UNIVERSIDADE FEDERAL DO PARANÁ

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

EMILIO GAUDEDA JUNIOR
GRR20191670

RELATÓRIO TÉCNICO DO TRABALHO 3:

TIMERS E PWM

Trabalho apresentado no curso de Engenharia Elétrica da disciplina de Microprocessadores e microcontroladores solicitado como requisito de avaliação parcial da disciplina.

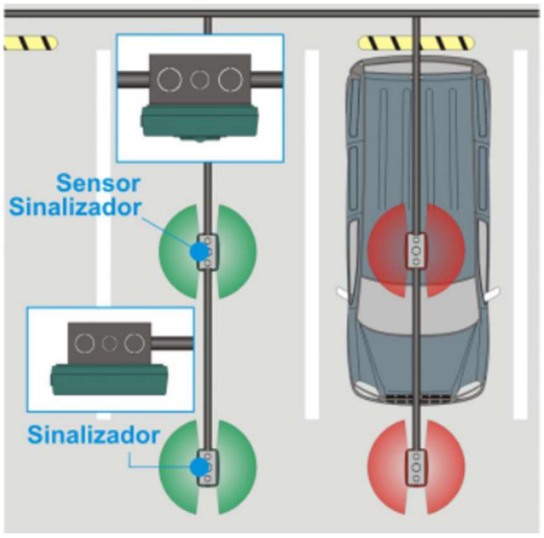
Orientador: Prof. Prof. Dr. Edson José Pacheco

CURITIBA 2022

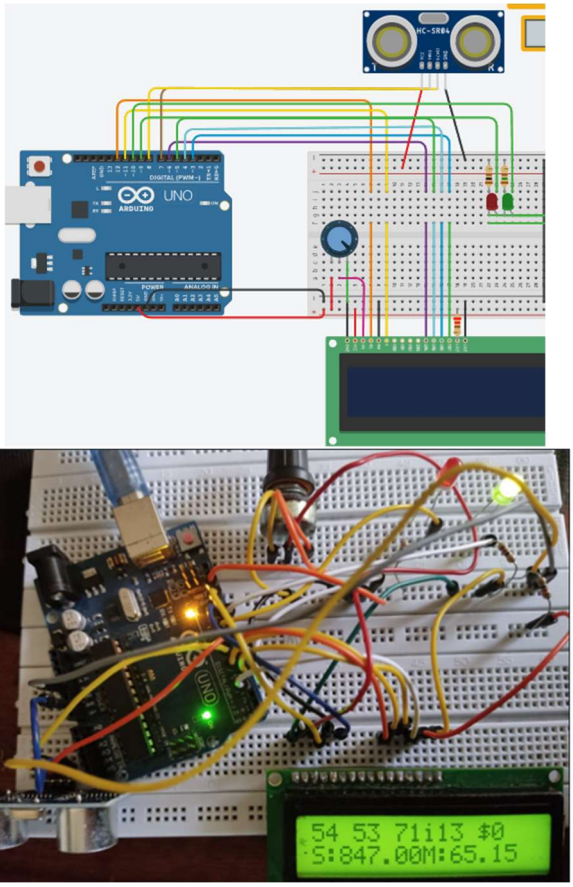
RESUMO

O objetivo do 3º trabalho é desenvolver habilidades relacionadas ao uso de sensores, *timers, pwm* e modularização de código.

Para atender esse objetivo foi desenvolvido um medidor de distância que pode ser utilizado para verificar status de vaga de garagem, proximidade do carro a objetos ou até mesmo distanciamento pessoal.



Um exemplo de aplicação prática



Topologia do projeto no TINKERCAD e ao lado o circuito real

OBJETIVOS

OBJETIVOS ESPECÍFICOS

a) Deve ser desenvolvido no Michoship Studio utilizando C/C++

Apesar do projeto ter sido simulado no Tinkercad, o desenvolvimento do código principal, a modularização das funções e as implementações do código foram feitas através do Microship Studio.

b) Deve utilizar um sensor

O sensor utilizado foi um HC-SR04. Sensor de distância ultrassônico com 4 pinos:

-VCC e GND.

-Trigger e Echo, que funcionam como um emissor e um receptor do sinal ultrassônico.

c) Deve utilizar um dispositivo de saída (Ex: LCD)

O dispositivo de saída Utilizado foi o LCD, utilizando um potenciômetro para controlar o contraste.

c) Deve aplicar a teoria de PWM

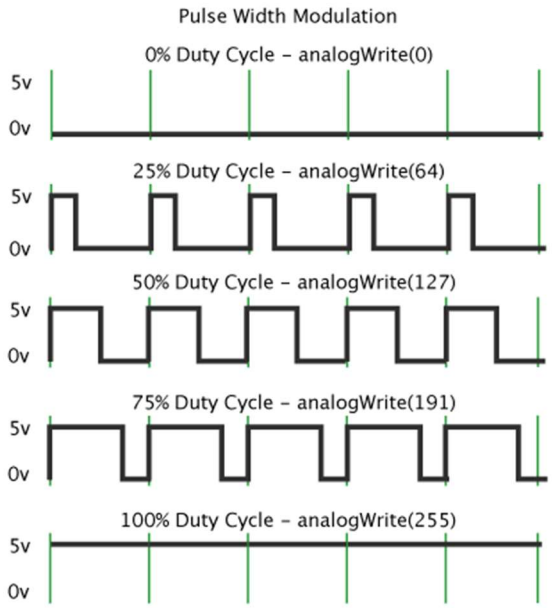


Figura 1 Teoria PWM (Pulso com Modulação)

A teoria de PWM foi aplicada da seguinte forma:

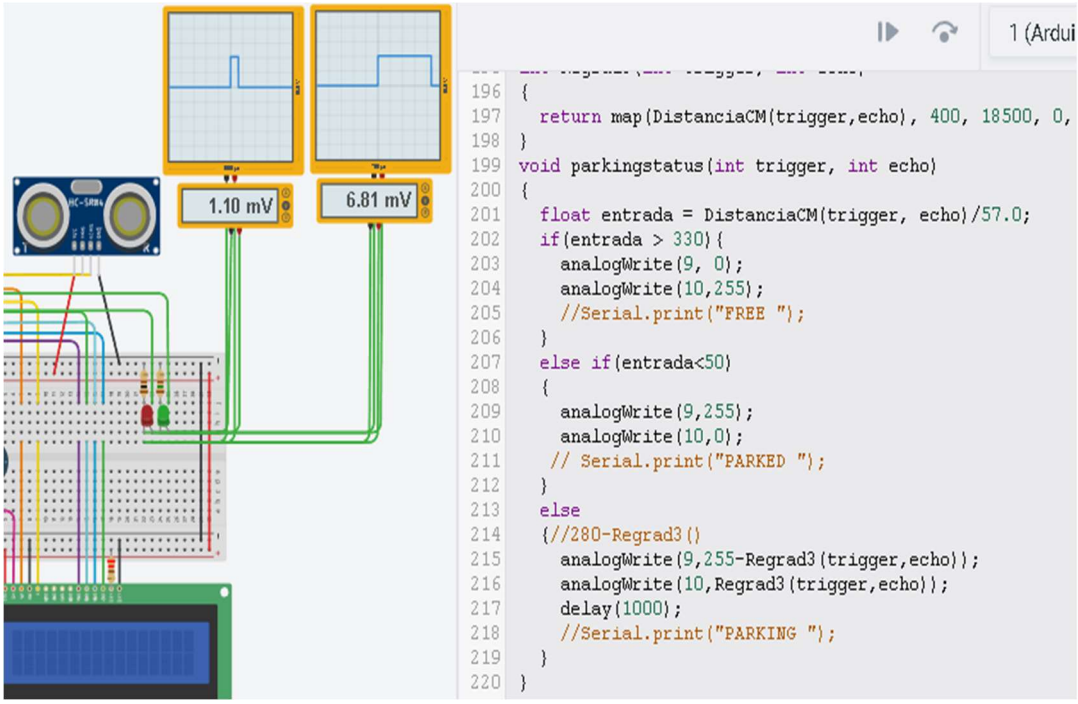


Figura 2 Teoria PWM aplicada ao projeto

Quanto mais longe o objeto estiver do sensor mais intensamente o LED verde acenderá e menos intensamente o LED vermelho. Quanto mais próximo o objeto estiver do sensor (respeitando a distancia mínima para referência) mais intensamente o LED vermelho irá ficar aceso. Essa lógica ficou registrada na função da imagem acima. Também é possível observar o Pulso por modulação (PWM) em cada osciloscópio da simulação.

d) Todas as funções de processamento devem estar em um arquivo .c/.cpp e .h separado do arquivo com as funções loop e setup

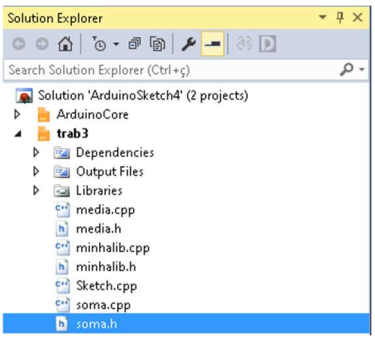


Figura 3Arquivos separados no "Solution Explorer" do Microship studio. Importante ressaltar que a função media e soma foram separadas em arquivos únicos como pedia o enunciado.

```
#include <LiquidCrystal.h>

#ifndef MINHALIB_H_
#define MINHALIB_H_

void alocacaoincial(int *pVet, int tamincial);
void printlinha2(int cont,int tam,int *pVet,LiquidCrystal &lcd);
float DistanciaCM(int triggerr, int echoo);
int Regrad3(int trigger, int echo);
void parkingstatus(int trigger, int echo);
float soma(int *pVet,int tam,int cont);
void printlinha1(int contsobrePos, int cont,int tam,int *pVet,LiquidCrystal &lcd);
int EntradaEsobrePos(int cont,int tam,int *pVet,int trigger, int echo, int contSobrepos);
float media(int cont,int *pVet,int tam);


#endif /* MINHALIB H  */
```

Figura 4 Declaração das funções na biblioteca “minhalib.h”

```
#include "minhalib.h"
#include <Arduino.h>
#include <LiquidCrystal.h>

float DistanciaCM(int triggerr, int echoo)...
int Regrad3(int trigger, int echo)...
void parkingstatus(int trigger, int echo)...
float soma(int *pVet,int tam,int cont)...
void printlinha1(int contsobrePos, int cont,int tam,int *pVet,LiquidCrystal &lcd)...
int EntradaEsobrePos(int cont,int tam,int *pVet,int trigger, int echo, int contSobrepos)...
float media(int cont,int *pVet,int tam)...
void printlinha2(int cont,int tam,int *pVet,LiquidCrystal &lcd)...
```

Figura 5 Definição das funções em “minhalib.cpp” -Vale ressaltar que nenhuma variável global foi declarada no arquivo “minhalib.cpp” foi apenas utilizada a passagem de variáveis por referência

.

```
#include <TimerOne.h>
#include <known_16bit_timers.h>
#include <LiquidCrystal.h>
#include "minhalib.h"

const int trigger= 7;
const int echo =  8;
int *pVet = NULL;
int i=0 ;
int contSobrepos=0 ;
LiquidCrystal lcd(12, 11, 6, 5, 4,3);
int tam = random(30,50);


void interrupt()
{
    lcd.clear();
    lcd.setCursor(0, 0);
    parkingstatus(trigger,echo);
    contSobrepos = EntradaEsobrePos(i,tam,pVet,trigger,echo,contSobrepos);
    i++;
    printlinha1(contSobrepos,i,tam,pVet,lcd);
    printlinha2(i,tam,pVet,lcd);
    lcd.print(" ");
```

Figura 6 Declaração da função interrupt que será chamada pela função Timer1.attachInterrupt(interrupt);

```
void setup()
{
    Timer1.initialize(1000000);
    Timer1.attachInterrupt(interrupt);
    pinMode(trigger,OUTPUT);
    pinMode(echo,INPUT);
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
    lcd.begin(16, 2);
    lcd.setCursor(0,0);
    lcd.print("Emilio Gaudeda Junior");
    lcd.setCursor(0,1);
    lcd.print("GRR20191670");
    pVet = (int*) malloc(tam);
}
void loop()
{
}
```

Figura 7 Figura 5 Definição de pinos de saída e inicialização do LCD junto com a alocação inicial de tamanho aleatório (30,50), visto que “tam” foi definido como: int tam = random(30,50);

DESENVOLVIMENTO

Materiais

Name	Quantity	Component
U2	1	Arduino Uno R3
U3	1	LCD 16 x 2
Rpot1	1	250 kΩ Potentiometer
R1	1	220 Ω Resistor
DIST1	1	Ultrasonic Distance Sensor
D1	1	Red LED
R3	1	100 Ω Resistor
D2	1	Green LED
R2	1	150 Ω Resistor

EXPLICAÇÃO DE CADA FUNÇÃO

DECLARAÇÃO DE VARIÁVEIS GLOBAIS NO SKETCH.CPP

```
const int trigger= 7;
const int echo =  8;
int *pVet = NULL;
int i=0 ;
int contSobrepos=0 ;
LiquidCrystal lcd(12, 11, 6, 5, 4,3);
int tam = random(30,50);
int contwait=0;
```

FUNÇÃO A SER CHAMADA PELA INTERRUPÇÃO POR TIMER

```
void interrupt()
{
	lcd.clear();
	lcd.setCursor(0, 0);
	parkingstatus(trigger,echo);
	contSobrepos = EntradaEsobrePos(i,tam,pVet,trigger,echo,contSobrepos);
	i++;
	printlinha1(contSobrepos,i,tam,pVet,lcd);
	printlinha2(i,tam,pVet,lcd);
	lcd.print(" ");
}
```

Função para lidar com a limitação de *clock* do Arduino.

```
void wait()
{
	contwait++;
	if(contwait%5 == 0)
	{
		interrupt();
	}
}
```

Por exemplo para esperar 5s por ciclo de iteração. Para realizar essa tarefa utilizei o operador resto. Toda vez que o "*contwait"* tiver resto 0 sendo divido por 5 a função principal -de medição do sensor e representação de valores- é chamada.

SETUP

```
void setup()
{

	Timer1.initialize(1000000);
	Timer1.attachInterrupt(wait);
	pinMode(trigger,OUTPUT);
	pinMode(echo,INPUT);
	pinMode(9,OUTPUT);
	pinMode(10,OUTPUT);
	lcd.begin(16, 2);
	lcd.setCursor(0,0);
	lcd.print("Emilio Gaudeda Junior");
	lcd.setCursor(0,1);
	lcd.print("GRR20191670");
	pVet = (int*) malloc(tam);

}
```

Alocação inicial (tamanho randômico (30,50)), inicialização do programa, do LCD e do Timer.

PROCESSAMENTOS (separados em minhalib.cpp e minhalib.h)

```
float DistanciaCM(int triggerr, int echoo)
{
	float dist;
	digitalWrite(triggerr,LOW);
	delayMicroseconds(5);
	digitalWrite(triggerr,HIGH);
	delayMicroseconds(10);
	digitalWrite(triggerr,LOW);

	dist=pulseIn(echoo,HIGH);
	return dist;

}
```

Essa função faz com que, alterando o nível lógico das portas definidas para trigger e echo, retorne um valor de "distância" que será tratado posteriormente no código

```
int Regrad3(int trigger, int echo)
{
	return map(DistanciaCM(trigger,echo), 400, 18500, 0, 255);

}
```

Utilização da função map para realizar uma “regra de 3” com o valor lido.

```
void parkingstatus(int trigger, int echo)
{
	float entrada = DistanciaCM(trigger, echo)/57.0;
	if(entrada > 60){
		analogWrite(9, 0);
		analogWrite(10,255);
	}
	else if(entrada<30)
	{
		analogWrite(9,255);
		analogWrite(10,0);
	}
	else
	{
		analogWrite(9,255-Regrad3(trigger,echo));
		analogWrite(10,Regrad3(trigger,echo));
	}

}
```

Função que aplica a teoria de PWM em conjunto com a medição do sensor (função já explicado nos objetivos específicos desse relatório) A parte não explicada nos objetivos foi a que se o objeto estiver muito perto do sensor o LED vermelho acenderá com intensidade total e caso o objeto esteja muito longe o LED verde acenderá totalmente também.

```
tem que imprimir os ultimos 3 valores
0,0,0,0,0; cont=0 contsobrepos =0
1,0,0,0,0; cont=1 contsobrepos =0                [ 0 ]
1,2,0,0,0; cont=2 contsobrepos =0                [ 0 ] [  1  ]
1,2,3,0,0; cont=3 contsobrepos =0                [ 0 ] [  1  ] [  2  ]
1,2,3,4,0; cont=4 contsobrepos =0                [ 1 ] [  2  ] [  3  ]
1,2,3,4,5; cont=5 contsobrepos =1               [tam-1] [tam-2] [tam-3]

6,2,3,4,5; cont=6 tam=5 contsobrepos =1          [  0  ] [tam-1] [tam-2]
6,7,3,4,5; cont=7 tam=5                          [  1  ] [ 0   ] [ tam ]
6,7,8,4,5; cont=8 tam=5                          [  2  ] [ 1   ] [  0  ]
6,7,8,9,5; cont=9 tam=5                          [  3  ] [ 2   ] [  1  ]
6,7,8,9,10;cont=10 tam=5                         [  4  ] [ 3   ] [  2  ]
                                                 [ ... ] [ ... ] [ ... ]
                                                 [ tam ] [tam-1] [tam-2]


11,7,8,9,10;[primeiro][ultimo][penultimo]        [  0  ] [tam] [tam-1]
11,12,8,9,10;[segundo][primeiro][ultimo]         [  1  ] [ 0 ] [ tam ]
```

Figura 8 Criação do algoritmo para representar os últimos 3 valores

Essa função lida com a impressão dos valores na primeira linha do LCD. Foi aplicado uma condicional para lidar com a primeira iteração de contagem da sobreposição (`if(contsobrePos == 0))` e outra para quando essa variável já é maior que zero(`else if(contsobrePos >0))`.

```
int EntradaEsobrePos(int cont,int tam,int *pVet,int trigger, int echo, int
contSobrepos)
{
	if(contSobrepos == 0)
	{
		if(cont == tam)
		{
			pVet[cont-(tam*contSobrepos)] =
DistanciaCM(trigger,echo)/57.0;
			contSobrepos++;
		}
		else
		{
			pVet[cont] = DistanciaCM(trigger,echo)/57.0;
		}
	}
	if(contSobrepos >0)
	{
		if(cont >= tam*(contSobrepos+1))
		{
			pVet[cont-(tam*contSobrepos)] =
DistanciaCM(trigger,echo)/57.0;
			contSobrepos++;
		}
		else
		{
```

```
                    pVet[cont-(tam*contSobrepos)] =
DistanciaCM(trigger,echo)/57.0;
                }
        }
        return contSobrepos;
}
```

Para receber os valores do sensor e armazena-los no vetor sobrepondo os valores iniciais quando o tamanho fosse excedido, foi aplicada a lógica acima. Além registrar os valores no sensor essa função também retorna por “int” o valor do contador de sobreposição.

REPRESENTAÇÃO DOS VALORES NO LCD

```
void printlinha1(int contsobrePos, int cont,int tam,int *pVet,LiquidCrystal &lcd)
{
        if(contsobrePos == 0)
        {
                if(cont == 1)
                {
                        lcd.print(pVet[0]);
                }
                else if(cont == 2)
                {
                        lcd.print(pVet[1]);
                        lcd.print(" ");
                        lcd.print(pVet[0]);
                }
                else if(cont >= 3)
                {
                        lcd.print(pVet[cont-1]);
                        lcd.print(" ");
                        lcd.print(pVet[cont-2]);
                        lcd.print(" ");
                        lcd.print(pVet[cont-3]);
                }
        }

        else if(contsobrePos >0)
        {
                if(cont-(contsobrePos*tam) == 1)//6-5 //11-10//16-15
                {
                        lcd.print(pVet[tam-1]);
                        lcd.print(" ");
                        lcd.print(pVet[tam-2]);
                        lcd.print(" ");
                        lcd.print(pVet[tam-3]);
                }
                if(cont-(contsobrePos*tam) == 2)
                {
                        lcd.print(pVet[0]);
                        lcd.print(" ");
                        lcd.print(pVet[tam-1]);
                        lcd.print(" ");
                        lcd.print(pVet[tam-2]);
                }
                if(cont-(contsobrePos*tam) == 3)
```

```
		{
			lcd.print(pVet[1]);
			lcd.print(" ");
			lcd.print(pVet[0]);
			lcd.print(" ");
			lcd.print(pVet[tam-1]);
		}
		if(cont-(contsobrePos*tam) >= 4)//9-5//14-10//19-15
		{
			lcd.print(pVet[cont-(contsobrePos*tam)-2]);
			lcd.print(" ");
			lcd.print(pVet[cont-(contsobrePos*tam)-1]);
			lcd.print(" ");
			lcd.print(pVet[cont-(contsobrePos*tam)]);
		}
	}
	lcd.setCursor(8,0);
	lcd.print("i");
	lcd.print(cont);
	lcd.print(" ");
	lcd.print("$");
	lcd.print(contsobrePos);

}
```

Impressão dos valores da linha 1, sendo eles:

Últimos 3 valores lidos, contador e contador de sobreposições no vetor.

```
void printlinha2(int cont,int tam,int *pVet,LiquidCrystal &lcd)
{
	lcd.setCursor(0,1);
	lcd.print("S:");
	lcd.print(soma(pVet,tam,cont));
	lcd.print("M:");
	lcd.print(media(cont,pVet,tam));
}
```

Impressão dos valores calculados de soma e média na linha dois do LCD 16x2.

SOMA E MÉDIA

```
float soma(int *pVet,int tam,int cont)
{
	float aux = 0.0;
	if(cont >= tam)
	{
		for (int a = 0; a < tam; a++)
		{
			aux += *(pVet + a);
```

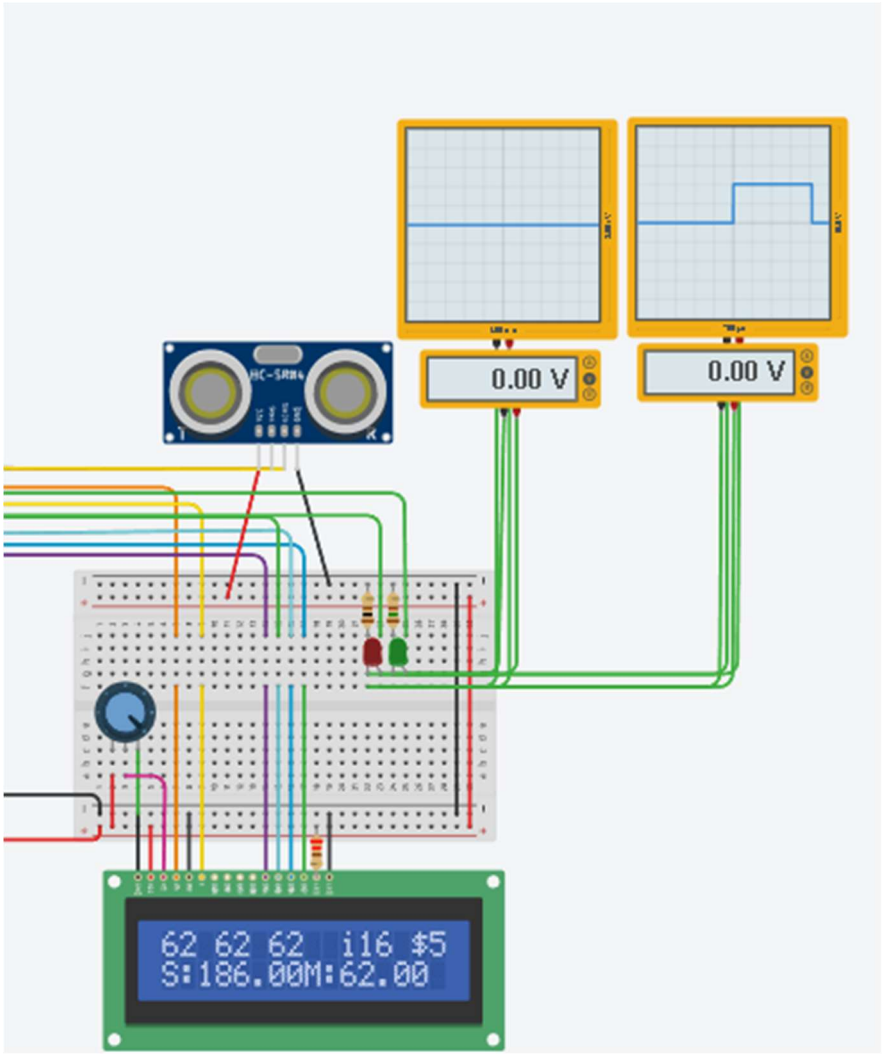
```
		}
	}
	else
	{
		for (int a = 0; a < cont; a++)
		{
			aux += *(pVet + a);
		}
	}

	return aux;

}
```

Função para obter a soma dos valores lidos. Nessa função foi utilizado uma condicional para verificar se o número de valores lidos é maior que o tamanho do vetor definido por random(30,50). Caso verdadeiro, o programa soma todos os valores do vetor, caso contrário são somados somente os valores lidos até então. (função foi declarada em um arquivo .cpp e .h separado do sketch.cpp)

```
float media(int cont,int *pVet,int tam)
{
	if(cont<=tam)
	{
		return (float)soma(pVet,tam,cont)/(float)cont;
	}
	else
	return (float)soma(pVet,tam,cont)/(float)tam;
}
```

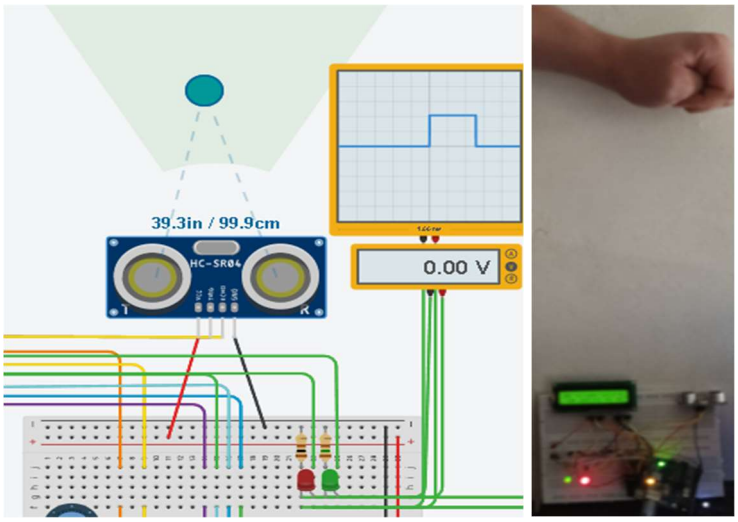
Cálculo da média levando em consideração as iterações pelo vetor antes e depois dele ser preenchido pela primeira vez.(função foi declarada em um arquivo .cpp e .h separado do sketch.cpp)

DEMONSTRAÇÃO DA SIMULAÇÃO NO TINKERCAD E DO CIRCUITO REAL DESENVOLVIDO NO MICROSHIP STUDIO



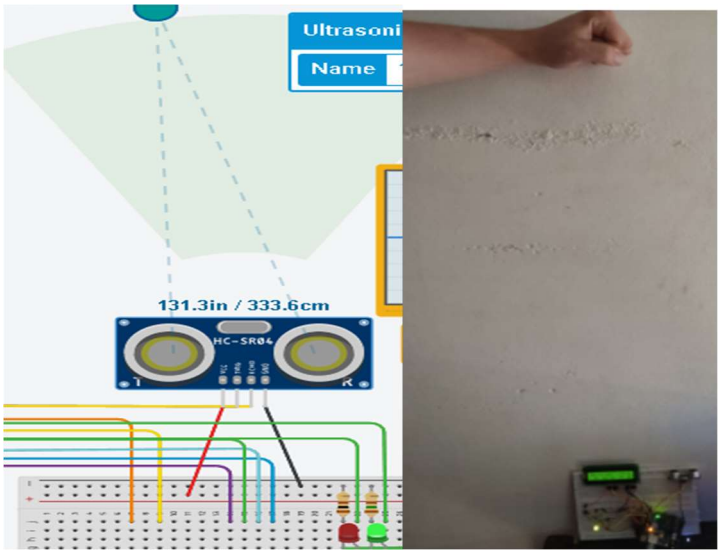
Últimos 3 valores lidos seguidos do contador e o contador de sobreposições no vetor (que foi colocado com tamanho 3 nesse caso apenas para demonstrar o funcionamento do circuito e de suas funções)

Na linha dois temos a soma dos valores lidos e ao lado a média entre eles.



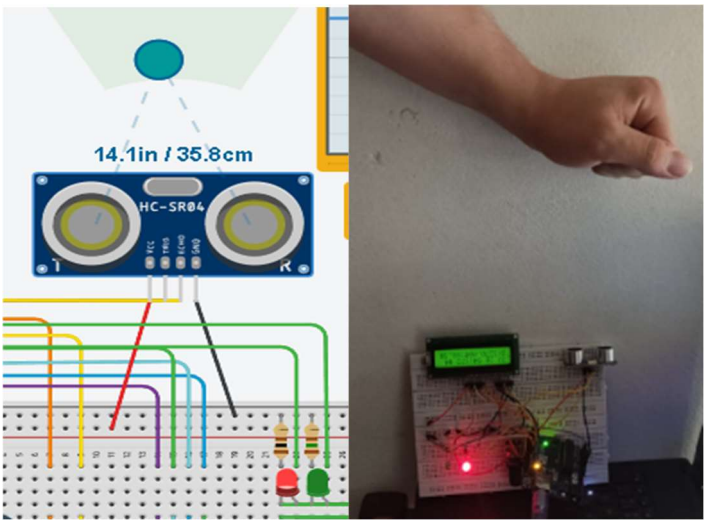
Quando o objeto está na metade do caminho (PARKING)

Os dois leds acesos contudo o led vermelho com um pouco mais de intensidade pois o objeto está mais próximo do que longe do sensor (para controlar essa intensidade foi utilizado PWM.



Objeto distante (FREE)

Objeto além de uma distância X do sensor, ativando o nível lógico alto para o LED vede e nível lógico baixo para o LED vermelho



Objeto próximo (PARKED)

Objeto próximo do sensor ativando nível lógico alto para o LED vermelho e nível lógico baixo para o LED verde.

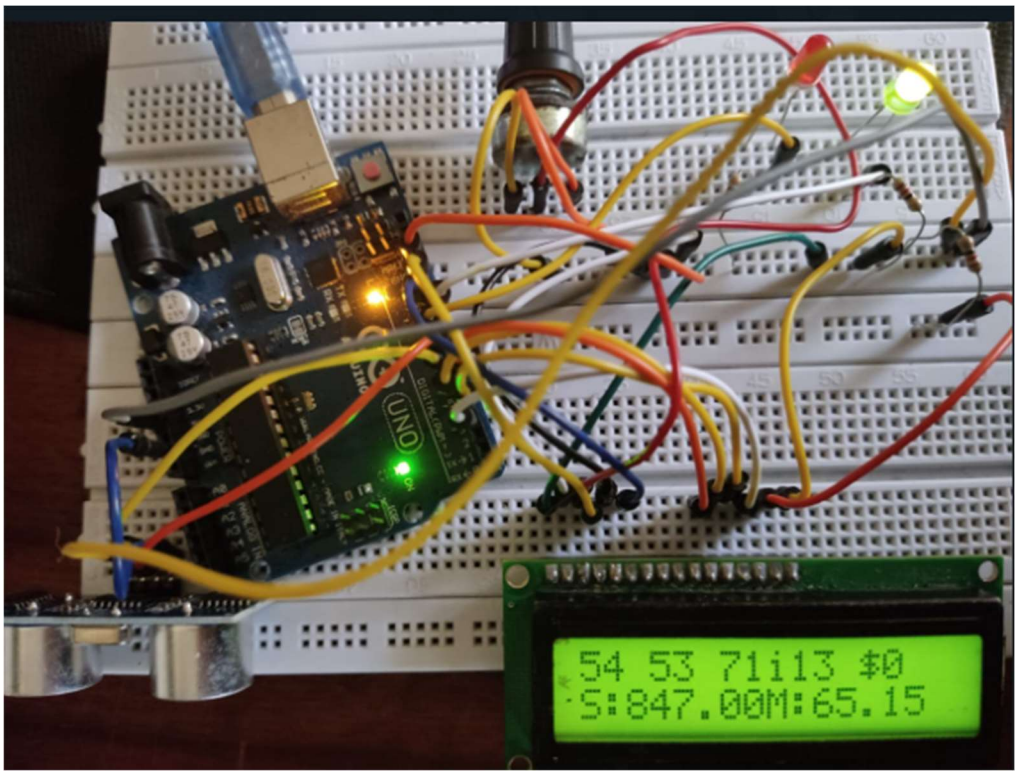


Figura 9 Demonstração do funcionamento do circuito real:

Primeira linha: Últimos 3 valores lidos de distância em centímetros seguidos pelo contador e ao lado o contador de sobreposição;

Segunda linha: a soma dos valores de dentro do vetor e ao lado a média desses valores.

CONCLUSÂO

Como demonstrado ao longo desse relatório, os objetivos foram concluídos. Esses foram: aplicar a teoria PWM ao projeto, utilizar um dispositivo de saída como LCD, escolha de um sensor, utilização de interrupção por timer, alocação inicial de memória com valor aleatório de (30,50) e representar os últimos valores lidos e apresentar a média de todos os valores lidos, a soma desses, a quantidade de valores lidos e a quantidade de vezes que o vetor foi totalmente preenchido.

Além disso, é válido destacar a melhora no código em comparação ao trabalho anterior, pois não foram declaradas variáveis globais nos arquivos das bibliotecas e as funções receberam os valores por referência.