvolvido neste projeto, mas para fins demonstrativos e de aplicação será utilizado nessa pratica, e terá seu código disponibilizado em anexo. Segue na FIGURA

FIGURA 48 - CARRINHO



Nossa maquete foi construída pensando em um sensor de perigo, para que o carrinho não bata no penhasco, mas esta é só uma das utilizações. Outras aplicabilidades que podemos usar o mesmo módulo HC-SR04 e um Arduino seriam sinaleiros, bebedouros automáticos, radares e sensores de presença, entre outras execuções.

6 CONSIDERAÇÕES FINAIS

Em suma, todos os objetivos foram alcançados, conseguimos realizar tanto a parte de Hardware, quanto a de software e aprendemos não só sobre a programação, mas a desenvolver e pensar logicamente sobre um projeto, partindo do zero, e adquirindo conhecimentos para formular muitas outras coisas.

6.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Tendo em vista que o Arduíno consegue atingir diversos públicos e diversas áreas, é possível criar muitos projetos com módulos diferentes, mas deixaremos aqui 3 ideias de projetos usando Arduínos.

- Leitura e alteração da cor de luzes com Sensor de captação de som
- Aspirador robô com sensor de proximidade.
- Sistema de automatização de luzes de uma casa via celular.

REFERÊNCIAS

<https://portal.vidadesilicio.com.br/o-que-e-arduino-e-como-funciona/>

<https://newtoncbraga.com.br/index.php/electronica/52-artigos-diversos/13263-o-basico-sobre-os-microcontroladores-parte-1-mic139>

<https://www.techtudo.com.br/noticias/noticia/2013/10/o-que-e-um-arduino-e-o-que-pode-ser-feito-com-ele.html>

<http://web.csulb.edu/~hill/ee400d/Technical%20Training%20Series/02%20Intro%20to%20Arduino.pdf>

<https://www.hostgator.com.br/blog/o-que-e-arduino/>

<https://www.embarcados.com.br/arduino-primeiros-passos/>

<https://canaltech.com.br/produtos/O-que-e-open-source/>

<https://www.newtoncbraga.com.br/index.php/como-funciona/733-como-funcionam-os-leds-art096>

<https://www.embarcados.com.br/arduino-uno/>

https://pt.wikipedia.org/wiki/Diodo_emissor_de_luz

<https://www.newtoncbraga.com.br/index.php/como-funciona/733-como-funcionam-os-leds-art096>

<https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-sensor-ultrasonico-hc-sr04/>

<https://portal.vidadesilicio.com.br/hc-sr04-sensor-ultrassonico/>

<https://store.arduino.cc/usa/arduino-uno-rev3>

<https://imasters.com.br/desenvolvimento/conheca-os-shields-e-incremente-seu-arduino-com-eles>

<https://portal.vidadesilicio.com.br/hc-sr04-sensor-ultrassonico/>