Arduino UNO R3: Sensores

Física Aplicada à Computação - Licenciatura em Engenharia Informática

Hugo Alexandre Silva

Beja - Portugal

Sumário

Introdução

Este trabalho visa testar o funcionamento dos sensores usando o Arduino como plataforma para os dados recolhidos, neste caso, da Temperatura do ar, Humidade Relativa e Ultra-sons, para distância. Neste trabalho também apresento os códigos necessários para seu pronto funcionamento, como funciona o próprio sensor e para que ele serve.

O projeto também visa também adaptar o código ao uso de suas ferramentas, sendo elas o dispositivo de Ultra-sons HC-SR04, Sensor de Temperatura do Ar e Humidade Relativa DHT-11 e Sensor de Ar e Pressão Atmosférica BMP280.

Utilização do Arduino que é uma plataforma de prototipagem eletrônica de hardware em placa única, projetada inicialmente com um microcontrolador Atmel AVR, trabalhando com periféricos com conexões de entrada e saída constituído por 14 pinos, assim como circuitos eletrônicos e manipuláveis em uma linguagem de programação padrão. A linguagem tem origem em Wiring e é essencialmente C/C++. A visão do projeto é criar ferramentas acessíveis, com baixo custo, flexíveis e fáceis de se usar por principiantes e profissionais, aqueles que não teriam alcance aos controladores mais sofisticados e ferramentas mais complicadas. Em relação a pinagem são: 6 podem ser usados como saídas PWM, 6 entradas analógicas, uma conexão USB, uma entrada de alimentação com conexão ICSP e um botão de reset. Cada um dos 14 pinos digitais do UNO podem ser utilizados como uma entrada ou saída utilizando-se as funções pinMode(), digitalWrite() e digitalRead(). Eles operam a 5.0V e cada pino pode fornecer ou receber um máximo de 40mA.

Neste trabalho utilizo o Arduino UNO R3 como sistema de aquisição de dados de sensores nomeadamente sensores da temperatura com cálculos e fórmulas.

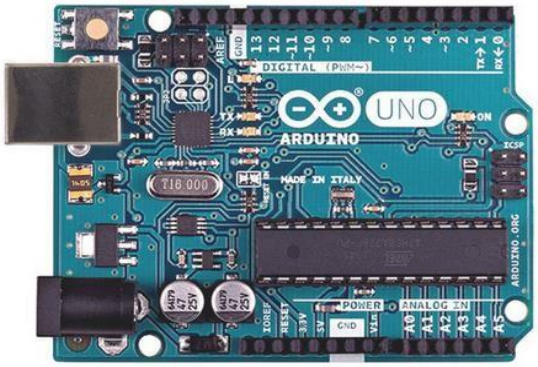


Figura 1 – Arduino UNO R3

O presente trabalho estruturar-se-á do seguinte modo:

"As versões do programa associadas a cada sensor, devem permitir: (i) visualizar os níveis de alerta já implementados para as grandezas medidas pelo sensor e (ii) a respectiva visualização num circuito com LEDs, montado numa breadboard. Adicionalmente, deve ser implementada a média M "deslizante". Neste caso, o valor é calculado de acordo com a últimas M medidas, de acordo com o exemplo da tabela seguinte (para o caso de M=3)"

k	G_k	$\langle G \rangle_3$
1	G_1	-
2	G_2	-
3	G_3	$(G_1+G_2+G_3)/3$
4	G_4	$(G_2+G_3+G_4)/3$
5	G_5	$(G_3+G_4+G_5)/3$

Citação do manual projeto FAC – Mestre Nuno.

Objetivo

O trabalho tem de dar resposta a quatro objetivos, que devem constar no relatório:

1) fazer o levantamento das características técnicas mais importantes de cada sensor como, por exemplo, o consumo, o tipo de comunicações e o intervalo de valores admissíveis para cada grandeza medida (consultar os respectivos “datasheets”)

2) testar o sensor com o código fornecido de modo a perceber o seu funcionamento;

3) desenvolver uma função que permite integrar as medições do sensor no código desenvolvido no trabalho anterior;

4) gerar “outputs” com o programa para o Serial Monitor, Serial Plotter, e gnuplot, usando as várias médias e smoothening.

Aparato Experimental

- Dispositivo de Ultra-Sons HC-SR04;
- Sensor de Temperatura do Ar e Humidade Relativa DHT-11;
- Sensor de Temperatura do Ar e Pressão Atmosférica BMP280;

Dispositivo de Ultra-Sons HC-SR04

O dispositivo de alcance de ultra sons HC-SR04 dispõe de uma função de medição de 2 a 400cm sem contato, a precisão de seu alcance pode chegar a 3mm, este dispositivo inclui transmissores e receptores ultrassônicos e circuito de controle.

Este dispositivo dispõe de mecanismos como:

- Um trigger IO de pelo menos 10us de sinal de alto nível

- O dispositivo automaticamente manda oito 40kHz e detecta se existe ou não um retorno do sinal.

- Se existir um retorno, ele calcula o tempo que demorou por alta intensidade do trigger IO em distância = Tempo x Velocidade do Som(340M/S)/2.

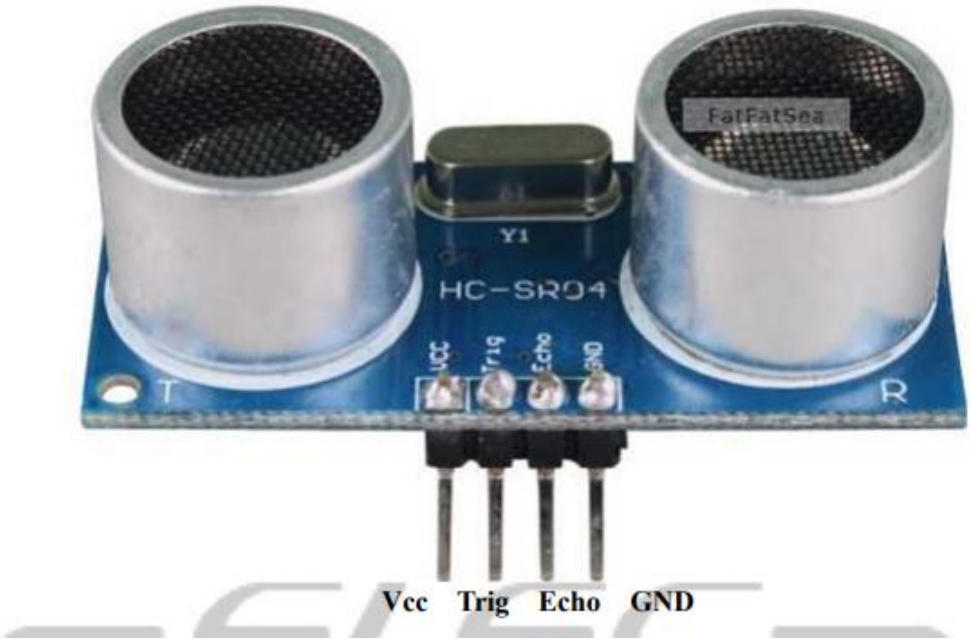


Figura 2 – HC-SR04

Timing Diagram

O diagrama de sincronização do sensor apresenta-se embaixo ou “Timing Diagram”, só sendo preciso de um pulso de 10uS para o trigger, o dispositivo começa o seu processo de medição com um disparo de 8 ciclos de ultrassom a 40 kHz e aumentando seu eco. O eco é um objeto de distância cujo pulso em amplitude e oscilação em proporção, conseguindo-se assim calcular a distância pelo intervalo de tempo entre mandar o sinal “trigger” e receber o sinal “eco”.

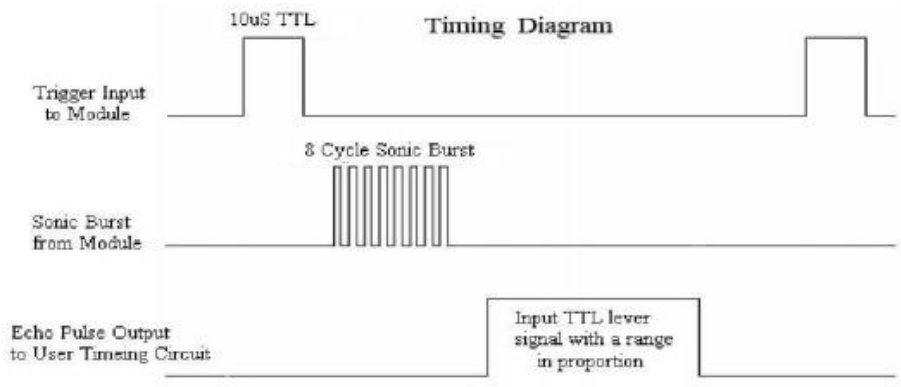


Figura 3 – Diagrama a mostrar o funcionamento do começo e retorno do sinal do sensor HC-SR04.

Sensor de Temperatura do Ar e Humidade Relativa DHT-11

O sensor DHT-11 é um sensor composto que tem um output de sinal digitar calibrado de temperatura e humidade, o sensor inclui componentes resistentes a água e dispositivos de medição de temperatura NTC (negative temperature coefficient), conectados entre um microcontrolador.

Este sensor tem várias utilizações, como controlador de temperatura e humidade, desde dentro de ambientes fechados (sendo utilizado como regulador) ou em ambientes aberto (como coletores de dados).

Como sensor, ele também tem um diagrama de sincronização, onde o usuário manda um sinal, que o sensor converte de baixa voltagem para alta velocidade, até o usuário mandar o final do sinal do sensor, para mandar uma reposta de 40bits de data. Este sinal é representado abaixo.

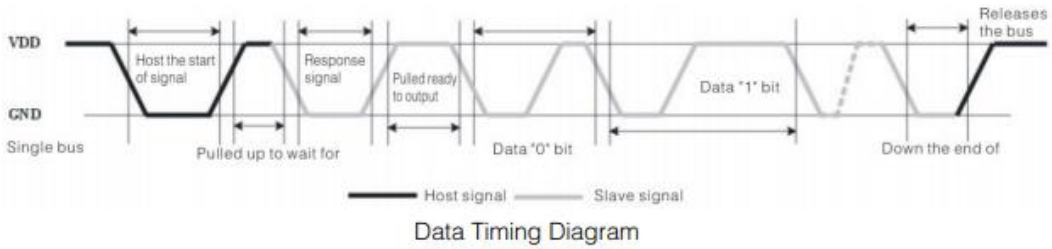


Figura 4 – Diagrama do sensor DHT11

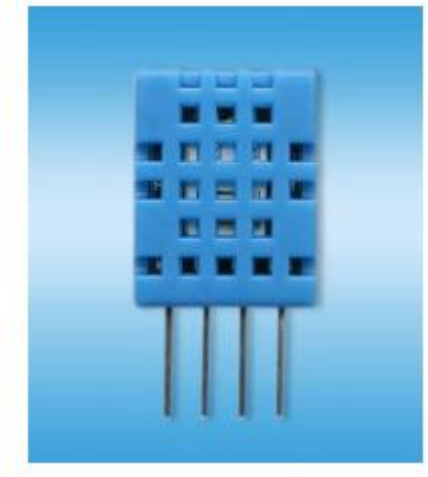


Figura 4 – Sensor DHT11

Sensor de Temperatura de Ar e Pressão Atmosférica BMP280

O módulo BMP280 funciona com interfaces I2C ou SPI e tensão de 3V, sendo que o baixo consumo de energia permite o funcionando por longos períodos com alimentação por bateria, e é indicado para projetos como drones, estações meteorológicas, dispositivos com GPS, relógios, etc.

Parte técnica do sensor: consumo, parâmetros, valores admissíveis, grandeza de valores.

– Tensão de operação: 3V

– Consumo de corrente: 2.7µA

– Interfaces: I2C e SPI

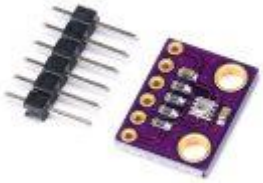
– Faixa de medição pressão: 300 – 1100hPa (equiv. +9000 à -500m acima/abaixo do nível do mar)

– Precisão: ±0.12hPa (equiv. ±1m)

– Faixa de temperatura: -40 a 85 °C

– Precisão temperatura: ±1.0 °C

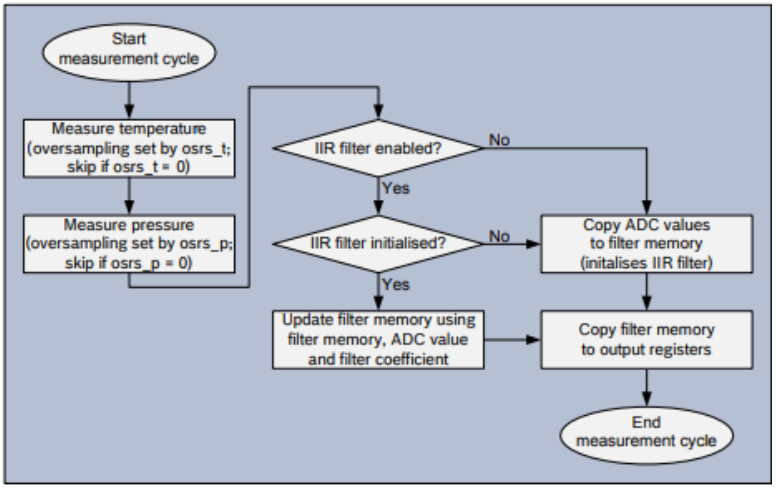
– Dimensões: 15 x 12 x 2,3mm (sem os pinos)



O BMP280 é um sensor de pressão barométrico desenhado para dispositivos moveis O módulo sensor está alojado em uma tampa metálica extremamente compacta de 8 pinos LGA pacote com uma pegada de apenas 2,0 × 2,5 mm2 e altura do pacote de 0,95 mm. É pequeno dimensões e seu baixo consumo de energia de 2,7 µA @ 1Hz permitem a implementação dispositivos acionados com bateria, como telefones celulares, módulos GPS ou relógios.

Fluxo de medição

O período de medição da BMP280 consiste em uma medição de temperatura e pressão com oversampling selecionável. Após o período de medição, os dados são passados por um filtro IIR opcional, que elimina flutuações de curto prazo na pressão. O fluxo é representado no diagrama abaixo.



Medição de pressão

A medição de pressão pode ser ativada ou ignorada. Ignorar a medição pode ser útil se o BMP280 for usado como sensor de temperatura. Quando ativado, existem várias opções de sobre amostragem. Cada etapa de sobre amostragem reduz o ruído e aumenta a resolução de saída em um bit, que é armazenado no registrador de dados XLSB 0xF9. A ativação / desativação das configurações de medição e sobre amostragem é selecionada através dos bits osrs_p [2: 0] no registrador de controle 0xF4.

Oversampling setting	Pressure oversampling	Typical pressure resolution	Recommended temperature oversampling
Pressure measurement skipped	Skipped (output set to 0x80000)	–	As needed
Ultra low power	×1	16 bit / 2.62 Pa	×1
Low power	×2	17 bit / 1.31 Pa	×1
Standard resolution	×4	18 bit / 0.66 Pa	×1
High resolution	×8	19 bit / 0.33 Pa	×1
Ultra high resolution	×16	20 bit / 0.16 Pa	×2

Medição de temperatura

A medição de temperatura pode ser ativada ou ignorada. Ignorando a medida pode ser útil para medir a pressão de forma extremamente rápida. Quando ativado, várias opções de sobre amostragem existir. Cada etapa de sobre amostragem reduz o ruído e aumenta a resolução de saída em um bit, que é armazenado no registrador de dados XLSB 0xFC.

osrs_t[2:0]	Temperature oversampling	Typical temperature resolution
000	Skipped (output set to 0x80000)	–
001	×1	16 bit / 0.0050 °C
010	×2	17 bit / 0.0025 °C
011	×4	18 bit / 0.0012 °C
100	×8	19 bit / 0.0006 °C
101, 110, 111	×16	20 bit / 0.0003 °C

Filtro de resposta ao impulso infinito (Infinite impulse response IIR filter):

A pressão ambiental está sujeita a muitas mudanças a curto prazo, causadas por exemplo: por bater uma porta ou janela, ou vento soprando no sensor. Para suprimir essas perturbações na saída dados sem causar tráfego de interface adicional e carga de trabalho do processador, os recursos do BMP280 um filtro IIR interno. Ele reduz efetivamente a largura de banda dos sinais de saída. A saída de uma próxima etapa de medição é filtrada usando a seguinte fórmula:

$$data\_filtered = \frac{data\_filtered\_old \cdot (filter\_coefficient - 1) + data\_ADC}{filter\_coefficient}$$

Código Ultrasom

Esse é o código utilizado juntamente com os cálculos e valores obtidos através do sensor de proximidade para elaboração da segunda parte do projeto. A saber DURAÇÃO E DISTANCIA.

```
//variáveis globais que irão mostrar os valores dos sensores e utilização durante a execução do trabalho;

byte Vcc = 11;

byte Trig = 12;

byte Echo = 13;

float distance;            // variável para receber valor de distância;

byte triggerDuration = 10;        // us, "conforme citação do mestre nuno e unidade de medida;

unsigned int sampleDelay = 10;      // ms, "conforme citação do mestre nuno e unidade de medida;

unsigned int timeout = 23530;           // tempo correspondente a d > 4m, "conforme citação do mestre nuno e unidade de medida;

//escolho a variável float por receber números decimais que ocupam 32 bits(4 bytes);

//assim não precisaríamos usar a variável double que ocupa o dobro em alguns casos, economizando até mesmo no espaço de memória;

//essas variáveis podem tomar valores entre -3.4028235E+38 e +3.4028235E+38.

float duration_a = 0.;     // "conforme citação do mestre nuno e unidade de medida. nome variável. recebe duração;

float distance_a = 0.;

float duration_suav = 0.;   // "conforme citação do mestre nuno e unidade de medida. nome variável. recebe suavização duração;
```

```
float distance_suav = 0.;   // "conforme citação do mestre nuno e unidade de medida. nome variável. recebe suavização distancia;

int contador = 0;          // contador usado para as formulas, contagem das medições. ideia baseada do primeiro trabalho FAC;

const int valor_media = 3;    // contador usado para as formulas, contagem das medições deslizante, ideia baseada do primeiro trabalho FAC;

const float suavizacao = 0.3; // suavização usado para as fórmulas, ideia baseada do primeiro trabalho FAC;

float media_a = 0.;          //variável para medias em geral;

float media_b = 0.;          //variável para medias em geral;

float duration_media = 0.;    //variável para media da duração;

float distance_media = 0.;    //variavel para media da distancia;

float duration_mmm = 0.;      //variavel para media acumulativa;

float distance_mmm = 0.;    //variavel para media acumulativa;

float duration_array[valor_media];    //array para valores de duração, ideia baseada do primeiro trabalho FAC;;

float distance_array[valor_media];    //array para valores de distância, ideia baseada do primeiro trabalho FAC;;

//após a formulação do cálculo os acendimentos dos led ocorreram nos seguintes pinos;

//assim como ocorre no trabalho 1 descritivo ao qual utilizo como base e ideia para as formulas.

  const int pino7 = 7; //byte pino7 = 7;    const int pino6 = 6; //byte pino6 = 6;

  const int pino5 = 5; //byte pino5 = 5;    const int pino4 = 4; //byte pino4 = 4;

  //#define ultrasom

    //#define mms

    //#define mmc

    //#define mmm

    //#define mmd

    //#define mserial

    //#define mplotter

    //#define gnuplot

#ifdef ultrasom

// Initialization

void setup() {  pinMode(Vcc, OUTPUT);

  pinMode(Trig, OUTPUT);

  pinMode(Echo, INPUT);
```

```
  digitalWrite(Vcc, HIGH);
  digitalWrite(Trig, LOW);
  Serial.begin(9600);
  while(!Serial);
}  // Main loop
void loop() {
  delay(sampleDelay);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(triggerDuration);
  digitalWrite(Trig, LOW);
  unsigned long duration = pulseIn(Echo, HIGH, timeout);
  distance = (float)duration/58.8;
#ifdef mmm                    //início da media mmm de 3 valores, de ambos valores, distancia e duração.
  duration_array[contador]=duration;    //aqui utilizo o mesmo seguimento e ação que foi feito no
trabalho 1.
  distance_array[contador]=distance;
  contador++;
if(contador == valor_media) {    duration_media = 0.;
  distance_media = 0.;
for(int k = 0; k < valor_media; k++){
  duration_media=duration_media + duration_array[k];
  distance_media=distance_media + distance_array[k];
}   duration_media=duration_media / valor_media;
  distance_media=distance_media / valor_media;
  contador = 0;
}  #endif mmm
 #ifdef mms                    // início dos cálculos das ações de suavização;
  duration_a = duration_suav;        //duração receber valor de suavização para armazenamento;
  distance_a = distance_suav;
  duration_suav = duration + (duration_a - duration) * suavizacao;  //função de cálculo para executar a
suavização;
  distance_suav = distance + (distance_a - distance) * suavizacao;
#endif mms
```

```
#ifdef mmc                    // início do cálculos das media correntes ou acumulativas;
  contador++;
  duration_mmm += duration;       // aqui utilizo as funções e seguimentos conforme trabalho 1,
baseando em criar resultados;
  distance_mmm += distance;              //para distancia e duração conforme valores pretendidos;
  duration_media = duration_mmm / contador;
  distance_media = distance_mmm / contador;
#endif mmc
#ifdef mmd                     //início da ação de cálculo das medias deslizantes;
if(contador < valor_media) {          //início de array para armazenar valores de duração e distancia;
    duration_array[contador]=duration;
    distance_array[contador]=distance;
    contador++;
}  if(contador == valor_media) {          // baseado no descritivo do quadro do mestre nuno, esta formula,
for(int k = 0; k < valor_media; k++){       //efetua a troca dos valores dos array e traz novas medições
trocando a mais antiga;
   if(k < (valor_media - 1)){     duration_array[k]=duration_array[k + 1];
     distance_array[k]=distance_array[k + 1];
} else {     duration_array[k]=duration;
     distance_array[k]=distance;
}}  for(int k = 0; k < valor_media; k++){              // dentro das medias já calculadas e armazenadas no
array,
    duration_media=duration_media + duration_array[k];    //irei receber novo valor de duração medida e
armazenar dentro de um array e
    distance_media=distance_media + distance_array[k];    //utilizar para efetuar o cálculo de nova media
com valores atualizados;
}     duration_media=duration_media / valor_media;
    distance_media=distance_media / valor_media;
}  #endif mmd
#ifdef gnuplot
  Serial.print(F("#Fisica Aplicada Computação | 2018-19"));
  Serial.print(F("#Duration (us)\tDistance (cm)"));
  Serial.print(F("\tSuavização duration (us)\tSuavização distance (cm)"));
  Serial.println(F("\tMedia duration (us)\tMedia distance (cm)"));
#endif gnuplot
```

```
#ifdef mplotter
  Serial.print(F("\tSuavização duration(us): "));
  Serial.print(duration_suav);        //imprimi a suavização da duração;
  Serial.print(F("\tSuavização distance(us): "));
  Serial.print(distance_suav);        //imprimi a suavização da distância;
  Serial.print(F("\tDuration(us): "));
  Serial.print(duration);             //imprimi a duração, "conforme citação do mestre nuno e unidade de medida. nome variavel. recebe duração;
  Serial.print(F("\tDistance(cm): "));
  Serial.println(distance);    //imprimi a distância, "conforme citação do mestre nuno e unidade de medida. nome variavel. recebe distancia;
#endif mplotter
#ifdef mserial
  Serial.print(F("\tMedia duration(us): "));
  Serial.print(duration_media);           //imprimi a média da duração, media final;
  Serial.print(F("\tMedia distance(cm): "));
  Serial.print(distance_media);         //imprimi a média da distância, media final;
  Serial.print(F("\tDuration(us): "));
  Serial.print(duration);             //imprimi a duração, "conforme citação do mestre nuno e unidade de medida. nome variavel. recebe duração;
  Serial.print(F("\tDistance(cm): "));
  Serial.println(distance);    //imprimi a distância, "conforme citação do mestre nuno e unidade de medida. nome variavel. recebe distancia;
  if (duration > 0 && duration <= 9000){   digitalWrite(pino7, HIGH);        // para os valores de if luz de led acessa, estará HIGH;
   Serial.print(F("LED Blue  "));
}  if (duration > 9000 && duration <= 55000){
   digitalWrite(pino6, HIGH);     // para os valores de if luz de led acessa, estará HIGH;
   Serial.print(F("LED Yellow  "));
}  if (distance > 0 && distance <= 1900){
   digitalWrite(pino5, HIGH);    // para os valores de if luz de led acessa, estará HIGH;
   Serial.print(F("LED Green  "));
}  if (distance > 1900 && distance <= 4100){
   digitalWrite(pino4, HIGH);        // para os valores de if luz de led acessa, estará HIGH;
   Serial.print(F("LED Red  "));
```

} #endif mserial }

#endif ultrasom //------------------------------

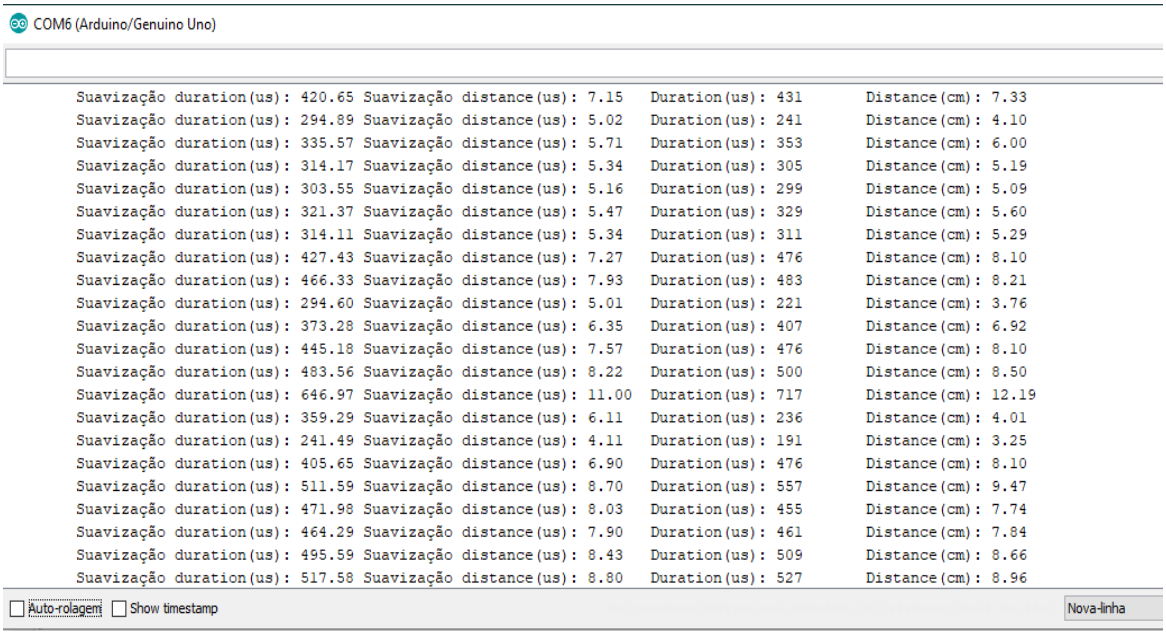


Figura – Serial ultrassom com suavização

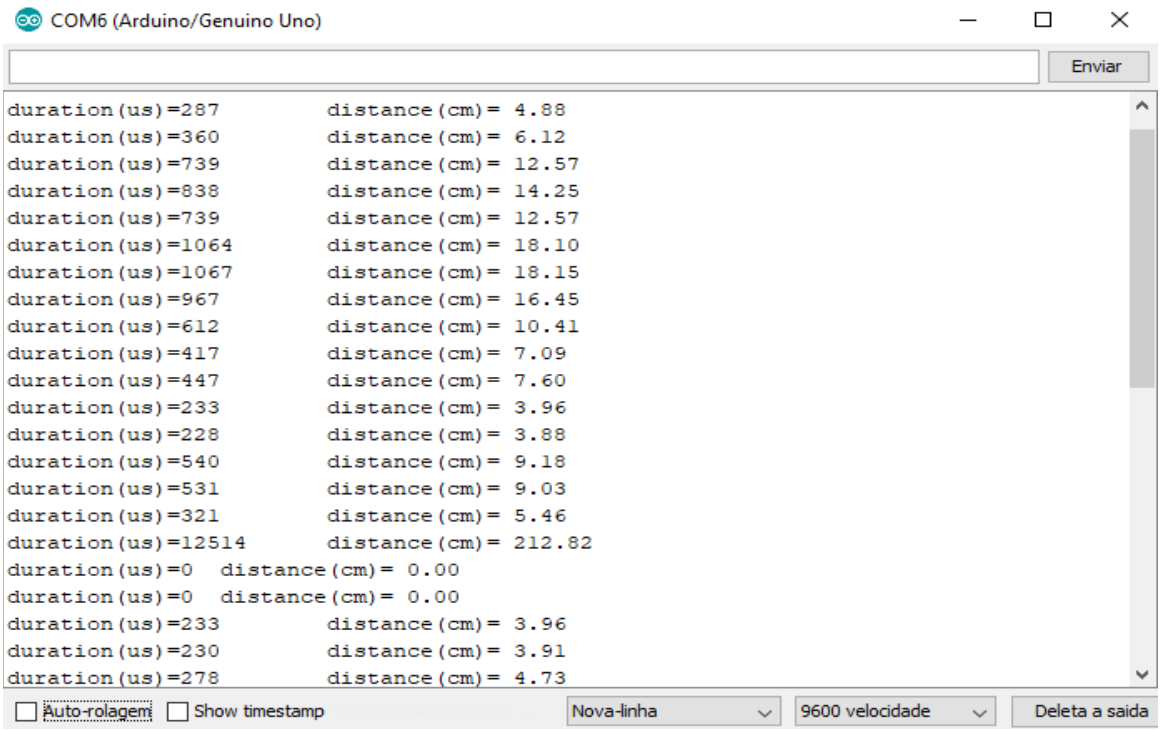


Figura – Serial ultrassom base principal com duração e distancia

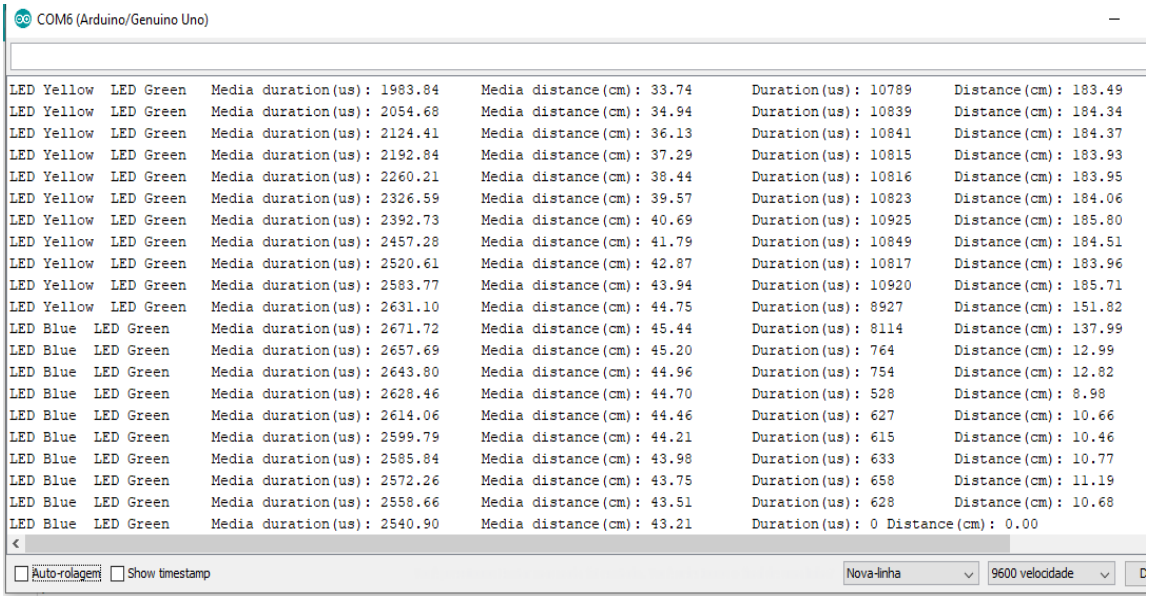


Figura – Serial ultrassom com medias e led

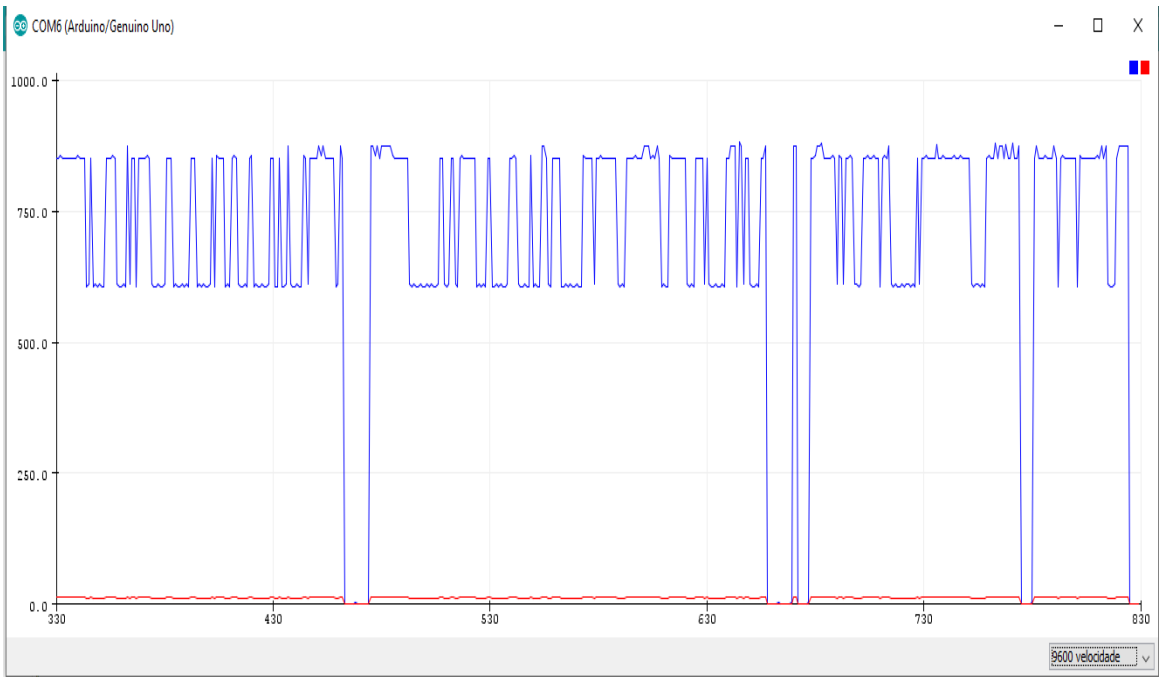


Figura – Monitor ultrassom base principal com duração e distancia

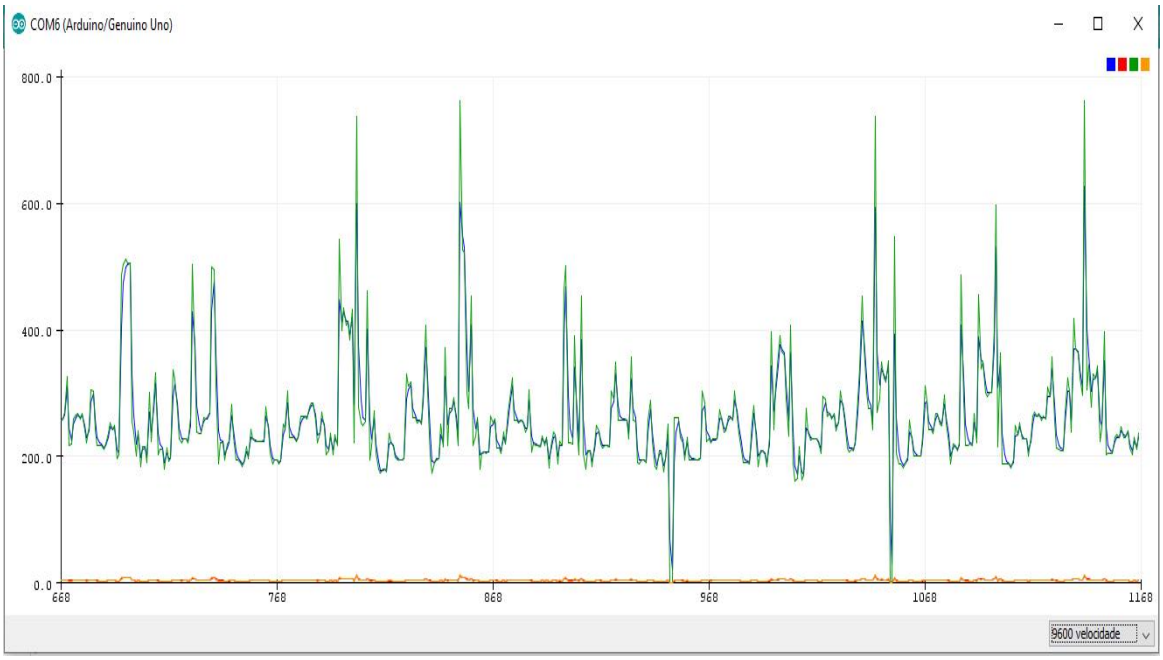


Figura – Monitor ultrassom duração e distancia com suavização

Temperatura e Humidade DHT11

Nesta etapa do projeto efetuarei as medições de temperatura e humidade relativas. Como base de testes utilizarei sopro de ar quente, isqueiro e ar frio para captação das possíveis variações de temperatura e visualização em serial monitor e plotter.

Utilizo a seguinte formulação e código abaixo:

```
/*
Sensor de Temperatura do Ar & Humidade Relativa DHT-11
Biblioteca: Adafruit DHT11 + Adafruit Unified Sensor
Código para teste:
*/
#ifdef humidade
#include "DHT.h"
// Declaration of global variables
```

```
byte DHTpin = 2;       // Digital pin to connect pin 2 of the sensor
int delaySample = 2000; // delay between measurements in ms
// Definition of the sensor
DHT dht(DHTpin, DHT11);
// Initialization
void setup() {   Serial.begin(115200); // Start serial port communications
  Serial.println(F("Fisica Aplicada Computação - EI (v1.0) | 2018-19\nDHT11 test\n"));
  dht.begin();  // Initialize sensor
  delay(delaySample);
  } // Main loop
  void loop() {    delay(delaySample);
  float h = dht.readHumidity();    // Read relative humidity
  float t = dht.readTemperature();  // Read temperature in Celcius
  // Check validity of readings
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Invalid Data!"));
    return;   // Loops to the reading instructions without printing data
    } #ifdef mms_humidade
    duration_a = duration_suav;
    distance_a = distance_suav;
    duration_suav = h + (duration_a - h) * suavizacao;
    distance_suav = t + (distance_a - t) * suavizacao;
#endif mms_humidade
ifdef mmm_humidade
  duration_array[contador] = h;
  distance_array[contador] = t;
  contador++;
if(contador == valor_media) {
  duration_media = 0.;
  distance_media = 0.;
for(int k = 0; k < valor_media; k++){
  duration_media=duration_media + duration_array[k];
  distance_media=distance_media + distance_array[k];
}   duration_media=duration_media / valor_media;
  distance_media=distance_media / valor_media;
```

```
  contador = 0;
} #endif mmm_humidade
 #ifdef mmc_humidade // início do cálculos das media correntes ou acumulativas
  contador++;
  duration_mmm += h;
  distance_mmm += t;
  duration_media = duration_mmm / contador;
  distance_media = distance_mmm / contador;
#endif mmc_humidade
#ifdef mmd_humidade  //início da ação de cálculo das medias deslizantes;
if(contador < valor_media) {
    duration_array[contador] = h;
    distance_array[contador] = t;
    contador++;
}  if(contador == valor_media) {  // baseado no descritivo do quadro do mestre nuno, esta formula,
for(int k = 0; k < valor_media; k++){   //efetua a troca dos valores dos array e traz novas medições trocando a mais antiga;
   if(k < (valor_media - 1)){
     duration_array[k]=duration_array[k + 1];
     distance_array[k]=distance_array[k + 1];
} else {     duration_array[k] = h;
     distance_array[k] = t;
}}   for(int k = 0; k < valor_media; k++){                // dentro das medias já calculadas e armazenadas no array,
    duration_media=duration_media + duration_array[k];    //irei receber novo valor de duração medida e armazenar dentro de um array e
    distance_media=distance_media + distance_array[k];    //utilizar para efetuar o cálculo de nova media com valores atualizados;
}    duration_media=duration_media / valor_media;
    distance_media=distance_media / valor_media;
}  #endif mmd_humidade
#ifdef gnuplot_humidade
  Serial.print(F("#Fisica Aplicada Computação | 2018-19"));
  Serial.print(F("#Duration (us)\tDistance (cm)"));
  Serial.print(F("\tSuavização Humidity (%)\tTemperature (ºC)"));
  Serial.println(F("\tMedia Humidity (%)\tMedia Temperature (ºC)"));
#endif gnuplot_humidade
```

```
#ifdef mplotter_humidade
  Serial.print(F("\tSuavização duration(us): "));
  Serial.print(duration_suav);         //imprimi a suavização da duração;
  Serial.print(F("\tSuavização distance(us): "));
  Serial.print(distance_suav);        //imprimi a suavização da distância;
Serial.print(F("\tHumidity (%): "));
  Serial.print(h);
  Serial.print(F("\tTemperature (ºC): "));
  Serial.println(t);
#endif mplotter_humidade
#ifdef mserial_humidade
  Serial.print(F("\tMedia duration(us): "));
  Serial.print(duration_media);           //imprimi a média da duração, media final;
  Serial.print(F("\tMedia distance(cm): "));
  Serial.print(distance_media);         //imprimi a média da distância, media final;
  Serial.print(F("\tHumidity (%): "));
  Serial.print(h);
  Serial.print(F("\tTemperature (ºC): "));
  Serial.println(t);
  if (h > 0 && h <= 75){
   digitalWrite(pino7, HIGH);        // para os valores de if luz de led acessa, estará HIGH;
   Serial.print(F("LED Blue  "));
}  if (h > 75 && h <= 150){
   digitalWrite(pino6, HIGH);     // para os valores de if luz de led acessa, estará HIGH;
   Serial.print(F("LED Yellow  "));
}  if (t > 0 && t <= 25){
   digitalWrite(pino5, HIGH);    // para os valores de if luz de led acessa, estará HIGH;
   Serial.print(F("LED Green  "));
}  if (t > 25 && distance <= 70){
   digitalWrite(pino4, HIGH);        // para os valores de if luz de led acessa, estará HIGH;
   Serial.print(F("LED Red  "));
}  #endif mserial_humidade
}   #endif humidade  ------------------------------------------
```

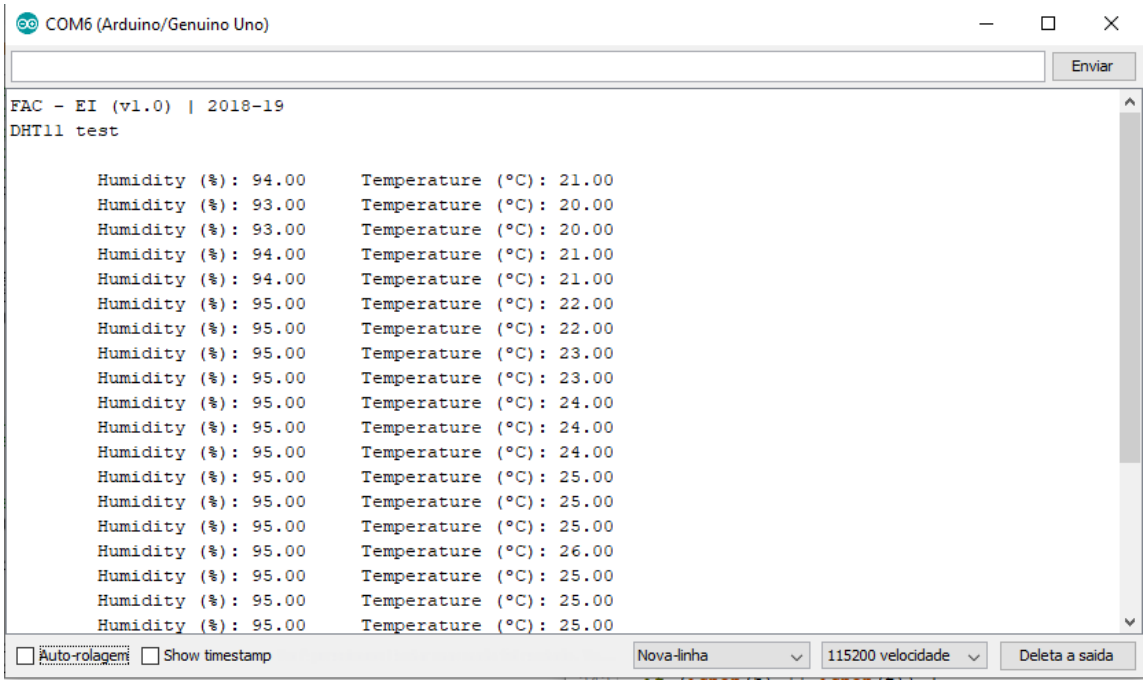


Figura – Serial temperatura e humidade com base principal

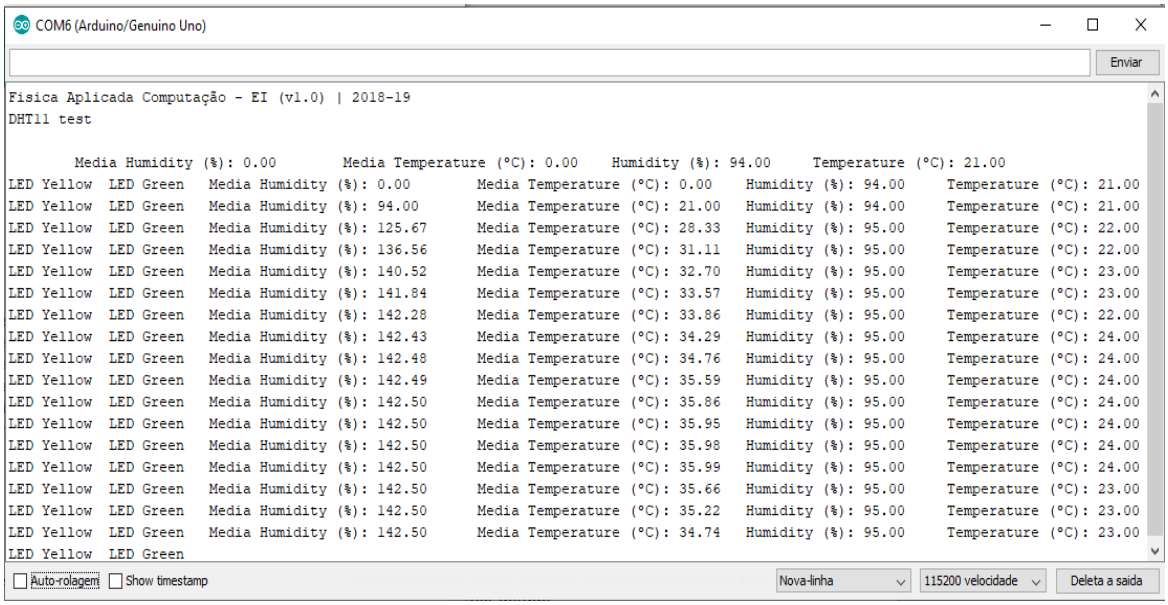


Figura – Serial temperatura e humidade com medias e led's

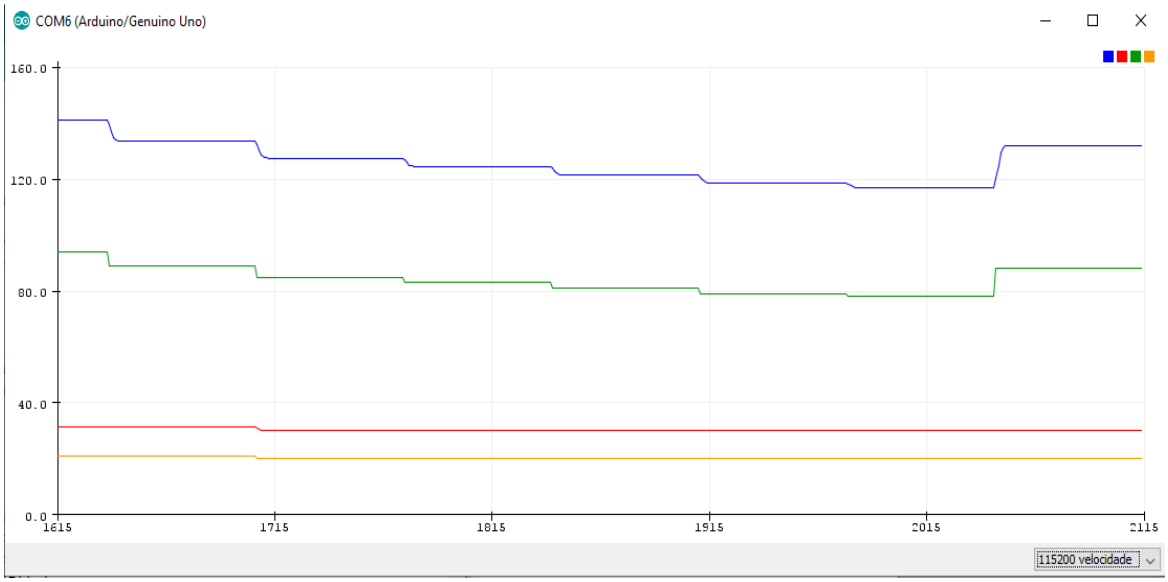


Figura – Plotter temperatura e humidade com medias

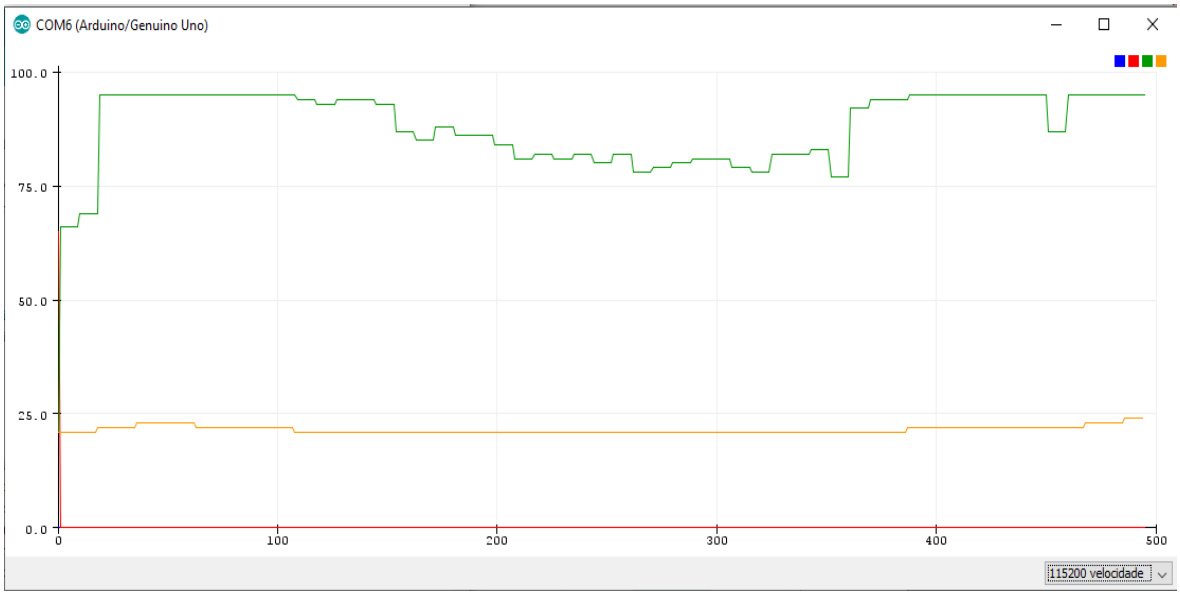


Figura – Plotter temperatura e humidade com base principal

Sensor BMP280

O objetivo das médias a seguir é ir somando todos os valores apresentados ou gerados em temperatura, após resultados das funções, valores

inseridos ou determinados para o ruído por exemplo e depois dividir pelo número de contador de medições realizada.

Sendo assim posso obter os seguintes resultados com a formula, código.

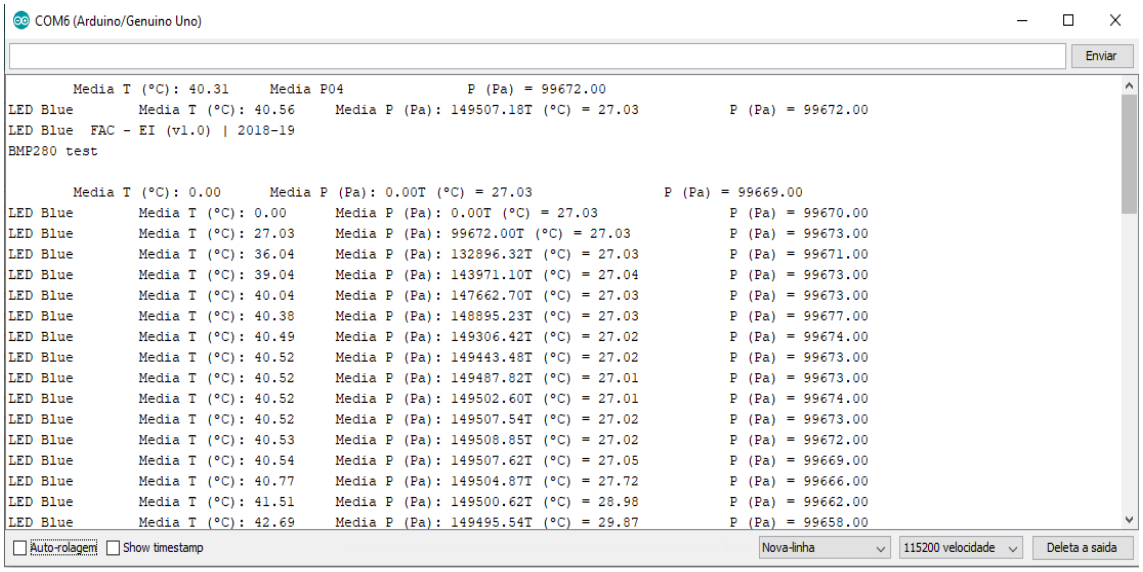


Figura – Serial BMP280 com medias e led's

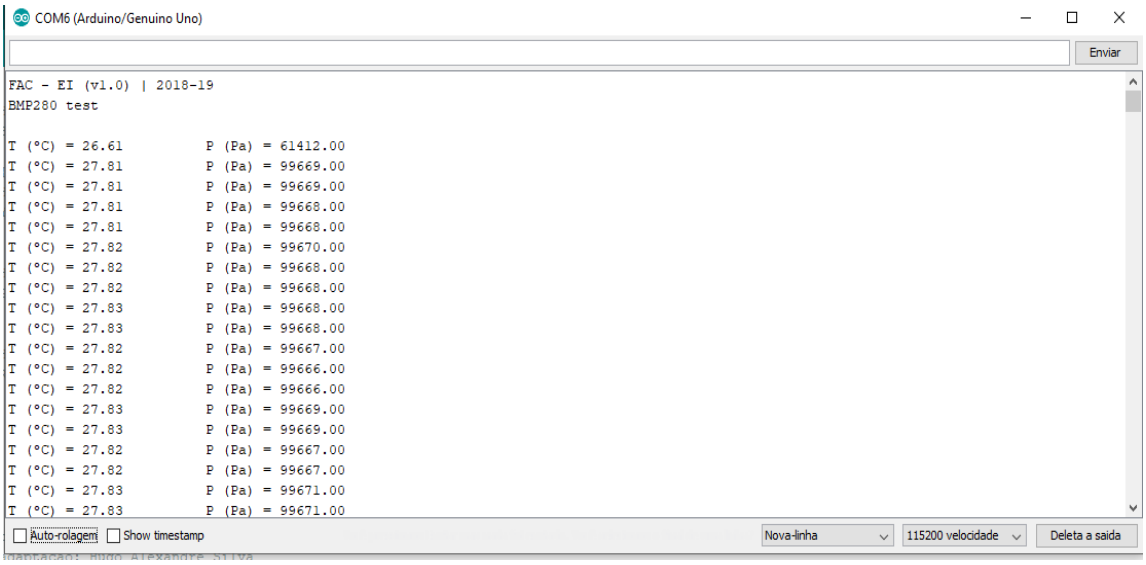


Figura – Serial BMP280 com base principal

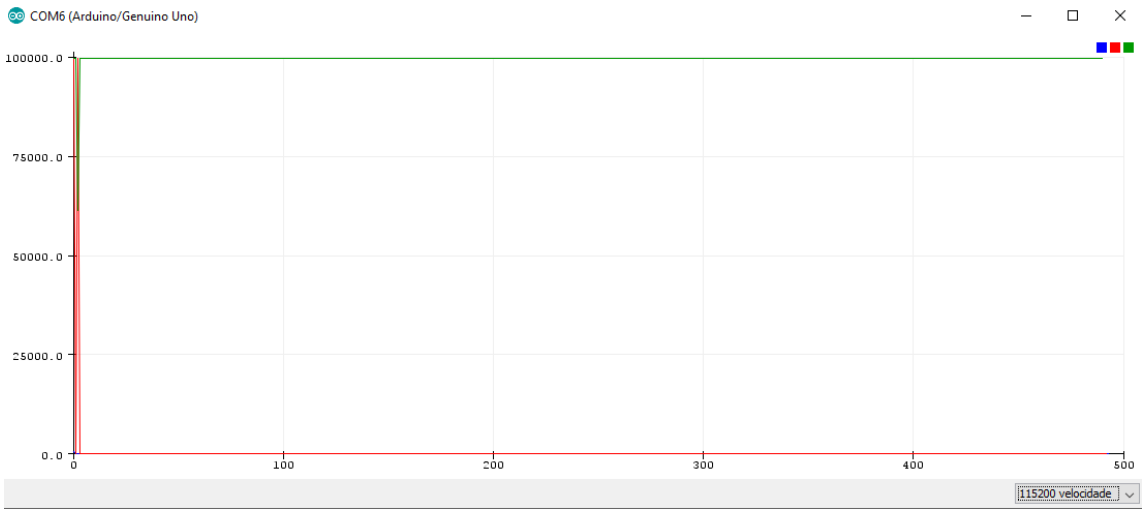


Figura – Plotter BMP280 com medias

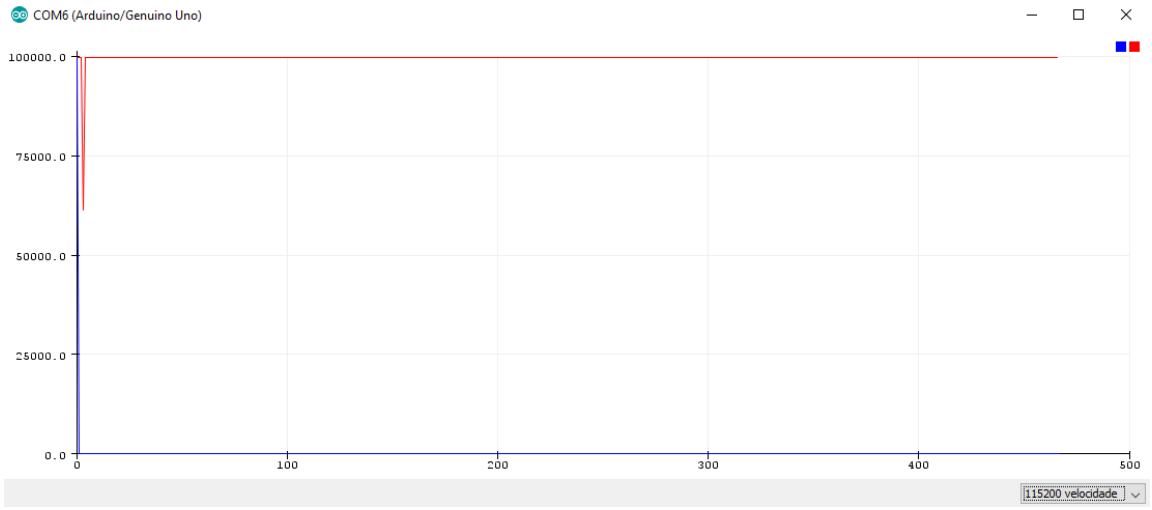


Figura – Plotter BMP280 com base principal

Circuito e esquemático em fritzing

Concluindo a etapa do projeto, elaborar o circuito em fritzing para visualização de como seria o circuito e o esquemático para essa solução.

No modo esquemático, que é um dos modelos abaixo desenvolvido, ele mostrará nosso componente com símbolos oficiais eletrônicos. Posso desenhar em modo Protoboard e, ao clicar na aba Esquemático o software converte automaticamente para os seus símbolos eletrônicos correspondentes.

Necessitando apenas de alguns ajustes de alinhamento e estabelecer a centralização como a verificação de todos os componentes estão devidamente corretos em suas ligações. Também posso fazer a criação do circuito no modo Esquemático, e ao clicar no modo Protoboard, o Fritzing criará o desenho.

Segue abaixo imagem elaborada dos mesmos.

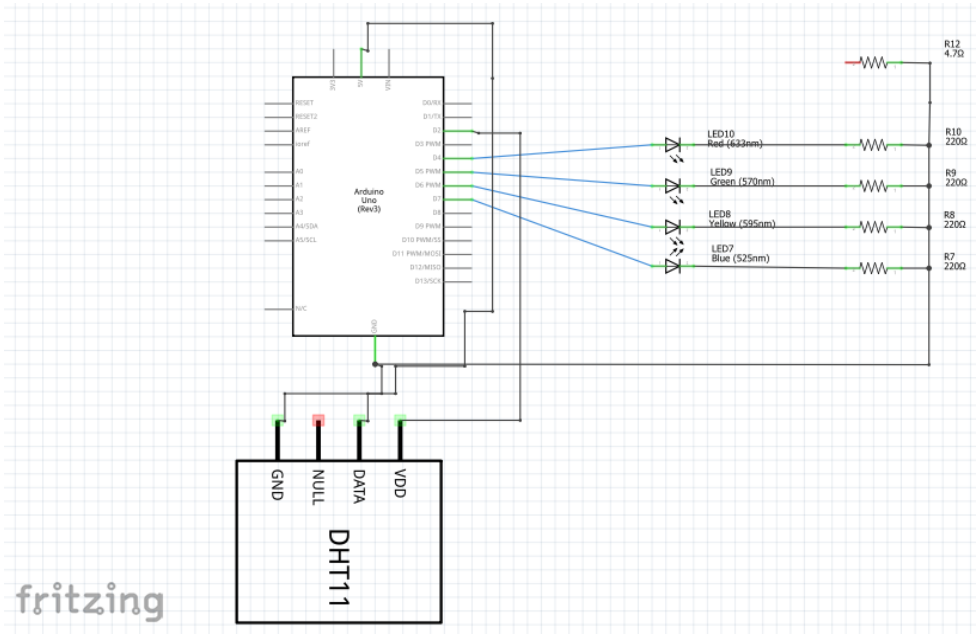


Figura – modelo esquemático DHT11

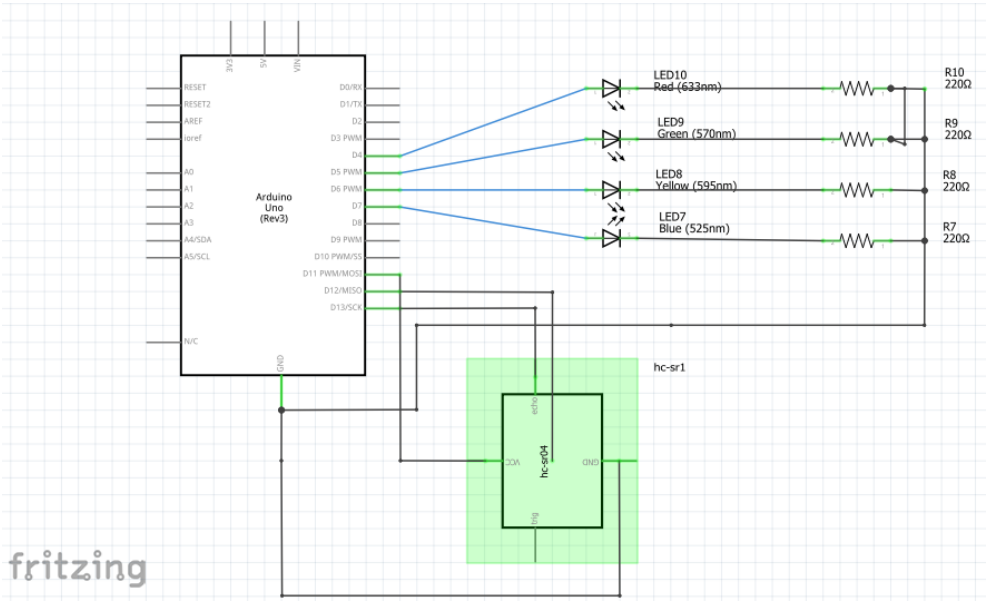


Figura – modelo esquemático HC-SR04

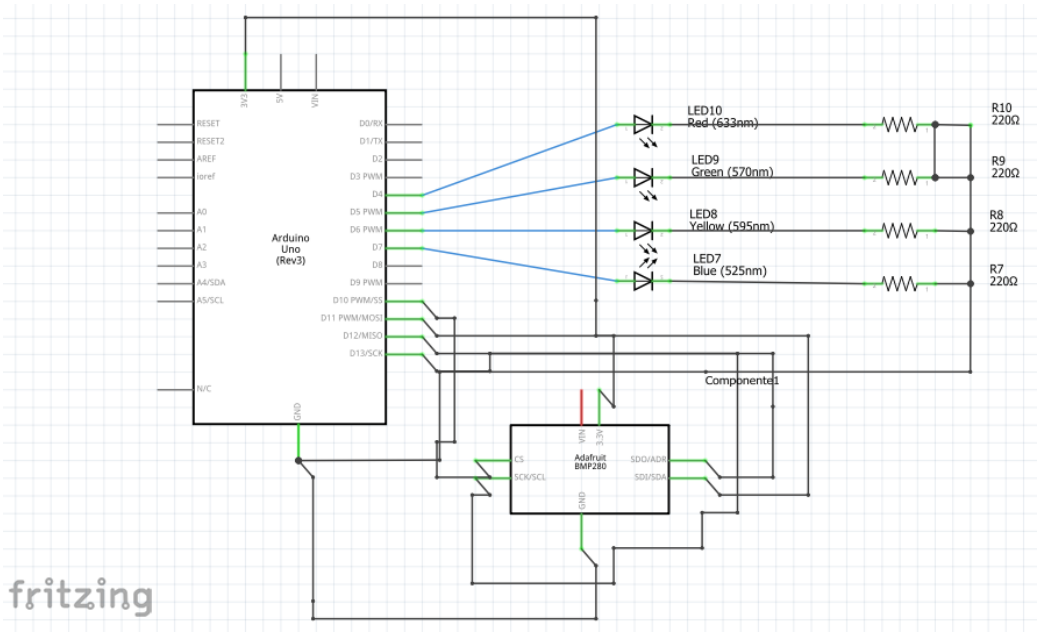


Figura – modelo esquemático BMP280

Também no Fritzing, temos vários modos de desenho pré-elaborados de circuitos de diversas funcionalidade e nesse caso pode ajudar a desenvolver o desenho correto e com seus componentes para utilizadores novos. Também irei demonstrar o modo Protoboard, abaixo, o software nos oferece a possibilidade de desenhar nossos circuitos em uma protoboard e posteriormente aparecer no modo esquemático, dando assim praticidade para o utilizador escolher onde melhor lhe convém efetuar a criação do circuito.

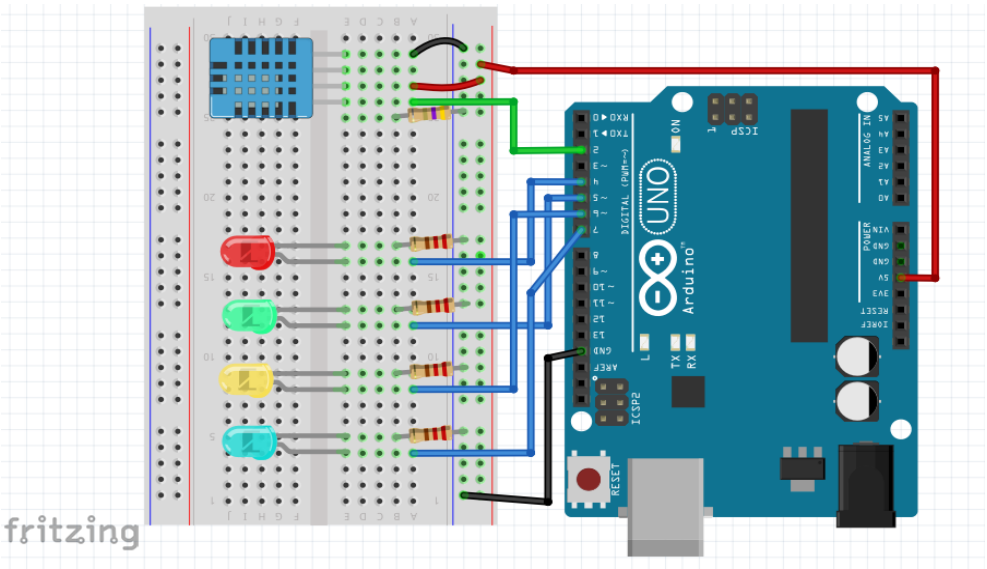


Figura – modelo protoboard DHT11

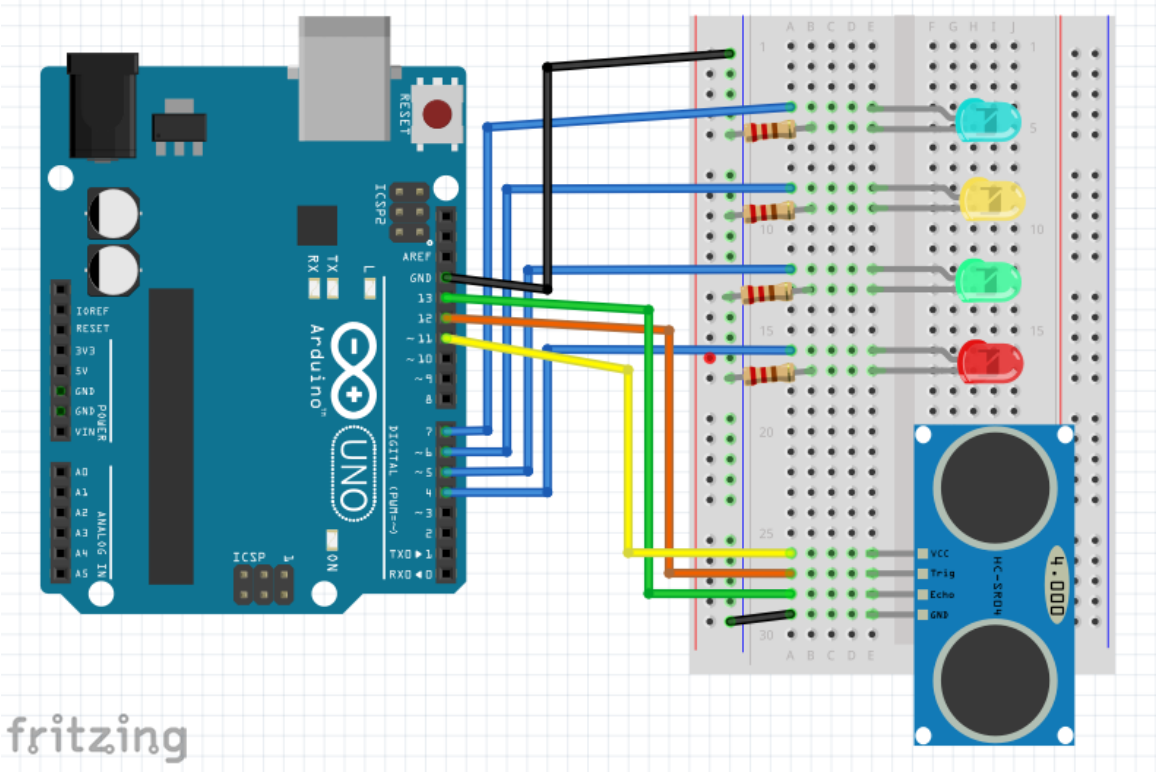


Figura – modelo protoboard HC-SR04 – PROXIMIDADE

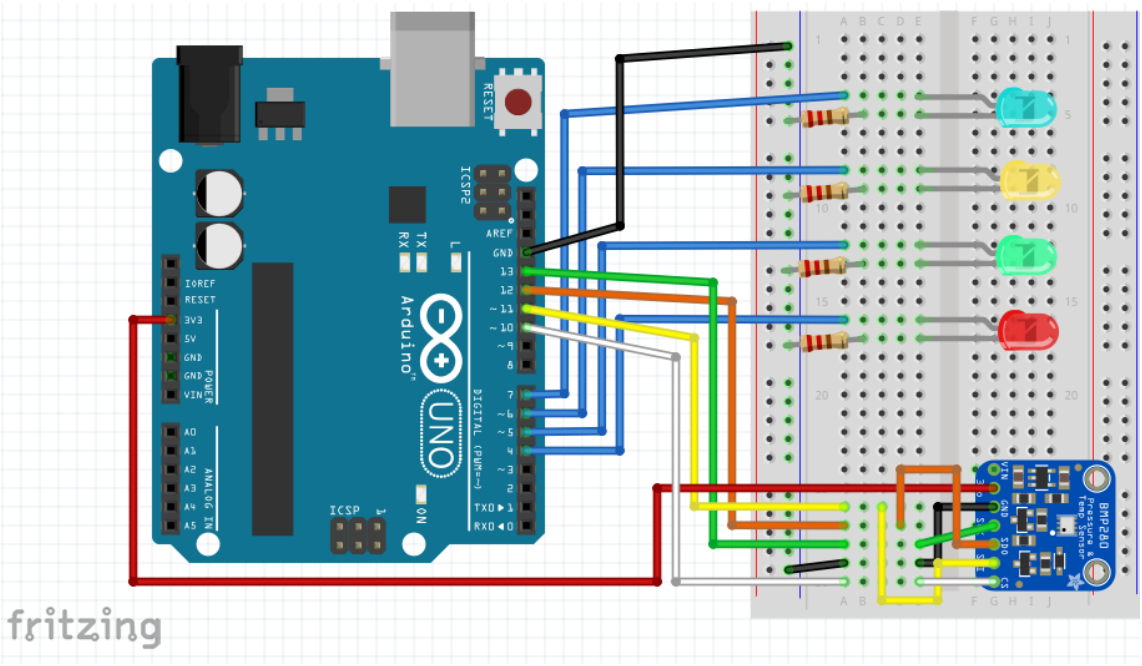


Figura – modelo protoboard BMP28

Gnuplot

Imagens relacionadas em edição no gnuplot, demonstração de gráficos com valores obtidos através do resultado com os sensores.

OBS: colocar estes dois exemplos em anexos, pois o gnuplot não está mais apresentando os gráficos, conforme poderá verificar no trabalho 1 do seno, foram todos feito e impresso, porém neste momento para o trabalho dois todos os gnuplot's aparecem neste formato. Pesquisei possíveis problemas de código, valores, ou manuseio, porém ao que parece o gnuplot apresenta essas falhas de edição em determinadas situações, sem solução prévia. Agradeço a compreensão do mestre em avaliar esta situação de não estar completo esta parte.

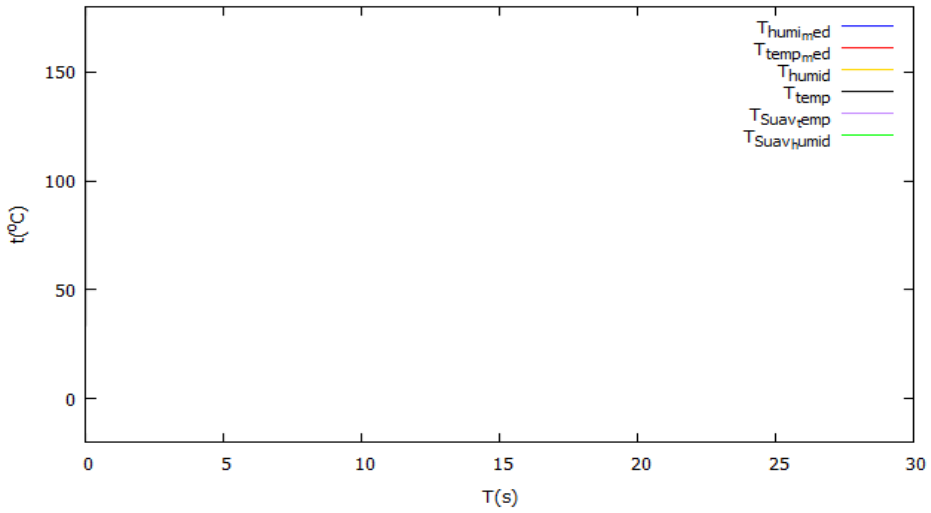


Figura – modelo gnuplot DHT com valores de media, suavização

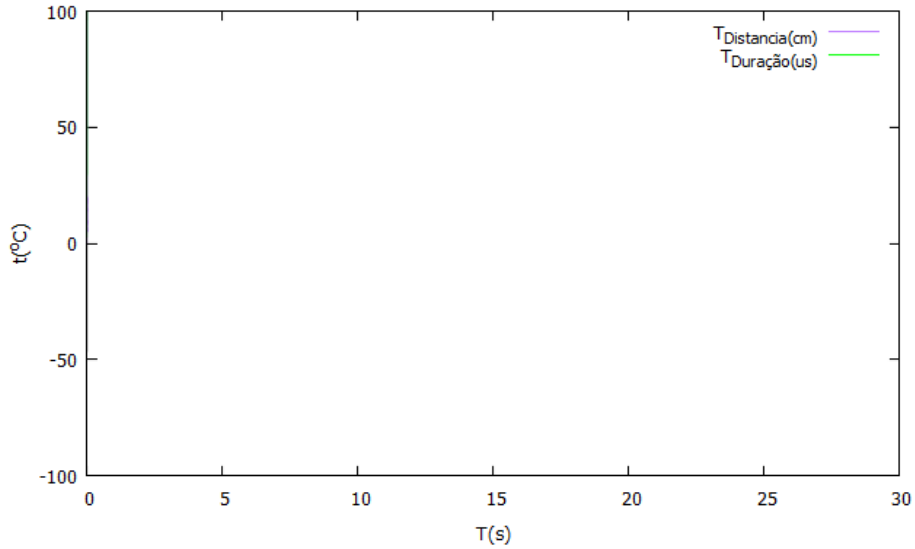


Figura – modelo gnuplot do sensor proximidade, valores de duração e distância

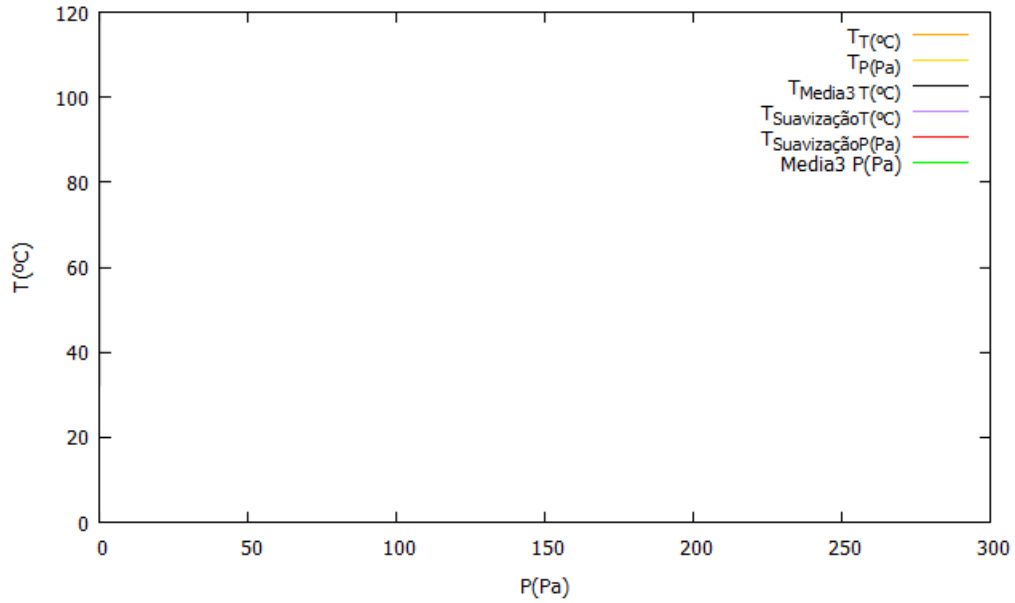


Figura – modelo gnuplot do sensor pressão, valores e medias

Criação dos Circuitos Arduino

Através das imagens abaixo vou demonstrar a montagem feita dos devidos sensores para execução do trabalho e obtenção dos valores de medidas. Todos sensores foram testados e pesquisados inúmeras maneiras de montagem do circuito.

Observe que no Fritzing existe alguns resistores para o funcionamento e segurança do circuito, como estou com falta de materiais e também de sensores o mestre Nuno em laboratório emprestou os sensores para criação do circuito, porém não tinha os led e os resistores, sendo assim o modelo em foto está nesta falta, em relação aos modelos completos do Fritzing. Mesmo assim nas pesquisas de montagem feita, posso usá-los dessa maneira, sem os devidos aparatos onde eles demonstram os valores normalmente e executam suas tarefas com satisfação.

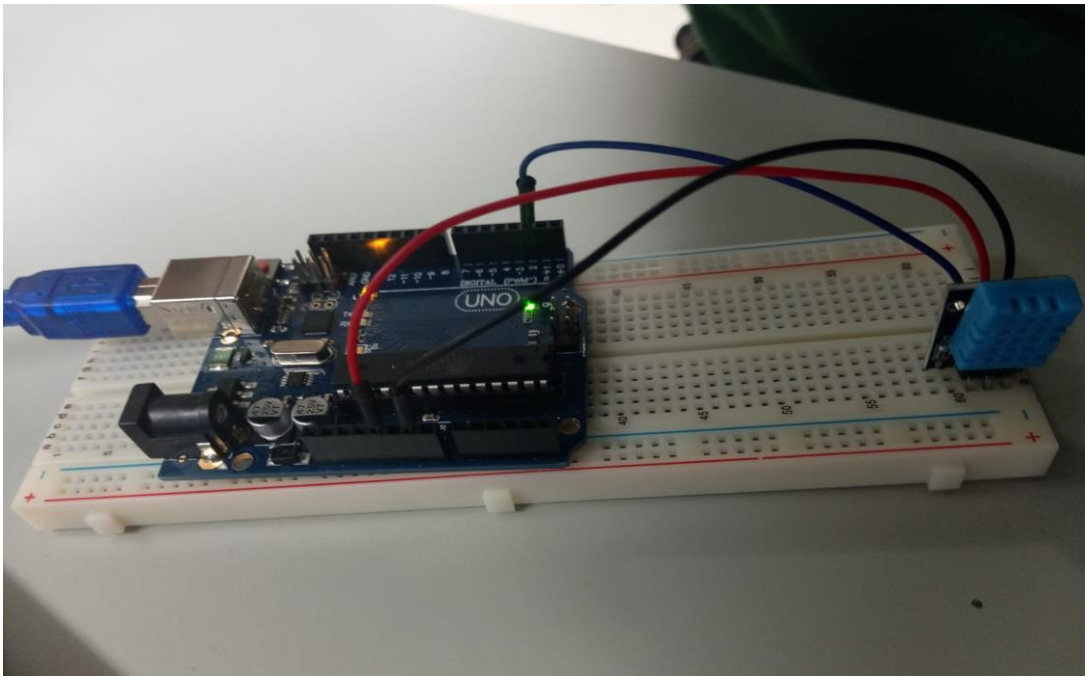


Figura – montagem sensor DHT11

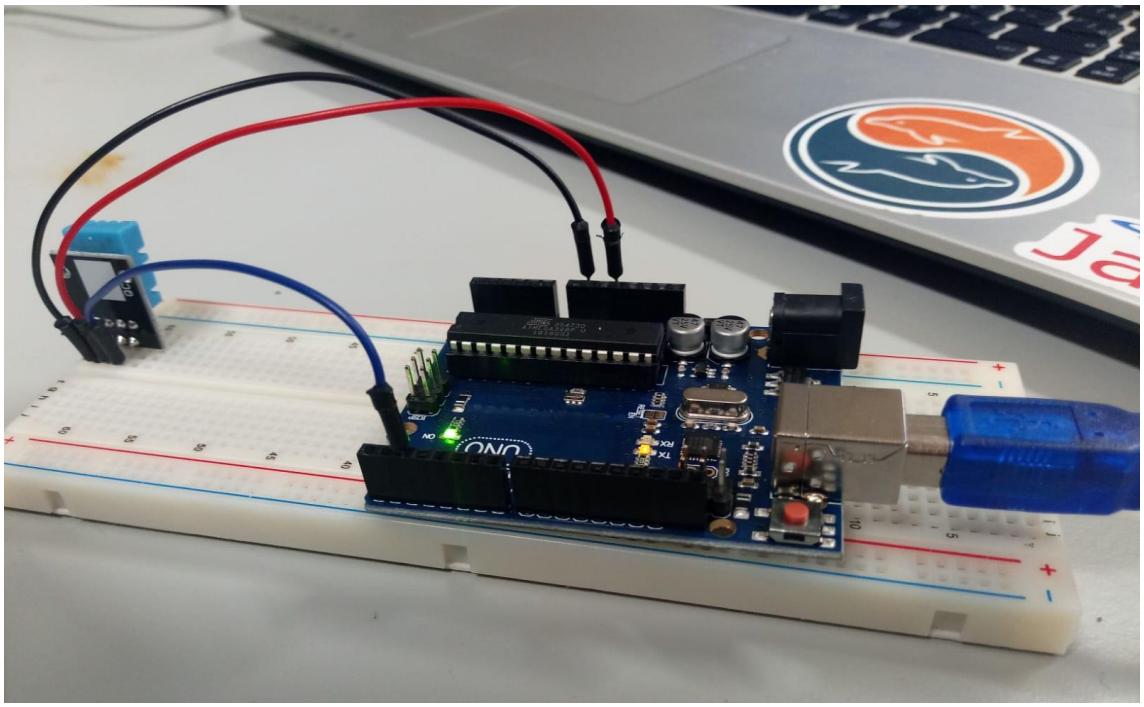


Figura – montagem sensor DHT11

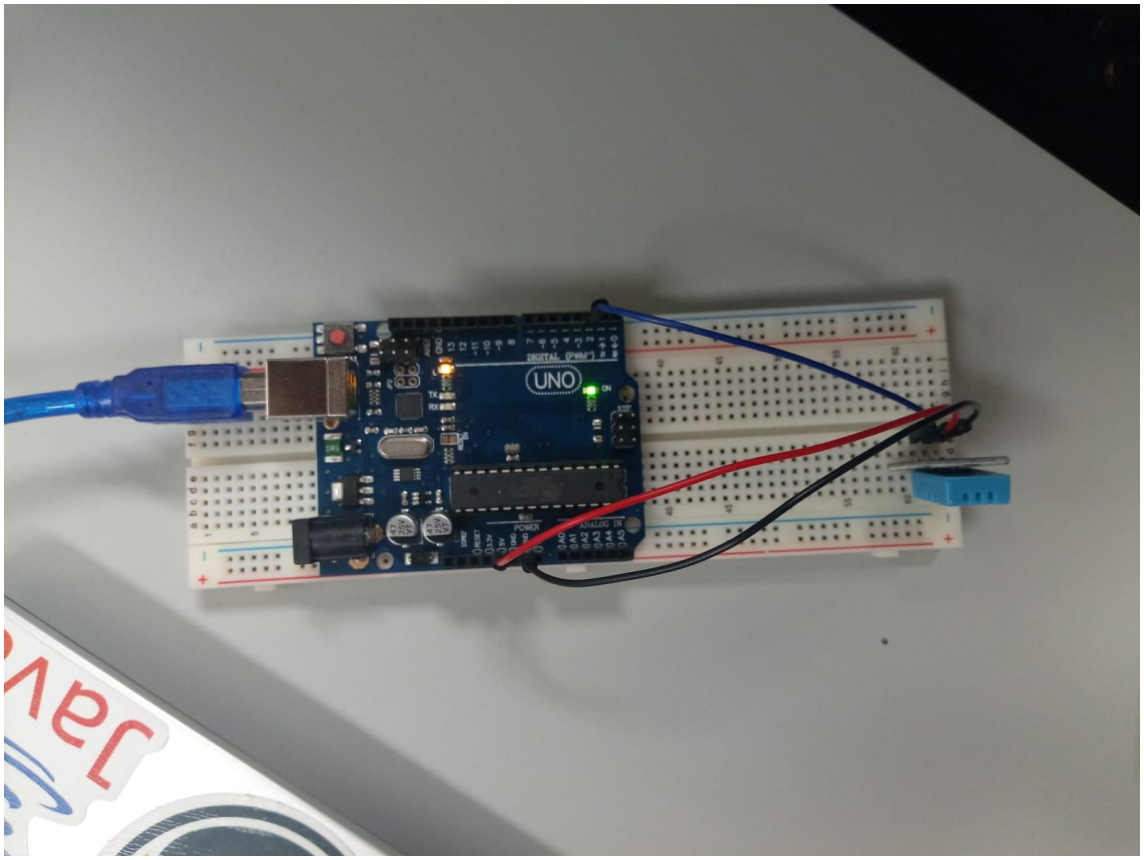


Figura – montagem sensor DHT11

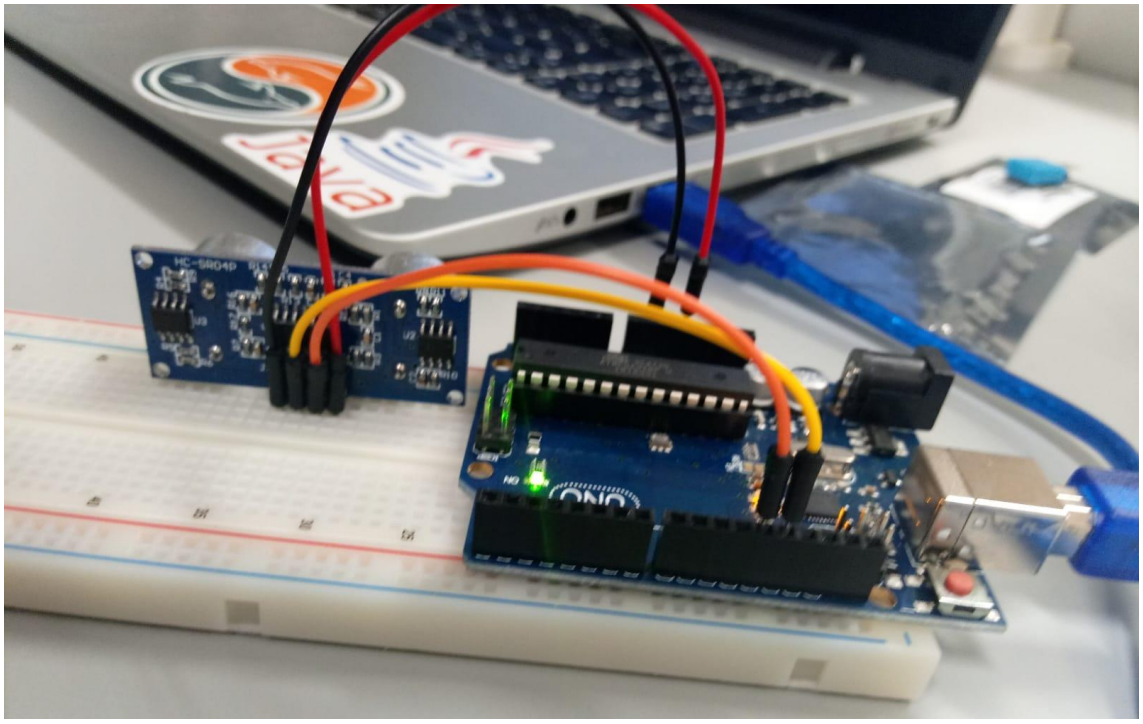


Figura – montagem sensor HC-SR04

Figura – montagem sensor HC-SR04

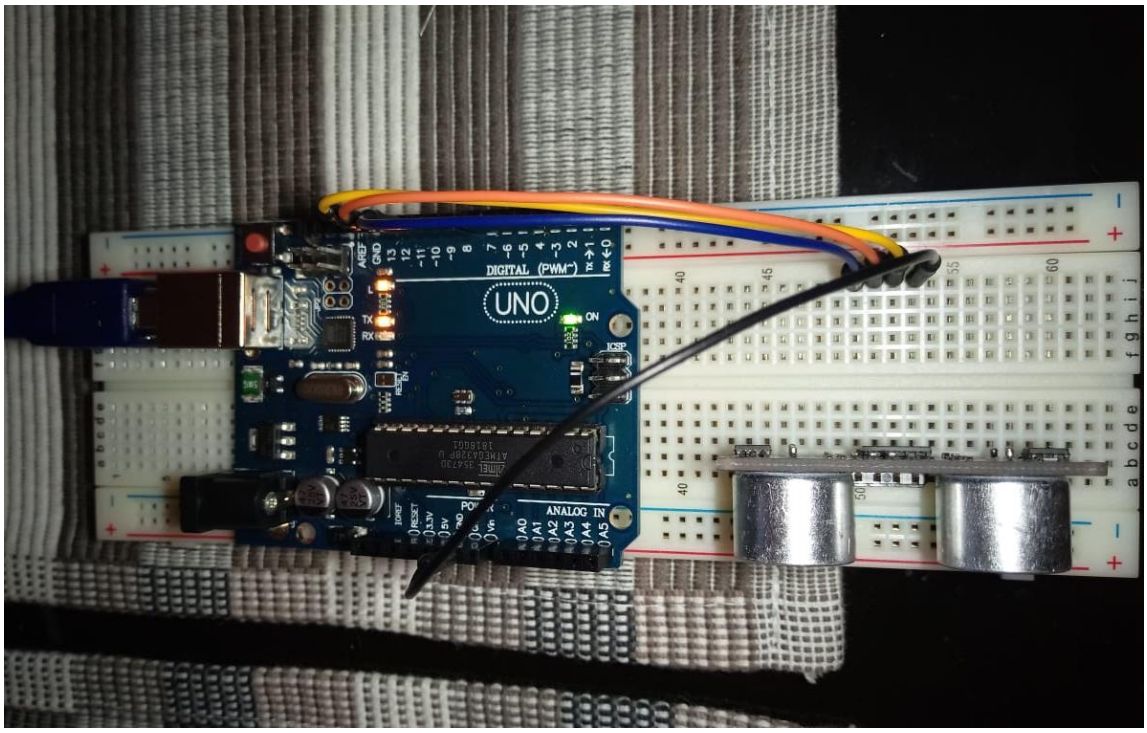


Figura – montagem sensor HC-SR04

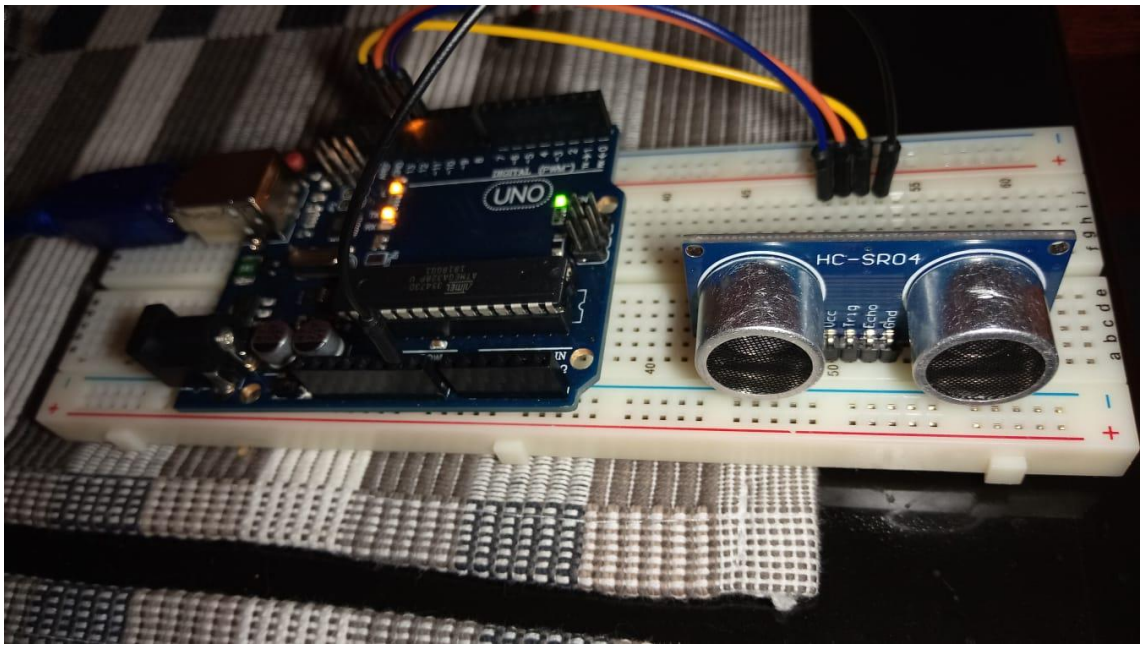


Figura – montagem sensor BMP280

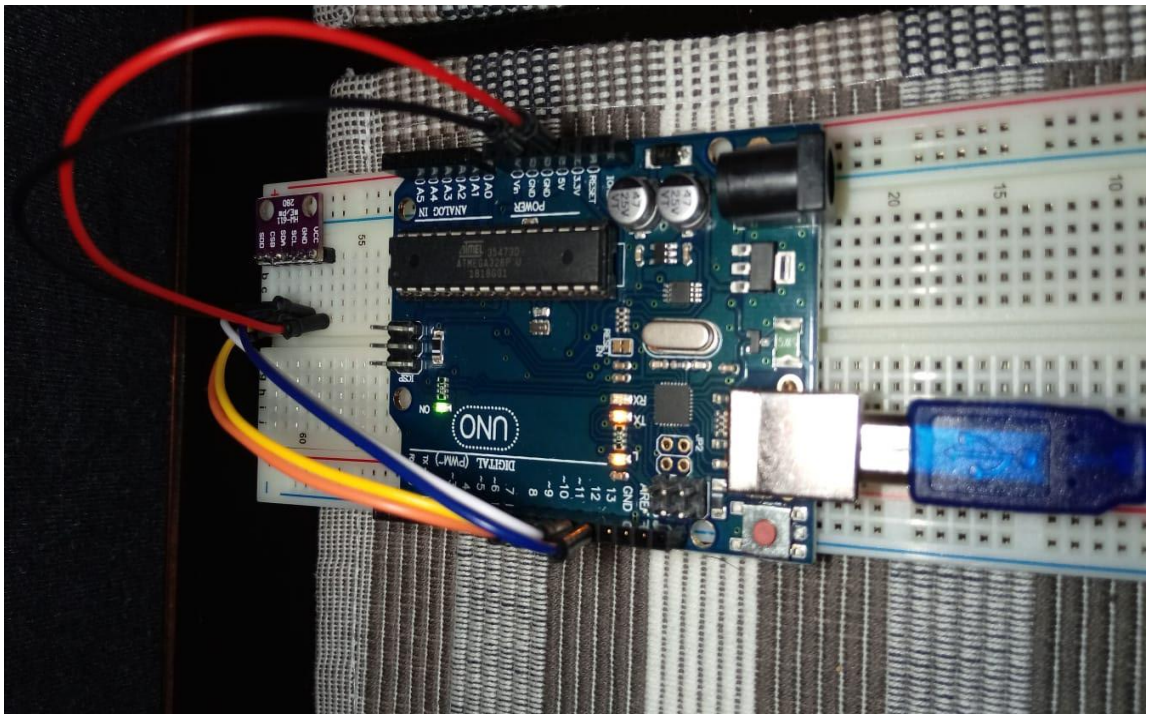


Figura – montagem sensor BMP280

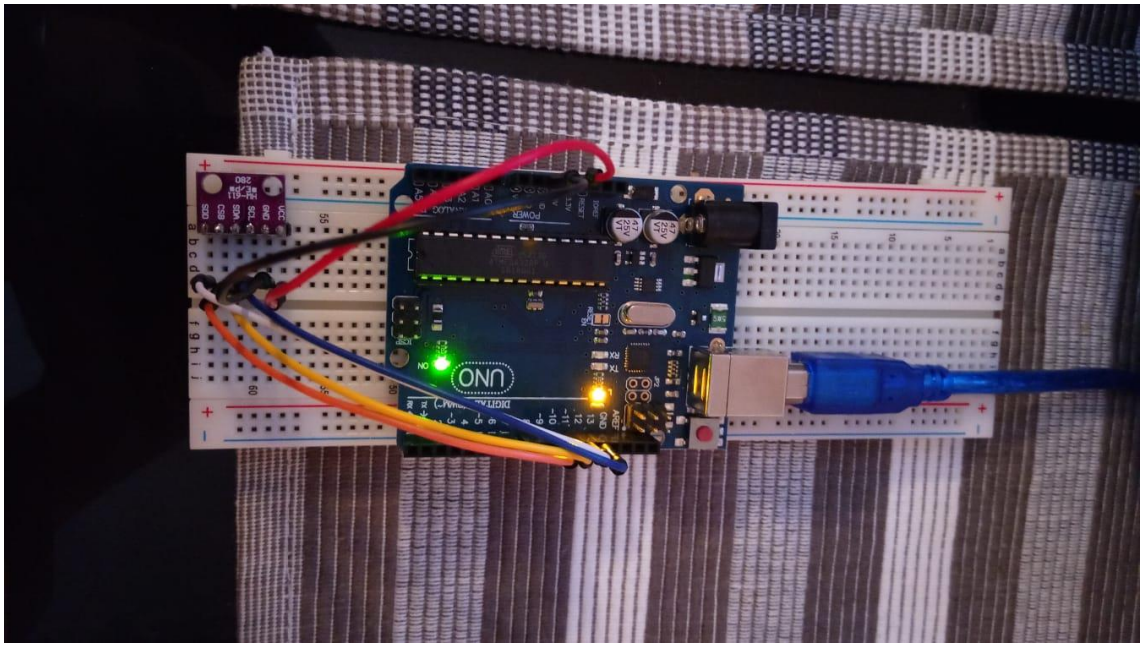
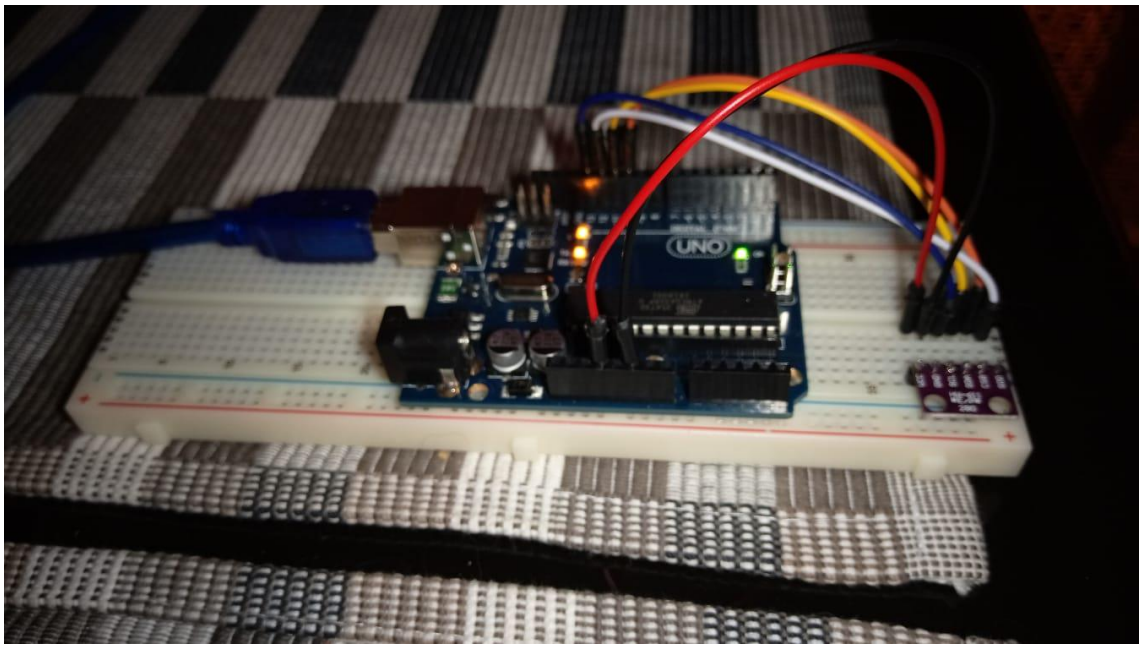


Figura – montagem sensor BMP280



Para uma segunda montagem do sensor bmp280 – devido a possibilidades de uso de bibliotecas entre Windows ou Linux pode se mudar a montagem conforme demonstro em fotos.

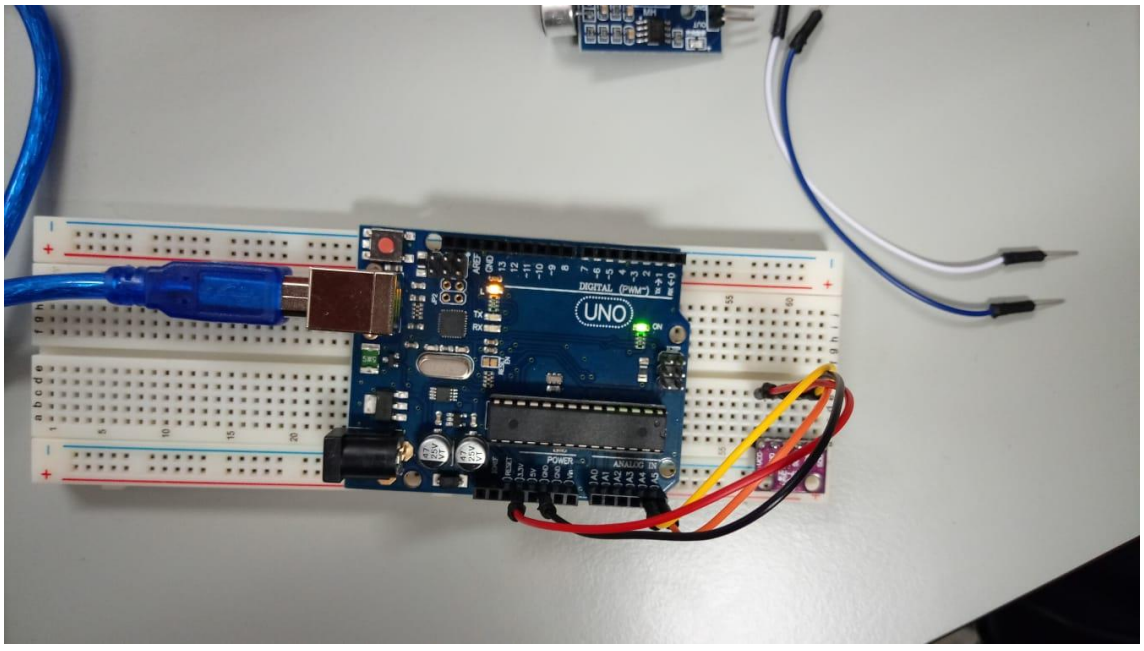


Figura – montagem sensor BMP280

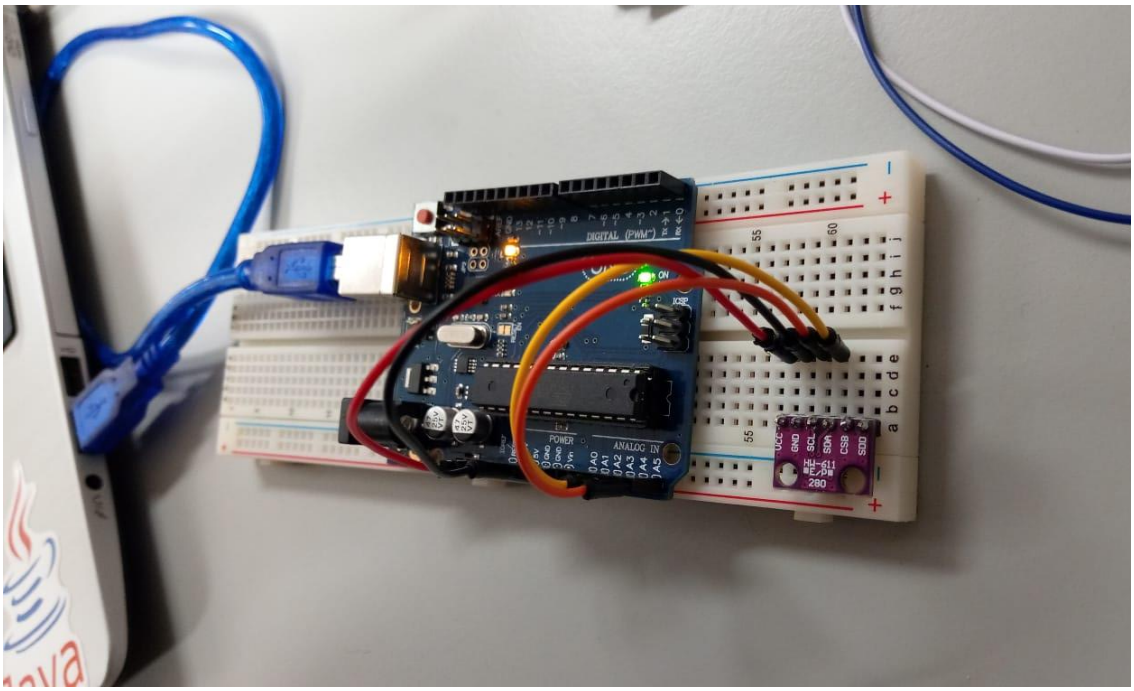


Figura – montagem sensor BMP280

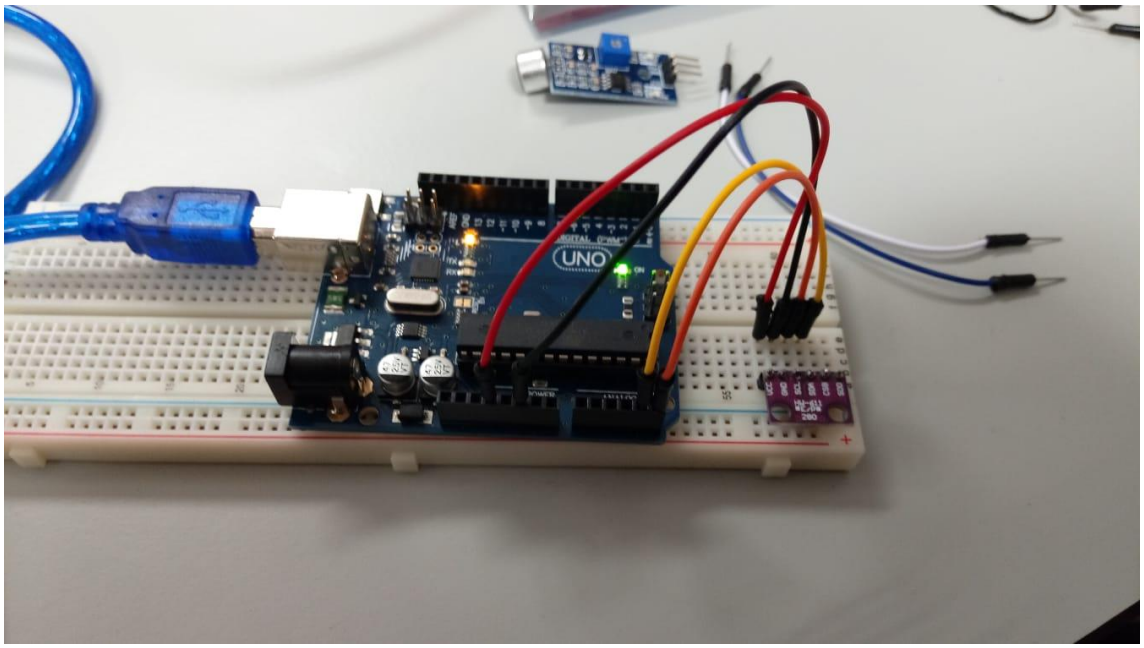


Figura – montagem sensor BMP280

Conclusão

Visando os resultados obtidos concluí tanto a plataforma Arduino em geral e também a parte dos sensores, led e circuito em si do programa, que podem ser usadas em áreas de pesquisas, estudos, aplicação de conhecimentos adquiridos com relevância e relacionadas em pesquisas futuras para um melhor resultado, apesar dos resultados satisfatórios obtidos, sempre se pode continuar melhorando e adequando os códigos de uma maneira mais integra e enxuta assim como ampliar as ações e execução do projeto.

Todos os conceitos estudados nesse projeto podem ser utilizados em outras aplicações e projetos similares, esse estudo utiliza o Arduino, considerado equipamento de baixo custo, torna-se viável e facilmente replicável em projetos futuros.

De uma forma simples e com todos os resultados obtidos, busquei demonstrar em geral todos os passos executados para a elaboração do projeto, assim como o código e suas impressões em serial monitor, plotter, permitindo assim sua reprodução até com certas modificações de valores, especificações e melhorias tanto na parte de programação e elaboração do circuito.

Concluí o presente trabalho com o sentimento de ter alcançado resultados satisfatórios, proposto pelo objetivo e manual de elaboração entregue pelo mestre Nuno Pereira e que não existe limite para a aplicação neste modelo de funcionamento e melhoria de qualquer parte do projeto, incluindo com novos modelos de placa e sensores como o de sequência WIFI que esteja relacionado ao desenvolvimento similar do projeto e que o limite das execuções está somente relacionado ao pensamento e conhecimento presente dos seres humanos, conforme várias vezes dito pelo mestre Nuno Pereira. Conforme estudamos os novos projetos e novos equipamentos e pode-se usar essa plataforma Arduino que é muito importante para o controle de vários outros sensores e equipamentos além de criar o conhecimento facilitado com o acoplamento mecânico, robótico e elétrico neste equipamento chamado ARDUINO.

Referências

https://cms.ipbeja.pt/pluginfile.php/192653/mod_resource/content/6/Trab.Lab01_Arduino%2BSeno_Ed.03_2018-19.pdf

hardware: Arduino Uno R3

programa: Arduino IDE

programa: Fritizing

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

http://forum.arduino.cc/index.php

http://labdegaragem.com/forum/topics

http://playground.arduino.cc//Referencia/Extendida

https://www.arduino.cc/en/Reference/FunctionDeclaration

https://www.arduino.cc/en/Reference/If

https://www.arduino.cc/en/Reference/For

https://www.filipeflop.com/produto/sensor-de-pressao-e-temperatura-bmp280/

https://www.robocore.net

https://img.filipeflop.com/files/download/Datasheet-BMP280-DS001-11.pdf

https://www.arduino.cc/en/Reference/SwitchCase

https://www.arduino.cc/en/Reference/While

https://www.arduino.cc/en/Reference/DoWhile

https://www.arduino.cc/en/Reference/Break

https://www.arduino.cc/en/Reference/Continue

https://www.arduino.cc/en/Reference/Return