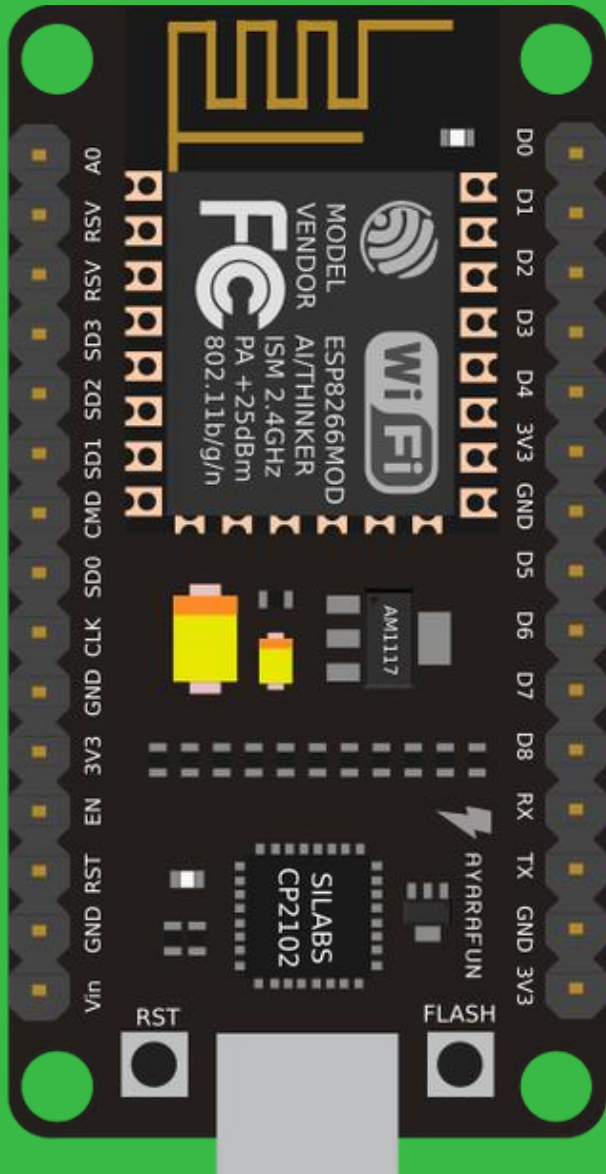


WACKER
{ school }



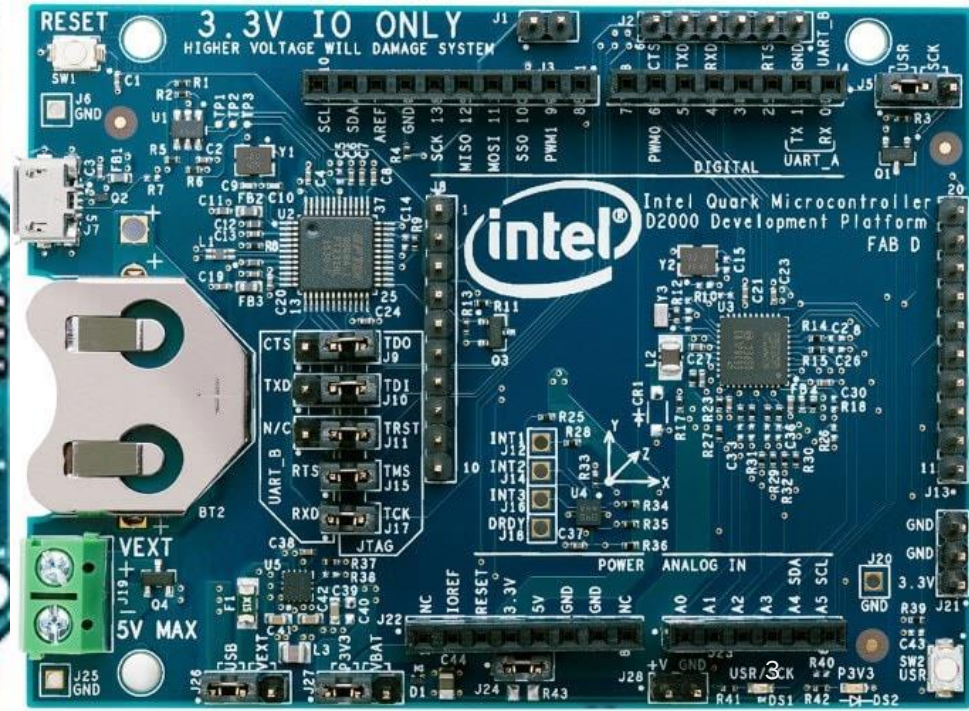
WORKSHOP IOT

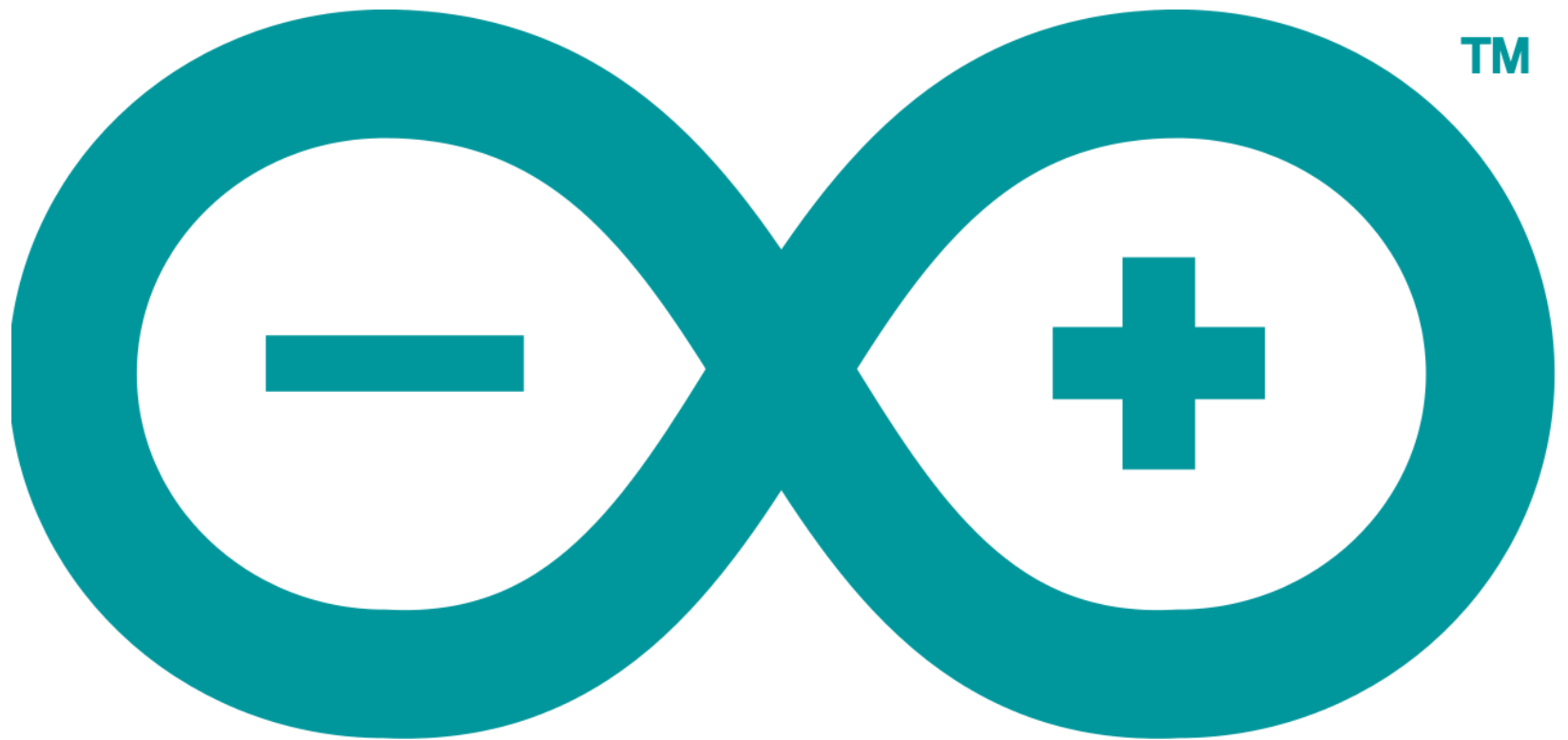
Relógio com estação
meteorológica



O que é um microcontrolador?

Pequeno computador feito num único circuito integrado





ARDUINO

Alimentação
do Arduino

USB

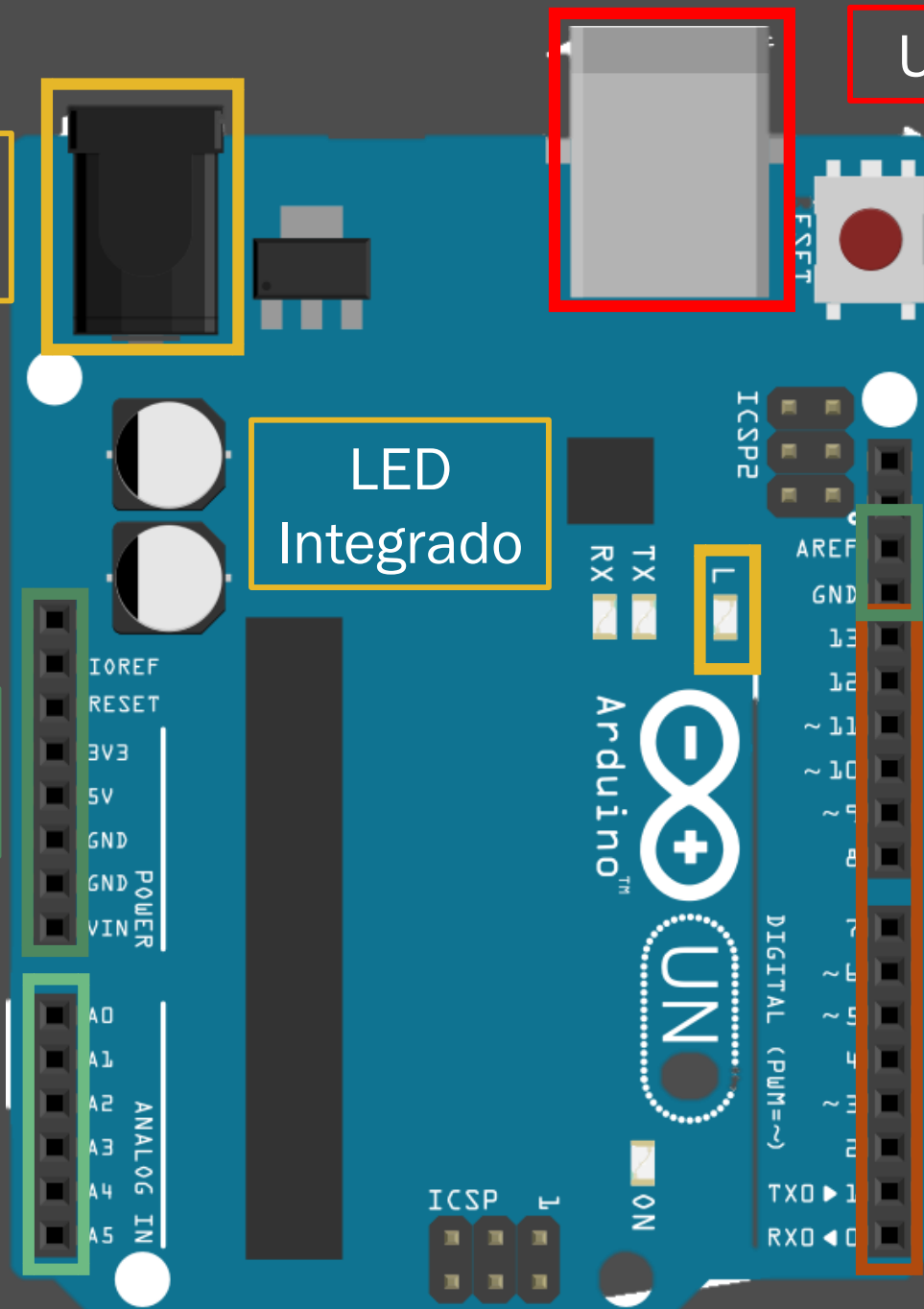
LED
Integrado

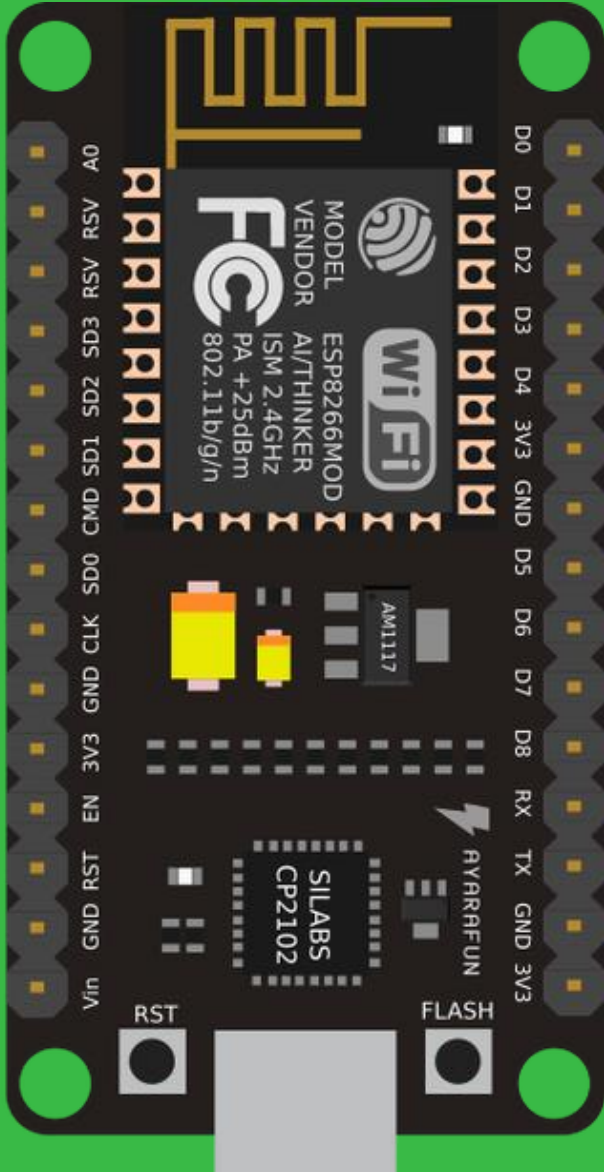
Zona de
alimentação

Zona de
alimentação

Pins digitais

Pins
Analógicos





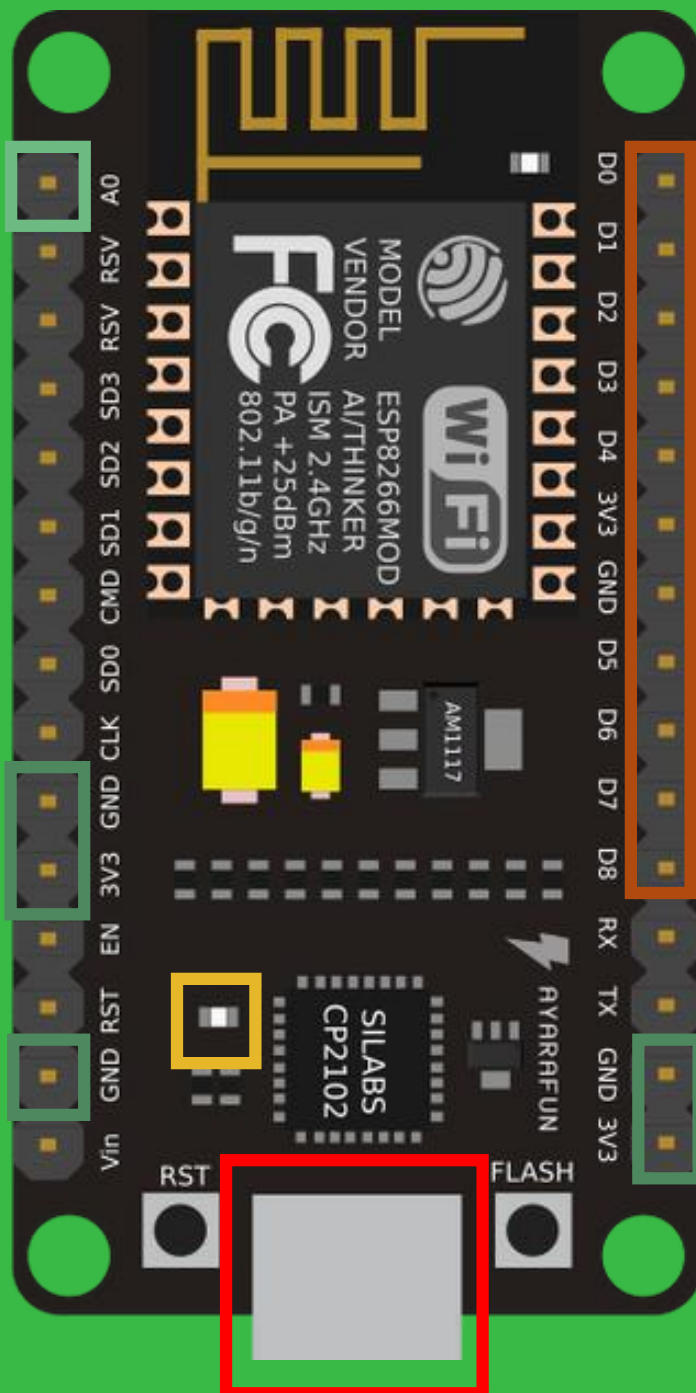
Diferenças:

- WiFi
- Funciona a 3,3V
- Indexação dos pin's

NODEMCU

Pino
Analógico

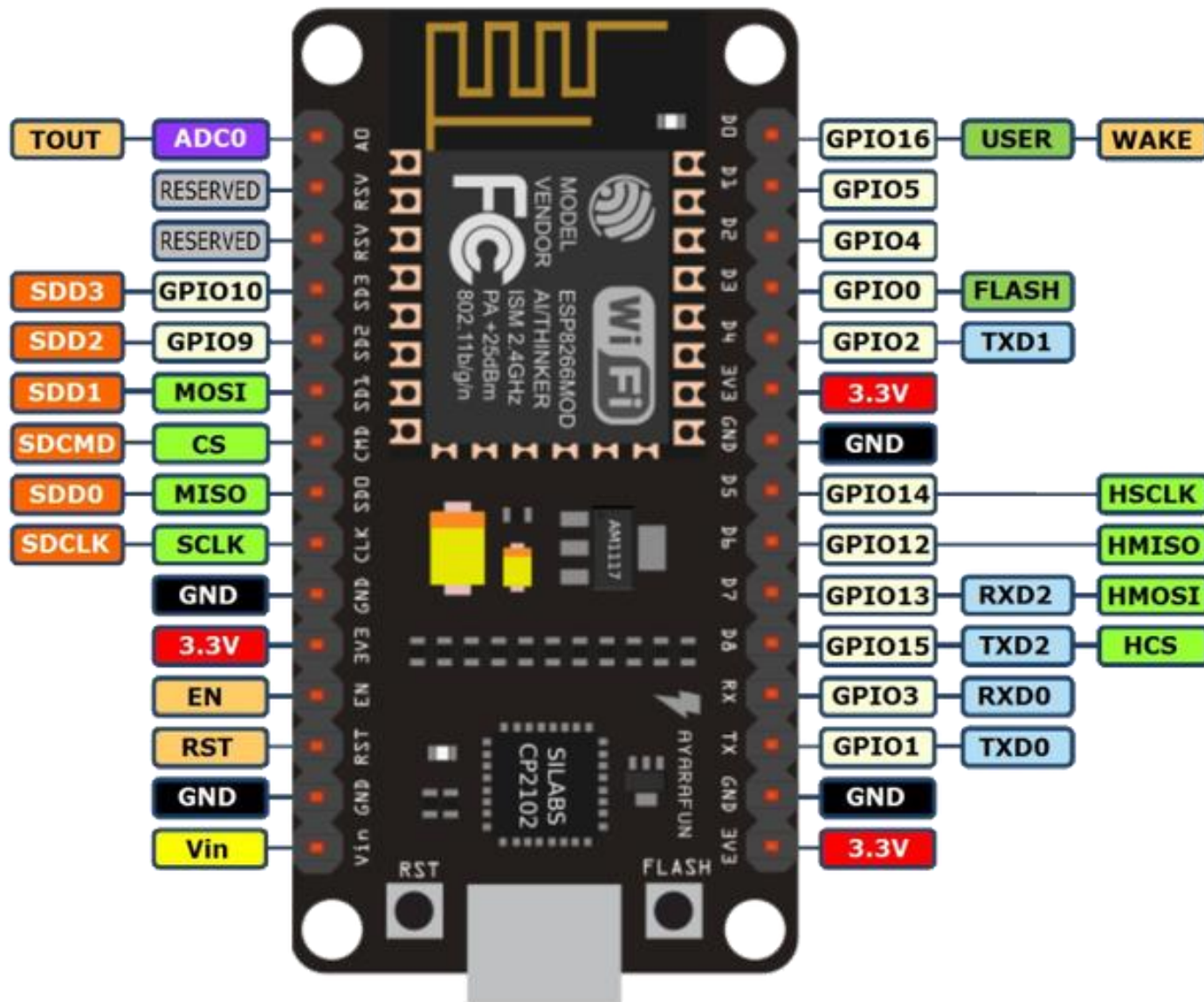
LED
Integrado



Pins
digitais

Zona de
alimentação

USB



INSTALAÇÃO DO NODEMCU

No Arduino IDE

Ficheiro Editar Rascunho Ferramentas Ajuda



sketch_feb04a

```
1 void setup() {  
2   // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8  
9 }
```

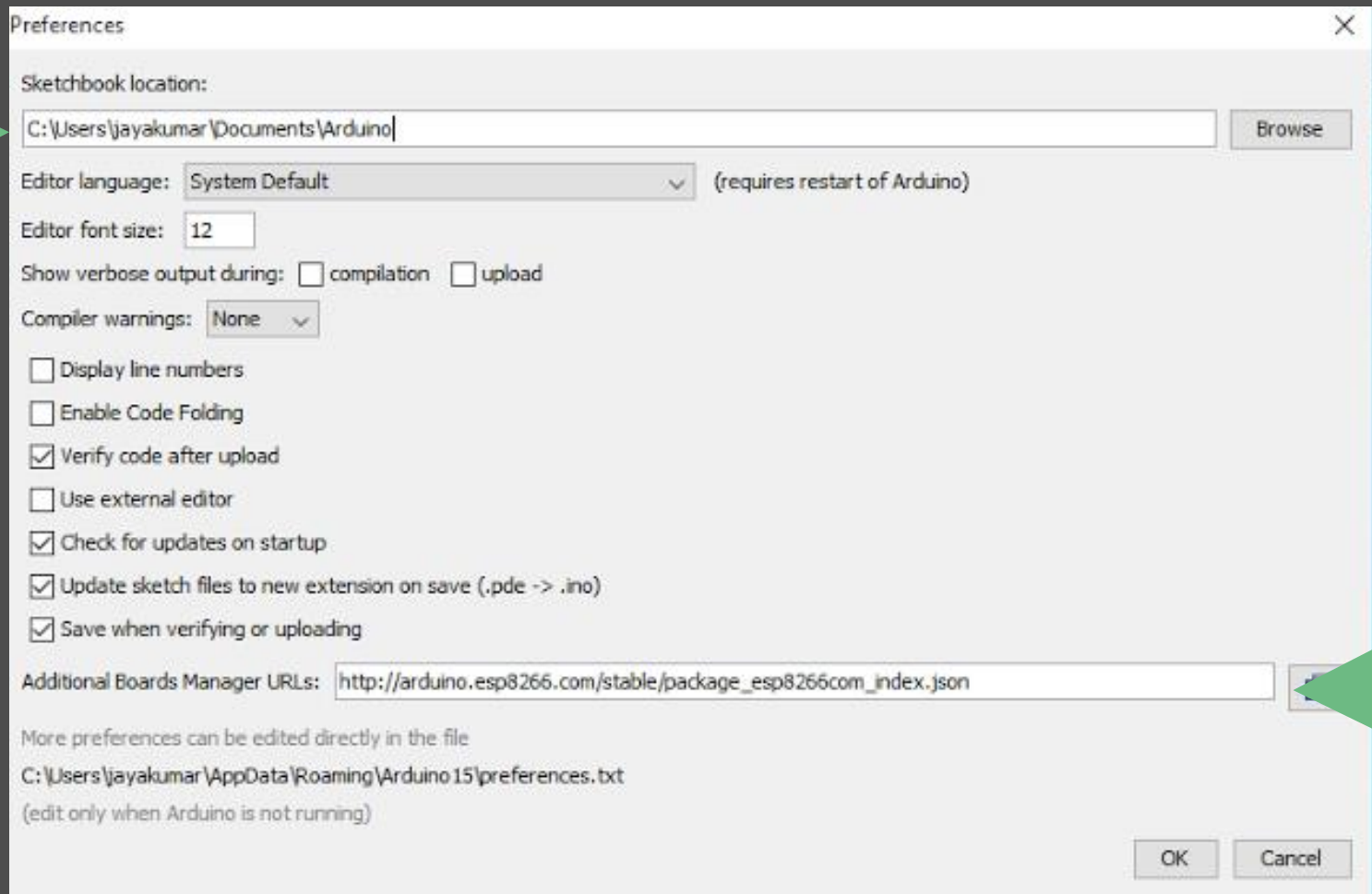
sketch_feb04a | Arduino 1.8.5

Ficheiro Editar Rascunho Ferramentas Ajuda

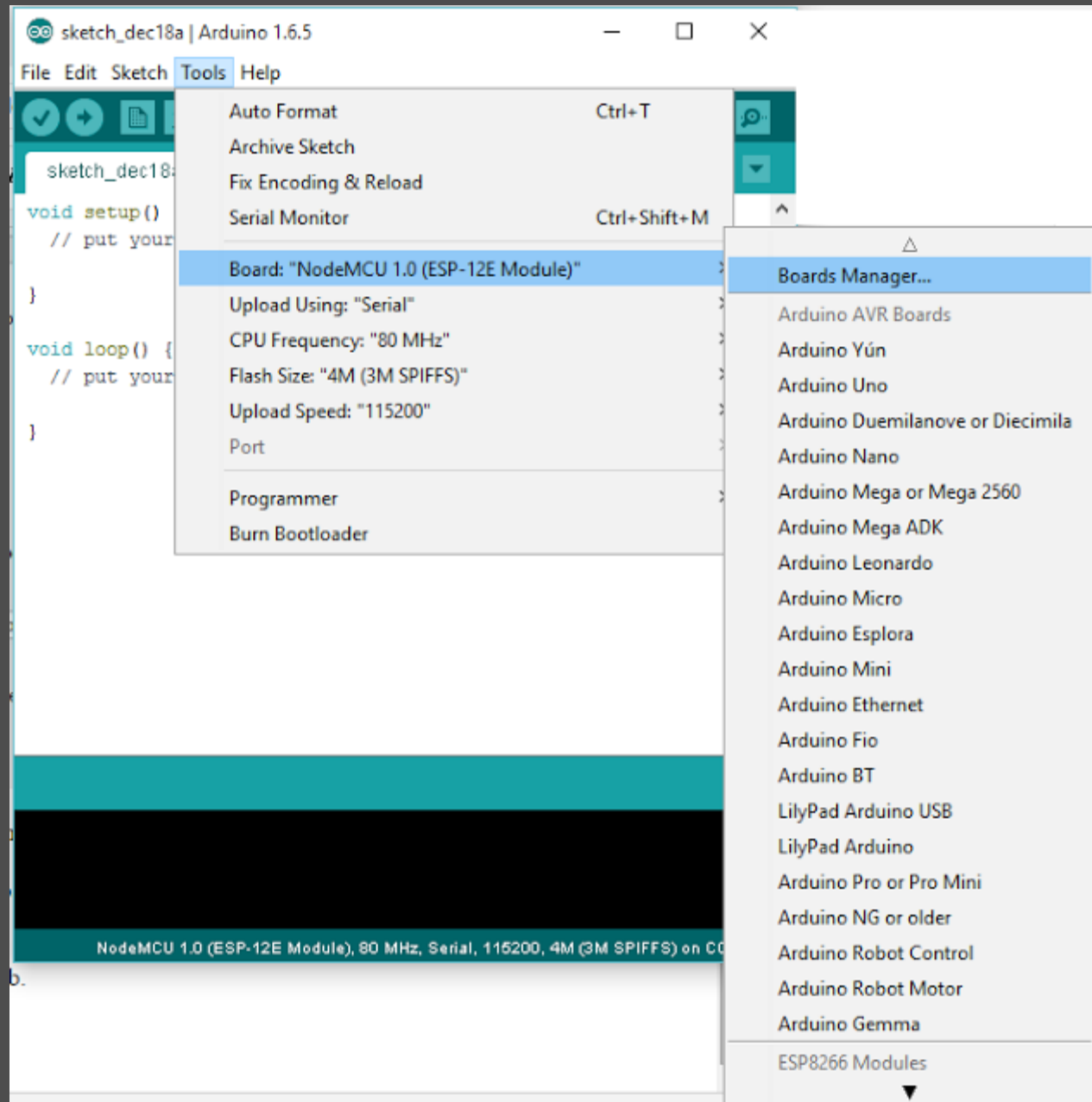
Novo	Ctrl+N
Abrir...	Ctrl+O
Abrir Recentes	>
Bloco de rascunhos	>
Exemplos	> eun
Fechar	Ctrl+W
Guardar	Ctrl+S
Guardar como...	Ctrl+Shift+S
Configuração da Página	Ctrl+Shift+P
Imprimir	Ctrl+P
Preferências	Ctrl+Comma
Sair	Ctrl+Q

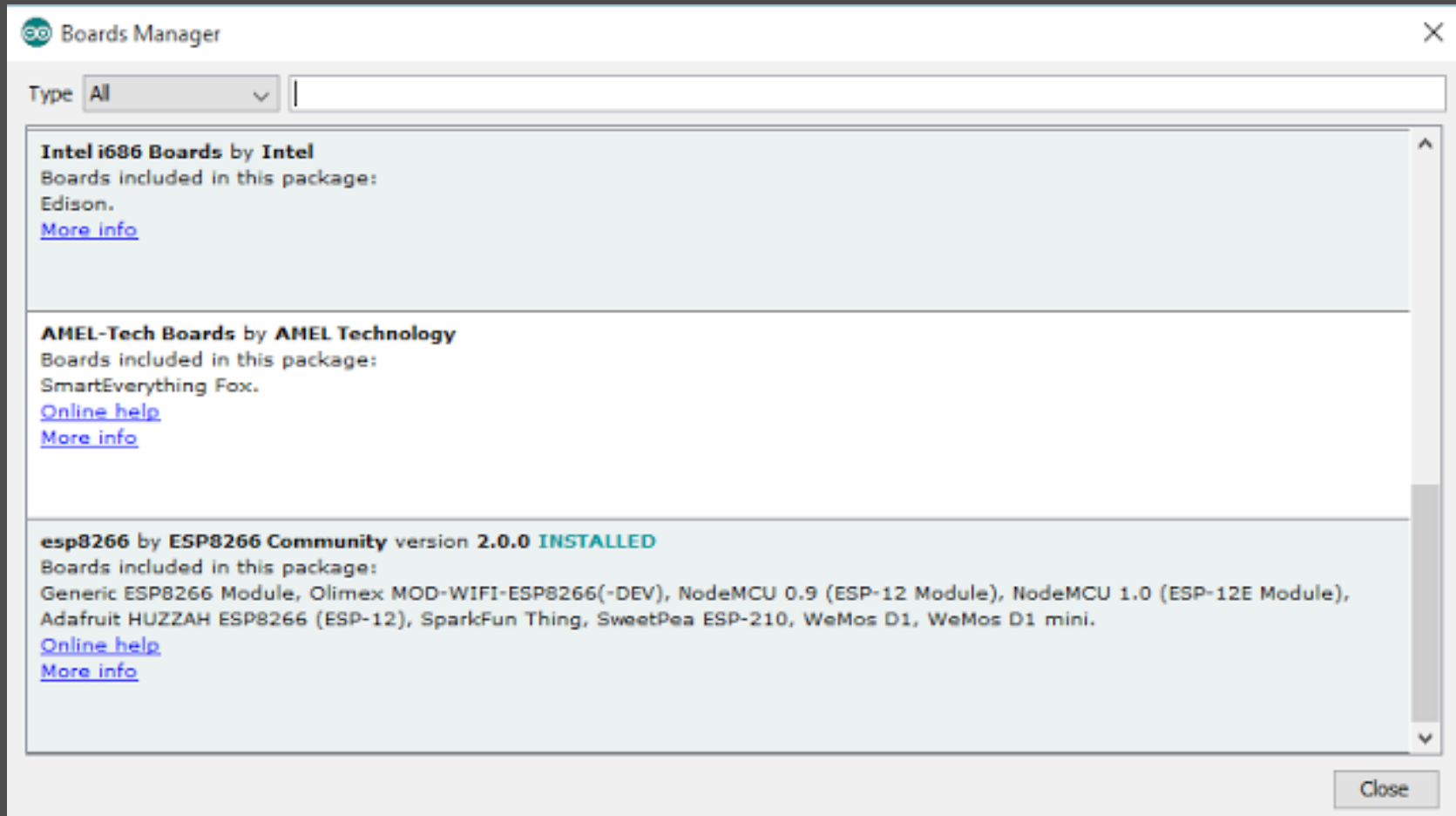
Clicar em Preferências

Permite mudar a localização de onde são gravados os programas



Permite mudar o fornecedor de bibliotecas de placas





Escrever esp8266 na caixa de pesquisa

PRIMEIROS PASSOS

No Arduino IDE

Funções iniciais

```
void setup() {  
  // put your setup code here, to run once:  
  
}
```

```
void loop() {  
  // put your main code here, to run repeatedly:  
  
}
```

CONTROLAR UM LED

No Arduino IDE, abrir “1º Exercício”

Funções Básicas

```
void setup() {  
    pinMode(pin, mode);    // Configura o comportamento do pino,  
                             // como pino de entrada ou saída  
}  
  
void loop() {  
    digitalWrite(pin, value); // Coloca o pino em estado de “1” (3,3V)  
                                // ou “0” (0V)  
  
    delay(ms)                // Para o microcontrolador durante x  
                                // milissegundos  
}
```


1º Exercício

To do list:

1. Definir o LED_BUILTIN como output
2. Ligar durante 2 segundo, desligar durante 1

Controlar um LED

```
void setup() {  
  // Inicializa o LED integrado como um pino de saída  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(LED_BUILTIN, LOW);           // Liga o LED  
  delay(1000);                             // Espera 1 segundo  
  digitalWrite(LED_BUILTIN, HIGH);          // Desliga o LED  
  delay(2000);                             // Espera 2 segundos  
}
```

ESCREVER PARA A CONSOLA

No Arduino IDE, abrir

“2º Exercício – escrever para a consola”

Funções Básicas

```
void setup() {  
  // Inicia o canal de comunicação em série  
  Serial.begin(velocidade);  
}  
  
void loop() {  
  // Escreve para o monitor de série e adiciona uma linha  
  Serial.println(variável);  
  
  // Lê bits do monitor de série  
  incomingByte = Serial.read();  
}
```

Velocidades:

- 300
- 600
- 1200
- 2400
- 4800
- **9600**
- 14400
- 19200
- 28800
- 38400
- 57600
- **115200**
- etc

Escrever para a consola

```
void setup() {  
  
  // Inicializa o LED integrado como um pino de saída  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
  
  digitalWrite(LED_BUILTIN, LOW);           // Liga o LED  
  delay(1000);                             // Espera 1 Segundo  
  
  digitalWrite(LED_BUILTIN, HIGH);          // Desliga o LED  
  delay(2000);                             // Espera 2 segundos  
}
```


2º Exercício

1. Iniciar a comunicação em série na velocidade 115200
2. Indicar que o LED está apagado
3. Indicar que o LED está ligado

Escrever para a consola

```
void setup() {  
  Serial.begin(115200);  
  // Inicializa o LED integrado como um pino de saída  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
  Serial.println("LED on");  
  digitalWrite(LED_BUILTIN, LOW);           // Liga o LED  
  delay(1000);                               // Espera 1 Segundo  
  Serial.println("LED off");  
  digitalWrite(LED_BUILTIN, HIGH);          // Desliga o LED  
  delay(2000);                               // Espera 2 segundos  
}
```

LIGAR AO WIFI

No Arduino IDE, abrir

“3º Exercício – ligar ao WiFi”

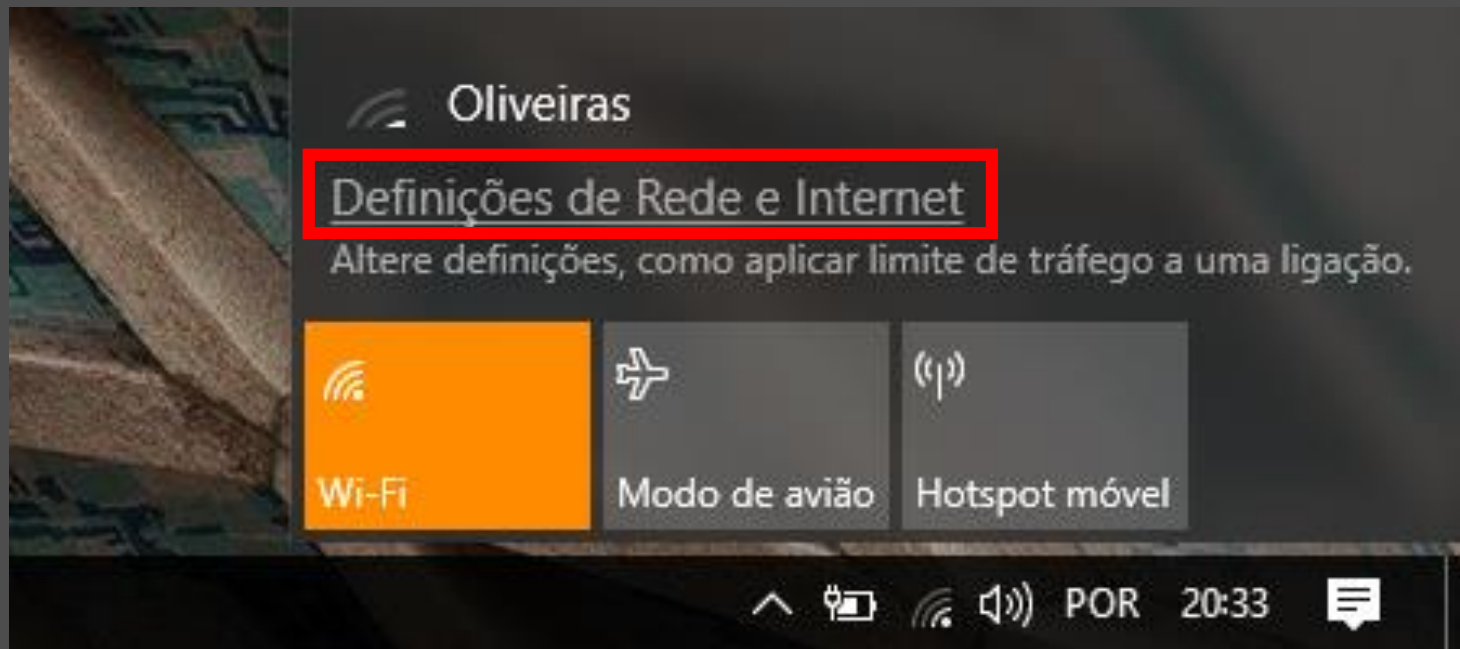


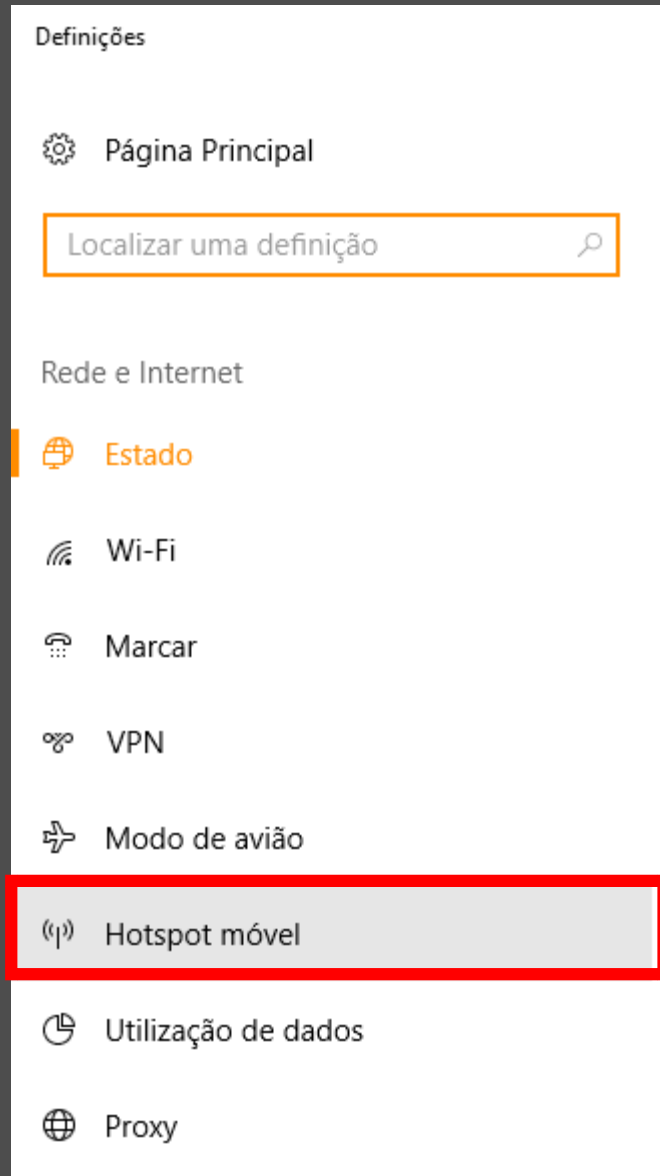
LIGAR A PARTILHA DE WIFI

Windows 10

CLICAR

Definições de rede e internet





CLICAR

Hotspot móvel

Hotspot móvel

Hotspot móvel

Partilhar a minha ligação à Internet com outros dispositivos

☐ Desligado

Partilhar a minha ligação à Internet de

Wi-Fi

Nome de rede: Chico

Palavra-passe de rede: 12345678

Banda de rede: Qualquer disponível

Editar

SSID = “Chico”

Palavra-passe = “12345678”

CLICAR

Ligar Hotspot móvel

Biblioteca Básicas

```
#include <ESP8266WiFi.h>
```

```
void setup() {  
  // Conectar ao wifi  
  WiFi.begin(ssid, password)  
}
```

```
void loop() {  
  
}
```

Ligar ao WiFi

```
#include <ESP8266WiFi.h>
```

```
const char* ssid = "";
```

```
const char* password = "";
```

```
void setup() {  
    delay(10);
```

```
// Connect to WiFi network
```

```
while (WiFi.status() != WL_CONNECTED) {  
    delay(500);  
    Serial.print(".");  
}
```

```
Serial.println("");  
Serial.println("WiFi connected");
```

```
void loop() {
```

```
}
```

3º Exercício

1. Preencher SSID e password
2. Iniciar a comunicação em série em 115200
3. Indicar no monitor de série qual o nome da rede que se está a conetar
4. Iniciar o wifi
5. Indicar que o wifi está bem ligado

Ligar ao WiFi

```
#include <ESP8266WiFi.h>

const char* ssid = "<Nome do WiFi>";
const char* password = "<password>";

void setup() {
    Serial.begin(115200);
    delay(10);

    // Connect to WiFi network
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);
```

```
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

}

void loop() {

}
```

CONTROLAR O LED NA NET

No Arduino IDE, abrir

“4º Exercício – controlo na web”

Controlar o LED na Net

```
#include <ESP8266WiFi.h>

WiFiServer server(80);  // Cria o objeto do servidor

void setup() {
  // Inicia o servidor
  server.begin();

  // Retorna o endereço IP do servidor
  WiFi.localIP()

}
```


Controlar o LED na Net

```
void loop() {  
  // Manuseamento dos clients do servidor  
  // Verifica se há algum cliente ligado  
  WiFiClient client = server.available();  
  
  // Le o botão que o cliente presionou  
  String request = client.readStringUntil('\r');  
  
  // Imprime para o webserver – funciona igual ao Serial.println  
  client.println()  
}
```

4º Exercício

-

1ª parte

1. Definir o ssid e password
2. Iniciar comunicação em série na porta 115200
3. Estabelecer a ligação ao Wifi
4. Iniciar o servidor
5. Imprimir o endereço IP
6. Imprimir que está tudo iniciado

Controlar o LED na Net

```
#include <ESP8266WiFi.h>

const char* ssid = "";
const char* password = "";

WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  delay(10);

  // Connect to WiFi network
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server

  Serial.println("Server started");

  // Print the IP address
  Serial.print("Use this URL to connect: ");
  Serial.print("http://");

  Serial.println("/");
}
```

4º Exercício

-

2ª parte

1. Atribuir um valor a “value” para controlar o website
2. Mudar o estado do LED
3. Imprimir o estado do LED no site

Controlar o LED na Net

```
#include <ESP8266WiFi.h>

const char* ssid = "";
const char* password = "";

WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  delay(10);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);

  // Connect to WiFi network
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
```

```
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

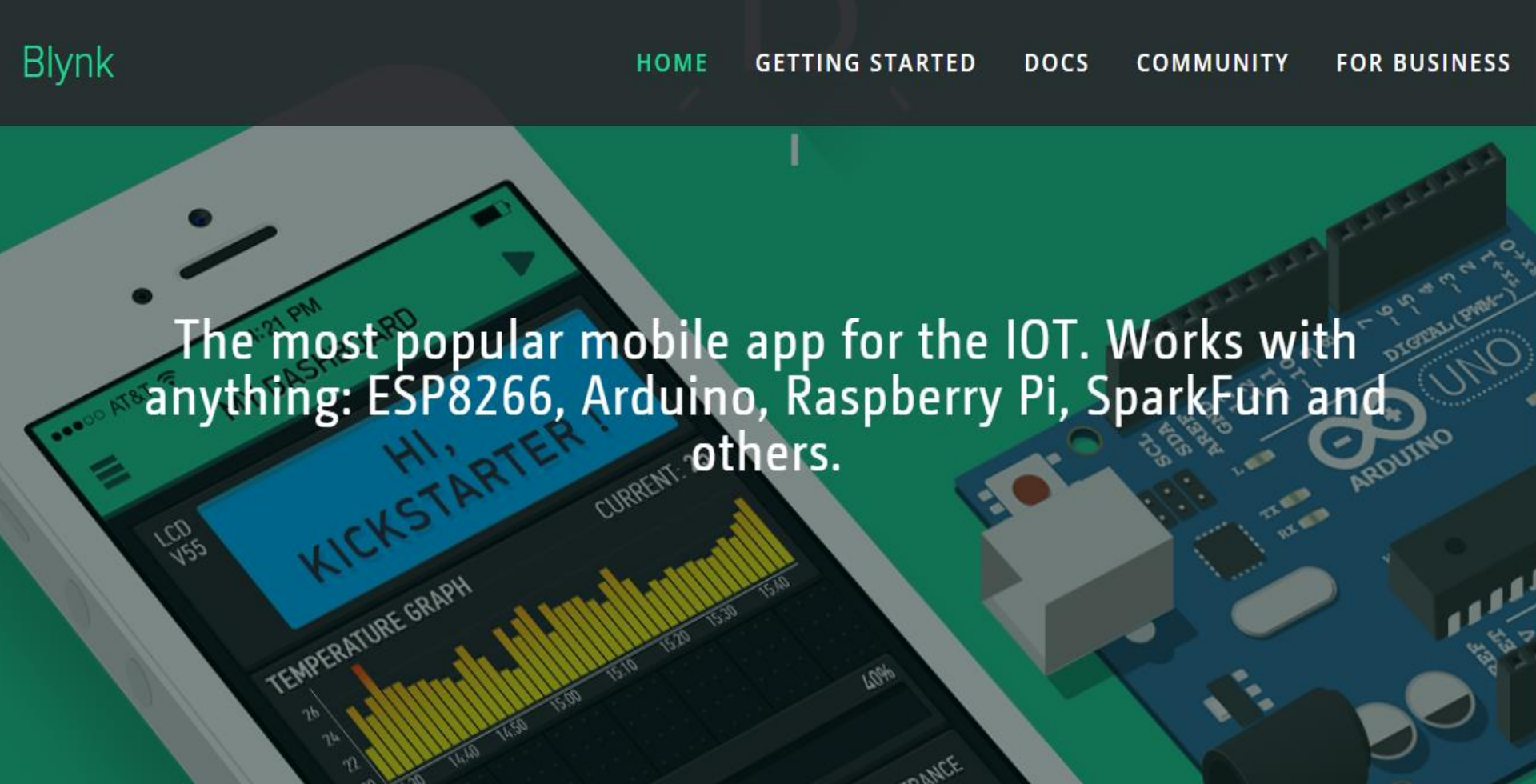
// Start the server
server.begin();
Serial.println("Server started");

// Print the IP address
Serial.print("Use this URL to connect: ");
Serial.print("http://");
Serial.print(WiFi.localIP());
Serial.println("/");
}
```

BLYNK - BOTÃO

No Arduino IDE, abrir

“5º Exercício – controlo móvel”



The most popular mobile app for the IoT. Works with anything: ESP8266, Arduino, Raspberry Pi, SparkFun and others.

Every project made with Blynk can be branded, and published to App Store and Google Play with your icon and app name.

Interested? Click [here](#)



BLYNK



Blynk is an Internet of Things platform with a drag-n-drop mobile application builder that allows to visualize sensor data and control electronics remotely in minutes. Blynk IoT cloud solution is open-source. Blynk hardware libraries support Arduino, Genuino, Raspberry Pi, Particle Photon, Electron, SparkFun.

Funções Básicas

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "YourAuthToken"; // Escrever "" para wifi sem pass
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

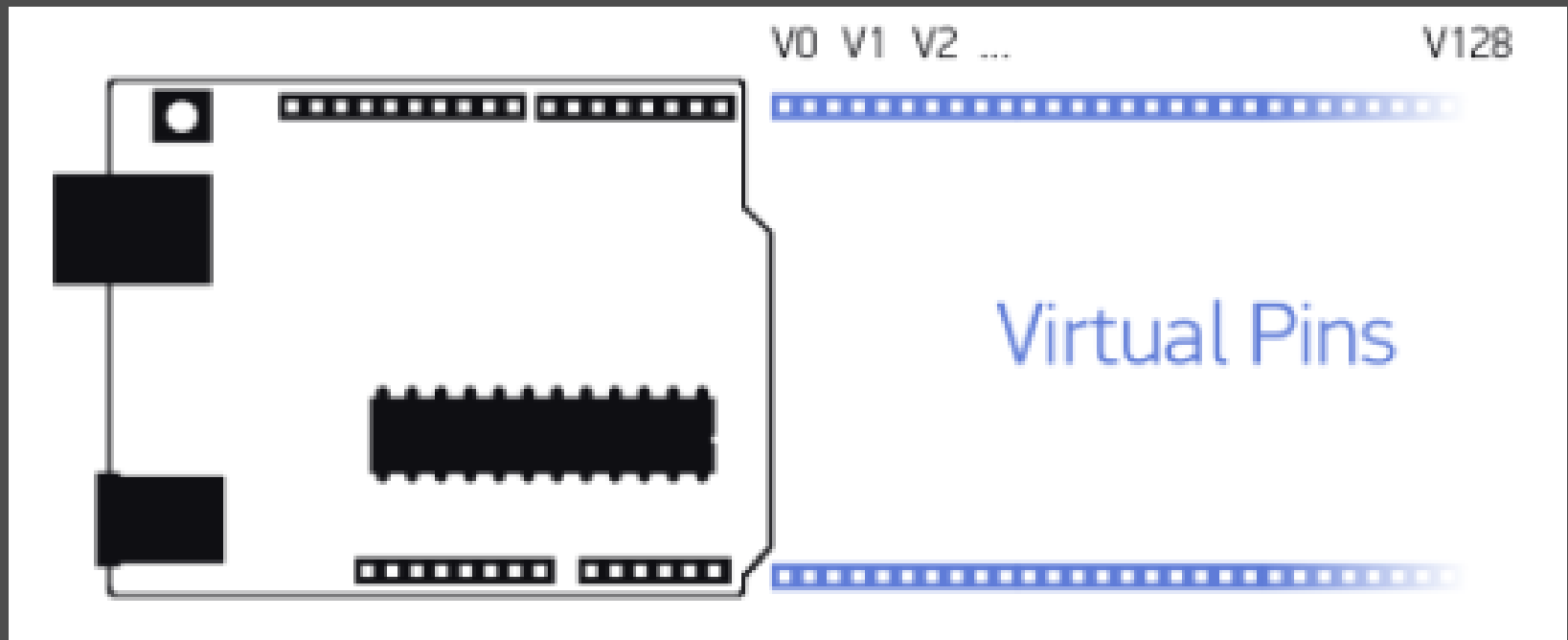
void setup() {
  // Inicializa a comunicação em série
  Serial.begin(9600);

  // Inicia a comunicação do wifi e liga-se ao servidor do Blynk
  Blynk.begin(auth, ssid, pass);
}
```

Funções Básicas

```
void loop() {  
  Blynk.run();  
  
  delay(ms) // EVITAR AO MÁXIMO CUSTO, usar timers  
}
```

Virtual Pins



Funcionam como pinos do arduino que só existem no servidor

Controlar um pino virtual na app

// Esta função é chamada sempre que existe um novo evento relacionado com este pino na app

```
BLYNK_WRITE(V1) {
```

```
    int pinValue = param.asInt(); // converte o valor da app para uma variável no código
```

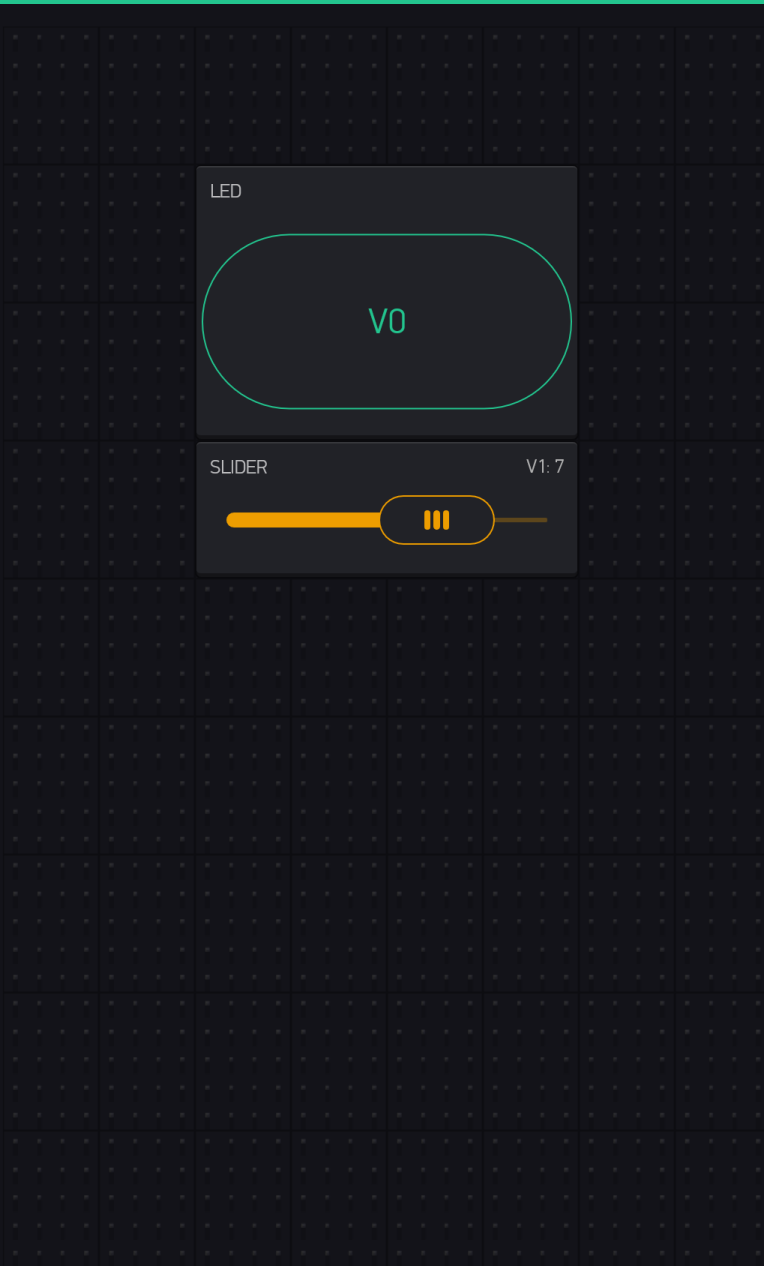
```
}
```

5º Exercício

-

1ª parte

1. Ir a <https://examples.blynk.cc/>
2. Selecionar a board – NodeMCU
3. Selecionar connection – wifi
4. Selecionar example – Virtual Pin Read
5. Copiar Código de exemplo para o Arduino
6. Criar projeto no Blynk



EXEMPLO DE APLICAÇÃO

5º Exercício

-

1ª parte

1. Criar duas funções de comando de pins virtuais . V0 e V1
2. V0 é um botão e tem de fazer acender e apagar o LED_BUILTIN
3. V1 é um slider e deve fazer aparecer na aplicação o valor do slider

Controlo pelo Blynk

```
BLYNK_WRITE(V0) {
```

```
  pinValueV1 = param.asInt(); // Atribui o valor do pino V0 para  
  uma variavel
```

```
  Serial.println(pinValueV1);  
}
```

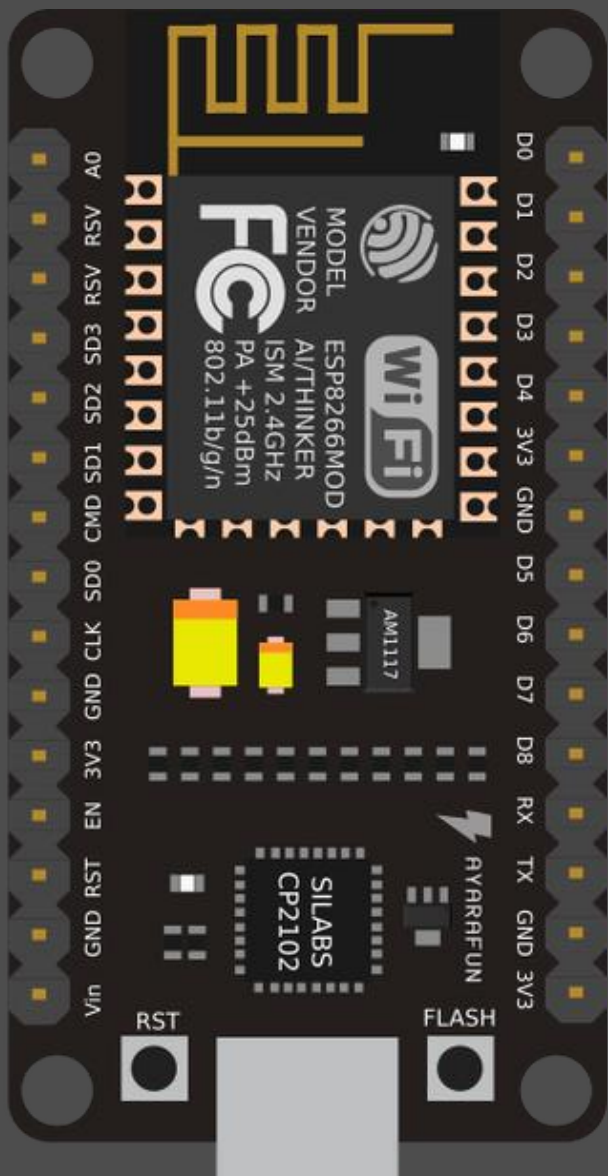
```
BLYNK_WRITE(V1) {
```

```
  sliderValue = param.asInt(); // Atribui o valor do pino V0 para  
  uma variavel
```

```
  // Processa a nova informação  
  Serial.println(sliderValue);
```

```
}
```


WACKER
{ school }



2ª sessão

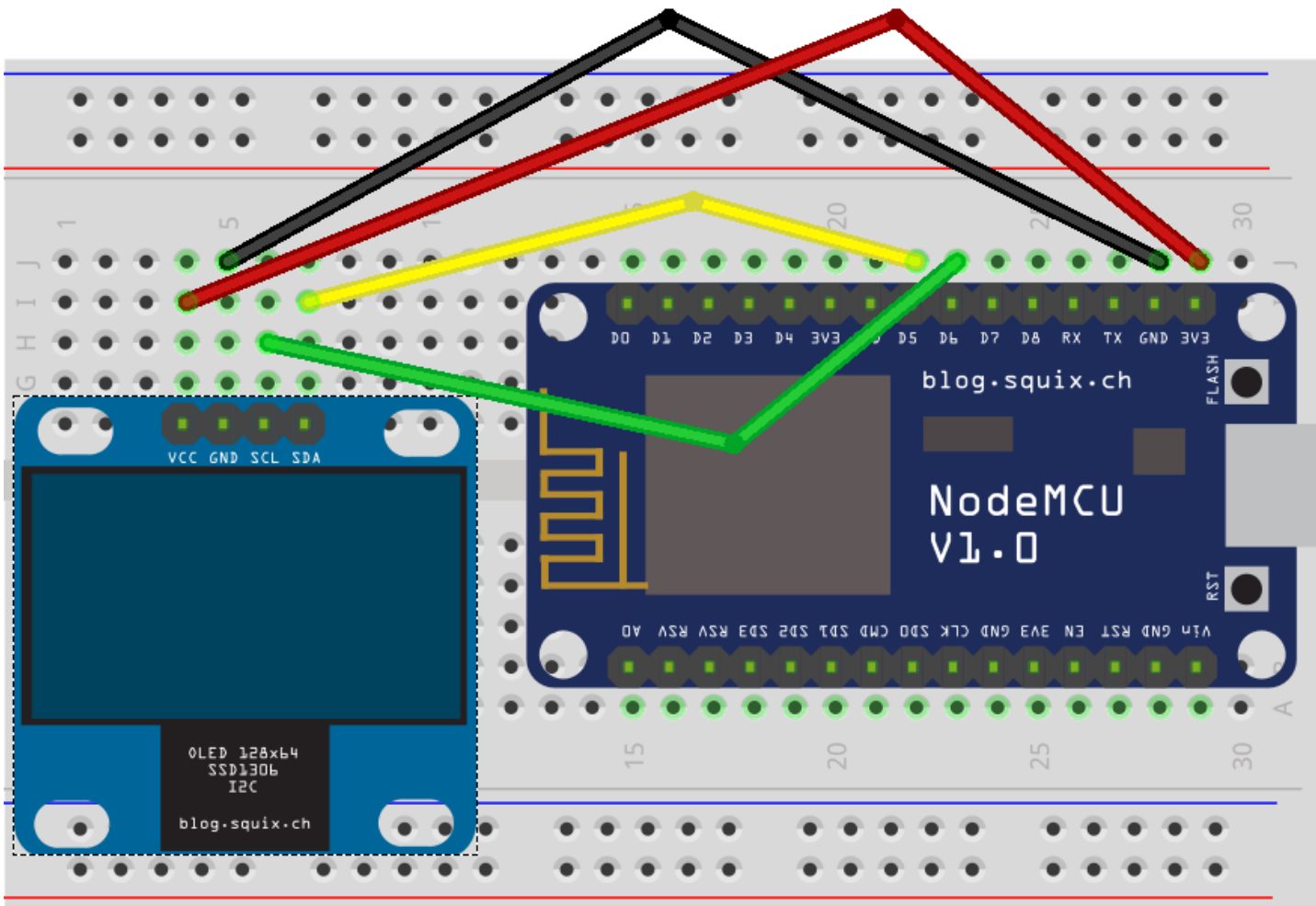
WORKSHOP IOT

Relógio com estação
meteorológica

CONTROLAR O LCD

No Arduino IDE, abrir

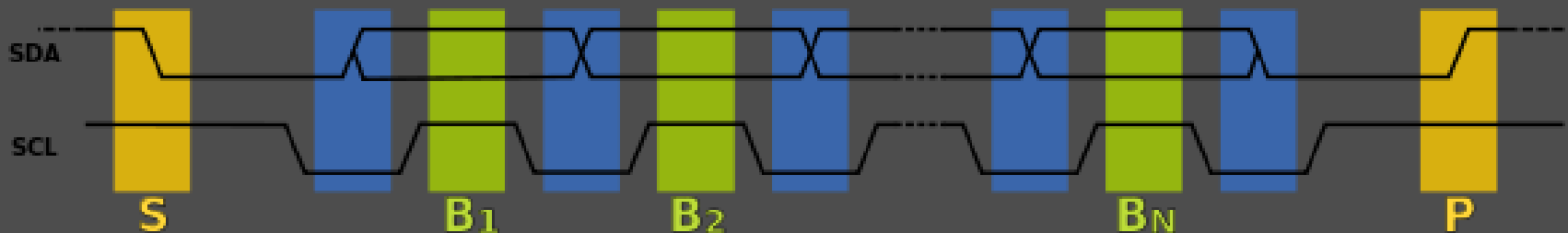
“6º Exercício – controlar LCD”



fritzing

Comunicação I²C

- Permite ligar dispositivos de baixa velocidade
 - *Microcontroladores*
 - *Sensores*
 - *Ecrãs*
- Usa apenas dois fios para conetar
 - *SDA – serial data*
 - *SCL – serial clock*
- Baixo custo



Inicialização

```
#include <SSD1306.h>
#include <OLEDDisplayUi.h>

// I2C_DISPLAY_ADDRESS, SDA_PIN, SCL_PIN
SSD1306 display(0x3c, D1, D2);
OLEDDisplayUi ui( &display );

void setup() {

}

void loop() {

}
```

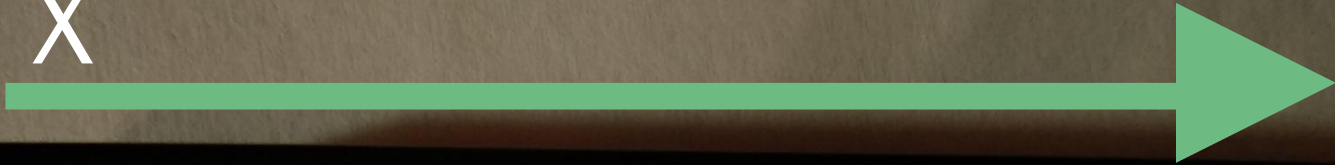
Inicialização

```
void setup() {  
  // Inicia a comunicação I2C com o LCD  
  display.init();  
  
  //Vira o ecrã ao contrário  
  display.flipScreenVertically();  
  
  //Define o contraste do ecrã para o máximo  
  display.setContrast(255);  
  
  // Limpa o conteúdo do ecrã  
  display.clear();  
}
```

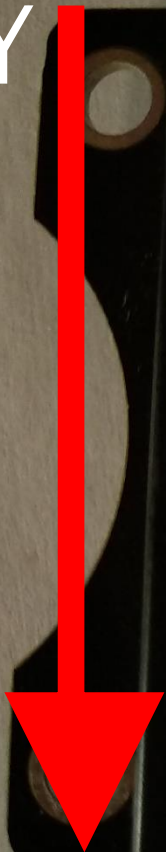
Escrever String

```
void loop() {  
  // Define o alinhamento da escrita  
  display.setTextAlignment(TEXT_ALIGN_LEFT);  
  // TEXT_ALIGN_CENTER  
  // TEXT_ALIGN_RIGHT  
  
  // Escreve um String na posição pretendida  
  display.drawString(x, y, "string");  
  
  // Atualiza o ecrã  
  display.display();  
}
```


X



Y



32504

Left aligned (0,10)

Center aligned (64,22)

Right aligned (128,33)

. . 0 . .

Desenhar


```
void loop() {  
  // Desenha um retângulo  
  display.drawRect(x, y, largura, altura);  
  
  // Desenha um círculo  
  display.drawCircle(x, y, raio);  
  
  // Desenha uma linha horizontal  
  display.drawLine(x, y, comprimento);  
  
  // Desenha uma linha vertical  
  display.drawLine(x, y, comprimento);  
}
```



HackerSchool



OBJETIVO



HackerSchool
Workshop IoT

CONTROLAR O LCD – IMAGENS

No Arduino IDE, abrir

“7º Exercício – controlar LCD - imagens”

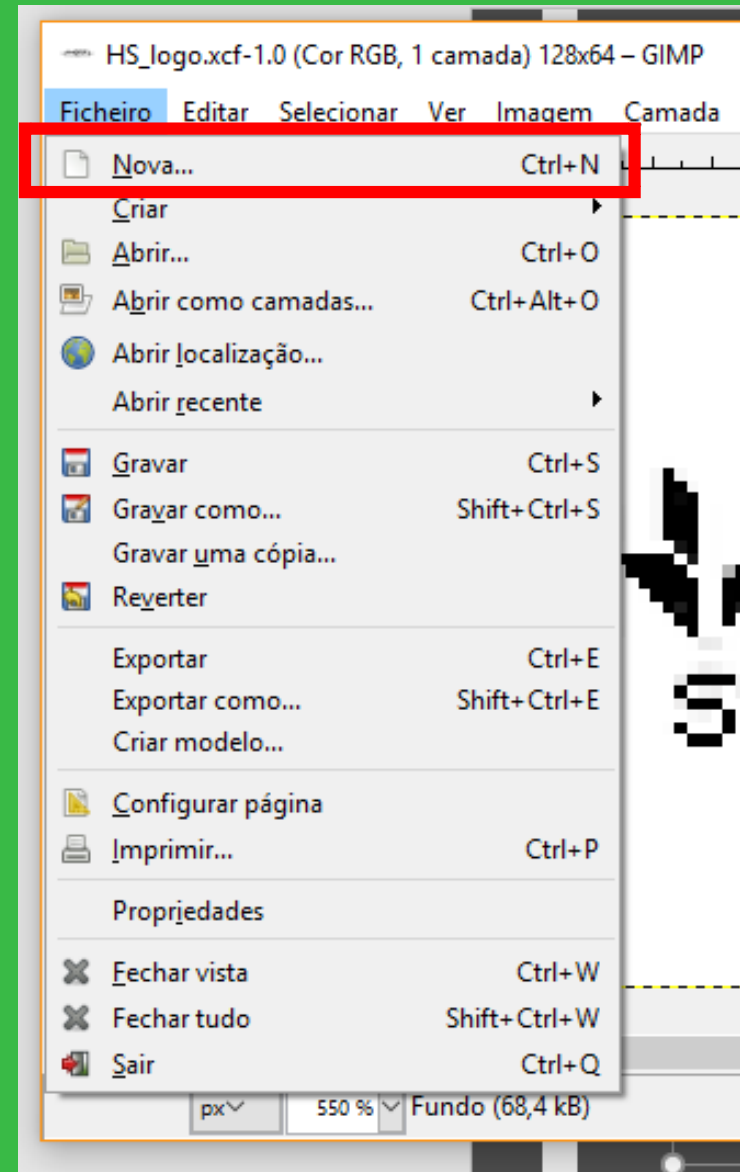
IMAGENS – formato .xbm

- Formato X bit map:
 - ‘1’ representa preto
 - ‘0’ representa branco
- Resolução até 128x64



Como criar .xbm

1. Abrir o GIMP
2. Clicar em Novo

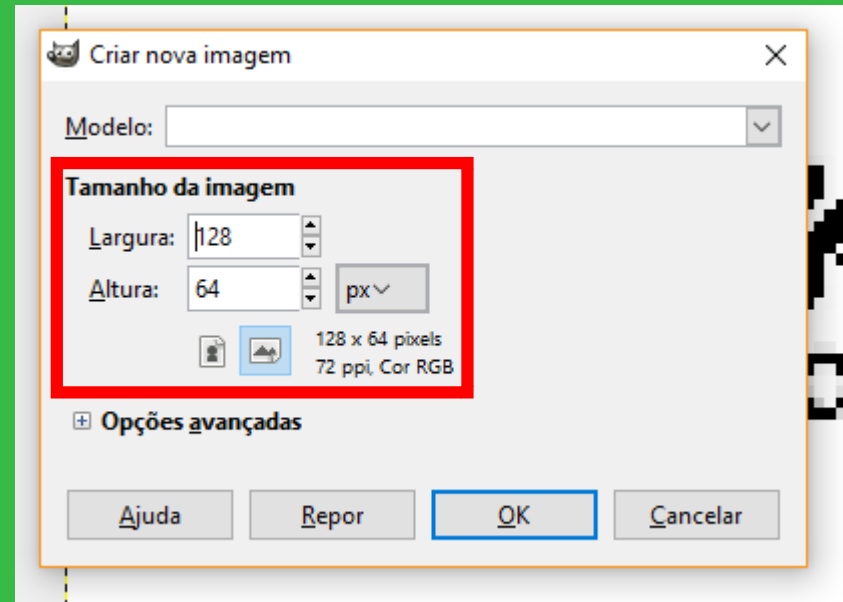


Como criar .xbm

3. Selecionar:

Largura: 128 px

Altura: 64 px



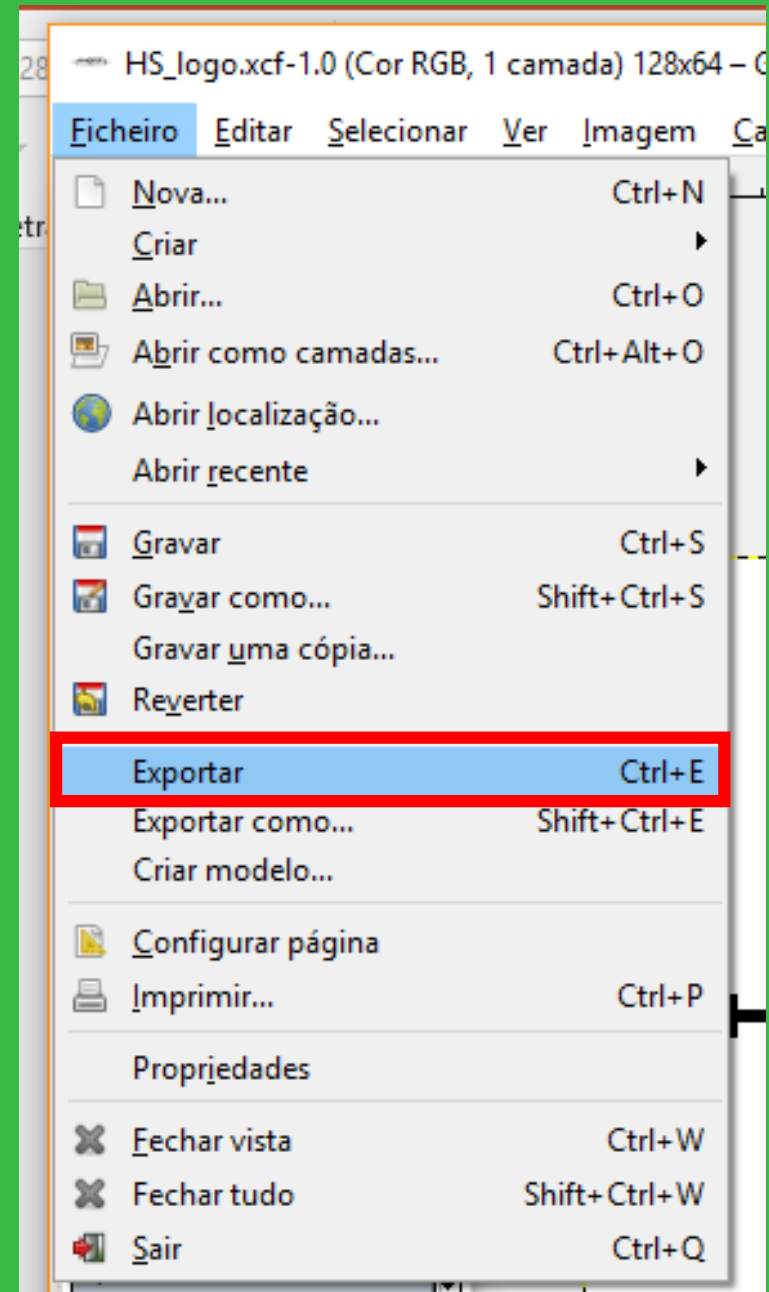
Como criar .xbm

4. Desenhar com a
ferramenta **LÁPIS** e tamanho
'1'



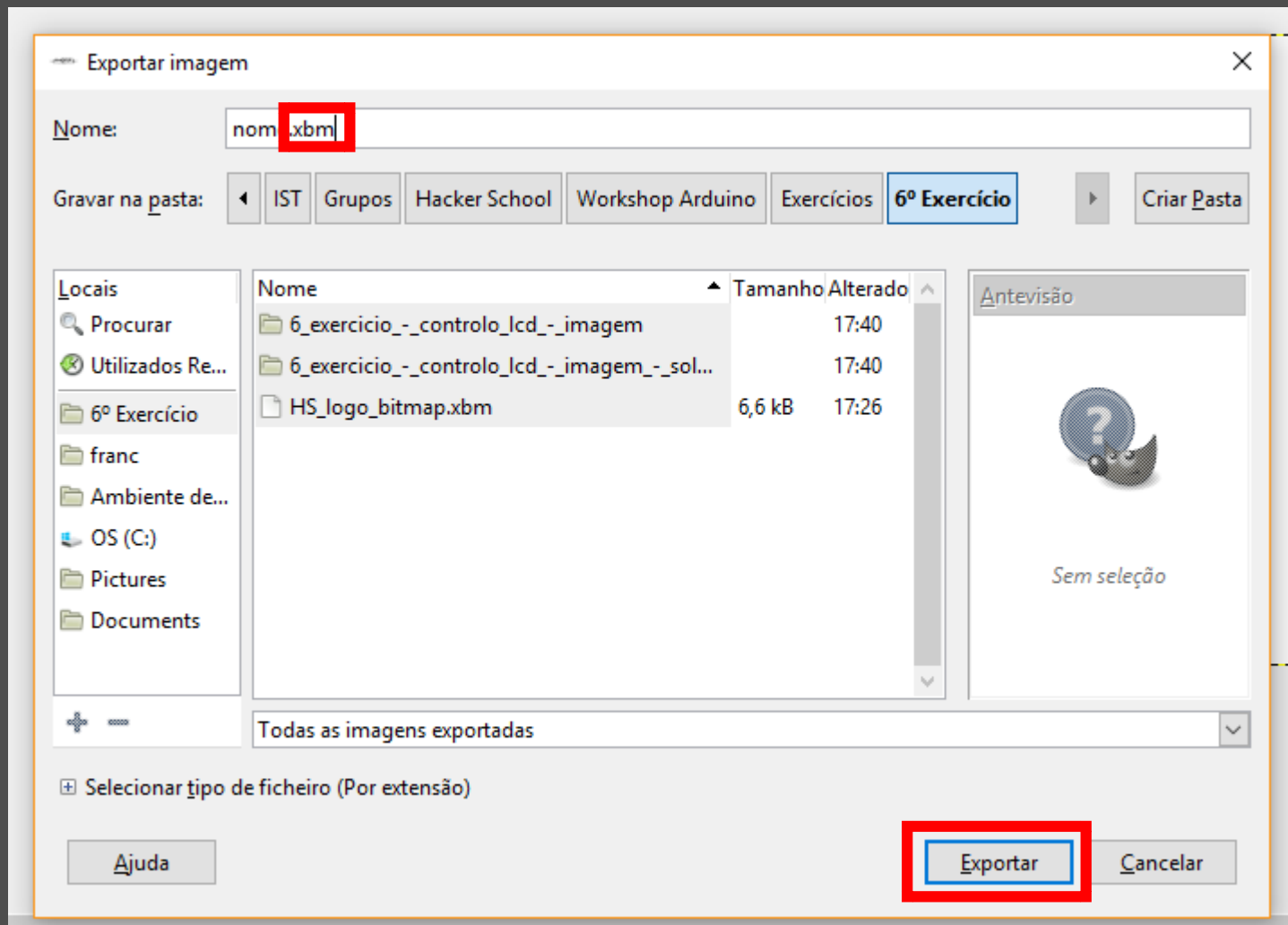
Como criar .xbm

5. Exportar



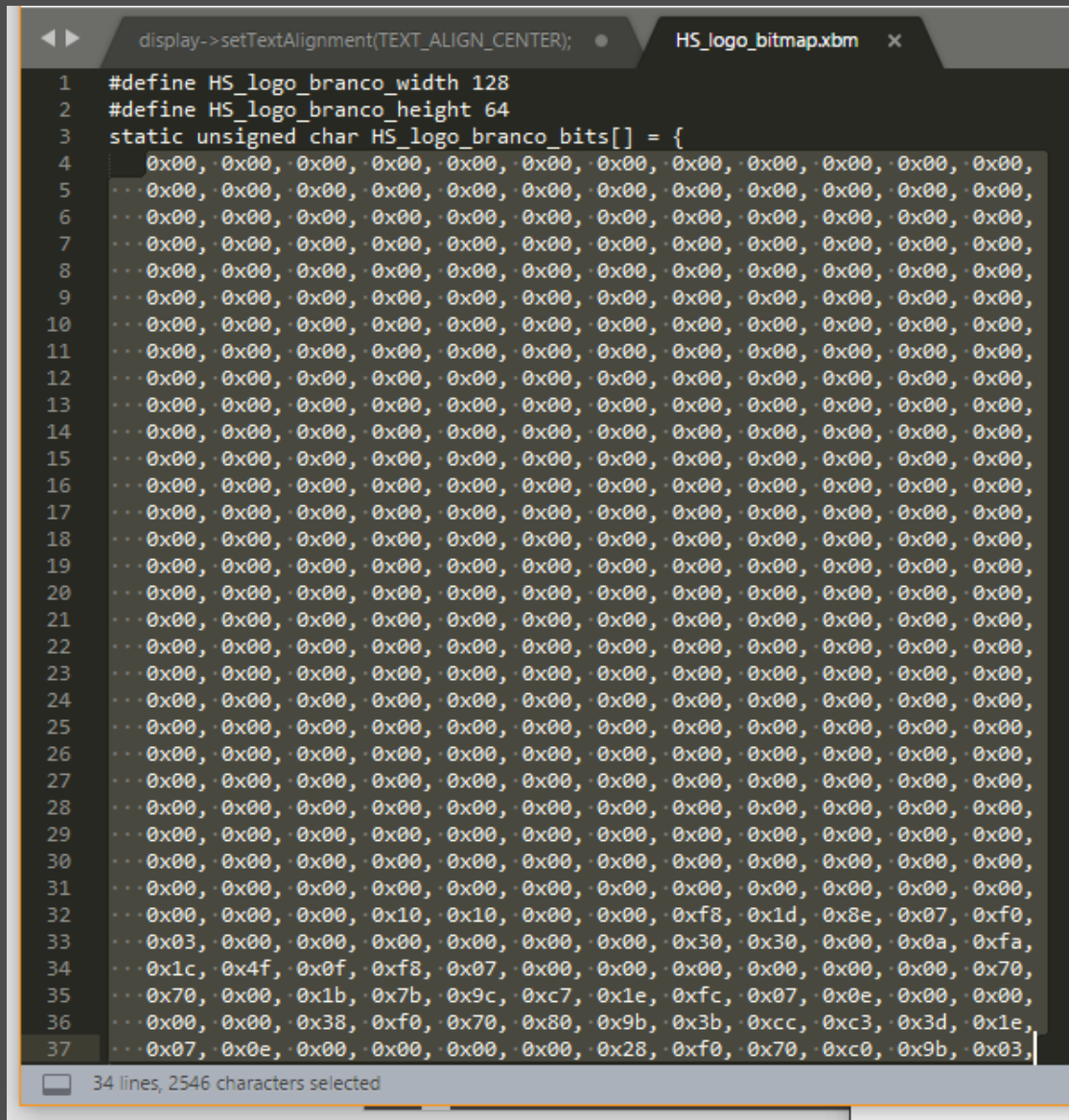
Como criar .xbm

6. Exportar com a extensão .xbm



Como criar .xbm

7. Abrir
documento
criado e
copiar vetor
para o
Arduino IDE



```
1 #define HS_logo_branco_width 128
2 #define HS_logo_branco_height 64
3 static unsigned char HS_logo_branco_bits[] = {
4 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
5 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
6 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
7 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
8 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
9 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
10 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
11 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
12 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
13 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
14 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
15 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
16 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
17 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
18 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
19 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
20 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
21 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
22 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
23 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
24 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
25 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
26 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
27 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
28 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
29 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
30 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
31 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
32 0x00, 0x00, 0x00, 0x10, 0x10, 0x00, 0x00, 0xf8, 0x1d, 0x8e, 0x07, 0xf0,
33 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x0a, 0xfa,
34 0x1c, 0x4f, 0x0f, 0xf8, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70,
35 0x70, 0x00, 0x1b, 0x7b, 0x9c, 0xc7, 0x1e, 0xfc, 0x07, 0x0e, 0x00, 0x00,
36 0x00, 0x00, 0x38, 0xf0, 0x70, 0x80, 0x9b, 0x3b, 0xcc, 0xc3, 0x3d, 0x1e,
37 0x07, 0x0e, 0x00, 0x00, 0x00, 0x00, 0x28, 0xf0, 0x70, 0xc0, 0x9b, 0x03, }
```

34 lines, 2546 characters selected

Imagem

```
void loop() {  
  
    // Desenha um bitmap no ecrã  
    display.drawXbm(int16_t x, int16_t y, int16_t width, int16_t  
height, const char* xbm);  
  
}
```

7º Exercício

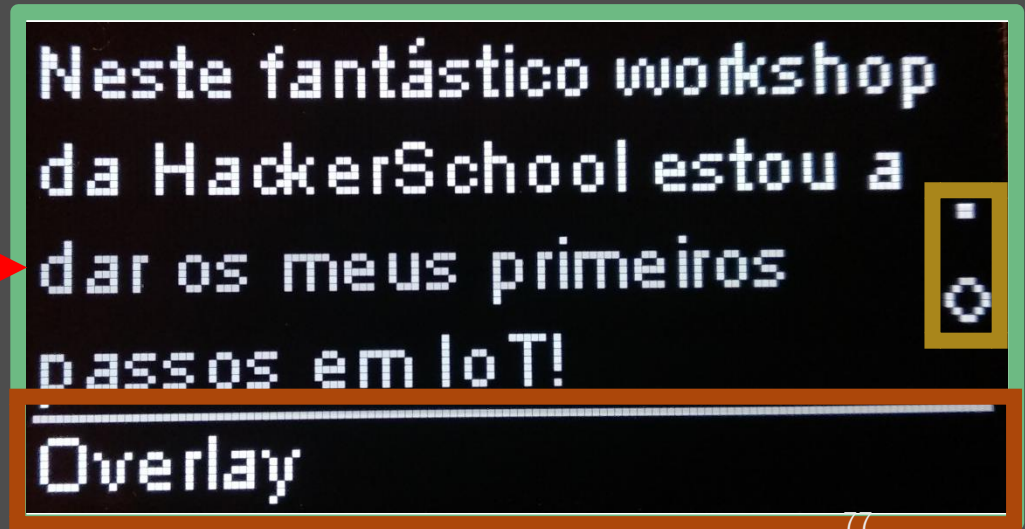
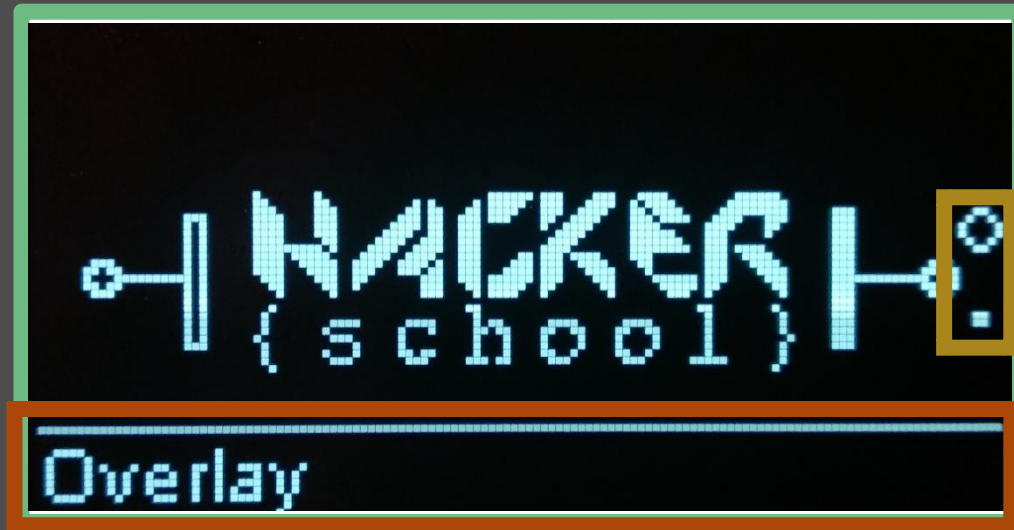
1. Criar um imagem com o GIMP.
2. Definir imagem no ficheiro *Imagem.h*.
3. Escrever a função que desenha a imagem no *setup()*.

CONTROLAR O LCD – FRAMES E OVERLAYS

No Arduino IDE, abrir

“8º Exercício – controlar LCD – frames e overlays”

FRAMES e OVERLAY



8º Exercício

1. Implementar a função *desenhaFrame1* e nela escrever o Código que imprime no ecrã o logo da HackerSchool;
2. Adicionar funções para definir *frames* e *overlays*;
3. Iniciar o *ui*.

API - Interface de programação de aplicações



Conjunto de funções que facilitam a criação de um programa

ESTAÇÃO METEOROLÓGICA – HORA E DATA

Funções iniciais

```
/* Time Client */  
#include "TimeClient.h"  
TimeClient timeClient(UTC_OFFSET);  
  
void loop() {  
    // Atualiza a hora  
    timeClient.updateTime();  
    // Devolve cada parcela de tempo como uma string  
    timeClient.getHours();  
    timeClient.getMinutes();  
    timeClient.getSeconds();  
    timeClient.getFormattedTime();  
}
```

ESTAÇÃO METEOROLÓGICA – INFORMAÇÃO METEOROLÓGICA

Funções de atualização

```
#include "WundergroundClient.h"
#include "WeatherStationFonts.h"
#include "WeatherStationImages.h"
WundergroundClient wunderground(SISTEMA_METRICO);

void loop() {
    // Atualiza o estado da meteorologia atual
    wunderground.updateConditions(WUNDERGRROUND_API_K
    EY, WUNDERGRROUND_LINGUA, WUNDERGROUND_PAIS,
    WUNDERGROUND_CIDADE);
}
```

Funções de atualização

```
void loop() {  
  // Atualiza a previsão meterológica  
  wunderground.updateForecast(WUNDERGRROUND_API_KEY  
    , WUNDERGRROUND_LINGUA, WUNDERGROUND_PAIS,  
    WUNDERGROUND_CIDADE);  
  
  // Devolve a data para uma string  
  wunderground.getDate()  
}
```


Funções de informação

```
void loop() {  
  // Devolve a data para uma string  
  wunderground.getDate()  
  
  // Devolve a previsão por escrito para uma string  
  wunderground.getWeatherText()  
  
  // Devolve a temperatura para uma string  
  wunderground.getCurrentTemp()  
  
  // Devolve o icon correspondente ao estado do tempo para uma string  
  wunderground.getTodayIcon()  
}
```

Funções de informação

```
void loop() {  
  // Resumo da previsão  
  wunderground.getForecastTitle(indiceDia)  
  
  // Icon da previsão  
  wunderground.getForecastIcon(indiceDia)  
  
  // Devolve a temp mínima para uma string  
  wunderground.getForecastLowTemp(indiceDia)  
  
  // Devolve a temp máxima para uma string  
  wunderground.getForecastHighTemp(indiceDia)  
}
```

Site com muitas API

<https://any-api.com/>

<http://www.instructables.com/id/loT-Wallet-smart-Wallet-With-Firebeetle-ESP32-Ardu/>

<http://www.instructables.com/id/loT-Air-Freshner-with-NodeMCU-Arduino-IFTTT-and-Ad/>

OBRIGADO!

WORKSHOP ARDUINO

Francisco Mendes

Francisco.mendes@técnico.ulisboa.pt