

# Programando a Internet das Coisas com o Esp8266

# Um minicurso baseado no meu TCC

Título: Desenvolvimento Open-Source para a Internet das Coisas

Orientador: Prof. Luciano Bertini



# Marcelo Rocha

marcelo\_rocha@id.uff.br

- Graduando em Tecnologia em Sis. de Computação ( UFF/Cederj )
- Técnico em programação de computadores
- Programação C para sistemas embarcados
- Interfaceamento USB usando C#
- Curso de Shell Script pelo LNCC
- Programação orientada a objetos com Java pelo ITA

# Um pequeno roteiro...

- IoT (História e sua importância hoje)
- O Esp8266
- Baixando e configurando a IDE do Arduino para programar o Esp8266 - Programando o Esp8266 em C++
- A linguagem LUA
- NodeMcu = Esp8266 + firmware LUA
- Gravando o firmware LUA usando o **esptool.py** (**Pyflasher.py**)
- Customizando o firmware (Economia de memória)
- Interagindo com o NodeMcu usando a IDE **ESplorer**
- Conhecendo o ThingSpeak
- Montando o projeto do sensor DHT11 com ThingSpeak

# IoT (Internet of Things)

O termo “Internet das Coisas” foi criado em 1999 por **Kevin Ashton** e desde então tem sido atribuído a vários jargões tecnológicos. Ao longo dos anos, tem-se ouvido os termos:

- “Internet das Coisas”
- “Internet de Tudo”
- “M2M” (Maquina-para-Máquina)
- “Computação Física”
- “Computação Ubíqua”, e assim por diante.

# IoT (Internet of Things)

## O que é “Internet das Coisas”?

Todos nós sabemos o que é “Internet” e nós todos provavelmente temos uma boa idéia do que são “coisas”. Então, se colocarmos as duas juntas nós podemos pensar nelas como “coisas na Internet”

# IoT (Internet of Things)

Para tornar as coisas mais confusas, vamos ver o que diz a Wikipedia:

- “A Internet das coisas (IoT) é uma rede de objetos físicos ou “coisas” embarcadas com eletrônica, software, sensores, e conectividade para que essa seja capaz de alcançar um grande valor e serviço, trocando dados com o fabricante (inventor, produtor), operador, e/ou outros dispositivos conectados.”
- Em resumo, a Internet das coisas diz respeito a valor, conhecimento, e oportunidades ganhas através da interconectividade do mundo físico e digital.

# IoT (Internet of Things)

Dependendo das suas necessidades específicas, “valor” pode vir de várias formas. Por exemplo, se sua necessidade inclui rastrear e monitorar frotas de veículos, seu “valor” poderia vir em forma de observação sobre a obediência do motorista, otimização da rota, melhorar a eficiência do combustível, e mais precisão no cumprimento dos horários.

Se suas necessidades incluem monitorar autonomamente a saúde de uma pessoa querida ou de um paciente, seu “valor” poderia vir em forma de manter a saúde dele e até mesmo mantê-lo vivo.

# IoT (Internet of Things)

Independetemente de como você rotule o termo, a Internet das Coisas é basicamente:

- A prática de **conectar**, **monitorar** suas “coisas” (ou dispositivos)
- É a habilidade de **integrar** seus dispositivos físicos com seus sistemas digitais
- **Analizar** esses dispositivos e os dados entre eles
- **processar** esses eventos e **executar** ações conforme o necessário

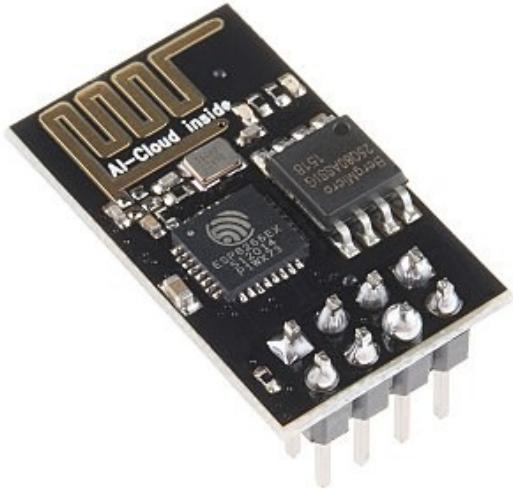
# IoT (Internet of Things)

A IoT é feita de três pilares principais:

- **Integração**
- **Análise**
- **Processamento de eventos.**

# O Esp8266

Esp01



- Wifi de baixo custo
- Pilha TCP/IP completa
- Possui um microcontrolador
- Fabricado pela Espressif Systems
- Possui de 512kb a 4mb de memória flash
- Freq. 80Mhz (default)
- 16 GPIOs
- SPI
- I<sup>2</sup>C
- UART
- ADC

\*\*\* Construído inicialmente para servir como “shield” wifi para arduino

# A família Esp8266



ESP-01



ESP-02



ESP-03



ESP-04



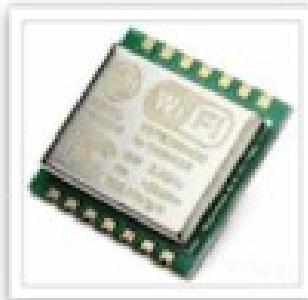
ESP-05



ESP-06



ESP-07



ESP-08



ESP-09



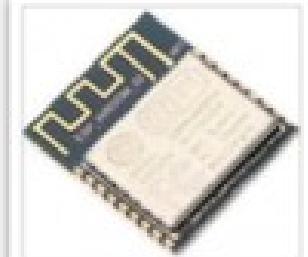
ESP-10



ESP-11.jpg



ESP-12



ESP-13



ESP-14

[www.iotbytes.com](http://www.iotbytes.com)

# A família Esp8266 (características)

**Summary Table**

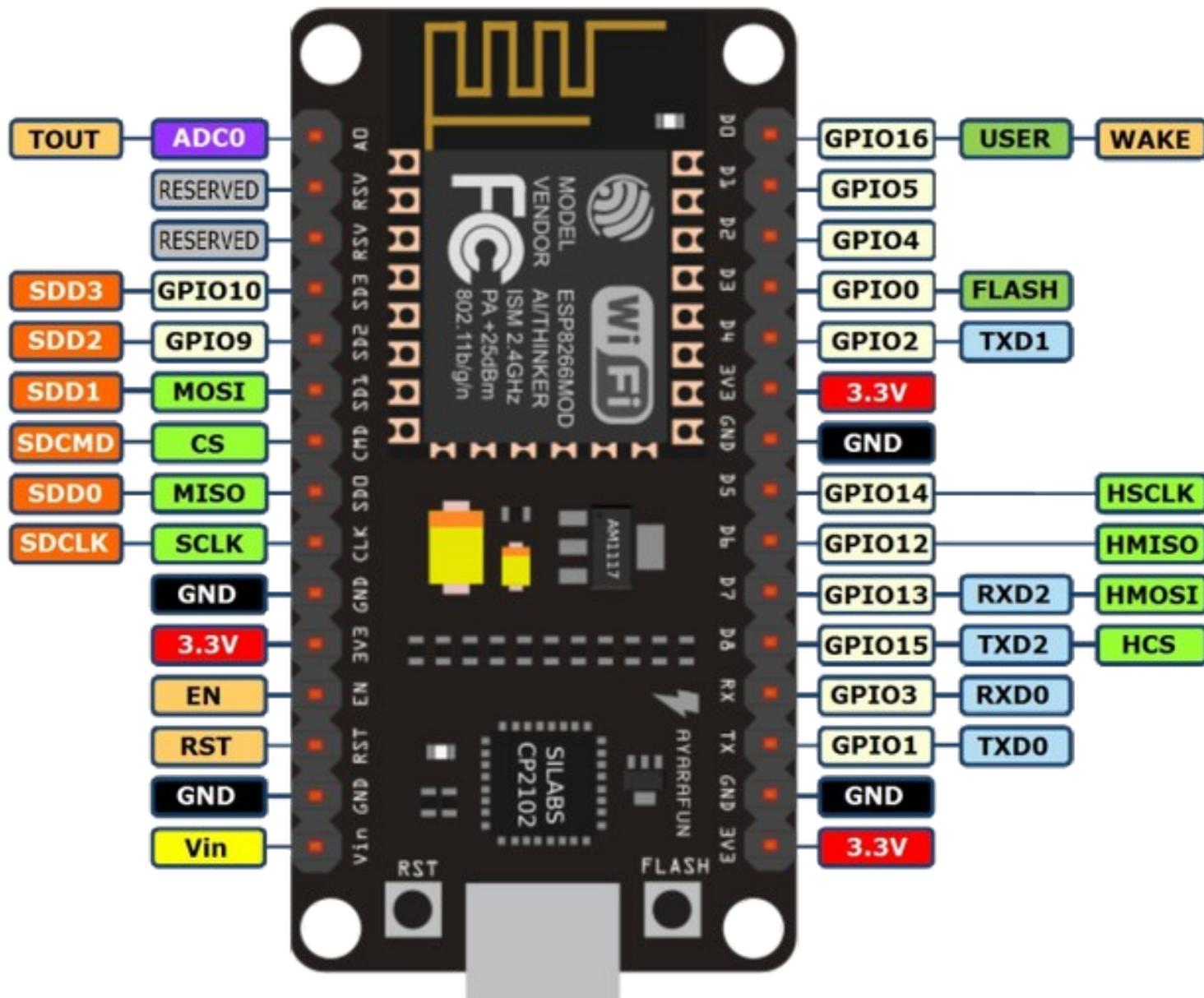
Board ID	#Pins	Pitch	Form factor	LEDs	Antenna	Ant.Socket	Shielded	Dimensions mm	Flash Size in Bytes and (bits)
ESP-01	8	0.1"	2x4 DIL	Yes	Etched-on PCB	No	No	14.3 x 24.8	512KB (4Mb) xx
ESP-02	8	0.1"	2x4 notch	No?	None	Yes	No	14.2 x 14.2	512KB (4Mb) x
ESP-03	14	2mm	2x7 notch	No	Ceramic	No	No	17.3 x 12.1	512KB (4Mb) x
ESP-04	14	2mm	2x4 notch	No?	None	No	No	14.7 x 12.1	512KB (4Mb) x
ESP-05	5	0.1"	1x5 SIL	No	None	Yes	No	14.2 x 14.2	512KB (4Mb) x
ESP-06	12+GND	misc	4x3 dice	No	None	No	Yes	16.3 x 13.1	512KB (4Mb) x
ESP-07	16	2mm	2x8 pinhole	Yes	Ceramic	Yes	Yes	21.2 x 16.0	1MB (8Mb) xx
ESP-07S	16	2mm	2x8 pinhole	No	None	Yes	Yes	17.0 x 16.0	4MB (32Mb)
ESP-08	14	2mm	2x7 notch	No	None	No	Yes	17.0 x 16.0	?? (please fill if you know)
ESP-08 New	16	2mm	2x8 notch	No	None	No	Yes	18.0 x 16.0	?? (please fill if you know)
ESP-09	12+GND	misc	4x3 dice	No	None	No	No	10.0 x 10.0	1MB (8Mb)
ESP-10	5	2mm ??	1x5 notch	No	None	No	No	14.2 x 10.0	512KB (4Mb) *
ESP-11	8	1.27mm	1x8 pinhole	No?	Ceramic	No	No	17.3 x 12.1	512KB (4Mb) *
ESP-12	16	2mm	2x8 notch	Yes	Etched-on PCB	No	Yes	24.0 x 16.0	4MB (32Mb) ??
ESP-12F	22	2mm	2x8 notch	Yes	Etched-on PCB	No	Yes	24.0 x 16.0	4MB (32Mb)
ESP-12E	22	2mm	2x8 notch	Yes	Etched-on PCB	No	Yes	24.0 x 16.0	4MB (32Mb)
ESP-12S	16	2mm	2x8 notch	Yes	Etched-on PCB	No	Yes	24.0 x 16.0	4MB (32Mb)
ESP-13	18	1.5mm	2x9	?	Etched-on PCB	No	Yes	20.0 x 19.9	4MB (32Mb)
ESP-14	22	2mm	2x8 + 6	1	Etched-on PCB	No	Yes	24.3 x 16.2	?? (please fill if you know)
ESP-201	22+4	0.1"	2x11 + 4	2	Etched-on PCB ***	Yes	No	33.5 x 25.5	512KB (4Mb)
WROOM-02	18	1.5mm	2x9	No	Etched on PCB	No	Yes	20.0 x 18.0	?? (please fill if you know)
WT8266-S1	18	1.5mm	3x6	1	Etched on PCB	No	Yes	15.0 x 18.6	4MB (32Mb)

\* New firmwares can only be flashed on boards with at least 1MB (8Mb) flash. \*\* May be different on different editions of the board. \*\*\* Antenna connector is connected by default, to use PCB antenna switch (solder) the 0Ω resistor to the corresponding position.

# Alguns SDKs para Esp8266

- NodeMCU - Firmware baseado em LUA
- Arduino - Like Arduino programming
- MicroPython
- ESP8266 BASIC Interpretador BASIC open source para IOT
- Zbasic for ESP8266 - Um subconjunto do MS visual basic
- Espruino - JavaScript SDK - Emula Node.js
- Mongoose Firmware - Cloud service
- ESP-Open-SDK - Implementação open source do SDK
- ESP-Open-RTOS - Baseado no FreeRTOS
- Zerynth - Programação Python

# O NodeMcu e suas portas



# Baixando a IDE do Arduino

## Download the Arduino IDE



**ARDUINO 1.8.5**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions.

**Windows** Installer  
**Windows** ZIP file for non admin install

**Windows app** 

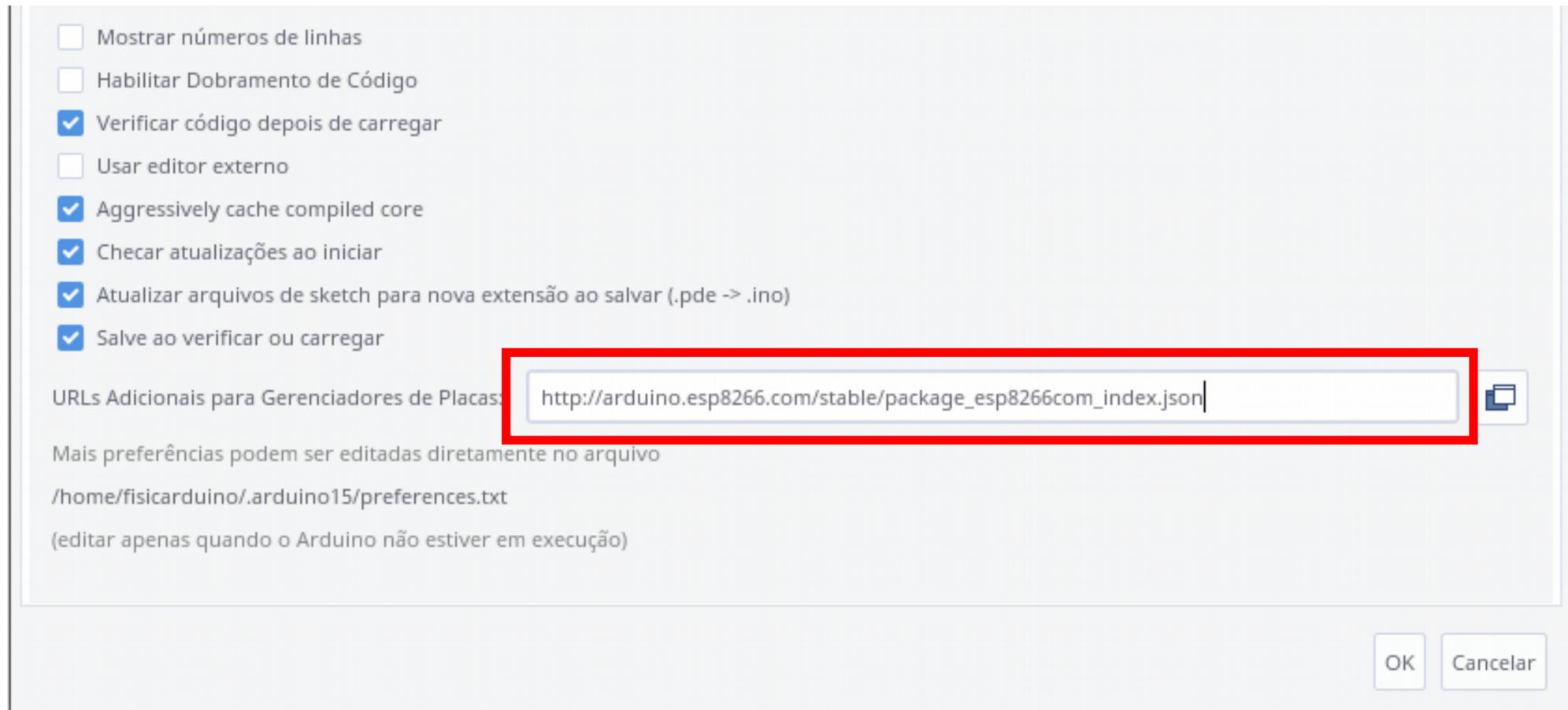
**Mac OS X** 10.7 Lion or newer

**Linux** 32 bits  
**Linux** 64 bits  
**Linux** ARM

[Release Notes](#)  
[Source Code](#)  
[Checksums \(sha512\)](#)

- <https://www.arduino.cc/en/Main/Software>
- No Debian sudo apt install arduino

- Vamos em Arquivo-->Preferências



# Configurando a IDE do Arduino (instalando a library esp8266)

- Vamos em Ferramentas-->Placa-->Gerenciador de placas
- Instalamos a library Esp8266



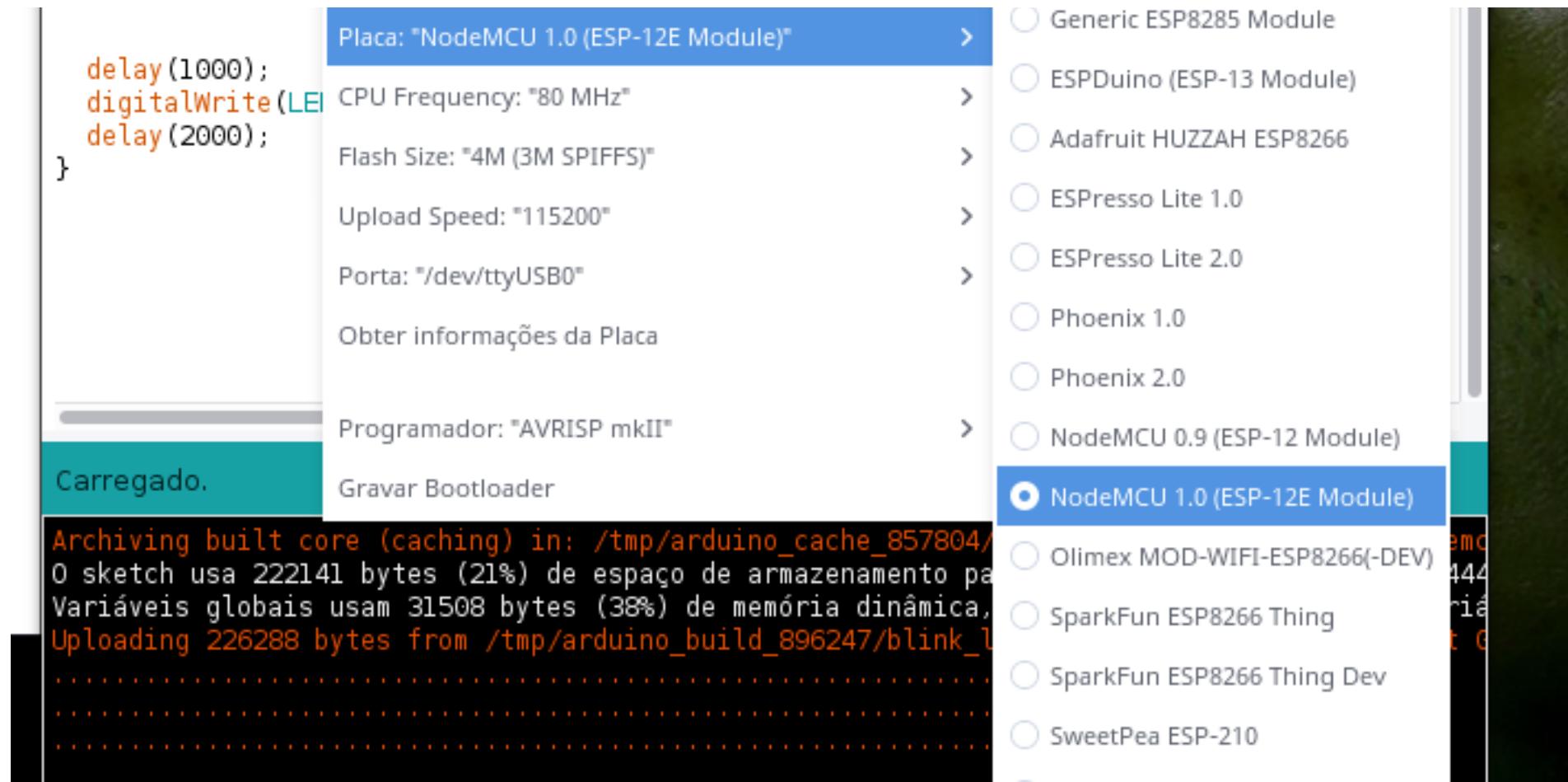
# Configurando a IDE do Arduino (selecionando a porta)

- Vamos em Ferramentas-->Porta-->/dev/tty/USB0

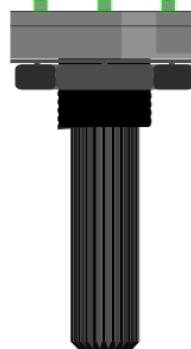
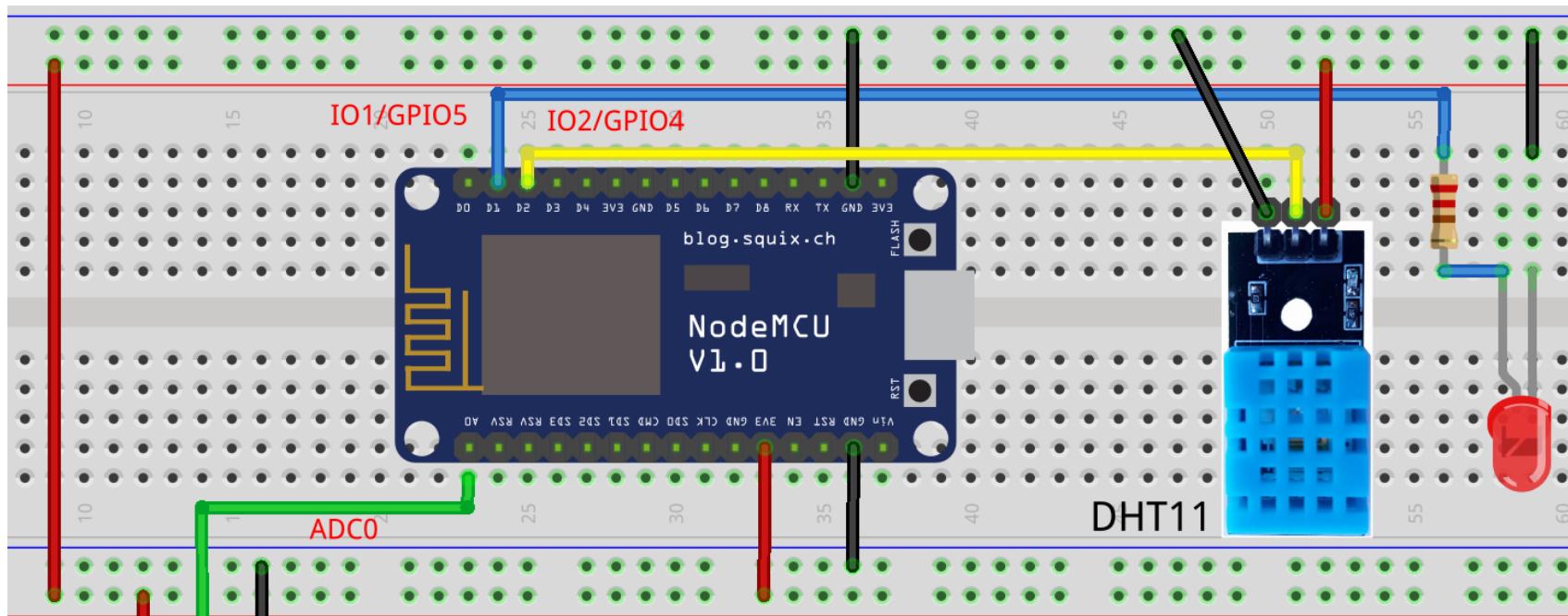


# Configurando a IDE do Arduino (selecionando o modelo)

- Vamos em Ferramentas-->Placa-->...modelo..



# O esquema das conexões na protoboard



fritzing

IO index	ESP8266 pin	IO index	ESP8266 pin
0[*]	GPIO16	7	GPIO13
1	GPIO5	8	GPIO15
2	GPIO4	9	GPIO3
3	GPIO0	10	GPIO1
4	GPIO2	11	GPIO9
5	GPIO14	12	GPIO10
6	GPIO12		

# **Gravando e executando códigos com a IDE do Arduino (Hands ON)**

# A linguagem LUA



- Projetada, implementada e desenvolvida no Brasil
- Criada na PUC-RJ
- Criadores:
  - Roberto Ierusalimschy
  - Waldemar Celes
  - Luiz Henrique de Figueiredo

# A linguagem LUA



- Poderosa, eficiente e leve, projetada para estender aplicações
- Ela permite: programação procedural; programação orientada a objetos; programação funcional; programação orientada a dados e descrição de dados
- Tipagem dinâmica
- Execução de bytecodes/Máquina virtual
- Gerenciamento automático de memória

# A linguagem LUA - Por que usar LUA?



## Por que usar LUA?

- Usada em várias aplicações industriais
  - **Adobe Lightroom**
  - Sistemas embarcados middleware **Ginga** (programas para Tv digital)
  - Jogos (**World of Warcraft** e **Angry Birds**)

# A linguagem LUA - Por que usar LUA?



## Por que usar LUA?

- LUA é rápida
  - Vários benchmarks mostram Lua como a linguagem mais rápida dentre as linguagens de script interpretadas

# A linguagem LUA - Por que usar LUA?



## Por que usar LUA?

- LUA é portátil
  - Lua é distribuída via um pequeno pacote
  - Escrita em ANSI C, compila em qualquer compilador C padrão
  - Roda em todos os tipos de UNIX e WINDOWS
  - Sistemas móveis: Android; IOS e outros
  - Em microprocessadores embarcados como: Arm e Rabbit

# A linguagem LUA - Por que usar LUA?



## Por que usar LUA?

- LUA é embutível
  - Lua é uma engine rápida e pequena
  - Tem uma API simples e bem documentada
  - Permite integração forte com código escrito em outras linguagens
  - É uma linguagem **estensível** e de **estensão**

# A linguagem LUA - Por que usar LUA?



- LUA é poderosa e simples
  - Lua fornece meta-mecanismos para a implementação de construções, em vez de fornecer uma multidão de construções diretamente na linguagem
  - Embora Lua não seja uma linguagem puramente orientada a objetos, ela fornece meta-mecanismos para a implementação de classes e herança. Os meta-mecanismos de Lua trazem uma economia de conceitos e mantêm a linguagem pequena, ao mesmo tempo que permitem que a semântica seja estendida de maneiras não convencionais.

# A linguagem LUA - Por que usar LUA?



## Por que usar LUA?

- LUA é pequena
  - O pacote de Lua 5.3.4, contendo o código fonte e a documentação, ocupa 297K comprimido e 1.1M descompactado
  - No Linux de 64 bits, o interpretador Lua contendo todas as bibliotecas padrão de Lua ocupa 246K e a biblioteca Lua ocupa 421K

# A linguagem LUA - Por que usar LUA?



## Por que usar LUA?

- LUA é livre
  - Lua é software livre de código aberto, distribuída sob uma licença muito liberal (a conhecida licença MIT)
  - Lua pode ser usada para quaisquer propósitos, incluindo propósitos comerciais, sem qualquer custo ou burocracia
  - Basta fazer um download e usá-la.

# A linguagem LUA - Por que usar LUA?



## Por que usar LUA?

- LUA tem importância global
  - Lua é a única linguagem de programação de impacto desenvolvida fora do primeiro mundo
  - Lua ganhou o *Front Line Award* 2011 da Game Developers Magazine.

# A linguagem LUA - Tipos em LUA



## Tipagem dinâmica

- Cada valor carrega o seu próprio tipo

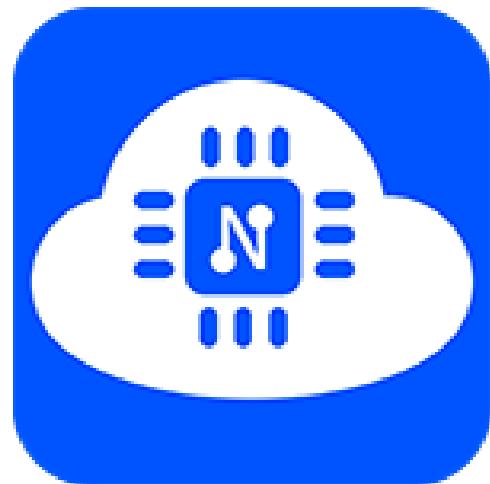
## Há 8 tipos básicos em LUA

- Nil
- Boolean
- Number
- String
- Table
- Function
- Userdata
- Thread

# A linguagem LUA - Tipos básicos



- **Nil** - Ausência de valor útil (diferente de qualquer outro)
- **Boolean** - false e true
- **Number** - Números reais (Lua não tem tipo inteiro)
- **String** - Sequência de caracteres
- **Table** - Arrays associativos - Unica estrutura de dados da linguagem
- **Function** - São valores de primeira classe em Lua
- **Userdata** - Armazena uma referência de algum ponteiro da linguagem C
- **Thread** - Representa fluxos de execução independentes



# NodeMcu

## Conecťe “Coisas” Facilmente

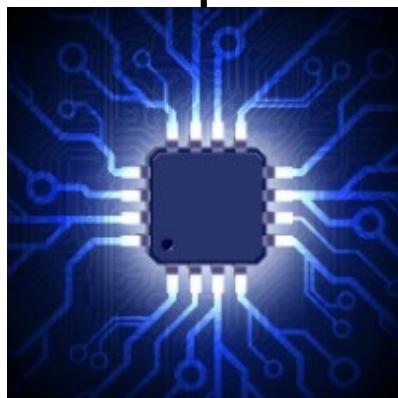
Um firmware e um kit de desenvolvimento de código aberto que ajuda você a prototipar seus produtos IOT com poucas linhas de código em Lua



## Características

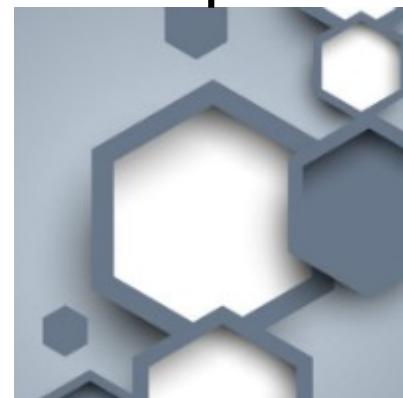
Open-source - Interativo - Programável - Barato - Simples - Inteligente -  
WI-FI

# NodeMCU (características...)



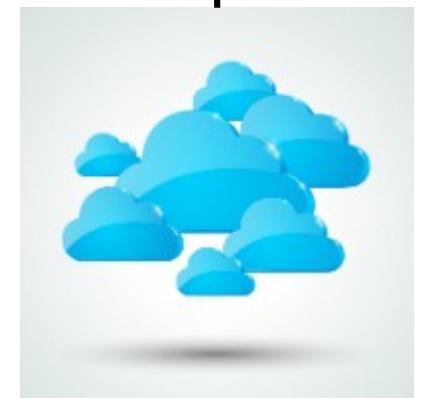
## Arduino Like Programming

Programe como Arduino, mas, interativamente em Lua script.



## Nodejs style network API

Modelo de programação similar ao Node.js, só que em LUA. É asincrono e orientado a eventos.



## Lowest cost WI-FI

Em torno de R\$ 20,00.



# O Kit De Desenvolvimento

- Baseado no ESP8266, integra GPIO, PWM, I<sup>2</sup>C, 1-Wire e ADC tudo em uma placa.
- Tudo isso combinado ao firmware NodeMcu.
- Conversor USB-TTL incluído
- 12 GPIOs, que podem ser configuradas como PWM, I2C e 1-Wire
- Módulo WI-FI, FCC certified, PCB antena

É possível gerar firmwares customizados com somente as bibliotecas que nos interessam, assim podemos economizar memória.

Fazemos isso com a ajuda do site: <https://nodemcu-build.com/>

The screenshot shows the homepage of the NodeMCU custom builds website. At the top, there is a dark navigation bar with a blue cloud icon containing a Wi-Fi symbol, followed by the text "Build", "Statistics", "FAQ", and "About". The main content area has a light gray background. In the center, the text "NodeMCU custom builds" is displayed in a large, bold, black font. Below this, a smaller text reads: "You customize your NodeMCU firmware and we build it. Just for you. On the spot." At the bottom of the main area is a blue button with white text that says "Check the build stats".

# Gravando o firmware (interpretador LUA dentro do Esp8266)

- Para usar o Esp8266 com a linguagem LUA, precisamos fazer o upload do firmware que roda o interpretador LUA.
- Para fazer o upload no linux podemos usar o programa em python chamado **esptool.py**
- Para baixar usamos: **git clone https://github.com/themadinventor/esptool.git**
- Para instalar: **sudo python setup.py install**
- \*\*\* tem como pré-requisito a instalação da library **pyserial** (instalação simples via APT)
- Precisamos do firmware...
- Baixando: **wget -c https://github.com/nodemcu/nodemcu-firmware/releases/download/0.9.6-dev\_20150704/nodemcu\_float\_0.9.6-dev\_20150704.bin**
- Agora conecte o Esp8266 na porta usb e digite este comando para ver em que porta ele está aparecendo no sistema.
- **dmesg | grep tty**

# Gravando o firmware (interpretador LUA dentro do Esp8266)

- `dmesg | grep tty`
- O Esp8266 está na porta `/dev/ttyUSB0`

```
[18064.174130] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[19190.724691] usb 1-1: cp210x converter now attached to ttyUSB0
[20229.095729] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[20234.433601] usb 1-1: cp210x converter now attached to ttyUSB0
[22359.184599] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[22364.125116] usb 1-1: cp210x converter now attached to ttyUSB0
[22793.735755] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[22803.364474] usb 1-1: cp210x converter now attached to ttyUSB0
[22813.002620] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[22821.325216] usb 1-1: cp210x converter now attached to ttyUSB0
larselo@Xubuntu-note: ~/Dropbox/CEDETE/_Mat_TCC/nodemCU/LUA/Esp8266-nodeMCU firmware upload$
```

- Para fazer o upload do firmware copie o arquivo do firmware para a pasta do esptool e execute a linha abaixo:
- `sudo python esptool.py --port /dev/ttyUSB0 --baud 115200 write_flash 0x00000 nodemcu_float_0.9.6-dev_20150704.bin`

# Gravando o firmware (interpretador LUA dentro do Esp8266)

- sudo python esptool.py --port /dev/ttyUSB0 --baud 115200 write\_flash 0x00000 nodemcu\_float\_0.9.6-dev\_20150704.bin

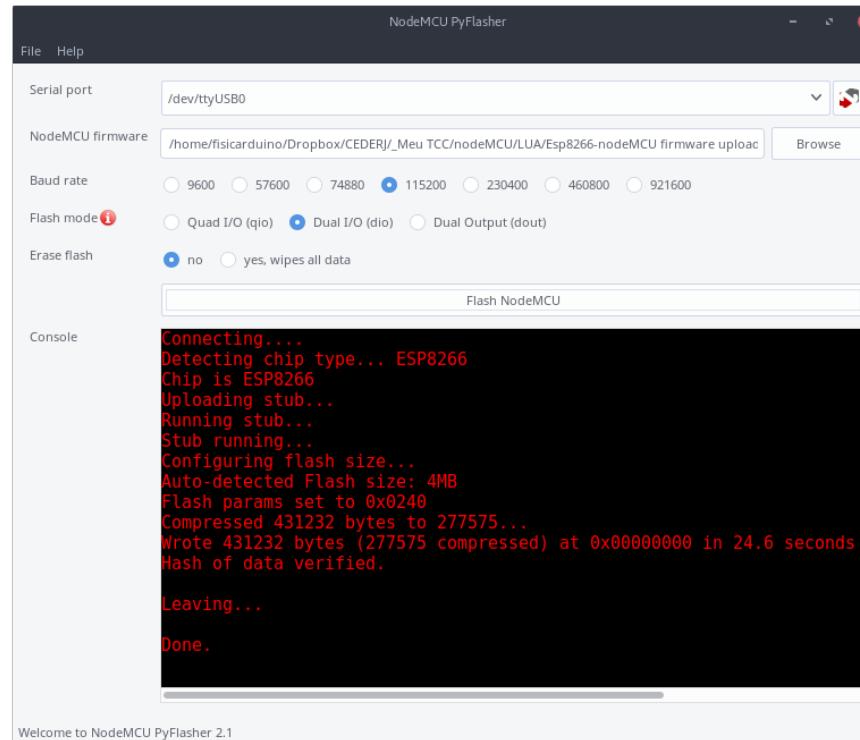
The screenshot shows a terminal window titled "Terminal - marcelo@Xubuntu-note: ~/Dropbox/CEDERJ/\_Meu TCC/nodeMCU/LUA/Esp8266-nodeMCU firm". The window contains the following terminal session:

```
marcelo@Xubuntu-note:~/Dropbox/CEDERJ/_Meu TCC/nodeMCU/LUA/Esp8266-nodeMCU firmware upload/esptool$ sudo python esptool.py --port /dev/ttyUSB0 --baud 115200 write_flash 0x00000 nodemcu_float_0.9.6-dev_20150704.bin
esptool.py v2.0-beta3
Connecting....
Detecting chip type... ESP8266
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Flash params set to 0x0040
Compressed 461984 bytes to 298188...
Writing at 0x00018000... (36 %)
```

# Gravando o firmware (facilitando a vida com o pyflasher)

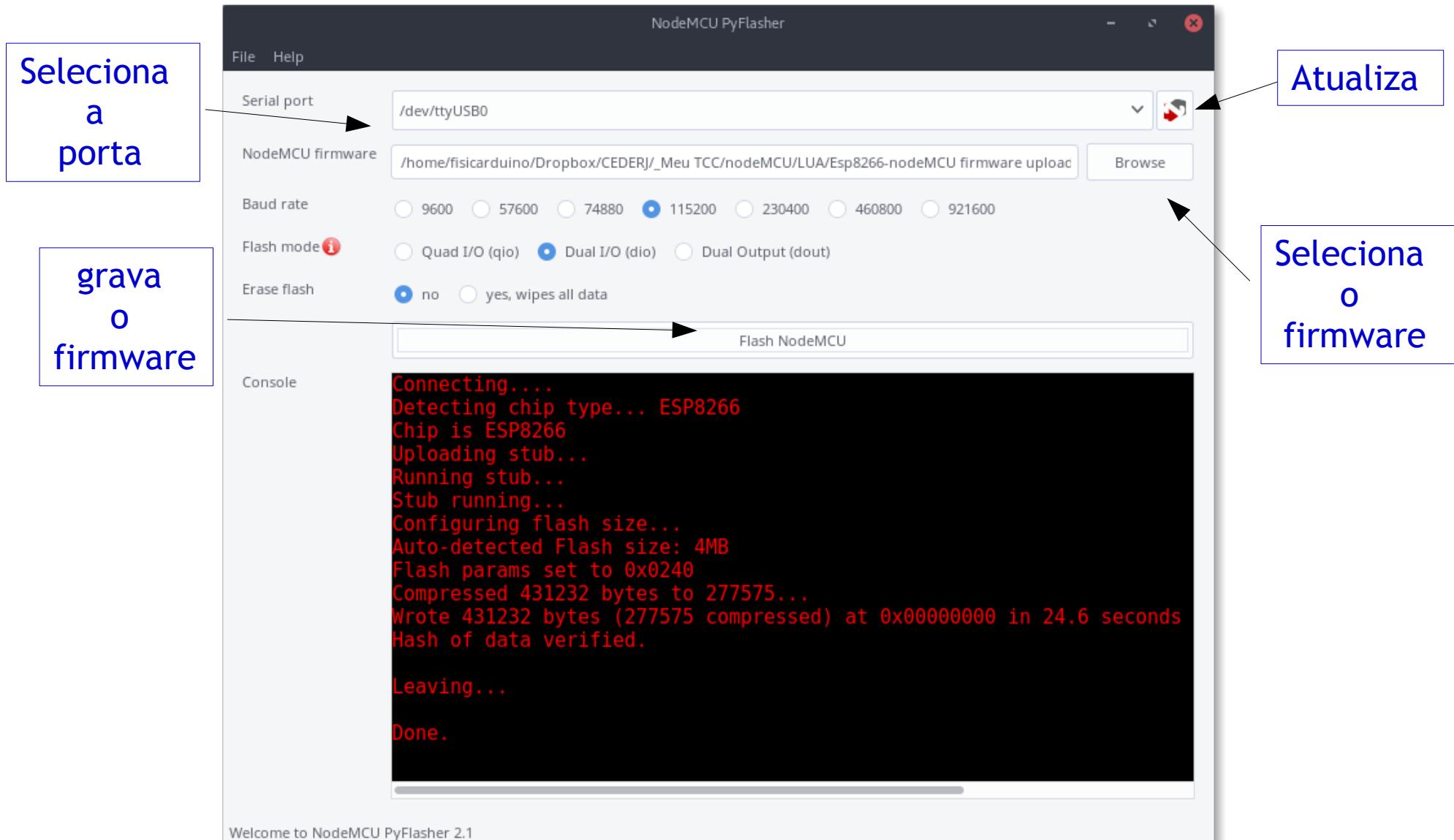
Usando a GUI tool **pyflasher** para a gravação do firmware LUA

- Download: <https://github.com/marcelstoer/nodemcu-pyflasher.git>
- Baseado no **esptool.py** (tem ele como pré-requisito)
- Usa o toolkit gráfico **wxpython** (também é um pré-requisito - fácil de instalar via APT)



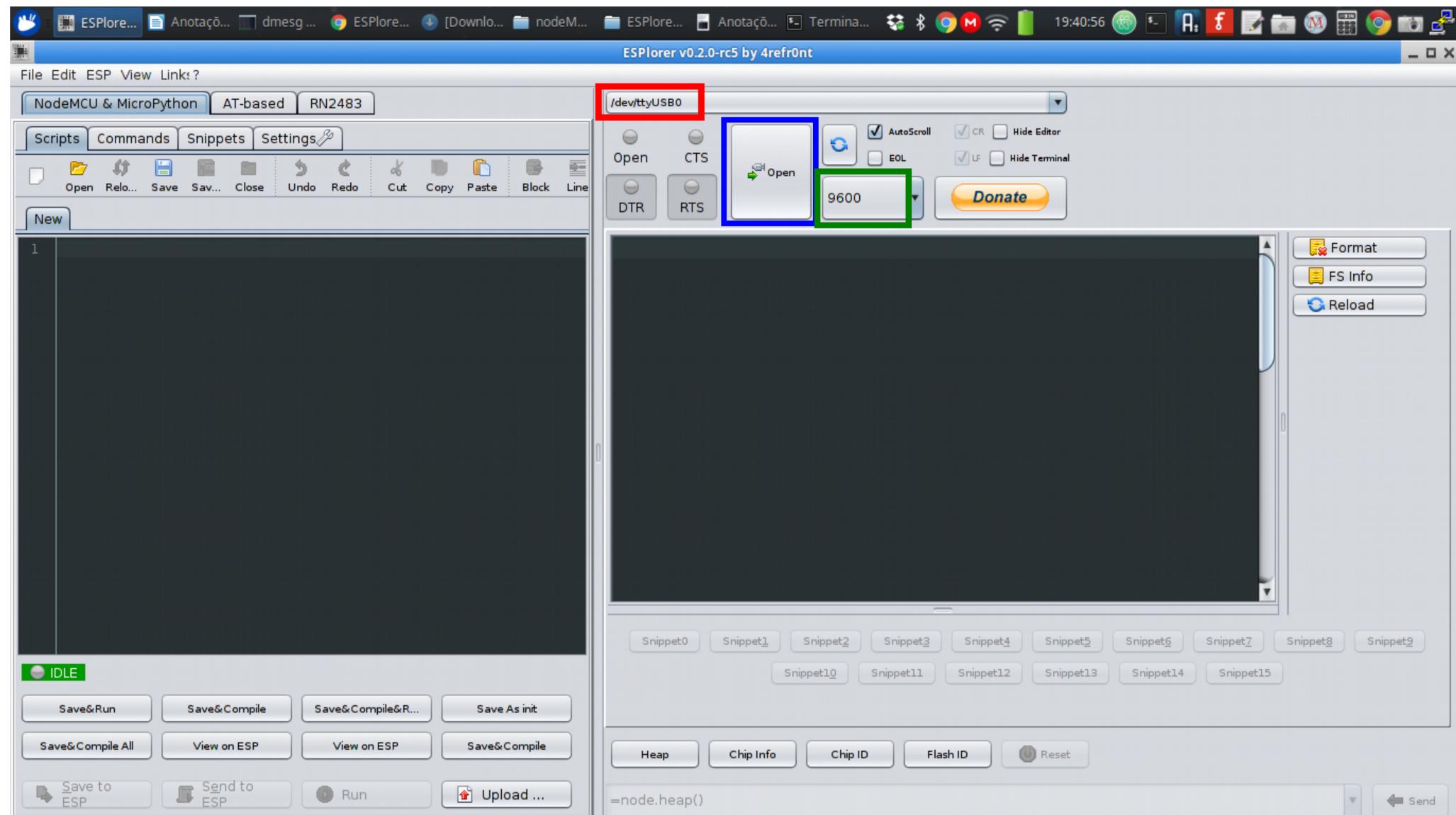
# Gravando o firmware (facilitando a vida com o pyflasher)

Pra rodar usamos: `python nodemcu-pyflasher.py`

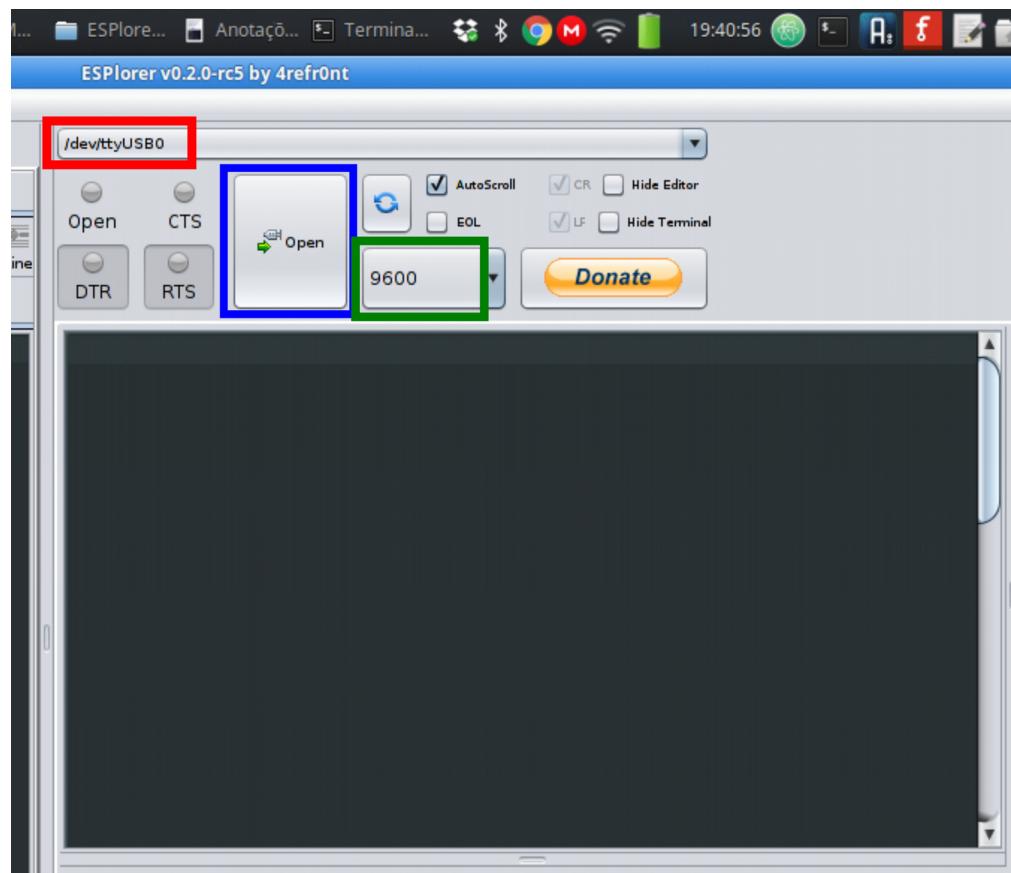


- Após o upload do firmware vamos usar o software **ESplorer** para programarmos Esp8266 com código em LUA.
- Baixe o **ESplorer** aqui:
- `wget -c http://esp8266.ru/esplorer-latest/?f=ESPlorer.zip`
- Descompacte o arquivo e entre na pasta
- Para rodar o **ESplorer** digite:
- `java -jar esplorer.jar`
- Obs.: (você precisa ter o Java instalado)

# Gravando o firmware (interpretador LUA dentro do Esp8266)



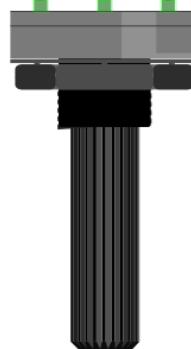
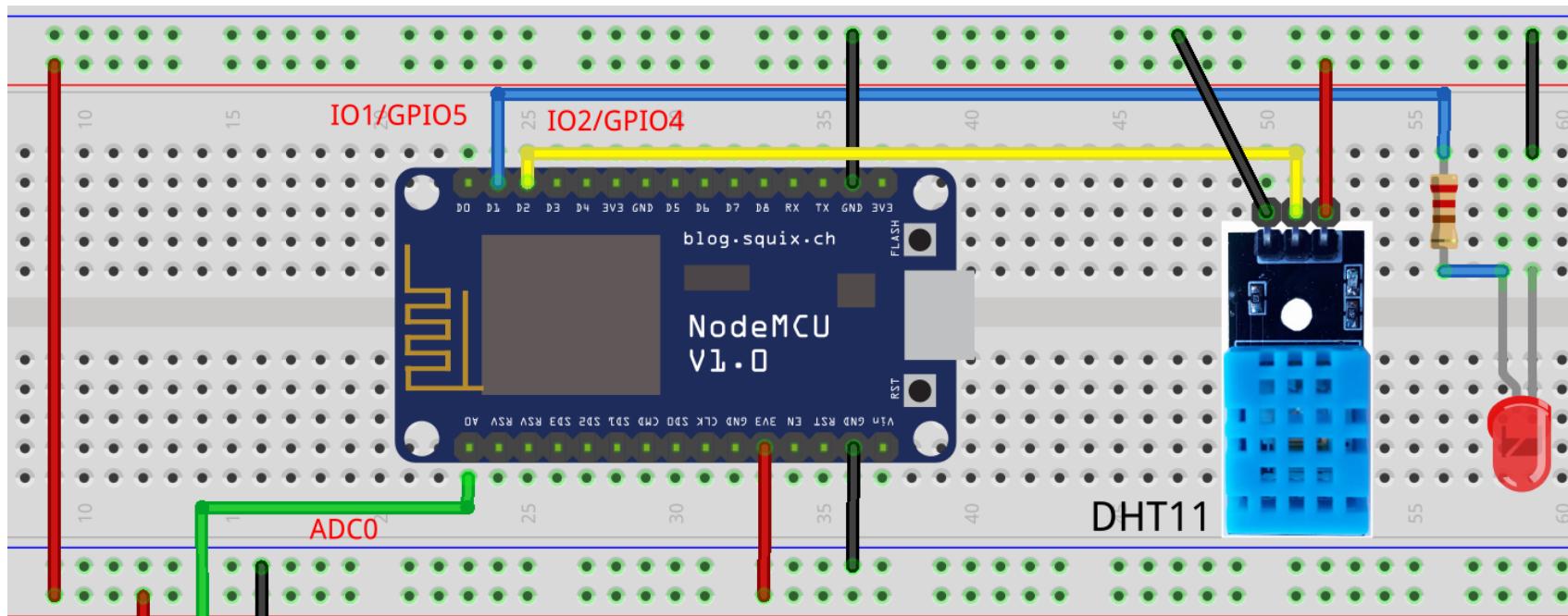
# Gravando o firmware (interpretador LUA dentro do Esp8266)



Na parte superior direita da janela temos:

- A porta em que o NodeMcu está configurado
- A velocidade de comunicação serial 9600 bauds
- Botão “open” que inicia a comunicação com o NodeMcu

# O esquema das conexões na protoboard



fritzing

IO index	ESP8266 pin	IO index	ESP8266 pin
0[*]	GPIO16	7	GPIO13
1	GPIO5	8	GPIO15
2	GPIO4	9	GPIO3
3	GPIO0	10	GPIO1
4	GPIO2	11	GPIO9
5	GPIO14	12	GPIO10
6	GPIO12		

# **Gravando e executando códigos LUA com a IDE Esplorer**



A Plataforma Open Source para IOT



A Plataforma Open Source para IOT

- Podemos enviar dados para o ThingSpeak
- Criar visualizações instantâneas de dados ao vivo
- Enviar alertas usando serviços da web como Twitter



A Plataforma Open Source para IoT

- O ThingSpeak permite que engenheiros e cientistas criem protótipos e sistemas IoT sem configurar servidores ou desenvolver software para Web



A Plataforma Open Source para IOT

- ThingSpeak é uma plataforma IoT que permite **coletar, armazenar, analisar, visualizar e atuar** em dados de sensores ou atuadores, como Arduino, Raspberry Pi, BeagleBone Black, Esp8266 e outros equipamentos



A Plataforma Open Source para IoT

- O elemento principal da atividade ThingSpeak é o canal, que contém campos de dados, campos de localização e um campo de status
- Depois de criar um canal ThingSpeak, você pode gravar dados no canal, processar e visualizar os dados com o código MATLAB e reagir aos dados com tweets e outros alertas



A Plataforma Open Source para IoT

O fluxo de trabalho típico ThingSpeak permite:

1. Crie um canal e colete dados
2. Analise e visualize os dados
3. Atualize os dados

## 1. Primeiro crie uma conta no ThingSpeak

### Create MathWorks Account

Email Address 

Missing required information

 To access your organization's MATLAB license, use your school or work email.

User ID 

Password  

Brazil 

First Name

Last Name

## 2. Crie um canal para sua aplicação

### My Channels

The screenshot shows the 'My Channels' section of the ThingSpeak web interface. At the top, there is a green button labeled 'New Channel'. Below it is a table with three columns: 'Name', 'Created', and 'Updated At'. The first row contains the channel information:

Name	Created	Updated At
NodeMcu - DHT11	2017-05-07	2017-10-22 15:53

Below the table, there is a horizontal navigation bar with six items: 'Private', 'Public', 'Settings', 'Sharing', 'API Keys', and 'Data Import / Export'. The 'Public' item is highlighted with a blue border.

## 3. Configure o canal da sua aplicação

### New Channel

Name

Description

Field	Field Name	Selected
1	Temperatura	<input checked="" type="checkbox"/>
2	Humidade	<input checked="" type="checkbox"/>
3		<input type="checkbox"/>
4		<input type="checkbox"/>
5		<input type="checkbox"/>
6		<input type="checkbox"/>
7		<input type="checkbox"/>
8		<input type="checkbox"/>

## 3. Configure o canal da sua aplicação

### NodeMcu - DHT11

Channel ID: 269034

Author: tccmrocha

Access: Public

Sensor de temperatura e umidade dht11 ligado ao NodeMcu.

Private View

Public View

Channel Settings

Sharing

API Keys

Data Import / Export

 Add Visualizations

 Data Export

### Channel Stats

Created: 5.months.ago

Updated: 20.minutes.ago

Entries: 0

## 4. Compartilhando o canal



### Channel Sharing Settings

- Keep channel view private
- Share channel view with everyone
- Share channel view only with the following users:



Email  
Address

Enter email here

Add User

## 5. Adicionando visualizações

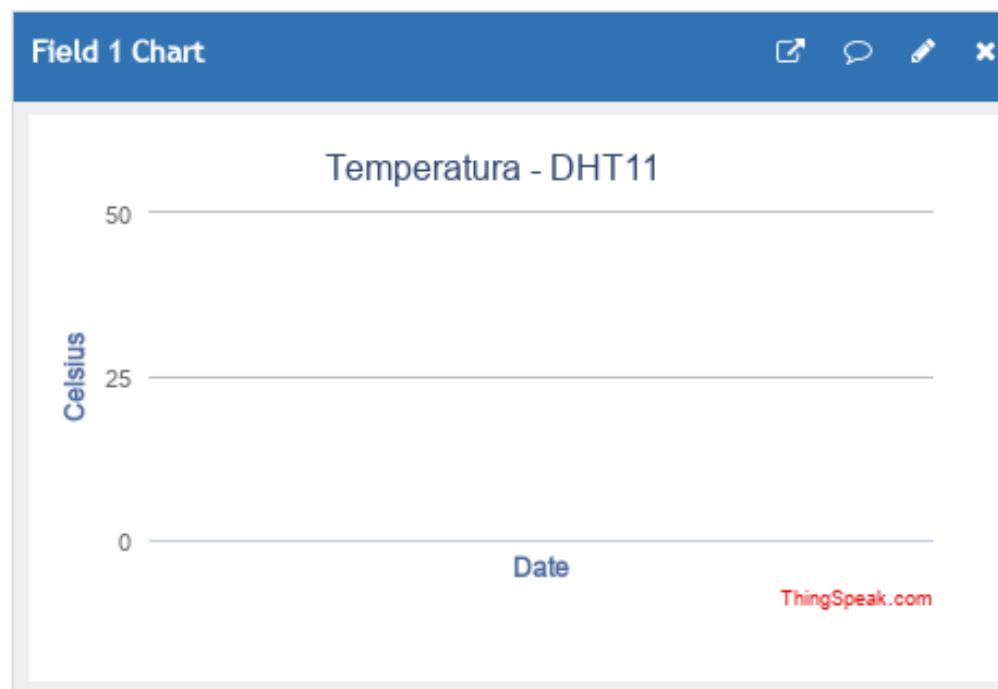


### Channel Stats

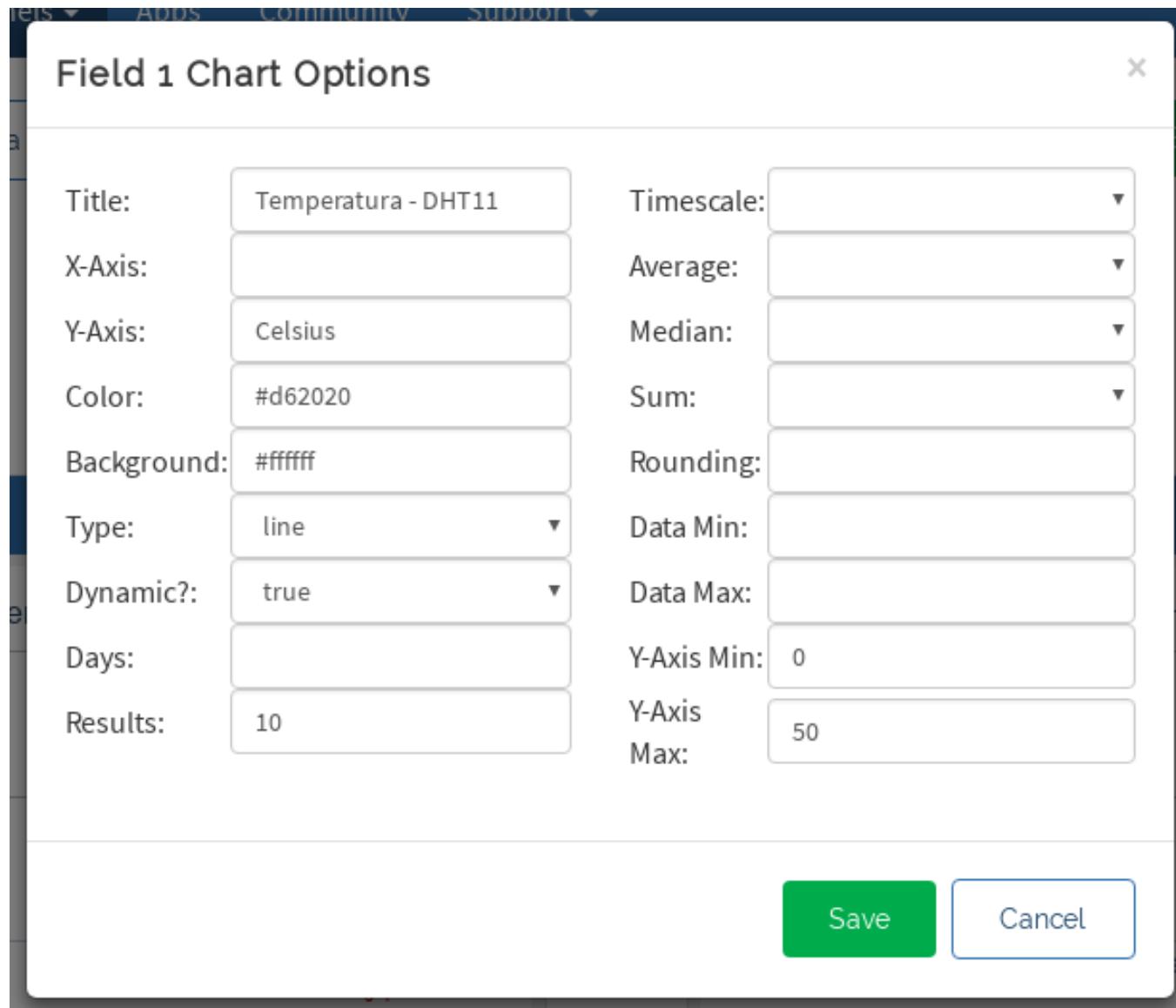
Created: 5.months.ago

Updated: 3.minutes.ago

Entries: 0



## 5. Editando as propriedades do gráfico



## 6. Anotando o Channel ID e API Keys

### NodeMcu - DHT11

Channel ID: 269034  
Author: tccmrocha  
Access: Public

Sensor de temperatura e umidade NodeMcu.

Private View    Public View    Channel Settings    Sharing    API Keys

Write API Key

Key RBAYI4C7T5VXHQTT

Generate New Write API Key

Read API Keys

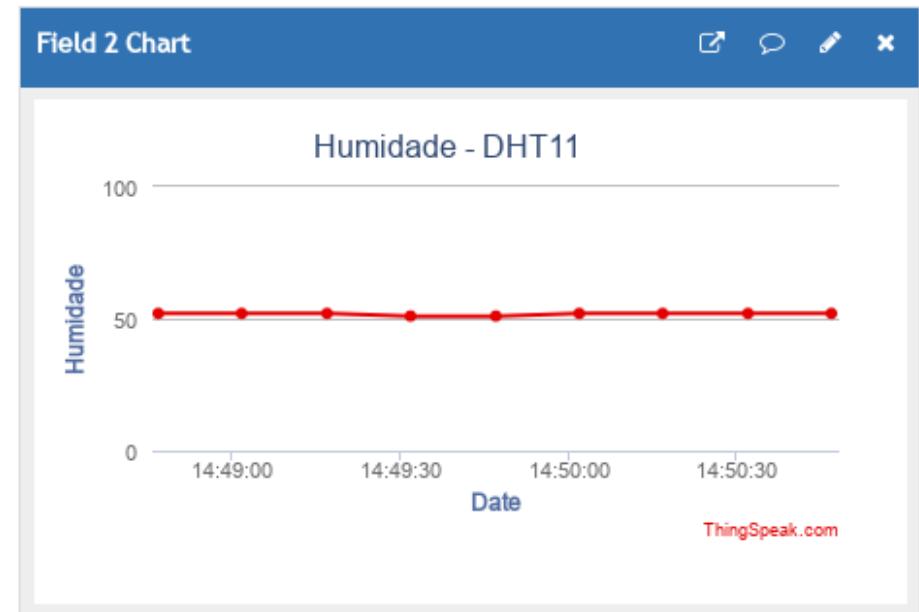
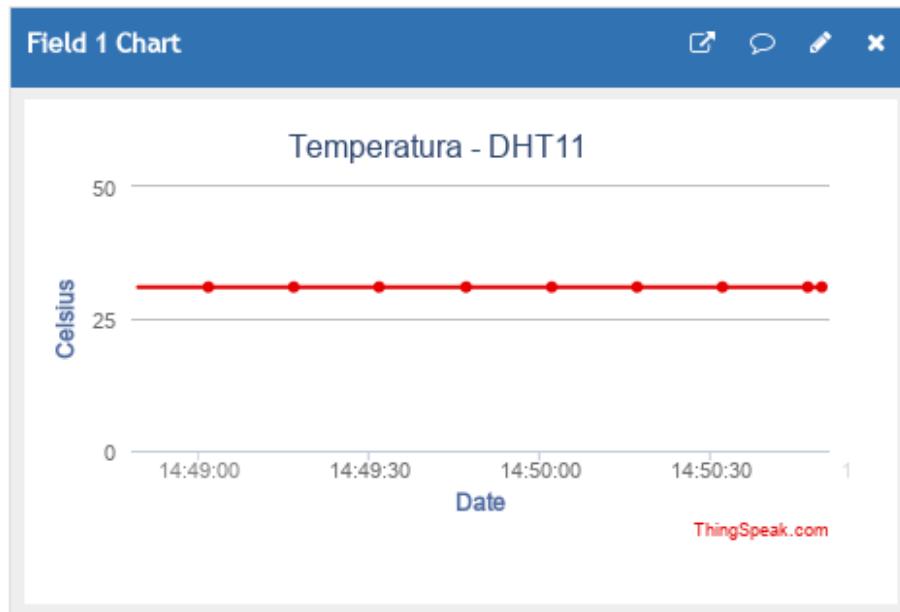
Key 7RWU085NZ96LB8ZG

The screenshot shows a ThingSpeak channel page for a NodeMcu - DHT11 sensor. At the top left, the Channel ID '269034' is circled in red. Below it, the Author 'tccmrocha' and Access level 'Public' are listed. To the right, a description reads 'Sensor de temperatura e umidade NodeMcu.'. A horizontal navigation bar below the channel info includes 'Private View', 'Public View', 'Channel Settings', 'Sharing', and 'API Keys'. The 'API Keys' tab is highlighted with a blue border. An arrow points from the left towards the 'API Keys' tab. Below the navigation bar, there are two sections: 'Write API Key' and 'Read API Keys'. Each section contains a 'Key' label and a text input field containing a long alphanumeric string. The 'Write API Key' section also features a 'Generate New Write API Key' button. Both the 'Key' labels and their respective text inputs are circled in red.

## Voltando ao NodeMcu/Esplorer/LUA

- Analisando o código
- Fazendo o upload do código
- Rodando a aplicação
- Visualizando o gráfico no ThingSpeak

## Visualizando o gráfico no ThingSpeak



Isso é tudo pessoal!!!



That's all Folks!