Curso de Introdução à Programação de Microcontroladores

Introdução

Objetivos

Esta apostila tem como objetivo principal oferecer uma base sólida para os leitores sobre o universo dos microcontroladores, com ênfase nos dispositivos ESP32 e Arduino. Os objetivos específicos incluem:

- **Apresentar os conceitos de microcontroladores:**
  Fornecer uma visão geral sobre o funcionamento, a estrutura e as aplicações dos microcontroladores no mundo real.
- **Ensinar o uso de plataformas de programação e simulação:**
  Introduzir ferramentas populares como IDEs (Ambientes de Desenvolvimento Integrado) e simuladores que auxiliam no desenvolvimento e teste de projetos com microcontroladores.
- **Ensinar conceitos básicos de programação de microcontroladores:**
  Abordar a escrita de códigos simples e eficientes para controlar diferentes periféricos e executar tarefas específicas.
- **Demonstrar o uso de diversos tipos de sensores:**
  Capacitar o leitor a integrar sensores de temperatura, umidade, pressão, entre outros, para coletar dados ambientais e processá-los em projetos práticos.
- **Permitir a criação de uma estação meteorológica:**
  Ensinar como construir uma estação meteorológica funcional utilizando microcontroladores, sensores e técnicas aprendidas ao longo da apostila.

Com essa abordagem, espera-se que o leitor desenvolva as competências necessárias para criar projetos práticos e criativos utilizando os microcontroladores ESP32 e Arduino, desde a concepção até a execução.

Introdução a Microcontroladores

Introdução a Microcontroladores

O que são microcontroladores?

Os microcontroladores são dispositivos eletrônicos altamente integrados que combinam em um único chip um processador (CPU), memórias (RAM e ROM/Flash), periféricos de entrada e saída (I/O) e outros componentes necessários para controlar sistemas embarcados. Eles são projetados para realizar tarefas específicas em dispositivos eletrônicos, como controlar motores, sensores, displays e outros elementos de um sistema.

Devido à sua simplicidade e eficiência, os microcontroladores são amplamente utilizados em aplicações como eletrodomésticos, automação industrial, dispositivos médicos, sistemas automotivos, brinquedos eletrônicos, entre outros. Sua principal característica é a capacidade de operar de forma autônoma, sem a necessidade de componentes adicionais complexos.

Diferença entre Microcontroladores e Microprocessadores

Embora os termos "microcontrolador" e "microprocessador" sejam frequentemente confundidos, eles se referem a componentes distintos com finalidades diferentes:

Microprocessadores: Um microprocessador é essencialmente a unidade central de processamento (CPU) de um sistema. Ele é responsável por executar instruções e processar dados, mas depende de componentes externos, como memórias (RAM e ROM), controladores de entrada e saída e outros periféricos, para funcionar.

São comumente usados em sistemas de maior complexidade, como computadores pessoais, laptops e servidores, onde a capacidade de processamento e expansibilidade é essencial.

Microcontroladores: Um microcontrolador, por outro lado, é uma solução "tudo-em-um", que inclui a CPU, memórias, e periféricos integrados no mesmo chip. Essa integração permite o desenvolvimento de sistemas compactos, de baixo custo e baixa potência.

São ideais para aplicações embarcadas, onde o foco é realizar tarefas específicas com eficiência e simplicidade.

Exemplo Prático: Considere um termostato inteligente usado para controlar a temperatura de um ambiente. Um microcontrolador é a escolha ideal porque pode integrar sensores de temperatura, controles de relé para o sistema de aquecimento/resfriamento e comunicação com outros dispositivos, tudo em um único chip. Em contrapartida, um microprocessador seria mais adequado para uma aplicação como um computador de bordo em um automóvel, onde é necessária maior capacidade de processamento e a integração com diversos sistemas complexos.

Compreender essas diferenças é essencial para escolher o componente mais adequado para cada projeto, garantindo desempenho, custo-benefício e eficiência no uso dos recursos.

Tipos de Entradas e Saídas

Os microcontroladores, como o Arduino e o ESP32, são dispositivos versáteis capazes de interagir com o mundo externo por meio de pinos configuráveis como entradas ou saídas. Esses pinos são os principais canais de comunicação entre o microcontrolador e sensores, atuadores ou outros dispositivos eletrônicos.

Compreender os diferentes tipos de entradas e saídas é essencial para aproveitar todo o potencial de um microcontrolador. Saber como configurar e utilizar esses recursos é o primeiro passo para desenvolver projetos inovadores, desde sistemas simples de automação até aplicações mais complexas em IoT (Internet das Coisas). A seguir, vamos explorar os diferentes tipos de entradas e saídas que você encontrará em microcontroladores.

Entradas Digitais

As entradas digitais permitem que o microcontrolador detecte o estado lógico de um sinal:

- **Nível ALTO (HIGH):** Geralmente corresponde a uma tensão próxima ao valor da alimentação (ex.: 3.3V ou 5V).
- **Nível BAIXO (LOW):** Normalmente representa 0V ou GND.

Essas entradas são usadas para ler sinais de botões, interruptores e sensores digitais que operam com lógica binária. Exemplo de uso: Um botão pressionado conecta o pino a GND, mudando o estado de HIGH para LOW.

Entradas Analógicas

As entradas analógicas permitem medir sinais de tensão variáveis. Esses sinais são convertidos em valores digitais através de um conversor Analógico-Digital (ADC).

Resolução: O número de bits do ADC determina a precisão da medição. Por exemplo, um ADC de 10 bits em um Arduino gera valores entre 0 e 1023. O intervalo de tensão depende do microcontrolador. No Arduino UNO, por exemplo, é de 0 a 5V.

Exemplo de uso: Leitura de um potenciômetro ou sensor de luz (LDR).

Saídas Digitais

As saídas digitais são usadas para controlar dispositivos que operam em dois estados: LIGADO (HIGH) ou DESLIGADO (LOW).

Podem acionar LEDs, relés, buzinas, entre outros dispositivos binários.

Muitas saídas podem fornecer corrente limitada (ex.: 20mA no Arduino UNO), necessitando de transistores ou módulos de driver para controlar dispositivos de maior potência.

Exemplo de uso: Acender um LED ou ligar/desligar um motor através de um relé.

Saídas Analógicas (PWM)

Embora a maioria dos microcontroladores não tenha saídas analógicas puras, eles podem simular sinais analógicos usando PWM (Pulse Width Modulation).

PWM: Uma técnica que controla a largura do pulso em uma série de sinais digitais para variar a potência ou tensão efetiva. Usado para controlar a intensidade de LEDs, a velocidade de motores ou criar sinais analógicos aproximados. Exemplo de uso: Ajustar o brilho de um LED ou controlar a velocidade de um motor.

Entradas e Saídas de Comunicação

Os microcontroladores possuem pinos dedicados para comunicação com outros dispositivos, incluindo:

- **UART (Serial):** Para comunicação serial, amplamente usada para debugar códigos ou comunicar com módulos externos, como módulos Bluetooth.
- **I2C:** Protocolo para conectar vários dispositivos com apenas dois pinos (SDA e SCL).
- **SPI:** Um protocolo mais rápido que I2C, mas que exige mais pinos (MISO, MOSI, SCLK e CS).

Exemplo de uso: Conectar um display OLED via I2C ou comunicar com um sensor IMU via SPI.

Entradas e Saídas Especiais

Pinos de interrupção: Permitem que o microcontrolador reaja imediatamente a um evento externo, ignorando o fluxo normal do programa.

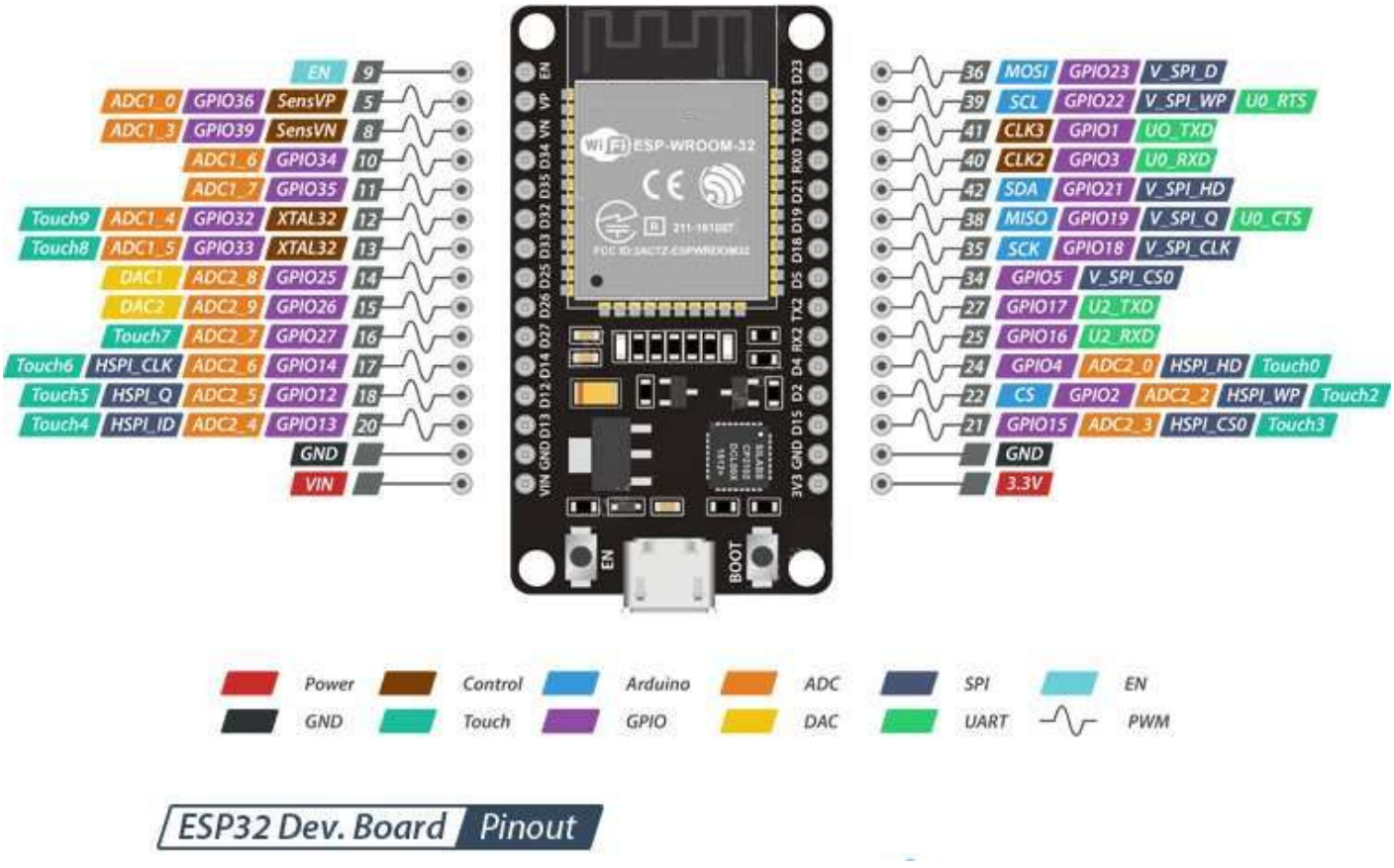
- **Entradas capacitivas (ESP32):** Usadas para criar botões touch sensíveis ao toque.
- **Conversores Digital-Analógico (DAC):** Disponíveis em alguns microcontroladores, como o ESP32, para gerar sinais analógicos reais.

Microcontroladores comuns

ESP32

Arquitetura

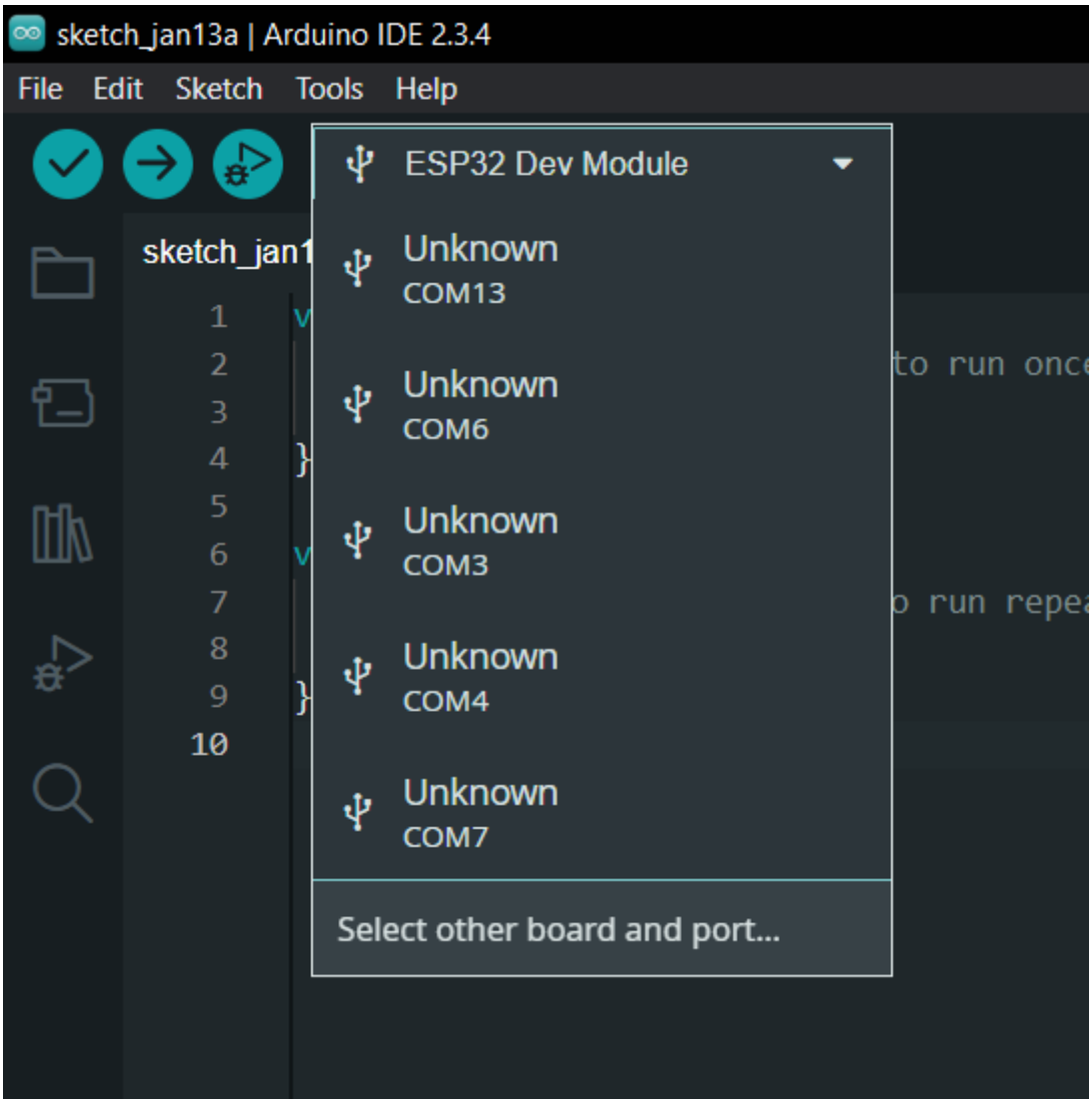
Abaixo temos uma imagem mostrando os tipos de conexões suportadas para cada porta da ESP32, as principais que serão usadas ao longo dessa apostila são: as entradas analógicas (ADC), digitais (GPIO) e as de comunicação I2C (GPIO21 E GPIO22).



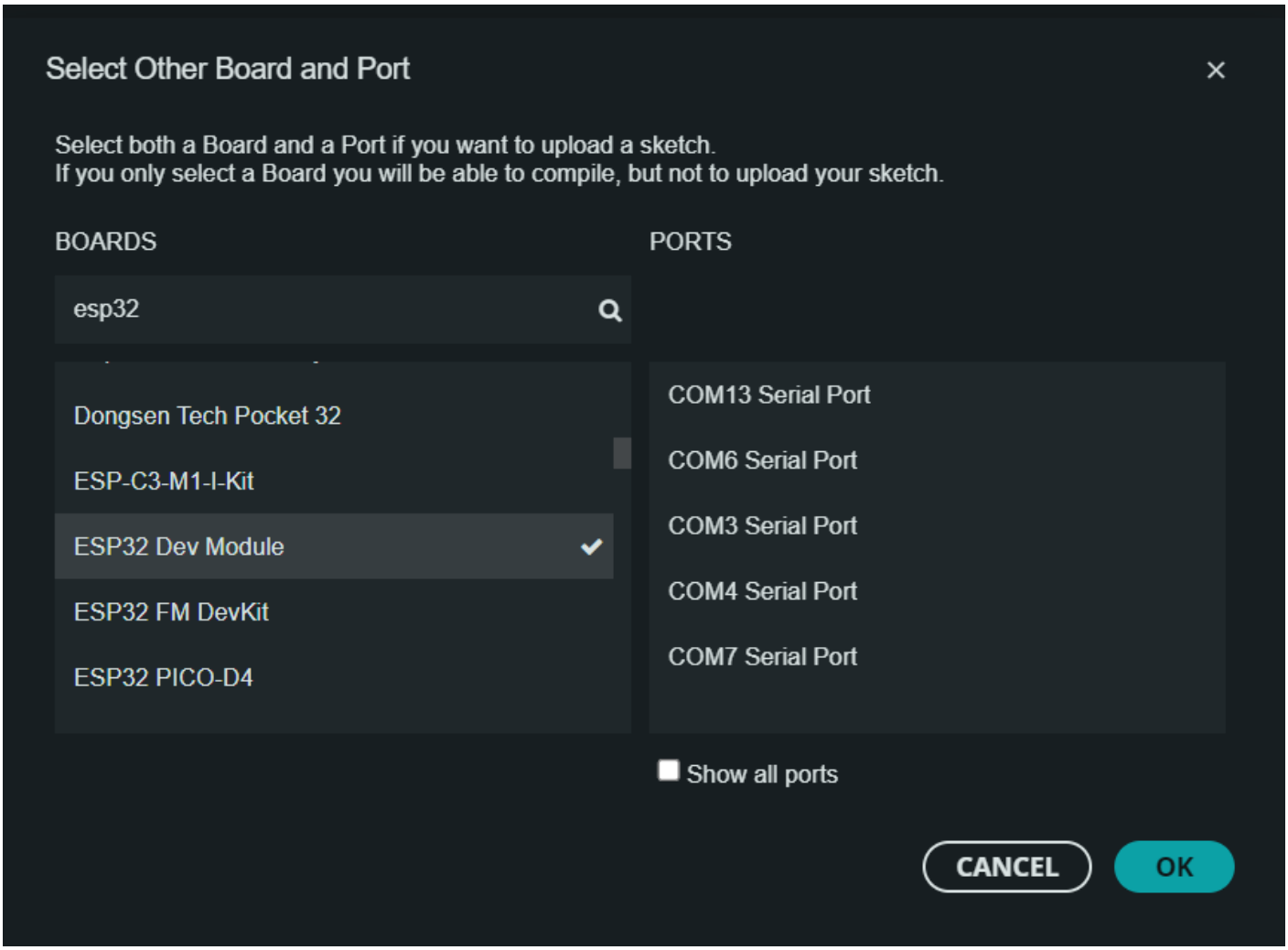
Guia de conexão

1. Baixar 2 drivers de USB para comunicação com a placa pelo Windows: CP210x e CH341SER. (Para o segundo driver, é necessário que a placa esteja conectada em uma porta USB para funcionar)
2. Com o aplicativo Arduino IDE (download na secção 1.1.1) será possível ver que, após a instalação dos drivers, uma nova porta COM aparecerá e ela é a que deve ser utilizada.
3. Para ver as portas disponíveis, abra o menu Tools $->$ Port: (figura \ref{fig: Econex1})

4. Escolher o modelo de placa que você está usando no menu de escolha de porta e placa (“Select other board and port...”): (figura \ref{fig: Econex2})



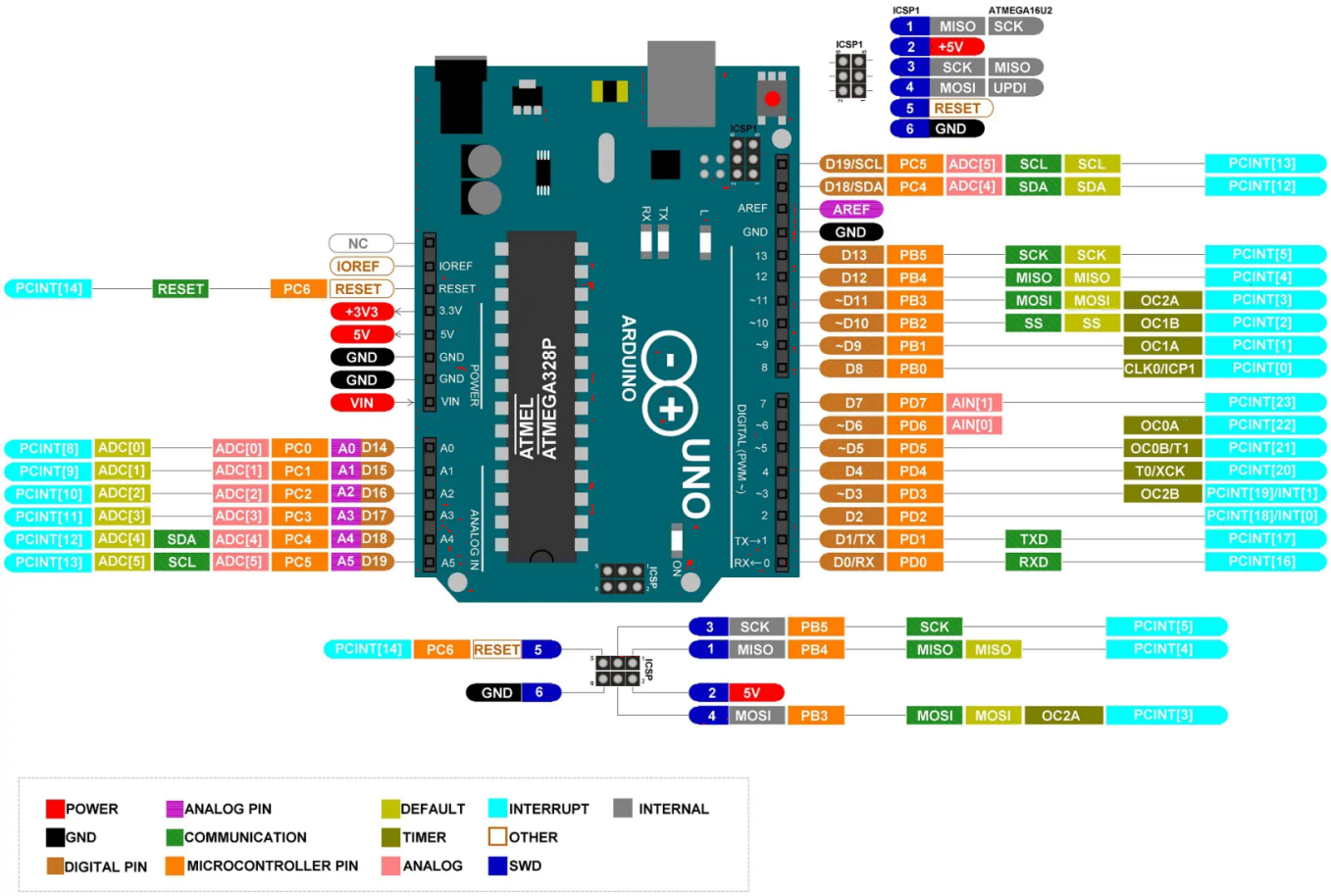
5. Com o menu aberto você irá pesquisar a placa que está usando e selecionar a porta COM que apareceu nova quando os drivers foram instalados: (figura \ref{fig: Econex3})



Arduino

Arquitetura

Abaixo temos uma imagem mostrando os tipos de conexões suportadas para cada porta dO aRDUINO, as principais que serão usadas ao longo dessa apostila são: as entradas analógicas (ANALOG PIN), digitais (DIGITAL PIN) e as de comunicação I2C (SCL e SDA).



Guia de Conexão

1. Fazer o download da interface Arduino IDE;
2. Com a IDE aberta, a placa Arduino que for conectada deve ser reconhecida imediatamente pelo programa que fará a conexão automática no sistema, pronta para uso.

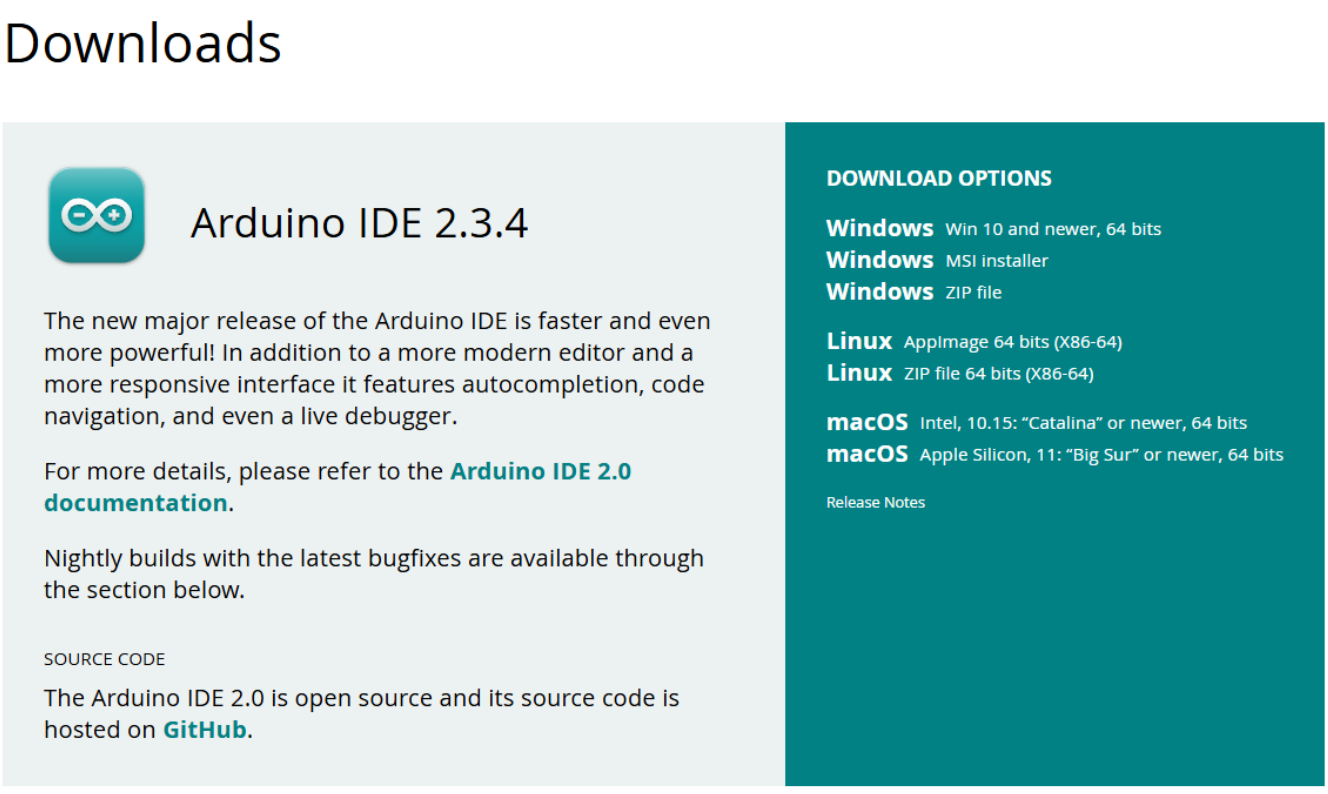
Interfaces de programação

Interface Arduino IDE

A Arduino IDE (Integrated Development Environment) é uma plataforma de desenvolvimento fundamental para programar placas Arduino e outros microcontroladores compatíveis. Sua principal função é oferecer um ambiente simples e intuitivo para escrever, compilar e enviar códigos (sketches) para os dispositivos físicos. A IDE suporta a linguagem de programação baseada em C/C++ e integra bibliotecas que facilitam a criação de projetos eletrônicos, desde automações simples até sistemas complexos de IoT. Sua importância reside na acessibilidade que proporciona a iniciantes e profissionais, tornando o desenvolvimento de soluções embarcadas mais ágil, didático e eficiente.

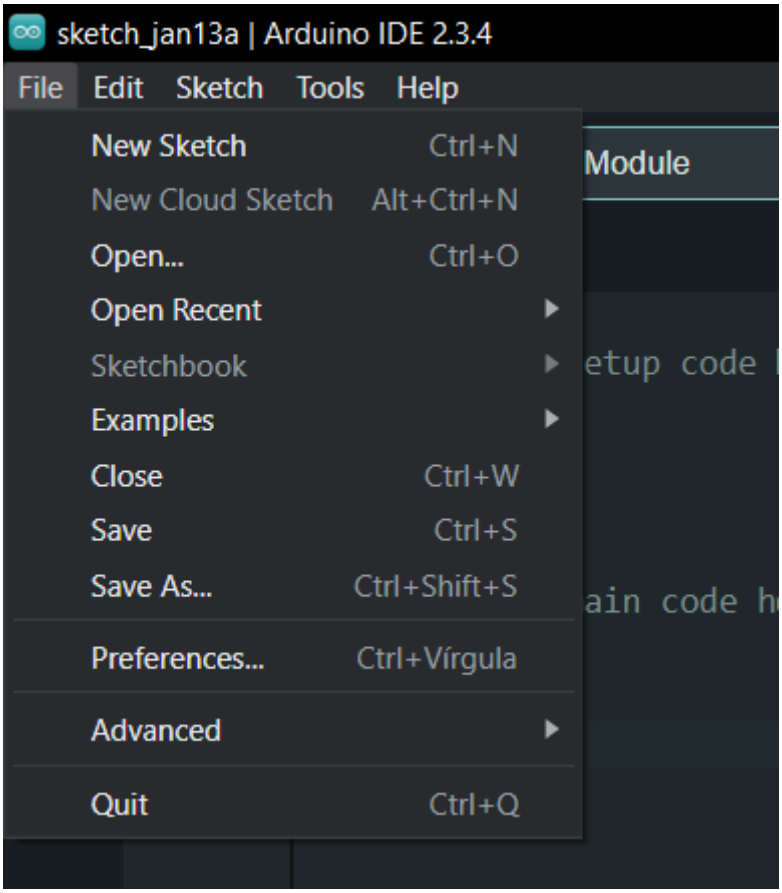
Download

Para fazer o download da interface de programação Arduino IDE entre no site https://www.arduino.cc/en/softwaree selecione a opção correspondente ao computador em que o aplicativo está sendo instalado.

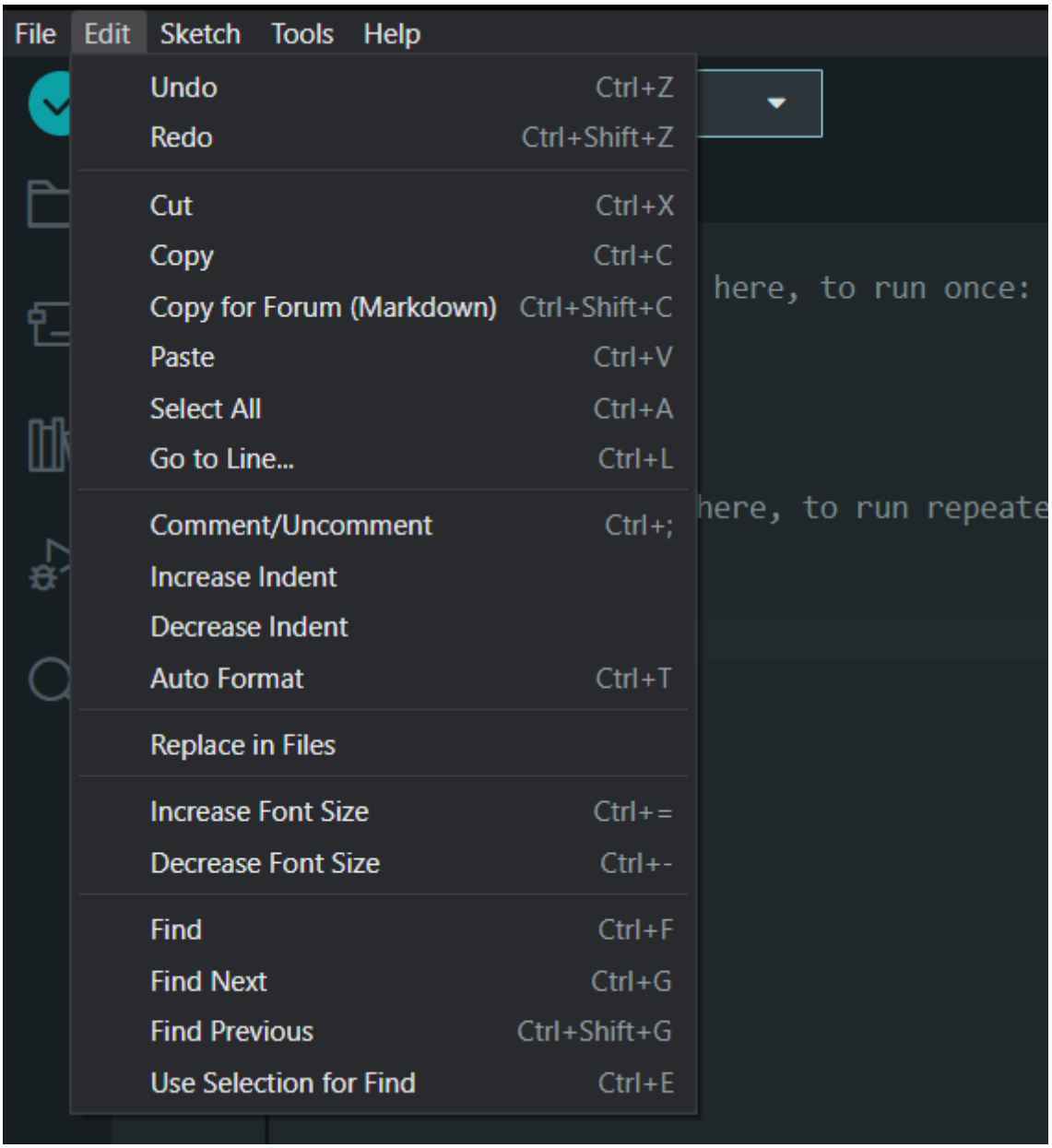


Menus e Funções

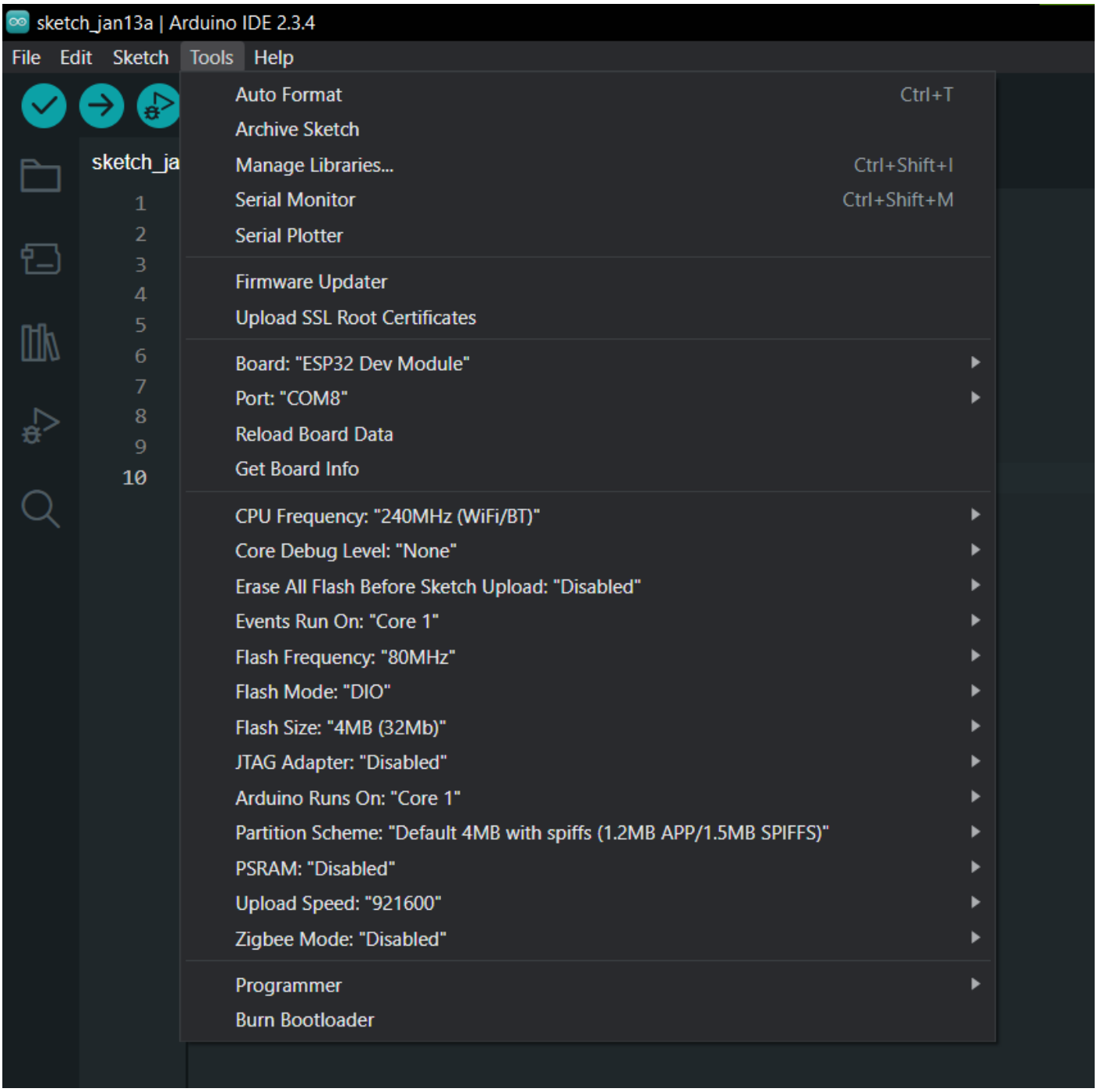
A Arduino IDE possui uma grande diversidade de funções e opções em seus menus, aqui abordaremos algumas das principais funções e onde elas se encontram. Começando com os menus no topo esquerdo da tela. Temos primeiramente o menu Files abaixo que é onde encontram-se as funções de criar novos sketchs ou abrir existentes no seu computador, abrir o menu de exemplos prontos da própria arduino IDE ou de bibliotecas que você pode baixar e as opções de salvar os sketchs.



Em seguida, temos o menu Edit, onde é possível encontrar os principais atalhos como copiar, colar, desfazer, etc, e algumas opções para alteração de tamanho da fonte.



No menu Tools, temos algumas ferramentas muito importantes, como a de abrir o monitor serial, escolher a placa que está sendo utilizada e definir a porta COM onde a placa está conectada.



Por fim, temos a interface principal com as funções:

- **Verify / Upload (Verificar / Carregar):** Compila e envia seu código para a placa.
- **Select Board and Port (Selecionar Placa e Porta):** Placas Arduino detectadas automaticamente aparecem aqui, junto com o número da porta.
- **Sketchbook:** Aqui você encontrará todos os seus sketches armazenados localmente no computador. Além disso, é possível sincronizar com a Arduino Cloud e acessar sketches do ambiente online.
- **Boards Manager (Gerenciador de Placas):** Explore pacotes da Arduino e de terceiros que podem ser instalados.
- **Library Manager (Gerenciador de Bibliotecas):** Explore milhares de bibliotecas Arduino criadas pela Arduino e pela comunidade.
- **Debugger (Depurador):** Teste e depure programas em tempo real.

- **Search (Pesquisar):** Busque por palavras-chave no seu código.
- **Open Serial Monitor (Abrir Monitor Serial):** Abre a ferramenta Monitor Serial em uma nova aba no console.

Para Informações mais detalhadas de funcionamento ou tirar dúvidas, entre no guia completo da Arduino clicando em help no topo do aplicativo ou acessando o site de documentação

https://docs.arduino.cc/

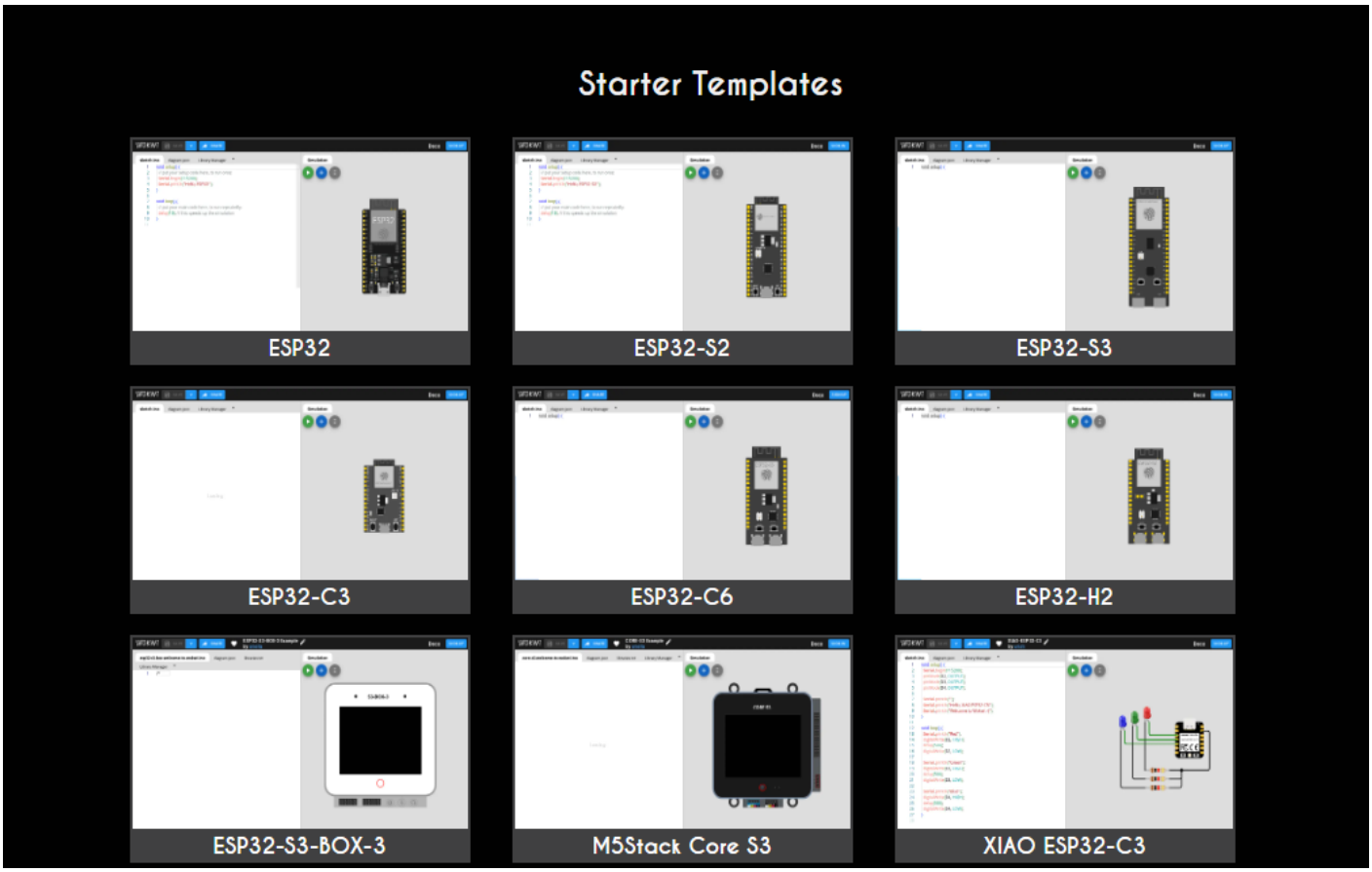
Interface WOKWI: Introdução à simulações

O site Wokwi é uma ferramenta poderosa para fazer testes de códigos e circuitos para aqueles que não possuem acesso a microcontroladores e sensores ou apenas precisam de uma forma mais prática de testar suas ideias. Nele, é possível montar circuitos com microcontroladores como a ESP32 e o Arduino e fazer o teste de programas neles em simulações que recriam o que ocorre num sistema. Para acessar o site, pesquise "Wokwi" no seu navegador ou acesse o site: https://wokwi.com/

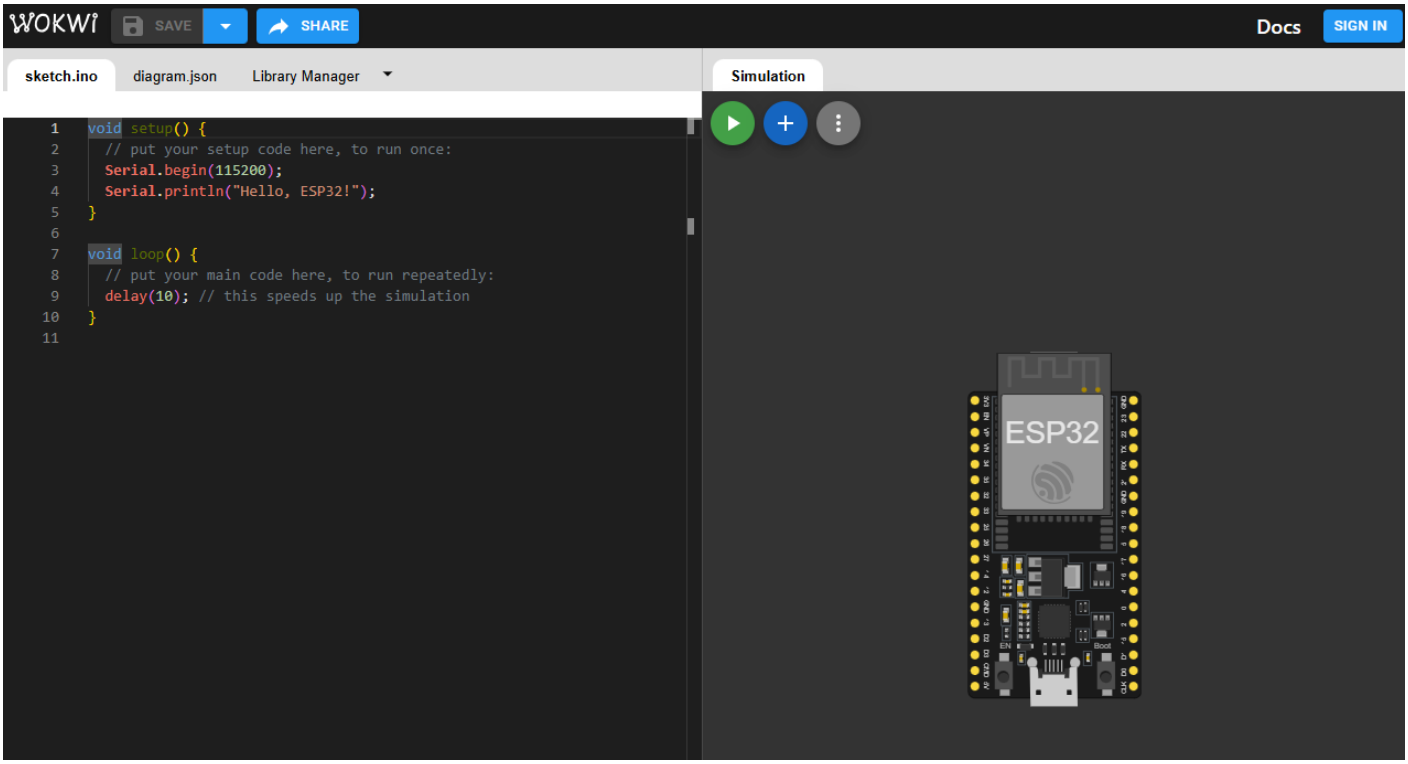
Interface e funções

Quando o site é aberto, a primeira coisa que aparece são as opções de microcontrolador que se deseja simular (neste curso utilizaremos principalmente a ESP32, mas o conteúdo deve funcionar para qualquer um dos microcontroladores sabendo-se a arquitetura dos mesmos). Descendo a página inicial, estão expostos alguns exemplos criados pela comunidade de montagens mais avançadas que podem ser utilizadas como referência em códigos mais complexos ou por curiosidade.

Após escolher o microcontrolador que deseja, a nova página mostrará exemplos específicos daquele microcontrolador, e abaixo terá a opção de escolher o modelo desejado e começar um novo template para simular. para este curso, recomenda-se escolher o primeiro template para ESP32 padrão.



Com o template escolhido, será aberta a interface principal, que será utilizada para adicionar sensores e o código na placa a fim de simula-los:



Na metade direita da tela, temos a seção onde é feita a montagem do circuito. Para adicionar componentes como sensores e leds, basta clicar no botão azul com o sinal de mais e escolher o que é preciso. Para mover os componentes apenas clique e segure e arraste ele para a posição

desejada, e para conectar fios entre os itens posicionados, apenas clique na perna do componente e depois clique onde deseja conectá-lo.

Na metade esquerda temos a parte de programação da placa, é nela em que deve ser inserido o código que deseja carregar nela. Nesta parte, também temos o menu de configuração de bibliotecas, basta clicar na aba "library manager" no topo da tela e na interface que aparecer, pesquise as bibliotecas que deseja adicionar (as bibliotecas que utilizaremos ao longo do curso são possíveis de encontrar diretamente no site sem a necessidade de baixá-las manualmente.

Após realizar a montagem dos componentes e escrever o código necessário para utilizá-los, clique no botão de "play" verde do lado direito e o site ira simular o funcionamento do que foi montado, mostrando os dados do monitor serial em um monitor que aparecerá no lado direito.

Para Informações mais detalhadas de funcionamento ou tirar dúvidas, entre no guia completo do próprio site clicando em "Docs" no topo direito do site ou acesse a documentação em https://docs.wokwi.com/pt-BR/

Programação em C++

Funções e variáveis

Agora que foi apresentado sobre as interfaces de programação que são utilizadas para configurar e simular microcontroladores, mostraremos nesta seção como funciona a linguagem C++ e o básico para que seja possível compreender os códigos das próximas seções e como criar os seus próprios.

Funções Básicas

No ambiente de programação do Arduino, as funções principais são `setup` e `loop`. Ambas têm papéis específicos na estrutura do programa e são essenciais para o funcionamento do código.

A função setup prepara o ambiente de execução. A função loop mantém o microcontrolador em operação contínua, executando tarefas de forma cíclica. Essa separação simplifica o desenvolvimento, organizando o código de forma clara e eficiente.

Função Setup

A função setup é executada apenas uma vez, logo após o microcontrolador ser ligado ou reiniciado. Ela é usada para configurar o ambiente inicial do programa, como:

- Configurar pinos como entrada ou saída.
- Inicializar comunicações seriais (ex.: Serial.begin(9600);).
- Configurar periféricos e bibliotecas necessárias para o funcionamento do programa.

Exemplo:

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT); // Define o LED como saída
}
```

Função Loop

A função loop é executada continuamente após a conclusão da setup. Todo o código que define o comportamento repetitivo ou contínuo do microcontrolador deve ser implementado aqui.

Exemplo:

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // Liga o LED
  delay(1000);                     // Aguarda 1 segundo
  digitalWrite(LED_BUILTIN, LOW);  // Desliga o LED
  delay(1000);                     // Aguarda 1 segundo
}
```

Tipos de variáveis em C++ e seus principais usos

Ao programar em C++ para aplicações de microcontroladores, é essencial entender os diferentes tipos de variáveis disponíveis. Cada tipo de variável é projetado para armazenar dados de formas específicas, otimizando o uso de memória e garantindo a eficiência do código. Abaixo, explicamos os principais tipos de variáveis em C++ e seus usos mais comuns:

Variáveis Inteiras

Variáveis inteiras são usadas para armazenar números inteiros (positivos, negativos ou zero), sem casas decimais.

- **Tipo "int":** É o tipo inteiro mais utilizado. Armazena valores entre -32.768 e 32.767 (em sistemas de 16 bits) ou entre -2.147.483.648 e 2.147.483.647 (em sistemas de 32 bits).
  Uso: Contadores, índices de laços, medições que não requerem frações.
- **Tipo "short":** Um tipo inteiro menor, que ocupa menos memória.
  Uso: Ideal para situações onde a memória é limitada e o intervalo de valores é pequeno.
- **Tipo "long" e "long long":** Usados para armazenar números inteiros muito grandes.
  Uso: Cálculos que exigem maior precisão ou intervalos mais amplos.
- **Modificador "unsigned":** Indica que a variável não armazenará valores negativos, dobrando o limite superior.
  Uso: Contagem de eventos ou medições sempre positivas, como leituras de sensores.

Variáveis de Ponto Flutuante

Usadas para representar números reais (com casas decimais).

- **Tipo "float":** Representa números com precisão simples (32 bits).
  Uso: Cálculos que exigem precisão moderada, como medições analógicas ou coordenadas em um sistema.
- **Tipo "double":** Representa números com precisão dupla (64 bits).
  Uso: Cálculos mais complexos que exigem maior precisão.

Nota: Em microcontroladores com recursos limitados, é importante usar `float` ou `double` apenas quando necessário, pois essas variáveis consomem mais memória e poder de processamento.

Variáveis de Caractere

- **Tipo "char":** Usado para armazenar um único caractere (como 'A', '1', ou '?') ou valores numéricos pequenos (de 0 a 255 no caso de `unsigned char`).
  Uso: Representa caracteres individuais ou pequenos números inteiros, como códigos ASCII.

Variáveis Booleanas

- **Tipo "bool":** Armazena valores lógicos, ou seja, `true` (verdadeiro) ou `false` (falso).
  Uso: Controle de fluxos lógicos, como verificar se uma condição foi atendida.

Variáveis Compostas

- **Tipo "array":** Conjunto de variáveis do mesmo tipo armazenadas em sequência.
  Uso: Armazenar múltiplos valores relacionados, como leituras de um sensor ao longo do tempo.
- **Estruturas (struct):** Agrupam variáveis de diferentes tipos sob um único nome.
  Uso: Representação de objetos mais complexos, como as informações de um dispositivo.

Ponteiros

- *Tipo* "*int*", "*float*", etc.:** Variáveis que armazenam o endereço de memória de outra variável.
  Uso: Manipulação direta de memória, comunicação com periféricos ou passagem eficiente de dados entre funções.

Considerações Importantes

Ao trabalhar com microcontroladores, é essencial escolher o tipo de variável adequado para:

- **Otimizar o uso de memória:** Microcontroladores geralmente têm recursos limitados.
- **Garantir eficiência:** Variáveis menores permitem operações mais rápidas.
- **Evitar erros:** Certifique-se de que o intervalo de valores suportado pela variável é suficiente para a aplicação.

Comunicação serial

Comunicação serial

A comunicação serial é um dos meios mais simples e eficientes para enviar e receber dados entre o microcontrolador e outros dispositivos, como computadores ou módulos externos. No Arduino, a biblioteca `Serial` facilita esse tipo de comunicação utilizando o protocolo UART.

Como Funciona

- **Velocidade de Comunicação:** Definida em bauds (ex.: 9600, 115200), representa o número de bits transferidos por segundo.
- **Transmissão e Recepção:** Usa dois pinos — TX (transmissão) e RX (recepção) ou comunicação direta com o computador na IDE por cabo.
- **Formato de Dados:** Tipicamente, cada pacote de dados inclui 8 bits, 1 bit de parada e nenhum bit de paridade.

Configuração inicial

Antes de utilizar a comunicação serial, é necessário inicializá-la na função `setup` com o comando:

```
Serial.begin(9600);
```

Envio e recebimento de dados

Para enviar dados do microcontrolador, utiliza-se o comando:

```
Serial.print("Mensagem"); // Envia uma mensagem sem pular linha
Serial.println("Mensagem"); // Envia uma mensagem e adiciona uma nova linha
```

Para receber dados, é possível verificar se há informações disponíveis com:

```
if (Serial.available() > 0) {
  char dado = Serial.read(); // Lê um byte disponível
  Serial.println(dado); // Ecoa o dado recebido
}
```

Exemplo completo

```
void setup() {
    Serial.begin(9600); // Inicializa a comunicação serial
    Serial.println("Iniciando comunicação serial...");
}

void loop() {
  if (Serial.available() > 0) {
    char recebido = Serial.read();
    Serial.print("Você enviou: ");
    Serial.println(recebido);
  }
  delay(100); // Pequena pausa para evitar sobrecarga
}
```

Aplicações comuns

- Depuração e monitoramento de código.
- Comunicação com módulos Bluetooth, Wi-Fi, ou GPS.
- Envio de comandos para o microcontrolador a partir de um computador ou outro dispositivo.

Compreender a comunicação serial é essencial para criar projetos interativos e para monitorar o comportamento do microcontrolador durante a execução de um programa.

Tutorial sensores básicos

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:

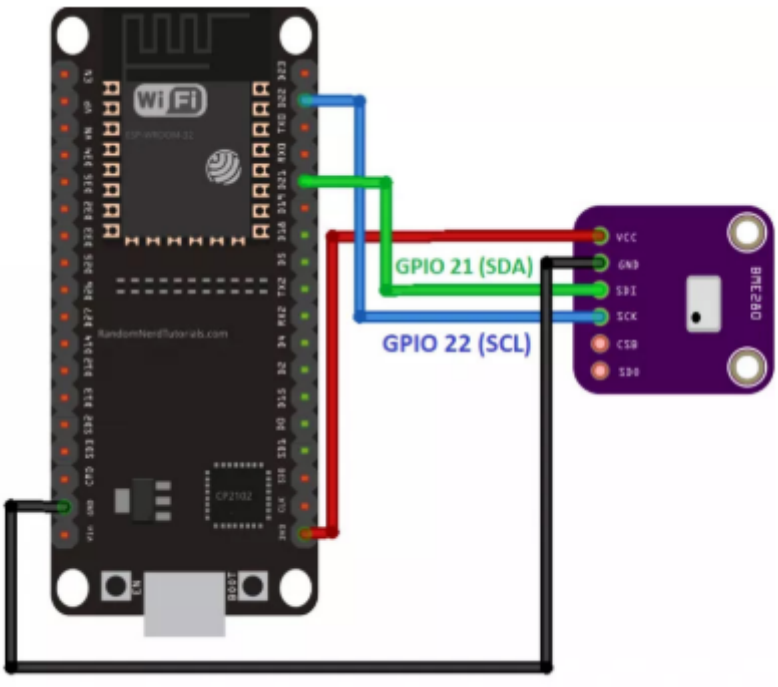
- **SDA (Serial Data Line):** Esta linha é usada para transmitir os dados entre os dispositivos.
- **SCL (Serial Clock Line):** Essa linha é responsável por sincronizar a transferência de dados.

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.

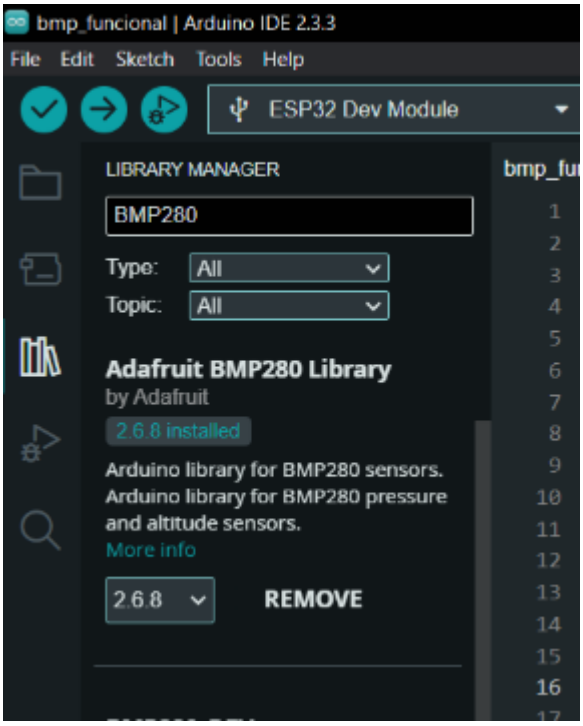
Sensores

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");
```

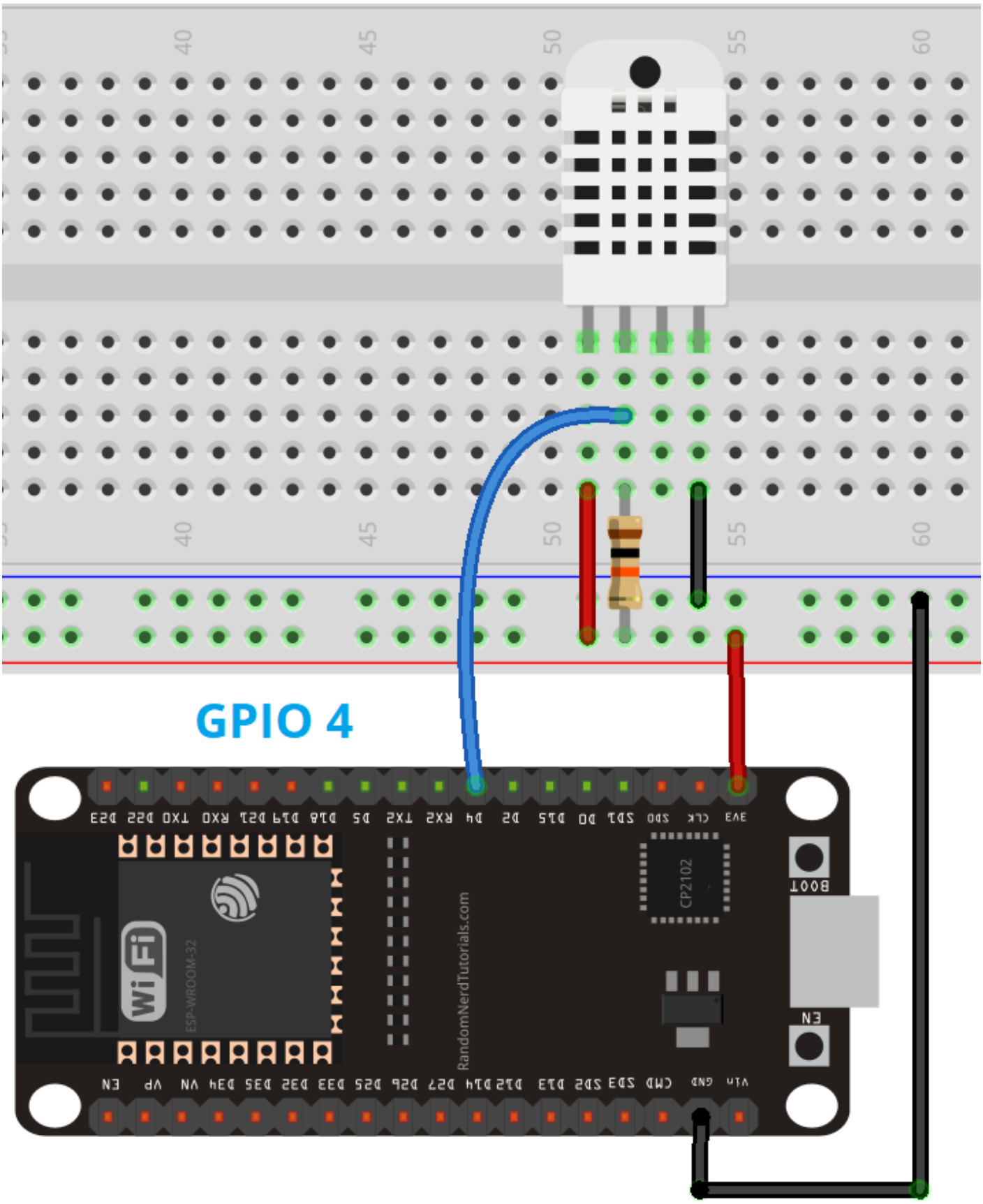
```
  delay(2000); // Aguarda 2 segundos entre as medicoes
}
```

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.

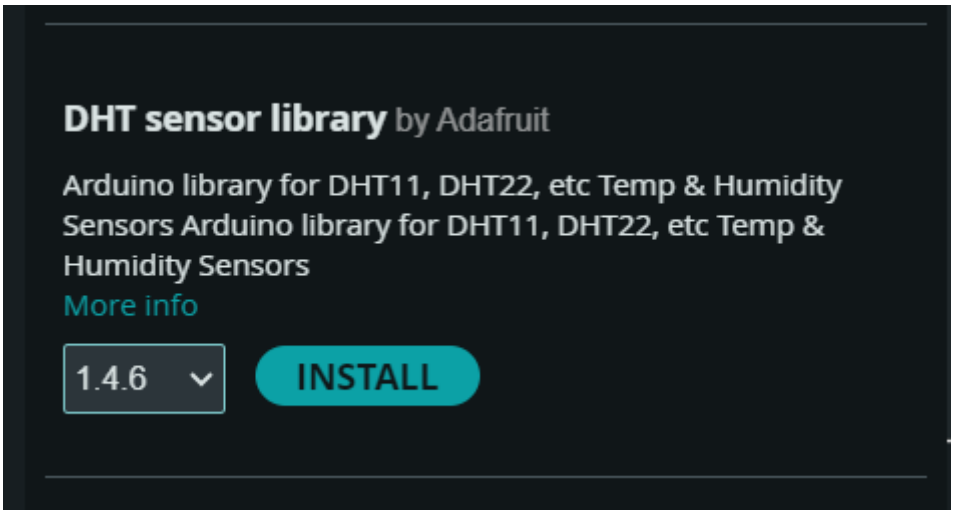
Referência: https://www.instructables.com/How-to-Connect-BMP-280-to-ESP32-Get-Pressure-Tempe/

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.

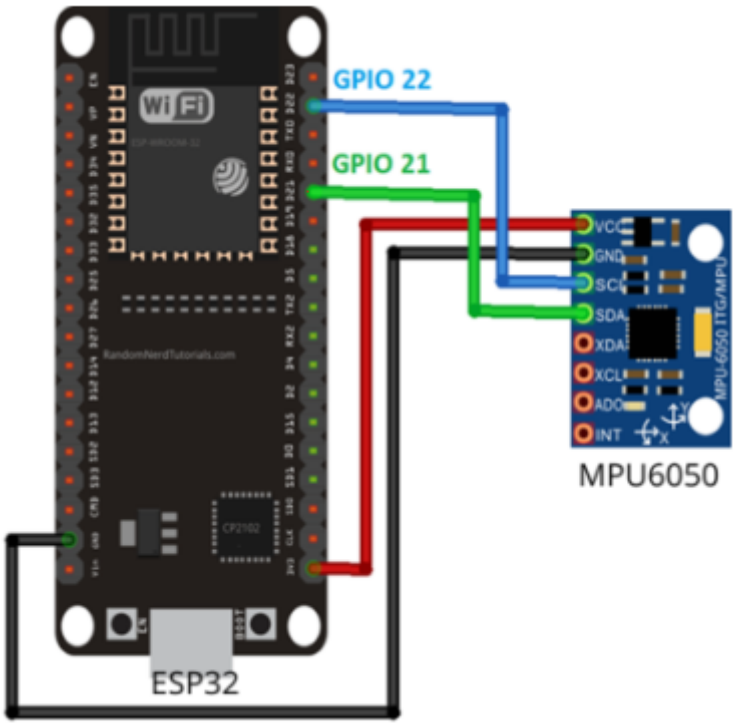
```
#include "DHT.h"

#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"));
}
```

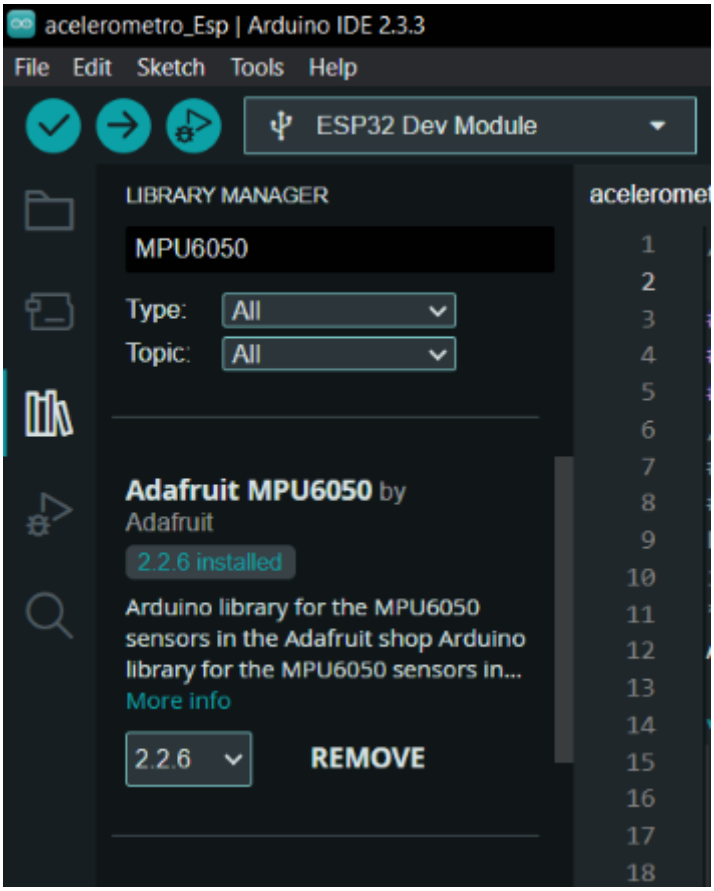
Referência: https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/

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;
```

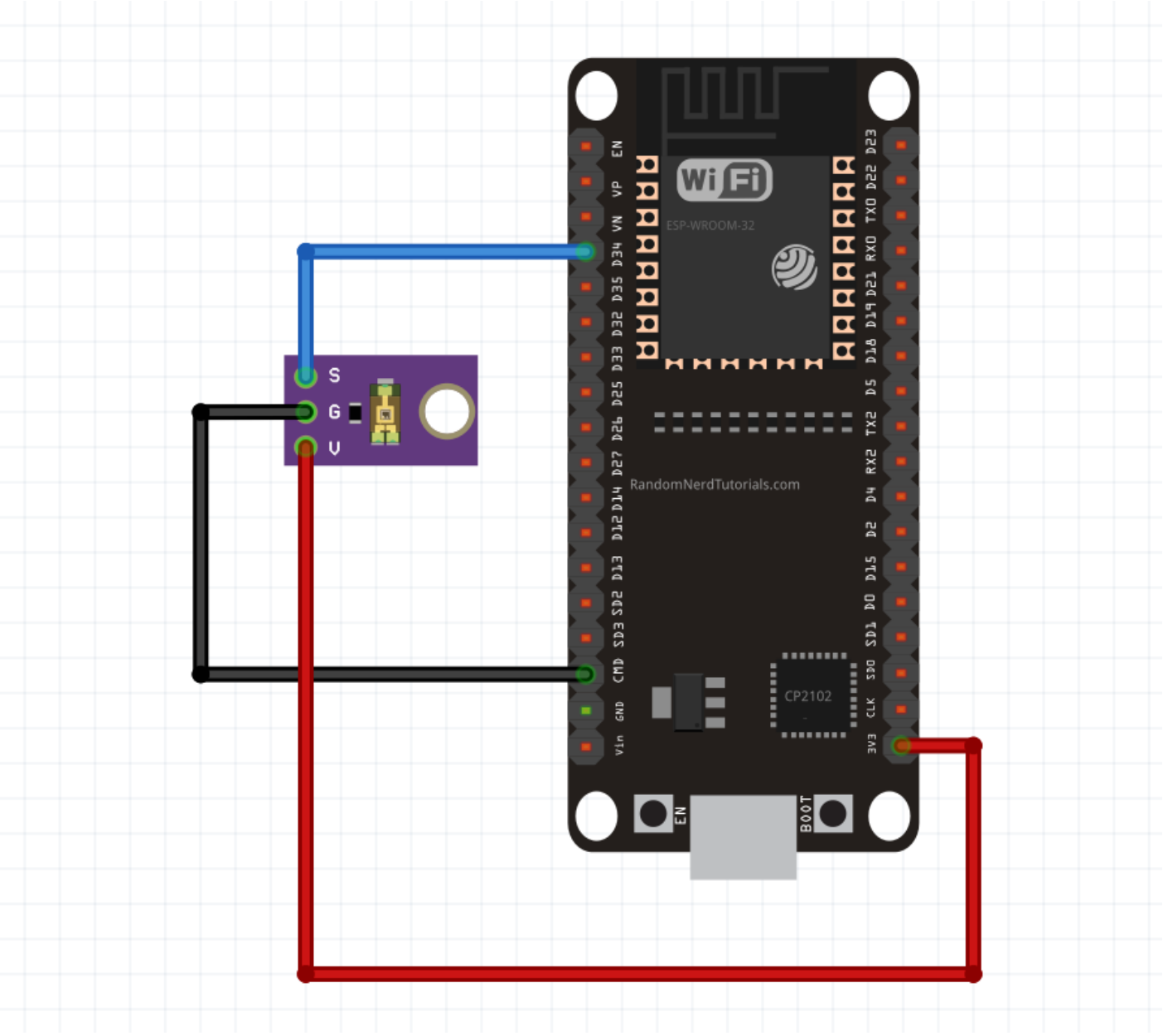
```
  case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
  case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
  }

  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.