



# IOT ESP32 AVEC IDE ARDUINO - PRISE EN MAIN

en cours...

1. Matériel utilisé.....	2
1.1 Hardware (Board).....	2
1.1.1 ESP32 IoT Starter kit with MongoosecOS :.....	2
1.1.2 ESP32 arducam.....	2
1.1.3 ESP32 Huzzah.....	2
1.2 Software.....	5
2. Test mode serveur en wifi (ok testé).....	5
2.1 Démarche.....	5
2.2 Code testé.....	6
2.3 Résultat obtenu.....	9
2.3.1 Test simple server sur Arducam (ok testé).....	10
3. Test multiUART (en cours de test).....	10
3.1 Protocole.....	11
3.2 Code.....	11
3.2.1 Test avec plusieurs UART Hard.....	11
3.2.2 Test avec plusieurs UART soft.....	12
3.3 Résultat obtenu.....	13
4. Test Bluetooth.....	14
4.1 Test de 'simple BLE'.....	14
4.2 Test de BLE uart.....	14
5. Test analogique.....	16
6. Ressources.....	17

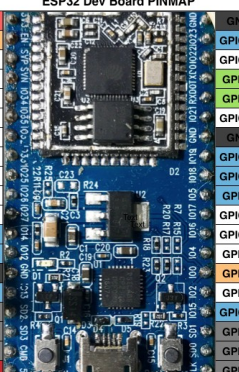
# 1. Matériel utilisé

## 1.1 Hardware (Board)

### 1.1.1 ESP32 IoT Starter kit with MongooseOS :

en cas d'utilisation d'une autre carte il faut modifier le board dans l'IDE.

Cablage du board :



Pin	Function	Pin	Function	Pin	Function
3.3V	EN	GPIO23	VSPI MOSI	GPIO23	SPI MOSI
RESET	GPIO36	GPIO22	VSPI MISO	GPIO22	Wire SCL
ADC0	GPIO39	GPIO1	TX0	GPIO1	Serial TX
ADC3	GPIO34	GPIO3	RX0	GPIO3	Serial RX
ADC6	GPIO35	GPIO21	VSPI SCK	GPIO21	Wire SDA
ADC7	GPIO32	GND	VSPI SS	GPIO19	SPI MISO
ADC4	GPIO33	GPIO18	VSPI SCK	GPIO18	SPI SCK
ADC5	GPIO25	GPIO5	VSPI SS	GPIO5	SPI SS (pu)
ADC18	GPIO26	GPIO17	VSPI SS	GPIO17	
ADC19	GPIO27	GPIO16	VSPI SS	GPIO16	
ADC17	GPIO14	GPIO4	VSPI SS	GPIO4	ADC10 TOUCH0 (pd)
HSPI SCK	GPIO12	GPIO0	BOOT	GPIO0	ADC11 TOUCH1 (pu)
HSPI MISO	GND	GPIO2	VSPI SS	GPIO2	ADC12 TOUCH2 (pd)
HSPI MOSI	GPIO13	GPIO15	VSPI SS	GPIO15	ADC13 TOUCH3 (pu)
HSPI SS	GPIO9	GPIO8	VSPI SS	GPIO8	FLASH D1
FLASH D2	GPIO10	GPIO7	VSPI SS	GPIO7	FLASH D0
FLASH D3	GPIO11	GPIO6	VSPI SS	GPIO6	FLASH SCK
FLASH CMD	SV				

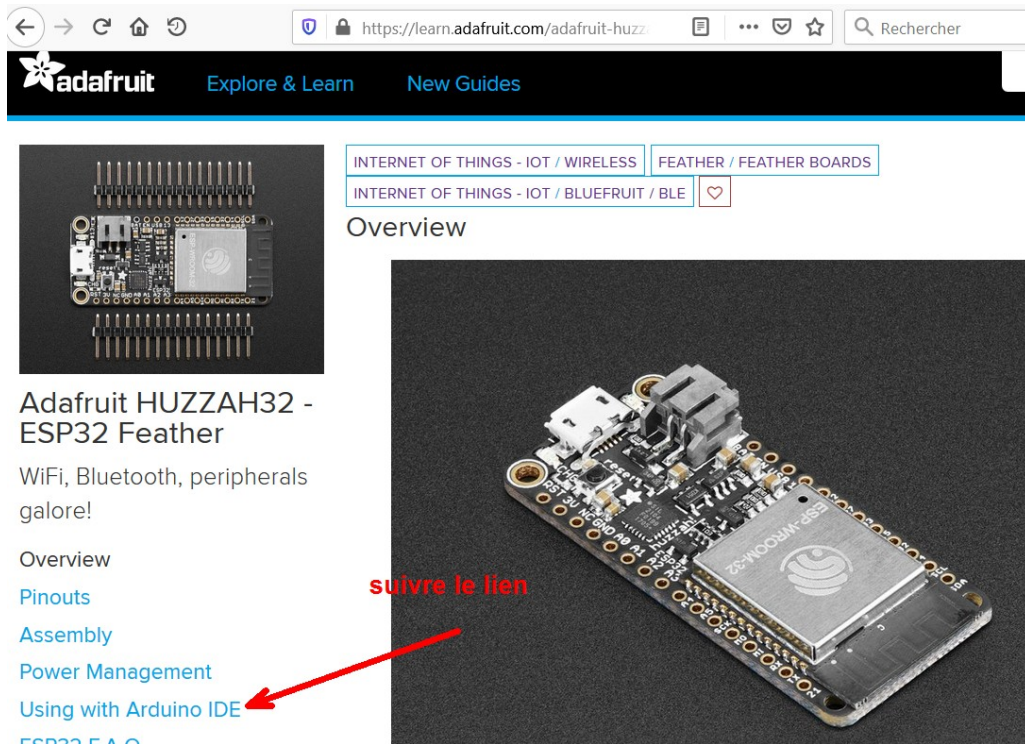
### 1.1.2 ESP32 arducam

Le même board fonctionne avec cette carte au format arduino uno3.

Les exemples fonctionnent aussi sur cette plateforme.

### 1.1.3 ESP32 Huzzah

Source : <https://www.adafruit.com/product/3405>



adafruit Explore & Learn New Guides

INTERNET OF THINGS - IOT / WIRELESS FEATHER / FEATHER BOARDS  
INTERNET OF THINGS - IOT / BLUEFRUIT / BLE

## Overview

Adafruit HUZAZH32 - ESP32 Feather

WiFi, Bluetooth, peripherals galore!

Overview  
Pinouts  
Assembly  
Power Management  
Using with Arduino IDE  
ESP32 FAQ

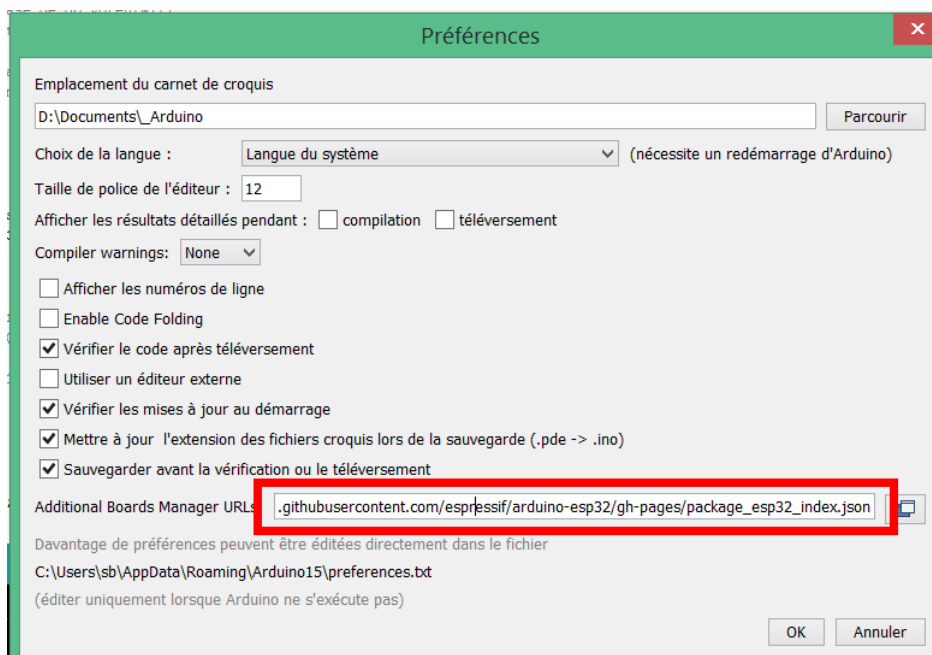
**suivre le lien**

Dans Arduino faire :

Fichier +preferences + ajouter la ligne :

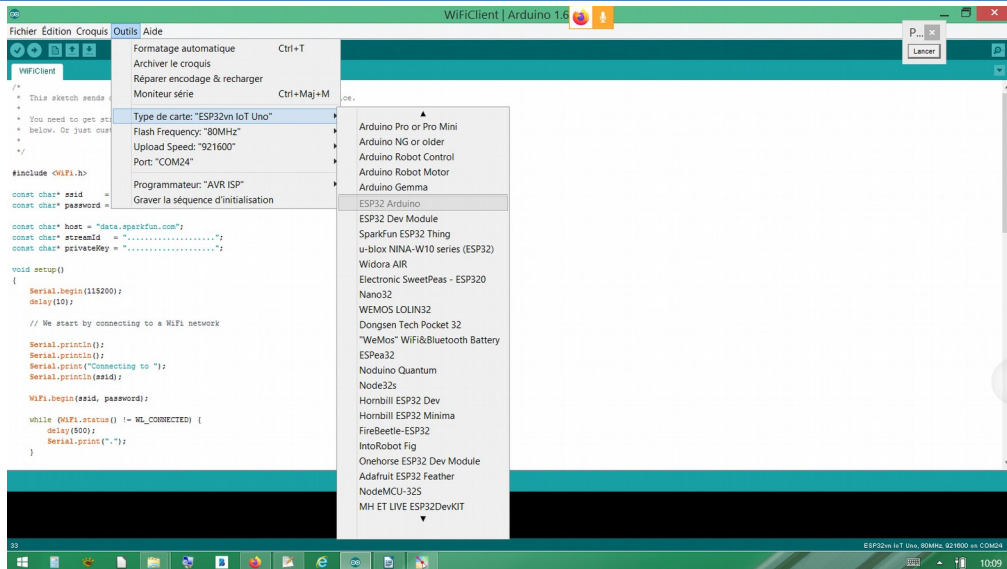
[https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json)

dans additional board :



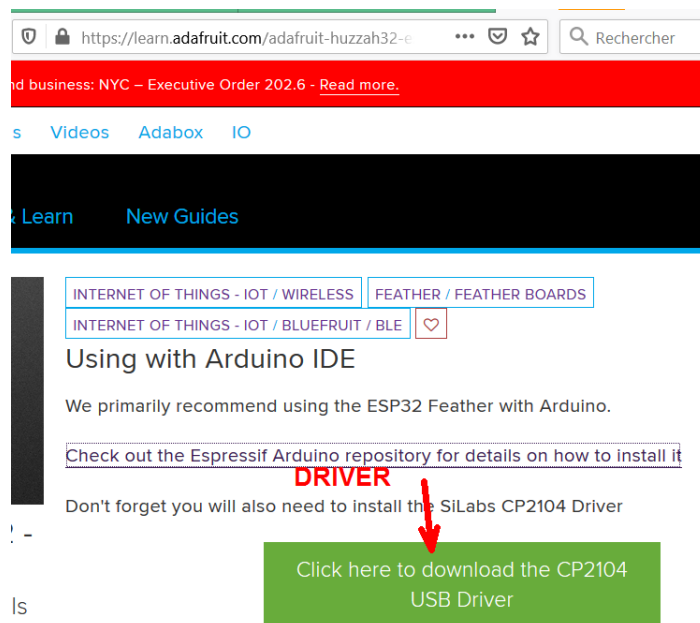
Une fois ceci effectué vous devez voir le board dans :

# IoT ESP32 avec IDE Arduino - prise en main

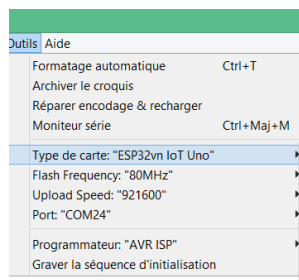


Choisir la carte ESP32vn IoT Uno (ou une autre adaptée)

ATTENTION : afin de configurer le port COM il faut installer le driver en suivant le lien :



Configurer le board :

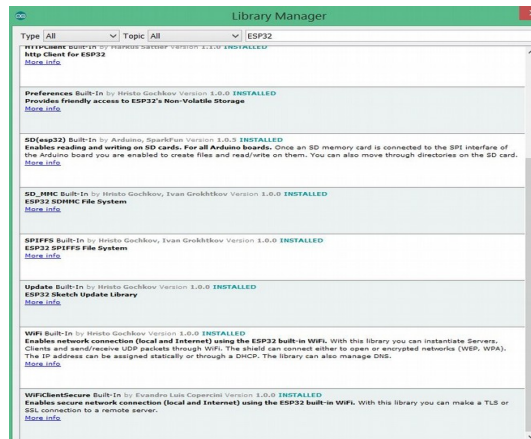
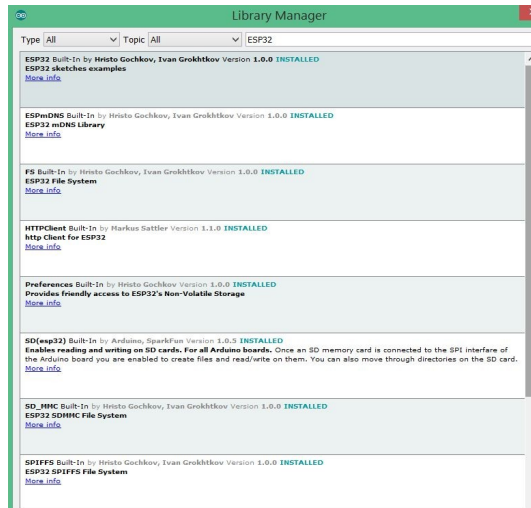


Des exemples sont apparus dans Fichier + exemples.

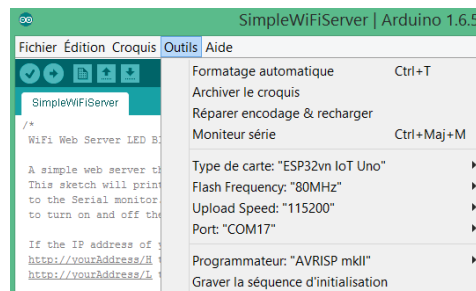
## 1.2 Software

IDE Arduino v1.6.5

Librairies installées :



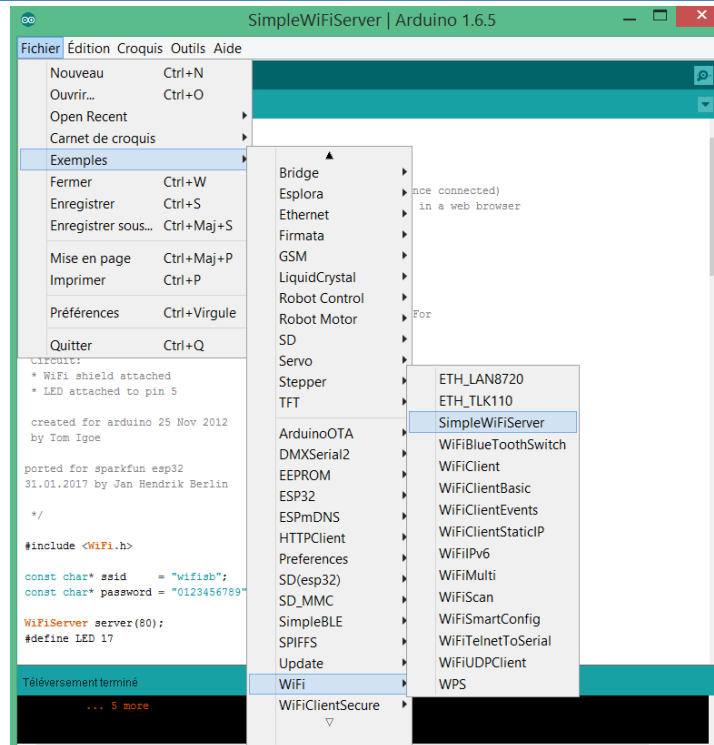
Configuration de la platine (board) :



## 2. Test mode serveur en wifi (ok testé)

### 2.1 Démarche

Créer un croquis à l'aide de l'exemple : Exemples + Wifi + SimpleServer



Créer un AP wifi avec votre PC grâce à la commande : netsh (cf [stssnsb.free.fr/loT](http://stssnsb.free.fr/loT))  
Compiler le code et téléverser le.

Le terminal permet de visualiser les commandes 'Serial.print' du code.

*ATTENTION : configuration de la liaison série : 115200 Bauds*

## 2.2 Code testé

```
/*  
WiFi Web Server LED Blink  
A simple web server that lets you blink an LED via the web.  
This sketch will print the IP address of your WiFi Shield (once connected)  
to the Serial monitor. From there, you can open that address in a web browser  
to turn on and off the LED on pin 17.  
  
If the IP address of your shield is yourAddress:  
http://yourAddress/H turns the LED on  
http://yourAddress/L turns it off  
  
This example is written for a network using WPA encryption. For  
WEP or WPA, change the Wifi.begin() call accordingly.  
Circuit:
```

```
* WiFi shield attached
* LED attached to pin 17
created for arduino 25 Nov 2012 by Tom Igoe
ported for sparkfun esp32 31.01.2017 by Jan Hendrik Berlin
modif SB 20180202
*/
#include <WiFi.h>

const char* ssid      = "wifisb";
const char* password = "0123456789";

WiFiServer server(80);
#define LED 17

void setup()
{
  Serial.begin(115200);
  pinMode(LED, OUTPUT);      // set the LED pin mode
  digitalWrite(LED, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(300);               // wait for a second
  digitalWrite(LED, LOW);   // turn the LED off by making the voltage LOW
  delay(300);               // wait for a second

  // We start by connecting to a WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    digitalWrite(LED, HIGH);  // turn the LED on (HIGH is the voltage level)
    delay(300);               // wait
    digitalWrite(LED, LOW);   // turn the LED off by making the voltage LOW
    delay(300);               // wait
  }
}
```

```
Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

server.begin();
}
int value = 0;

void loop(){
  WiFiClient client = server.available(); // listen for incoming clients

  if (client) { // if you get a client,
    Serial.println("New Client."); // print a message out the serial port
    String currentLine = ""; // make a String to hold incoming data
from the client
    while (client.connected()) { // loop while the client's connected
      if (client.available()) { // if there's bytes to read from the
client,
        char c = client.read(); // read a byte, then
        Serial.write(c); // print it out the serial monitor
        if (c == '\n') { // if the byte is a newline character

          // if the current line is blank, you got two newline characters in a
row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200
OK)
            // and a content-type so the client knows what's coming, then a blank
line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            // the content of the HTTP response follows the header:
            client.print("Click <a href=\"/H\">here</a> to turn the LED on pin 5
on.<br>");
            client.print("Click <a href=\"/L\">here</a> to turn the LED on pin 5
off.<br>");
```



```
        // The HTTP response ends with another blank line:
        client.println();
        // break out of the while loop:
        break;
    } else {    // if you got a newline, then clear currentLine:
        currentLine = "";
    }
    } else if (c != '\r') {    // if you got anything else but a carriage
return character,
        currentLine += c;    // add it to the end of the currentLine
    }

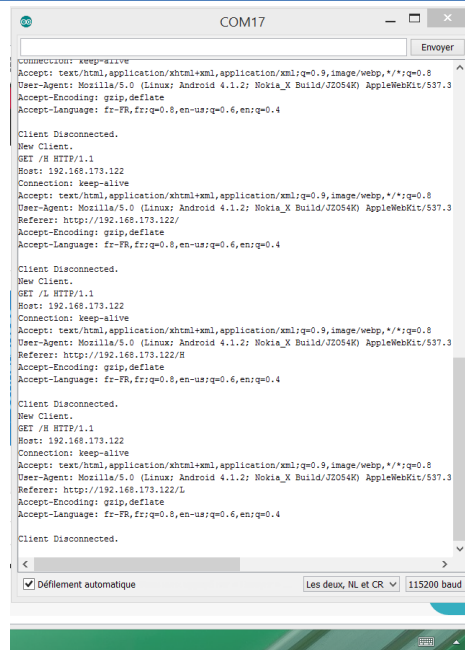
    // Check to see if the client request was "GET /H" or "GET /L":
    if (currentLine.endsWith("GET /H")) {
        digitalWrite(LED, HIGH);    // GET /H turns the LED on
    }
    if (currentLine.endsWith("GET /L")) {
        digitalWrite(LED, LOW);    // GET /L turns the LED off
    }
}
}
// close the connection:
client.stop();
Serial.println("Client Disconnected.");
}
}
```

### **2.3 Résultat obtenu**

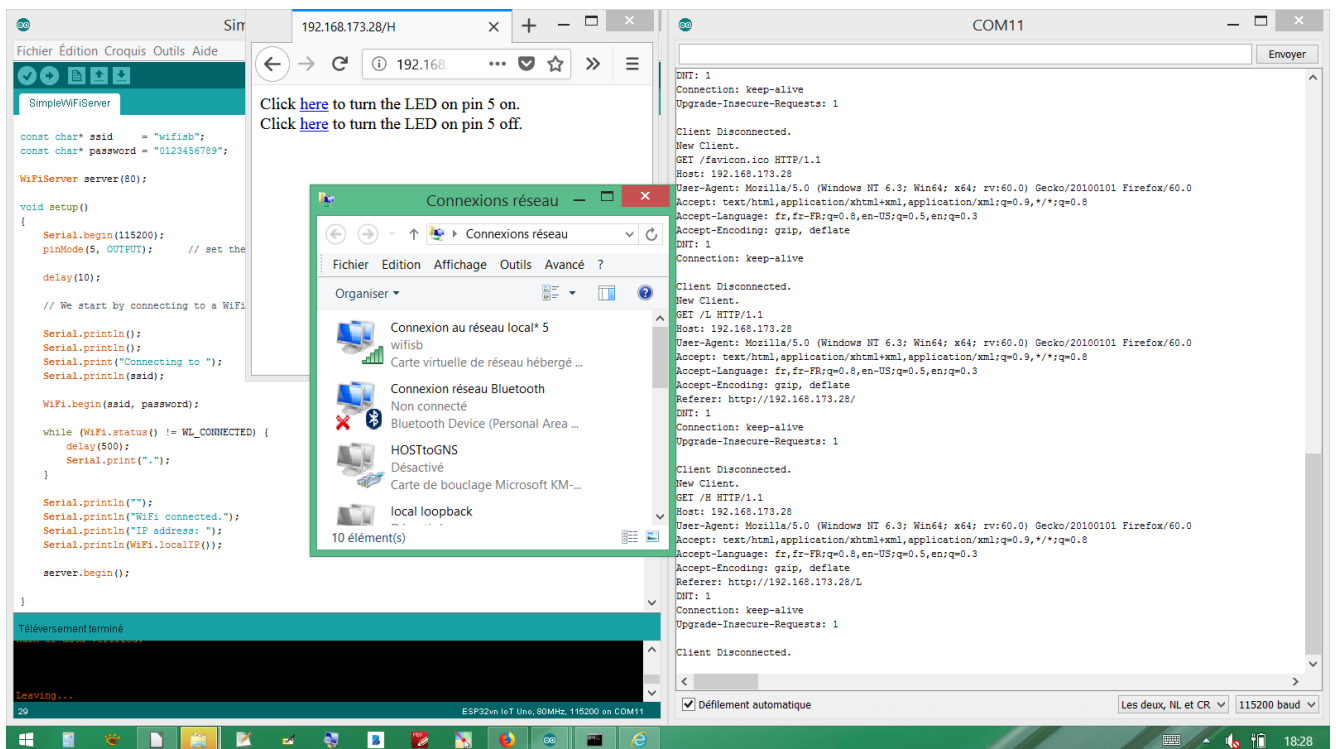
La connexion est parfois capricieuse et un RESET de la carte permet la connexion sur le wifi. La LED clignote tant que la connexion wifi n'est pas établie.

A l'aide d'un hôte wifi (PC ou smartphone) et d'un navigateur la commande de la LED fonctionne bien.

Le terminal indique les valeurs suivantes :



## 2.3.1 Test simple server sur Arducam (ok testé)



## 3. Test multiUART (en cours de test)

Apparemment seule une liaison est implémentée par l'IDE Arduino.

Objectif : utiliser les 3 liaisons de l'ESP32.

EXTRAIT doc. Technique ESP32 :

### 13. UART Controllers

13.1 Overview Embedded applications often require a simple method of exchanging data between devices that need minimal system resources. The Universal Asynchronous Receiver/Transmitter (UART) is one such standard that can realize a flexible full-duplex data exchange among different devices. The three UART controllers available on a chip are compatible with UART-enabled devices from various manufacturers. The UART can also carry out an IrDA (Infrared Data Exchange), or function as an RS-485 modem. All UART controllers integrated in the ESP32 feature an identical set of registers for ease of programming and flexibility. In this documentation, these controllers are referred to as UARTn, where n = 0, 1, and 2, referring to UART0, UART1, and UART2, respectively.

13.2 UART Features The UART modules have the following main features:

- Programmable baud rate
- 1024 x 8-bit RAM shared by three UART transmit-FIFOs and receive-FIFOs
- Supports input baud rate self-check
- Supports 5/6/7/8 bits of data length
- Supports 1/1.5/2/3/4 STOP bits
- Supports parity bit
- Supports RS485 Protocol
- Supports IrDA Protocol
- Supports DMA to communicate data in high speed
- Supports UART wake-up
- Supports both software and hardware flow control

## 3.1 Protocole

On utilise un cordon USB/TTL afin de visualiser les sorties RX/TX de l'ESP32.

## 3.2 Code

### 3.2.1 Test avec plusieurs UART Hard

Le code suivant donne une erreur : no serial1 declared in this scope

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial1.begin(115200);
  Serial2.begin(115200);

  pinMode(LED, OUTPUT);      // set the LED pin mode

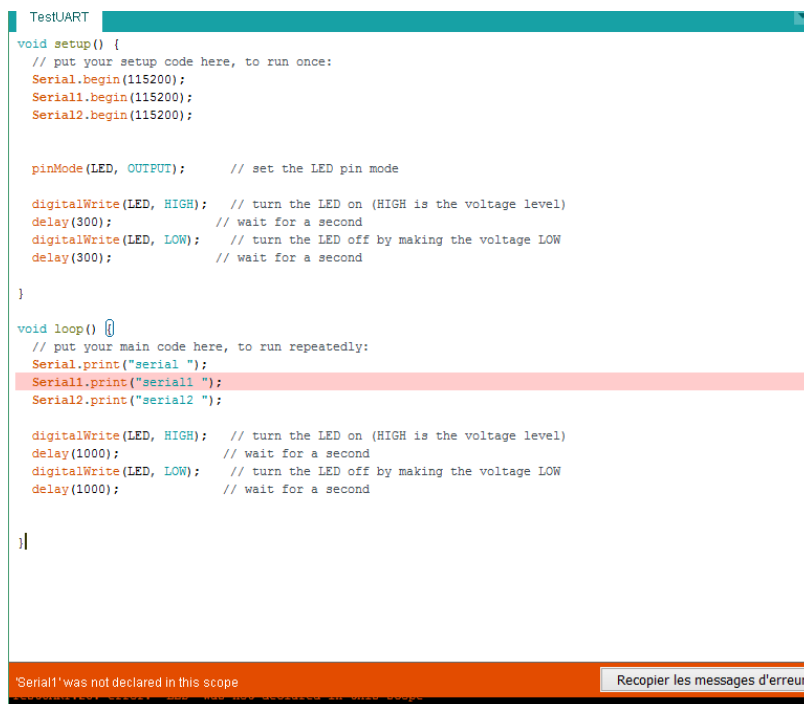
  digitalWrite(LED, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(300);                // wait for a second
  digitalWrite(LED, LOW);    // turn the LED off by making the voltage LOW
  delay(300);                // wait for a second
}
void loop() {
```

## IoT ESP32 avec IDE Arduino - prise en main

```
// put your main code here, to run repeatedly:
Serial.print("serial ");
Serial1.print("serial1 ");
Serial2.print("serial2 ");

digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000);             // wait for a second
digitalWrite(LED, LOW);  // turn the LED off by making the voltage LOW
delay(1000);             // wait for a second

}
```



```
TestUART
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial1.begin(115200);
  Serial2.begin(115200);

  pinMode(LED, OUTPUT); // set the LED pin mode

  digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(300); // wait for a second
  digitalWrite(LED, LOW); // turn the LED off by making the voltage LOW
  delay(300); // wait for a second
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.print("serial ");
  Serial1.print("serial1 ");
  Serial2.print("serial2 ");

  digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
}

'Serial1' was not declared in this scope Recopier les messages d'erreur
```

Les 3 UART ne sont pas implémentées sur l'IDE Arduino

### 3.2.2 Test avec plusieurs UART soft

Test SoftwareSerial : manque bibliothèque

Test NeoSWSerial : incompatible avec ESP32

### ***3.3 Résultat obtenu***

## 4. Test Bluetooth

Installer la librairie ESP32 Ble

De nouveaux exemples permettent de tester la connexion Bluetooth.

### 4.1 Test de 'simple BLE'.

Charger l'exemple dans l'IDE

Compiler et téléverser

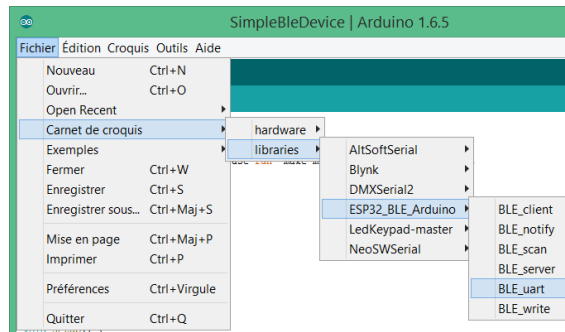
Ouvrir le terminal

Avec un smartphone activé le bluetooth et rechercher les nouveaux appareils.

ESP32 simple BLE apparaît (l'appairement n'est pas prévu dans le sketch)

Le changement sur D0 devrait changer le nom du périphérique BLE. (perso ça marche pas ! Par contre la modif du nom dans le code montre que ça tourne)

### 4.2 Test de BLE uart



Charger l'exemple dans l'IDE

Compiler et téléverser

Ouvrir le terminal

Avec un smartphone activé le bluetooth et rechercher les nouveaux appareils.

L'appareil apparaît sous le terme : 'UART service'

Lancer une application capable de faire une communication BLE (exemple : debug Lux-  
e (voir [stssnsb.free.fr](http://stssnsb.free.fr)))

Problème avec smartphone : connexion en erreur

Ok sur PC windows 8.1 : connexion ok mais pas de réception des textes.



## 5. Test analogique



## 6. Ressources

<https://www.youtube.com/watch?v=SBG7ccW5gpA> : vidéo de prise en main

<http://www.instructables.com/id/ESP32-BLE-Android-App-Arduino-IDE-AWESOME/> :  
bluetooth

