Tutorial sensores básicos

- Sensores Estação meteorológica

Sensores Estação meteorológica

Nesta seção, serão utilizados os conceitos de programação e arquitetura de microcontroladores dos módulos anteriores de forma aplicada para poder se conectar e comunicar corretamente com alguns dos principais sensores utilizados na OBSAT. Primeiro, introduziremos a comunicação I2C, uma modalidade muito importante para o uso de múltiplos sensores em uma mesma porta do microcontrolador, e em seguida, serão abordados cada um dos sensores individualmente, mostrando como é feita a conexão correta com o microcontrolador e como funcionam os códigos para obter os dados deles.

Comunicação I2C

A comunicação I2C (Inter-Integrated Circuit) é um protocolo de comunicação usado para interconectar dispositivos eletrônicos dentro de um sistema. O protocolo I2C permite que vários dispositivos compartilhem a mesma linha de comunicação, consistindo em duas linhas principais:

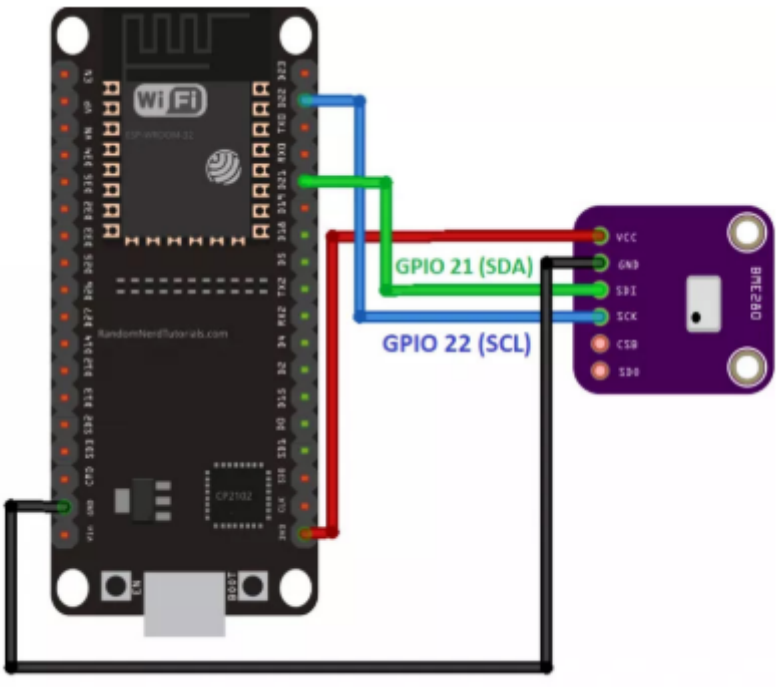
- **SDA (Serial Data Line):** Esta linha é usada para transmitir os dados entre os dispositivos.
- **SCL (Serial Clock Line):** Essa linha é responsável por sincronizar a transferência de dados.

Esse tipo de comunicação com múltiplos sensores nas mesmas portas só é possível pela diferenciação em "endereços" específicos para cada sensor que são definidos no código. Nos exemplos abaixo, mostra-se como iniciar e configurar essa modalidade de comunicação nos códigos e quais sensores a utilizam.

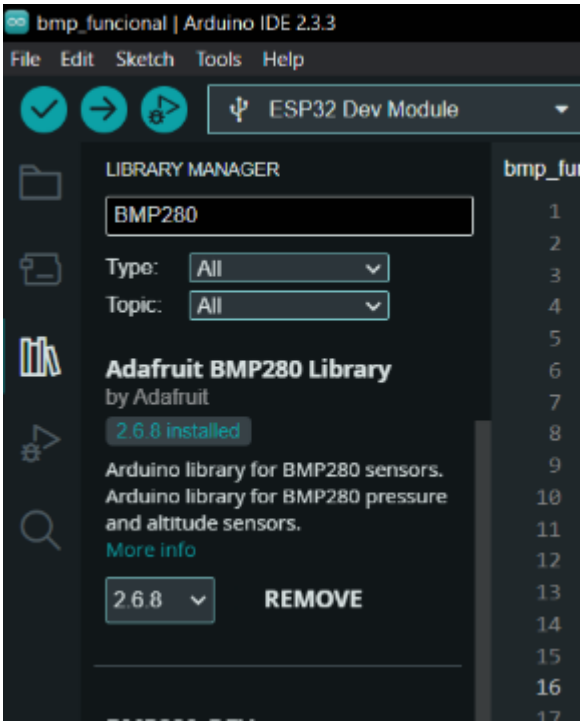
Sensores

BMP280 - Pressão e temperatura

1. Conectar o sensor nas portas de comunicação I2C da seguinte maneira:



2. Baixar a biblioteca “Adafruit BMP280 Library” no menu de bibliotecas da Arduino IDE:



3. Com o circuito conectado corretamente e com a biblioteca baixada, utilizar o código abaixo na IDE:

```
#include <Wire.h>               // Biblioteca para comunicacao I2C
#include <Adafruit_BMP280.h>    // Biblioteca do sensor BMP280
#include <Arduino.h>

#define SDA_PIN 21              // Define o pino SDA
```

```
#define SCL_PIN 22              // Define o pino SCL

Adafruit_BMP280 bmp;           // Cria o objeto do sensor BMP280

#define SEALEVELPRESSURE_HPA (1013.25)   //Defini a pressao para nivel do mar

void setup() {
  Serial.begin(9600);            // Inicializa a comunicacao serial

  // Inicializa a comunicacao I2C com os pinos definidos
  Wire.begin(SDA_PIN, SCL_PIN);

  // Inicializa o BMP280 com o endereco I2C padrao (0x76 ou 0x77)
  if (!bmp.begin(0x76)) {
    Serial.println("Nao foi possivel inicializar o sensor BMP280!");
    while (1);
  }

  // Configura o sensor BMP280
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,  // Modo de medicao normal
                  Adafruit_BMP280::SAMPLING_X2,  // Oversampling da temperatura
                  Adafruit_BMP280::SAMPLING_X16, // Oversampling da pressao
                  Adafruit_BMP280::FILTER_X16,   // Filtro
                  Adafruit_BMP280::STANDBY_MS_500); // Tempo de espera em modo standby
}

void loop() {
  // Le a temperatura e pressao
  float temperature = bmp.readTemperature();
  float pressure = bmp.readPressure() / 100.0F; // converte para hPa
  float altitude = bmp.readAltitude(SEALEVELPRESSURE_HPA); // calcula altitude aproximada pela pressao

  // Exibe os valores no monitor serial
  Serial.print("Temperatura: ");
  Serial.print(temperature);
  Serial.print(" C, Pressao: ");
  Serial.print(pressure);
  Serial.print(" hPa, Altitude: ");
  Serial.print(altitude);
  Serial.println(" m");
```

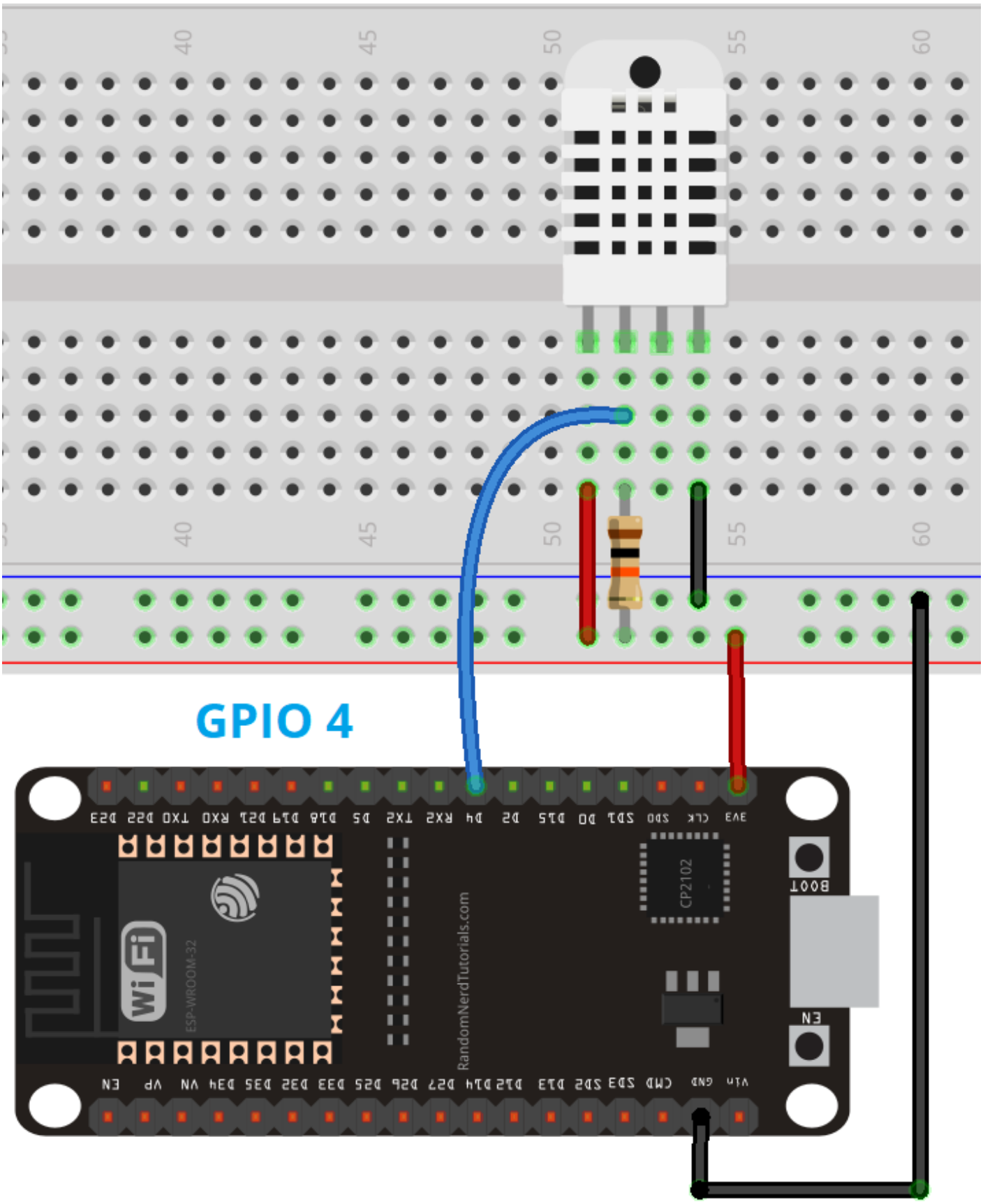
```
  delay(2000); // Aguarda 2 segundos entre as medicoes
}
```

4. Clicar no ícone de setinha para a direita no topo da IDE e o programa será carregado para ESP32. (Caso a ESP32 esteja com o MicroPython instalado, esse processo o removerá e mudará a placa para C++ automaticamente.)
5. Para receber as informações, basta abrir o monitor serial com a velocidade definida no código; nesse caso, é 9600.

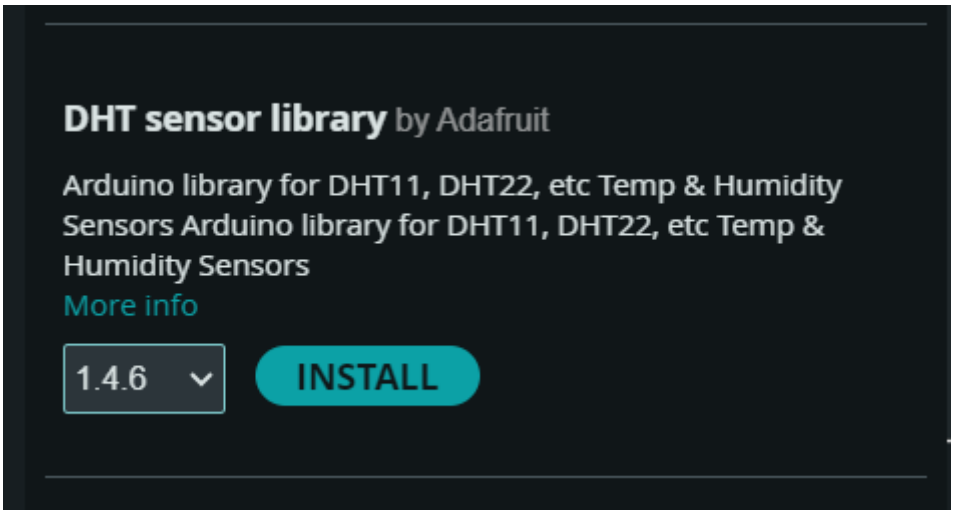
Referência: https://www.instructables.com/How-to-Connect-BMP-280-to-ESP32-Get-Pressure-Tempe/

DHT11 - Umidade e temperatura

1. Conectar o sensor DHT11 de acordo com a figura abaixo em uma porta digital, no exemplo foi utilizada a porta GPIO4.



2. Com o sensor corretamente conectado, baixar a biblioteca " DHT sernsor library " by Adafruit e todas as suas dependências que a IDE sugerir.



3. Carregue na placa o código abaixo e abra o monitor serial para ver os resultados.

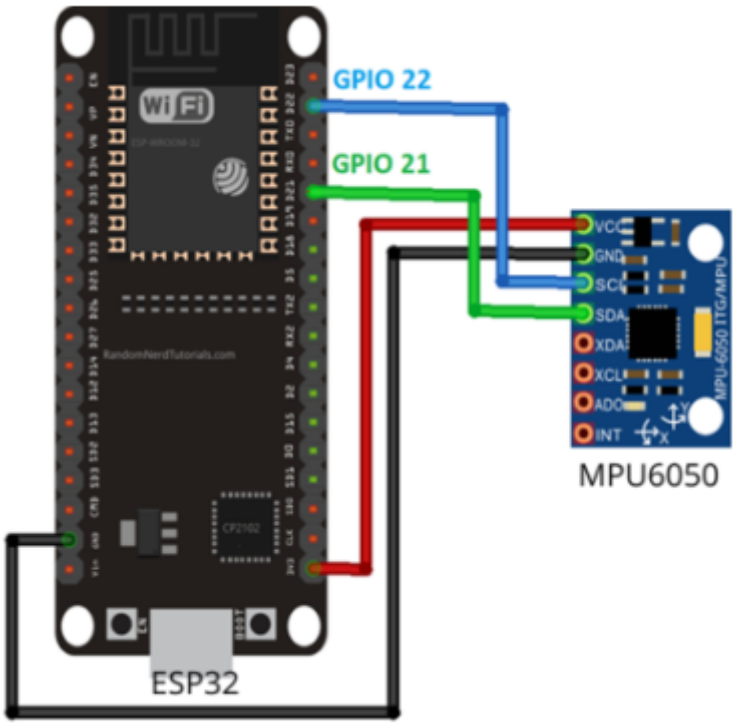
```
#include "DHT.h"

#define DHTPIN 4         // Define qual pino esta sendo usado
#define DHTTYPE DHT11    // Define o tipo de DHT (DHT11 ou DHT22)

DHT dht(DHTPIN, DHTTYPE);  // configura o sensor para as opcoes definidas acima

void setup() {
  Serial.begin(9600);

  dht.begin();            // inicializa o sensor
}

void loop() {
  // Espera alguns segundos entre as leituras.
  delay(2000);

  // Leitura dos dados do sensor
  float h = dht.readHumidity();             // Le o valor de humidade
  float t = dht.readTemperature();          // Le o valor de temperatura em Celcius (padrao)
  float f = dht.readTemperature(true);      // Le a temperatura em Fahrenheit

  // Checa se alguma das leituras falhou e sai do loop mais cedo para tentar novamente
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
```

```
  // Calcula sensacao termica em Fahrenheit (por padrao)
  float hif = dht.computeHeatIndex(f, h);
  // Calcula sensacao termica em Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);



  // Imprime os valores lidos no monitor serial
  Serial.print(F("Humidade: "));
  Serial.print(h);
  Serial.print(F("%,  Temperatura: "));
  Serial.print(t);
  Serial.print(F(" Celcius "));
  Serial.print(f);
  Serial.print(F(" Fahreheit,  Sensacao termica: "));
  Serial.print(hic);
  Serial.print(F(" Celcius "));
  Serial.print(hif);
  Serial.println(F(" Fahreheit"));
}
```

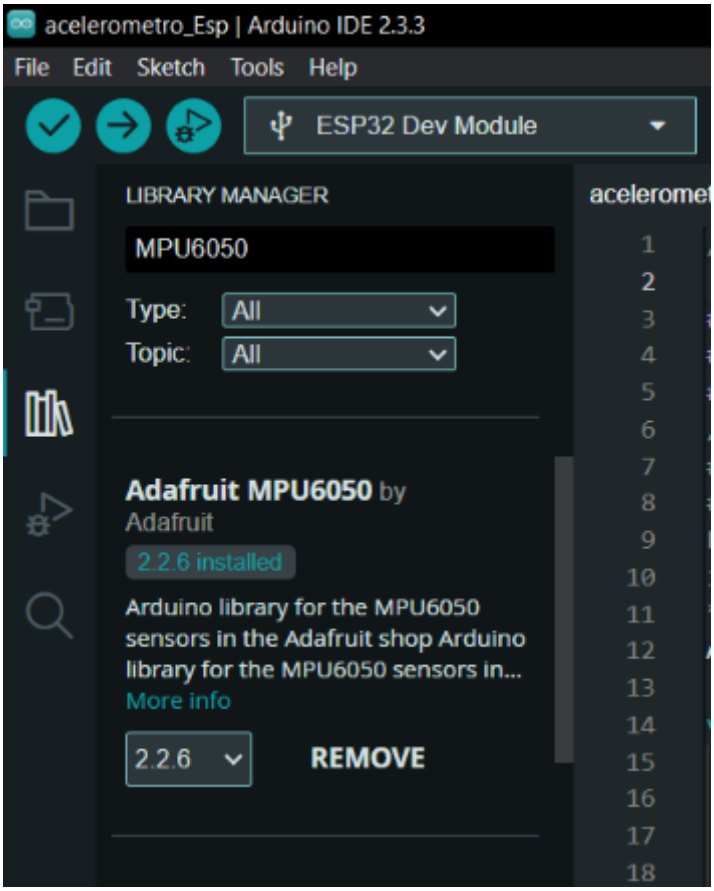
Referência: https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/

MPU6050 - Giroscópio e acelerômetro

1. Conectar o MPU6050 da forma representada na figura abaixo nas portas I2C:



2. Baixar a biblioteca “Adafruit MPU6050” e todas as suas dependências que a IDE sugeri:



3. Com a biblioteca instalada, basta utilizar o código a seguir:

```
// Codigo basico de leitura da Adafruit MPU6050

// Bibliotecas utilizadas
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

void setup(void) {
  Serial.begin(115200); // Definindo velocidade do monitor serial
  while (!Serial)
    delay(10); // Pausa a placa para esperar o console abrir

  Serial.println("Adafruit MPU6050 test!");

// Teste de conexao com o sensor
  if (!mpu.begin()) {
    Serial.println("Falha ao encontrar sensor MPU6050");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Encontrado!");

// Definindo amplitude do acelerometro
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
  case MPU6050_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case MPU6050_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case MPU6050_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case MPU6050_RANGE_16_G:
    Serial.println("+-16G");
```

```
    break;
  }

// Definindo amplitude do giroscopio
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  Serial.print("Gyro range set to: ");
  switch (mpu.getGyroRange()) {
  case MPU6050_RANGE_250_DEG:
    Serial.println("+- 250 deg/s");
    break;
  case MPU6050_RANGE_500_DEG:
    Serial.println("+- 500 deg/s");
    break;
  case MPU6050_RANGE_1000_DEG:
    Serial.println("+- 1000 deg/s");
    break;
  case MPU6050_RANGE_2000_DEG:
    Serial.println("+- 2000 deg/s");
    break;
  }

// Definindo filtro de largura de banda
  mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
  Serial.print("Filter bandwidth set to: ");
  switch (mpu.getFilterBandwidth()) {
  case MPU6050_BAND_260_HZ:
    Serial.println("260 Hz");
    break;
  case MPU6050_BAND_184_HZ:
    Serial.println("184 Hz");
    break;
  case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
  case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
  case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
```

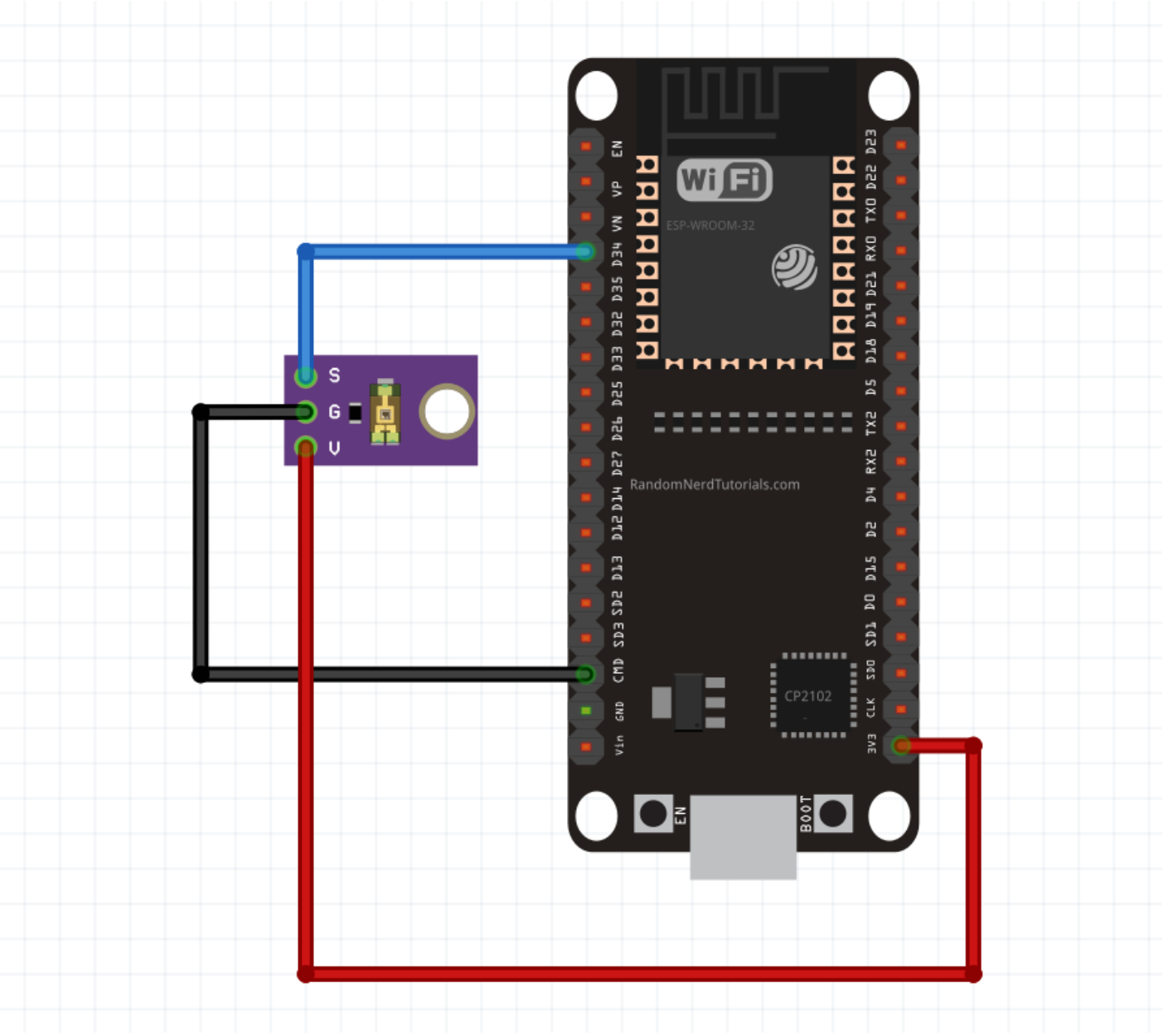
```
  case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
  case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
  }

  Serial.println("");
  delay(300);
}

void loop() {
  /* Get new sensor events with the readings */
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  /* Imprimindo valores */
  Serial.print("Acceleration X: ");
  Serial.print(a.acceleration.x);
  Serial.print(", Y: ");
  Serial.print(a.acceleration.y);
  Serial.print(", Z: ");
  Serial.print(a.acceleration.z);
  Serial.println(" m/s^2");

  Serial.print("Rotation X: ");
  Serial.print(g.gyro.x);
  Serial.print(", Y: ");
  Serial.print(g.gyro.y);
  Serial.print(", Z: ");
  Serial.print(g.gyro.z);
  Serial.println(" rad/s");

  Serial.println("");
  delay(750);
}
```

4. Conectar a placa ao computador na porta COM selecionada
5. Compilar o código para a placa ESP32;

6. Abrir o monitor serial para a configuração definida no código.

TEMT6000 - Luminosidade

1. Conectar o sensor TEMT6000 da forma representada na figura abaixo em alguma porta analógica, no exemplo abaixo utilizou-se a porta 34.



2. Com o sensor conectado corretamente, não é necessário instalar nenhuma biblioteca adicional, basta utilizar o código abaixo e as leituras devem aparecer no monitor serial.

```
#define TEMT6000 34 //PINO conectado no TEMT6000


void setup()
{
```

```
  // PIN MODE
  pinMode(TEMT6000, INPUT);

  Serial.begin(9600);
}

void loop() {

  // Leitura de luminosidade - TEMT6000
    analogReadResolution(10); // define resolucao da leitura na porta analogica

    float volts =  analogRead(TEMT6000) * 5 / 1024.0; // Converte leitura analogica para VOLTS
    float VoltPercent = analogRead(TEMT6000) / 1024.0 * 100; //Leitura em porcentagem da tensao

    //converte a leitura em LUX
    float amps = volts / 10000.0;  // converte para ampere considereando resitencia de 10,000 Ohms
    float microamps = amps * 1000000; // Converte para Microampere
    float lux = microamps * 2.0; // Converte para lux
    delay(1000);

  // Output no monitor Serial
  Serial.print("LUX - ");
  Serial.print(lux);
  Serial.println(" lx");
  Serial.print(VoltPercent);
  Serial.println("%");
  Serial.print(volts);
  Serial.println(" volts");
  Serial.print(amps);
  Serial.println(" amps");
  Serial.print(microamps);
  Serial.println(" microamps");
  delay(1000);
}
```

Referência: https://github.com/CraftzAdmin/esp32/blob/main/Sensors/TEMT6000/README.md

Projeto final

Agora que você sabe como funciona a arquitetura de um microcontrolador, como funciona a programação em C++ e como utilizar cada um dos sensores acima, teste seus conhecimentos montando a estação meteorológica completa. A montagem da estação meteorológica completa consiste em conectar todos os sensores simultaneamente às mesmas portas utilizadas nos exemplos apresentados na seção anterior e criar um código que consiga imprimir no monitor serial todos os dados.

É importante destacar que os sensores BMP280 e MPU6050 compartilham as mesmas portas de comunicação. Contudo, isso não representa um problema, pois o protocolo de comunicação I2C permite a utilização de múltiplos dispositivos na mesma linha de dados. No código, os endereços específicos de cada sensor são utilizados para diferenciar e realizar a leitura simultânea de ambos.