TP ESP8266 MODE AUTONOME

Objectifs : Etre capable d'utiliser le module ESP8266 en mode standalone (fonctionnement sans microcontroleur externe)

Le module ESP8266 ESP01 comporte un microcontroleur intégré et 2 GPIO disponible (GP0, GP02).

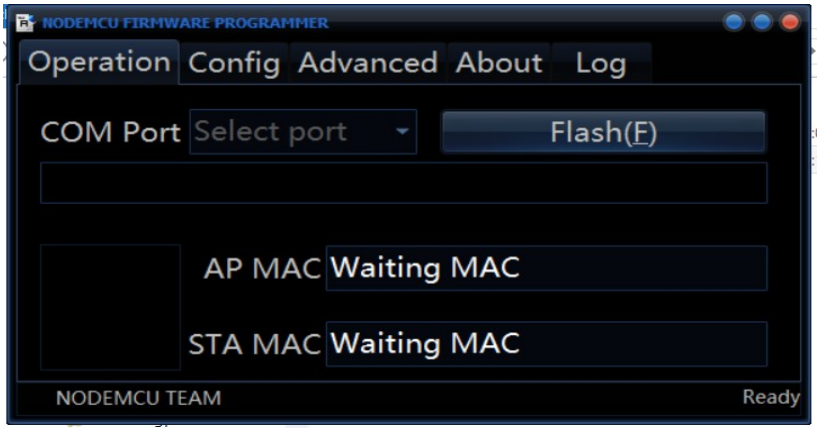
Afin de facilité le développement d'application on va utiliser une chaine de développement composé de NODEMCU + LUA (langage interprété dérivé du ANSI C)
Pour mettre à jour le firmware (flasher le module avec un fichier bianire .bin) on utilisera : nodemcuflasher

Télécharger :

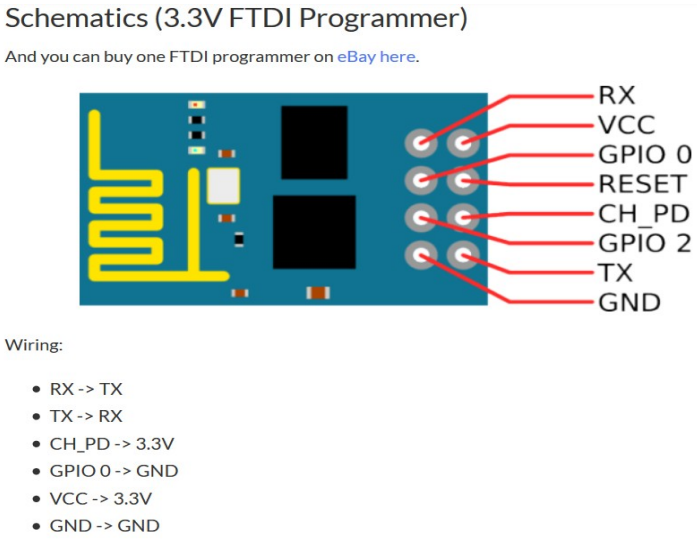
nodemcu-flasher-master.zip : le décompresser (le flasher + le .bin)
ESP8266_Lua_Uploader-master.zip : le décompresser

1. FLASHER LE MODULE

Pour flasher le module :
Placer GP0 à la masse et redémarrer le module.
Lancer : modemcu-flasher : ESP8266Flasher.exe



Brancher le module+adaptateur+USB/serie : le port serie est détecté automatiquement.

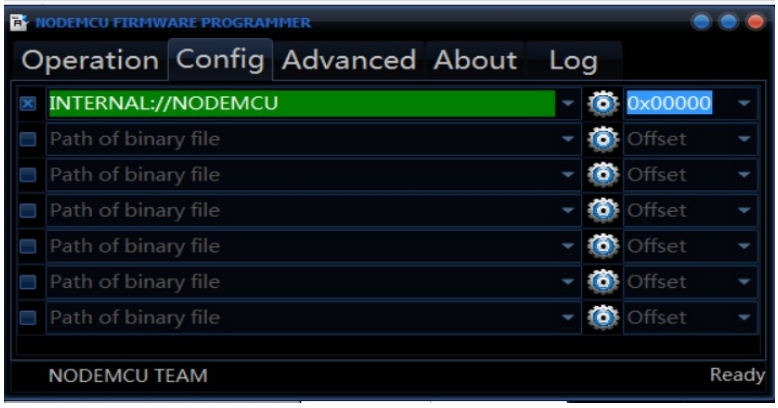


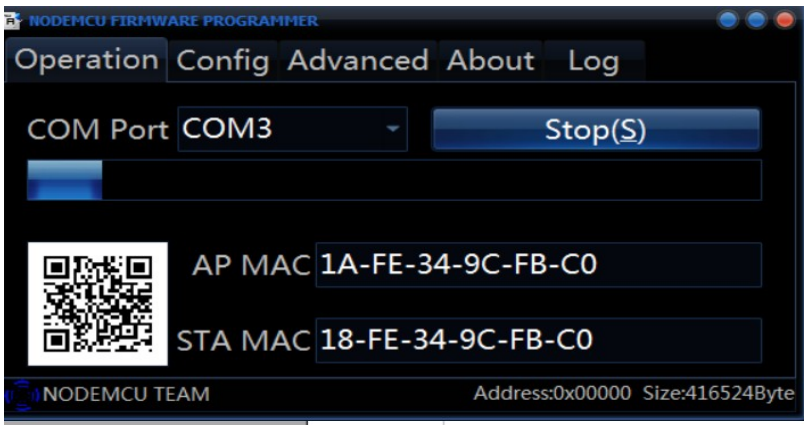
Cablage entre cable USB/TTL et adaptateur ESP8266 5V/3V

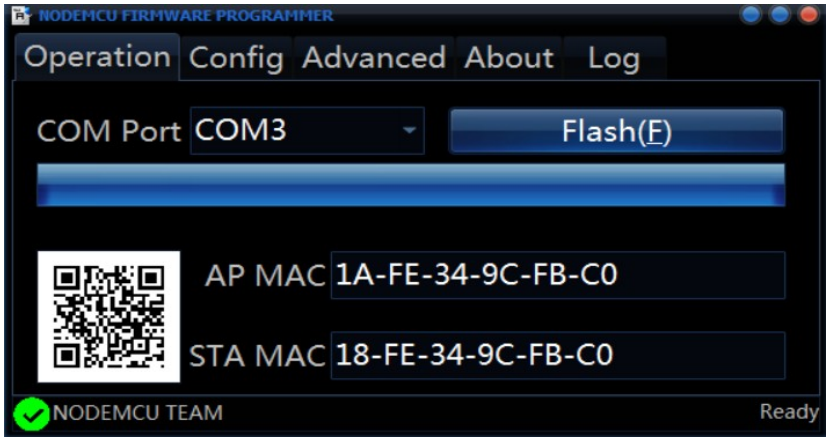
USB/TTL	**adaptateur**
rouge	+5V
vert	T
blanc	R
noir	0

Configurer les fichiers BIN à flasher :
Config+ Choisir le BIN (déjà prédéfini) + Adresse

Address	Size	Name	Description
00000h	248k	app.v6.flash.bin	User application
3E000h	8k	master_device_key.bin	OTA device key. **Unconfirmed**: Not used without OTA (on the air programmation)
40000h	240k	app.v6.irom0text.bin	SDK libraries
7C000h	8k	esp_init_data_default.bin	Default configuration, see note below
7E000h	8k	blank.bin	Filled with FFh. May be WiFi configuration





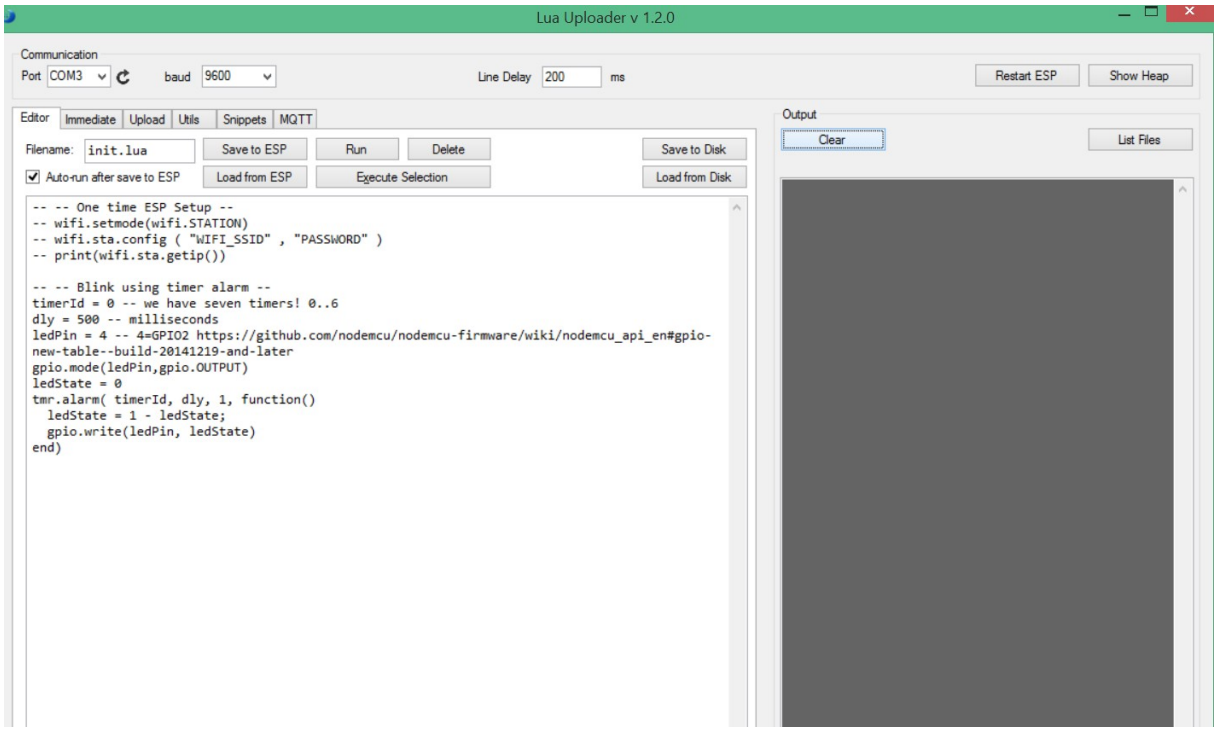


Une fois le module flasher, retirer le strap entre GP00 et GND.
Redémarrer le module
Maintenant vous pouvez utiliser LUA.

2. PROGRAMMER AVEC LUA (HTTP://WWW.LUA.ORG)

Installer le soft LUA uploader pour windows (LuaUploader.exe) afin de créer un programme et de programmer le module.

Lancer le soft.



Configurer le port série (COMx, 9600)
Vérification de la liaison avec le module

2.1. Onglet "utils"

Aller dans l'onglet "utils"
Brancher une Led+Résistance (275ohm min) entre GND et GP02
Dans GPIO :
Set GPIO mode [4]GP02 OUTPUT
set GPIO Value [4]GP02 OFF (la LED s'éteint)
set GPIO Value [4]GP02 ON (la LED s'allume)

2.2. Onglet "Editor"

Cet éditeur de texte permet d'écrire le code Lua, de l'exécuter dans le module et de programmer dans la ROM "SaveTo ESP"

2.3. Onglet "Immediat"

Cet éditeur de texte permet d'écrire du code Lua et de le tester immédiatement sans programmation dans la ROM programme.

3. EXEMPLES DE PROGRAMMES

3.1. Faire clignoter une LED (Tester ok)

Brancher la LED avec une résistance série 330ohm entre GP02 et GND.
Placer les lignes de code suivantes dans la partie "immédiat" ou "Editor" + SaveTo ESP + Run

```
-- -- Blink using timer alarm --
timerId = 0 -- we have seven timers! 0..6
dly = 500 -- milliseconds
ledPin = 4 -- 4=GPIO2 https://github.com/nodemcu/nodemcu-
firmware/wiki/nodemcu_api_en#gpio-new-table--build-20141219-and-later
gpio.mode(ledPin,gpio.OUTPUT)
ledState = 0
tmr.alarm( timerId, dly, 1, function()
  ledState = 1 - ledState;
  gpio.write(ledPin, ledState)
end)
```

La Led doit clignoter avec une période de 1seconde.

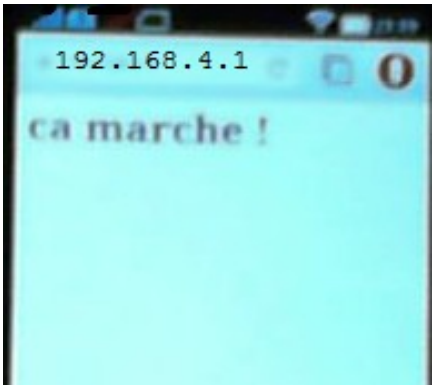
Important : Le courant max des GPIO est de 12mA ! Sous 3,3V donc avec une LED rouge de Vf = 1,2V on obtient :
R = (3,3 – 1,2) / 0,012 = 175 ohm = 220 ohm en E12 pour être tranquille !

3.2. Afficher une page HTML (Tester ok)

Placer le code suivant :

```
-- a simple http server
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
   conn:on("receive",function(conn,payload)
   print(payload)
   conn:send("<h1> ca marche ! </h1>")
   end)
end)
```

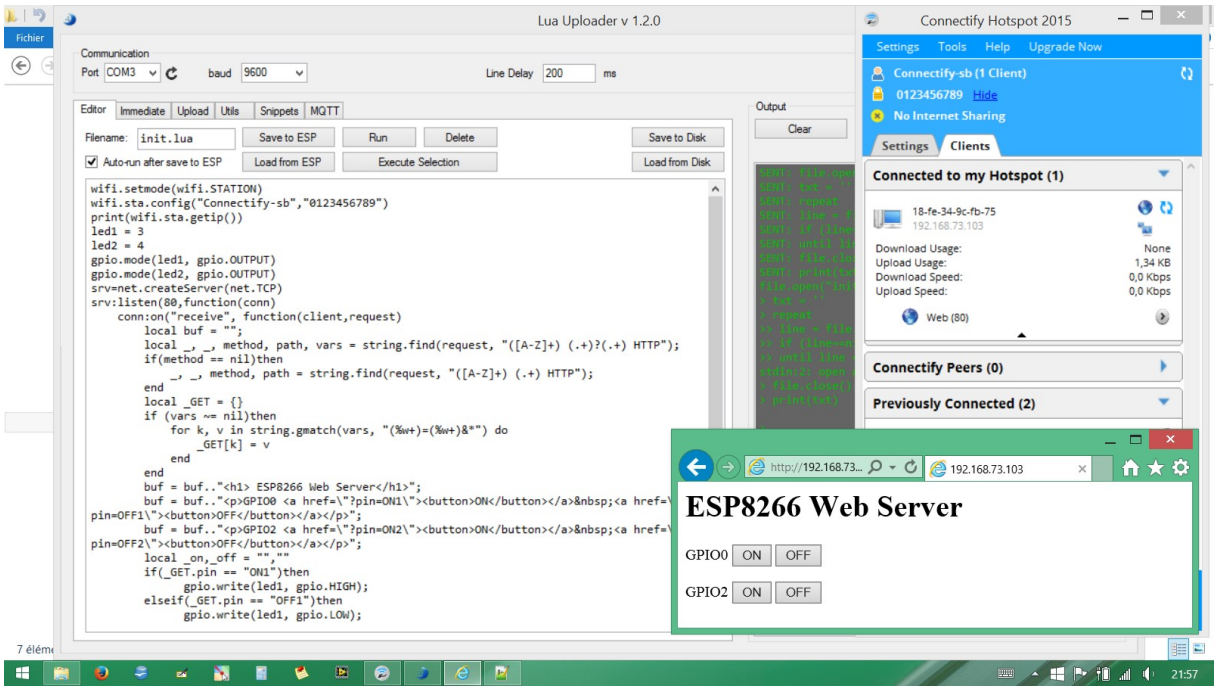
Vérifier le bon fonctionnement en allant à l'adresse IP du module avec un navigateur.
Une page HTML s'ouvre avec le texte "ca marche !".



3.3. Contrôler 2 LEDs (GP00 et GP02) avec une page Web (Tester ok)

Code Lua :

```
wifi.setmode(wifi.STATION)
wifi.sta.config("YOUR_NETWORK_NAME","YOUR_NETWORK_PASSWORD")
print(wifi.sta.getip())
led1 = 3
led2 = 4
gpio.mode(led1, gpio.OUTPUT)
gpio.mode(led2, gpio.OUTPUT)
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
   conn:on("receive", function(client,request)
      local buf = "";
      local _, _, method, path, vars = string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
      if(method == nil)then
         _, _, method, path = string.find(request, "([A-Z]+) (.+) HTTP");
      end
      local _GET = {}
      if (vars ~= nil)then
         for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
            _GET[k] = v
         end
      end
      buf = buf.."<h1> ESP8266 Web Server</h1>";
      buf = buf.."<p>GPIO0 <a href=\"?pin=ON1\"><button>ON</button></a> <a
href=\"?pin=OFF1\"><button>OFF</button></a></p>";
      buf = buf.."<p>GPIO2 <a href=\"?pin=ON2\"><button>ON</button></a> <a
href=\"?pin=OFF2\"><button>OFF</button></a></p>";
      local _on,_off = "",""
      if(_GET.pin == "ON1")then
           gpio.write(led1, gpio.HIGH);
      elseif(_GET.pin == "OFF1")then
           gpio.write(led1, gpio.LOW);
      elseif(_GET.pin == "ON2")then
           gpio.write(led2, gpio.HIGH);
      elseif(_GET.pin == "OFF2")then
           gpio.write(led2, gpio.LOW);
      end
      client:send(buf);
      client:close();
      collectgarbage();
   end)
end)
```



Dans l'exemple une connexion WiFi est crée grâce à Connectify avec comme SSID : Connectify-sb et Password : 0123456789

Le tableau ci dessous indique le lien entre les broches du CI ESP8266 et les broches du module ESP01 et ESP03 :

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		

3.4. Controler une LED (GP02) et tester un interrupteur en entrée (GP00)

(tester Ok)

```
wifi.setmode(wifi.STATIONAP)
--wifi.sta.config("Connectify-sb","0123456789")
print(wifi.ap.getip())
bp1 = 3 ;--GP00
led2 = 4 ;--GP02
gpio.mode(bp1, gpio.INPUT);
gpio.mode(led2, gpio.OUTPUT);
compteur=1;

srv=net.createServer(net.TCP) --ouvre une connexion TCP


srv:listen(80,function(conn)
   conn:on("receive", function(client,request)
      local buf = "";
      local _, _, method, path, vars = string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
      if(method == nil)then
         _, _, method, path = string.find(request, "([A-Z]+) (.+) HTTP");
      end
      local _GET = {}
      if (vars ~= nil)then
         for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
            _GET[k] = v
         end
      end
      --test du bp1
      if gpio.read(bp1)==1 then compteur = 1 ; --detecte un appui sur bp1 et compte
      if gpio.read(bp1)==0 then compteur = 0 ;
      end --if

      buf = buf.."<h1> ESP8266 Web Server</h1>";
      buf = buf.."<p>GPIO0 est = ";
      buf = buf.. compteur;
      buf = buf.. " </p>";
       buf = buf.."<p>GPIO2 <a href=\"?pin=ON2\"><button>ON</button></a> <a
href=\"?pin=OFF2\"><button>OFF</button></a></p>";
      local _on,_off = "",""

      if(_GET.pin == "ON2") then
           gpio.write(led2, gpio.HIGH);
                        elseif(_GET.pin == "OFF2")then
                              gpio.write(led2, gpio.LOW);
      end
```

```
      client:send(buf);
      client:close();
      collectgarbage();
   end)
end)
```

Le module est utilisé en mode AP (access point) afin de tester le progamme directement avec un appareil wifi.
On place un inter (fil) entre GP00 et la masse ou non
On place une LED+résistance (220 ohm) entre GP02 et la masse
Une page web indique l'état de GP00 et un bouton permet d'allumer ou d'éteindre la LED.

3.5. Contrôler un CAN I2C max11609EEE (A tester)

Fichier driver du MAX11609 :

```
-- Break out board product page: https://www.tindie.com/products/AllAboutEE/esp8266-analog-inputs-expander/
-- Description: Library for the MAX11609 10-bit analog to digital converter with 8 analog input channels
local moduleName = ...
local M = {}
_G[moduleName] = M

-- Default value for i2c communication as per NodeMCU API
local id = 0

-- Constant device address.
local MAX11609EEE_ADDRESS = 0x33

M.REF_EXTERNAL = 0x02
M.REF_INTERNAL = 0x05
M.REF_VDD = 0x00

function M.begin(pinSDA, pinSCL,vRef)
  i2c.setup(id,pinSDA,pinSCL,i2c.SLOW)
end

function M.setup(data)
  i2c.start(id)
  i2c.address(id,MAX11609EEE_ADDRESS,i2c.TRANSMITTER)
  i2c.write(id,bit.bor(data,0x80))
  i2c.stop(id)
end

function M.configuration(data)
  i2c.start(id)
  i2c.address(id,MAX11609EEE_ADDRESS,i2c.TRANSMITTER)
  i2c.write(id,bit.band(data,bit.bnot(0x80)))
  i2c.stop(id)
end

function M.read(channel)
 local result = 0x0000
 local configurationByte = bit.bor(bit.band((bit.lshift(channel,1)),0x0E),0x61) -- equivalent to ( (channel<<1) & 0b00001110) | 0b01100001

 M.configuration(configurationByte)
```

```
 i2c.start(id)

 if(i2c.address(id,MAX11609EEE_ADDRESS,i2c.RECEIVER) ~= false) then
   local data = i2c.read(id,2)
   i2c.stop(id)
   local msb, lsb = string.byte(data,1,2)
   msb = bit.lshift(bit.band(msb,0x03),8)
   lsb = bit.band(lsb,0xff)
   result = bit.bor(msb,lsb)

   return result
 else

   i2c.stop(id)
   return -1
 end

end

return M
```

Fichier exemple d'utilisation du MAX11609EEE

```
--
-- Break out board product page: https://www.tindie.com/products/AllAboutEE/esp8266-analog-inputs-expander/
--
-- @description An example that shows how you can read from all the pins (channel A0 to A7)
-- every 1000 milliseconds (1 second). Assumes you power the board from 3.3V Note: The max voltage is 3.6V so DO NOT use 5V
--

require("max11609eee")

gpio0, gpio2 = 3, 4 -- NodeMCU maps pins GPIO0 and GPIO2 to indices 3 and 4 respectively

pinSDA = gpio0 -- We'll use GPIO0 for I2C's SDA
pinSCL = gpio2 -- We'll use GPIO2 for I2C's SCL

max11609eee.begin(pinSDA,pinSCL,max11609eee.REF_VDD)  -- assign pins connect VDD to 3.3v

-- Continously execute the function code every 1000 milli seconds (1 second)
tmr.alarm(0,1000,1,function()
```

```
  local digitalValue = max11609eee.read(3) -- read from pin A3 (analog input 3)
  local voltageValue = digitalValue * 3.3 / 1024
  print("Digital Value: "..digitalValue)
  print("Voltage Value: "..voltageValue)
end)
```

4. RÉFÉRENCES :

Références aide sur ESP8266 avec Lua:

http://benlo.com/esp8266/esp8266QuickStart.html

Référence pour Lua avec NodeMcu : Toutes les fonctions utilisables.
https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodemcu-api-instruction

Nodemcu : fonction de référence et lien entre broche ESP§8266 et module :
https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en

Modules Lua pour différents circuits intégrés :
https://github.com/nodemcu/nodemcu-firmware/tree/master/lua_modules