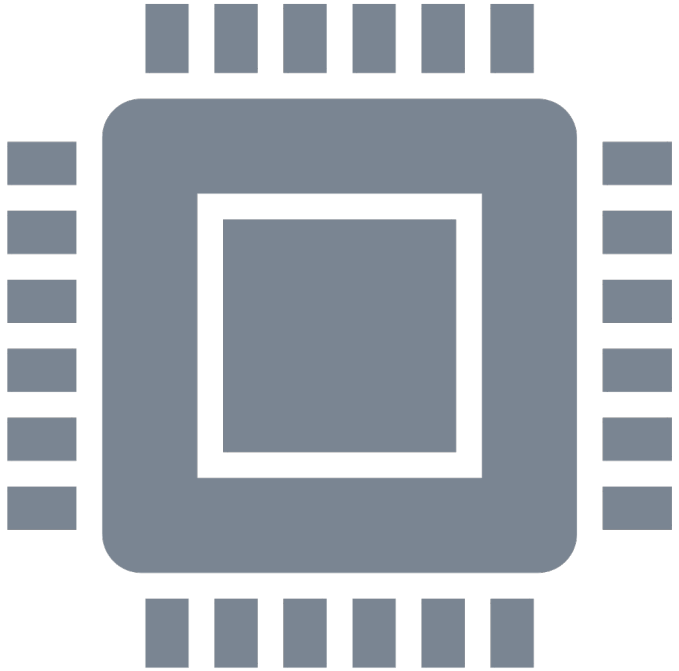
FAB LAB HENRIQUES NOGUEIRA

Noções Básicas de Eletrónica e Robótica com Arduino

Oficina de Formação

Agenda

- Apresentação
  - O que é o Arduino?
  - O Kit Grove Beginner
  - O Software de Programação do Arduino
  - A Estrutura Base do Código do Arduino
- "Mãos na Massa"
  - Realização das tarefas propostas

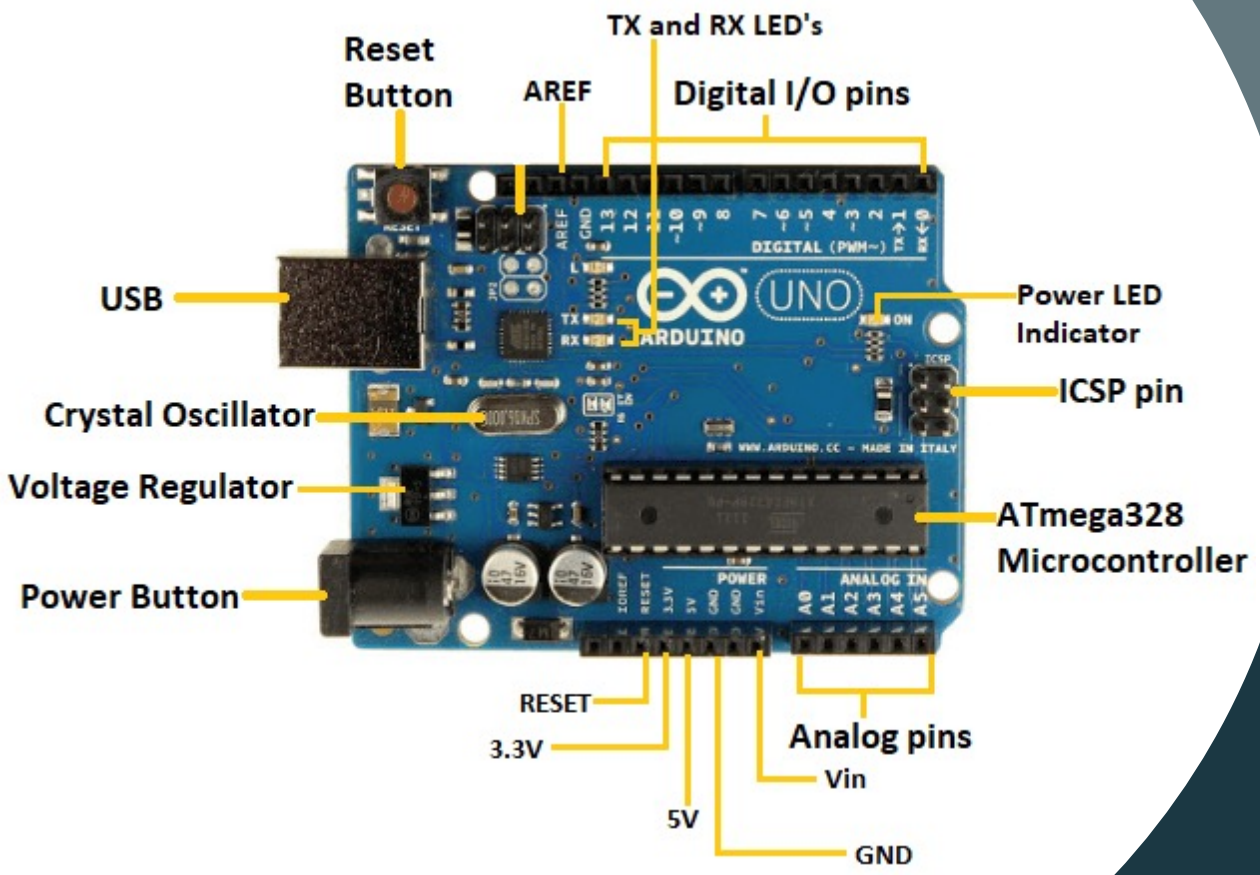


O que é o Arduino?

- Arduino é uma plataforma de prototipagem eletrónica, criado por Massimo Banzi e David Cuartielles, em 2005;
- O objetivo é permitir o desenvolvimento de controlo de sistemas interativos, de baixo custo e acessível a todos;
- Todo material envolvido (software, bibliotecas, hardware) é open-source;
- Utilizam um microcontrolador da família Atmel AVR e uma linguagem de programação baseada em C/C++.
- Possibilita a criação de projetos variados em eletrónica, desde os mais simples até aplicações intermediárias como Internet das Coisas (IoT), Robôs, Sistemas de Automação Residencial ou Industrial, Alarmes e outros.

O que ó o Arduino?

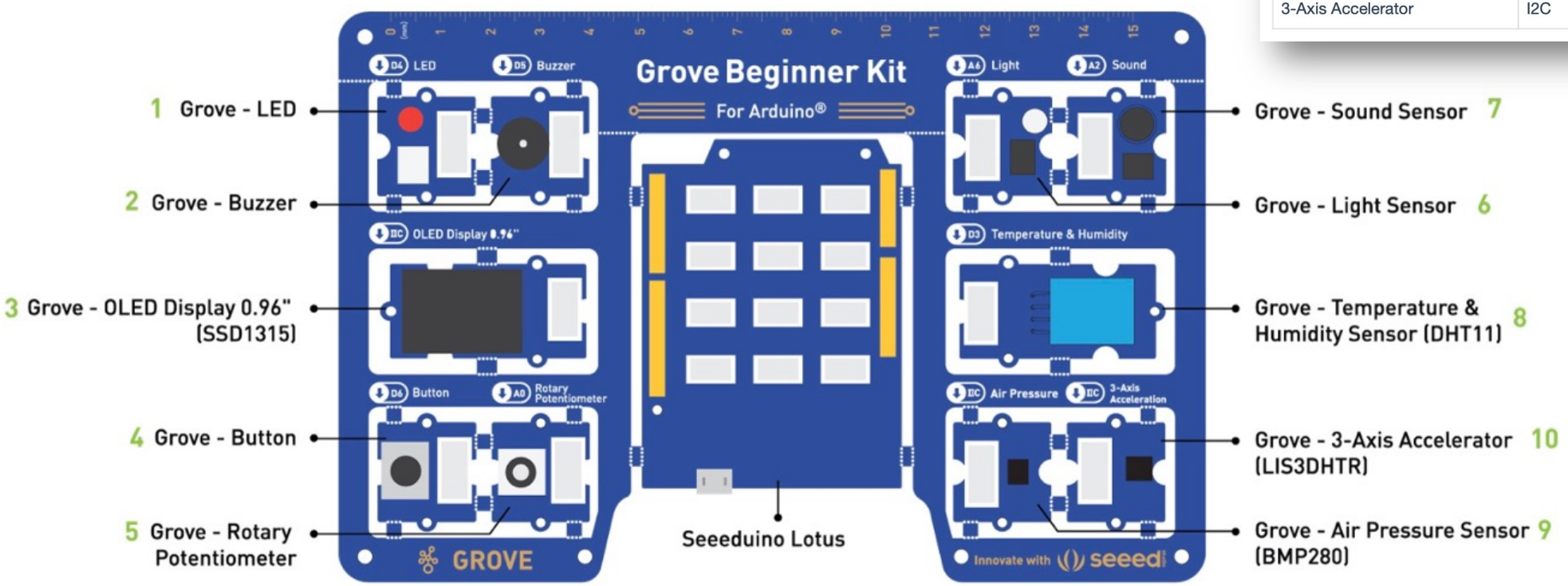
- Constituição as placas Arduino:
  - Fonte de Alimentação;
  - Núcleo CPU;
  - Entradas e Saídas;
  - Pinos com Funções;
  - Firmware.



O Kit Grove Beginner

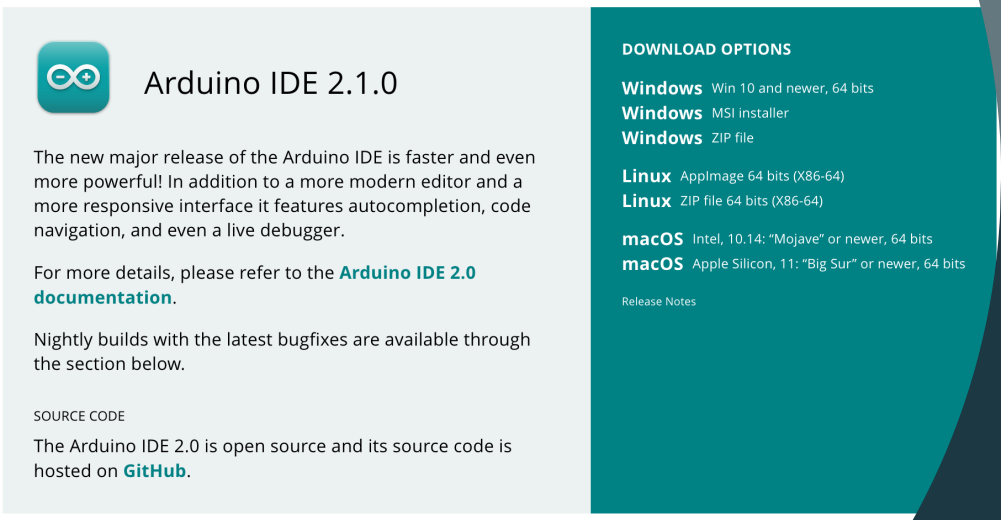
- Kit eletrónico baseado no Arduino Uno;
- Disponibiliza vários sensores todos integrados;

Modules	Interface	Pins/Address
LED	Digital	D4
Buzzer	Digital	D5
OLED Display 0.96”	I2C	I2C, 0x78(default)
Button	Digital	D6
Rotary Potentiometer	Analog	A0
Light	Analog	A6
Sound	Analog	A2
Temperature & Humidity Sensor	Digital	D3
Air Pressure Sensor	I2C	I2C, 0x77(default) / 0x76(optional)
3-Axis Accelerator	I2C	I2C, 0x63(default)



O Software de Programação

- O IDE Arduino pode ser descarregado do site oficial do Arduino (https://www.arduino.cc/en/software)
- É importante instalar o driver USB (***CP2102 USB Driver***) de comunicação com o computador. Pode ser descarregado a partir do site https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads

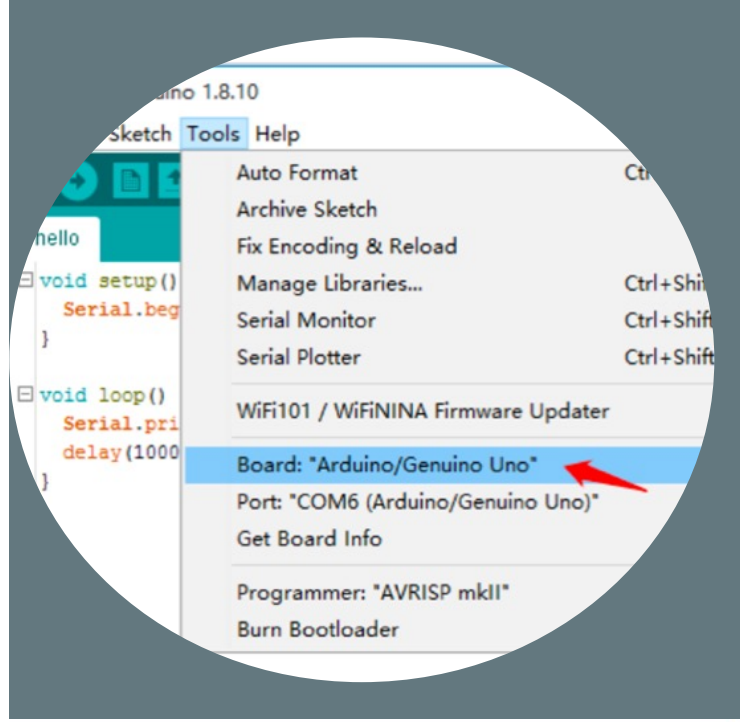


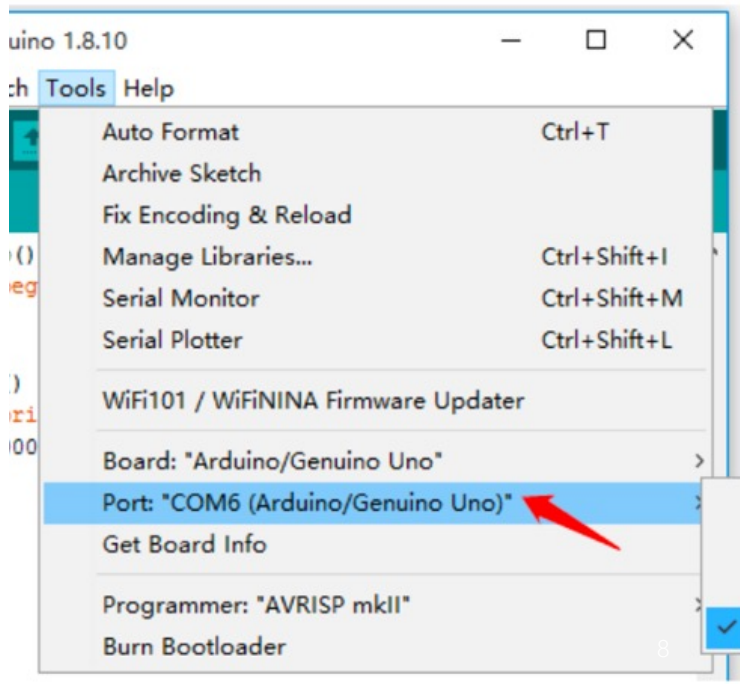
Estrutura Base do Código no Arduino

```
// Zona da importação das Bibliotecas
// e definição de constantes e variáveis globais

// Bloco de código de inicialização (só é executado uma vez, no início)
void setup() {
// put your setup code here, to run once:

}

// Bloco de execução (o código é executado repetidamente)
void loop() {
// put your main code here, to run repeatedly:

}
```

Oficina 0:
Testar o IDE e a ligação ao Arduino

1. Executar o IDE Arduino;
2. Ligar o Arduino ao computador através do cabo USB;
3. No menu **Tools -> Board** selecionar a placa *"Arduino/Genuino Uno"* ou apenas *"Arduino Uno"*;
4. No menu **Tools -> Port** definir a porta de comunicação COM de ligação do computador ao Arduino;



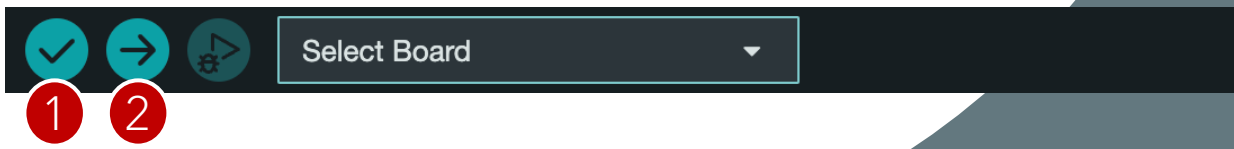


Oficina 0: Testar o IDE e a ligação ao Arduino

5. Criar um novo ficheiro com o nome hello.ino e digitar o seguinte código:

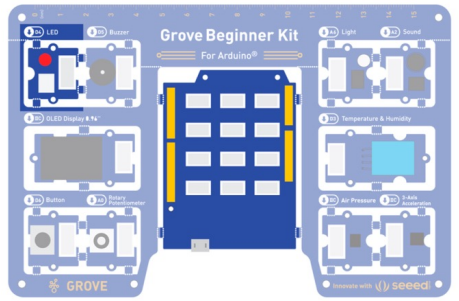
```
void setup() {
  Serial.begin(9600); // initializes the serial port with a baud rate of
 9600
}
void loop() {
  Serial.println("hello, world"); // prints a string to a serial port
  delay(1000); //delay of 1 second
}
```

6. Depois de digitado o código, é necessário compilá-lo, clicando no ícone 1;
7. Efetuar o upload o programa para o Arduino, clicando no ícone 2 ;
8. Verificar o funcionamento do programa menu Tools-> Serial Monitor.



Oficina 1: LED

```
//LED Blink
//The LED liga e desliga em intervalos de 1 segundo

int ledPin = 4;

// Bloco de inicialização e definição
void setup() {
    pinMode(ledPin, OUTPUT);}

//Bloco de execução que se repete indefinidamente
void loop() {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

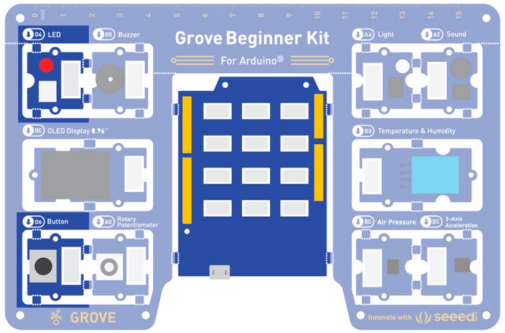


Componentes envolvidos:

1. Arduino
2. Módulo LED
3. Cabo (se necessário)

Oficina 2: Ligar o LED com o Botão

```
// Botão liga e desliga o LED

const int buttonPin = 6;
const int ledPin =  4;

int buttonState = 0;

// Bloco das definições
void setup() {
  // Inicializa o pin do led como output:
  pinMode(ledPin, OUTPUT);
  // Inicializa o pin do botão como input:
  pinMode(buttonPin, INPUT);

}
```

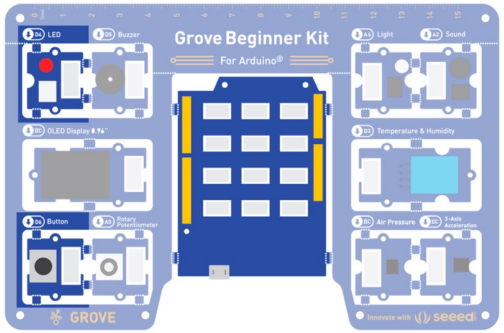


Componentes envolvidos:

1. Arduino
2. Módulo LED
3. Módulo Button
4. Cabos (se necessário)

Oficina 2: Ligar o LED com o Botão

```
// Bloco de execução
void loop() {
  // Lê o estado do botão:
  buttonState = digitalRead(buttonPin);

  // Verifica se o botão está pressionado (HIGH):
  if (buttonState == HIGH) {
     digitalWrite(ledPin, HIGH);  // Liga lED

  } else {
    digitalWrite(ledPin, LOW);    // Desliga LED
  }
}
```

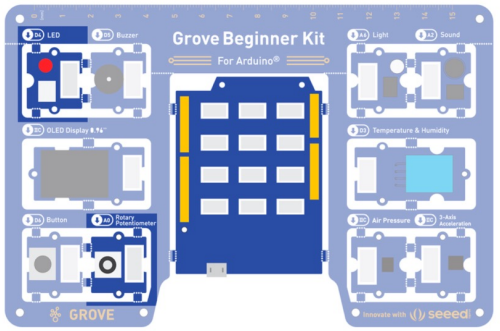


Componentes envolvidos:

1. Arduino
2. Módulo LED
3. Módulo Button
4. Cabos (se necessário)

Oficina 3: Controlar a velocidade do blink do LED

```
//Controlar piscar LED com potenciómetro
const int rotaryPin = A0;
const int ledPin = 4;
int rotaryValue = 0;

void setup() {
  // Define o pin do LED como output:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Lê o valor do potenciómetro:
  rotaryValue = analogRead(rotaryPin);
  digitalWrite(ledPin, HIGH); // Liga o LED
  delay(rotaryValue);
  digitalWrite(ledPin, LOW);  // Desliga o LED
  delay(rotaryValue);
}
```

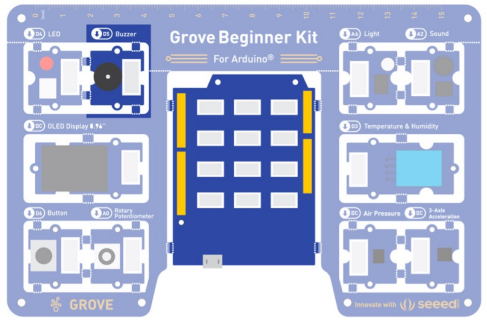


Componentes envolvidos:

1. Arduino
2. Módulo LED
3. Módulo Rotary Switch
4. Cabos (se necessário)

Oficina 4: Produzir Som no Buzzer

```
// Trabalhar com o Buzzer

const int BuzzerPin = 5;       // pin do buzzer
const int Potentiometer = A0; // pin do potenciómetro

void setup() {
  pinMode(BuzzerPin, OUTPUT);
}

void loop() {
  int potentioValue, Value;

  potentioValue = analogRead(Potentiometer);
  Value = map(potentioValue, 0, 1023, 0, 255);

  analogWrite(BuzzerPin, Value);
}
```

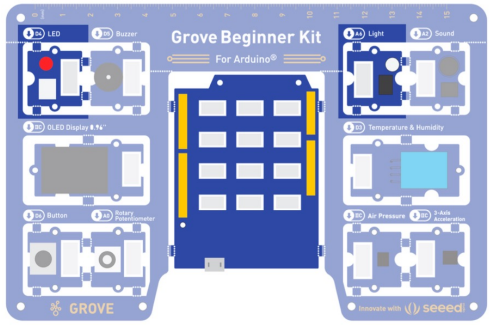


Componentes envolvidos:

1. Arduino
2. Módulo Buzzer
3. Cabos (se necessário)

Oficina 5: Controlar o LED com Sensor de Luz

```
const int sensorpin = A6;
const int ledPin = 4;

int sensorValue = 0;
int outputValue = 0;

void setup() {
  pinMode(ledPin,OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // Obtem o valor analógico do pin do sensor de luz
  sensorValue = analogRead(sensorpin);
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  Serial.println(sensorValue);
  // Altera do valor do pin do LED
  analogWrite(ledPin, outputValue);
  delay(30);
}
```

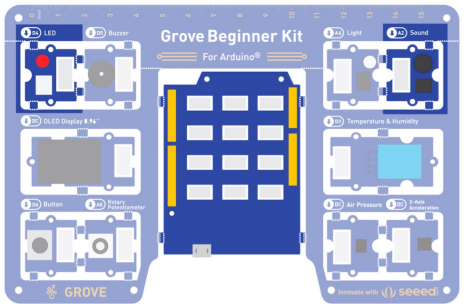


Componentes envolvidos:

1. Arduino
2. Módulo LED
3. Módulo Light Sensor
4. Cabos (se necessário)

Oficina 6: Controlar o LED com o Sensor de Som

```
const int soundPin = A2;
const int ledPin = 4;

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  int soundState = analogRead(soundPin);
  Serial.println(soundState);
  // Se o valor do sensor de som > 200, liga o LED durante 100 ms
  // Senão, desliga o LED
  if (soundState > 200) {
    digitalWrite(ledPin, HIGH);   // Liga o LED
    delay(100);                   // durante 100 milisegundos
  }else{
    digitalWrite(ledPin, LOW);    // Desliga o LED
  }
}
```

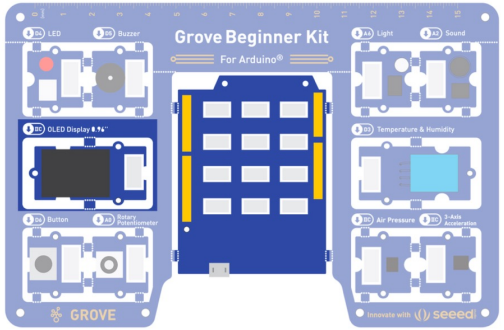


Componentes envolvidos:

1. Arduino
2. Módulo LED
3. Módulo Sound Sensor
4. Cabos (se necessário)

Oficina 7: Mostrar Dados no OLED Display

```
#include <Arduino.h>    // Inclusão da biblioteca arduino
#include <U8x8lib.h>    // Inclusão da biblioteca "U8g2" que
controla o OLED

// Define o objeto de programação que trabalha com o OLED
U8X8_SSD1306_128X64_ALT0_HW_I2C oled(U8X8_PIN_NONE);

void setup(void) {
  oled.begin();         // Inicializa o dispositivo OLED
  oled.setFlipMode(0);  // Define a rotação do display;
}

void loop(void) {
  oled.setFont(u8x8_font_chroma48medium8_r);  // Define a fonte
  oled.setCursor(0, 0);           // Posiciona o cursor de escrita
  oled.print("Hello World!");     // Escreve o texto
}
```

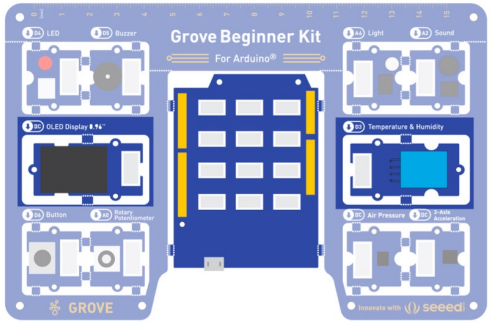


Componentes envolvidos:

1. Arduino
2. Módulo OLED
3. Cabos (se necessário)

Oficina 8: Sensor de Temperatura e Humidade

```
#include "DHT.h"      // Inclusão da biblioteca DHT11
#include <Arduino.h>  // Inclusão da biblioteca standard de controle do Arduino
#include <U8x8lib.h>  // Inclusão da biblioteca do OLED

#define DHTPIN 3     // Definição do pin do sensor de temperatura
#define DHTTYPE DHT11   // Definição do código do tipo de sensor DHT 11

DHT dht(DHTPIN, DHTTYPE); // definição de um objeto de programação
                          // para controlo do sensor de humidade e temperatura

U8X8_SSD1306_128X64_ALT0_HW_I2C u8x8(U8X8_PIN_NONE);// definição do objeto do OLED

// Bloco de definição
void setup(void) {
  Serial.begin(9600);             // Define a porta da consola serial
  Serial.println("DHTxx test!");
  dht.begin();                    // Inicializa o sensor de temperatura e humidade
  u8x8.begin();                   // Inicializa o OLED
  u8x8.setPowerSave(0);           // Desativa o modo power do display
  u8x8.setFlipMode(1);            // Define a orientação do ecrã
}
```

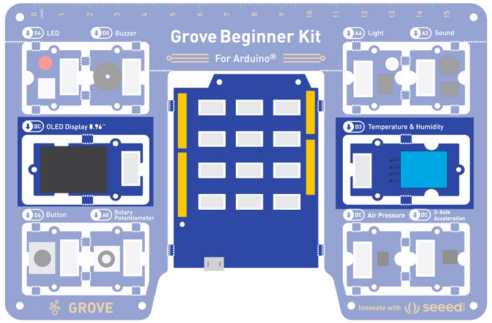


Componentes envolvidos:

1. Arduino
2. Módulo OLED
3. Módulo de Temperature & Humidity
4. Cabos (se necessário)

Oficina 8: Sensor de Temperatura e Humidade

```
// Bloco de execução
void loop(void) {
  float temp, humi;                 // Define as variáveis da temperatura e humidade

  temp = dht.readTemperature();     // Obtém o valor da temperatura do sensor
  humi = dht.readHumidity();        // Obtém o valor da humidade do sensor

  u8x8.setFont(u8x8_font_chroma48medium8_r);  // Define o tipo de letra do OLED
  u8x8.setCursor(0, 33);            // Posiciona o cursor de escrita no OLED

  u8x8.print("Temp:");              // Escreve
  u8x8.print(temp);                 // o valor da temperatura obtida no sensor
  u8x8.print("C");                  // no OLED

  u8x8.setCursor(0,50);             // Reposiciona o cursor no OLED
  u8x8.print("Humidity:");          // Escreve
  u8x8.print(humi);                 // o valor de percentagem de humidade
  u8x8.print("%");                 // no OLED

  u8x8.refreshDisplay();            // refresca o conteúdo o OLED
  delay(200);                       // Para durante 200 milisegundos
}
```

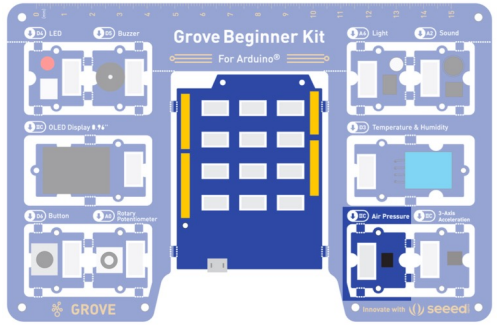


Componentes envolvidos:

1. Arduino
2. Módulo OLED
3. Módulo de Temperature & Humidity
4. Cabos (se necessário)

Oficina 9: Sensor de Pressão Atmosférica

```
#include "Seeed_BMP280.h"   // Incluir a biblioteca BMP280
#include <Wire.h>           // Incluir a biblioteca de comunicação I2C

BMP280 bmp280;  // Definição do objeto do sensor de pressão atmosférica

// Bloco de definições
void setup() {
    Serial.begin(9600); // Define a porta do consola serial
                        // Inicializa o sensor de pressão atmosférica

    if (!bmp280.init()) {     // e verifica (if) se tal foi possível
        Serial.println("Device not connected or broken!");  // Se não
                                                // informa na consola Serial
    }
}
```

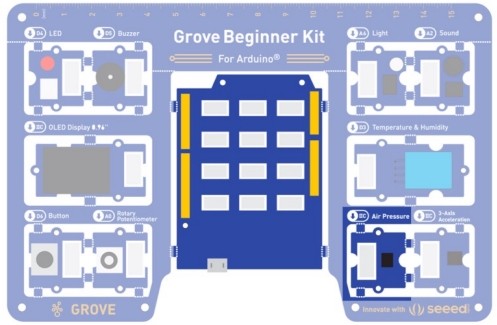


Componentes envolvidos:

1. Arduino
2. Módulo de Air Pressure
3. Cabos (se necessário)

Oficina 9: Sensor de Pressão Atmosférica

```
// Bloco de execução
void loop() {
    float pressure;    // Define a variável da pressão atmosférica

    Serial.print("Temp: ");  // Mostra o valor da temperatura obtida
    Serial.print(bmp280.getTemperature());  // do sensor de pressão atmosférica
    Serial.println("C");                    //(bmp280.getTemperature())em Celsius

    Serial.print("Pressure: "); // Mostra o valor da pressão atmosférica
    Serial.print(pressure = bmp280.getPressure()); // obtida do sensor
    Serial.println("Pa");                         // (bmp280.getPressure())em PA

    Serial.print("Altitude: ");   // Mostra o valor da altitude
    Serial.print(bmp280.calcAltitude(pressure));  // obtida do sensor
    Serial.println("m");                  // (bmp280.calcAltitude()) em metros

    Serial.println("\n");       //Acrescenta uma linha vazia entre medições
    delay(1000);

}
```

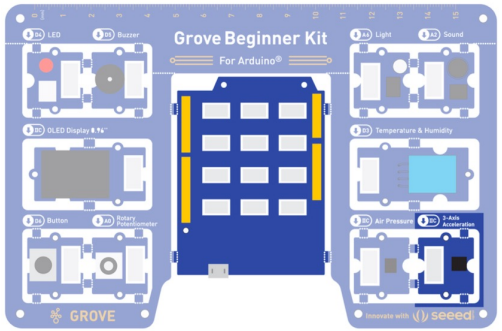


Componentes envolvidos:

1. Arduino
2. Módulo de Air Pressure
3. Cabos (se necessário)

Oficina 10: Sensor de Movimento /Gravidade (Acelerómetro)

```
#include "LIS3DHTR.h"     // Incluir a biblioteca LIST3DHTR do acelerómetro

#ifdef SOFTWAREWIRE       // Código de definição da comunicação por software
  #include <SoftwareWire.h>
  SoftwareWire myWire(3, 2);
  LIS3DHTR<SoftwareWire> LIS;   // Objeto de controle do acelerómetro
  #define WIRE myWire
#else                     // ou por hardware
  #include <Wire.h>
  LIS3DHTR<TwoWire> LIS;   // Objeto de controle do acelerómetro
  #define WIRE Wire
#endif

// Bloco de definições
void setup() {
  Serial.begin(9600);
  while (!Serial) {};   // Espera pela ativação da comunicação serial
  LIS.begin(WIRE, 0x19); // Inicializa o senso acelerómetro
  delay(100);
  // Define a velocidade de comunicação 50HZ
  LIS.setOutputDataRate(LIS3DHTR_DATARATE_50HZ);
}
```

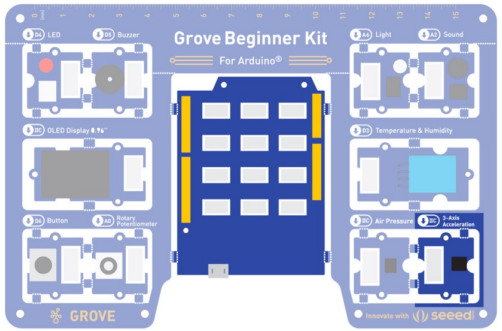


Componentes envolvidos:

1. Arduino
2. Módulo de 3-Axis Acceleration
3. Cabos (se necessário)

Oficina 10: Sensor de Movimento /Gravidade (Acelerómetro)

```
// Bloco de execução
void loop() {
  if (!LIS) {     // Se não houver comunicação com o acelerómetro
    Serial.println("LIS3DHTR didn't connect.");   // Informa na consola serial
    while (1);
    return;         // e termina
  }

  //Obtém os dados do acelerómetro e apresenta-os na consola serial
  Serial.print("x:");
  Serial.print(LIS.getAccelerationX());   // Obtém (LIS.getAccelerationX()) e mostra
  Serial.print("  ");                     // o parametro X da gravidade

  Serial.print("y:");
  Serial.print(LIS.getAccelerationY());   // Obtém (LIS.getAccelerationY()) e mostra
  Serial.print("  ");                     // o parametro Y da gravidade

  Serial.print("z:");
  Serial.println(LIS.getAccelerationZ()); // Obtém (LIS.getAccelerationZ()) e mostra
                                          // o parametro Z da gravidade
  delay(500);
}
```



Componentes envolvidos:

1. Arduino
2. Módulo de 3-Axis Acceleration
3. Cabos (se necessário)