ESP8266-12

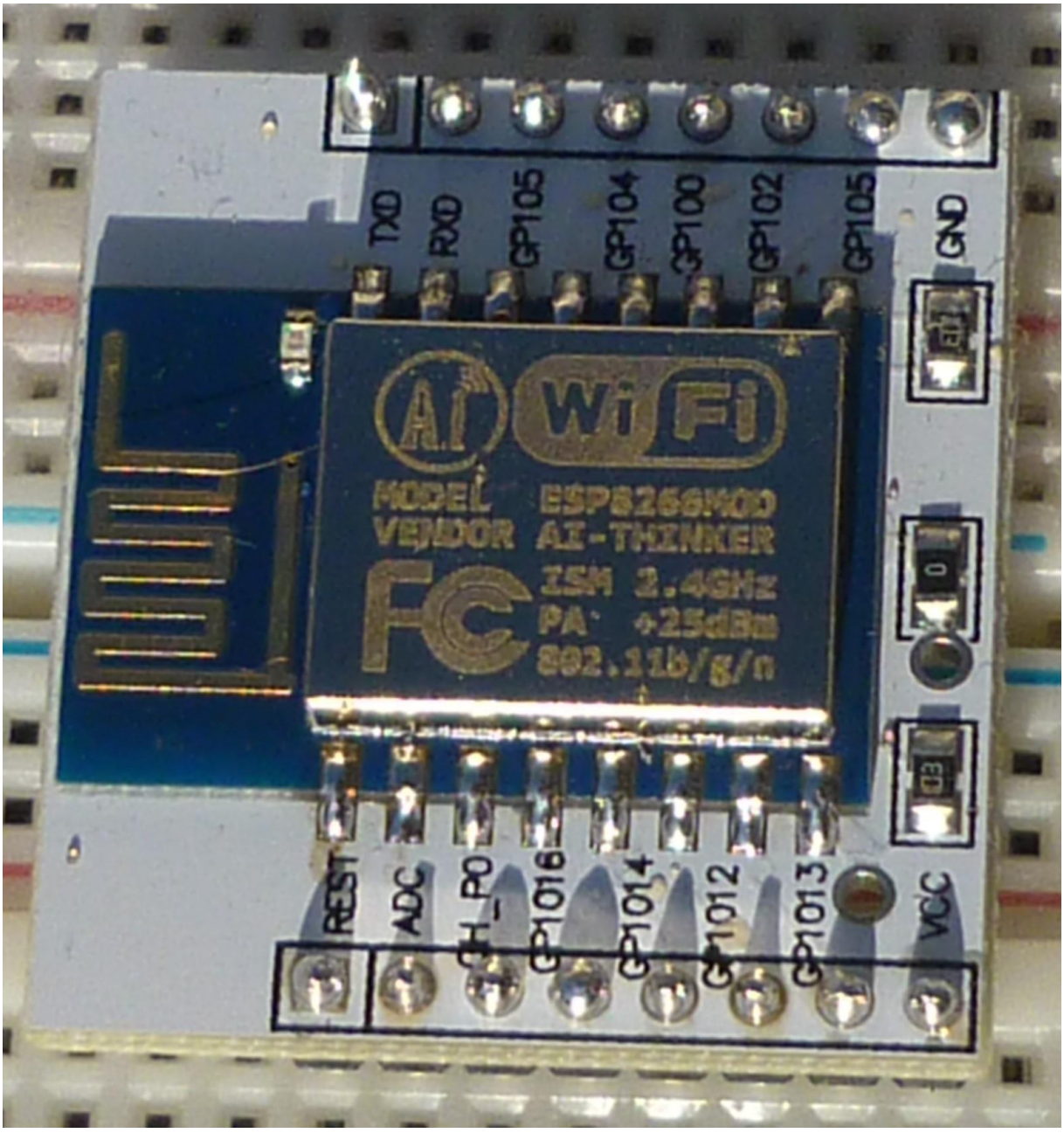

TXD
RXD
GPIO5
GPIO4
GPIO0
GPIO2
GPIO5
GND
Wi Fi
MODEL ESP8266MOD
VENDOR AI-THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
RST
ADC
CH_PD
GPIO16
GPIO14
GPIO12
GPIO13
VCC

Table des matières

1 Présentation

Tout d'abord, qu'est-ce qu'un ESP8266 ?

C'est un microcontrôleur comme il en existe des centaines, avec des ports d'entrées sorties, un convertisseur analogique digital, des ports série, I2C, SPI, comme la plupart des autres microcontrôleurs. Ce qui le différencie des autres, c'est qu'il a un émetteur récepteur Wi-Fi. Il est surtout conçu pour ce qu'on appelle l'internet des objets pour que n'importe quel objet courant puisse dialoguer sur Internet. Son seul défaut est sa consommation assez importante interdisant les objets autonomes sur pile ou batterie.

Pour un prix défiant toute concurrence (environ 2€), on peut avoir un petit serveur web qui pourra envoyer des informations mesurées sur ses ports d'entrée sortie en Wifi et donc à la planète entière via une Box.

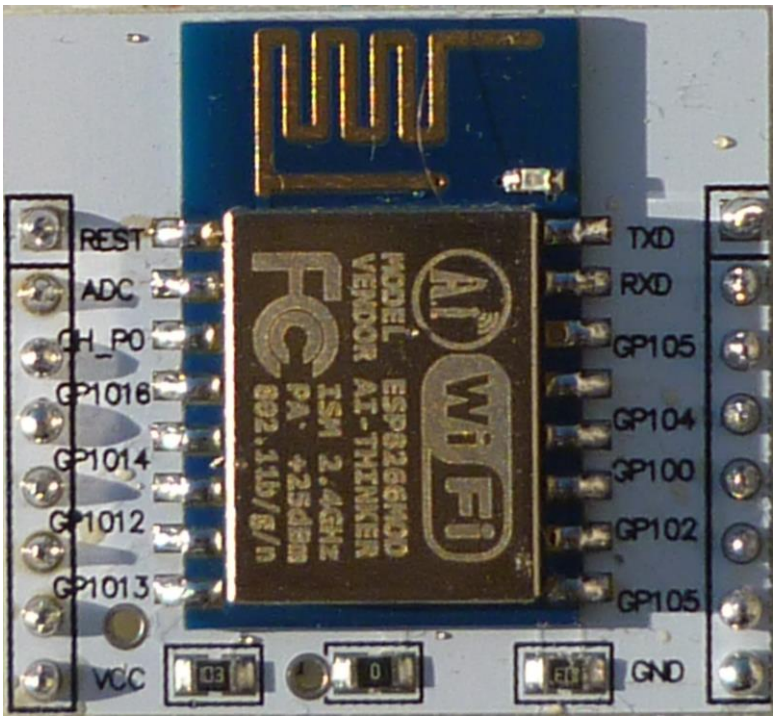
La version ESP-12 que je vais utiliser ici est le successeur d'une ligne de produit dont voici un petit historique https://fr.wikipedia.org/wiki/ESP8266

Pour une présentation technique, voici le datasheet https://mintbox.in/media/esp-12e.pdf. Ce datasheet décrit le microcontrôleur seul, le module que j'utilise ici contient en plus une mémoire Flash pour le stockage des programmes. Donc, Sur mon module, les broches du bas sont inaccessibles car elles sont utilisées pour la mémoire flash.

J'ai acheté ce module sur EBay pour 2€40 frais de port inclus avec un adaptateur permettant une utilisation plus facile.

Il a été livré bien emballé dans un petit sachet et le circuit est dans une mini boite en plastique.

Attention : le module n’est pas soudé, c’est à vous de le faire. Si possible avec un fer à souder à panne fine et bien isolée.



2 Développement

2.1 Présentation

Pour la programmation, il y a plusieurs possibilités, les commandes Hayes de base qui sont le langage natif du microcontrôleur, les scripts LUA, le JavaScript, le Python et le C. Personnellement, je vais utiliser le C grâce à l'IDE de l'Arduino qui peut gérer ce microcontrôleur (http://esp8266.github.io/Arduino/versions/2.0.0/doc/reference.html)

Afin d'envoyer les programmes sur l'ESP8266, il vous faudra par exemple un convertisseur USB/Série de type FTD1232.

Je l'ai, lui aussi, acheté sur EBay pour 1€67 frais de port inclus.

2.2 Installation de l'environnement de développement Arduino

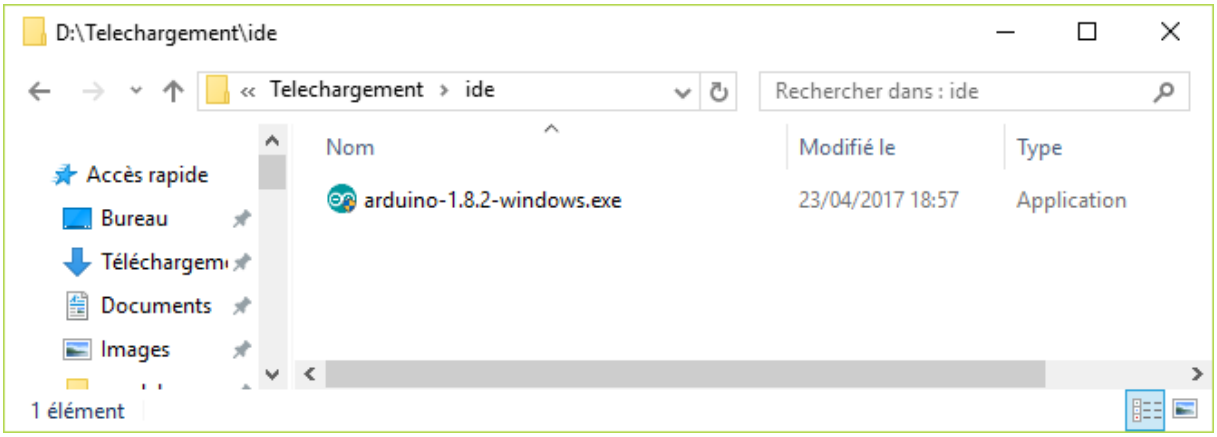
Comme indiqué plus haut, je vais programmer le module en C grâce à l'IDE de l'Arduino. Cet IDE permet l'ajout de module supplémentaire afin de programmer d'autres microcontrôleurs que l'Arduino.

L'IDE est à télécharger à l'adresse https://www.arduino.cc/en/main/software.

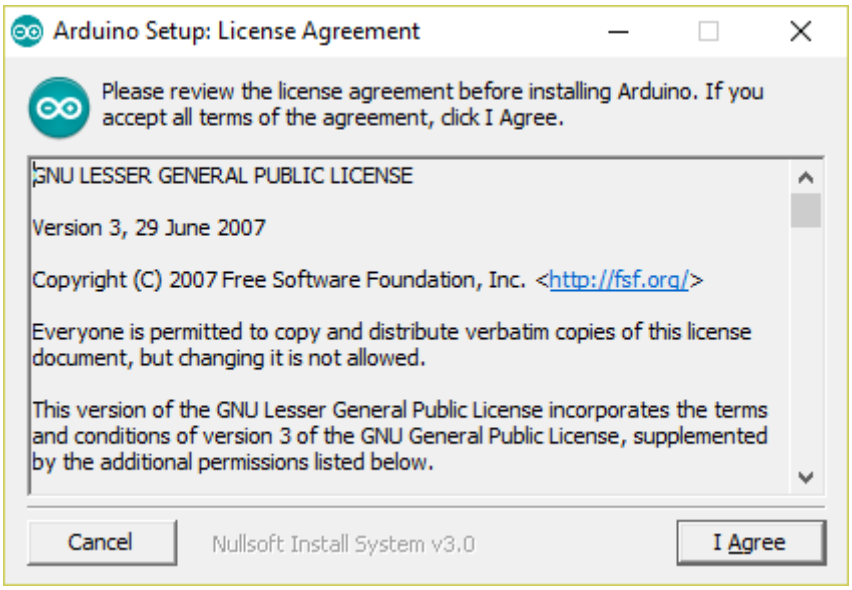
Il y a plusieurs versions pour différents OS.

Voici l'explication pour l'installation de la version Windows.

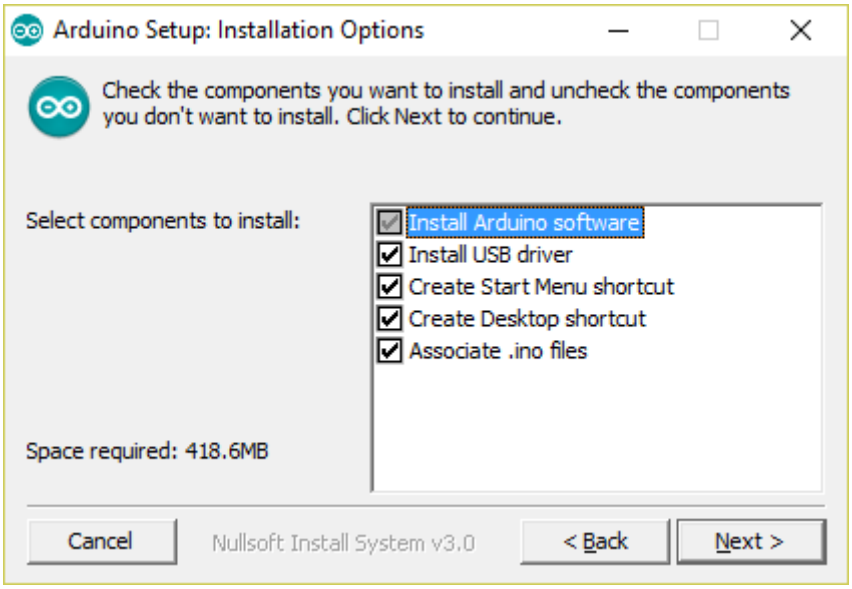
Double clic pour lancer l'exécutable



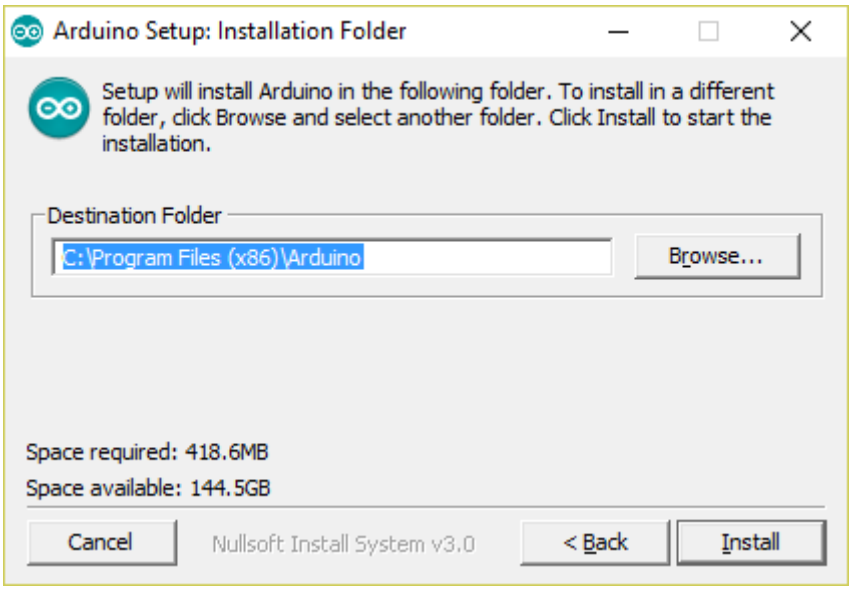
Clic sur le bouton I Agree

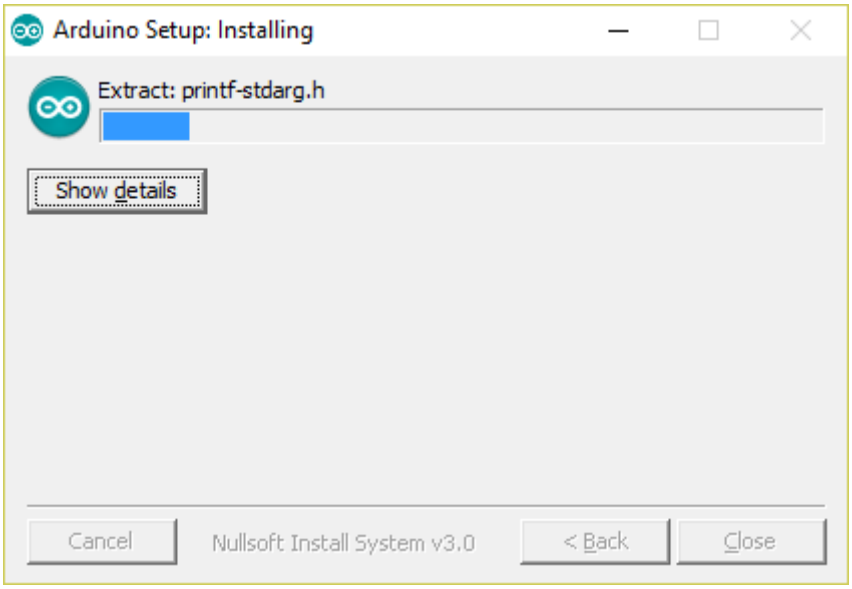


Clic sur le bouton Next

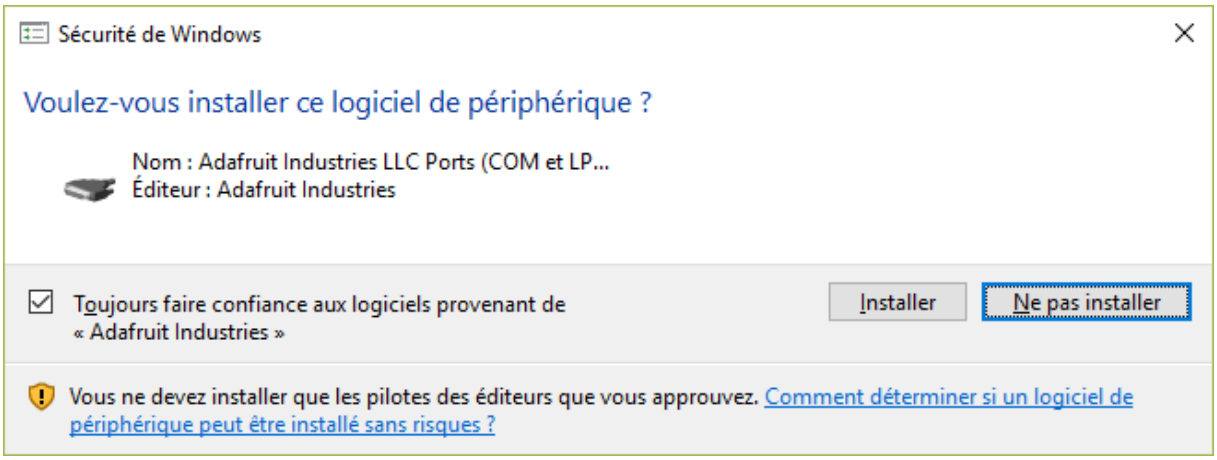


Clic sur le bouton Install

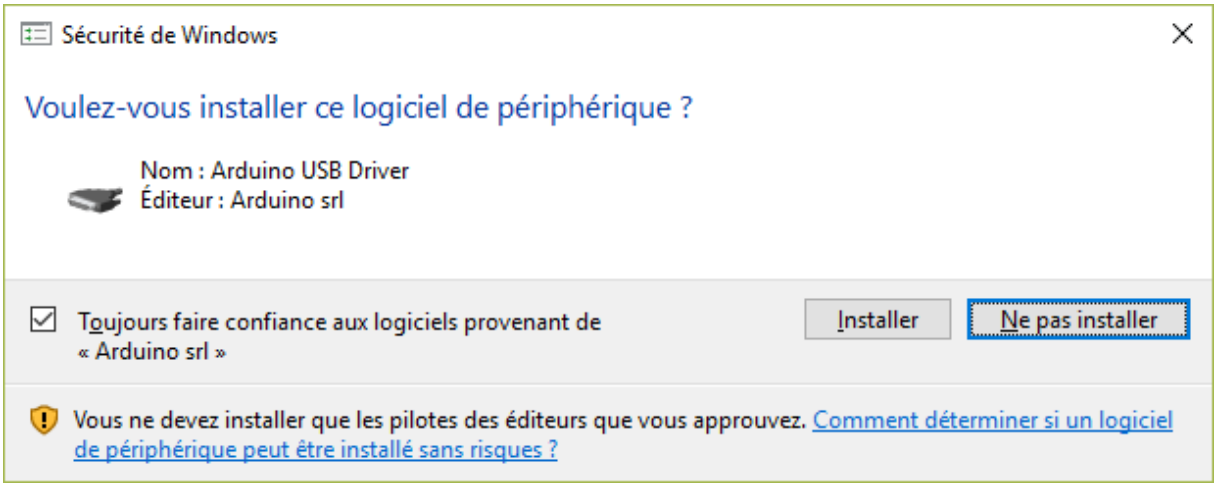




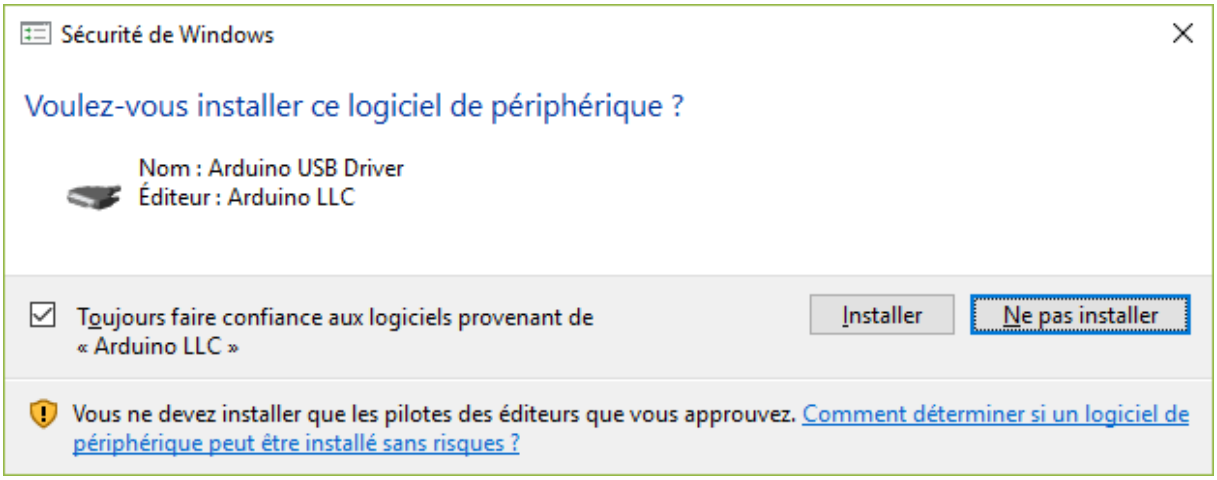
Clic sur le bouton Installer (Attention, c'est le bouton « Ne pas Installer » qui est par défaut, ne pas taper sur Entrée directement)



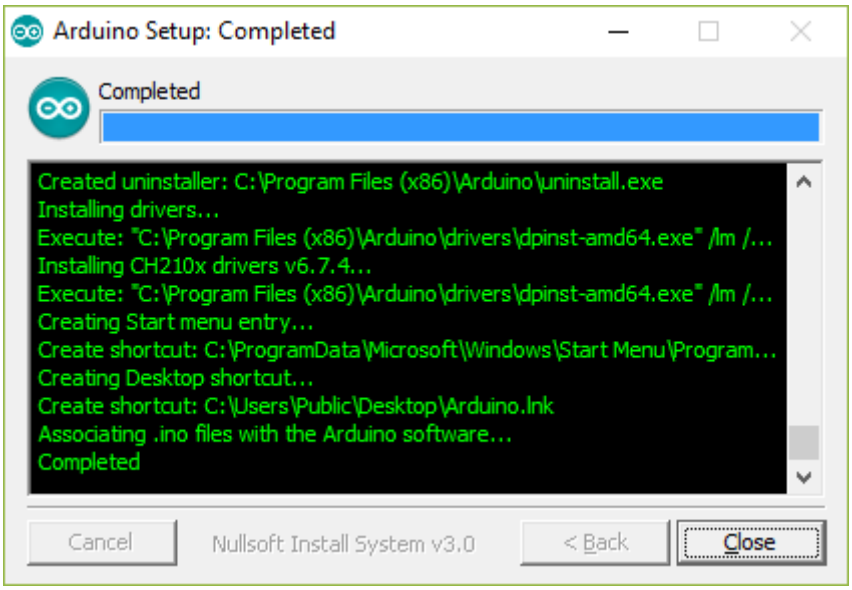
Clic sur le bouton Installer, là aussi ne pas taper sur Entrée directement



Clic sur le bouton Installer, là aussi ne pas taper sur Entrée directement

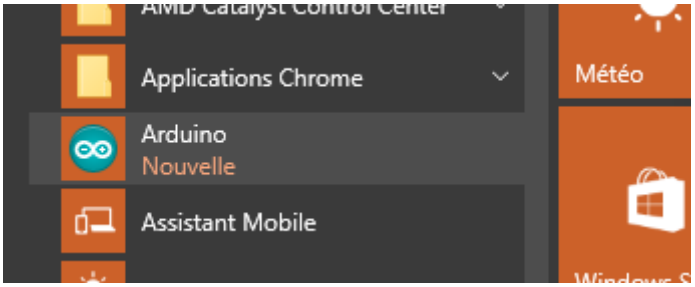


Clic sur le bouton Close quand le message « Completed » apparait

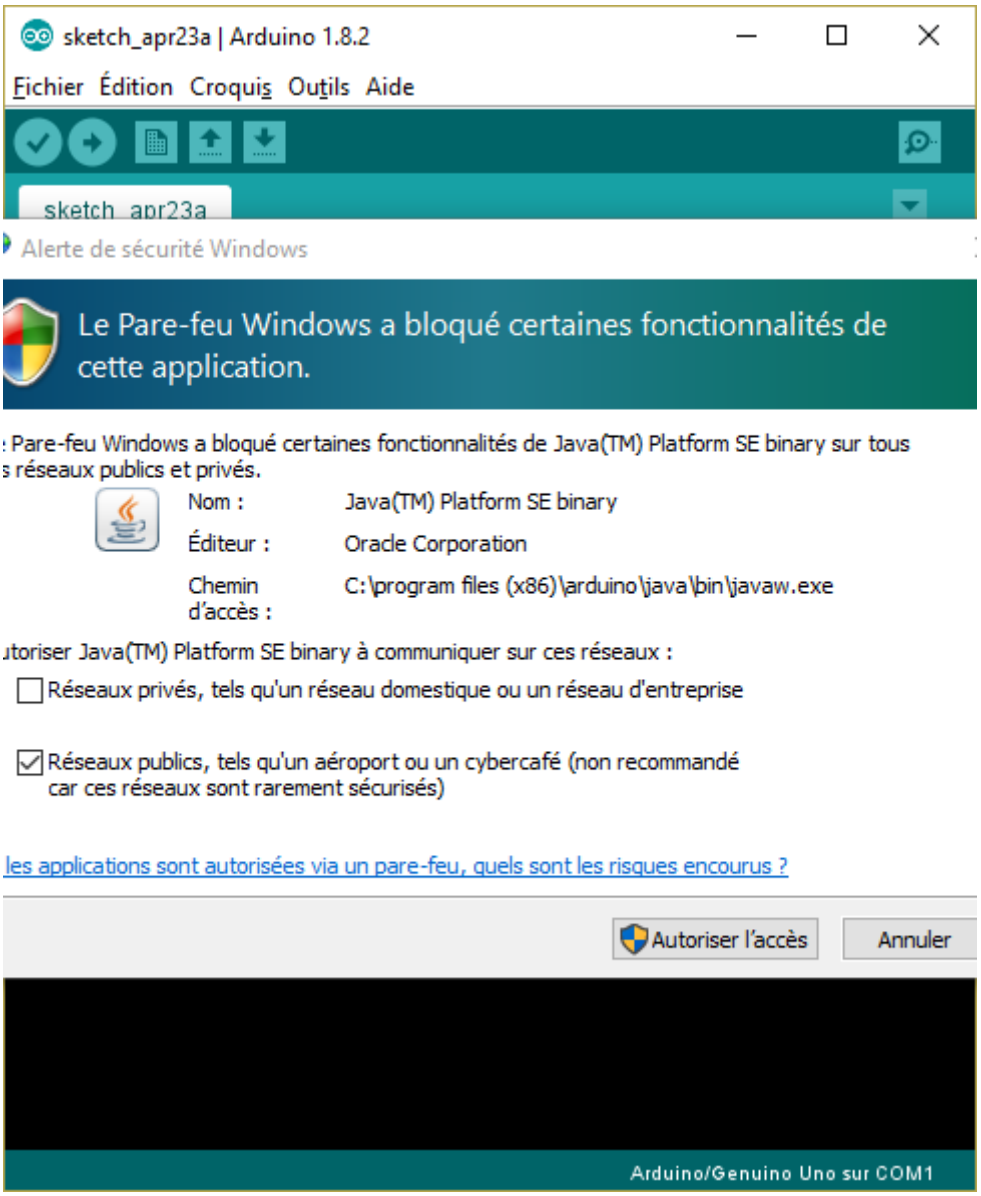


A cette étape, l'IDE est installé, mais il n'est pas encore configuré pour l'ESP8266, c'est ce qu'il va être fait maintenant.

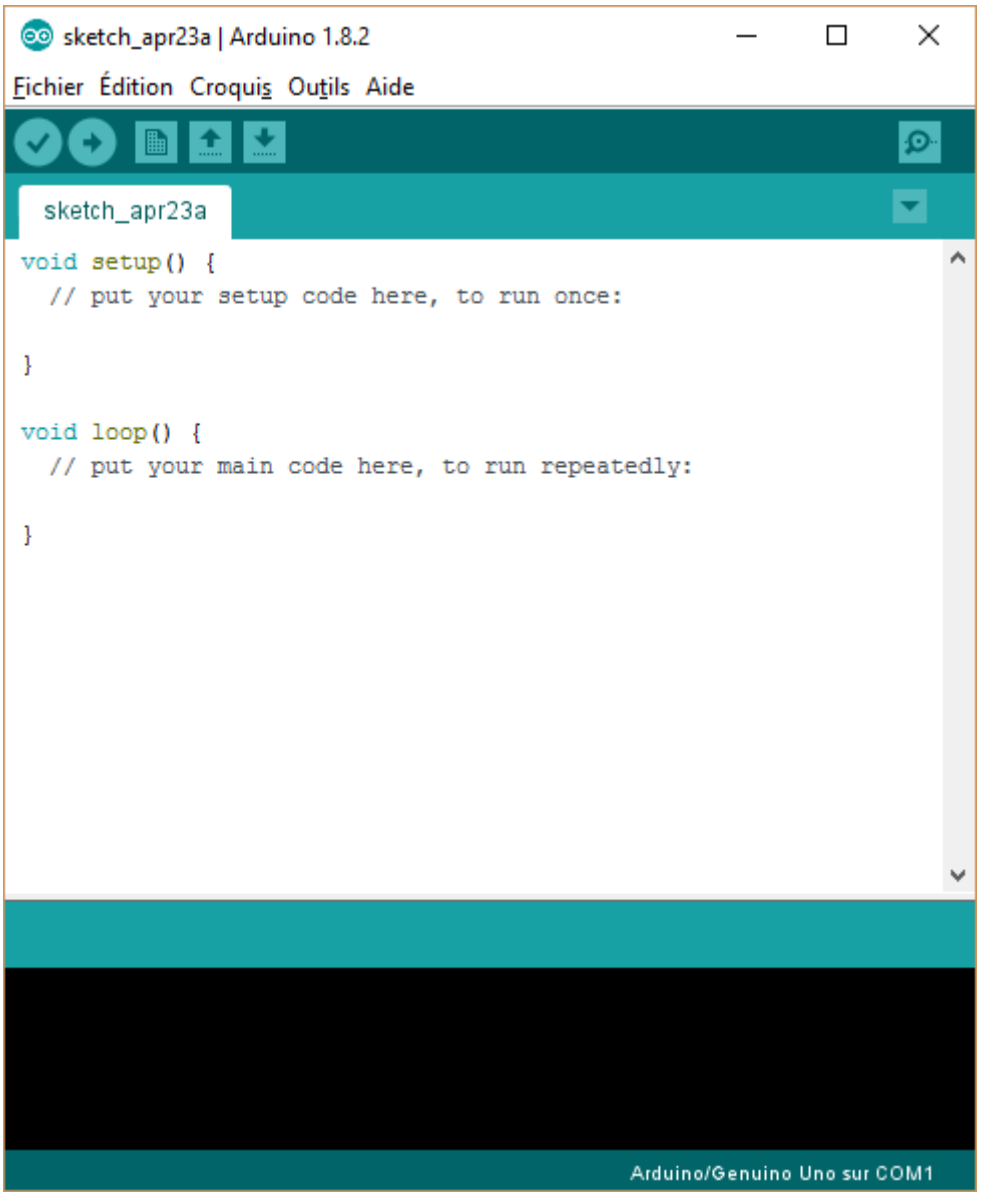
Pour lancer l'IDE, cliquez sur l'icône du logiciel dans le menu démarrer.



Au premier lancement du logiciel, le pare-feu de Windows peut vous demander d'autoriser l'accès

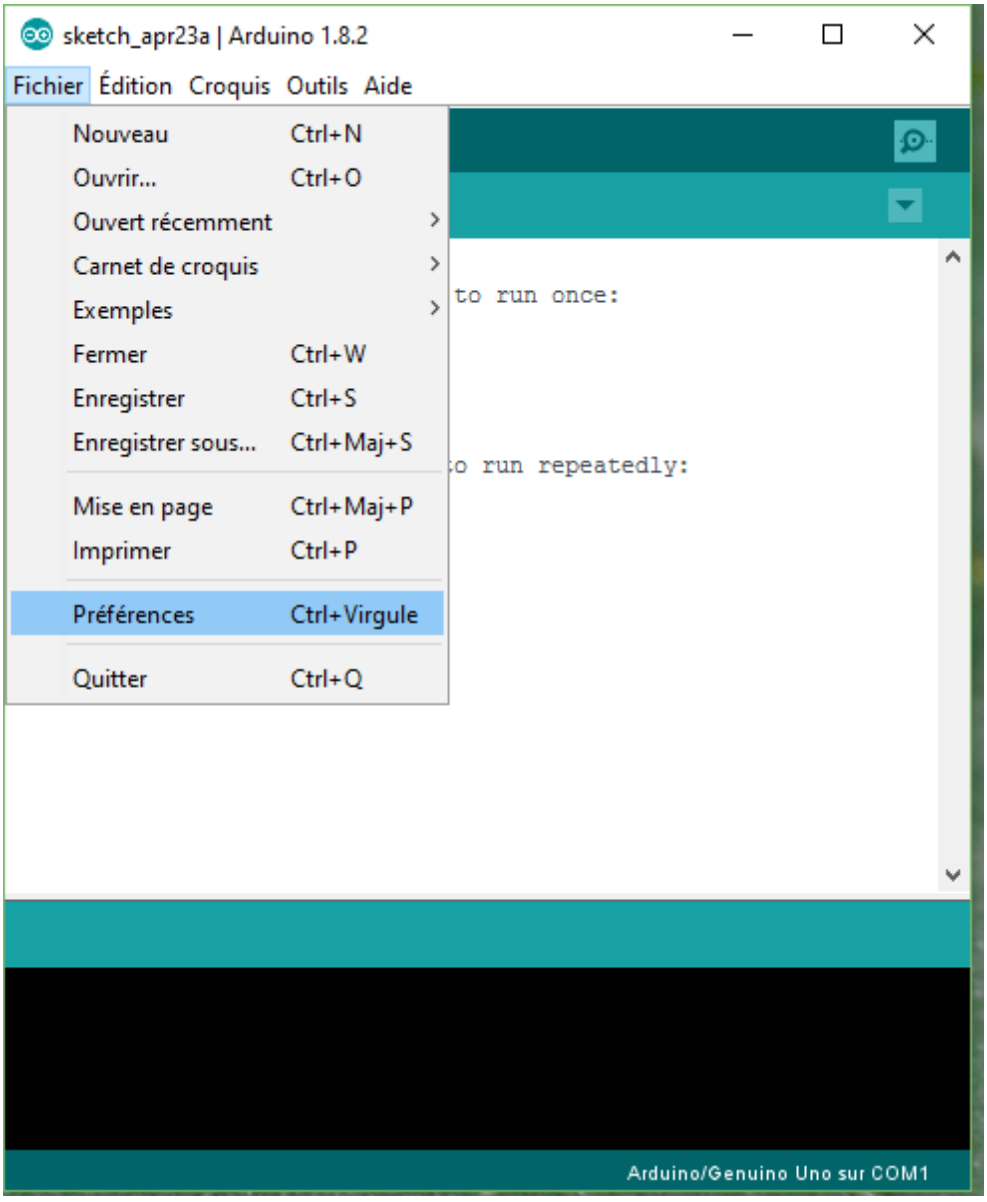


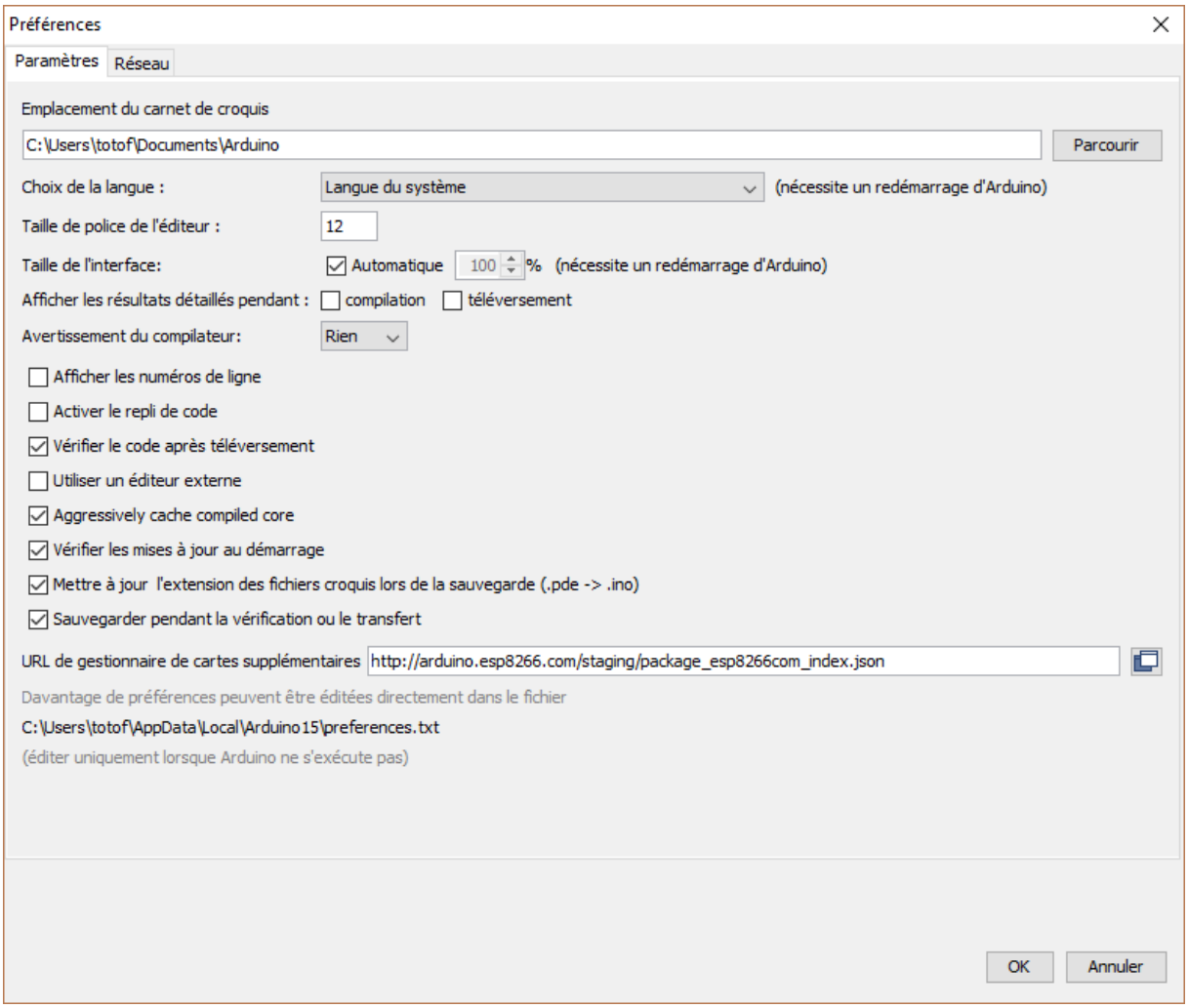
Puis, vous arrivez sur l'écran de base avec un squelette de programme pour un Arduino



L’IDE de base ne gérant que les Arduino, on va lui ajouter la gestion des microcontrôleurs ESP8266

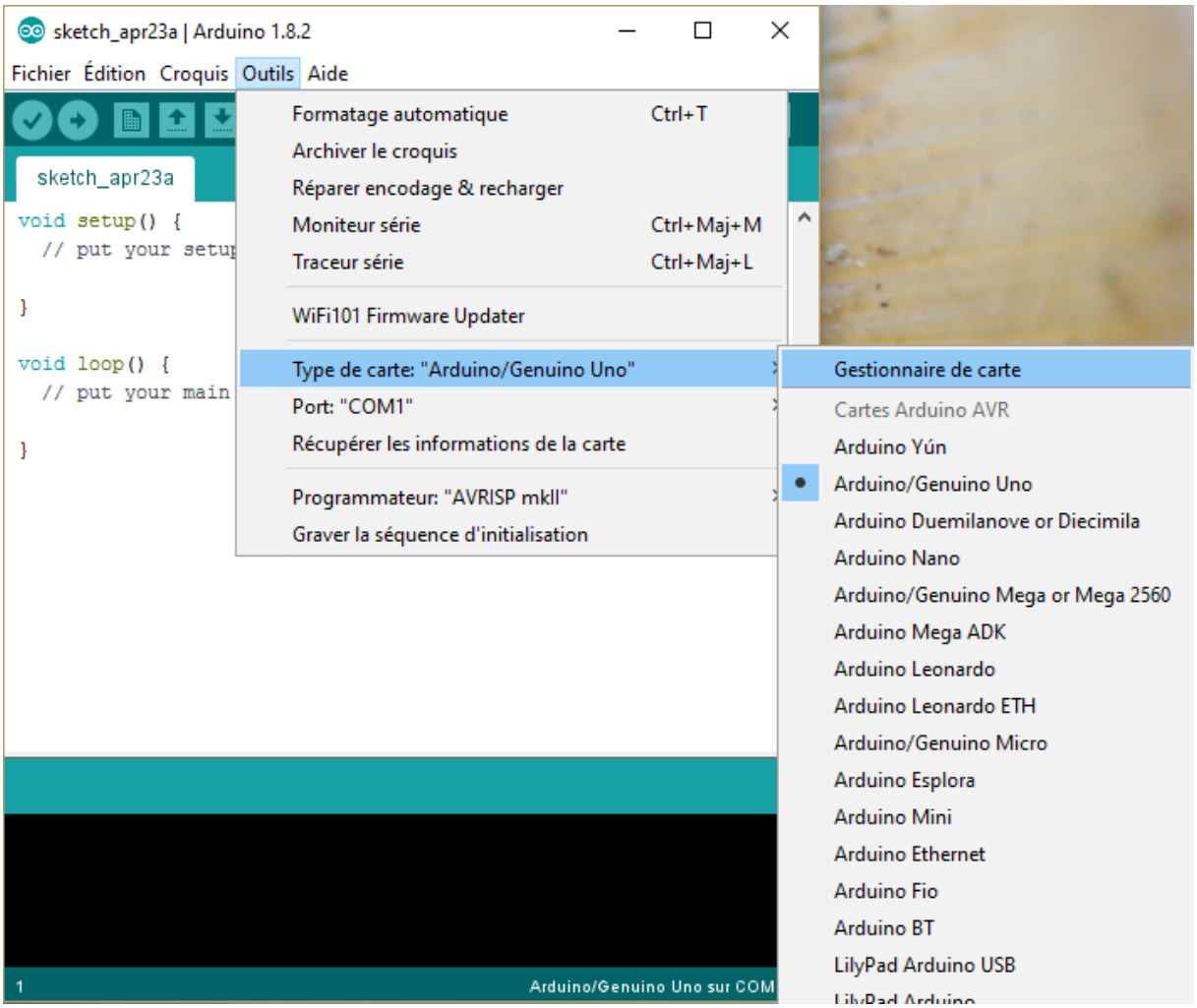
Ajoutez l’URL http://arduino.esp8266.com/staging/package_esp8266com_index.json dans la zone « URL de gestionnaire de cartes supplémentaires » du menu « Préférences ».



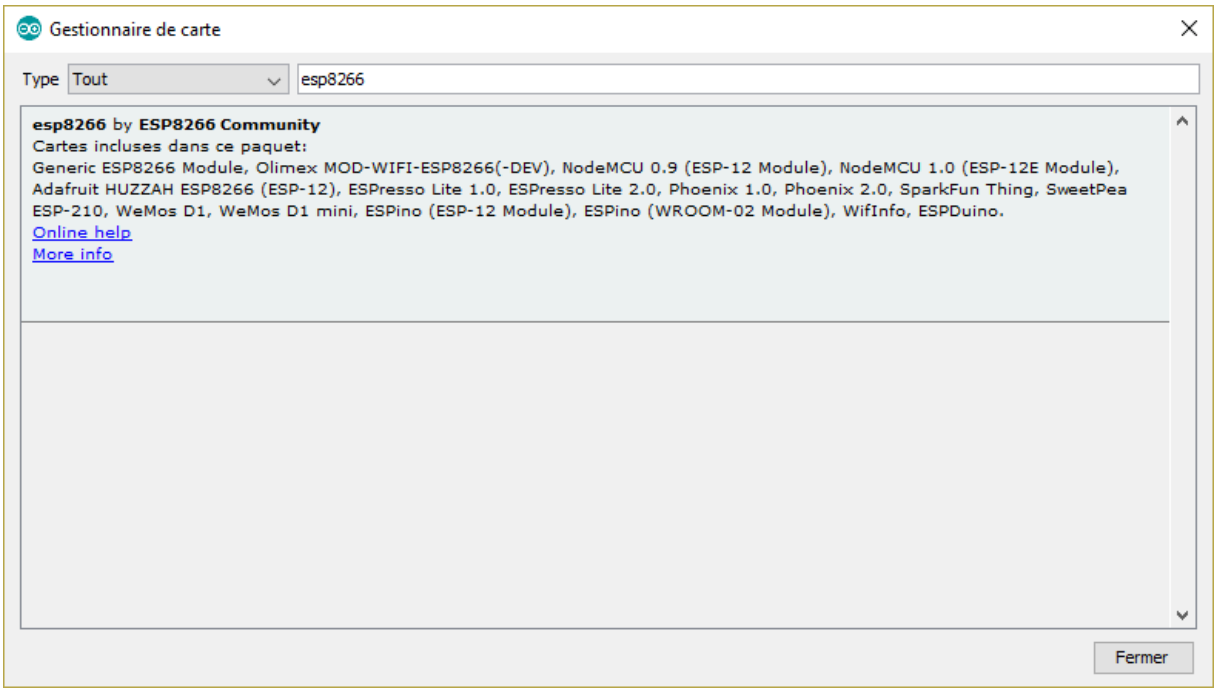


Cliquer sur « OK » pour valider

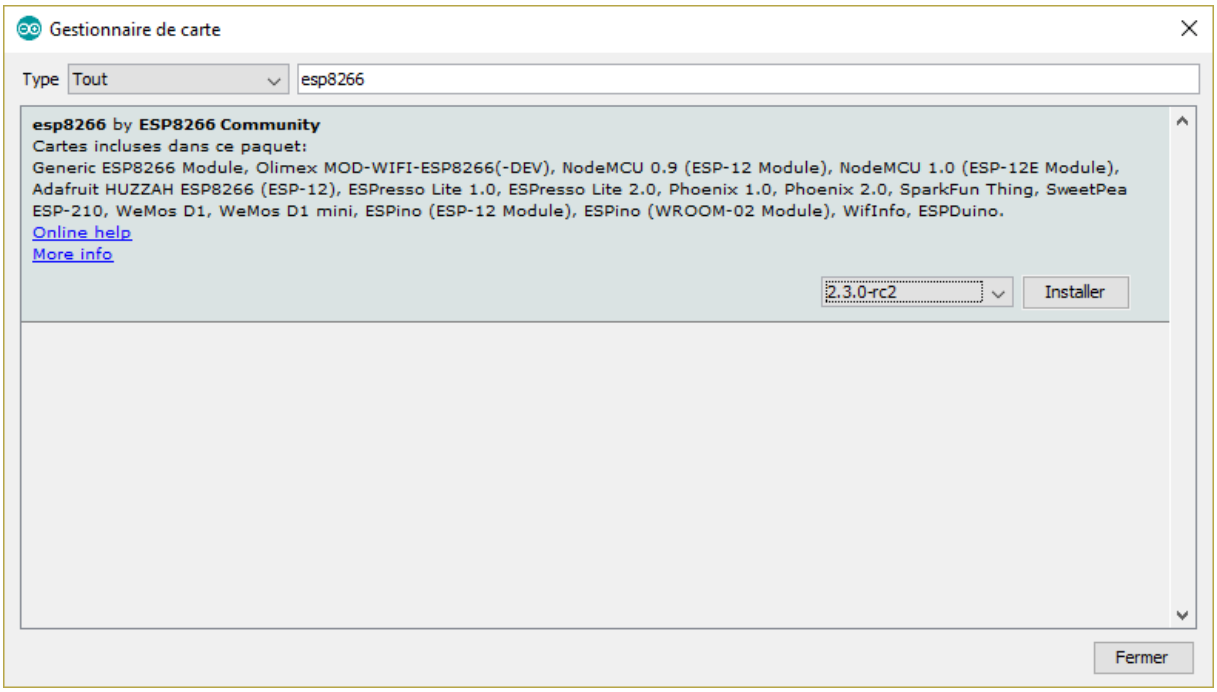
Puis, sélectionnez le menu Outils/Type de carte/Gestionnaire de carte

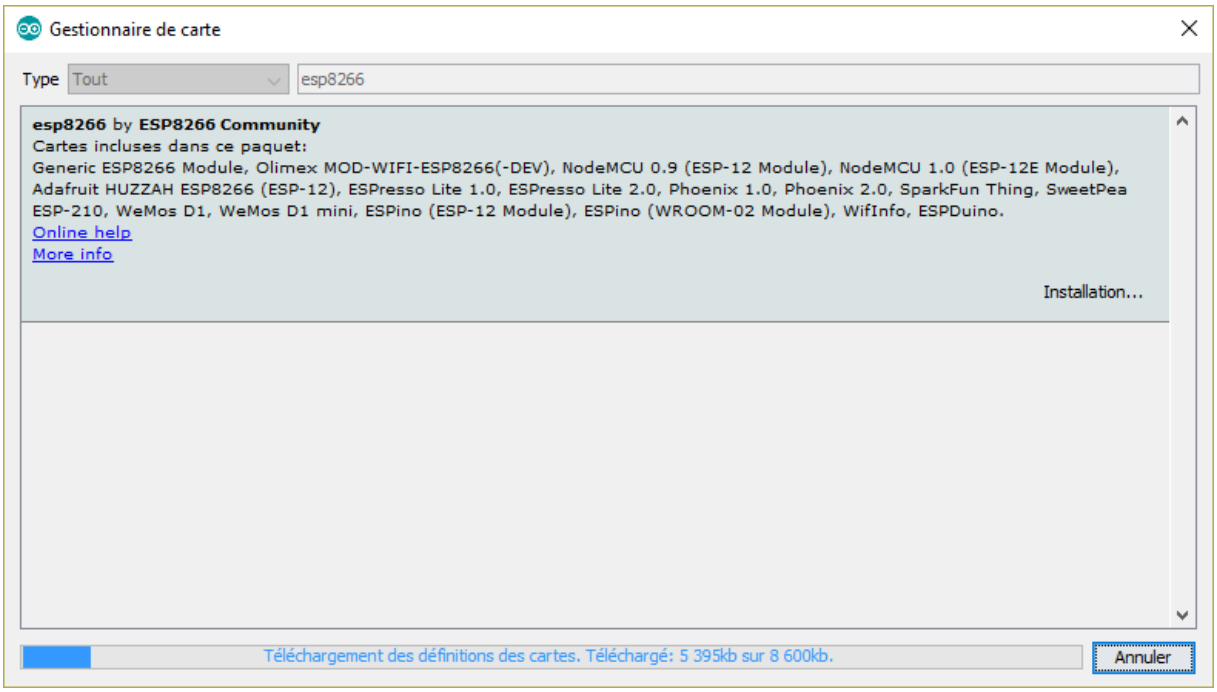


Dans le gestionnaire, demandé « esp8266 » et tapez sur Entrée pour valider

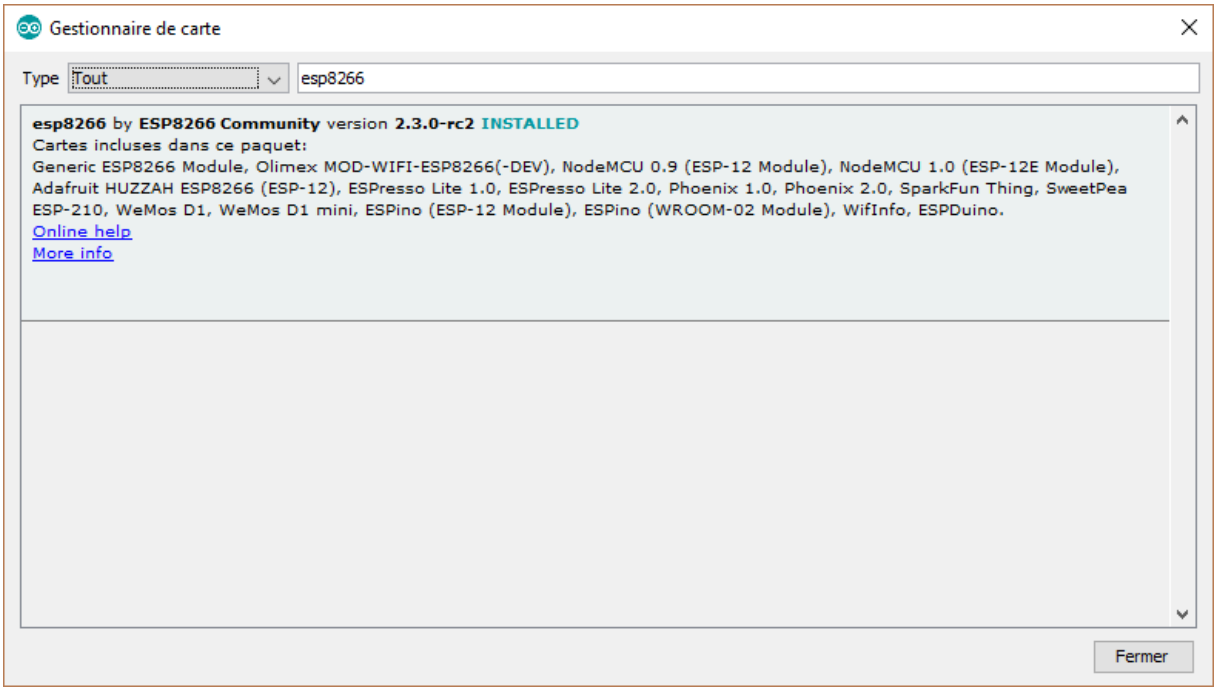


Puis, cliquez sur le bouton Installer

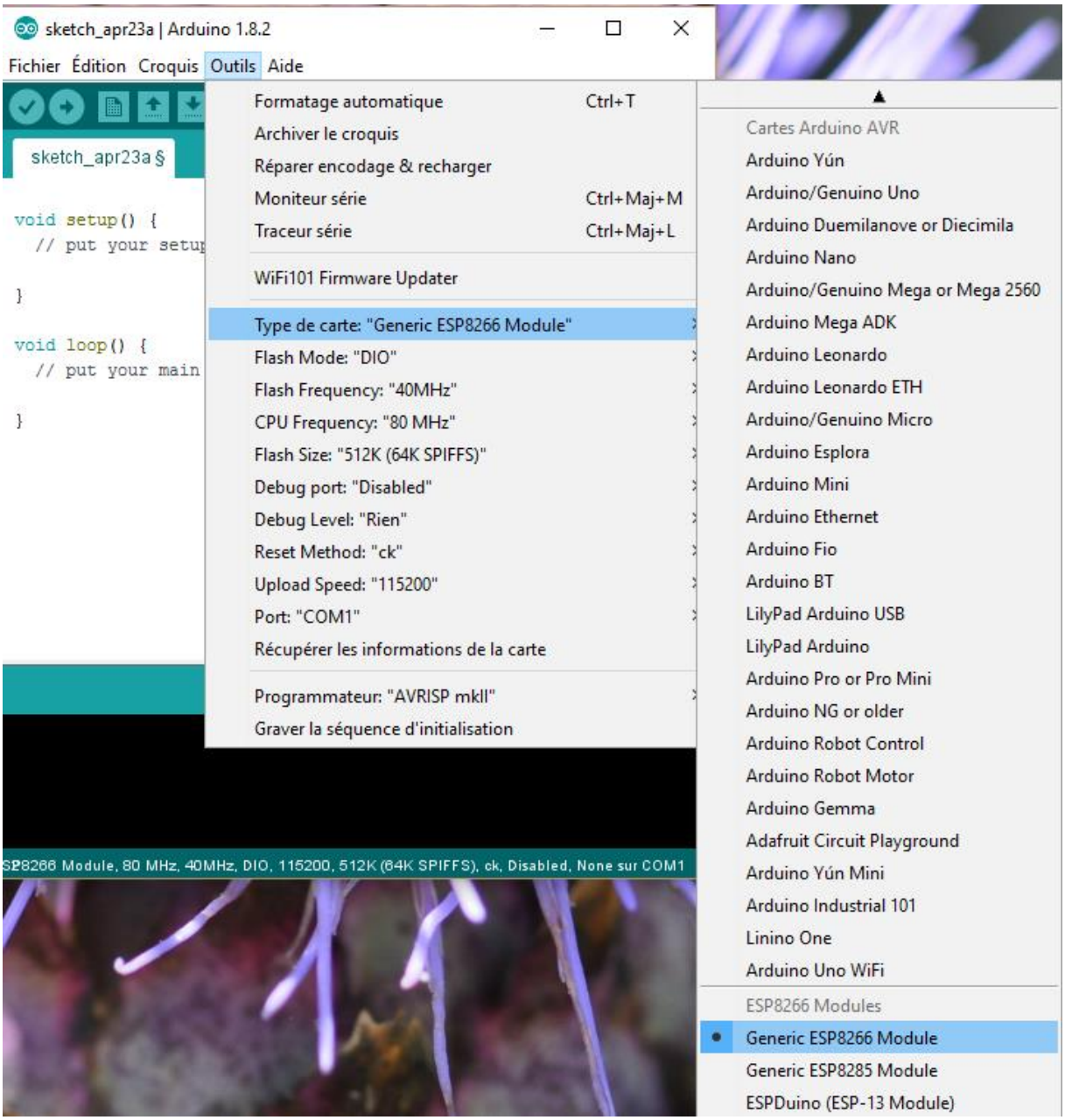




Puis « Fermer » lorsque c'est installé.



De nouveau dans le menu « Outil/Type de carte », sélectionnez maintenant le type de carte « Generic ESP8266 Module »

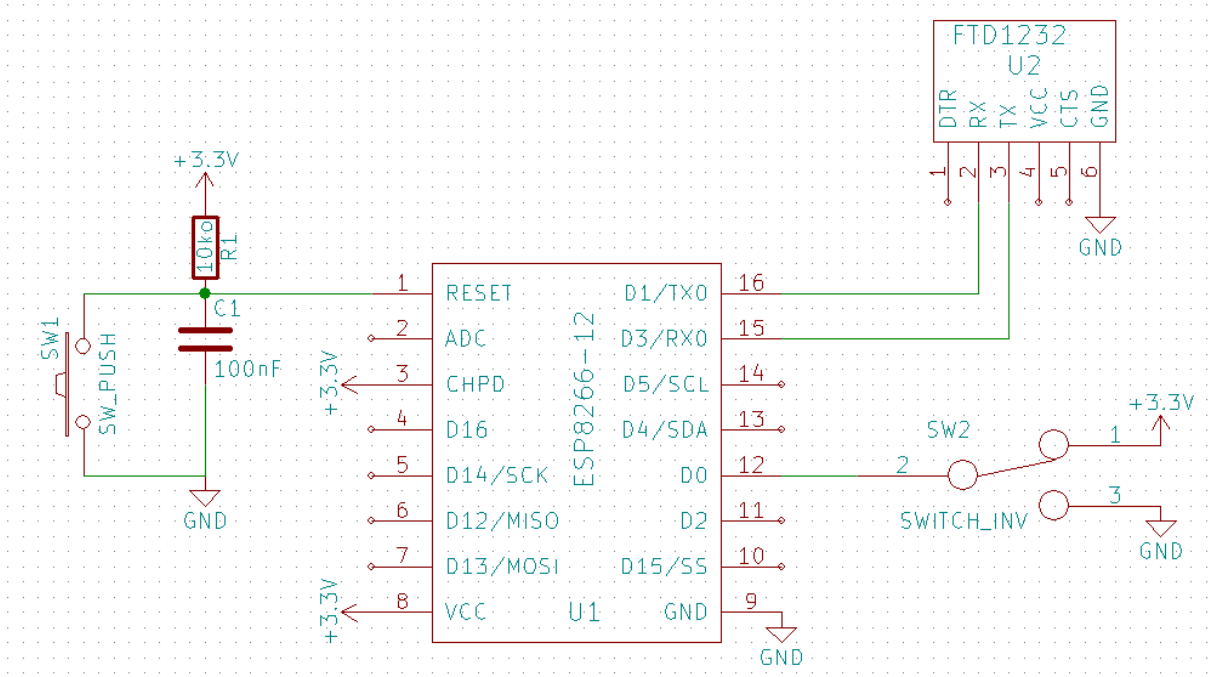


Un autre environnement possible est l'excellent Visual Studio qui peut être paramétré pour l'ESP8266

https://devenez-pro-en-electronique.com/programmer-son-arduino-avec-visual-studio/?fbclid=IwAR3ezwyIv3zZ3ceMGxcJS-cLK9_IJaQ8CQsO00sLQtNHCDR7IGWaCF234H4

3 Câblage de l'ESP8266-12 et du FTD1232

Le câblage sera le suivant :



Contrairement à ce qui est indiqué sur pas mal de site, les GPIO2 et GPIO15 ne doivent pas obligatoirement être branchés pour la programmation et le démarrage. Si vous branchez la broche GPIO2 à la masse ou au 3.3V, cela bloque la LED bleu qui est sur le module. Seule la broche D0 est vraiment utile pour basculer du mode programmation au mode fonctionnement et inversement. De plus, entre chaque bascule de mode, un Reset est indispensable pour le redémarrage correct de l'ESP8266.

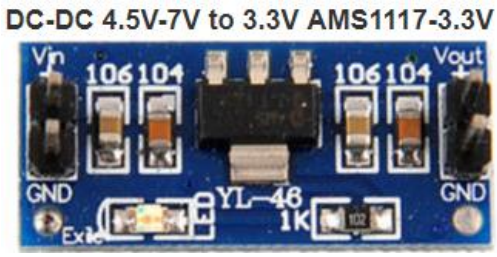
La résistance R1 et le condensateur C1 permettent d'éviter les rebonds du bouton afin d'avoir un seul RESET.

Pour l'alimentation, prévoir une alimentation externe de 3.3V qui peut délivrer environ 500mA. Ne pas le câbler sur l'alimentation du FTD1232 ou sur un Raspberry, elles ne délivreront pas assez de courant.

On trouve, toujours sur EBay, des petites alimentations pour quelques euros

Il existe également des mini convertisseurs 4.5V-7V → 3.3V pour une alimentation sur batterie de téléphone par exemple. Toutefois, ce module ne fournit pas énormément de courant (800mA max),

juste assez pour l'ESP8266 et un ou deux modules. Par contre, on le trouve à un prix très intéressant, j'en ai acheté 5 pour 1.69€ port compris.



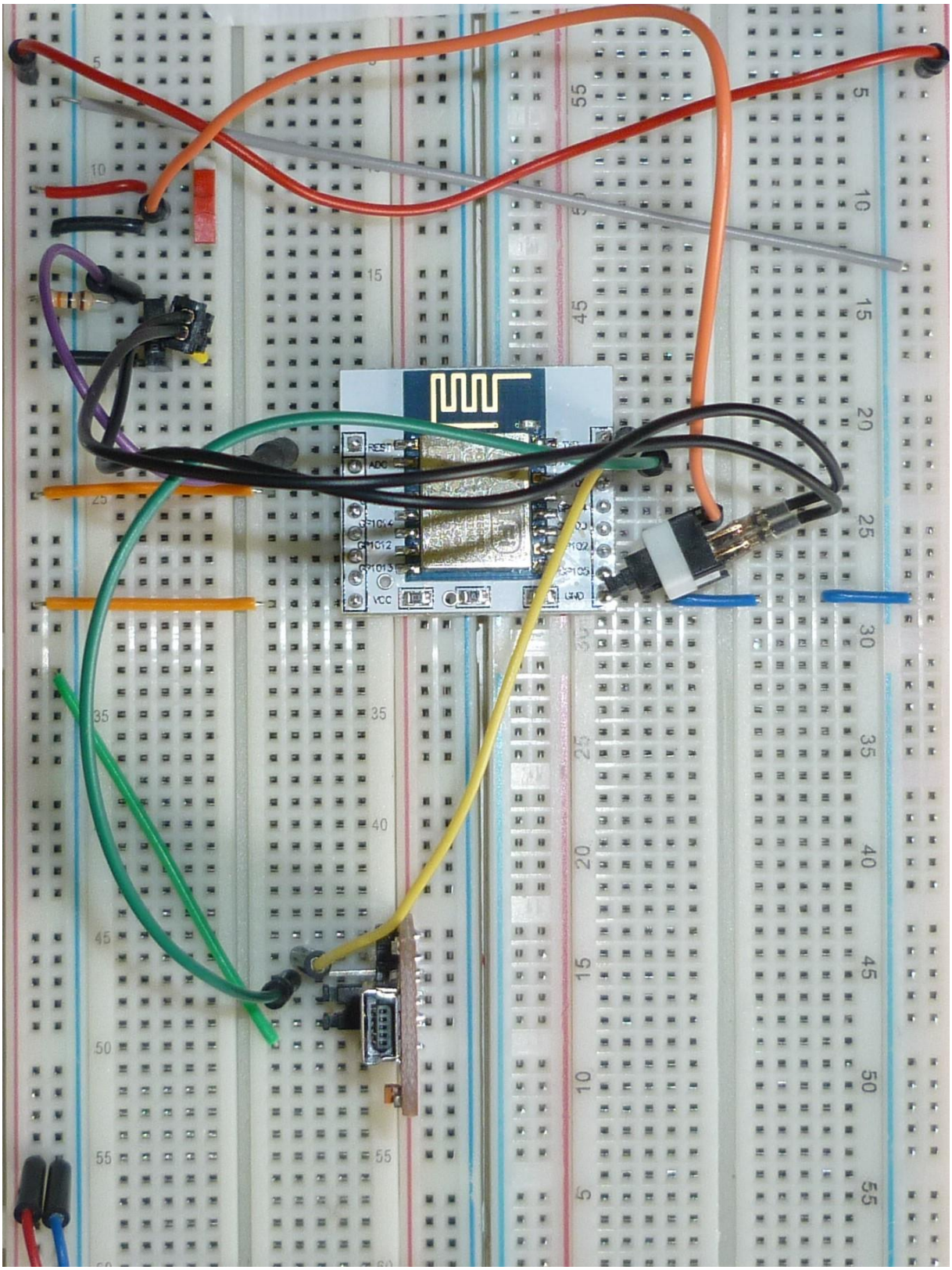
Sinon, pour les tests une alimentation de PC pourra aussi très bien faire l'affaire.

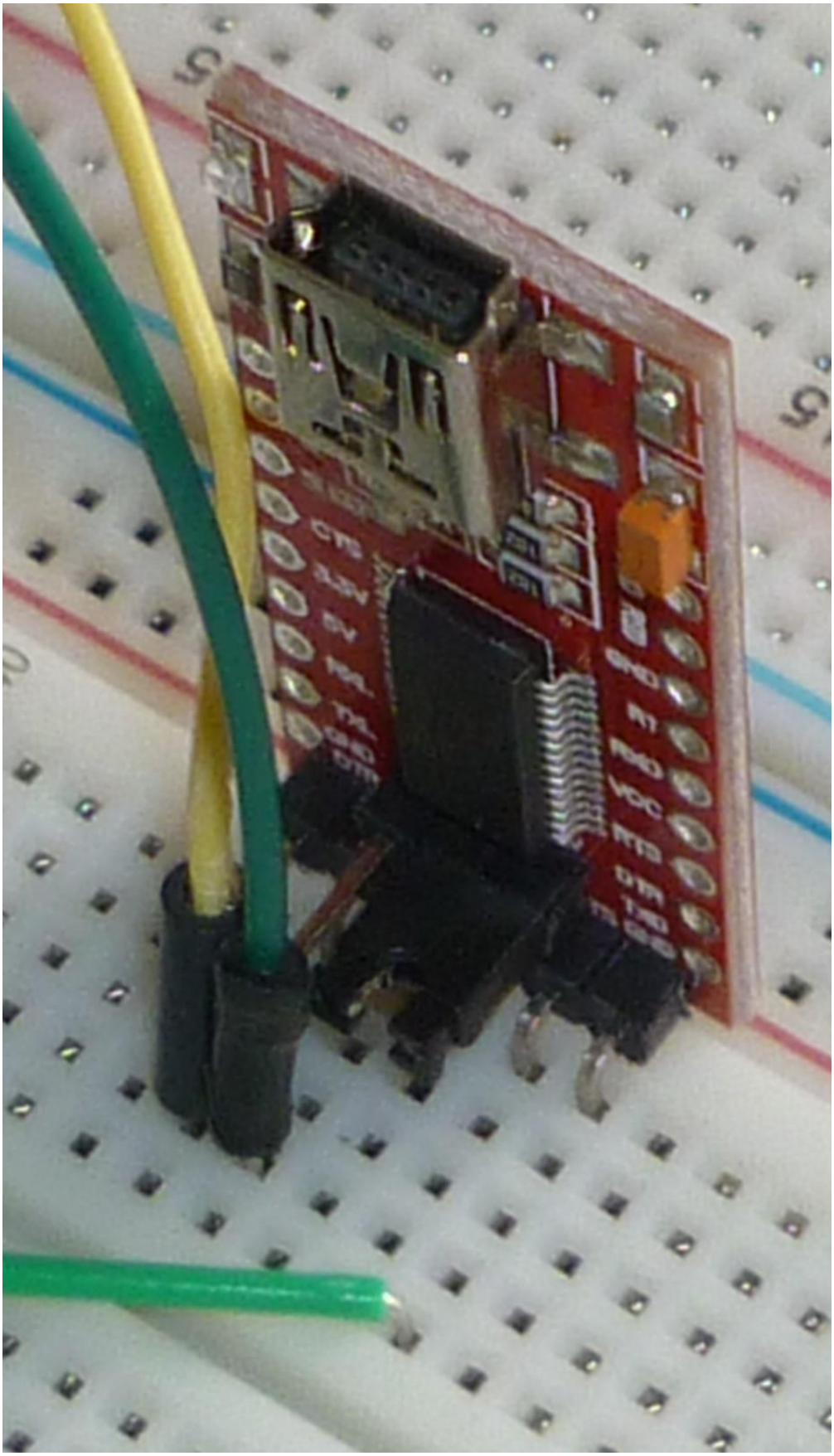
Couleur	Signal	Pin	Pin	Signal	Couleur
Orange	+3.3 V	1	13	+3.3 V	Orange
				+3.3 V sense	Brun
Orange	+3.3 V	2	14	−12 V	Bleu
Noir	Masse	3	15	Masse	Noir
Rouge	+5 V	4	16	Power on	Vert
Noir	Masse	5	17	Masse	Noir
Rouge	+5 V	6	18	Masse	Noir
Noir	Masse	7	19	Masse	Noir
Gris	Power good	8	20	Réservé	N/C
Violet	+5 V standby	9	21	+5 V	Rouge
Jaune	+12 V	10	22	+5 V	Rouge
Jaune	+12 V	11	23	+5 V	Rouge
Orange	+3.3 V	12	24	Masse	Noir

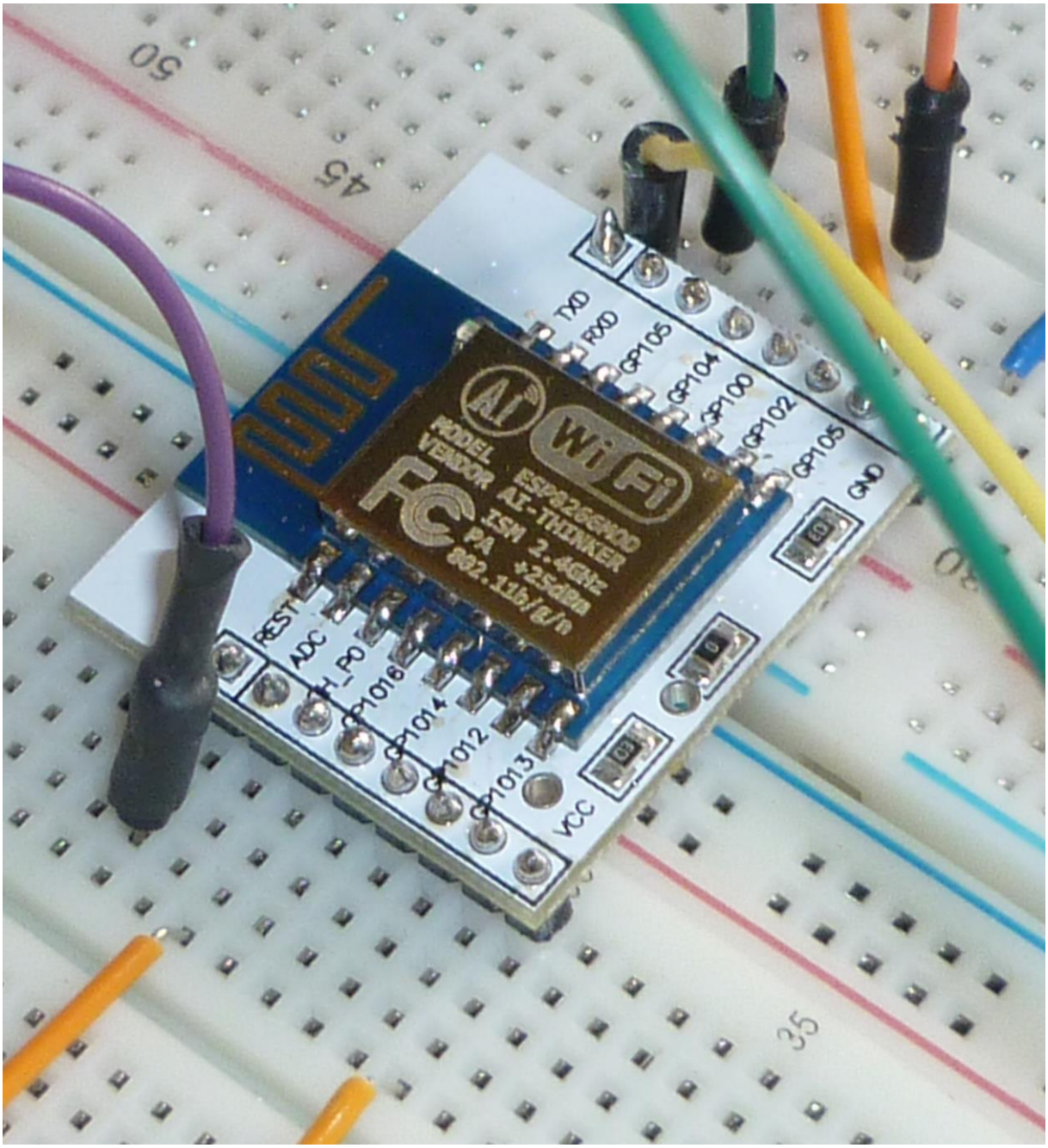
Pour assurer la connexion entre le PC et le convertisseur, ne pas oublier le câble USB

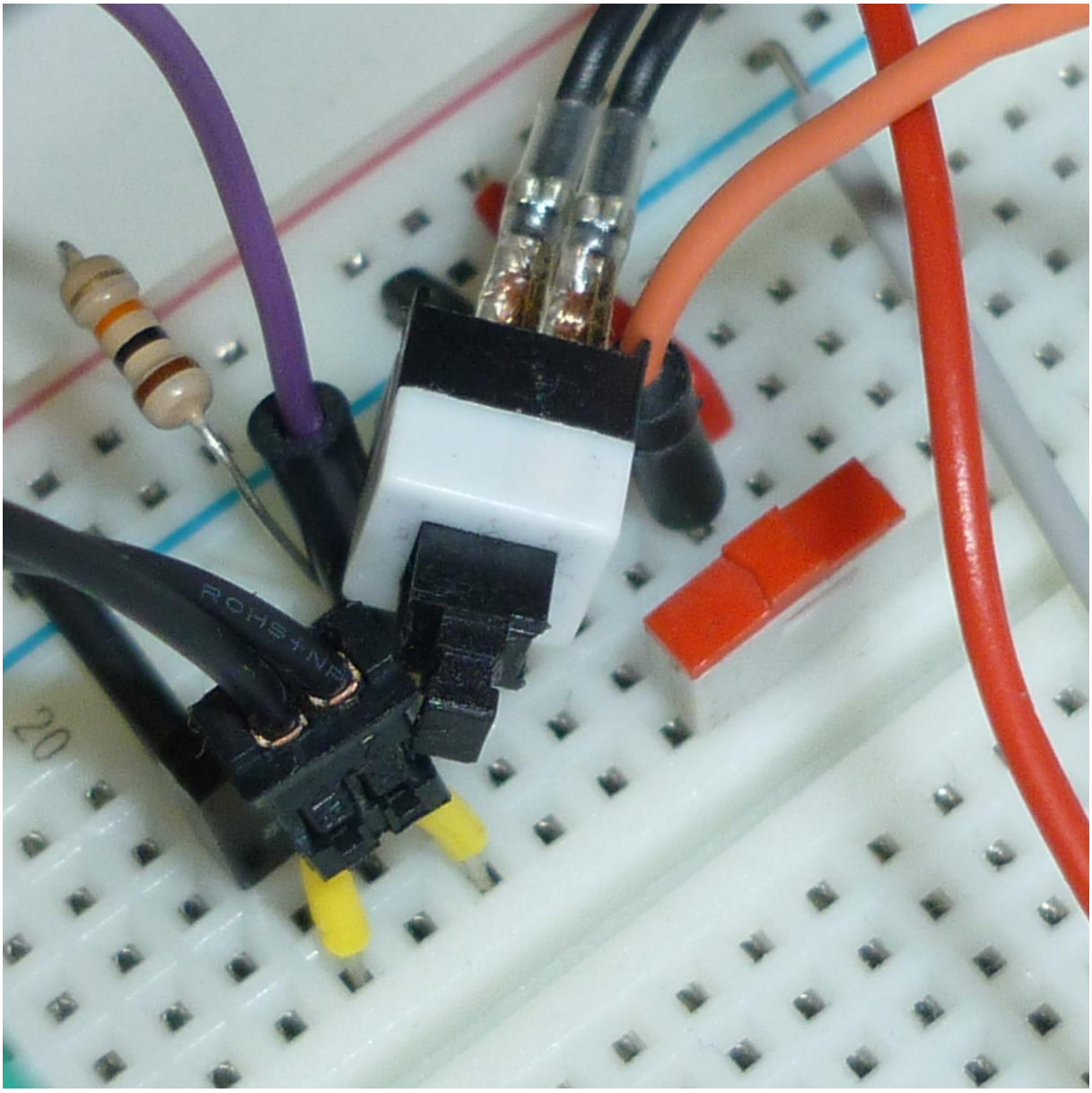
Il faut un USB A male, mini B male

J’ai fait le montage grâce à deux plaquettes de câblage côte à côte (le circuit est trop large pour une seule).






TXD
RXD
GPIO5
GPIO4
GPIO0
GPIO2
GPIO5
GND
VCC
GPIO13
GPIO12
GPIO14
GPIO16
CH_PD
ADC
REST
Wi Fi
MODEL
VENDOR
ESP8266MOD
AI-THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n



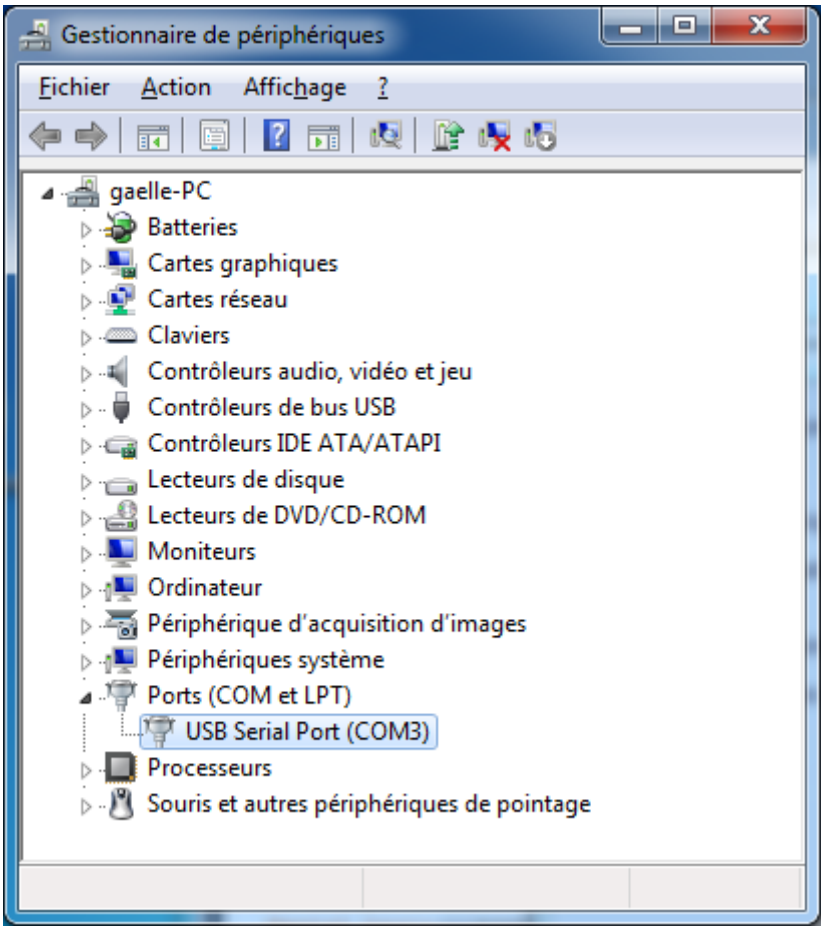
3.1 Installation du driver pour le convertisseur USB/Série FTD1232

Dès que vous brancherez le cable USB entre le convertisseur et un PC, ce dernier vous indiquera un nouveau matériel sur le port USB, mais n'aura pas le driver, il faudra donc l'installer à la main.

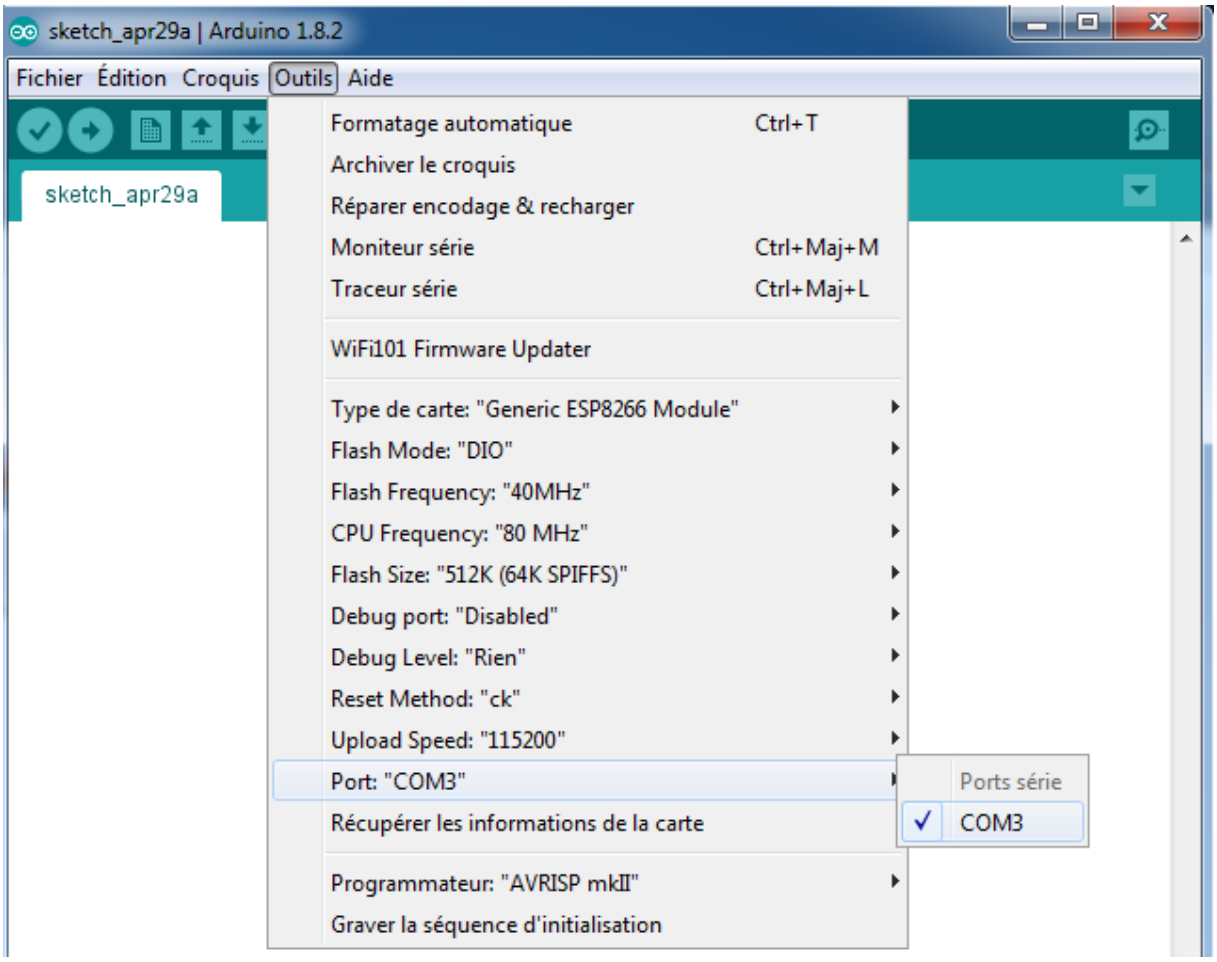
Ce module est basé sur le circuit FT232R. Le téléchargement et l'installation du driver pour ce circuit sont décrits ici http://www.usb-drivers.org/ft232r-usb-uart-driver.html.

Je n'ai par contre pas eut les faux ports USB Serial Port (COM19) et USB Serial Converter décrit dans la procédure (je suis en Windows 7). Je suis directement passé à l'installation du FT232R USB UART.

Mon convertisseur s’est positionné sur le port COM3



Il faut paramétrer ce numéro de port sur l'IDE Arduino



Voilà, l'environnement de développement est prêt.

4 Premier programme

4.1 Présentation

Voici le premier programme.

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

const char* ssid = "Livebox-XXXX"; // remplacer par le SSID de votre WiFi
const char* password = "......"; // remplacer par le mot de passe de votre WiFi

ESP8266WebServer server(80); // on instancie un serveur ecoutant sur le port 80

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

  // on demande la connexion au WiFi
  WiFi.begin(ssid, password);
  Serial.println("");

  // on attend d'etre connecte au WiFi avant de continuer
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // on affiche l'adresse IP qui nous a ete attribuee
  Serial.println("");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  // on definit ce qui doit etre fait lorsque la route /bonjour est appelee
  // ici on va juste repondre avec un "hello !"
  server.on("/bonjour", [](){
    server.send(200, "text/plain", "hello !");
  });

  // on commence a ecouter les requetes venant de l'exterieur
  server.begin();
}

void loop(void){
  // a chaque iteration, on appelle handleClient pour que les requetes soient traitees
  server.handleClient();
}
```

Il est issu du site https://www.fais-le-toi-meme.fr/fr/electronique/tutoriel/programmes-arduino-executes-sur-esp8266-arduino-ide qui m'a également aidé pour le câblage et l'installation de l'IDE Arduino.

Les programmes pour l'Arduino et l'ESP8266 ont tous la même structure, ils comportent au moins deux blocs.

- Un bloc « setup » qui sera exécuté une fois au démarrage du programme
- Un bloc « loop » qui est la boucle principal du programme, elle est exécutée indéfiniment jusqu'à l'arrêt ou le Reset du module

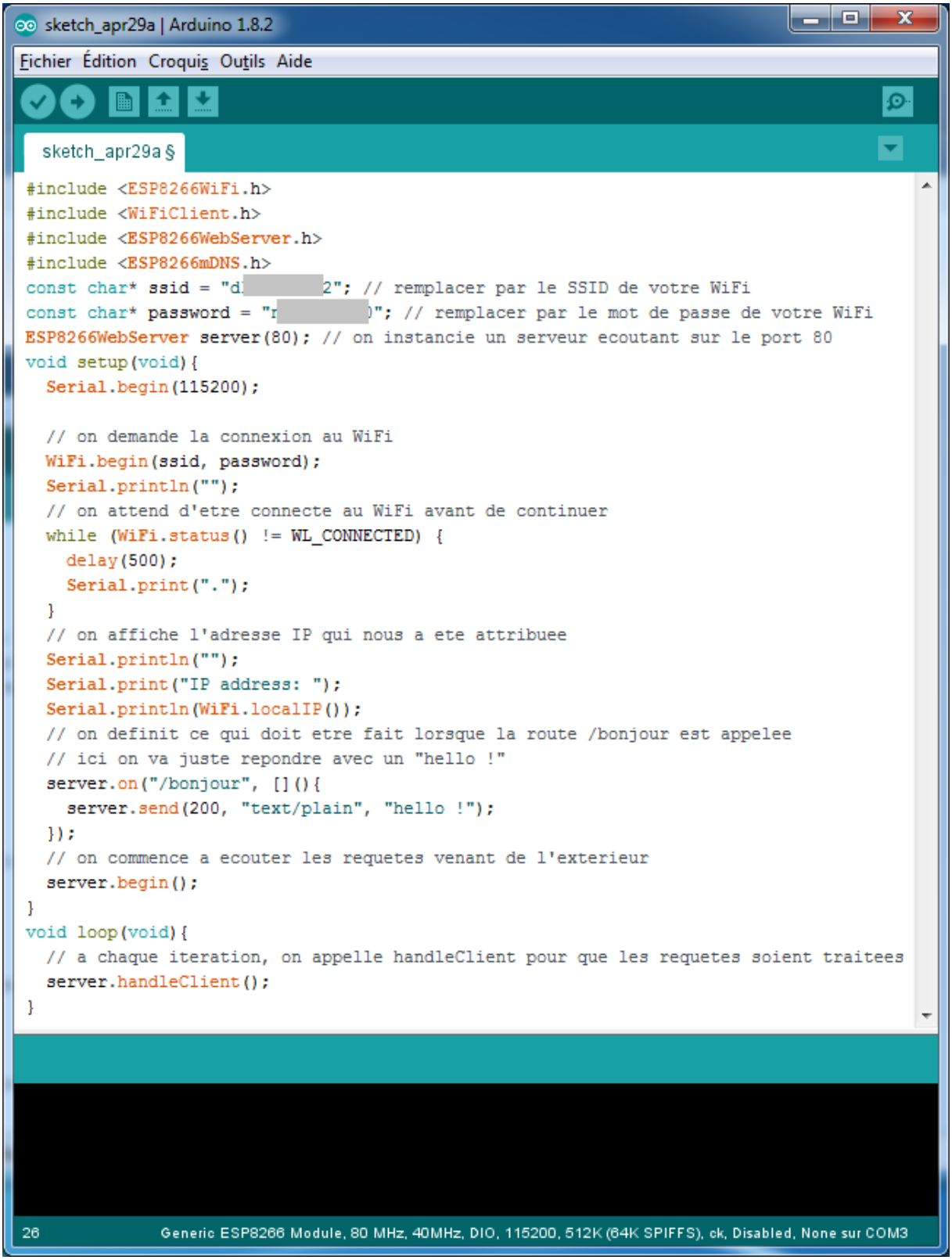
Attention : comme indiqué dans les commentaires, il faut bien alimenter les variables ssid et password avec les paramètres de votre Box.

Le programme est plutôt simple et bien commenté. Après une phase d'initialisation et de connexion à la box, il écoute une URL « /bonjour » et répondra « hello ! » en mode texte avec un code de retour HTML 200 indiquant le succès de la commande.

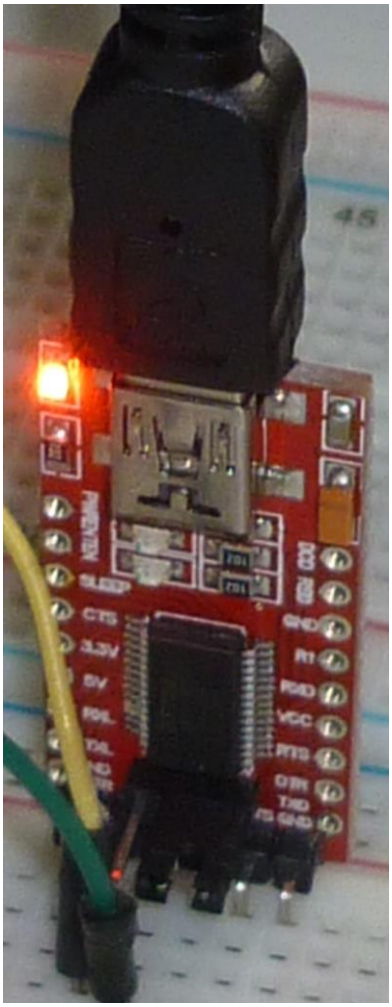
4.2 Compilation et chargement

Afin que notre programme fonctionne, il va falloir passer par une phase de compilation pour le traduire en langage du microcontrôleur. Puis l'envoyer du PC vers le microcontrôleur via notre convertisseur USB/Série

Taper ou copier le programme dans l'IDE



On va maintenant envoyer ce programme dans l'ESP8266. Il faut bien sûr que votre montage soit alimenté et que le câble USB soit branché entre le PC et le convertisseur. Une LED rouge doit normalement être allumée sur le convertisseur.

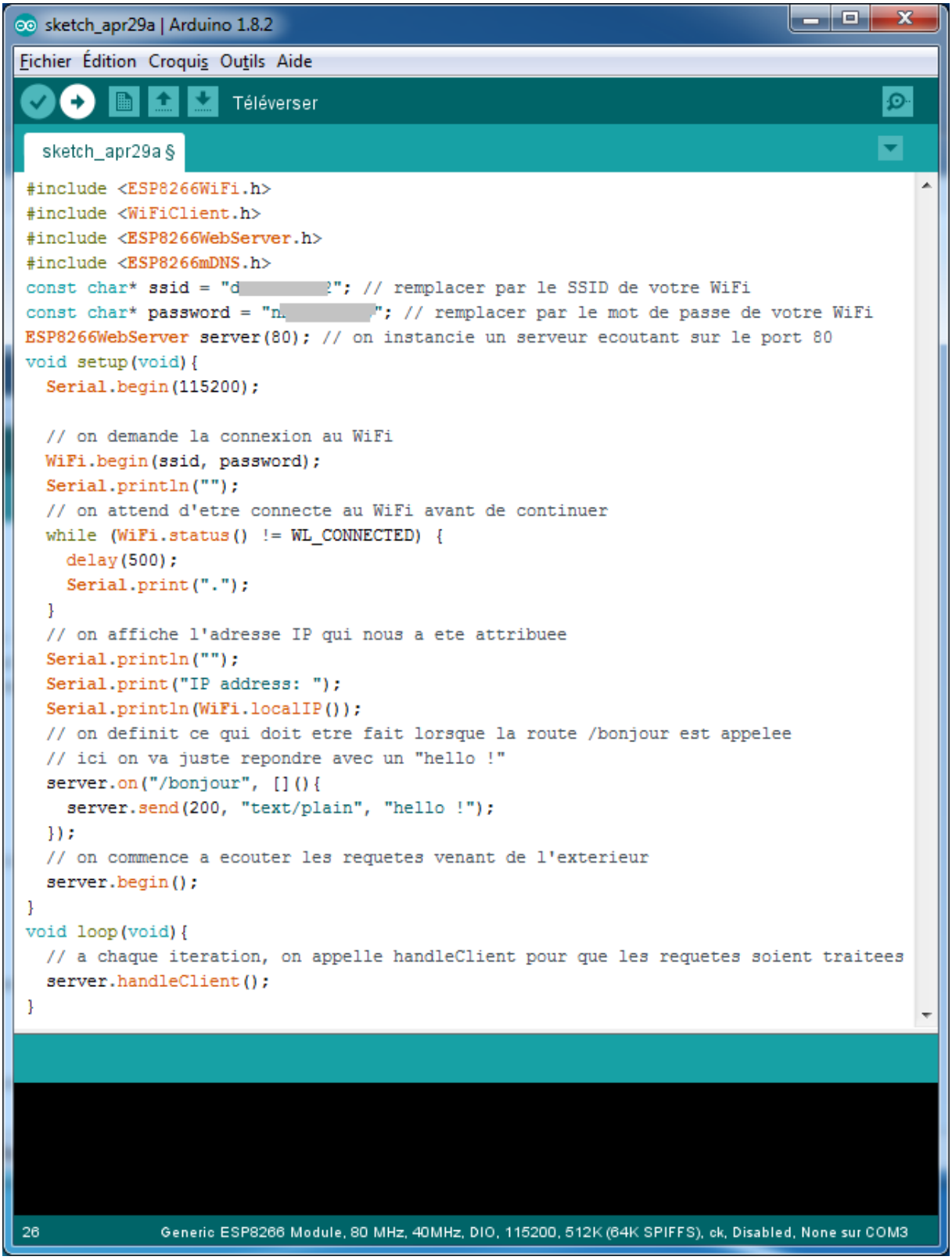


Pour envoyer le programme, il faut positionner l'inverseur de la broche GPIO0 sur GND et envoyer un RESET au microcontrôleur en appuyant sur le bouton (une LED bleu envoie un flash lorsque lors du RESET).

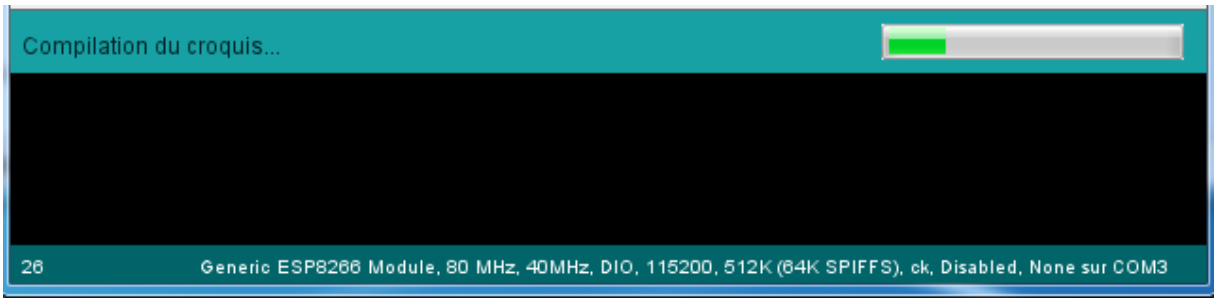
GPIO0 →	**VCC** ou **GND**	GND pour uploader un nouveau programme, 3.3v pour lancer le programme déjà présent
RST →	**VCC** ou **GND**	GND pour redémarrer l'ESP8266, 3.3v (ou non connecté) le reste du temps

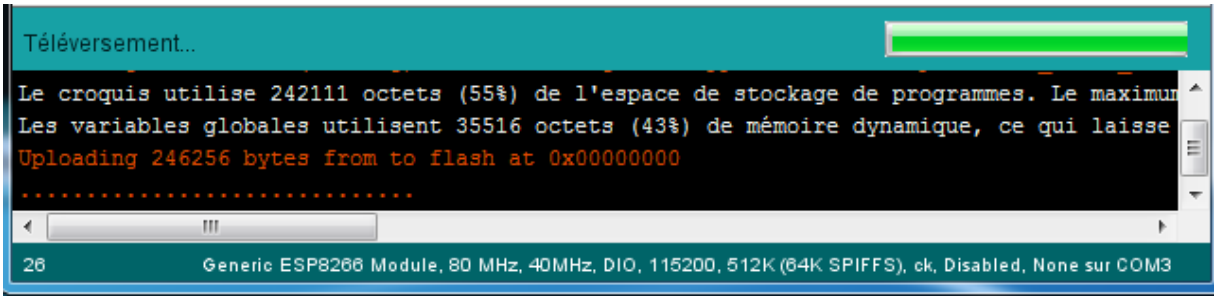
Dans tous les cas, si une opération ne fonctionne pas, n'hésitez pas à jouer du RESET et à recommencer depuis le début, le module est parfois assez capricieux.

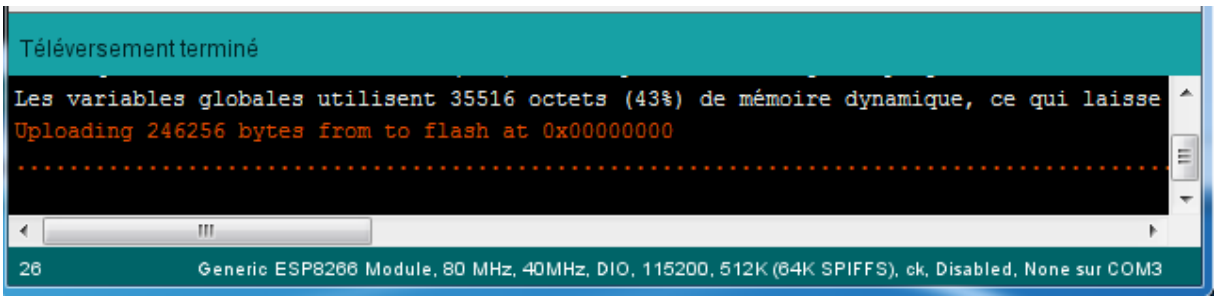
Pour démarrer l'opération, cliquez sur le bouton  en haut à gauche : Téléverser (que ce mot est vilain…)



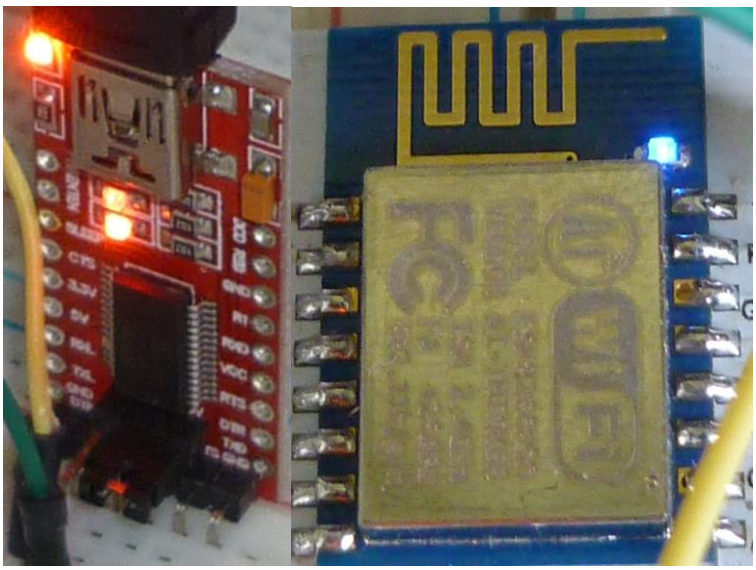
Il doit y avoir pas mal d'activité en bas de l'IDE







Lors de l'upload, le convertisseur doit clignoter ainsi que le microcontrôleur



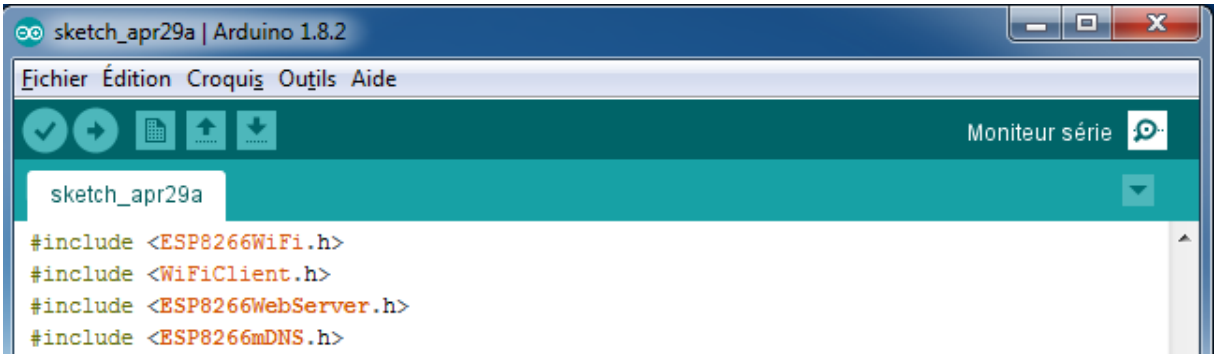
Si vous recevez un message de ce genre, c'est que l'inverseur GPIO0 est mal positionné ou que le RESET n'a pas été fait. Positionnez le GPIO correctement et faite un RESET afin de recommencer toute l'opération.

4.3 Exécution

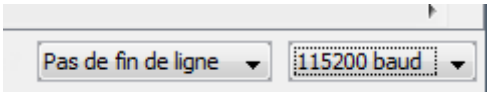
Après avoir envoyé le programme dans le microcontrôleur, on va l'exécuter.

Pour l'exécution du programme, il faut soit laisser brancher le convertisseur au port USB afin qu'il soit alimenté, soit l'enlever du montage, sinon le microcontrôleur ne démarrera pas.

Dans un premier temps, ouvrir le moniteur série avec l'icône en haut à droite. Cela permettra de voir les informations que nous envoie le microcontrôleur. Sans ce moniteur, nous n'aurions aucune information.



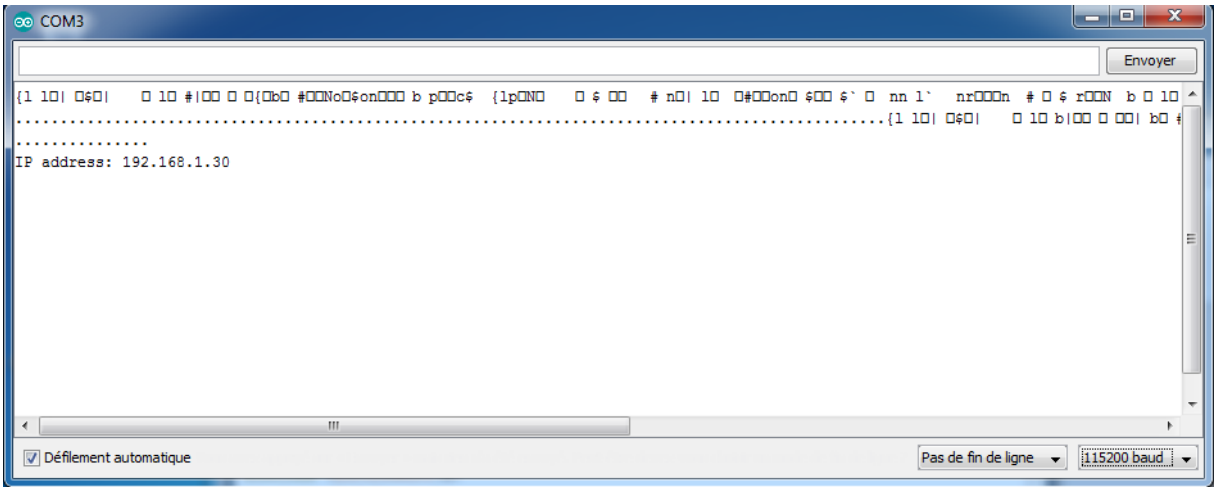
Et positionnez sa vitesse sur 115200 baud



Afin de démarrer le programme, positionnez le GPIO0 sur 3.3V et faite un RESET

Le moniteur va recevoir pleins de signes cabalistiques puis une série de point indiquant que le programme est en train de se connecter à la Box.

Lorsque le programme est connecté, il envoi l'adresse IP de l'ESP8266, notez la bien elle va beaucoup nous resservir.

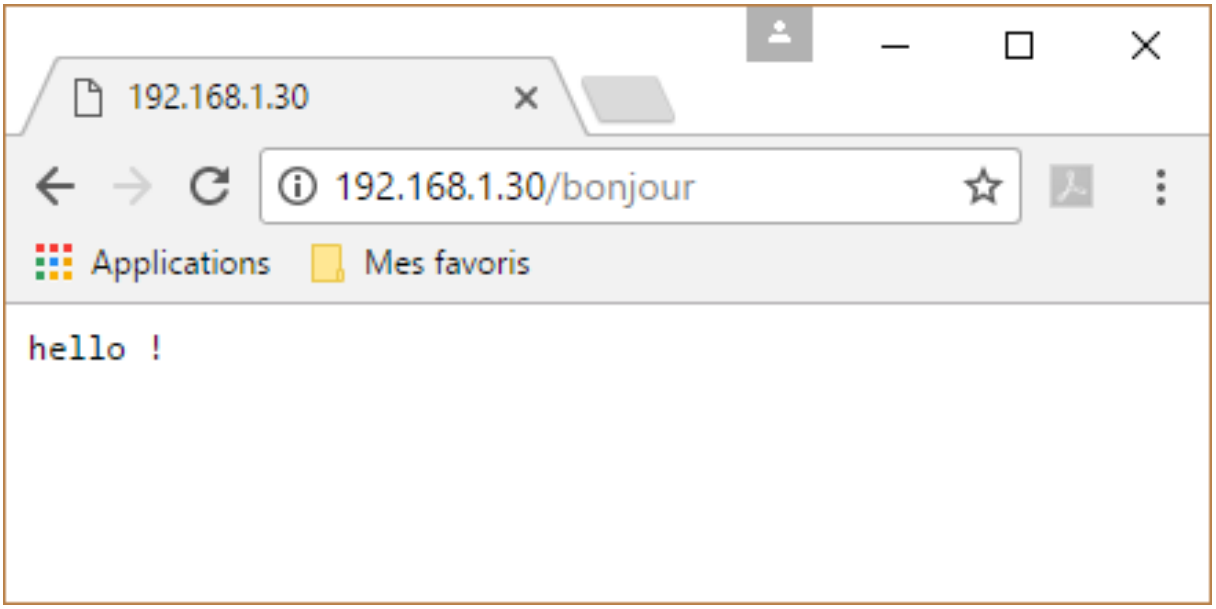


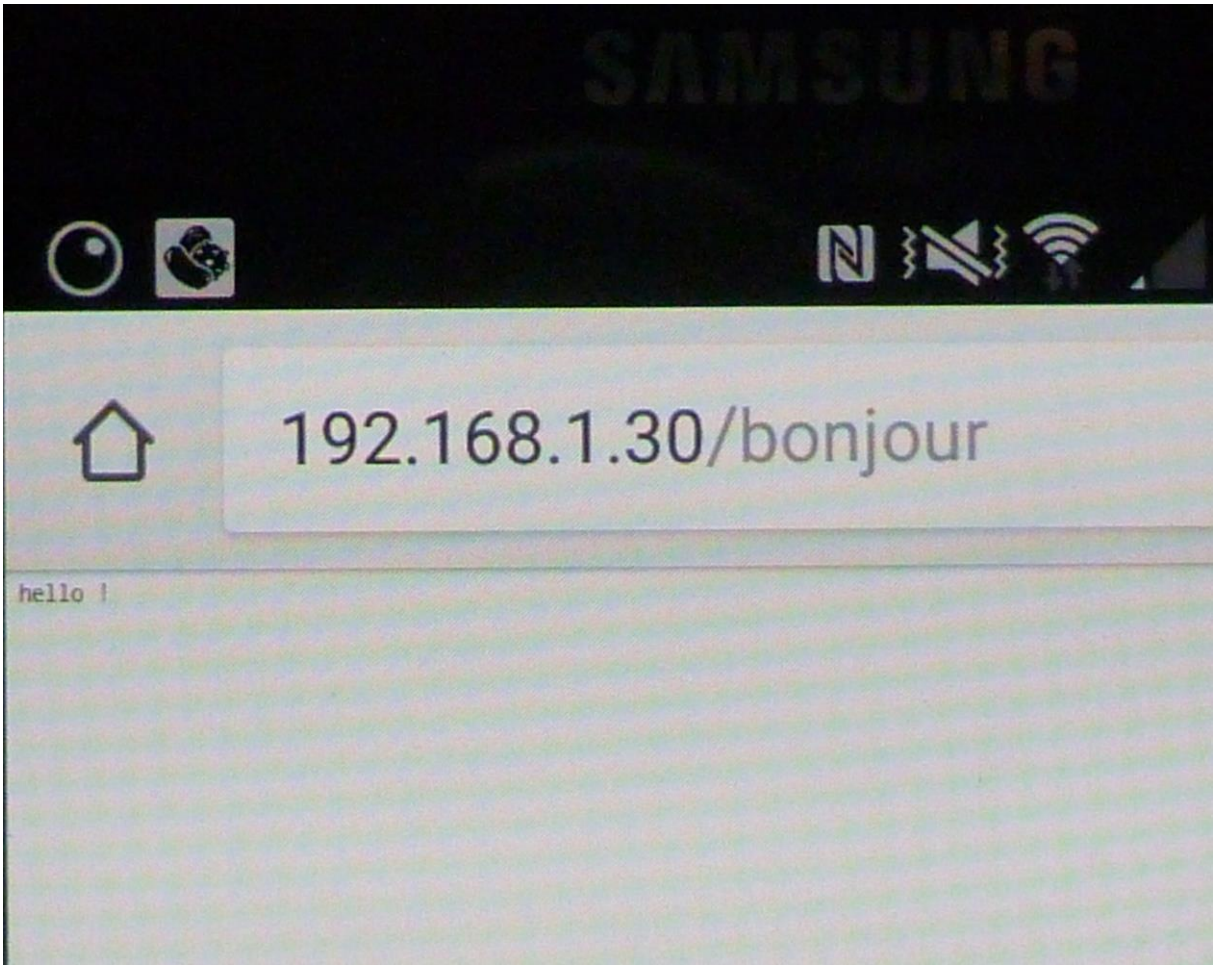
Là encore, si c'est un peu long, un coup de RESET… (Si c'est très long, vérifiez les informations de connexion en début de programme)

Lorsqu'il a affiché son adresse IP, le microcontrôleur est démarré et attend une requête.

Pour tester, il suffit d'appeler l'URL http://192.168.1.30/bonjour (changez l'adresse IP avec la vôtre si nécessaire). Cet appel peut être fait via un navigateur sur un ordinateur ou un téléphone quelconque, du moment qu'il connecté à la même Box que le microcontrôleur. (En cas de doute, n'hésite pas à faire un ping de l'adresse du microcontrôleur pour confirmer qu'il est bien visible de votre PC ou téléphone)

Le microcontrôleur affichera alors le « hello ! » sur le navigateur.





4.4 Liste des programmes

https://github.com/montotof123/esp8266-12/blob/master/010_Bonjour/bonjour.ino

5 Faire clignoter une LED

En farfouillant sur Internet, on finit par s'apercevoir que la LED bleue sur le module est câblée sur le GPIO2.

Donc, on peut faire un programme très simple pour la faire clignoter

```
/*
 ESP8266 Blink
 Blink the blue LED on the ESP-12 module

 The blue LED on the ESP-12 module is connected to GPIO2
 (which is also the TXD pin; so we cannot use Serial.print() at the same time)
*/

#define LED 2   //Define connection of LED

void setup() {
  pinMode(LED, OUTPUT);     // Initialize the LED_BUILTIN pin as an output
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED, LOW);   // Turn the LED on (Note that LOW is the voltage level
 // but actually the LED is on; this is because
 // it is active low on the ESP-12)
  delay(1000);              // Wait for a second
  digitalWrite(LED, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(1000); // Wait for two seconds (to demonstrate the active low LED)
}
```

Ce programme vient du site **http://circuits4you.com/2016/12/16/esp8266-led-blink/**

Son fonctionnement est très simple, dans le setup il paramètre le GPIO2 où la LED est câblé en sortie. Ensuite il boucle en basculant la GPIO2 de haut à bas toutes les secondes.

Même manipulation qu'au chapitre précédent :

- GPIO0 sur GND
- RESET
- Upload du programme
- GPIO0 sur 3.3V
- RESET

5.1 Liste des programmes

https://github.com/montotof123/esp8266-12/blob/master/020_Blink/blink.ino

6 Allumer/éteindre une LED

6.1 La théorie

On va maintenant coupler les deux précédents chapitres en commandant l'allumage ou l'extinction de la LED avec une commande sur le serveur web.

6.2 L'électronique

Coté électronique, aucun changement, on va encore utiliser la LED bleu qui est câblée sur le microcontrôleur.

6.3 Le logiciel

Pour le logiciel on va prendre le premier qui gérait l'URL /bonjour et on va en ajouter trois autres.

Ne pas oublier que la LED est active à l'état bas, donc tous les ordres sont inversés (on = low et off = high)

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

#define LED 2    //Define connection of LED

const char* ssid = "**********"; // remplacer par le SSID de votre WiFi
const char* password = "***********"; // remplacer par le mot de passe de votre WiFi

ESP8266WebServer server(80); // on instancie un serveur ecoutant sur le port 80

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

  // on demande la connexion au WiFi
  WiFi.begin(ssid, password);
  Serial.println("");

  // on attend d'etre connecte au WiFi avant de continuer
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // on affiche l'adresse IP qui nous a ete attribuee
  Serial.println("");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  // On bascule le port en sortie
  pinMode(LED, OUTPUT);

  // on definit ce qui doit etre fait lorsque la route /bonjour est appelee
  // ici on va juste repondre avec un "hello !"
  server.on("/bonjour", [](){
    server.send(200, "text/plain", "hello !");
  });

  // Allumer la LED avec la route /on
  server.on("/on", [](){
       digitalWrite(LED, LOW);
    server.send(200, "text/plain", "allumer LED !");
  });

  // Eteindre la LED avec la route /off
  server.on("/off", [](){
       digitalWrite(LED, HIGH);
    server.send(200, "text/plain", "éteindre LED !");
  });

  // Allumer ou éteindre la LED avec un paramètre /led?state=on /led?state=off
  server.on("/led", [] () {
    String state=server.arg("state");
    if (state == "on") digitalWrite(LED, LOW);
    else if (state == "off") digitalWrite(LED, HIGH);
    server.send(200, "text/plain", "Led is now " + state);
  });

  // on commence a ecouter les requetes venant de l'exterieur
  server.begin();
}

void loop(void){
  // a chaque iteration, on appelle handleClient pour que les requetes soient traitees
  server.handleClient();
}
```

Les ordres possibles sont :

- http://192.168.1.30/bonjour
- http://192.168.1.30/off
- http://192.168.1.30/on
- http://192.168.1.30/led?state=off
- http://192.168.1.30/led?state=on

6.4 Liste des programmes

https://github.com/montotof123/esp8266-12/blob/master/030_Led/led.ino

7 Allumer/éteindre une LED depuis n'importe où sur la planète

On laisse tourner le microcontrôleur avec le programme précédent, mais on va lui permettre d'être appelé depuis n'importe où du moment qu'on a un accès internet.

Si vous avez Une box, il suffit d'ouvrir l'accès de votre ESP8266 vers l'extérieur.

Pour une LiveBox, voici un petit tuto https://communaute.orange.fr/t5/les-offres-Internet-Orange-et/Parametrage-d-une-DMZ/td-p/468333, je vous laisse chercher pour les autres modèles de box.

Chez moi, j'ai paramétré ceci.

C'est-à-dire que pour l'appareil ESP_D527B1 que la Box connait comme étant le 192.168.1.30, j'ouvre vers l'extérieur son port 80 (c'est celui qu'on a mis dans le programme) sans redirection de port, c'est-à-dire que j'aurais également le port 80 visible de l'extérieur.

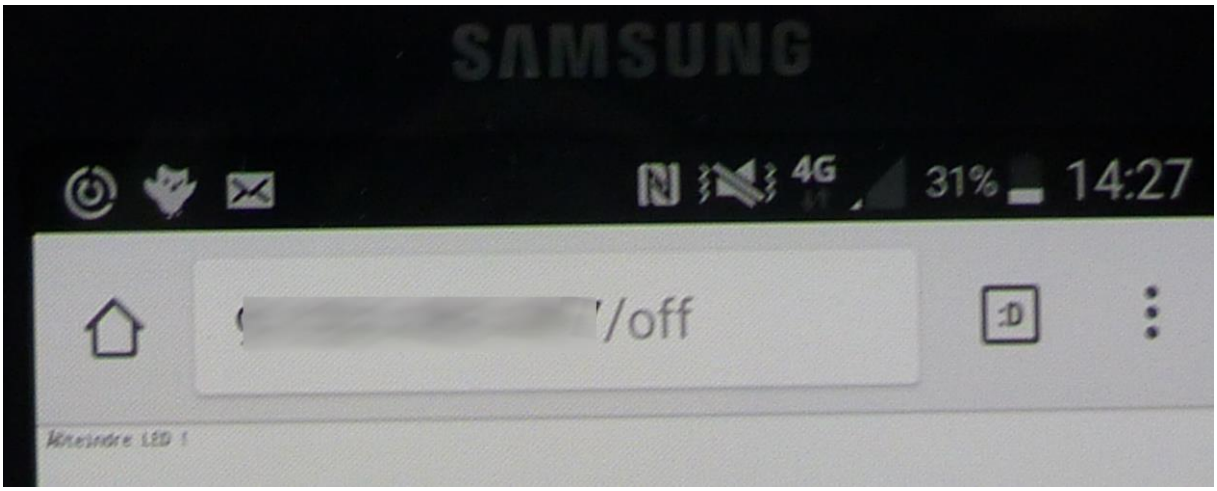
Je récupère l'adresse IP de ma LiveBox vue de l'extérieur.

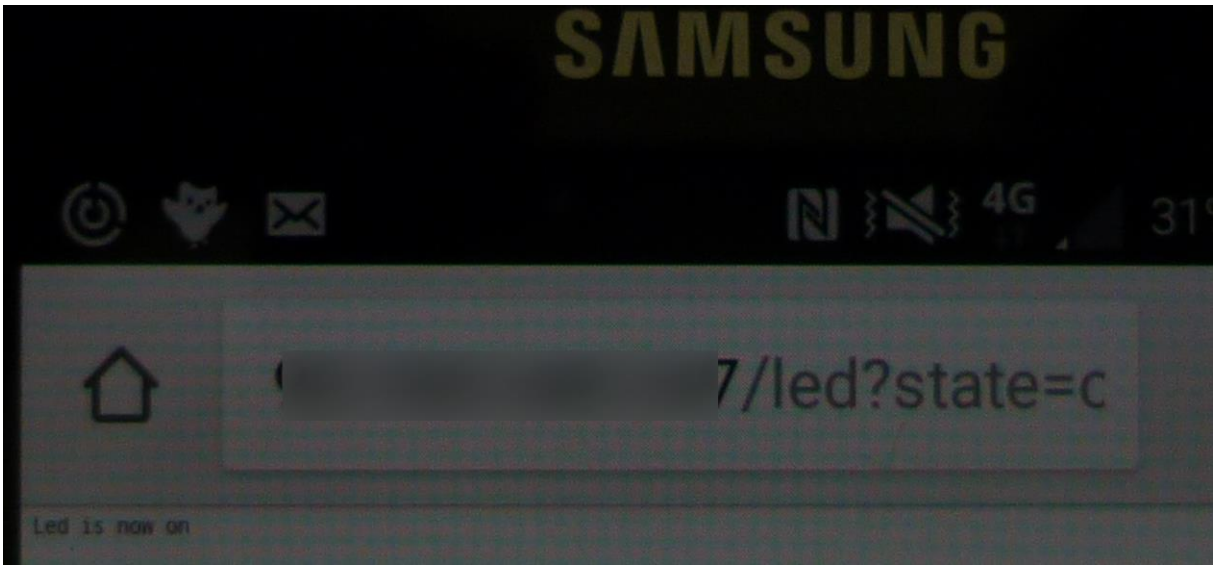
informations système

3.1	statut de la connexion Internet	connecté
3.2	nom d'utilisateur	
3.3	dernière connexion	20 avril 2017, 19 h 05 m
3.4	durée de la connexion	00 j 19 h 16 m 56 s
3.5	type du protocol	ppp
3.6	code d'erreur de la dernière connexion	ERROR_NONE
3.7	date de la dernière connexion	0
3.8	ATM VP/VC ou VLAN	8/35
3.9	taille MTU	1492
3.10	adresse IPv4 WAN	[redacted]
3.12	adresse IP du DNSv4 primaire	80.10.246.3
3.13	adresse IP du DNSv4 secondaire	81.253.149.10

aid
Informat
connexic
le s
Vous pou
plages h
WiFi de
l'interfac
Livebox.

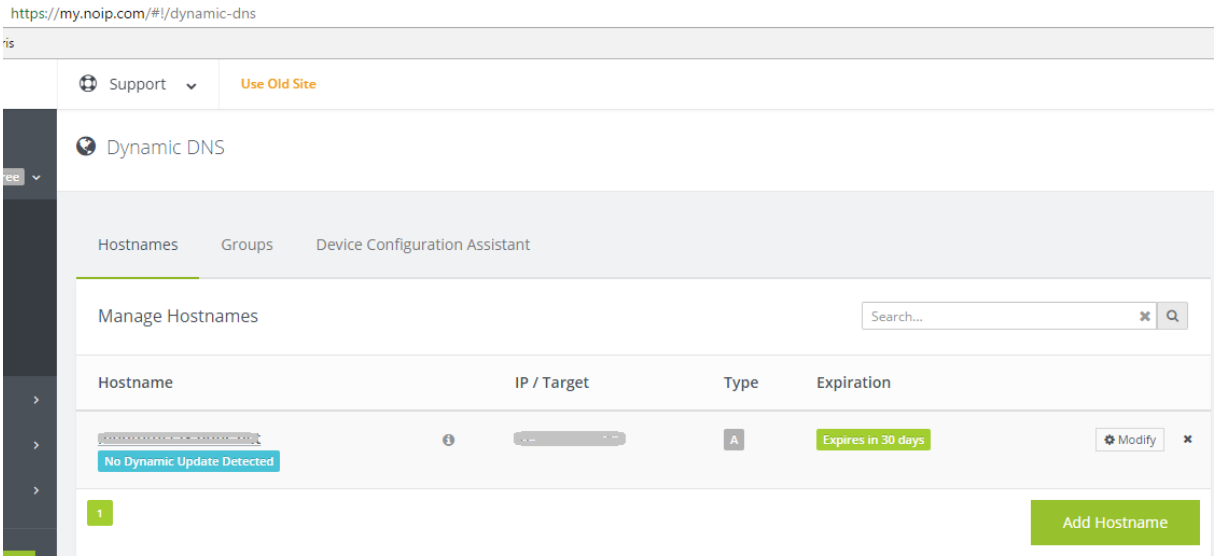
Et un petit test sur un téléphone en 4G (pensez à bien désactiver le Wifi) avec l'adresse IP de la LiveBox et les url





Bon, c'est bien, mais le seul problème c'est que régulièrement, ma LiveBox va changer d'adresse IP. Et que je ne serai pas au courant.

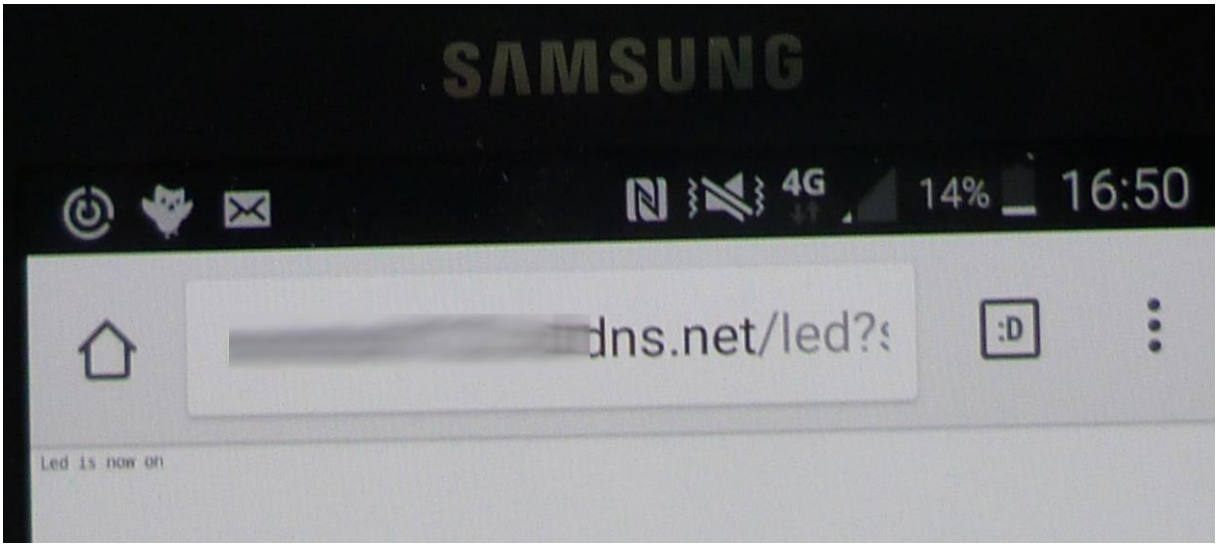
La solution, c'est de s'inscrire à un service qui attribue un nom de domaine à votre adresse IP de Livebox. Personnellement, j'utilise NoIP (https://www.noip.com/).



Je me suis inscrit et j'ai paramétré le service sur ma Live Box afin qu'elle envoie les informations à NoIP à chaque changement d'adresse IP (Le message Expire in 30 days indique que si vous ne faite rien, le service sera interrompu dans 30 jours. Mais NoIP vous enverra un mail 7 jours avant pour renouveller)

Il suffit de mettre le Hostname pour le nom d'hôte et les noms d'utilisateur et mot de passe que vous avez choisis à l'inscription à NoIP

Maintenant, au lieu d'utiliser l'adresse IP, il suffit de faire l'appel par le Hostname donné par NoIP

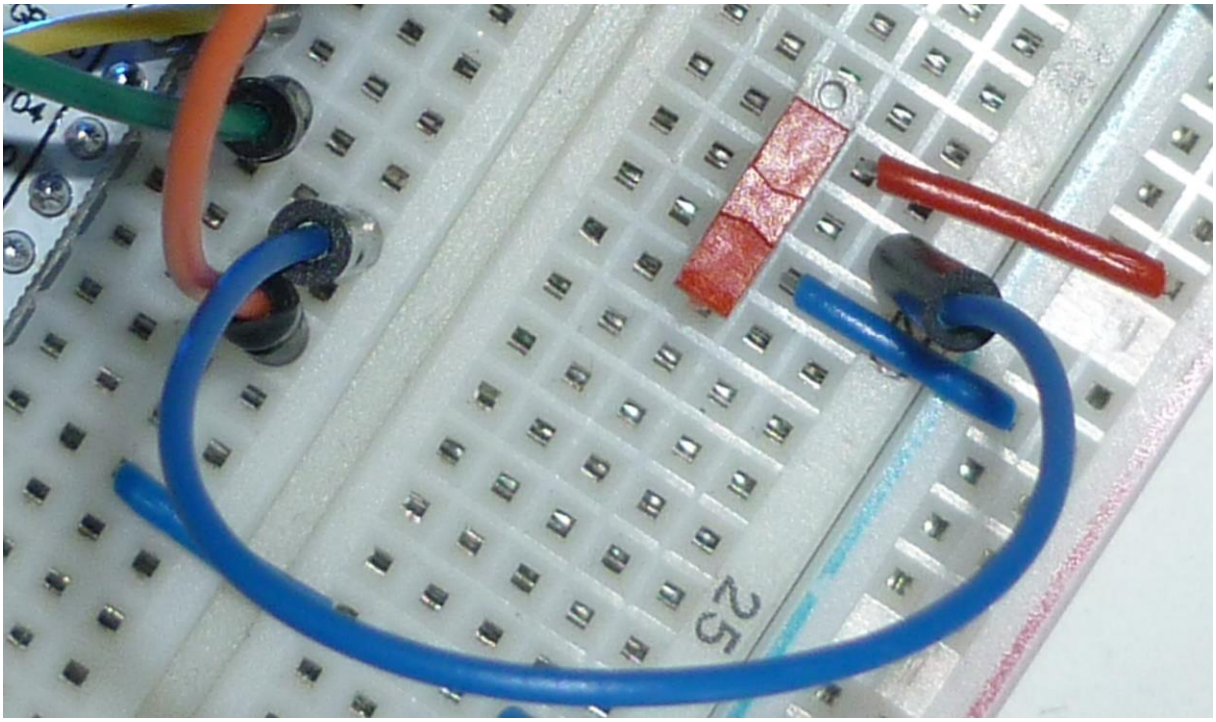

SAMSUNG
4G
14%
16:50
dns.net/led?s
Led is now on

8 Lire un port GPIO

Le but de ce chapitre est de savoir lire l'état d'un port GPIO.

8.1 L'électronique

On ajoute un inverseur qui enverra 3.3V ou 0V sur la GPIO5

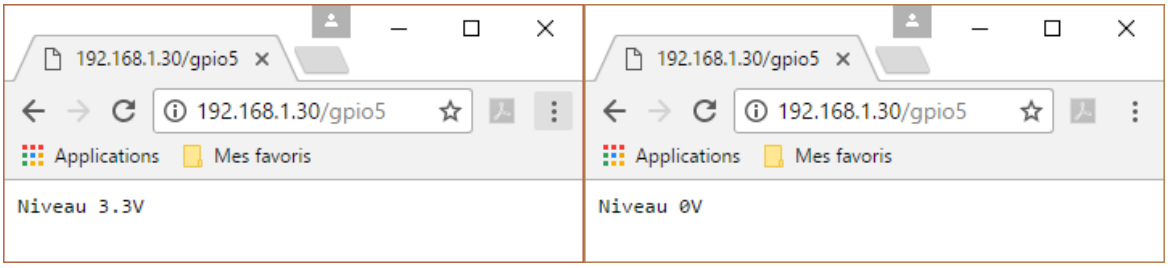


8.2 Le logiciel

On utilise encore le programme précédent, mais on va ajouter une route pour lire l'état du GPIO5 en entrée

```
// Lire le GPIO5
server.on("/gpio5", [](){
      if(digitalRead(5) == HIGH) {
            server.send(200, "text/plain", "Niveau 3.3V");
      } else {
            server.send(200, "text/plain", "Niveau 0V");
      }
});
```

Ce qui donne



8.3 Liste des programmes

https://github.com/montotof123/esp8266-12/blob/master/040_ReadGpio/readGpio.ino

9 Lire le convertisseur a/d

10 Envoi d'un mail au changement d'état d'un port

Tout est dans le titre, le but est d'envoyer un mail lorsqu'il se passe quelque chose qui change l'état d'un port GPIO, par exemple l'ouverture d'une porte ou d'une fenêtre.

10.1 La théorie

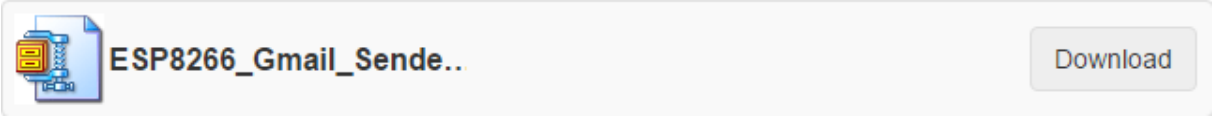
Dans ce sujet, il y a deux problèmes :

- Envoyer un mail
- Détecter le changement d'état d'un port

10.2 Logiciel pour l'envoi d'un mail

Je vais utiliser la librairie et l'exemple trouvé sur ce site http://www.instructables.com/id/ESP8266-GMail-Sender/

Il faut récupérer le code



Le dézipper dans un répertoire et l'ouvrir dans l'IDE Arduino

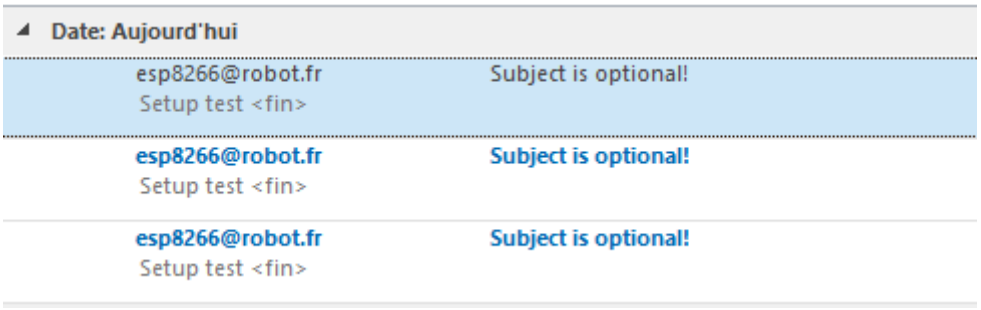
On se retrouve avec trois fichiers ouverts, une librairie et son entête et un exemple.

Il faut ensuite changer quelques paramètres afin de se connecter à la Box bien sûr mais aussi à son serveur de messagerie.

Attention : le login et le password de messagerie doivent être encodés en base64, tout est bien expliqué dans le tutoriel pour y arriver.

Par contre, il est basé pour l'envoi d'un mail via Gmail, si vous êtes avec un autre fournisseur, il faudra changer la constante SMTP_SERVER. Pour moi, chez Orange, il me faut smtp.orange.fr. On peut facilement trouver la liste sur Internet, par exemple sur http://www.serversmtp.com/fr/liste-serveur-smtp

On charge le programme dans l'ESP8266 et à chaque RESET, un mail est envoyé.



Comme on peut le voir ici, le FROM ne doit pas obligatoirement représenter votre adresse email réelle. (Évitez quand même les bêtises, un mail est toujours traçable…)

10.3 Logiciel pour la détection du changement d'état d'un port

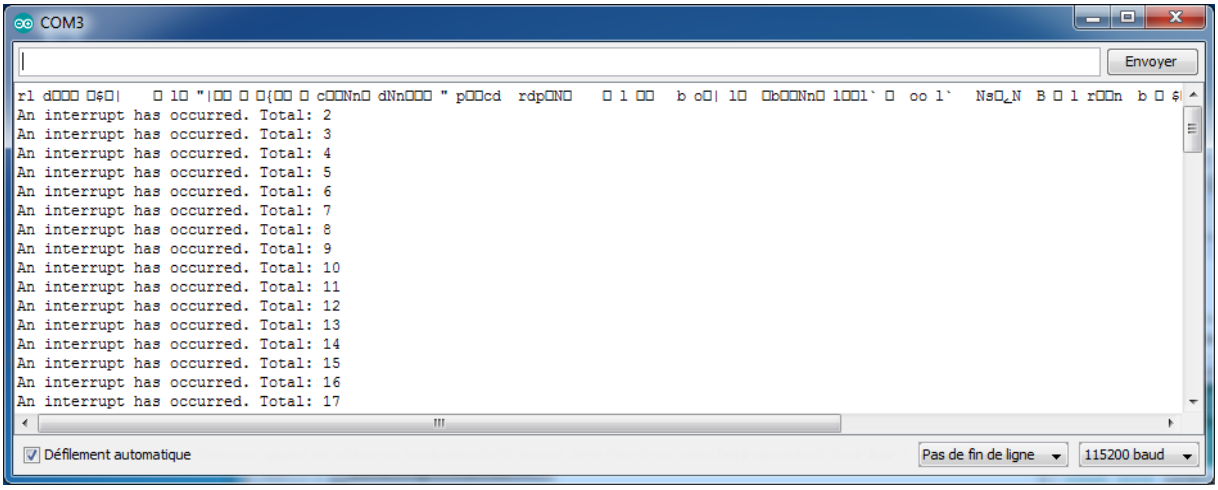
On va faire de la détection d'interruption, je vais encore me baser sur un tutoriel trouvé sur Internet https://techtutorialsx.com/2016/12/11/esp8266-external-interrupts/.

La seule chose que je vais changer c'est la broche pour l'interruption, je vais reprendre le montage du sujet sur la lecture d'un port GPIO et c'était la broche 5

Donc, le code sera

```
const byte interruptPin = 5;
volatile byte interruptCounter = 0;
int numberOfInterrupts = 0;

void setup() {
  Serial.begin(115200);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), handleInterrupt, FALLING);
}

void handleInterrupt() {
  interruptCounter++;
}

void loop() {
  if(interruptCounter>0){

      interruptCounter--;
      numberOfInterrupts++;

      Serial.print("An interrupt has occurred. Total: ");
      Serial.println(numberOfInterrupts);
  }
}
```

Quand on test, on se retrouve avec plusieurs d'interruptions à chaque changement d'état.



En fait, on détecte les rebonds de l'interrupteur. Plus l'interrupteur est de mauvaise qualité, plus il y en a.

Si on ne veut qu'une interruption, il faut soit faire un anti rebond matériel avec un filtre RC, soit un anti rebond logiciel avec une temporisation.

Le plus simple est d'ajouter un petit temps (ici 200ms) dans la gestion de l'interruption afin d'éviter d'en gérer de trop rapproché, cela enlèvera pas mal de parasite, mais pas tous, il faut souvent faire avec.

L'un des gros défauts, c'est que le programme ne devrait détecter que les interruptions de 1→ 0 (FALLING) mais qu'on détecte également l'autre sens à cause des rebonds.

```
#include <time.h>

const byte interruptPin = 5;
volatile byte interruptCounter = 0;
int numberOfInterrupts = 0;
int horloge;

void setup() {
  Serial.begin(115200);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), handleInterrupt, FALLING);
  horloge = millis();
}

void handleInterrupt() {
        if(millis() > horloge + 200) {
                interruptCounter++;
                horloge = millis();
        }
}

void loop() {
  if(interruptCounter>0){
      interruptCounter--;
      numberOfInterrupts++;

      Serial.print("An interrupt has occurred. Total: ");
      Serial.println(numberOfInterrupts);
  }
}
```

Bon, le principal, c'est qu'on détecte quelque chose. Le but sera de recevoir un mail (ou deux) si quelqu'un ouvre la porte par exemple.

10.4 Logiciel complet

On va maintenant mixer les deux précédents logiciels pour recevoir un mail à chaque interruption sur la broche GPIO5

10.5 Liste des programmes

https://github.com/montotof123/esp8266-12/blob/master/050_Mail_Sender/ESP8266_Gmail_Sender.ino

https://github.com/montotof123/esp8266-12/blob/master/050_Mail_Sender/Gsender.cpp

https://github.com/montotof123/esp8266-12/blob/master/050_Mail_Sender/Gsender.h

https://github.com/montotof123/esp8266-12/blob/master/060_Interrupt/sketch_may01d.ino

11 Envoi d’un Twitte au changement d’état d’un port

12 Envoyer un sms lors du changement d'état d’un port

13 Connection I2C

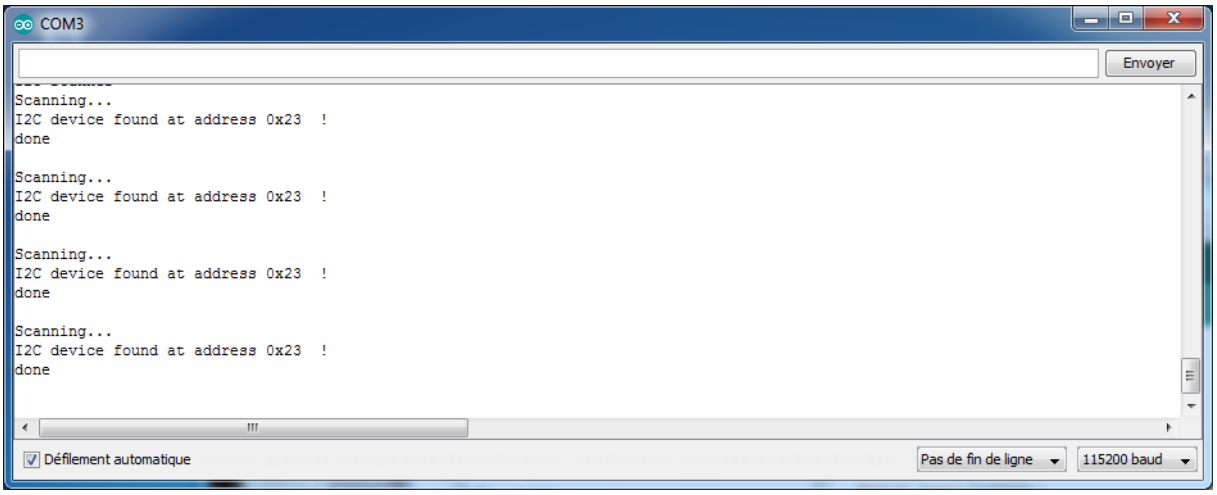
Les modules I2C doivent être câblés sur les broches SCL (GPIO5) et SDA (GPIO4).

Voici un petit programme trouvé à l'URL http://www.esp8266learning.com/i2c-scanner.php

Il permet de scanné le bus I2C pour trouver les adresses des appareils I2C y étant branchés. Il boucle toutes les 5 secondes.

```
#include <Wire.h>


void setup()
{
  Wire.begin();

  Serial.begin(115200);
  Serial.println("\nI2C Scanner");
}


void loop()
{
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {

    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknow error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);           // wait 5 seconds for next scan
}
```

Par exemple un module GY-30 câblé sur le port I2C indiquera



Cela prouve que le module est bien reconnu.

Liste des programmes

https://github.com/montotof123/esp8266-12/blob/master/070_ScanI2C/Scan.ino

14 Station météo

14.1 Mesure de la lumière ambiante

14.1.1 Programme en C

Dans un premier temps, on va faire fonctionner un module GY-30 afin de mesurer la lumière ambiante.

Le Datasheet du circuit de mesure se trouve à cette URL
http://www.mouser.com/ds/2/348/bh1750fvi-e-186247.pdf

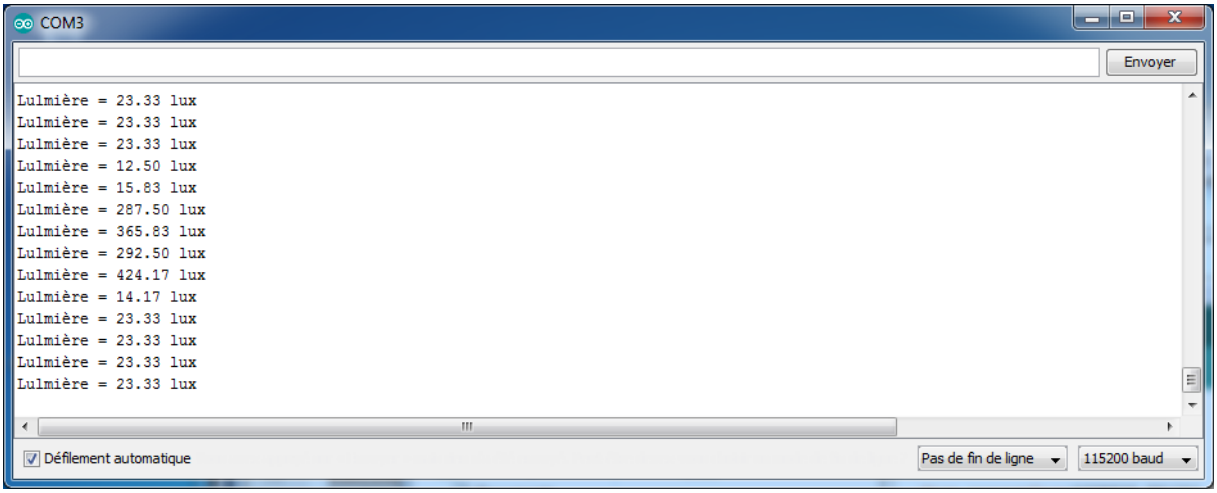
Ce module est accessible en I2C, il suffit donc de le câbler sur les broches SCL (GPIO5) et SDA (GPIO4). La broche ADD permet de brancher deux modules sur un même bus I2C. L'adresse sera 0x23 si cette broche est à la masse et 0x5C si elle est à 3.3V.

Le programme ci-dessous permet de tester le module afin qu'il envoie la mesure de la luminosité toutes les secondes.

```
#include <Wire.h>

#define ADDRESS_BH1750 0x23

#define BH1750_POWER_DOWN 0x00
#define BH1750_POWER_ON 0x01
#define BH1750_RESET 0x07
#define BH1750_CONTINUOUS_HIGH_RES_MODE 0x10
#define BH1750_CONTINUOUS_HIGH_RES_MODE_2 0x11
#define BH1750_CONTINUOUS_LOW_RES_MODE 0x13
#define BH1750_ONE_TIME_HIGH_RES_MODE 0x20
#define BH1750_ONE_TIME_HIGH_RES_MODE_2 0x21
#define BH1750_ONE_TIME_LOW_RES_MODE 0x23

union i2cData{
      unsigned short uSData;
      unsigned char uCData[2];
};

void setup() {
  Serial.begin(115200);  // Initialisation Terminal Série
  Wire.begin();         // Initialisation I2C

  /* Allumage et Reset du BH1750 */
  Wire.beginTransmission(ADDRESS_BH1750);
  Wire.write(BH1750_POWER_ON);
  Wire.endTransmission();
  Wire.beginTransmission(ADDRESS_BH1750);
  Wire.write(BH1750_RESET);
  Wire.endTransmission();
}

void loop() {
  i2cData data;
  float valeur;

  // Sensibilité standard (0100_0101)
  Wire.beginTransmission(ADDRESS_BH1750);
  Wire.write(0x42); // 01000_010
  Wire.endTransmission();
  Wire.beginTransmission(ADDRESS_BH1750);
  Wire.write(0x65); // 011_00101
  Wire.endTransmission();
```

```
  Wire.beginTransmission(ADDRESS_BH1750);
  Wire.write(BH1750_CONTINUOUS_HIGH_RES_MODE); // Demande d'une mesure
  Wire.endTransmission();

  delay(150); // Temps de la mesure

  Wire.requestFrom(ADDRESS_BH1750, 2); // Deux octets sont requis
  if (2 <= Wire.available()) {
    data.uCData[1] = Wire.read();  // Octet de poids fort
    data.uCData[0] = Wire.read();  // Octet de poids faible
  }

  valeur = (float)data.uSData / 1.2; // Calcul de la valeur

  // Affichage
  Serial.print("Lumière = ");
  Serial.print(valeur);
  Serial.println(" lux");

  delay(1000);  // Attendre 1 s avant de recommencer
}
```

Le résultat sera le suivant :



14.1.2 Programme en C++

L'IDE est également capable de compiler du C++.

On peut donc créer une classe qui gèrera le module avec un fichier d'entête en .h et un fichier implémentant la classe en .cpp. Ainsi qu'un programme pour le test.

Le fichier d'entête GestionBH1750.h

```
/*
 * GestionBH1750.h
 *
 *  Created on: 15 may 2017
 *      Author: totof
 * Controle un module BH1750
 */

#ifndef  GESTION BH1750 H
#define _GESTION_BH1750_H_

namespace std {

enum Adresse {
       ADDRESS_AD0_LOW = 0x23,
```

```
        ADDRESS_AD0_HIGH = 0x5C
};

enum ModeState {
        // No active state
        BH1750_POWER_DOWN = 0x00,
        // Wating for measurment command
        BH1750_POWER_ON = 0x01,
        // Reset data register value - not accepted in POWER_DOWN mode
        BH1750_RESET = 0x07
};

enum ModeMesure {
        // Start measurement at 1lx resolution. Measurement time is approx 120ms.
        BH1750_CONTINUOUS_HIGH_RES_MODE = 0x10,
        // Start measurement at 0.5lx resolution. Measurement time is approx 120ms.
        BH1750_CONTINUOUS_HIGH_RES_MODE_2 = 0x11,
        // Start measurement at 4lx resolution. Measurement time is approx 16ms.
        BH1750_CONTINUOUS_LOW_RES_MODE = 0x13,
        // Start measurement at 1lx resolution. Measurement time is approx 120ms.
        // Device is automatically set to Power Down after measurement.
        BH1750_ONE_TIME_HIGH_RES_MODE = 0x20,
        // Start measurement at 0.5lx resolution. Measurement time is approx 120ms.
        // Device is automatically set to Power Down after measurement.
        BH1750_ONE_TIME_HIGH_RES_MODE_2 = 0x21,
        // Start measurement at 1lx resolution. Measurement time is approx 120ms.
        // Device is automatically set to Power Down after measurement.
        BH1750_ONE_TIME_LOW_RES_MODE = 0x23
};

class GestionBH1750 {
    public:
                GestionBH1750(Adresse);

                void setState(ModeState);
                float readMinResolutionLightLevel(ModeMesure);
                float readLightLevel(ModeMesure);
                float readDoubleResolutionLightLevel(ModeMesure);
                float readMaxResolutionLightLevel(ModeMesure);

                virtual ~GestionBH1750();

    private:
                union i2cData{
                        unsigned short uSData;
                        unsigned char uCData[2];
                };
                Adresse adresse;
};

}

#endif /* _GESTION_BH1750_H_ */
```

La classe GestionBH1750.cpp

```
/*
 * GestionBH1750.cpp
 *
 *  Created on: 5 may 2017
 *      Author: totof
 * Controle un module BH1750
 */

#include <Wire.h>
#include <Arduino.h>
#include "GestionBH1750.h"

namespace std {

// *************************
// Constructeur
// @param adresse du circuit
// *************************
GestionBH1750::GestionBH1750(Adresse pAddress) {
	adresse = pAddress;
	Wire.begin();
	setState(BH1750_POWER_ON);
	setState(BH1750_RESET);
}

// ********************
// Activation d'un état
// @param l'état
// ********************
void GestionBH1750::setState(ModeState mode) {
	Wire.beginTransmission(adresse);
	Wire.write(mode);
	Wire.endTransmission();
}

// *************************
// Lecture de la mesure
// @param le mode de mesure
// @return la valeur mesuree
// *************************
float GestionBH1750::readMinResolutionLightLevel(ModeMesure mode) {
	i2cData data;
	float valeur;
	// Sensibilité max (0001_1111)
	Wire.beginTransmission(adresse);
	Wire.write(0x40); // 01000_000
	Wire.endTransmission();
	Wire.beginTransmission(adresse);
	Wire.write(0x7F); // 011_11111
	Wire.endTransmission();
	Wire.beginTransmission(adresse);
	Wire.write(mode);
	Wire.endTransmission();

	// Temps de mesure
	switch(mode) {
		case BH1750_CONTINUOUS_HIGH_RES_MODE_2:
		case BH1750 ONE TIME HIGH RES MODE 2:
		case BH1750 CONTINUOUS HIGH RES MODE:
		case BH1750 ONE TIME HIGH RES MODE: delay(70); break;
		case BH1750_CONTINUOUS_LOW_RES_MODE:
		case BH1750_ONE_TIME_LOW_RES_MODE: delay(8); break;
	}

	// Mesure
	Wire.requestFrom(adresse, 2); // Deux octets sont requis
	if (2 <= Wire.available()) {
		data.uCData[1] = Wire.read();  // Octet de poids fort
		data.uCData[0] = Wire.read();  // Octet de poids faible
	}

	// Mise à l'échelle
	switch(mode) {
		case BH1750_CONTINUOUS_HIGH_RES_MODE_2:
```

```
                case BH1750_ONE_TIME_HIGH_RES_MODE_2: valeur = (float)data.uSData / 1.08;
break;
                case BH1750 CONTINUOUS HIGH RES MODE:
                case BH1750 CONTINUOUS LOW RES MODE:
                case BH1750_ONE_TIME_HIGH_RES_MODE:
                case BH1750_ONE_TIME_LOW_RES_MODE: valeur = (float)data.uSData / 0.54; break;
        }
        return valeur;
}

// ************************
// Lecture de la mesure
// @param le mode de mesure
// @return la valeur mesuree
// ************************
float GestionBH1750::readLightLevel(ModeMesure mode) {
        i2cData data;
        float valeur;
        // Sensibilité standard (0100_0101)
        Wire.beginTransmission(adresse);
        Wire.write(0x42); // 01000_010
        Wire.endTransmission();
        Wire.beginTransmission(adresse);
        Wire.write(0x65); // 011_00101
        Wire.endTransmission();
        Wire.beginTransmission(adresse);
        Wire.write(mode);
        Wire.endTransmission();

        // Temps de mesure
        switch(mode) {
                case BH1750 CONTINUOUS HIGH RES MODE 2:
                case BH1750_ONE_TIME_HIGH_RES_MODE_2:
                case BH1750_CONTINUOUS_HIGH_RES_MODE:
                case BH1750_ONE_TIME_HIGH_RES_MODE: delay(150); break;
                case BH1750 CONTINUOUS LOW RES MODE:
                case BH1750_ONE_TIME_LOW_RES_MODE: delay(20); break;
        }

        // Mesure
        Wire.requestFrom(adresse, 2); // Deux octets sont requis
        if (2 <= Wire.available()) {
                data.uCData[1] = Wire.read();  // Octet de poids fort
                data.uCData[0] = Wire.read();  // Octet de poids faible
        }

        // Mise à l'échelle
        switch(mode) {
                case BH1750 CONTINUOUS HIGH RES MODE 2:
                case BH1750_ONE_TIME_HIGH_RES_MODE_2: valeur = (float)data.uSData / 2.4; break;
                case BH1750_CONTINUOUS_HIGH_RES_MODE:
                case BH1750_CONTINUOUS_LOW_RES_MODE:
                case BH1750 ONE TIME HIGH RES MODE:
                case BH1750_ONE_TIME_LOW_RES_MODE: valeur = (float)data.uSData / 1.2; break;
        }
        return valeur;
}

// ************************
// Lecture de la mesure
// @param le mode de mesure
// @return la valeur mesuree
// ************************
float GestionBH1750::readDoubleResolutionLightLevel(ModeMesure mode) {
        i2cData data;
        float valeur;
        // Sensibilité double (1000_1010)
        Wire.beginTransmission(adresse);
        Wire.write(0x44); // 01000_100
        Wire.endTransmission();
        Wire.beginTransmission(adresse);
        Wire.write(0x6A); // 011_01010
        Wire.endTransmission();
        Wire.beginTransmission(adresse);
        Wire.write(mode);
        Wire.endTransmission();
```

```
        // Temps de mesure
        switch(mode) {
                case BH1750_CONTINUOUS_HIGH_RES_MODE_2:
                case BH1750_ONE_TIME_HIGH_RES_MODE_2:
                case BH1750_CONTINUOUS_HIGH_RES_MODE:
                case BH1750_ONE_TIME_HIGH_RES_MODE: delay(300); break;
                case BH1750_CONTINUOUS_LOW_RES_MODE:
                case BH1750_ONE_TIME_LOW_RES_MODE: delay(40); break;
        }

        // Mesure
        Wire.requestFrom(adresse, 2); // Deux octets sont requis
        if (2 <= Wire.available()) {
                data.uCData[1] = Wire.read();  // Octet de poids fort
                data.uCData[0] = Wire.read();  // Octet de poids faible
        }

        // Mise à l'échelle
        switch(mode) {
                case BH1750_CONTINUOUS_HIGH_RES_MODE_2:
                case BH1750_ONE_TIME_HIGH_RES_MODE_2: valeur = (float)data.uSData / 4.8; break;
                case BH1750_CONTINUOUS_HIGH_RES_MODE:
                case BH1750_CONTINUOUS_LOW_RES_MODE:
                case BH1750_ONE_TIME_HIGH_RES_MODE:
                case BH1750_ONE_TIME_LOW_RES_MODE: valeur = (float)data.uSData / 2.4; break;
        }
        return valeur;
}

// ************************
// Lecture de la mesure
// @param le mode de mesure
// @return la valeur mesuree
// ************************
float GestionBH1750::readMaxResolutionLightLevel(ModeMesure mode) {
        i2cData data;
        float valeur;
        // Sensibilité max (1111 1110)
        Wire.beginTransmission(adresse);
        Wire.write(0x47); // 01000_111
        Wire.endTransmission();
        Wire.beginTransmission(adresse);
        Wire.write(0x7E); // 011_11110
        Wire.endTransmission();
        Wire.beginTransmission(adresse);
        Wire.write(mode);
        Wire.endTransmission();

        // Temps de mesure
        switch(mode) {
                case BH1750_CONTINUOUS_HIGH_RES_MODE_2:
                case BH1750_ONE_TIME_HIGH_RES_MODE_2:
                case BH1750_CONTINUOUS_HIGH_RES_MODE:
                case BH1750_ONE_TIME_HIGH_RES_MODE: delay(570); break;
                case BH1750_CONTINUOUS_LOW_RES_MODE:
                case BH1750_ONE_TIME_LOW_RES_MODE: delay(76); break;
        }

        // Mesure
        Wire.requestFrom(adresse, 2); // Deux octets sont requis
        if (2 <= Wire.available()) {
                data.uCData[1] = Wire.read();  // Octet de poids fort
                data.uCData[0] = Wire.read();  // Octet de poids faible
        }

                // Mise à l'échelle
        switch(mode) {
                case BH1750_CONTINUOUS_HIGH_RES_MODE_2:
                case BH1750_ONE_TIME_HIGH_RES_MODE_2: valeur = (float)data.uSData / 9.12;
break;
                case BH1750_CONTINUOUS_HIGH_RES_MODE:
                case BH1750_CONTINUOUS_LOW_RES_MODE:
                case BH1750_ONE_TIME_HIGH_RES_MODE:
                case BH1750_ONE_TIME_LOW_RES_MODE: valeur = (float)data.uSData / 4.56; break;
        }
        return valeur;
}
```

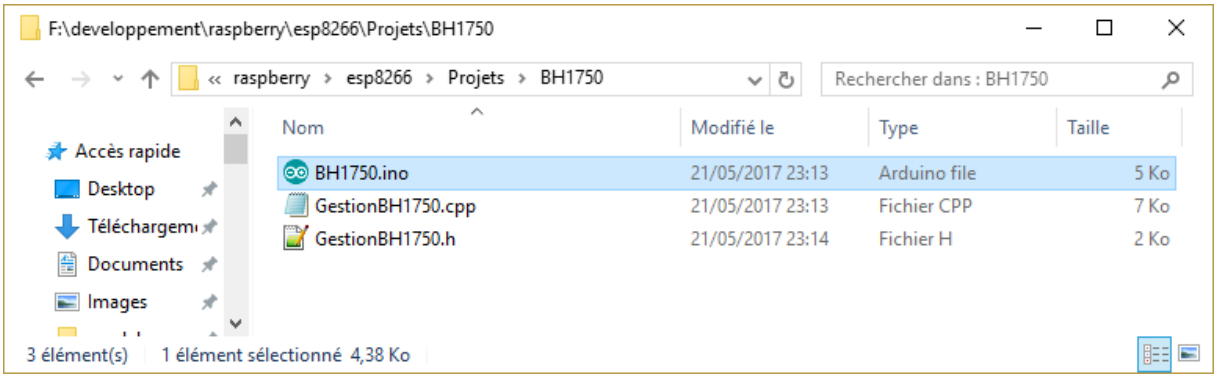
```
// ************************
// Destructeur
// ************************
GestionBH1750::~GestionBH1750() {
        setState(BH1750_RESET);
        setState(BH1750_POWER_DOWN);
}

}
```

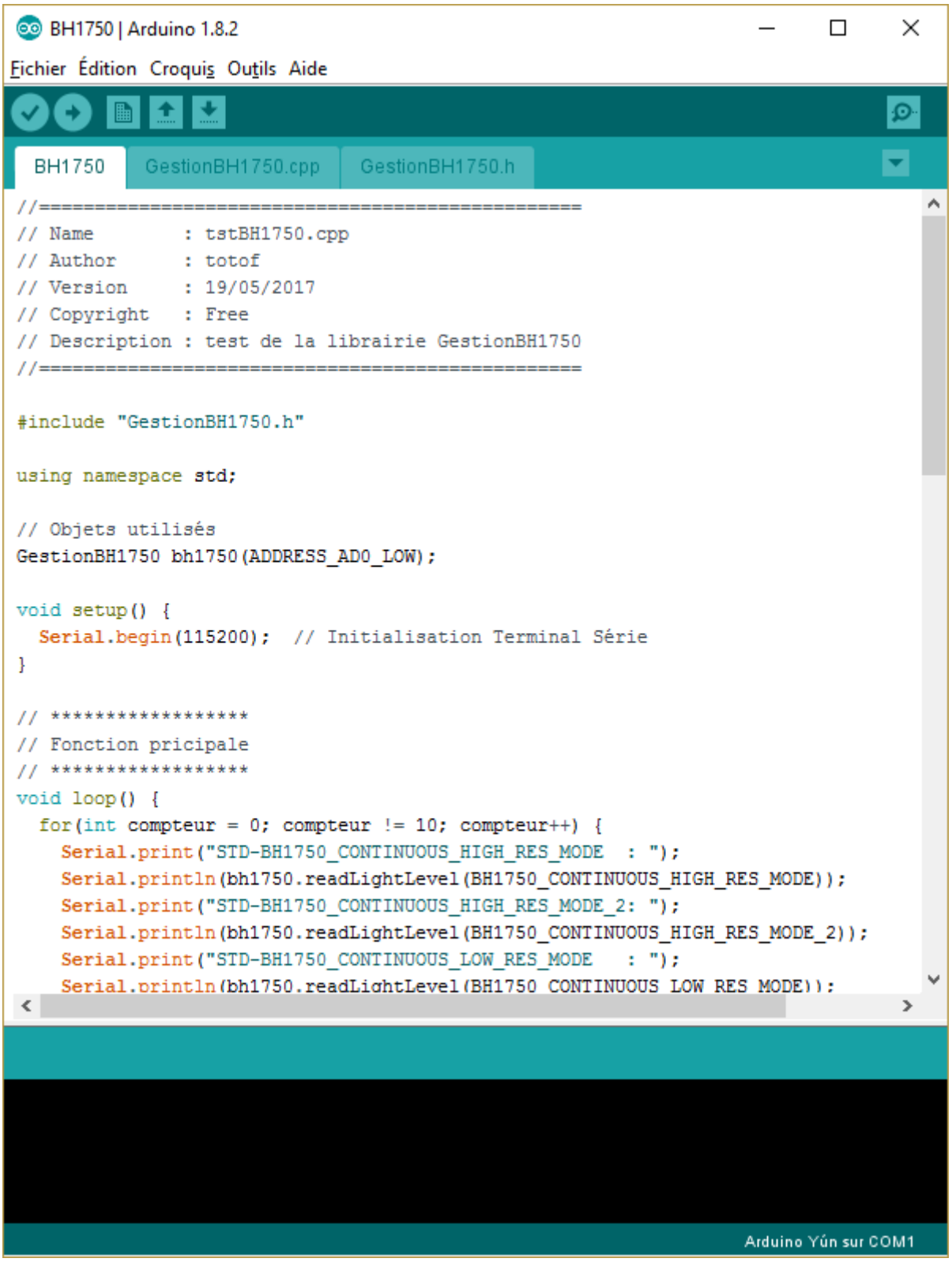
Le fichier de test BH1750.ino

```
//===============================================
// Name        : tstBH1750.cpp
// Author      : totof
// Version     : 19/05/2017
// Copyright   : Free
// Description : test de la librairie GestionBH1750
//===============================================

#include "GestionBH1750.h"

using namespace std;

// Objets utilisés
GestionBH1750 bh1750(ADDRESS_AD0_LOW);

void setup() {
  Serial.begin(115200);  // Initialisation Terminal Série
}

// ******************
// Fonction pricipale
// ******************
void loop() {
        for(int compteur = 0; compteur != 10; compteur++) {
                Serial.print("STD-BH1750_CONTINUOUS_HIGH_RES_MODE  : ");
    Serial.println(bh1750.readLightLevel(BH1750 CONTINUOUS HIGH RES MODE));
                Serial.print("STD-BH1750 CONTINUOUS HIGH RES MODE 2: ");
    Serial.println(bh1750.readLightLevel(BH1750_CONTINUOUS_HIGH_RES_MODE_2));
                Serial.print("STD-BH1750_CONTINUOUS_LOW_RES_MODE   : ");
    Serial.println(bh1750.readLightLevel(BH1750_CONTINUOUS_LOW_RES_MODE));
                Serial.print("STD-BH1750 ONE TIME HIGH RES MODE    : ");
    Serial.println(bh1750.readLightLevel(BH1750 ONE TIME HIGH RES MODE));
                Serial.print("STD-BH1750 ONE TIME HIGH RES MODE 2  : ");
    Serial.println(bh1750.readLightLevel(BH1750_ONE_TIME_HIGH_RES_MODE_2));
                Serial.print("STD-BH1750_ONE_TIME_LOW_RES_MODE     : ");
    Serial.println(bh1750.readLightLevel(BH1750_ONE_TIME_LOW_RES_MODE));
        }
        Serial.println("BH1750 RESET                        : ");
        bh1750.setState(BH1750 RESET);
        for(int compteur = 0; compteur != 10; compteur++) {
                Serial.print("DBL-BH1750_CONTINUOUS_HIGH_RES_MODE  : ");
    Serial.println(bh1750.readDoubleResolutionLightLevel(BH1750_CONTINUOUS_HIGH_RES_MODE));
                Serial.print("DBL-BH1750 CONTINUOUS HIGH RES MODE 2: ");
    Serial.println(bh1750.readDoubleResolutionLightLevel(BH1750 CONTINUOUS_HIGH_RES_MODE_2));
                Serial.print("DBL-BH1750 CONTINUOUS LOW RES MODE   : ");
    Serial.println(bh1750.readDoubleResolutionLightLevel(BH1750_CONTINUOUS_LOW_RES_MODE));
                Serial.print("DBL-BH1750_ONE_TIME_HIGH_RES_MODE    : ");
    Serial.println(bh1750.readDoubleResolutionLightLevel(BH1750_ONE_TIME_HIGH_RES_MODE));
                Serial.print("DBL-BH1750 ONE TIME HIGH RES MODE 2  : ");
    Serial.println(bh1750.readDoubleResolutionLightLevel(BH1750_ONE_TIME_HIGH_RES_MODE_2));
                Serial.print("DBL-BH1750_ONE_TIME_LOW_RES_MODE     : ");
    Serial.println(bh1750.readDoubleResolutionLightLevel(BH1750_ONE_TIME_LOW_RES_MODE));
        }
        Serial.println("BH1750 RESET                        : ");
        bh1750.setState(BH1750 RESET);
        for(int compteur = 0; compteur != 10; compteur++) {
                Serial.print("MAX-BH1750_CONTINUOUS_HIGH_RES_MODE  : ");
    Serial.println(bh1750.readMaxResolutionLightLevel(BH1750_CONTINUOUS_HIGH_RES_MODE));
                Serial.print("MAX-BH1750_CONTINUOUS_HIGH_RES_MODE_2: ");
    Serial.println(bh1750.readMaxResolutionLightLevel(BH1750 CONTINUOUS_HIGH_RES_MODE_2));
                Serial.print("MAX-BH1750_CONTINUOUS_LOW_RES_MODE   : ");
```

```
    Serial.println(bh1750.readMaxResolutionLightLevel(BH1750_CONTINUOUS_LOW_RES_MODE));
              Serial.print("MAX-BH1750_ONE_TIME_HIGH_RES_MODE    : ");
    Serial.println(bh1750.readMaxResolutionLightLevel(BH1750 ONE TIME HIGH_RES_MODE));
              Serial.print("MAX-BH1750 ONE TIME HIGH RES MODE 2  : ");
    Serial.println(bh1750.readMaxResolutionLightLevel(BH1750_ONE_TIME_HIGH_RES_MODE_2));
              Serial.print("MAX-BH1750_ONE_TIME_LOW_RES_MODE     : ");
    Serial.println(bh1750.readMaxResolutionLightLevel(BH1750_ONE_TIME_LOW_RES_MODE));
        }
        Serial.println("BH1750 RESET                        : ");
        bh1750.setState(BH1750 RESET);
        for(int compteur = 0; compteur != 10; compteur++) {
              Serial.print("MIN-BH1750_CONTINUOUS_HIGH_RES_MODE  : ");
    Serial.println(bh1750.readMinResolutionLightLevel(BH1750_CONTINUOUS_HIGH_RES_MODE));
              Serial.print("MIN-BH1750 CONTINUOUS HIGH RES MODE 2: ");
    Serial.println(bh1750.readMinResolutionLightLevel(BH1750 CONTINUOUS_HIGH_RES_MODE_2));
              Serial.print("MIN-BH1750 CONTINUOUS LOW RES MODE   : ");
    Serial.println(bh1750.readMinResolutionLightLevel(BH1750_CONTINUOUS_LOW_RES_MODE));
              Serial.print("MIN-BH1750_ONE_TIME_HIGH_RES_MODE    : ");
    Serial.println(bh1750.readMinResolutionLightLevel(BH1750_ONE_TIME_HIGH_RES_MODE));
              Serial.print("MIN-BH1750 ONE TIME HIGH RES MODE 2  : ");
    Serial.println(bh1750.readMinResolutionLightLevel(BH1750 ONE TIME HIGH_RES_MODE_2));
              Serial.print("MIN-BH1750_ONE_TIME_LOW_RES_MODE     : ");
    Serial.println(bh1750.readMinResolutionLightLevel(BH1750_ONE_TIME_LOW_RES_MODE));
        }
}
```

Il faut mettre les trois fichiers dans un répertoire ayant le même nom que le fichier de test (BH1750). L'IDE chargera les trois fichiers et les compilera ensemble.





14.1.3 Liste des programmes

https://github.com/montotof123/esp8266-12/blob/master/080_BH1750C/BH1750.ino

https://github.com/montotof123/esp8266-12/blob/master/090_BH1750CPP/BH1750.ino

https://github.com/montotof123/esp8266-12/blob/master/090_BH1750CPP/GestionBH1750.cpp

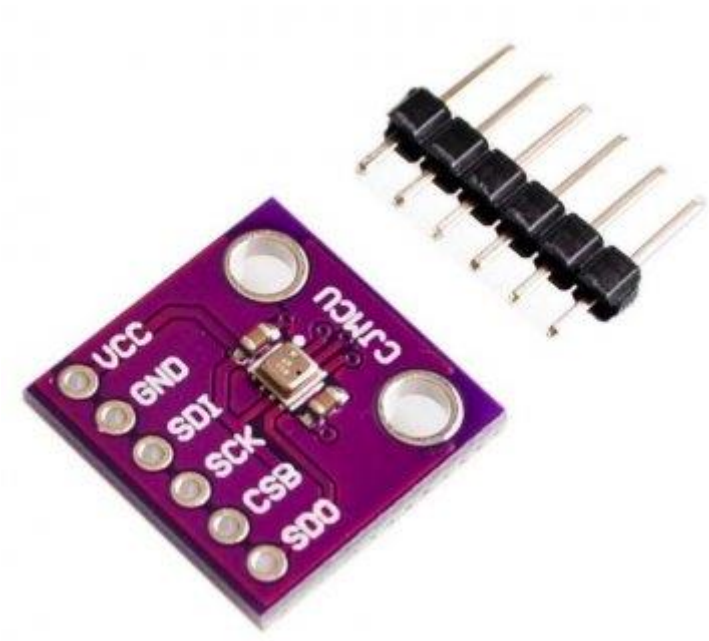
https://github.com/montotof123/esp8266-12/blob/master/090_BH1750CPP/GestionBH1750.h

14.2 Mesure de la température, de l'humidité et de la pression avec un capteur BME280

Le BME280 est un circuit développé par Bosch, il concentre dans un même mini boitier un thermomètre, un mesureur de pression atmosphérique et un mesureur d'humidité. Il est surtout utilisé dans les stations météo intérieures et extérieures, des petits altimètres, l'automatisation de la ventilation ou de l'air conditionné, le sport…

Il peut mesurer la température entre -40 et 85° Celsius avec une précision d'un degré, la pression entre 300hPa et 1100hPa à +/- 1hPa et l'humidité entre 0 et 100%.

On peut le trouver comme d'habitude sur EBay pour moins de 5€.



Il peut se connecter en SPI ou en I2C, accepte une tension jusqu'à 3.6V et consomme très peu de courant. Bosch fourni une documentation très complète https://www.bosch-sensortec.com/bst/products/all_products/bme280 avec Datasheet et même une librairie en C pour l'utilisation de l'appareil.

Les connections seront les suivantes :

Pour le SPI

BME280	ESP8266
VCC	3.3V
GND	Masse
SDI	SPI MOSI (GPIO 12)
SCK	SPI SCLK (GPIO 14)
CSB	SS (GPIO 15)
SDO	SPI MISO (GPIO 13)

Pour l'I2C

BME280	ESP8266
VCC	3.3V

GND	Masse
SDI	SDA (GPIO 4)
SCK	SCL (GPIO 5)
CSB	3.3V
SDO	Masse : adresse 0x76 ou 3.3V : adresse 0x77

Le positionnement entre I2C et SPI se fait au démarrage par la broche CSB. Si CSB est à 0 au démarrage, c'est le SPI, si CSB est à 1 ou en l'air, c'est l'I2C.

14.2.1 Programme en C++ pour le SPI

Je vais utiliser la librairie Adafruit_BME280 qui se trouve à l'URL
https://github.com/adafruit/Adafruit_BME280_Library

Il faut mettre dans le même répertoire les fichiers suivants :

Adafruit_BME280.h, Adafruit_BME280.cpp, bme280test.ino et la librairie Adafruit_Sensor.h

Et faire quelques petites modifications :
Sur le fichier Adafruit_BME280.h changer la ligne `#include <Adafruit_Sensor.h>` par
`#include "Adafruit_Sensor.h"`

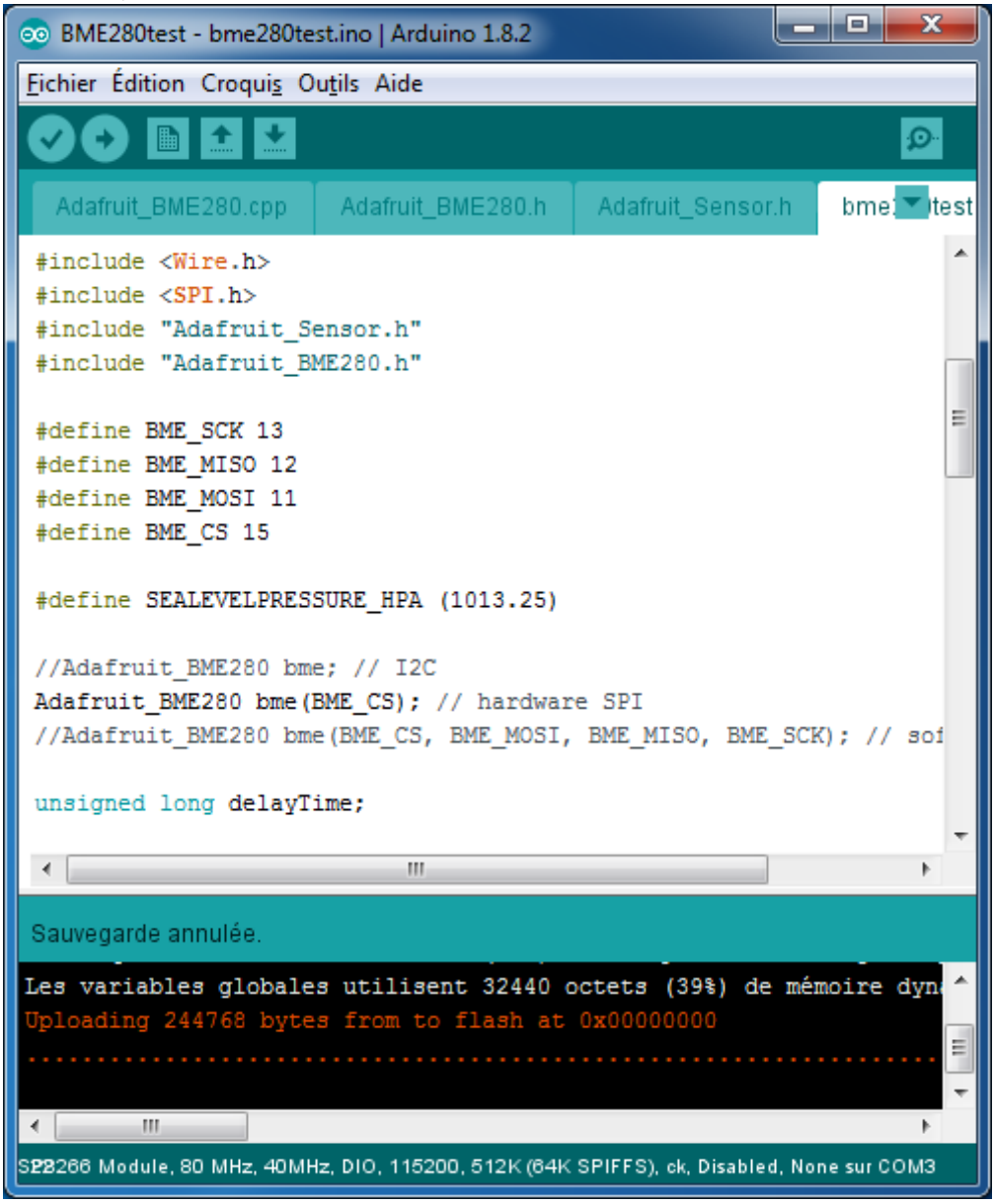
```
/***************************************************************************
*
  This is a library for the BME280 humidity, temperature & pressure sensor
  Designed specifically to work with the Adafruit BME280 Breakout
  ----> http://www.adafruit.com/products/2650
  These sensors use I2C or SPI to communicate, 2 or 4 pins are required
  to interface.
  Adafruit invests time and resources providing this open source code,
  please support Adafruit andopen-source hardware by purchasing products
  from Adafruit!
  Written by Limor Fried & Kevin Townsend for Adafruit Industries.
  BSD license, all text above must be included in any redistribution

***************************************************************************
/
#ifndef __BME280_H__
#define __BME280_H__

#if (ARDUINO >= 100)
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif

#include "Adafruit_Sensor.h"
#include <Wire.h>

/*=========================================================================
    I2C ADDRESS/BITS
    -----------------------------------------------------------------------
*/
    #define BME280_ADDRESS                (0x77)
/*=========================================================================
*/

/*=========================================================================
```

```
    REGISTERS
    ------------------------------------------------------------------
*/
```

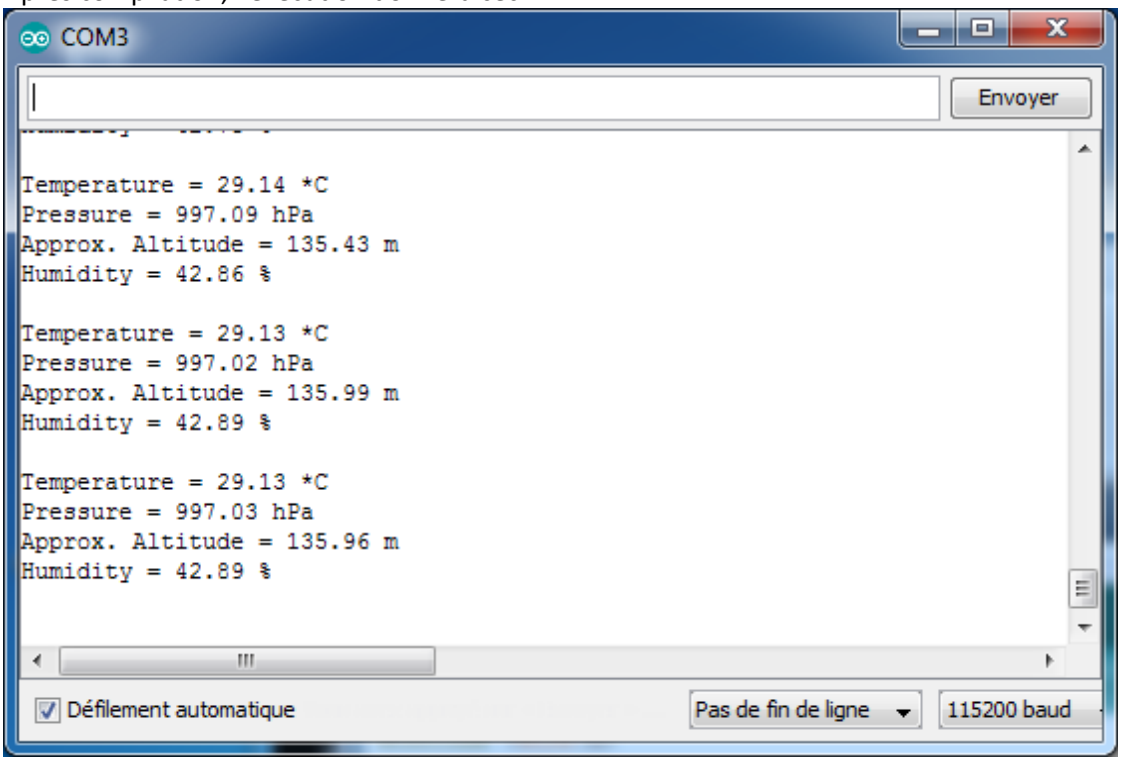
Sur le fichier bme280test.ino changer la ligne `#include <Adafruit_Sensor.h>` par `#include "Adafruit_Sensor.h",` mettre la ligne `#define BME_CS 15` au lieu de `10,` commentez le constructeur I2C pour prendre le hardware SPI

```
/**************************************************************************
*
  This is a library for the BME280 humidity, temperature & pressure sensor
  Designed specifically to work with the Adafruit BME280 Breakout
  ----> http://www.adafruit.com/products/2650
  These sensors use I2C or SPI to communicate, 2 or 4 pins are required
  to interface. The device's I2C address is either 0x76 or 0x77.
  Adafruit invests time and resources providing this open source code,
  please support Adafruit andopen-source hardware by purchasing products
  from Adafruit!
  Written by Limor Fried & Kevin Townsend for Adafruit Industries.
  BSD license, all text above must be included in any redistribution

**************************************************************************
/

#include <Wire.h>
#include <SPI.h>
#include "Adafruit_Sensor.h"
#include "Adafruit_BME280.h"

#define BME_SCK 13
#define BME_MISO 12
#define BME_MOSI 11
#define BME_CS 15

#define SEALEVELPRESSURE_HPA (1013.25)

//Adafruit_BME280 bme; // I2C
Adafruit_BME280 bme(BME_CS); // hardware SPI
```

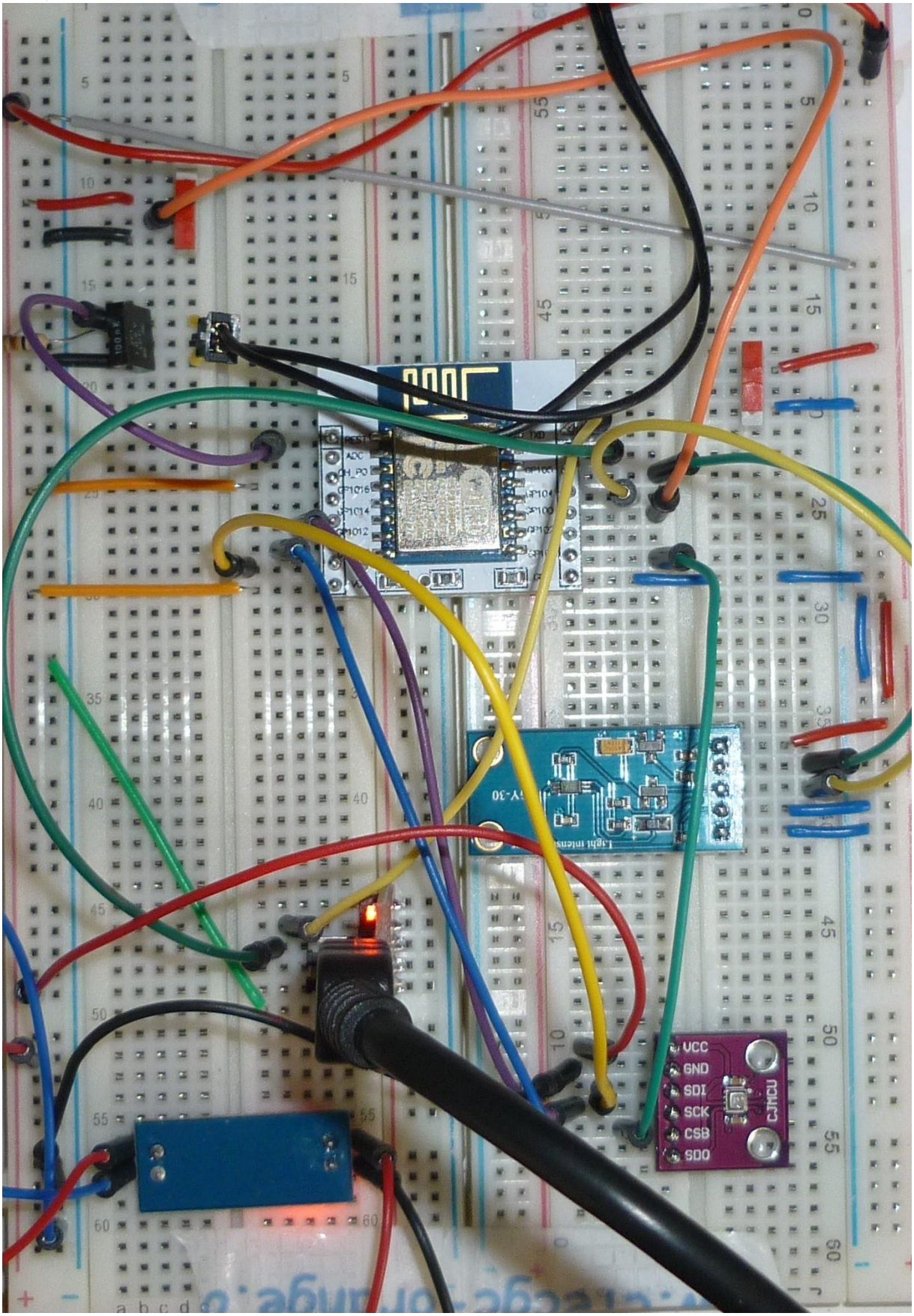
Dans l'IDE, cela donnera ceci



Après compilation, l'exécution donnera ceci



Le montage avec le mesureur de lumière, le module BME280, un régulateur 3.3V, le convertisseur série/USB et l'ESP8266



Et les programmes complets


Adafruit_BME280.cpp


Adafruit_BME280.h


Adafruit_Sensor.h


bme280test.ino

14.2.2 Liste des programmes

https://github.com/montotof123/esp8266-12/blob/master/100_BME280Spi/Adafruit_BME280.cpp

https://github.com/montotof123/esp8266-12/blob/master/100_BME280Spi/Adafruit_BME280.h

https://github.com/montotof123/esp8266-12/blob/master/100_BME280Spi/Adafruit_Sensor.h

https://github.com/montotof123/esp8266-12/blob/master/100_BME280Spi/bme280test.ino

14.2.3 Programme en C++ pour l'I2C

Il faut mettre dans le même répertoire les fichiers suivants :

Adafruit_BME280.h, Adafruit_BME280.cpp, bme280test.ino et la librairie Adafruit_Sensor.h

Et faire une petite modification :
Vérifiez qu'il y a la bonne adresse dans le fichier Adafruit_BME280.h 0x76 ou 0x77
Sur le fichier bme280test.ino commentez le constructeur SPI et activer l'I2C

```
/**************************************************************************
*
  This is a library for the BME280 humidity, temperature & pressure sensor
  Designed specifically to work with the Adafruit BME280 Breakout
  ----> http://www.adafruit.com/products/2650
  These sensors use I2C or SPI to communicate, 2 or 4 pins are required
  to interface. The device's I2C address is either 0x76 or 0x77.
  Adafruit invests time and resources providing this open source code,
  please support Adafruit andopen-source hardware by purchasing products
  from Adafruit!
  Written by Limor Fried & Kevin Townsend for Adafruit Industries.
  BSD license, all text above must be included in any redistribution

**************************************************************************
/

#include <Wire.h>
#include <SPI.h>
#include "Adafruit_Sensor.h"
#include "Adafruit_BME280.h"

#define BME_SCK 13
#define BME_MISO 12
#define BME_MOSI 11
#define BME_CS 15

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C
//Adafruit_BME280 bme(BME_CS); // hardware SPI
//Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // software SPI
```

14.2.4 Liste des programmes

https://github.com/montotof123/esp8266-12/blob/master/110_BME280I2c/Adafruit_BME280.cpp

https://github.com/montotof123/esp8266-12/blob/master/110_BME280I2c/Adafruit_BME280.h

https://github.com/montotof123/esp8266-12/blob/master/110_BME280I2c/Adafruit_Sensor.h

https://github.com/montotof123/esp8266-12/blob/master/110_BME280I2c/bme280test.ino

15 Télémètre à ultrason

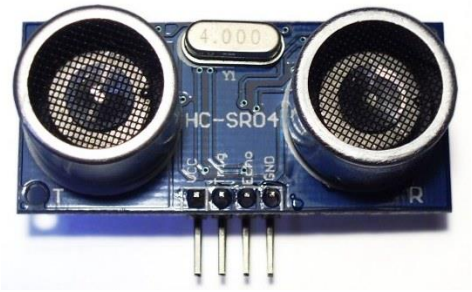
15.1 La théorie

Le but de ce montage sera de fournir un télémètre à ultrason avec un affichage de la distance.

En plus de l'ESP8266 qui fera la coordination, les modules suivants seront utilisés :

- Le télémètre HC-SR04 :

Le datasheet est bien sûr sur Internet, https://www.gotronic.fr/pj2-hc-sr04-utilisation-avec-picaxe-1343.pdf. On voit que le module doit être alimenté en 5V, et que deux broches Trigger et Echo permettent la demande et la lecture de la mesure. La demande de mesure consiste dans une impulsion d'au moins 10µs sur la broche Trigger, le résultat étant fourni par la broche Echo qui restera au niveau haut le temps que le signal revienne au module. La distance parcourue sera donc le temps de retour du signal multiplié par la vitesse du son et divisé par deux car le signal à fait un aller-retour.

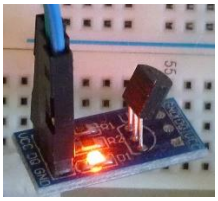


- Un thermomètre DS18B20 :

Ce thermomètre permettra de compenser la vitesse du son en fonction de la température. En effet, plus il fait chaud plus le son se déplace rapidement.

Influence de la température sur l'air

θ en °C	c en $m \cdot s^{-1}$	ρ en $kg \cdot m^{-3}$	Z en $N \cdot s \cdot m^{-3}$
- 10	325,4	1,341	436,5
- 5	328,5	1,316	432,4
0	331,5	1,293	428,3
+ 5	334,5	1,269	424,5
+ 10	337,5	1,247	420,7
+ 15	340,5	1,225	417,0
+ 20	343,4	1,204	413,5
+ 25	346,3	1,184	410,0
+ 30	349,2	1,164	406,6



- Des afficheurs HP5082-7300.

http://www.s100computers.com/My%20System%20Pages/ZFDC%20Board/HP5082-73xx.pdf

Ces afficheurs ont un buffer intégré et comporte une mémoire.

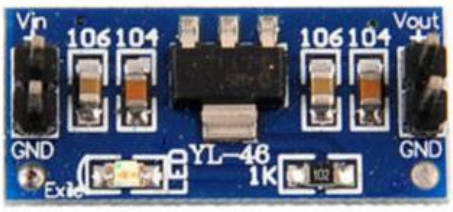
- MCP23S17

Afin d'augmenter le nombre de sortie qui ne sont pas assez importantes pour le montage, les afficheurs seront pilotés par un extendeur d'entrées-sorties MCP23S17 adressable en SPI

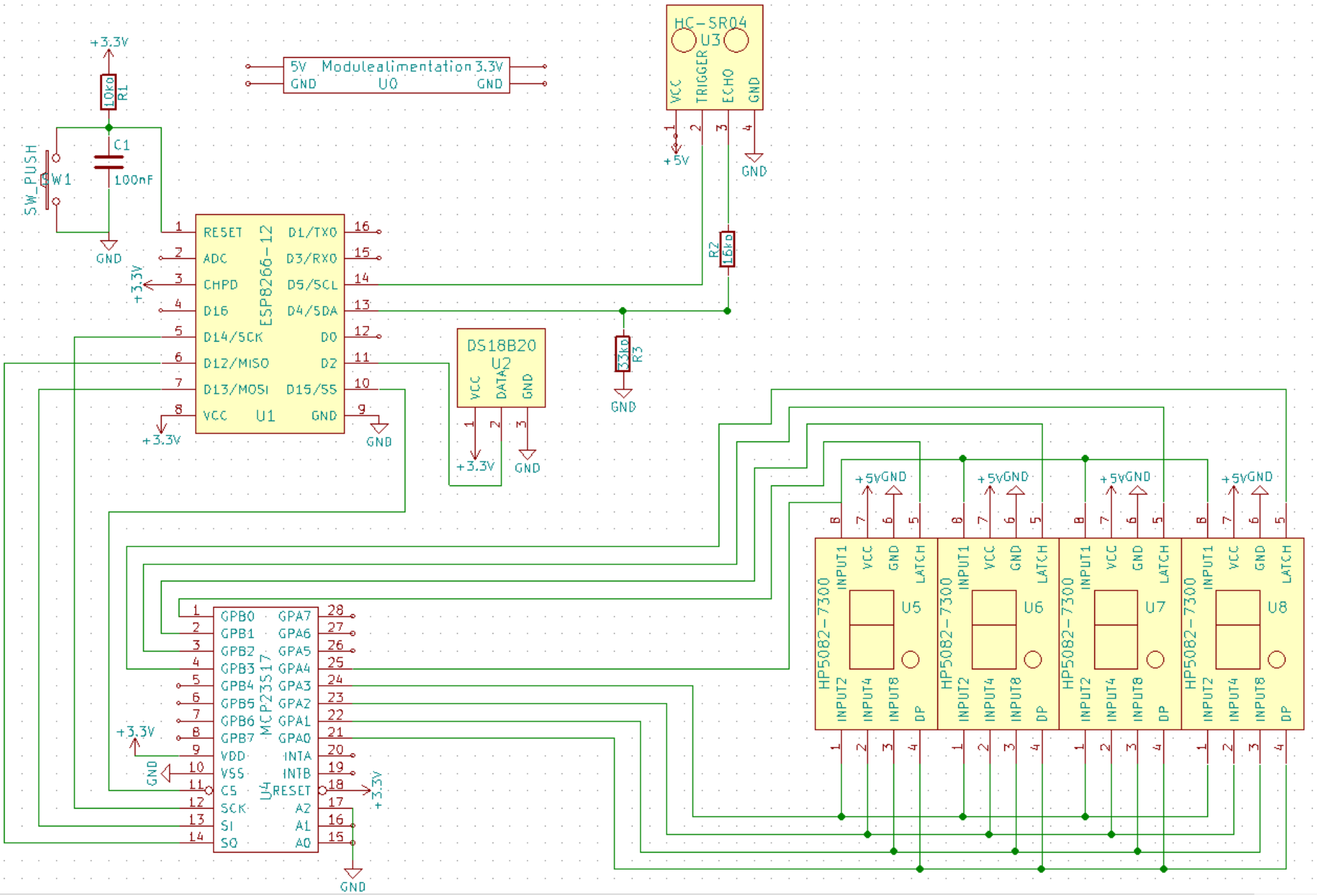
http://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf

- Alimentation 3.3V

L'alimentation générale sera en 5V, par une batterie extérieure de téléphone par exemple et un module AMS1117-3.3V permettra de fournir du 3.3V pour les circuits alimentés avec cette tension.



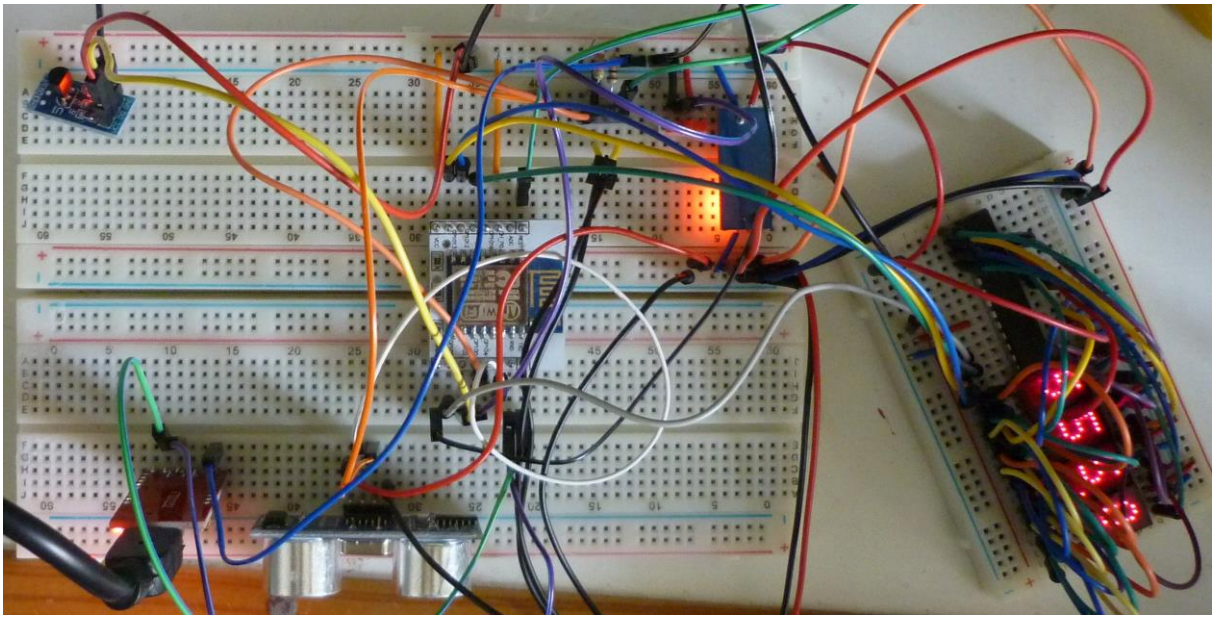
15.2 L’électronique



Il faut faire très attention aux alimentations, l'ESP8266, le MCP23S17 et le DS18B20 sont alimentés en 3.3V, le HC-SR04 et les afficheurs en 5V. Le pont diviseur R2/R3 sert à faire chuter la tension de sortie du HC-SR04 de 5V à 3.3V. J'utilise des résistances de 16kΩ et 33kΩ, mais toutes les combinaisons faisant chuter la tension d'1/3 sont utilisables dans la limite de quelques kΩ à quelques dizaines de kΩ.

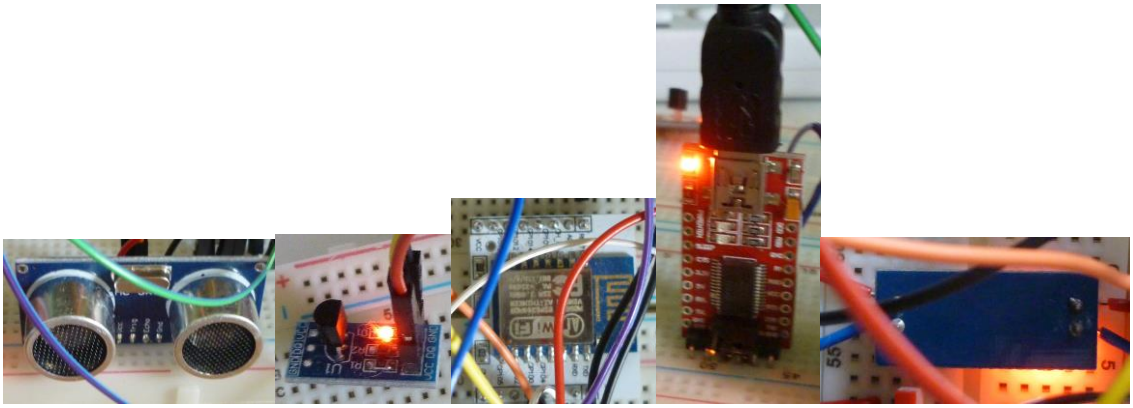
Quelques photos du prototype :

Le montage complet

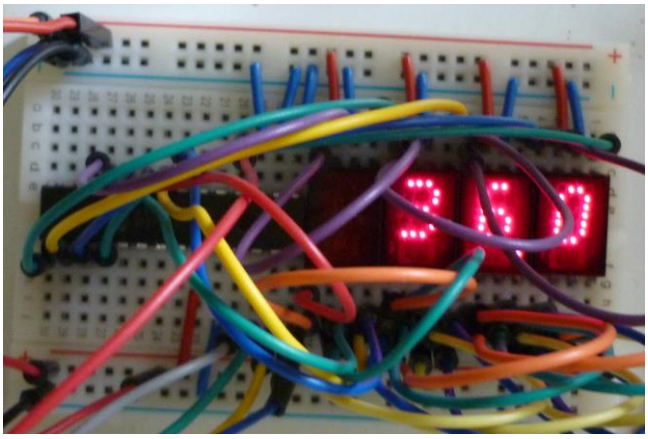


Les modules principaux :

Dans l'ordre le HC-SC04, le DS18B20, l'ESP8266, le FTD1232 et le module AMS1117-3.3V (il aurait été judicieux de mettre les picots de ce module dans l'autre sens afin de pouvoir lire facilement les indications)



L’afficheur avec le MCP23S17



15.3 Le logiciel

Le logiciel sera constitué de plusieurs librairies permettant de faire fonctionner chaque module et d'un programme général assurant l'interfaçage des modules et le fonctionnement global du télémètre.

15.3.1 Librairie pour le DS18B20

Ce thermomètre est connecté en OneWire. Il nécessite deux librairies pour fonctionner. https://github.com/PaulStoffregen/OneWire/archive/master.zip et https://github.com/milesburton/Arduino-Temperature-Control-Library/archive/master.zip

L'installation est décrite ici http://randomnerdtutorials.com/esp8266-ds18b20-temperature-sensor-web-server-with-arduino-ide/

Ci-dessous, un petit programme de test qui affichera en boucle la température d'un DS18B20 branché sur le GPIO2.

```
// Including DS18B20 librairies
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into pin D1 on the ESP8266 12-E - GPIO 2
#define ONE_WIRE_BUS 2

// Setup a oneWire instance to communicate with any OneWire devices
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature DS18B20(&oneWire);

// only runs once on boot
void setup() {
        // Initializing serial port for debugging purposes
        Serial.begin(115200);
        delay(10);
        DS18B20.begin(); // IC Default 9 bit.
}

// runs over and over again
void loop() {
        DS18B20.requestTemperatures();
        float tempC  = DS18B20.getTempCByIndex(0);
        char temperatureCString[6];
        dtostrf(tempC, 2, 2, temperatureCString);
        Serial.print("Temperature celcius = ");
        Serial.print(temperatureCString);
        Serial.println("°");
        delay(1000);
}
```

15.3.2 Librairie pour le HC-SR04

La librairie pour le module permettra de positionner la température pour corriger les mesures et donnera bien sûr la distance mesurée. La distance est donnée en mètre et la température doit être donnée en degrés Celsius. Si aucune valeur de température n'est donnée, la température sera positionnée à 20° par défaut.

```
/*
 * GestionHCSR04.h
 *
 *  Created on: 30/07/2017
 *      Author: totof
 */

#ifndef GESTIONHCSR04_H
#define GESTIONHCSR04_H_

// Classe de gestion des HC-SR04
```

```
class GestionHCSR04 {
public:
        GestionHCSR04(int, int);
        void setTemperature(float);
        float getTemperature(void);
        float getDistance(void);
        virtual ~GestionHCSR04();

private:
        int trigger;
        int echo;
        float temperature = 20.0;
};

#endif /* GESTIONHCSR04_H_ */
```

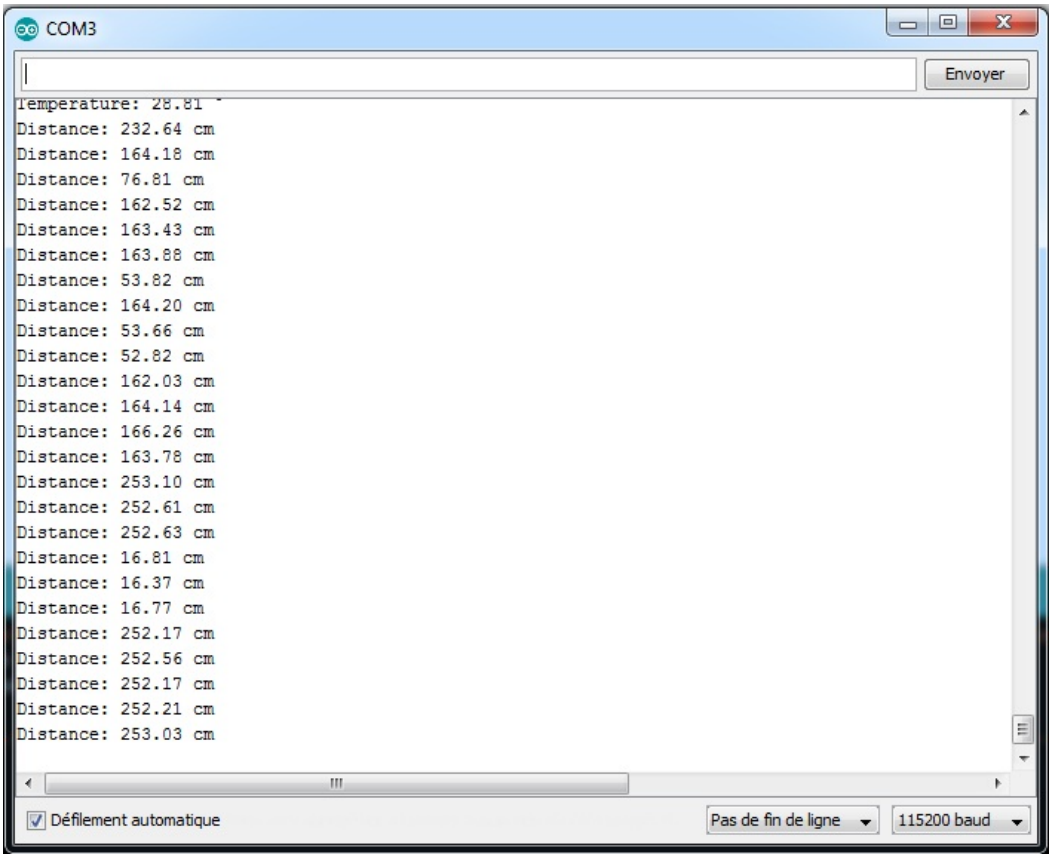
```
/*
 * GestionHCSR04.cpp
 *
 *  Created on: 14 août 2017
 *      Author: totof
 */

#include "GestionHCSR04.h"
#include <Arduino.h>

// ************************************
// Constructeur
// Mémorise le numéro des pins utilisées
// @param numéro de pin du trigger
// @param numéro de pin de l'echo
// ************************************
GestionHCSR04::GestionHCSR04(int pinTrigger, int pinEcho) {
        trigger = pinTrigger;
        echo = pinEcho;
        pinMode(trigger, OUTPUT);
        pinMode(echo, INPUT);
        digitalWrite(trigger, LOW);
}

// ************************************************************
// Positionne la temperature pour calcul de la vitesse du son
// ************************************************************
void GestionHCSR04::setTemperature(float pTemperature) {
        temperature = pTemperature;
}

// ********************************
// Donne la temperature positionnée
// ********************************
float GestionHCSR04::getTemperature(void) {
        return temperature;
}

// ***************************
// Renvoie la distance mesurée
// ***************************
float GestionHCSR04::getDistance(void) {
        long duration;
        digitalWrite(trigger, LOW);
        delayMicroseconds(2);

        digitalWrite(trigger, HIGH);
        delayMicroseconds(10);

        digitalWrite(trigger, LOW);
        duration = pulseIn(echo, HIGH);

        // Calcul de la distance
        float vitesse = 0.62 * temperature + 331.6;
        return vitesse * (float)duration / 2000000.0;
}

// ************************************************************
// Destructeur
```

```
// ********************************************************
GestionHCSR04::~GestionHCSR04() {
	pinMode(trigger, INPUT);
	pinMode(echo, INPUT);
	digitalWrite(trigger, LOW);
	digitalWrite(echo, LOW);
}
```

Ainsi qu'un petit programme de test qui utilise la librairie

```
/***********************
  Test HC-SR04
  Created on: 14/08/2017
  Author: totof
***********************/

#define TRIGGER 5
#define ECHO    4
#define ONE_WIRE_BUS 2

#include <OneWire.h>
#include <DallasTemperature.h>
#include "GestionHCSR04.h"

GestionHCSR04 mesureDistance(TRIGGER, ECHO);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);

void setup() {
    Serial.begin (115200);
    Serial.println("Init");
    DS18B20.begin();
    DS18B20.requestTemperatures();
    mesureDistance.setTemperature(DS18B20.getTempCByIndex(0));
    char temperatureString[6];
    dtostrf(mesureDistance.getTemperature(), 2, 2, temperatureString);
    Serial.print("Temperature: ");
    Serial.print(temperatureString);
    Serial.println(" °");
}

void loop() {
    float distanceCM = mesureDistance.getDistance() * 100.0;
    char distanceString[6];
    dtostrf(distanceCM, 2, 2, distanceString);
    Serial.print("Distance: ");
    Serial.print(distanceString);
    Serial.println(" cm");
    delay(1000);
}
```

Mettre les trois fichiers dans un répertoire et compiler.



15.3.3 Librairie pour le MCP23S17

Une librairie est disponible à l'emplacement suivant :
https://github.com/n0mjs710/MCP23S17/tree/master/MCP23S17

Elle fonctionne, mais comporte certains défauts.

Je l'ai modifié en ajoutant un constructeur vide et une méthode init afin qu'elle puisse être utilisée dans d'autres librairies.

De plus, il faut faire attention en utilisant les méthodes positionnant un seul bit d'un port car elles positionnent les autres bits du port à 0, leur utilisation est donc problématique. Par contre les méthodes positionnant un registre entier voir les 16 ports d'un coup fonctionnent très bien.

15.3.4 Librairie pour le HP5082-7300

Le fonctionnement de l'affichage sera composé de deux librairies. Une gérant un seul afficheur et une gérant l'affichage entier. Pour simplifier la librairie, les ports de donnée sur le GPIOA sont positionnés en dur, seul le port de verrouillage (latch) est modifiable mais doit obligatoirement être sur le GPIOB.

Voici un programme de test pour vérifier le fonctionnement des afficheurs. Le MCP23S17 doit être à l'adresse 0 et le CS sur la broche 15. Le câblage devant être identique au schéma du montage.

```
//===================================================
// Name        : tstAffichage.cpp
// Author      : totof
// Version     :
// Copyright   : Free
// Description : test de la librairie GestionAffichage
//===================================================

#include <stdlib.h>   // La librairie standard
#include "GestionAffichage.h"
```

```
GestionAffichage affichage(0, 15);

void setup() {
	Serial.begin (115200);
	Serial.println("Test du HP Affichage");
}

void loop() {
	Serial.println("9999, 0");
	affichage.affiche(9999, 0);
	delay(1000);

	Serial.println("999, 0");
	affichage.affiche(999, 0);
	delay(1000);

	Serial.println("99, 0");
	affichage.affiche(99, 0);
	delay(1000);

	Serial.println("9, 0");
	affichage.affiche(9, 0);
	delay(1000);

	Serial.println("999, 1");
	affichage.affiche(999, 1);
	delay(1000);

	Serial.println("99, 1");
	affichage.affiche(99, 1);
	delay(1000);

	Serial.println("9, 1");
	affichage.affiche(9, 1);
	delay(1000);

	Serial.println("99, 2");
	affichage.affiche(99, 2);
	delay(1000);

	Serial.println("9, 2");
	affichage.affiche(9, 2);
	delay(1000);

	Serial.println("9, 3");
	affichage.affiche(9, 3);
	delay(1000);

	Serial.println("9999, 0");
	affichage.affiche(9999, 0);
	delay(1000);

	Serial.println("7777, 0");
	affichage.affiche(7777, 0);
	delay(1000);

	Serial.println("5555, 0");
	affichage.affiche(5555, 0);
	delay(1000);

	Serial.println("333.3, 1");
	affichage.affiche(333.3, 1);
	delay(1000);

	Serial.println("11.11, 2");
	affichage.affiche(11.11, 2);
	delay(1000);

	Serial.println("0.011, 3");
	affichage.affiche(0.011, 3);
	delay(1000);

	Serial.println("-999, 0");
	affichage.affiche(-999, 0);
	delay(1000);

	Serial.println("-99, 0");
```

```
        affichage.affiche(-99, 0);
        delay(1000);

        Serial.println("-9, 0");
        affichage.affiche(-9, 0);
        delay(1000);

        Serial.println("-99, 1");
        affichage.affiche(-99, 1);
        delay(1000);

        Serial.println("-9, 1");
        affichage.affiche(-9, 1);
        delay(1000);

        Serial.println("-9, 2");
        affichage.affiche(-9, 2);
        delay(1000);

        Serial.println("-999, 0");
        affichage.affiche(-999, 0);
        delay(1000);

        Serial.println("-777, 0");
        affichage.affiche(-777, 0);
        delay(1000);

        Serial.println("-444, 0");
        affichage.affiche(-444, 0);
        delay(1000);

        Serial.println("-99.9, 1");
        affichage.affiche(-99.9, 1);
        delay(1000);

        Serial.println("-6.66, 2");
        affichage.affiche(-6.66, 2);
        delay(1000);

        Serial.println("-0.22, 3");
        affichage.affiche(-0.22, 3);
        delay(1000);

        Serial.println("-12.3, 1");
        affichage.affiche(-12.3, 1);
        delay(1000);

        Serial.println("-999, 1");
        affichage.affiche(-999, 1);
        delay(1000);

        Serial.println("9999, 1");
        affichage.affiche(9999, 1);
        delay(1000);

        Serial.println("-999, 2");
        affichage.affiche(-999, 2);
        delay(1000);

        Serial.println("9999, 2");
        affichage.affiche(9999, 2);
        delay(1000);

        Serial.println("-999, 3");
        affichage.affiche(-999, 3);
        delay(1000);

        Serial.println("9999, 3");
        affichage.affiche(9999, 3);
        delay(1000);

        Serial.println("0, 3");
        affichage.affiche(0, 3);
        delay(1000);

        Serial.println("50, 2");
        affichage.affiche(50, 2);
```

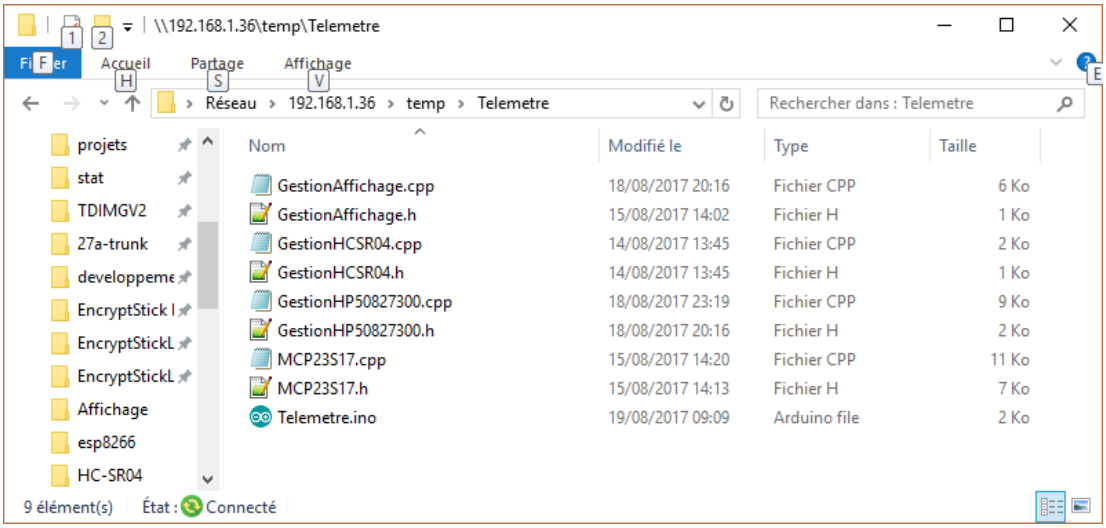
```
        delay(1000);

        Serial.println("-0, 3");
        affichage.affiche(-0, 3);
        delay(1000);

        Serial.println("1, 1");
        affichage.affiche(1, 1);
        delay(1000);

        Serial.println("10, 1");
        affichage.affiche(10, 1);
        delay(1000);

        Serial.println("-1, 1");
        affichage.affiche(-1, 1);
        delay(1000);

        Serial.println("1.234567, 3");
        affichage.affiche(1.234567, 3);
        delay(1000);

        Serial.println("1.234567, 2");
        affichage.affiche(1.234567, 2);
        delay(1000);

        Serial.println("1.234567, 1");
        affichage.affiche(1.234567, 1);
        delay(1000);

        Serial.println("1.234567, 0");
        affichage.affiche(1.234567, 0);
        delay(1000);

        Serial.println("-1.234567, 3");
        affichage.affiche(-1.234567, 3);
        delay(1000);

        Serial.println("-1.234567, 2");
        affichage.affiche(-1.234567, 2);
        delay(1000);

        Serial.println("-1.234567, 1");
        affichage.affiche(-1.234567, 1);
        delay(1000);

        Serial.println("-1.234567, 0");
        affichage.affiche(-1.234567, 0);
        delay(1000);

        Serial.println("1.034567, 3");
        affichage.affiche(1.034567, 3);
        delay(1000);

        Serial.println("1.204567, 3");
        affichage.affiche(1.204567, 3);
        delay(1000);

        Serial.println("1.230567, 3");
        affichage.affiche(1.230567, 3);
        delay(1000);

        Serial.println("1.204567, 2");
        affichage.affiche(1.204567, 2);
        delay(1000);

        Serial.println("-1.034567, 1");
        affichage.affiche(-1.034567, 1);
        delay(1000);

        Serial.println("-1.034567, 0");
        affichage.affiche(-1.034567, 0);
        delay(1000);

        Serial.println("-0.034567, 3");
        affichage.affiche(-0.034567, 3);
        delay(1000);
```

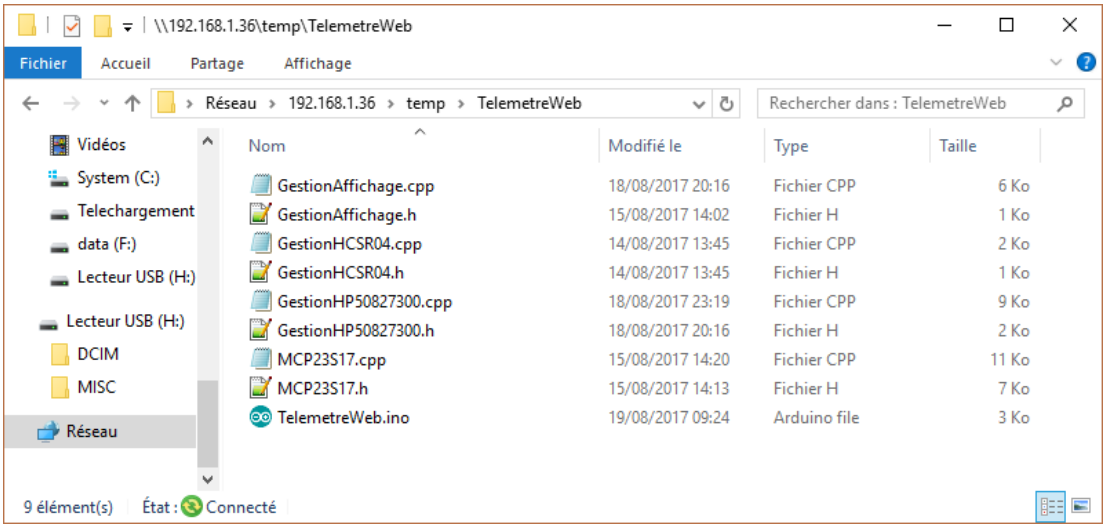
```
        Serial.println("-0.204567, 3");
        affichage.affiche(-0.204567, 3);
        delay(1000);

        Serial.println("-0.230567, 3");
        affichage.affiche(-0.230567, 3);
        delay(1000);

        Serial.println("-0.034567, 2");
        affichage.affiche(-0.034567, 2);
        delay(1000);

        Serial.println("-0.204567, 2");
        affichage.affiche(-0.204567, 2);
        delay(1000);

        Serial.println("-0.030567, 1");
        affichage.affiche(-0.030567, 1);
        delay(1000);

        Serial.println("-0.030567, 0");
        affichage.affiche(-0.030567, 0);
        delay(1000);

        Serial.println("10000, 0");
        affichage.affiche(10000, 0);
        delay(1000);

        Serial.println("-1000, 1");
        affichage.affiche(-1000, 1);
        delay(1000);

        Serial.println("-9, 4");
        affichage.affiche(-9, 4);
        delay(1000);

        Serial.println("50, -1");
        affichage.affiche(50, -1);
        delay(1000);
}
```

15.3.5 Programme général

Deux programmes sont proposés, l'un ne faisant qu'afficher les résultats et l'autre affichant les résultats et les mettant également à disposition sur un serveur Web (/thermometre et /telemetre).

Pour la compilation avec l'IDE, il faut bien sûr mettre tous les fichiers dans un même répertoire, ce répertoire devant être nommé comme le programme principal.





Le programme telemetre.ino

```
//========================
// Name        : telemetre
// Author      : totof
// Version     :
// Copyright   : Free
// Description : telemetre
//========================

#define TRIGGER 5
#define ECHO    4
#define ONE_WIRE_BUS 2
#define ADRESSE 0
#define PIN_CS 15

#include <OneWire.h>
#include <DallasTemperature.h>
#include "GestionHCSR04.h"
#include "GestionAffichage.h"
```

```
GestionHCSR04 mesureDistance(TRIGGER, ECHO);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
GestionAffichage affichage(ADRESSE, PIN_CS);

void setup() {
        Serial.begin (115200);
        Serial.println("Télémesure");
        DS18B20.begin();
        DS18B20.requestTemperatures();
        mesureDistance.setTemperature(DS18B20.getTempCByIndex(0));
        affichage.affiche(DS18B20.getTempCByIndex(0), 2);
        delay(5000);
        affichage.eteindre();
        delay(2000);
}

void loop() {
        // Mesure distance en cm et affichage
        float distanceCM = mesureDistance.getDistance() * 100.0;
        affichage.affiche(distanceCM, 1);

        // distance a l'ecran
        char distanceString[6];
        dtostrf(distanceCM, 2, 2, distanceString);
        Serial.println(distanceString);

        delay(1000);
}
```

Le programme TelemetreWeb

```
//============================
// Name        : telemetre Web
// Author      : totof
// Version     :
// Copyright   : Free
// Description : telemetre Web
//============================

#define TRIGGER 5
#define ECHO    4
#define ONE WIRE BUS 2
#define ADRESSE 0
#define PIN_CS 15

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

#include <OneWire.h>
#include <DallasTemperature.h>
#include "GestionHCSR04.h"
#include "GestionAffichage.h"

GestionHCSR04 mesureDistance(TRIGGER, ECHO);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
GestionAffichage affichage(ADRESSE, PIN_CS);

const char* ssid = "dlink-4212"; // remplacer par le SSID de votre WiFi
const char* password = "nhyyi53680"; // remplacer par le mot de passe de votre WiFi
ESP8266WebServer server(80); // on instancie un serveur ecoutant sur le port 80

void setup() {
        // Initialisation serial
        Serial.begin (115200);
        Serial.println("Télémesure");

         // on demande la connexion au WiFi
        WiFi.begin(ssid, password);
        Serial.println("");

        // on attend d'etre connecte au WiFi avant de continuer
        while (WiFi.status() != WL_CONNECTED) {
```

```
                delay(500);
                Serial.print(".");
        }

        // on affiche l'adresse IP qui nous a ete attribuee
        Serial.println("");
        Serial.print("IP address: ");
        Serial.println(WiFi.localIP());

        // Mesure et affichage temperature pour correction
        DS18B20.begin();
        DS18B20.requestTemperatures();
        mesureDistance.setTemperature(DS18B20.getTempCByIndex(0));
        affichage.affiche(DS18B20.getTempCByIndex(0), 2);
        delay(5000);
        affichage.eteindre();
        delay(2000);

        // Lire le thermometre
        server.on("/thermometre", []() {
                float temperature = DS18B20.getTempCByIndex(0);
                char temperatureString[6];
                dtostrf(temperature, 2, 2, temperatureString);
                server.send(200, "text/plain", temperatureString);
        });

        // Lire le telemetre
        server.on("/telemetre", []() {
                float mesureCM = mesureDistance.getDistance() * 100.0;
                char mesureCMString[6];
                dtostrf(mesureCM, 2, 2, mesureCMString);
                server.send(200, "text/plain", mesureCMString);
        });

        // on commence a ecouter les requetes venant de l'exterieur
        server.begin();
}

void loop() {
        // Mesure distance en cm et affichage
        float distanceCM = mesureDistance.getDistance() * 100.0;
        affichage.affiche(distanceCM, 1);

        // distance a l'ecran
        char distanceString[6];
        dtostrf(distanceCM, 2, 2, distanceString);
        Serial.println(distanceString);

        // a chaque iteration, on appelle handleClient pour que les requetes soient traitees
        server.handleClient();

        delay(1000);
}
```

15.4 Appareil autonome

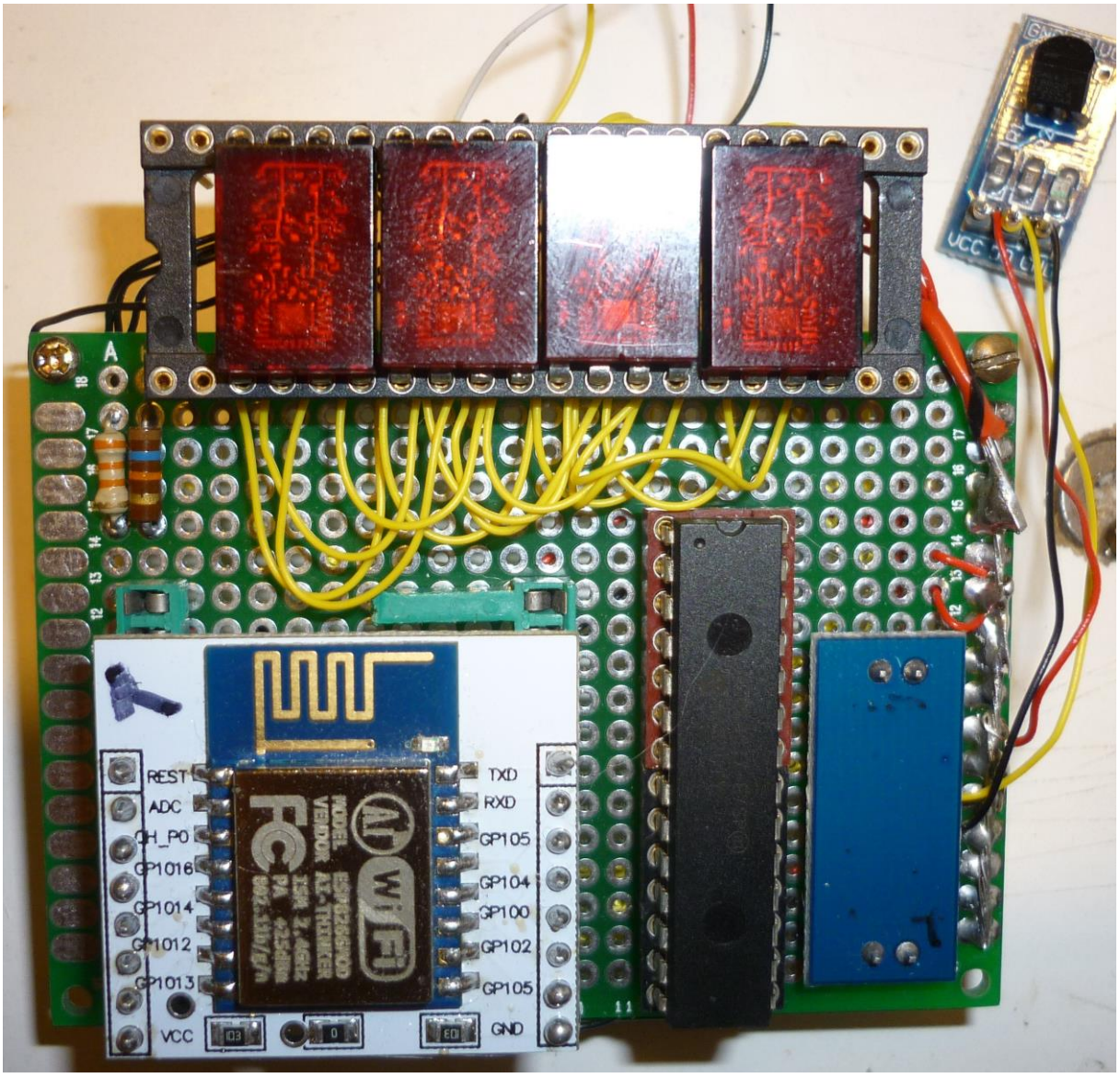
Pour que l'appareil soit vraiment utilisable, il faut l'intégrer dans un boitier avec possibilité de brancher une alimentation externe. Cela permettra d'en faire un vrai appareil de mesure autonome pour le bricolage ou toute autre activité demandant des mesures de distance précises.

15.4.1 Electronique et câblage

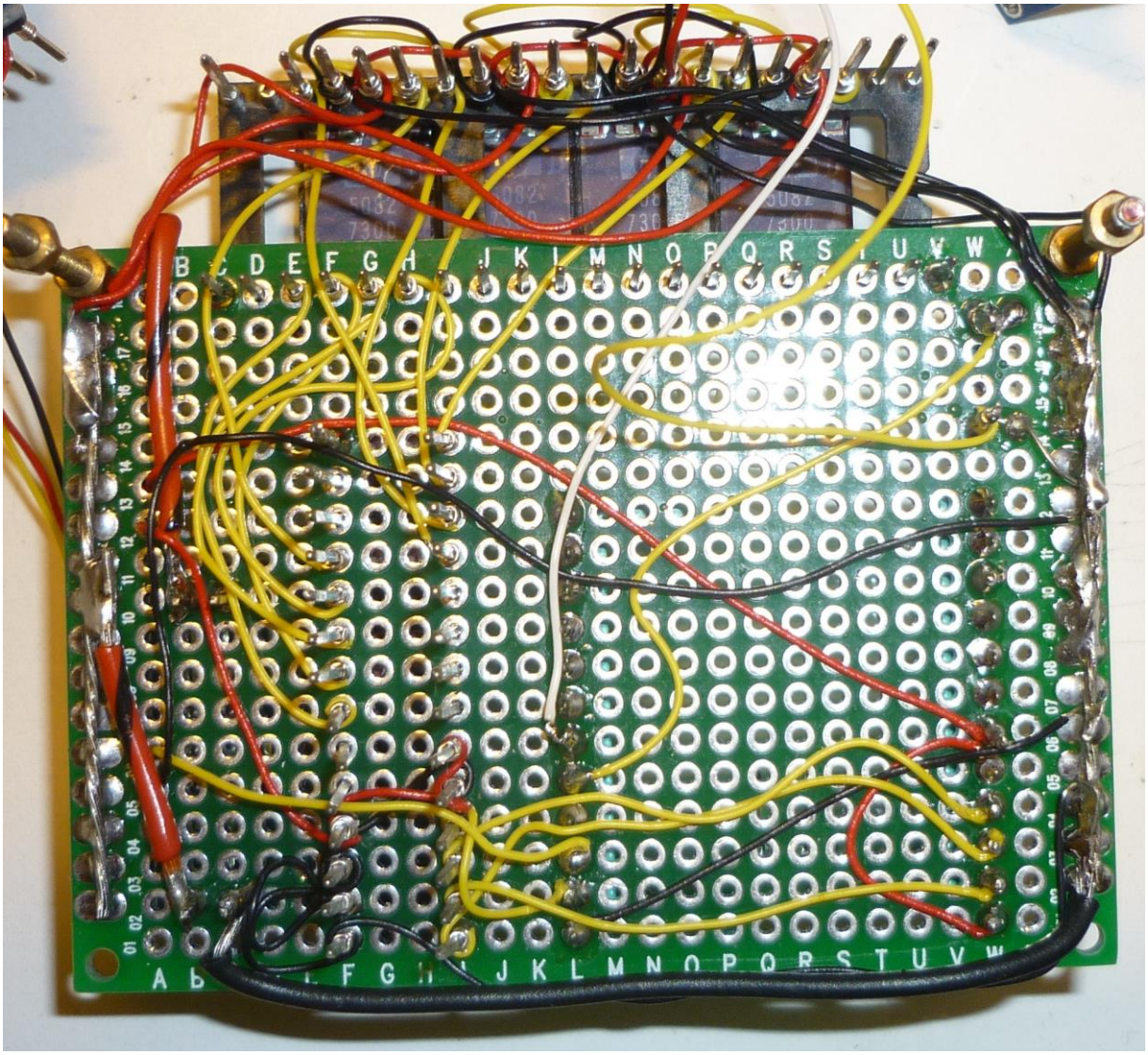
Pour l'électronique, il n'est bien sûr plus question d'utiliser les plaques de câblage et les fils volants. Le câblage sera entièrement refait sur une plaquette de 7 * 5 cm avec un savant mélange de soudure et de wrapping. Les afficheurs seront enficher sur un support de circuit intégré 40 broches. Le télémètre et le thermomètre seront câblés également en wrapping avec des fils assez long pout être positionnés sur le boitier. Le télémètre devant être positionné sans obstacle pour pouvoir effectuer les mesures et le thermomètre étant positionné à l'extérieur du boitier pour mesurer la température extérieure et non celle à l'intérieure du boitier.

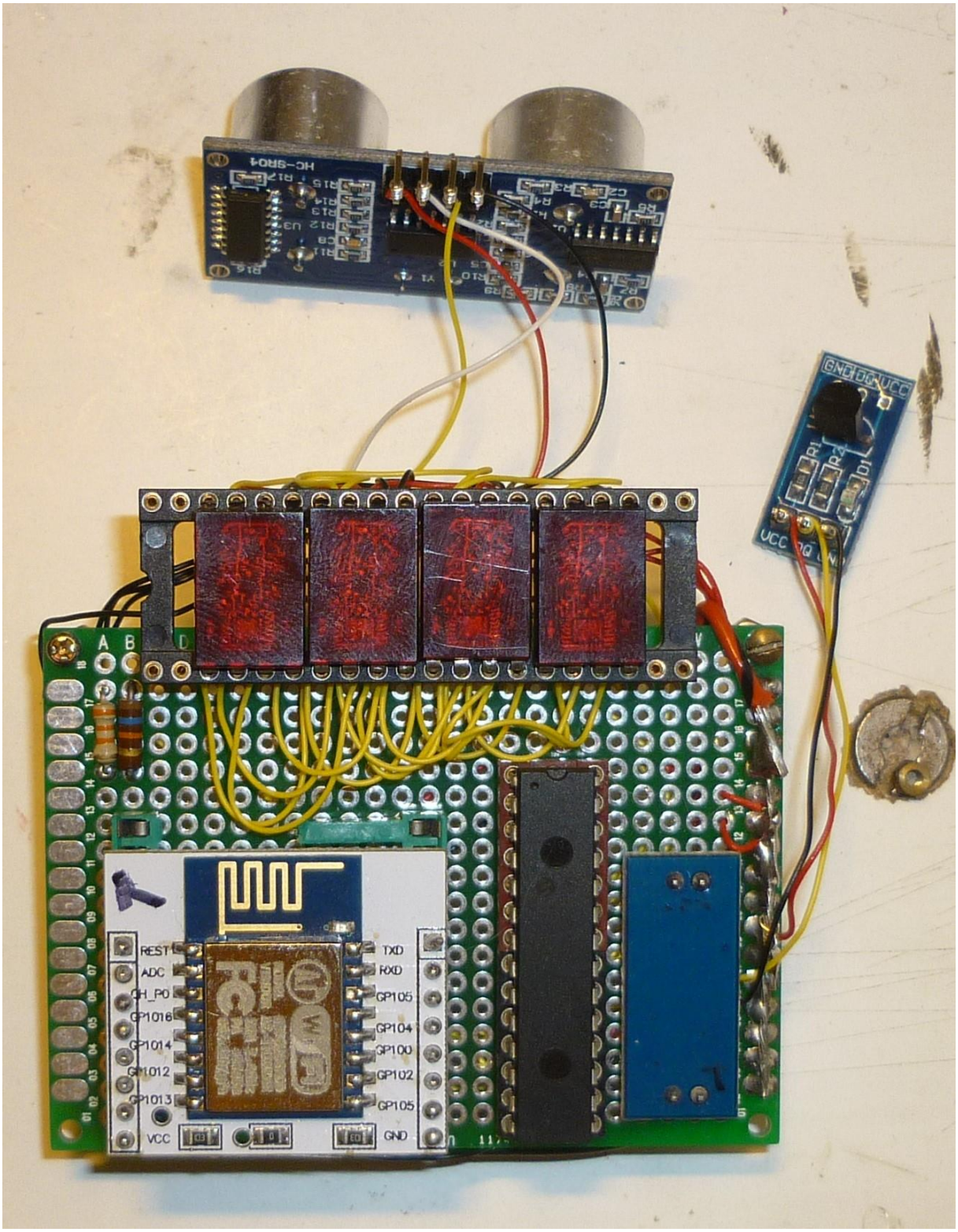
L'ESP8266 et le MCP23S17 sont également positionné sur des supports de circuit intégré.

L'ESP8266 sera bien sûr programmé avant d'être positionné sur son support.



Le câblage en wrapping et soudures.



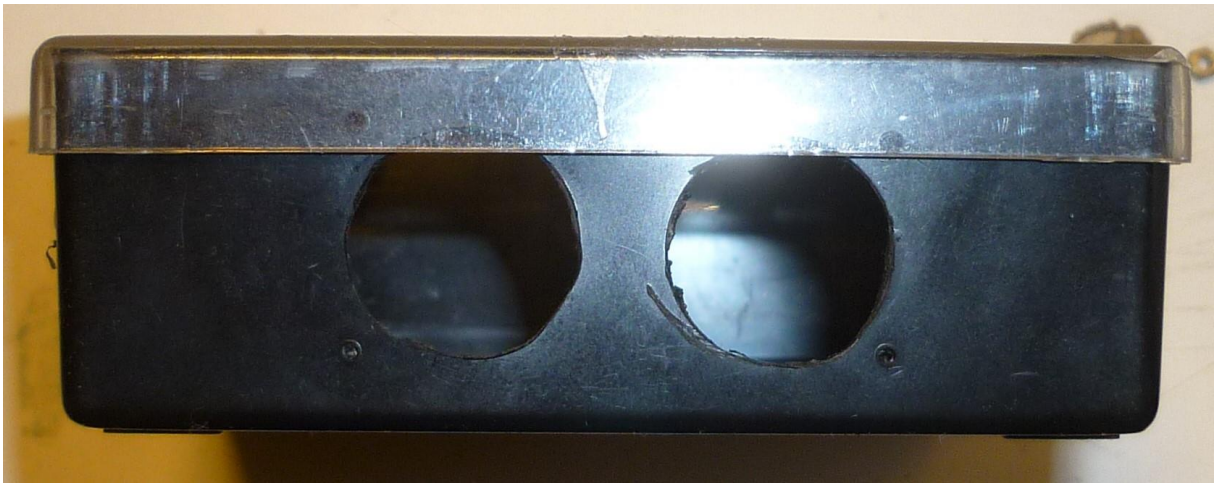

HC-SR04
GND DO VCC
VCC
REST
ADC
CH_PD
GP1016
GP1014
GP1012
GP1013
VCC
TXD
RXD
GP105
GP104
GP100
GP102
GP105
GND

15.4.2 Mécanique et boitier

Le boitier sera une simple boite en plastique carré de 9 cm de côté et de 3cm d'épaisseur avec un couvercle transparent pour pouvoir lire les afficheurs. Il sera percé afin de positionner tous les éléments du montage.

Le couvercle de la boite, un peu de nettoyage va s'imposer.

L’avant de la boite avec le passage pour le télémètre



Le côté droit avec le passage du thermomètre

Le côté gauche avec le passage du câble d’alimentation et les trous pour le bloquer.

L’arrière avec également quelques trous de blocage du câble d’alimentation

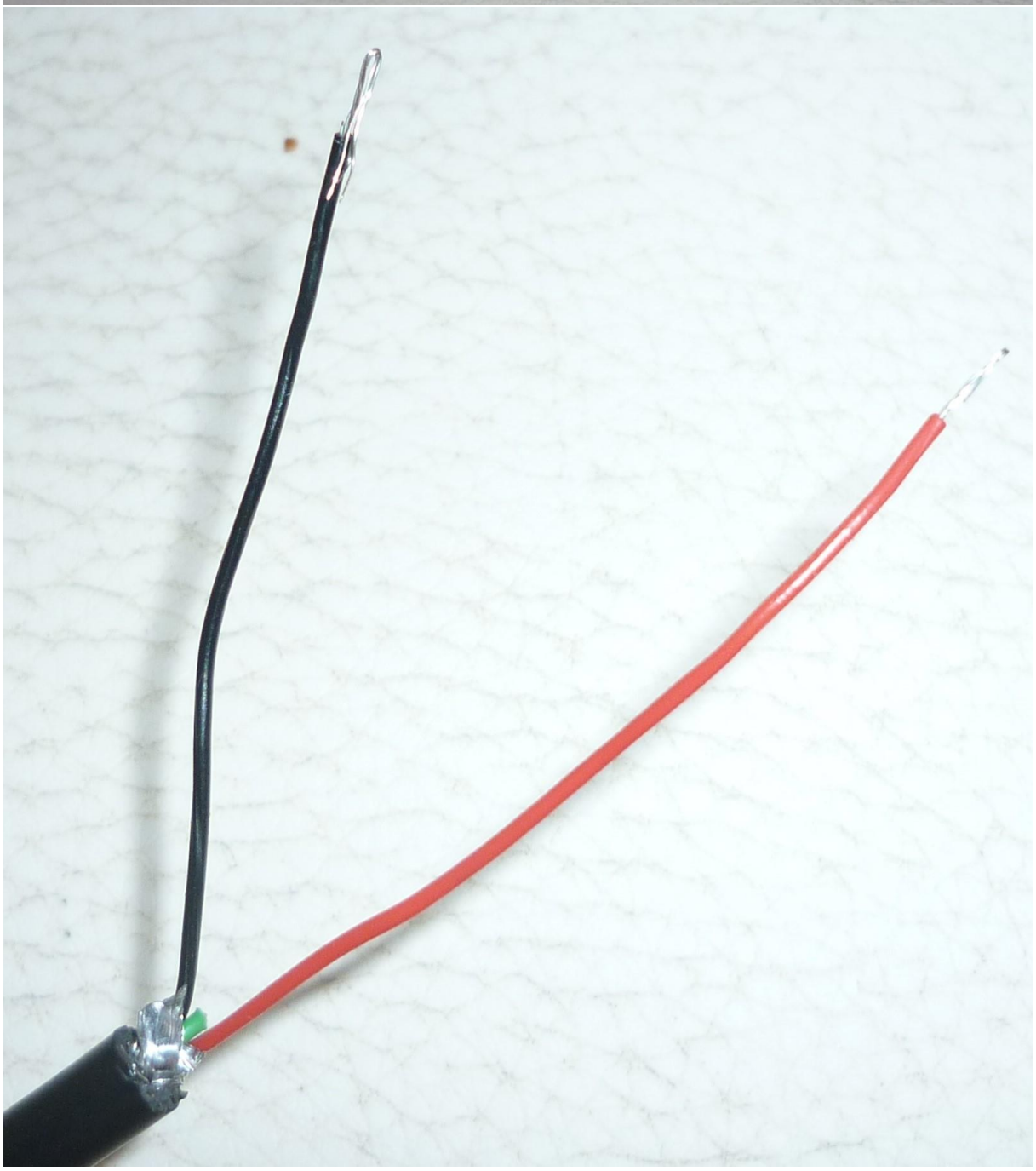
Le dessous avec deux trous pour le passage des vis supportant le circuit

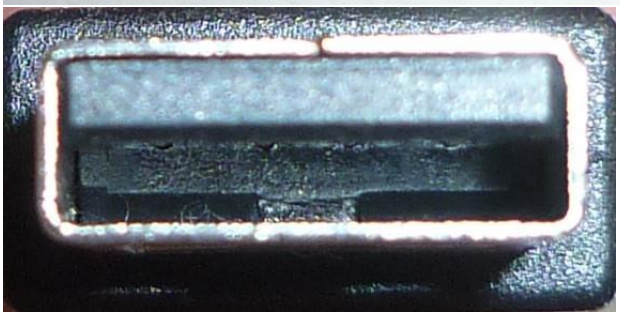
15.4.3 Alimentation

L'alimentation se fera par une batterie de secours de téléphone fournissant du 5V

Le câble d'alimentation sera donc un câble USB où on aura récupéré les deux fils d'alimentation rouge et noir.

Attention : les fils d'alimentation étant très fin, il faut en mettre une longueur la plus courte possible afin d'éviter un échauffement. Selon la norme, l'USB ne doit fournir que 500 mA au maximum (https://openclassrooms.com/courses/comprendre-l-usb-et-bricoler-un-peripherique) et là on risque d'être à la limite voir au-dessus. Mais dans la pratique, cela ne dérange personne de recharger par exemple une tablette à 2 voir 3A sur des câbles aussi fin…

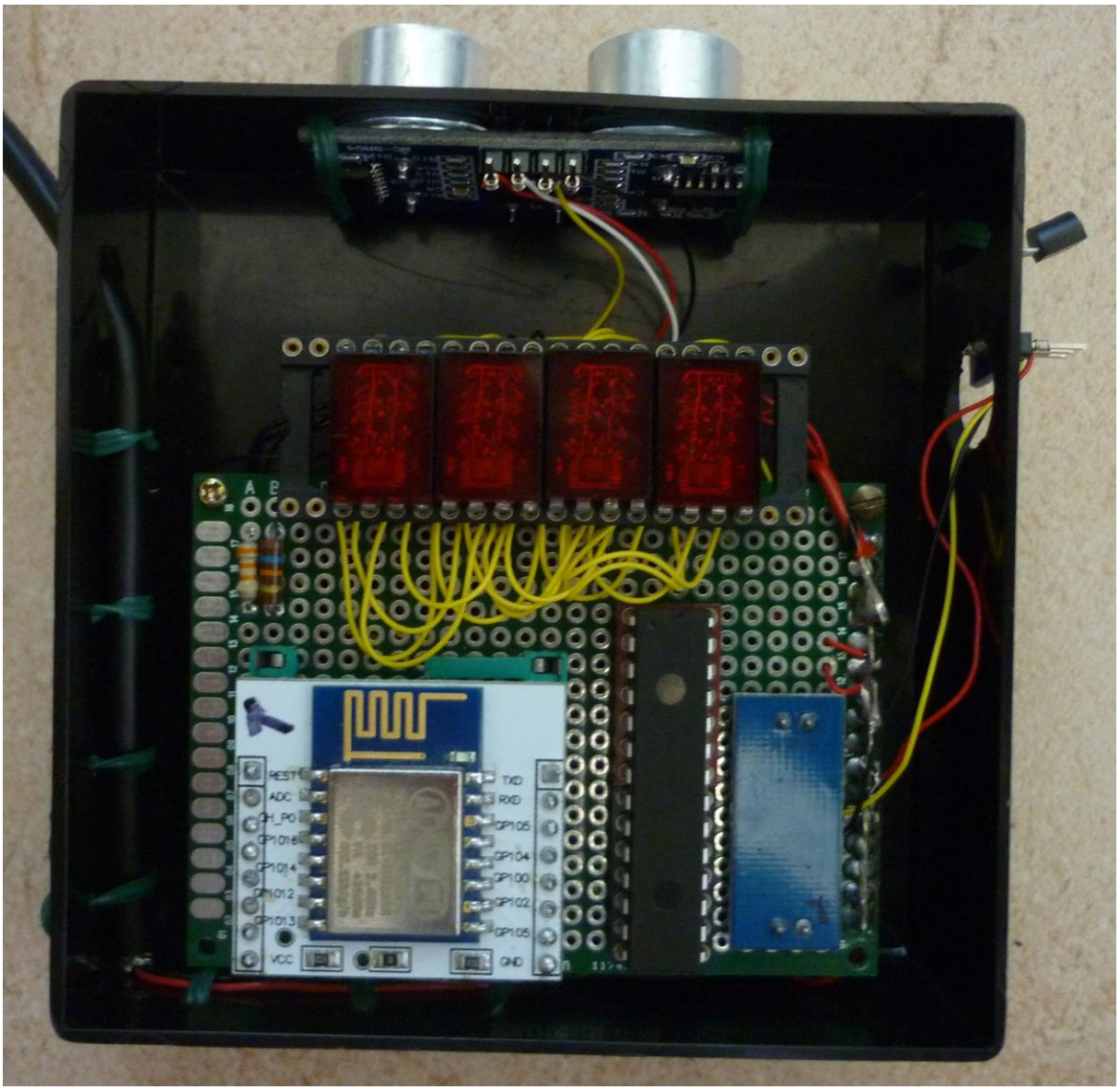




15.4.4 Montage complet

Différentes photos du boitier complet :

L'électronique dans le boitier

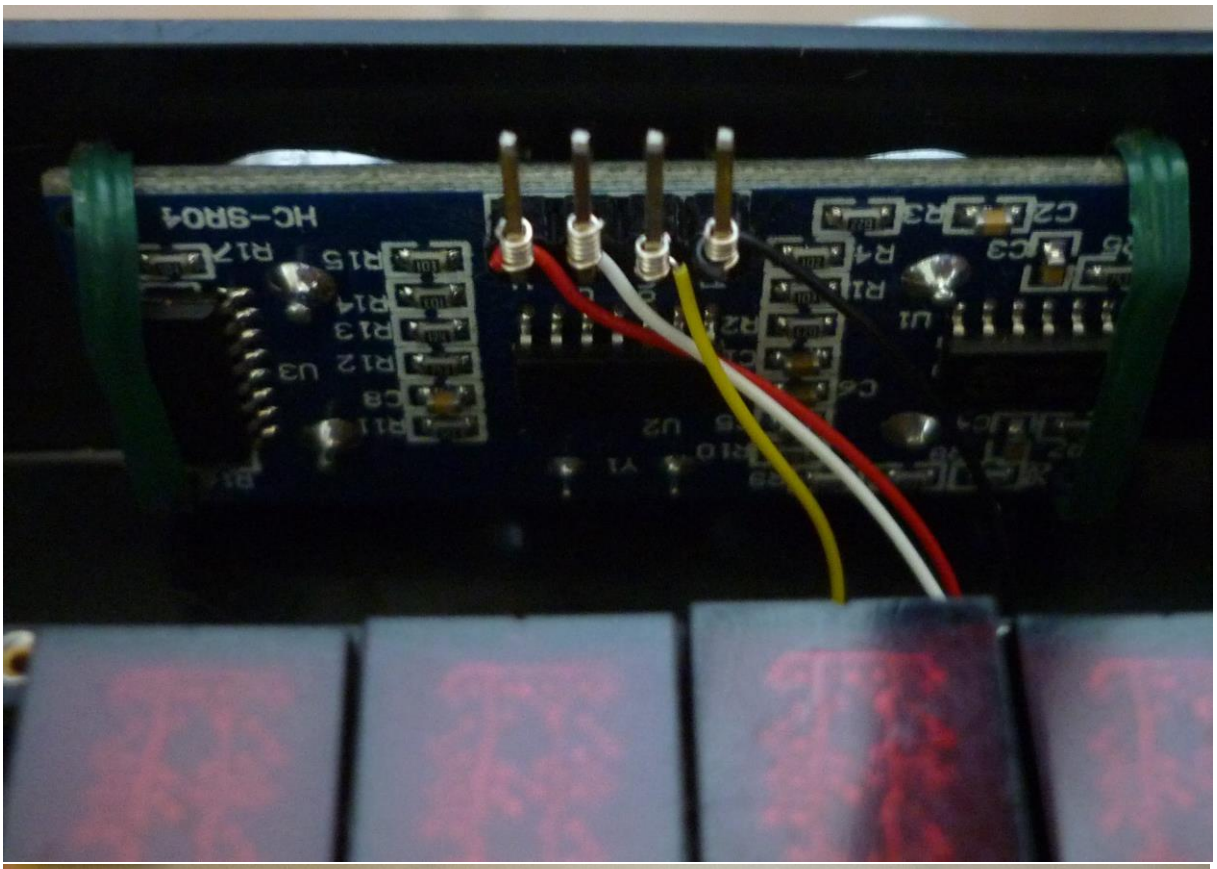


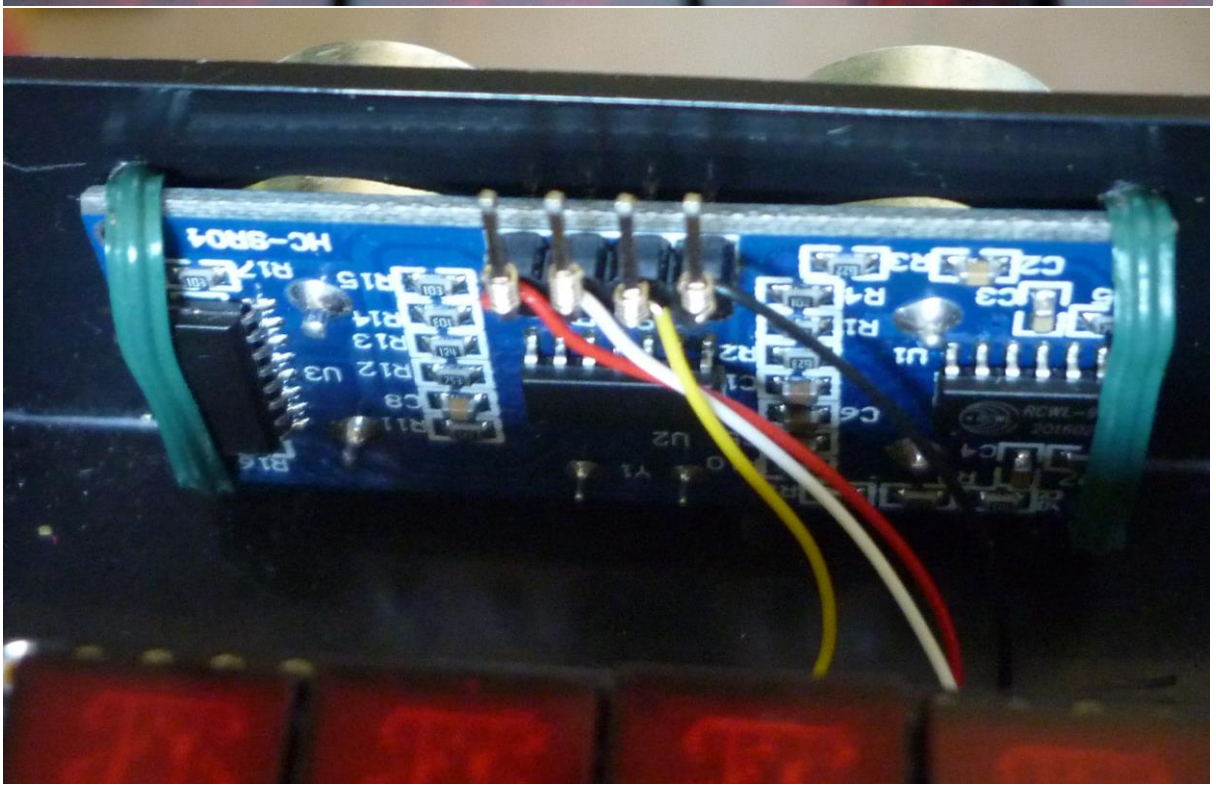
Le thermomètre sur le côté droit

Le télémètre à l’avant

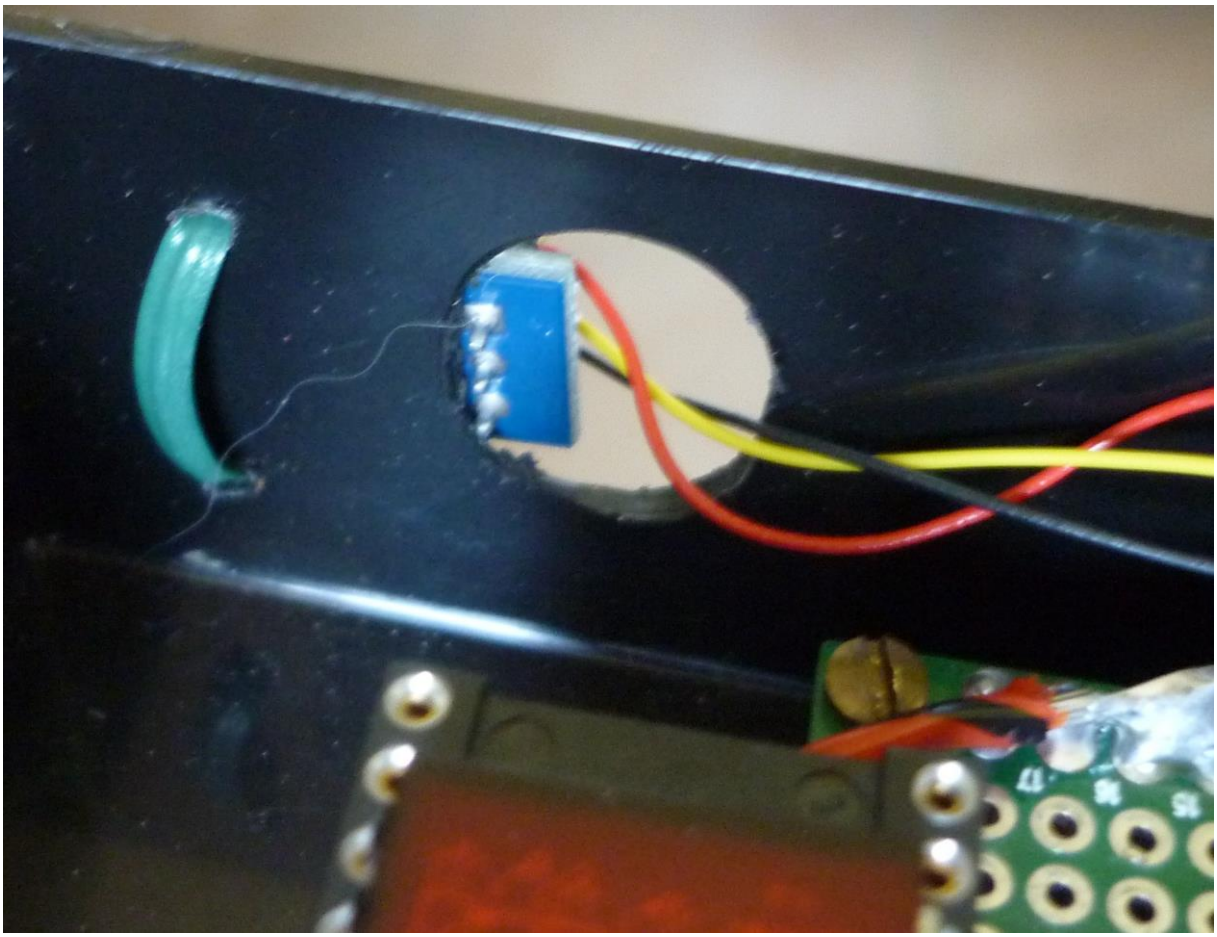
Le blocage du câble d’alimentation afin d’éviter de l’arracher

Le télémètre avec son câblage en wrapping et ses liens pour le tenir

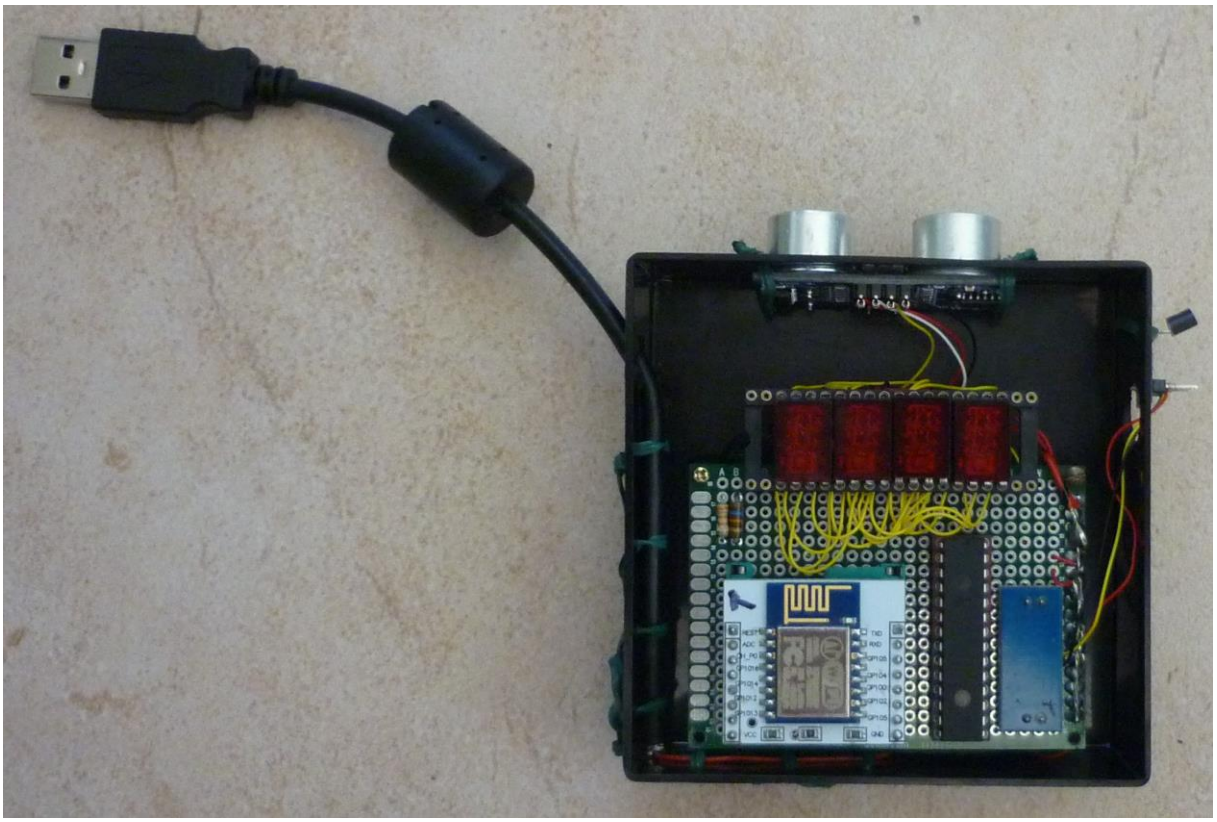




Le thermomètre et ses liens pour le tenir

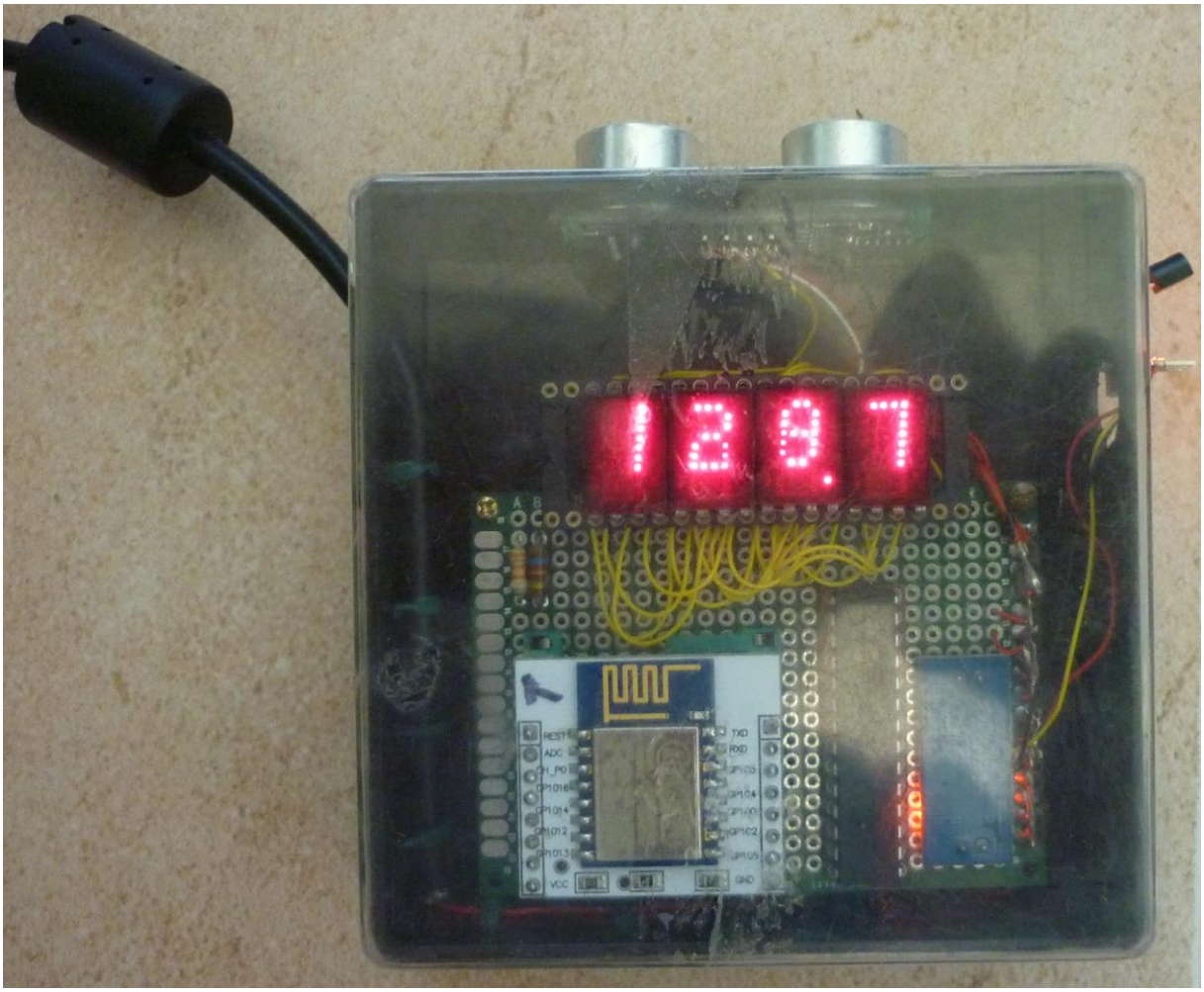


Le montage complet



Le montage branché à une batterie de téléphone en fonctionnement avec l'affichage de la température en premier

Puis les mesures



15.5 Liste des programmes

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/GestionAffichage.cpp

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/GestionAffichage.h

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/GestionHCSR04.cpp

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/GestionHCSR04.h

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/GestionHP50827300.cpp

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/GestionHP50827300.h

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/MCP23S17.cpp

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/MCP23S17.h

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/Telemetre.ino

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/TelemetreWeb.ino

Et les programmes de test intermédiaires

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/test/Affichage.ino

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/test/DS18B20.ino

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/test/HC-SR04.ino

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/test/MCP23S17.ino

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/test/MCP23S17_2.ino

https://github.com/montotof123/esp8266-12/blob/master/110_Telemetre/test/MCP23S17_3.ino

16 Mesurer la consommation réelle