2023

Oficina de Formação

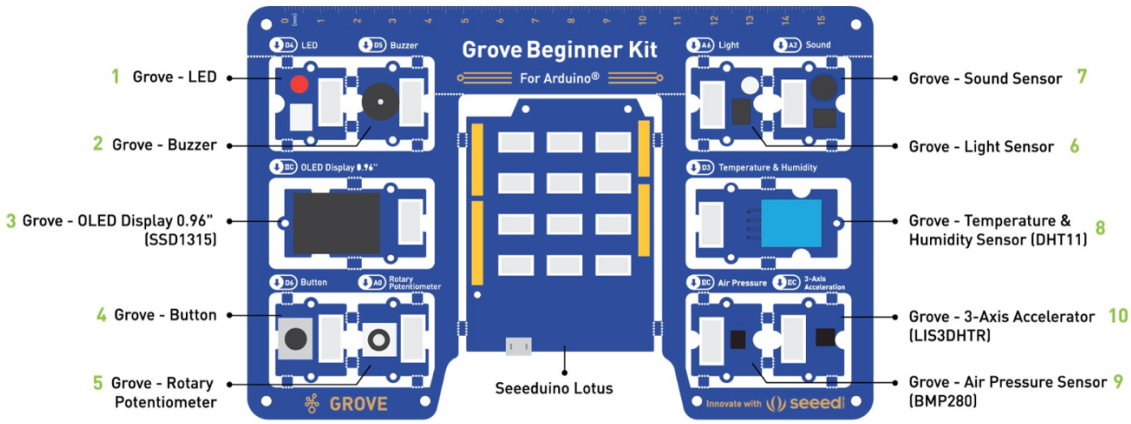
INTRODUÇÃO À ROBÓTICA E ELETRÓNICA COM ARDUINO

FAB LAB HENRIQUES NOGUEIRA - MÁRIO VIANA

Índice

O Kit Arduino e Sensores

O ***Grove Beginner Kit*** constitui uma peça de hardware que facilita a aprendizagem da utilização da plataforma Arduino e da sua ligação a diversos sensores. Constitui-se como uma única peça de eletrónica, podendo, numa segunda instância, utilizar-se todos os componentes de forma isolada em projetos futuros. A imagem seguinte apresenta kit e identifica os diversos sensores disponibilizados:



Por defeito os módulos de sensores e atuadores estão conectados ao Arduino através do PCB. Isto significa que não é necessário ligar cabos. A tabela seguinte apresenta os módulos e os pins de ligação definidos por defeito:

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)

Instalação do Software Necessário

O IDE do Arduino

A primeira tarefa consiste em instalar o software de programação. Para tal deve aceder ao site do arduino, https://www.arduino.cc/en/software, e descarregar o software de instalação do ***Arduino IDE***.

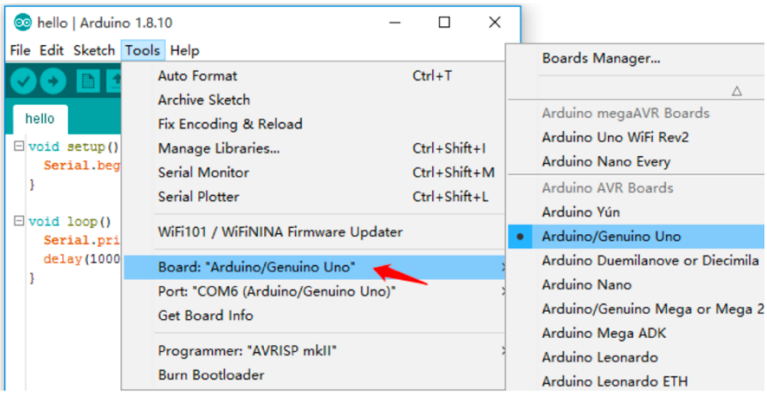
O Driver USB de ligação do Arduino

É necessário, também, instalar o driver de conexão do Arduino ao computador. Assim, devemos descarregar o driver USB (***CP2102 USB Driver***) disponibilizado no sítio https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads .

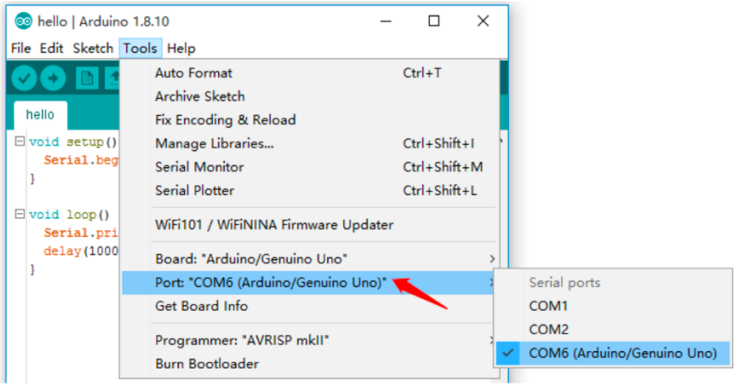
Testar o IDE e a ligação ao Arduino

Depois de instalados o software necessário, devemos executar os seguintes passos de forma a configurar e testar a ligação do computador ao *kit Grove*:

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 (NB: a identificação da porta pode não ser exatamente como é mostrado na figura):



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
}
```

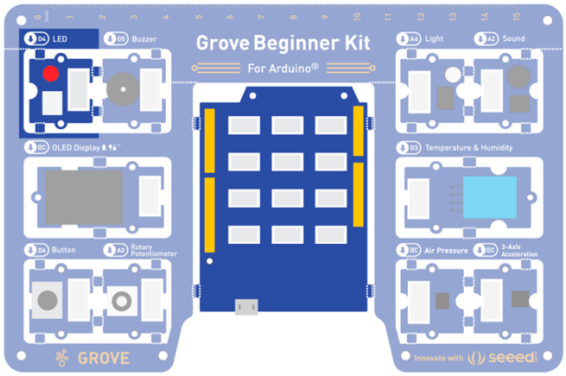
6. Depois de digitado o código, é necessário compilá-lo, clicando no ícone  ; é ainda necessário corrigir os erros, caso existam.

7. Depois da compilação, é necessário efetuar o upload o programa para o Arduino, clicando no ícone  .
8. Para verificar o funcionamento do programa devemos aceder ao **menu -> Serial Monitor**. Como podemos verificar, abrir-se-á uma janela/secção onde aparece a mensagem "hello, world".

Oficina 1: LED

Componentes envolvidos

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



O código

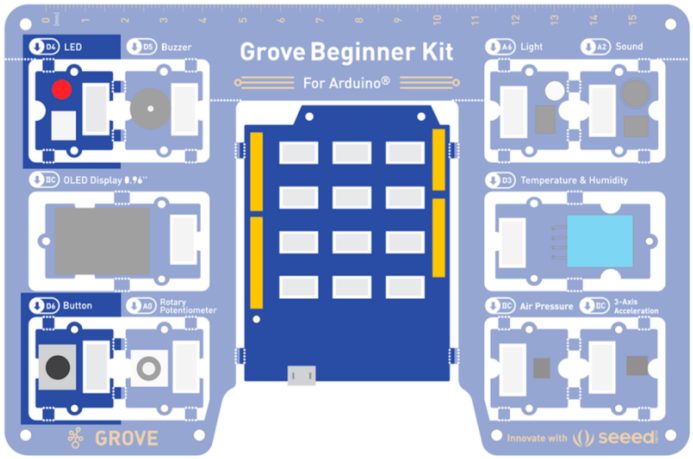
```
//LED Blink
// O LED liga e desliga em intervalos de 1 segundo

int ledPin = 4; // define uma variável com o valor do pin onde está ligado o LED (pin 4)
```

```
// Bloco de inicialização e definição
void setup() {
    pinMode(ledPin, OUTPUT);  // define o tipo de ligação (OUTPUT) ao PIN do LED (pin 4)
}

//Bloco de execução que se repete indefinidamente
void loop() {
    digitalWrite(ledPin, HIGH); // Liga o LED = coloca o valor HIGH (5V) no pin 4
    delay(1000);                // Para 1 segundo (= 1000 milisegundos)
    digitalWrite(ledPin, LOW);  // Desliga o LED = coloca o valor LOW (0V) no pin 4
    delay(1000);                // Para 1 segundo
}
```

Oficina 2: Ligar o LED com o Botão

Componentes envolvidos

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



O código

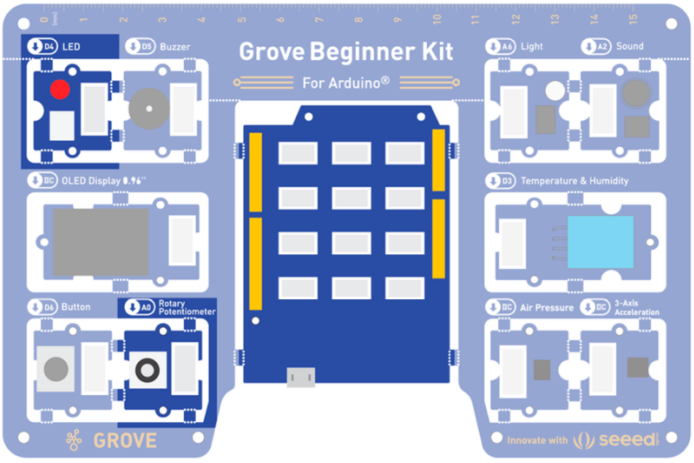
```
// Botão liga e desliga o LED
// As constantes não permitem a alteração do seu valor.
// São muito usadas para definir os números dos pins

const int buttonPin = 6;  // definição do número do pin do botão
const int ledPin =  4;    // definição do número do pin do LED

// Por seu lado, as variáveis podem ser alteradas
int buttonState = 0;      // variáveis para guardar o estado do botão
                          // pressionado (HIGH) e libertado (LOW)

// 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);
}

// Bloco de execução
void loop() {
  // Lê o estado do botão:
  buttonState = digitalRead(buttonPin);
  // Verifica se o botão está pressionado (HIGH):
  if (buttonState == HIGH) {
    // Liga o LED:
    digitalWrite(ledPin, HIGH);
  } else {  // Se não
    // Desliga o LED:
    digitalWrite(ledPin, LOW);
  }
}
```

Oficina 3: Controlar a velocidade do blink do LED

Componentes envolvidos

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



O código

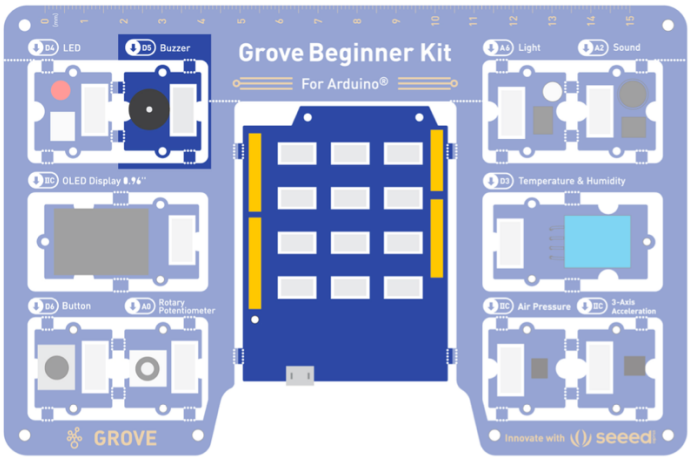
```
//Controlar piscar LED com potenciómetro
const int rotaryPin = A0;    // Define o pin do potenciómetro
const int ledPin = 4;        // Define o pin do LED
int rotaryValue = 0;  // Define a variável que guarda o valor do potenciómetro
```

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

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

Oficina 4: Produzir Som no Buzzer

Componentes envolvidos

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



O código

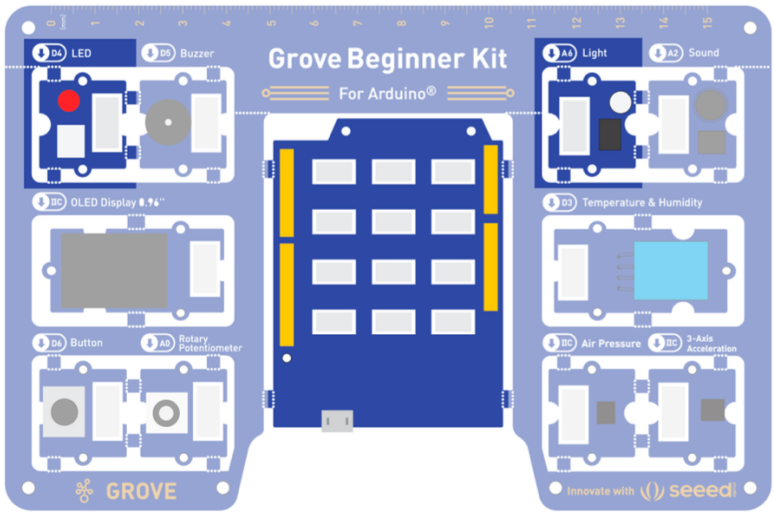
```
// Trabalhar com o Buzzer

const int BuzzerPin = 5;      // Define o pin de ligação do buzzer
const int Potentiometer = A0; // Define o pin de ligação do potenciómetro

// Bloco de definições
void setup() {
  pinMode(BuzzerPin, OUTPUT); // Define o tipo de interação com o pin do buzzer (output)
```

```
}

// Bloco de execução
void loop() {
  // define as variáveis que trabalham com os valores do potenciómetro
  int potentioValue, Value;
  potentioValue = analogRead(Potentiometer);    // Obtém o valor do pin do potenciómetro

  Value = map(potentioValue, 0, 1023, 0, 255);  // Adapta o valor do potenciómetro para
                                                // a escala de valores do buzzer

  analogWrite(BuzzerPin, Value);                // Coloca o valor de ativação do buzzer
}
```

Oficina 5: Controlar o LED com Sensor de Luz

Componentes envolvidos

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



O código

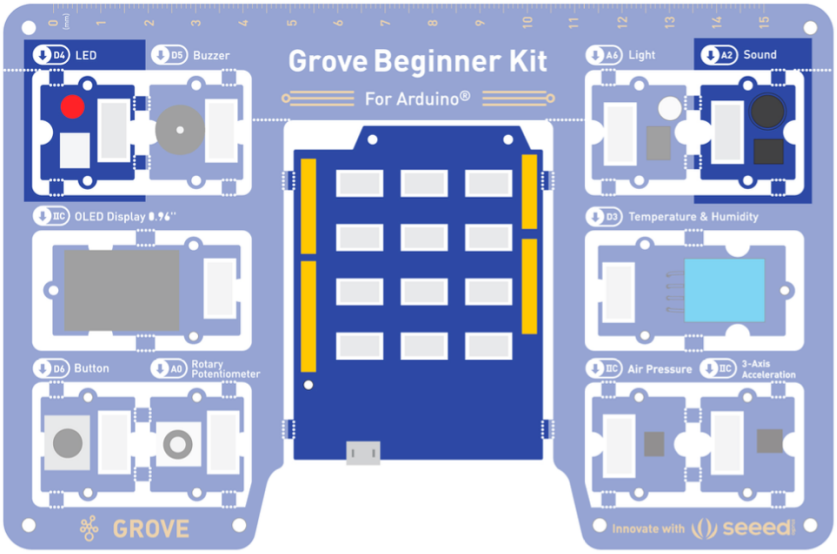
```
// Trabalhar com o sensor de luz (Light Switch)

const int sensorpin = A6; // Define o pin do sensor de luz (A6)
const int ledPin = 4;
int sensorValue = 0;      // Define a variável que guarda o valor do senso de luz
int outputValue = 0;      // Define a variável que guarda o valor de ativação do LED

// Bloco de definição
void setup() {
  pinMode(ledPin,OUTPUT); // Define o tipo de interação com o pin do LED
```

```
  Serial.begin(9600);     // Ativa a porta de ligação do monitor Seral
}

// Bloco de execução
void loop() {
  // Obtem o valor analógico do pin do sensor de luz
  sensorValue = analogRead(sensorpin);
  // Converte o valor obtido na escala de valores válidos para o LED
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // MOstra o valor do sensor de luz no monitor serial
  Serial.println(sensorValue);
  // Altera do valor do pin do LED
  analogWrite(ledPin, outputValue);
  delay(30);
}
```

Oficina 6: Controlar o LED com o Sensor de Som

Componentes envolvidos

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



O código

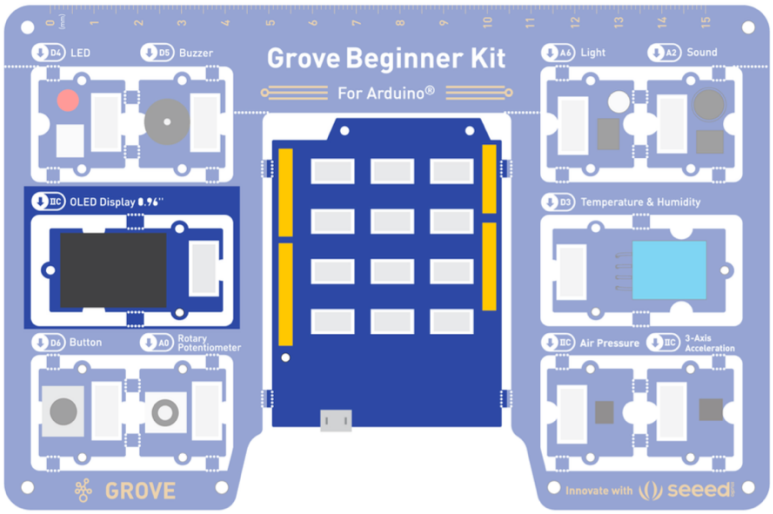
```
// Controlar LED com Sensor de Som

const int soundPin = A2;  // Definição do pin do sensor de som
const int ledPin = 4;     // Definição do pin do LED

// Bloco de definições
void setup() {
```

```
  pinMode(ledPin, OUTPUT);  // Definição do tipo de operação no pin do LED (OUTPUT)
  Serial.begin(9600);       // Definição da porta de ligação ao consola serial
}

// Bloco de execução
void loop(){
  int soundState = analogRead(soundPin);  // Obtém o valor do sensor de som
  Serial.println(soundState);             // Mostra o valar na consola serial

  // Se o valor do sensor de som é maior que 200, liga o LED durante 100 milissegundos
  // Senão, desliga o LED
  if (soundState > 200) {
    digitalWrite(ledPin, HIGH);   // Liga o LED
    delay(100);                   // durante 100 milissegundos
  }else{
    digitalWrite(ledPin, LOW);    // Desliga o LED
  }
}
```

Oficina 7: Mostrar Dados no OLED Display

Componentes envolvidos

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



Para trabalhar com o dispositivo OLED é necessário instalar/ativar a biblioteca "**U8g2**". Para tal é necessário:

1. Aceder ao menu **Sketch -> Include Library -> Manage Libraries…**;
2. Pesquisar a biblioteca através da palavra-chave "U8g2";
3. Selecionar a biblioteca e clicar no botão [Instalar]

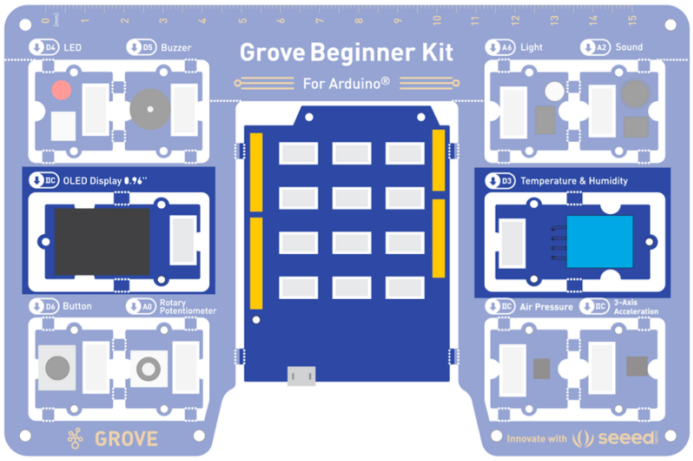
O código

```
/ Trabalhar com o display OLED

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

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

// Bloco das definições
void setup(void) {
  oled.begin();          // Inicializa o dispositivo OLED
  oled.setFlipMode(0);   // Define a rotação do display;
                         // os valores podem ser 0 (0 graus) ou 1 (180 graus)
}

// Bloco de execução
void loop(void) {
  oled.setFont(u8x8_font_chroma48medium8_r);  // Define a fonte do texto do display
  oled.setCursor(0, 0);                       // Posiciona o curso de escrita
  oled.print("Hello World!");                 // Escreve o texto "Hello World!" no display
}
```

Oficina 8: Trabalhar com o sensor de Temperatura e Humidade

Componentes envolvidos

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



Para trabalhar com o sensor de temperatura e humidade é necessário instalar/ativar a biblioteca "DHT11" que disponibiliza todas as funções associadas a este sensor. Para tal é necessário:

1. Aceder ao menu **Sketch -> Include Library -> Manage Libraries…**;

2. Pesquisar a biblioteca através da palavra-chave "**Grove Temperature and Humidity Sensor(DHT11)**";
3. Selecionar a biblioteca e clicar no botão [Instalar]

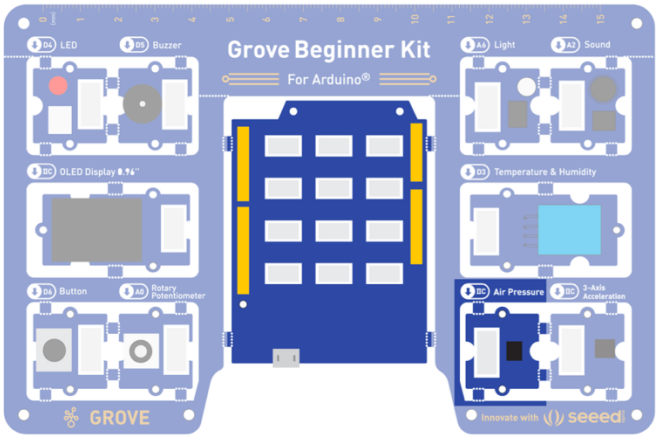
O código

```
// Trabalhar com o 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 onde está ligado o 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 de programação
para controlo 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
}

// Bloco de execução
void loop(void) {
  float temp, humi;               // Define as variáveis que guardam os valores da temperatura
e da 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 para escrever
humidade
  u8x8.print("Humidity:");                    // Escreve
  u8x8.print(humi);                           // o valor de percentagem de humidade obtida no
sensor
  u8x8.print("%");                            // no OLED
  u8x8.refreshDisplay();                      // refresca o conteúdo o OLED
  delay(200);                                 // Para durante 200 milissegundos
}
```

Oficina 9: Trabalhar com o Sensor de Pressão Atmosférica

Componentes envolvidos

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



Para trabalhar com o sensor de medição da pressão atmosférica é necessário instalar/ativar a biblioteca "**BMP280**" que disponibiliza todas as funções associadas a este sensor. Para tal é necessário:

4. Aceder ao menu **Sketch -> Include Library -> Manage Libraries…**;
5. Pesquisar a biblioteca através da palavra-chave "**Grove BMP280**";
6. Selecionar a biblioteca e clicar no botão [Instalar]

O código

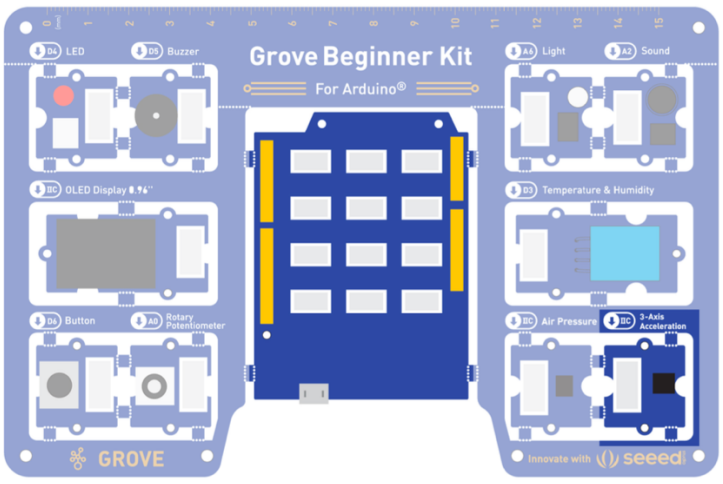
```
// Detetar o valor da pressão atmosférica

#include "Seeed_BMP280.h"      // Incluir a biblioteca BMP280
#include <Wire.h>              // Incluir a biblioteca que trabalha com o sistema de comunicação
I2C

BMP280 bmp280;                 // Definição do objeto de programação que inclui
                               // todas as funcionalidades 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 foi possível,
                                                            // informa na consola Serial
    }
}

// Bloco de execução
```

```
void loop() {
    float pressure;    // Define a variável que guarda o valor da pressão atmosférica

    Serial.print("Temp: ");  // Mostra na consola serial, o valor da temperatura obtida
    Serial.print(bmp280.getTemperature());  // no sensor de pressão atmosférica
    Serial.println("C");                    //(bmp280.getTemperature())em Celsius


    Serial.print("Pressure: "); // Mostra na consola serial, o valor da pressão atmosférica
    Serial.print(pressure = bmp280.getPressure());  // obtida no sensor de pressão atmosférica
    Serial.println("Pa");                          // (bmp280.getPressure())em PA

    Serial.print("Altitude: ");   // Mostra na consola serial, o valor da altitude
    Serial.print(bmp280.calcAltitude(pressure));  // obtida no sensor de pressão atmosférica
    Serial.println("m");                          // (bmp280.calcAltitude()) em metros
    Serial.println("\n");       //Acrescenta um alinha vazia entre medições
    delay(1000);
}
```

Oficina 10: Trabalhar com o Sensor de Movimento/Gravidade (Acelerómetro)

Componentes envolvidos

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



Para trabalhar com o acelerómetro é necessário instalar/ativar uma biblioteca "**3-Axis Digital Accelerometer (+-2g to 16g**" que disponibiliza todas as funções associadas a este sensor. Esta biblioteca deverá ser obtida através da plataforma **GitHub**. Para tal é necessário:

1. Descarregar do GitHub (**https://github.com/Seeed-Studio/Seeed_Arduino_LIS3DHTR**) o ficheiro ZIP da biblioteca;
2. Aceder ao menu **Sketch -> Include Library -> Add .ZIP library…** e importar o ficheiro ZIP descarregado;

O código

```
//Trabalhar com a gravidade através de um acelerómetro

#include "LIS3DHTR.h"     // Incluir a biblioteca LIST3DHTR que controla o acelerómetro

#ifdef SOFTWAREWIRE       // Código de definição da comunicação por software
  #include <SoftwareWire.h>
  SoftwareWire myWire(3, 2);
  LIS3DHTR<SoftwareWire> LIS;   // Objeto de programação que permite controlar o acelerómetro
  #define WIRE myWire
#else                     // ou se não for possível, a comunicação é realizada por hardware
  #include <Wire.h>
  LIS3DHTR<TwoWire> LIS;   // Objeto de programação que permite controlar o acelerómetro
  #define WIRE Wire
#endif

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

// Bloco de execução
void loop() {
  if (!LIS) {     // Se não hover 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 parâmetro X da gravidade
  Serial.print("y:");
  Serial.print(LIS.getAccelerationY());   // Obtém (LIS.getAccelerationY()) e mostra
  Serial.print("  ");                     // o parâmetro Y da gravidade
  Serial.print("z:");
  Serial.println(LIS.getAccelerationZ()); // Obtém (LIS.getAccelerationZ()) e mostra
                                          // o parâmetro Z da gravidade
  delay(500);
}
```