



ENG-1450

Microcontroladores e Sistemas Embarcados

– NodeMCU –

<http://www.inf.puc-rio.br/~abranco/eng1450/>

Roteiro

Parte A

1. NodeMCU – Hw + Sw
2. Biblioteca de funções
3. Introdução à linguagem LUA
4. Tarefas - A

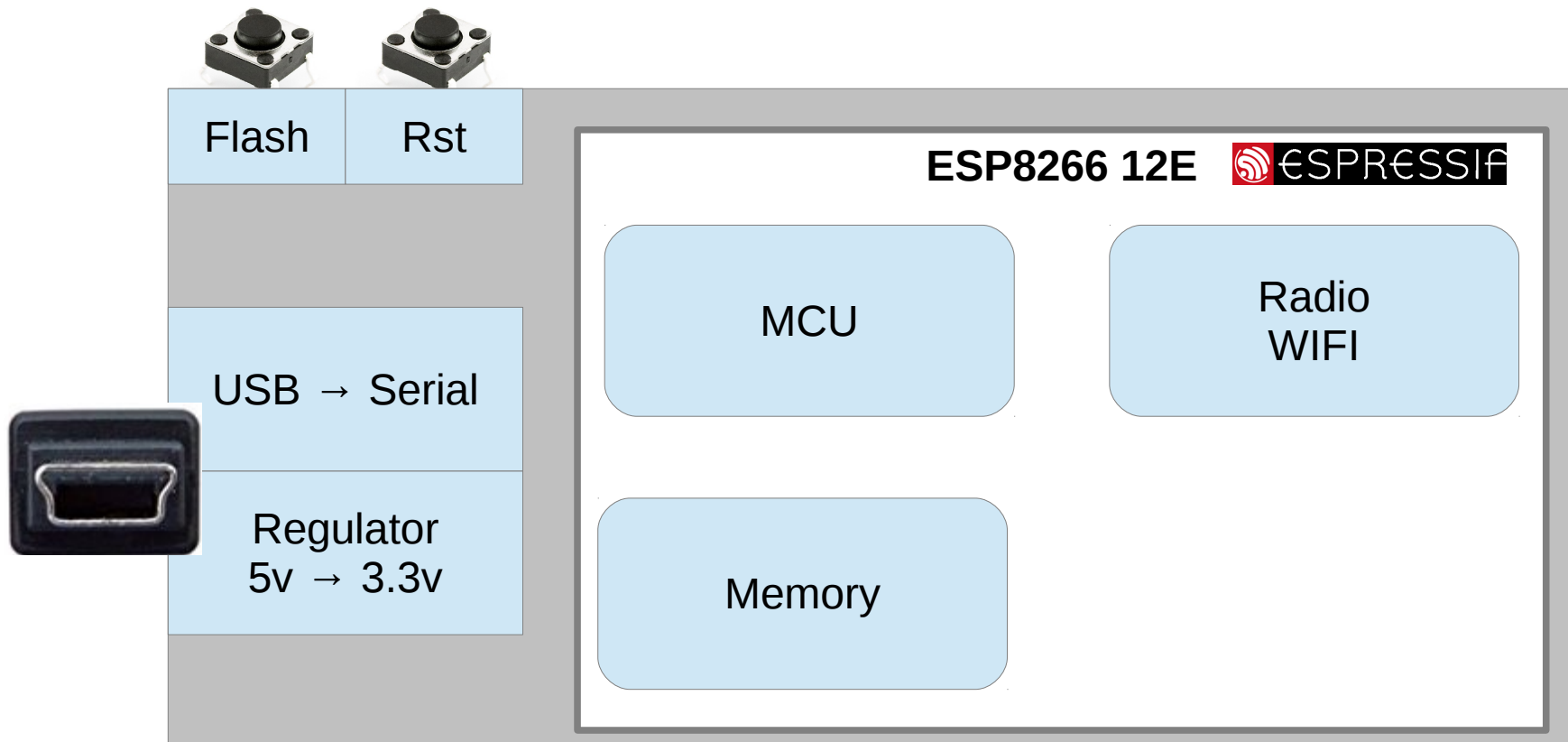
Parte B

5. Facilitador de Interface Web
6. Exemplo de uso
7. Tarefas - B

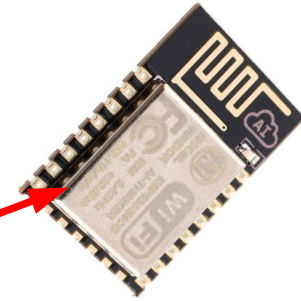
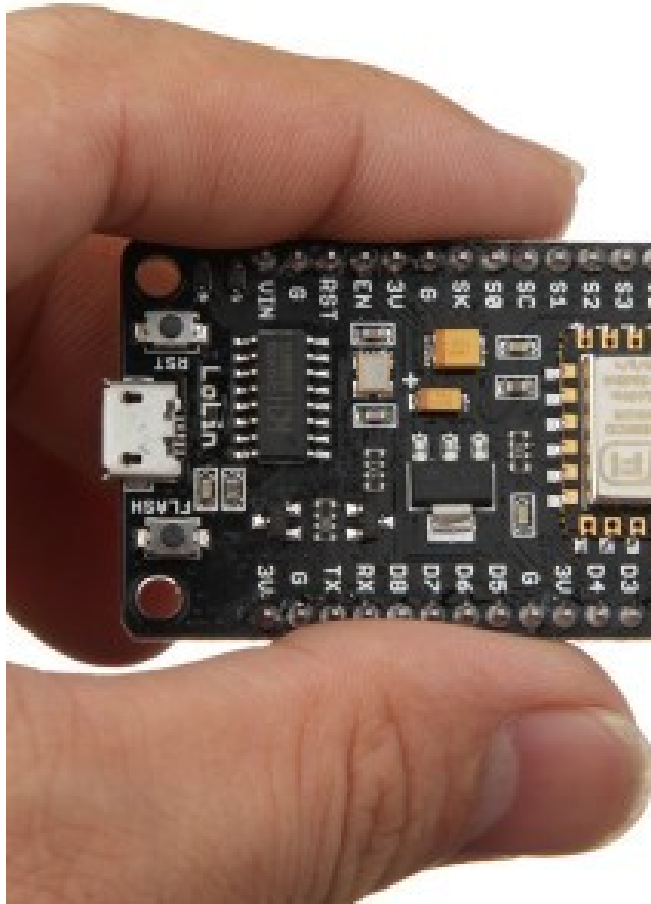
NodeMCU – HW + SW



NodeMCU DevKit

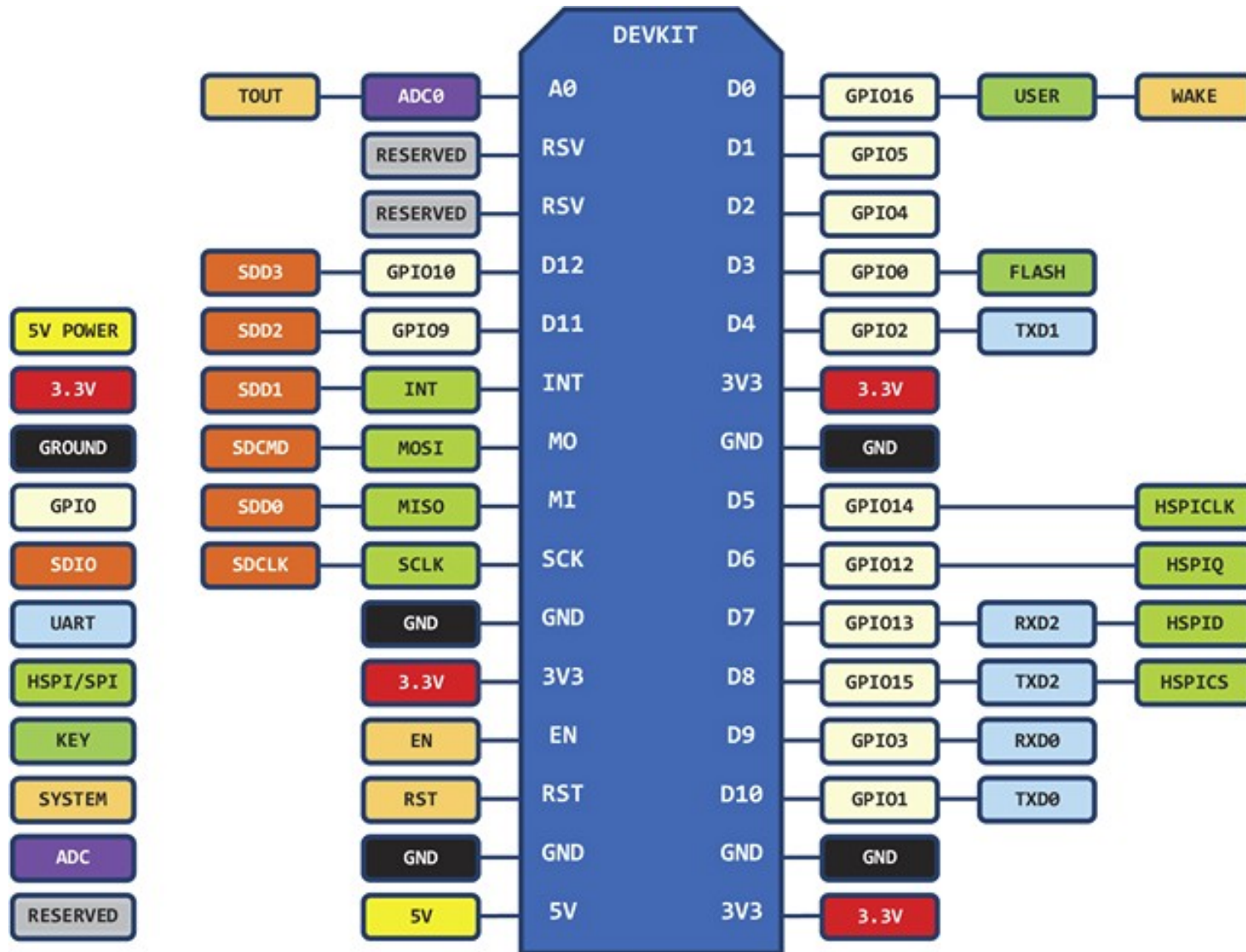


NodeMCU



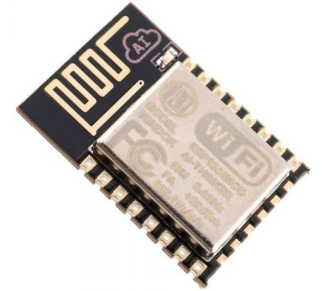
ESP8266 12E

NodeMCU – Pin definition



D0(GPI016) can only be used as gpio read/write, no interrupt supported, no pwm/i2c/ow supported.

Programação – ESP8266



- Ambientes + Biblioteca do fabricante
 - Diversos ambientes de programação → ?
- Biblioteca do fabricante (Espressif)
 - Timer, Sleep Mode, Task, etc..
 - Flash memory
 - WI-FI – AP, Station, SNTP, WPA2
 - TCP/IP, UDP/IP
 - GPIO, ADC, UART/Serial, I2C, PWM

Ambientes de programação

SDKs para ESP8266

- C
 - Espressif SDK
 - Arduino
- Python
 - MicroPython
- **Lua**
 - **NodeMCU**
 - LuaNode
- Basic
 - Esp8266 BASIC
 - Zbasic for ESP8266
- JavaScript
 - Espruino

Obs: Alguns SDKs são mais instáveis que outros.

ESPlorer - "IDE" para nodeMCU

The screenshot displays the ESPlorer v0.2.0-rc2 interface. The left pane shows the code editor with a Lua script named 'init.lua'. The right pane shows the terminal output, which includes the command 'PORT OPEN 9600' and a message indicating that the firmware could not be auto-detected. The bottom of the interface features a control panel with buttons for saving, compiling, and uploading code to the device.

Conexão USB

Comandos disco interno

Área de edição

Saída da execução

Comandos salvar+upload

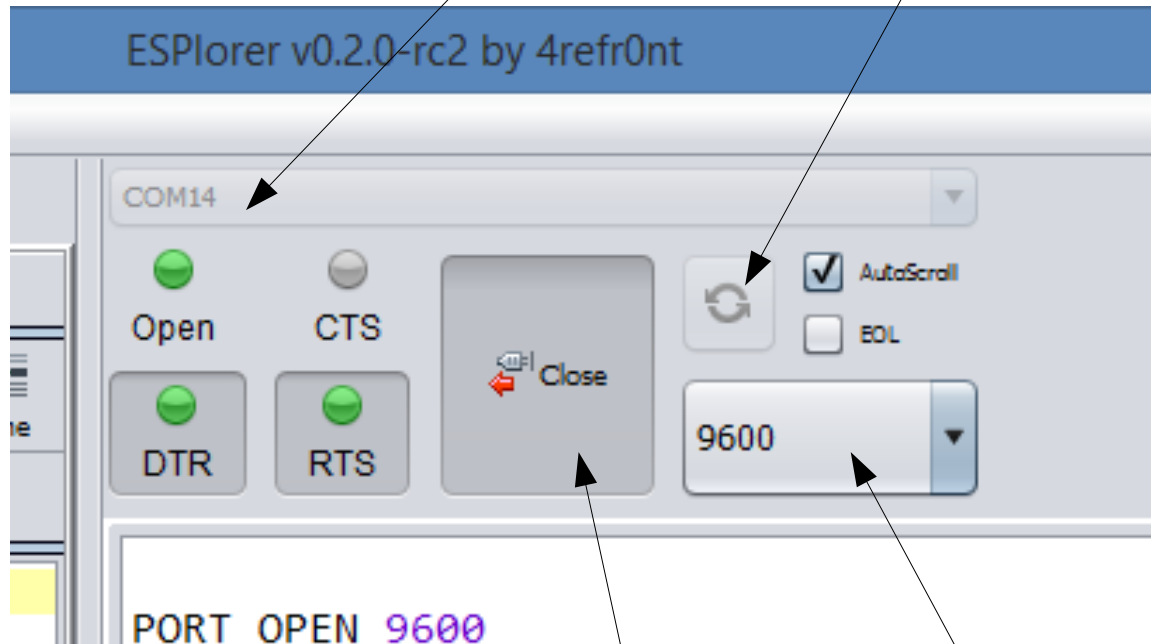
```
1 -- Config
2 local pin = 4 --> GPIO2
3 local value = gpio.LOW
4 local duration = 1000 --> 1 second
5
6 -- Function toggles LED state
7 function toggleLED ()
8   if value == gpio.LOW then
9     value = gpio.HIGH
10  else
11    value = gpio.LOW
12  end
13
14  gpio.write(pin, value)
15 end
16
17 -- Initialise the pin
18 gpio.mode(pin, gpio.OUTPUT)
19 gpio.write(pin, value)
20
21 -- Create an interval
22 tmr.alarm(0, duration, 1, toggleLED)
```

```
PORT OPEN 9600
Communication with MCU...
Got answer! AutoDetect firmware...
Can't autodetect firmware, because proper answer not received.
> file.remove("init.lua");
> file.open("init.lua","w+");
> w = file.writeline
> w([[-- Config]]);
> w([[local pin = 4 --> GPIO2]]);
> w([[local value = gpio.LOW]]);
> w([[local duration = 1000 --> 1 second]]);
> w([[function toggleLED ()
> w([[   if value == gpio.LOW then
> w([[     value = gpio.HIGH
> w([[   else
> w([[     value = gpio.LOW
> w([[   end
> w([[   gpio.write(pin, value)
> w([[ end
> w([[ ]]
> w([[ -- Initialise the pin
> w([[ gpio.mode(pin, gpio.OUTPUT)
> w([[ gpio.write(pin, value)
> w([[ ]]
> w([[ -- Create an interval
> w([[ tmr.alarm(0, duration, 1, toggleLED)
> file.close();
> dofile("init.lua");
>
```

Conexão USB

Porta USB selecionada

Verifica portas disponíveis



Abre ou Fecha a conexão USB

Selecionar Baudrate
115200

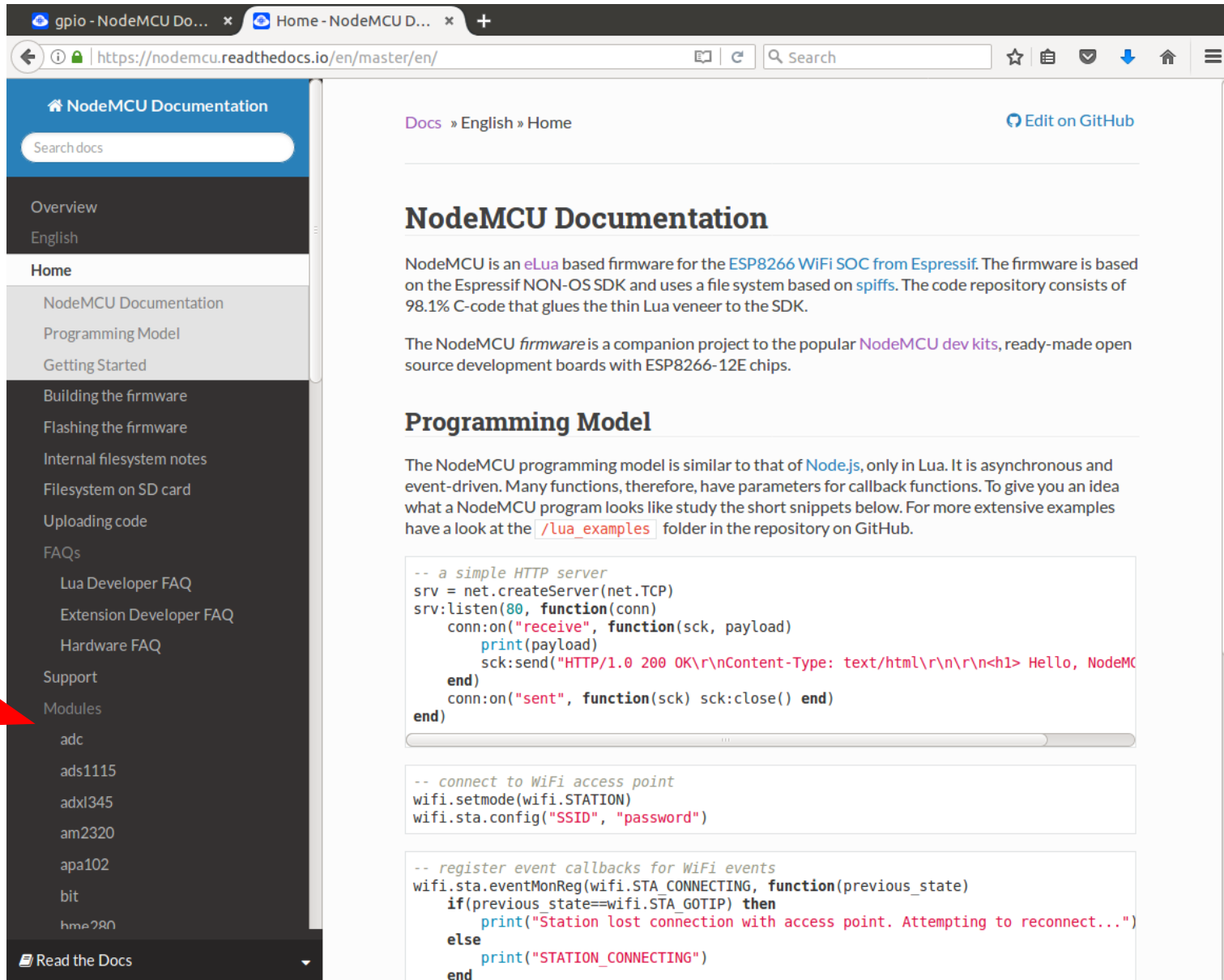
Obs: Para evitar erro de time-out na conexão, ao conectar o ESPlorer deve-se apertar o botão de reset no NodeMCU.

Biblioteca NodeMCU

Lua 5.1.4

NodeMCU SDK

<https://nodemcu.readthedocs.io/>



The screenshot shows the NodeMCU Documentation website. The browser address bar displays `https://nodemcu.readthedocs.io/en/master/en/`. The left sidebar contains a navigation menu with the following items: Overview, English, Home, NodeMCU Documentation, Programming Model, Getting Started, Building the firmware, Flashing the firmware, Internal filesystem notes, Filesystem on SD card, Uploading code, FAQs, Lua Developer FAQ, Extension Developer FAQ, Hardware FAQ, Support, Modules, adc, ads1115, adxl345, am2320, apa102, bit, hmc2901, and Read the Docs. A red arrow points to the 'Modules' item in the sidebar, with the word 'Módulos' written in red text next to it. The main content area shows the 'NodeMCU Documentation' page, which includes a search bar, a breadcrumb trail 'Docs » English » Home', and an 'Edit on GitHub' link. The main heading is 'NodeMCU Documentation', followed by a paragraph describing the firmware: 'NodeMCU is an eLua based firmware for the ESP8266 WiFi SOC from Espressif. The firmware is based on the Espressif NON-OS SDK and uses a file system based on spiiffs. The code repository consists of 98.1% C-code that glues the thin Lua veneer to the SDK.' Below this is another paragraph: 'The NodeMCU firmware is a companion project to the popular NodeMCU dev kits, ready-made open source development boards with ESP8266-12E chips.' The next section is 'Programming Model', which states: 'The NodeMCU programming model is similar to that of Node.js, only in Lua. It is asynchronous and event-driven. Many functions, therefore, have parameters for callback functions. To give you an idea what a NodeMCU program looks like study the short snippets below. For more extensive examples have a look at the /lua_examples folder in the repository on GitHub.' Three code snippets are provided: 1. A simple HTTP server:

```
-- a simple HTTP server
srv = net.createServer(net.TCP)
srv:listen(80, function(conn)
  conn:on("receive", function(sck, payload)
    print(payload)
    sck:send("HTTP/1.0 200 OK\r\nContent-Type: text/html\r\n\r\n<h1> Hello, NodeMCU")
  end)
  conn:on("sent", function(sck) sck:close() end)
end)
```

 2. Connect to WiFi access point:

```
-- connect to WiFi access point
wifi.setmode(wifi.STATION)
wifi.sta.config("SSID", "password")
```

 3. Register event callbacks for WiFi events:

```
-- register event callbacks for WiFi events
wifi.sta.eventMonReg(wifi.STA_CONNECTING, function(previous_state)
  if(previous_state==wifi.STA_GOTIP) then
    print("Station lost connection with access point. Attempting to reconnect...")
  else
    print("STATION_CONNECTING")
  end
end)
```

NodeMCU SDK

<https://nodemcu.readthedocs.io/>

- Módulos instalados no NodeMCU do LCA
 - GPIO
 - ADC, GPIO, PWM
 - Rede e Comunicação
 - MQTT, HTTP, Net(TCP,UDP), WiFi
 - Interfaces seriais
 - UART, SPI, 1-wire, I²C
 - Suporte
 - Node, Timer, File, Bit
 - Sensor Humidade/Temperatura DHT11
 - DHT

Exemplos de uso

-- Configura pin 1 como output e escreve valor alto

```
pin = 1
```

```
gpio.mode(pin, gpio.OUTPUT)
```

```
gpio.write(pin, gpio.HIGH)
```

-- Configura WIFI como Ponto de Acesso (AP)

```
wifi.setmode(wifi.SOFTAP)
```

```
wifi.ap.config({ssid=" RedeTeste" ,pwd=" 12345678" })
```

```
wifi.ap.setip({ip=" 192.168.0.1" ,netmask="255.255.255.0" ,gateway=" 192.168.0.1" })
```



Linguagem LUA

www.lua.org

Learn Lua in 15 Minutes

<http://tylerneylon.com/a/learn-lua/>

<https://coronalabs.com/learn-lua/>

Reference Manual (5.1)

<http://www.lua.org/manual/5.1/>

Book Programming in Lua (5.0)

<http://www.lua.org/pil/contents.html>

Google....

Projetos que utilizam Lua [[editar](#) | [editar código-fonte](#)]

Em 2013, a [Wikimedia Foundation](#) começou a utilizar a linguagem nas predefinições.^[5]

Projetos [[editar](#) | [editar código-fonte](#)]

- [Adobe Photoshop Lightroom](#)
- [Celestia](#)
- [Cheat Engine](#)
- [ClanLib](#)
- [CryEngine 3](#)
- [Corona SDK](#)
- [Damn Small Linux](#)
- [Ginga](#)
- [Kepler \(software\)](#)
- [lighttpd](#)
- [Liquid Feedback](#)
- [MinGW](#)
- [Monotone](#)
- [Nmap](#)
- [PlayStation Home](#)

Jogos [[editar](#) | [editar código-fonte](#)]

Exemplos de empresas que desenvolveram jogos usando a linguagem Lua: LucasArts, Croteam, BioWare, Microsoft, Relic Entertainment, [Absolute Studios](#), [Monkeystone Games](#)^[2], [Blizzard](#)^[5], SNKPlaymore, Facepunch Studios.

- [Angry Birds](#)
- [Baldur's Gate](#)^[2]
- [The Battle for Wesnoth](#)^[6]
- [Civilization V](#)
- [Escape from Monkey Island](#)^[2]
- [Fable II](#)
- [Far Cry](#)^[2]
- [FlyFF](#)
- [Freeciv](#)
- [Freelancer](#)
- [Garry's Mod](#)
- [Grim Fandango](#)^[2]
- [Impossible Creatures](#)^[2]
- [Lego Universe](#)
- [MapleStory](#)
- [MDK2](#)
- [Monopoly Tycoon](#)
- [Multi Theft Auto](#)
- [Psychonauts](#)^[2]
- [Ragnarok Online](#)
- [Roblox](#)
- [Street Fighter IV](#)
- [The King of Fighters XIII](#)
- [Tibia](#)
- [Transformice](#)
- [The Talos Principle](#)
- [World of Warcraft](#)^[5]

Learn Lua in 15 Minutes

<http://tylernelson.com/a/learn-lua/>

Learn Lua in 15 Minutes

```
-- Two dashes start a one-line comment.

--[
  Adding two ['s and ]'s makes it a
  multi-line comment.
--]]

-----

-- 1. Variables and flow control.
-----

num = 42 -- All numbers are doubles.
-- Don't freak out, 64-bit doubles have 52 bits for
-- storing exact int values; machine precision is
-- not a problem for ints that need < 52 bits.

s = 'walternate' -- Immutable strings like Python.
t = "double-quotes are also fine"
u = [[ Double brackets
      start and end
      multi-line strings.]]
t = nil -- Undefined t; Lua has garbage collection.

-- Blocks are denoted with keywords like do/end:
while num < 50 do
  num = num + 1 -- No ++ or += type operators.
end

-- If clauses:
if num > 40 then
  print('over 40')
elseif s ~= 'walternate' then -- ~= is not equals.
  -- Equality check is == like Python; ok for strs.
  io.write('not over 40\n') -- Defaults to stdout.
else
  -- Variables are global by default.
  thisIsGlobal = 5 -- Camel case is common.
```

Tarefas – Parte A

1. Timer + Print

- `Tmr.alarm()` + `print()`

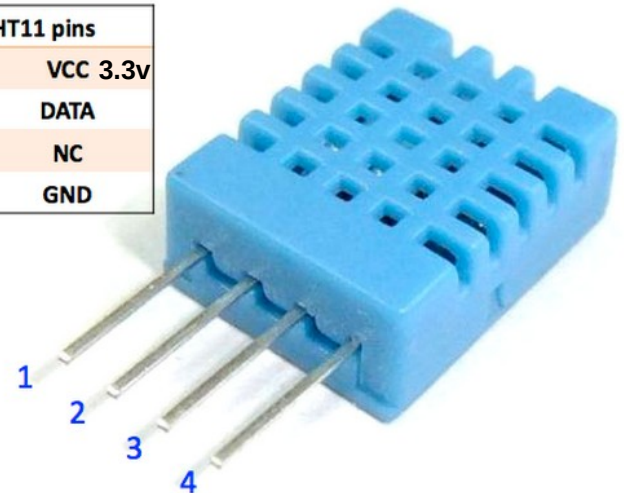
2. LED Blink

- `Tmr.alarm()` + `gpio.mode()` + `gpio.write()`

3. Sensor de Temperatura e Humidade DHT11

- `Tmr.alarm()` + `dht.read()`
+ `print()`

DHT11 pins	
1	VCC 3.3v
2	DATA
3	NC
4	GND



Web Server no NodeMCU

- Servidor para conexões TCP/HTTP.
- Permite conexões de navegadores Web.
- Quantidade limitada a 5 conexões.
- A aplicação tem que processar a requisição HTML e gerar uma página HTML.
- Qualquer uso exagerado do sistema implica em falhas e estouro de memória.
- Ideal para páginas simples e acesso de um usuário.

Facilitador de Interface Web

Controles pré-definidos para facilitar as construções HTML.



The screenshot shows a web browser window with the address bar displaying 'http://192.168.3.1/'. The page content includes:

- PUC Rio - DEE/DI** with an **Atualizar** button.
- NodeMCU Web Server** title.
- Text: **Meu texto...**
- System info: **Tempo: 504.20 segundos. Heap:17432 [1448]**
- UART:USB : **PIC** **USB**
- Temperatura: 0.0°C**
- Umidade: 0.0%**
- Led 1:OFF : **ON** **OFF**
- Led 2:Apagado
- Acoes: **Comando 1** **Envia Acoes**
- Teste: **Opt1** **Envia Teste**
- Slider: [0:255] with a slider bar at 0 and **Envia Slider** button.
- InText: **Envia InText** button.

Facilitador de Interface Web

`html.outText(label,text)`

`html.time()`

`html.booleanState(label,id,text_true,text_false)`

`html.sensor(label, valor, formato, unidade)`

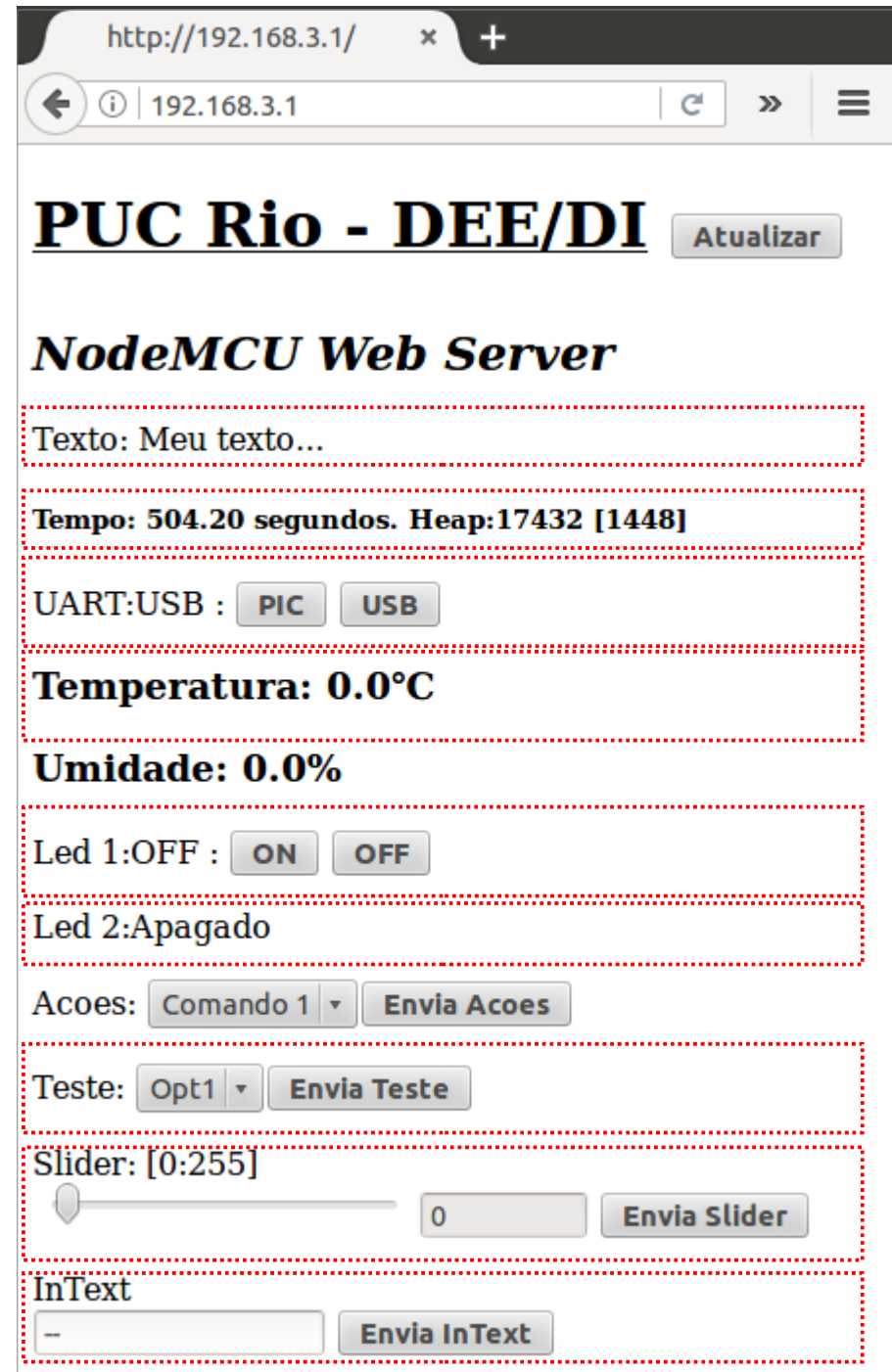
`html.pin_wr(label,pin,text_high,text_low)`

`html.pin_rd(label,pin,text_high,text_low)`

`html.select(label,optTable,id)`

`html.slider(label,id, min, max)`

`html.inText(label,id,size)`



Facilitador de Interface Web

html.**outText**(label,text) → Imprime label+text

html.**time**() → Imprime o relógio interno e memória restante.

html.**booleanState**(label,id,text_true,text_false) → Controle para variável True/False

html.**sensor**(label, valor, formato, unidade) → Imprime valor+unidade do sensor

html.**pin_wr**(label,pin,text_high,text_low) → Controle de High/Low diretamente no pino

html.**pin_rd**(label,pin,text_high,text_low) → Exibe o valor do pino.

html.**select**(label,optTable,id) → Seleção de um item de uma lista

html.**slider**(label,id, min, max) → Seleção de um valor dentro de um faixa

html.**inText**(label,id,size) → Entrada de valor digitado.

Caracteres alfanuméricos e () [] : ; . , - _ *

Uso dos controles

Por exemplo, a definição do seguinte controle:

```
page = page .. html.select("Teste", {"Opt1", "Opt2"}, "myOpts");
```

espera a função `myOpts` dentro da tabela `user.execCommand`

```
user.execCommand = {  
    myOpts = function(value)  
        print("myOpt:", value);  
    end,  
}
```

Facilitador de Interface Web

- Programa principal do framework:
 - **mainService.lua**
 - Obs: Normalmente não deve ser alterado pelo usuário
- Usuário define um módulo para ser usado pelo “framework”.
 - **userScript.lua**
- Arquivos auxiliares para criação/conexão da/na rede
 - **configAP.lua** ou **configStation.lua**
 - Contém a conexão do AP ou da Station.
 - A configuração fica no userScript.lua

Componentes do userScript.lua

```
local user = {}
user.ap={ssid="LCA-10",pwd="12345678",ip="192.168.3.1"}
-- user.station={ssid="terra_iot",pwd="projeto_iot"}

function user.setup()
    ...
end

function user.page(html)
    ...
end

user.execCommand = {
    newRequest = function() ... end,
    setPin = function(pin,value) ... end,
    ... -- Outras funções dos controles do usuário.
}

return user
```

Ações/Funções dos controles

html.**outText**(label,text)

html.**time**()

html.**booleanState**(label,**id**,text_true,text_false) → **id**(value); value: 0 ou 1

html.**sensor**(label, valor, formato, unidade)

html.**pin_wr**(label,pin,text_high,text_low) → **setPin**(pin,value); pin: #pin; value: 0 ou 1

html.**pin_rd**(label,pin,text_high,text_low)

html.**select**(label,optTable,**id**) → **id**(value); value: Posição da opção na lista

html.**slider**(label,**id**, min, max) → **id**(value); value: valor selecionado

html.**inText**(label,**id**,size) → **id**(value); value: Texto digitado

Obs: Id(value) → id = function(value) end;

Exemplo simples para UserScript.lua

```
local user = {}
user.ap={ssid="LCA-10",pwd="12345678",ip="192.168.3.1"}

-- Configuração do pino do LED
local gbl={ledPin = 3}

function user.setup()
    -----
    -- Configurações/Inicialização dos pinos utilizados
    -----

    gpio.mode(gbl.ledPin, gpio.OUTPUT)
    gpio.write(gbl.ledPin, gpio.LOW);
end

-----

-- Definição/Customização da Interface WEB
-----

function user.page(html)
    local page="";
    page = page .. html.time();
    page = page .. html.pin_wr("Led 1",gbl.ledPin, "ON", "OFF");
    return page;
end
```

```
-----  
-- Tratadores dos comandos do usuário para os select/options, sliders e setPin  
-----  
user.execCommand = {  
-- Função chamada após cada comando select/options, sliders e setPin.  
-- É executado antes de montar a página HTML  
newRequest = function()  
    -- Faz nada...  
    end,  
-- Função chamada para todos setPins  
setPin = function(pin,value)  
    gpio.write(pin,value);  
    print("setPin_"..pin.." = " .. value);  
    end,  
}  
  
return user;
```

Configuração da comunicação entre o NodeMCU e o PIC

Baudrate da USB no NodeMCU

- Só tem uma UART. No reset aponta para USB.
- O default do NodeMCU é 115200.
- O NodeMCU tenta identificar diferentes baudrates logo após o reset.
- Baseado nas mensagens enviadas logo após o reset o novo baurate é descoberto.
- Por exemplo, pode-se acionar o botão “Heap” no ESPlorer até o NodeMCU responder um texto legível.

PIC + NodeMCU

- O maior valor de baudrate no PIC que não dá erro de comunicação é 57600.
- Integração NodeMCU ↔ PIC via UART/Serial
 - Conectar o Tx/Rx no Rx/Tx:
 - PIC:Tx(C6) → NodeMCU:Rx2(D7)
 - PIC:Rx(C7) → NodeMCU:Tx2(D8)
 - PIC:GND → NodeMCU:GND
 - Configurar o ESPlorer, o PIC e o NodeMCU com o mesmo baudrate, por exemplo 57600.
 - Na conexão do ESPlorer, fazer o reset no NodeMCU e acionar o “Heap” até acertar o texto.

PIC + NodeMCU

- Programação no NodeMCU
 - `uart.alt(1)` e `uart.alt(0)`
 - 1 - Redireciona a UART da USB para os pinos D7 e D8.
 - 0 – Desfaz o redirecionamento.
 - `uart.setup(0,57600,8,uart.PARITY_NONE,uart.STOPBITS_1,1)`
 - Configura a UART
 - `printOn()` e `printOff()`
 - Ativa/Desativa a saída do `print()`.
 - `uart.write(0, text)`
 - Escreve *text* na UART selecionada.

PIC + NodeMCU

- Programação PIC

- `UART1_Init(57600);`

- Inicializa a UART

- `UART1_Data_Ready()`

- Verifica se existe alguma byte para ser lido

- `UART1_Read()`

- Lê um byte da UART

Tarefas – Parte B

1. Controle WEB

- Ligar/Desligar LED
- Ativar/Desativar Blink
- Alterar frequência do Blink

2. Controle WEB + PIC

- NodeMCU – enviar comando via UART.
- PIC – Imprimir comando da UART no LCD