Rapport de projet

**Ecole supérieure**

Électronique

Laboratoire PROJ

Salle R110

Capteur Météo Wifi

**Réalisé par :**

Diego Rickenbach

Table des matières

[Capteur Météo Wifi 1](#_Toc50817390)

[Pré-étude 5](#_Toc50817391)

[Description du projet 5](#_Toc50817392)

[Etapes du projet 5](#_Toc50817393)

[Schéma bloc hardware 6](#_Toc50817394)

[Liste de composants 7](#_Toc50817395)

[Evaluation du budget 9](#_Toc50817396)

[Planning du projet 10](#_Toc50817397)

[Design 11](#_Toc50817398)

[Schéma bloc 11](#_Toc50817399)

[Principe 12](#_Toc50817400)

[Convertisseur USB to UART 13](#_Toc50817401)

[Régulateur linéaire 16](#_Toc50817402)

[LEDs RGB 17](#_Toc50817403)

[Capteur 18](#_Toc50817404)

[Module Wifi 20](#_Toc50817405)

[Reset et connecteur de programmation 23](#_Toc50817406)

[Microcontrôleur 24](#_Toc50817407)

[Liste des composants 26](#_Toc50817408)

[PCB 27](#_Toc50817409)

[Contraintes 27](#_Toc50817410)

[Routage 28](#_Toc50817411)

[Particularités de routage 28](#_Toc50817412)

[Montage et tests 31](#_Toc50817413)

[Domoticz 31](#_Toc50817414)

[Alimentation 37](#_Toc50817415)

[Test de la communication UART 38](#_Toc50817416)

[Schéma de mesure 40](#_Toc50817417)

[Mesure 40](#_Toc50817418)

[Test de la communication I2C 41](#_Toc50817419)

[Schéma de mesure 41](#_Toc50817420)

[Mesure 42](#_Toc50817421)

[Erreurs 43](#_Toc50817422)

[Première erreur 43](#_Toc50817423)

[Deuxième erreur 43](#_Toc50817424)

[Troisième erreur 44](#_Toc50817425)

[Software 45](#_Toc50817426)

[Réglages Harmony 45](#_Toc50817427)

[Clock diagram 45](#_Toc50817428)

[Pin settings 46](#_Toc50817429)

[Configuration des périphériques 46](#_Toc50817430)

[Timer et interruption 47](#_Toc50817431)

[Organigrammes du programme 48](#_Toc50817432)

[Description des fonctions 51](#_Toc50817433)

[BME280.c 51](#_Toc50817434)

[ESP8266.c 58](#_Toc50817435)

[MenuGen.c 59](#_Toc50817436)

[Fonctionnement du menu 60](#_Toc50817437)

[Réglages PuTTY 60](#_Toc50817438)

[Gestion du menu 61](#_Toc50817439)

[Etat d’avancement 64](#_Toc50817440)

[Améliorations 64](#_Toc50817441)

[Conclusion 65](#_Toc50817442)

[Annexes 66](#_Toc50817443)

[Schéma 66](#_Toc50817444)

[Listing du programme 66](#_Toc50817445)

[Datasheets utiles 66](#_Toc50817446)

[BME280 66](#_Toc50817447)

[FT230X 66](#_Toc50817448)

[MAX1793 66](#_Toc50817449)

[PIC32MX130F256B 66](#_Toc50817450)

[Prévision Météo 66](#_Toc50817451)

[Wifi ESP click 66](#_Toc50817452)

[Procès verbaux 68](#_Toc50817453)

[Première semaine 68](#_Toc50817454)

[Deuxième semaine 69](#_Toc50817455)

[Troisième semaine 70](#_Toc50817456)

[Quatrième semaine 71](#_Toc50817457)

[Cinquième semaine 72](#_Toc50817458)

[Journal de travail 73](#_Toc50817459)

[Première semaine 73](#_Toc50817460)

[Deuxième semaine 74](#_Toc50817461)

[Troisième semaine 75](#_Toc50817462)

[Quatrième semaine 76](#_Toc50817463)

[Cinquième semaine 77](#_Toc50817464)

# Pré-étude

## Description du projet

Le but de ce projet sera de réaliser un capteur météorologique permettant de communiquer ses mesures et prévisions via Wifi sur Domoticz.

Domoticz est un système domotique qui vous permet de surveiller et de configurer divers appareils tels que: lumières, commutateurs divers capteurs/compteurs tels que la température, la pluie, le vent, les UV, l'électricité, le gaz, l'eau et bien plus encore.

Pour se faire, sur la carte se trouvera un capteur de température, humidité et pression sur lequel je viendrai récupérer les données toutes les 10min. Grâce aux données de pression, je pourrai créer un algorithme de prévision météo.

Les données et les prévisions seront envoyées via des requêtes HTTP à travers un module Wifi ESP click.

Un connecteur USB sera aussi présent afin d'alimenter le toute mais aussi permettant de communiquer avec le microcontrôleur depuis un ordinateur via un convertisseur UART-USB.

## Etapes du projet

* Recherche de composants électroniques
* Documentation et recherche du capteur et du module Wifi
* Réalisation schématique et PCB (Altium)
* Commande du PCB et composants
* Montage et test du PCB
* Programmation du mircocontrôleur
* Démontration par différentes mesures que les parties hardware et software ont bien été implantées.

## Schéma bloc hardware

**Code couleur des   
tensions d'alimentation**

**Rouge** : 5V

**Orange** : 3.3V

µC

LED

Micro USB

5V - 500mA

UART

Régulateur linéaire 3.3V

Communication USB

Convertisseur

UART-USB

USB

Capteur

Temp/Hum/Press

Module Wifi

I2C

UART

## Liste de composants

Pour ce projet, la majorité des composants m'ont été imposés. La partie alimentation est la seule dont j'ai pu choisir personnellement.

**Alimentation**

Pour l'alimentation, la carte sera alimentée par le microUSB et la tension de 5V sera abaissée grâce au **MAX1793EUE-33** à 3.3V afin d'alimenter tous les autres modules.

Caractéristiques :

* Tension de sortie 3.3V
* Courant maximum de sortit 1A

**Microcontrôleur**

Le microcontrôleur m'a été imposé et sera donc le **PIC32MX130** à 28 pins.

Caractéristiques :

* Tension d'alimentation de 2.3V à 3.6V
* Vitesse de fonctionnement maximum de 50 MHz
* 2 modules UART et I2C
* 2 modules SPI/I2S
* 5 module Output compare
* 5 Timers 16 bits

J'ai choisis de prendre le PIC32MX130F256B permettant d'avoir 256kB de mémoire flash afin d'avoir suffisament de place. Sachant que c'est un prototype, la mémoire pourra être optimisée lorsque le programme sera terminé et fonctionnel.

**Module Wifi**

Le module Wifi m'a aussi été imposé et sera le **WIFI ESP click board** de chez MIKROE. Ce module est équipé du ESP-WROOM-02 qui intègre un ESP8266EX.

Caractéristiques :

* Tension d'alimentation de 3.3V
* Modes de fonctionnements : AP (point d'accès), mode client WIFI
* Protocoles réseau : IPv4, TCP, UDP, HTTP, FTP
* Communication UART

L'avantage de ce type de module est leurs compatibilités mikroBUS qui permet d'être remplacé/changé par un autre module à tout moment.

**Capteur**

Le **BME280** sera le capteur que je devrai utiliser, il permet de mesurer l'humidité, la température et la pression.

Lors de la remise du cahier des charges, Monsieur Castoldi, m'a conseiller de prendre le module SEN-14348. Ce dernier est équipé du BME280 (capteur température, humidité et pression) ainsi que du CCS811 (capteur de gaz) qui à déjà été utilisé lors d'un projet.

Caractéristiques :

* Tension d'alimentation de 1.8V à 3.6V
* Consommation :1.8µA pour la température et l'humidité, 2.8µA pour la pression et l'humidité et 3.6µA pour la température, l'humidité et la pression
* Communication en SPI ou I2C
* Tolérances : humidité ±3 % relative, pression ±1.0 hPa, température ±1.0 °C

**Convertisseur**

Pour la converssion USB-UART, le **FT230X** est le composant qui m'a été imposé par mon client. Ce composant me permettera de faire le pont entre la communication USB du PC et UART du microcontrôleur.

Caractéristiques :

* Tension d'alimentation de 3.3V
* Vitesse de transfère de données de 300 baud à 3 Mbaud
* Normes : USB à RS232/RS422/RS485
* Buffer de transmission de 512 byte
* Mémoire interne programme de 2048 byte

## Evaluation du budget

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nom** | | | **Quantité** | | | **Type** | | | **Fabriquant** | | | **N° Fabriquant** | | | **Fournisseur** | | | **Prix u** | **Total** |
|
| Micro USB | | | 1 | | | Micro USB - B | | | Molex | | | 105164-0001 | | | Mouser | | | fr. 0.85 | fr. 0.85 |
|
| Régulateur linéaire 3.3V | | | 1 | | | MAX1793 | | | Maxim Integrated | | | MAX1793EUE33+ | | | Mouser | | | fr. 3.62 | fr. 3.62 |
|
| microcontrôleur | | | 1 | | | PIC32MX130F256B | | | Microchip | | | PIC32MX130F256B-I/SS | | | Mouser | | | fr. 2.63 | fr. 2.63 |
|
| Module Wifi | | | 1 | | | WIFI ESP click board | | | Mikroe | | | MIKROE-2542 | | | Mouser | | | fr. 14.31 | fr. 14.31 |
|
| Capteur Temp/Hum/Press | | | 1 | | | SEN-14348 | | | SparkFun | | | SEN-14348 | | | Mouser | | | fr. 35.04 | fr. 35.04 |
|
| Convertisseur UART-USB | | | 1 | | | FT230XS-U | | | FTDI | | | FT230XS-U | | | Mouser | | | fr. 1.95 | fr. 1.95 |
|
| LED | | | 1 | | | LED Green | | | Kingbright | | | APTD2012LZGCK | | | Mouser | | | fr. 0.63 | fr. 0.63 |
|
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Total** | fr. 59.03 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## Planning du projet

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N° semaine projet** | 1 | 1 | 1 | 1 | 1 |  |  | 2 | 2 | 2 | 2 | 2 |  |  | 3 | 3 | 3 | 3 | 3 |  |  | 4 | 4 | 4 | 4 | 4 |  |  | 5 | 5 | 5 | 5 | 5 |  |  |  |  |
| **N° semaine** | 33 | 33 | 33 | 33 | 33 |  |  | 34 | 34 | 34 | 34 | 34 |  |  | 35 | 35 | 35 | 35 | 35 |  |  | 36 | 36 | 36 | 36 | 36 |  |  | 37 | 37 | 37 | 37 | 37 |  |  |  |  |
| **Date** | 10.08.2020 | 11.08.2020 | 12.08.2020 | 13.08.2020 | 14.08.2020 | 15.08.2020 | 16.08.2020 | 17.08.2020 | 18.08.2020 | 19.08.2020 | 20.08.2020 | 21.08.2020 | 22.08.2020 | 23.08.2020 | 24.08.2020 | 25.08.2020 | 26.08.2020 | 27.08.2020 | 28.08.2020 | 29.08.2020 | 30.08.2020 | 31.08.2020 | 01.09.2020 | 02.09.2020 | 03.09.2020 | 04.09.2020 | 05.09.2020 | 06.09.2020 | 07.09.2020 | 08.09.2020 | 09.09.2020 | 10.09.2020 | 11.09.2020 | 02.10.2020 | 05.10.2020 | 06.10.2020 | 07.10.2020 |
| **Tâches** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Pré-étude |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Design + Schéma |  |  |  | R |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| PCB |  |  |  | R |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Montage |  |  |  |  |  |  |  |  | Commande | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Software |  |  |  |  |  |  |  |  |  |  | R |  |  |  |  |  |  | R |  |  |  |  |  |  | R |  |  |  |  |  |  | R |  |  |  |  |  |
| Test et Debug |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | R |  |  |  |  |  |  | R |  |  |  |  |  |  | R |  |  |  |  |  |
| Rédaction du rapport |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Présentations |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Finitions/corrections |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Bien évidemment ce planning sera sujet à des changements tout au long du déroulement du projet

# Design

## Schéma bloc

µC

LED

Micro USB

5V - 500mA

UART

Régulateur linéaire 3.3V

Communication USB

Convertisseur

UART-USB

USB

Capteur

Temp/Hum/Press

Module Wifi

I2C

UART

**Code couleur des   
tensions d'alimentation**

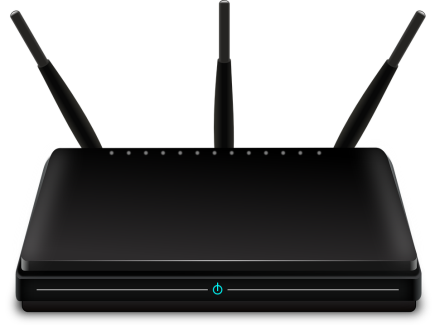
**Rouge** : 5V

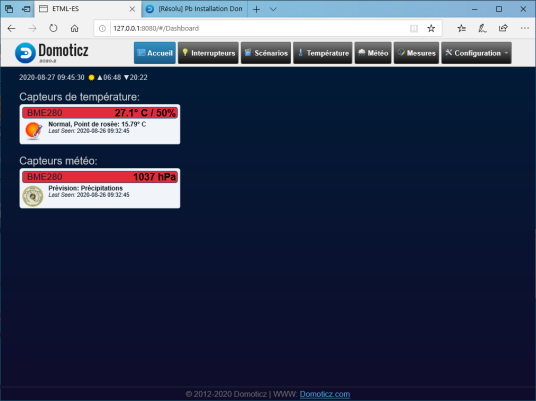
**Orange** : 3.3V

## Principe

Comme expliqué dans la pré-étude, le but de cette carte sera de récupérer les données du capteur qu'elle intègre puis de les envoyer périodiquement via le module Wifi sur Domoticz.

La carte sera alimentée en continue par micro USB et nous pourrons aussi communiquer avec le microcontroleur, au travers du convertisseur USB to UART (FT230X).





Communication Wifi

Communication Wifi



PCB

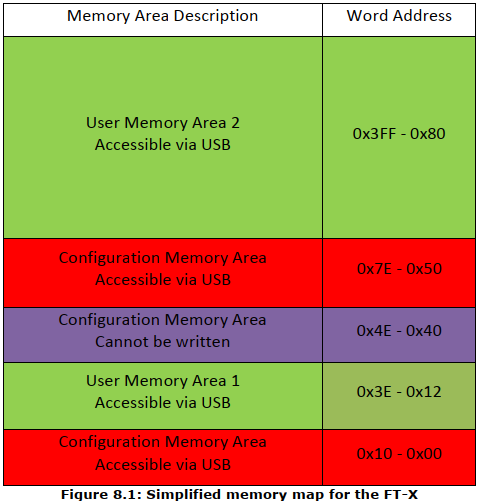
Communication USB



## Convertisseur USB to UART

Le FT230X est un convertisseur USB à UART, il permet de convertir directement les données reçues dans un sens comme de l'autre.

J'ai repris un schéma qui existait déjà (projet 1740), qui est lui même tirer du schéma proposé dans le datasheet fournis par le fabriquant.

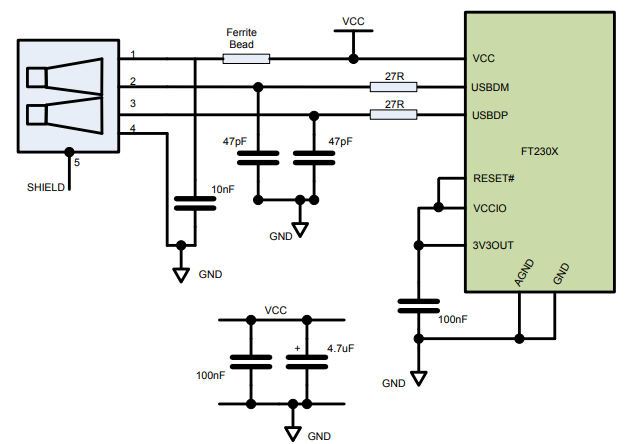
Ce composant est doté d'un mémoire interne programmable de 2048 bits.

Les zone vertes (User Memory Area), peuvent être écrites et lus via USB à n'importe quelle moment.

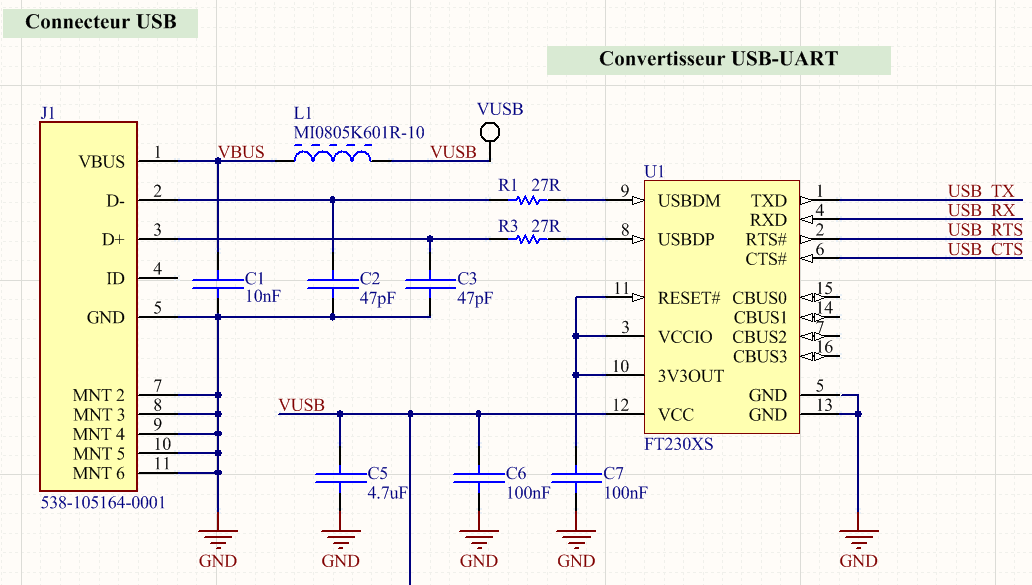
Les zones rouge (Configuration Memory Area), peuvent être écrites et lus via USB à n'importe quelle moment. Ces zones stockent les données de configuration matérielle de l'appareil, (le signal attribué à chaque pin CBUS pour exemple). Ces valeurs peuvent avoir un effet significatif sur le comportement du composant.

Les zones violettes (Configuration Memory Area), ne peuvent pas être modifiées. Il s'agit d'une zone réservée et toute tentative d'écriture sur ces emplacements échouera.

Il existe plusieurs manières de brancher ce composant, pour ma part j'ai utilisé celle permettant de directement alimenter le FT230X grâce au port USB.

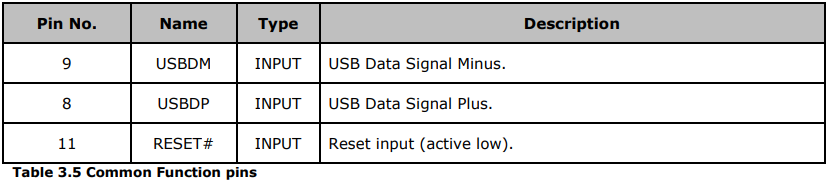


Voici le schéma pour mon convertisseur dans son ensemble, détails du dimensionnement des composants et des pins après.



Pour le dimensionnement de cette partie, tous les composants m'ont été fournis par le datasheet du fabriquant ainsi que le branchement.

Lors de la communication USB, la synchronisation du temps de montée/descente des signaux ne dépend pas seulement des drivers de signaux USB. Elle peut dépendre également du système et être affectée par des facteurs tels que la disposition du PCB, les composants externes ou encore des composants de protections pour des lignes différentielles. La synchronisation peut également être modifiée en ajoutant des composants passifs appropriés aux signaux USB. D'où l'utilisation des condensateurs C2 et C3 ainsi que des résistances R1 et R3. Il est aussi fortement conseillé d'ajouter un condensateur de découpage (C1) ainsi qu'une ferrite (L1) le plus proche possible du connecteur USB afin d'avoir un signal 5V le plus propre possible.

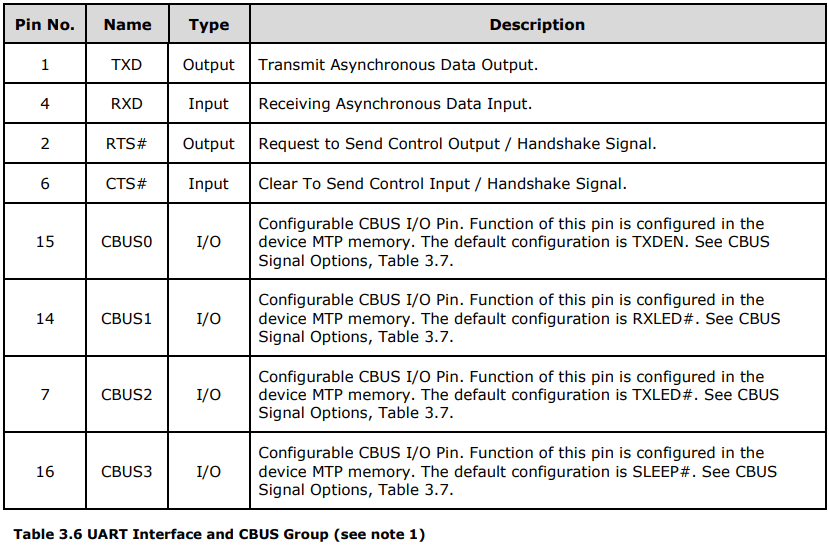


**PIN 9, USBDM PIN 8, USBDP**

Les pins USBDM et USBDP seront utilisées pour la communication USB. Ces lignes sont une paire différentielle afin de limiter leurs sensibilités aux interférences.

**PIN 11, RESET#**

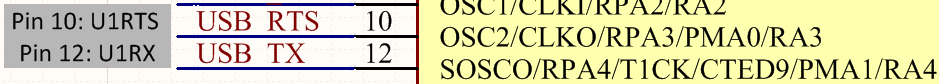
Le reset du module est activé lorsque l'état de la pin passe LOW. Dans notre cas, elle sera reliée à la pin 10 (3V3OUT) afin que le device ne soit jamais reseté.



**PIN 1, TXD PIN 4, RXD PIN 2, RTS# PIN 6, CTS#**

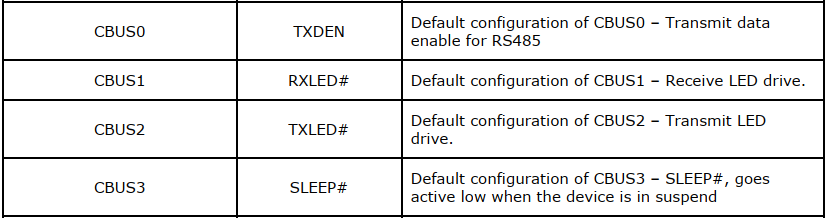
Ces pins seront utilisées pour la communication UART. Chaqu'unes d'elles du composant sont reliées au microcontroleur en faisant bien attention d'inverser le TX et RX.





**PIN 15, CBUS0 PIN 14, CBUS1 PIN 7, CBUS2 PIN 16, CBUS3**

Ces pins sont utilisées pour configurer le device de façon hardware. Dans notre cas, nous n'allons pas les utilisé d'où leurs "non connection".

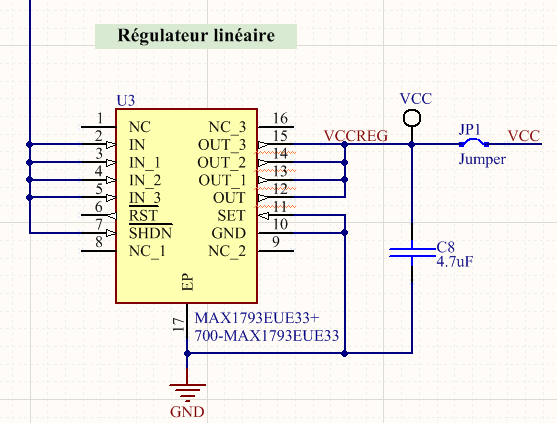


## Régulateur linéaire

Le régulateur MAX1793EUE33+ est le même que celui utilisé sur le kit PIC32MX795F256B fournit par l'ETML-ES. Il permettra d'abaisser la tension du micro USB 5V à 3.3V afin d'alimenter tous les modules.

Un jumper a été ajouté afin de pouvoir tester l'alimentation avant de la connecter au circuit.

Voici le schéma de mon régulateur linéaire.



## **LEDs** RGB

Aux premiers abords, il m'a été demandé de ne mettre qu'un LED mono couleur mais finalement après discussion avec M. Castoldi, je l'ai remplacé par une LED RGB car il restait des pins de libre sur le microcontrôleur.

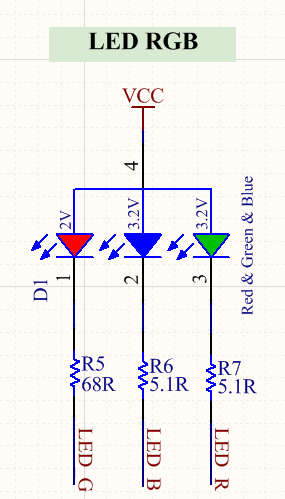
J'ai donc opté pour la LED RGB, 598-8710-307F.



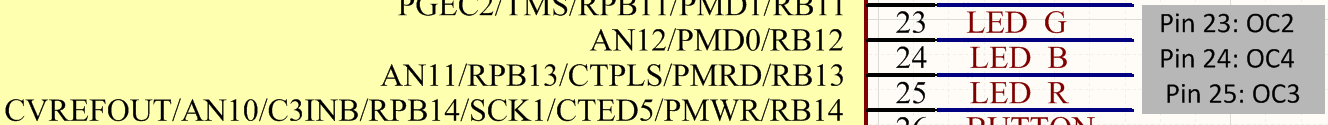
**Calcul de la résistance pour la LED rouge**

**Calcul des résistances pour la LED verte et la LED bleue**

Résultat sur le schéma ci-dessous. Les pattes du microcontrôleur devront donc être mises à 0 pour qu'une des LEDs s'allume.



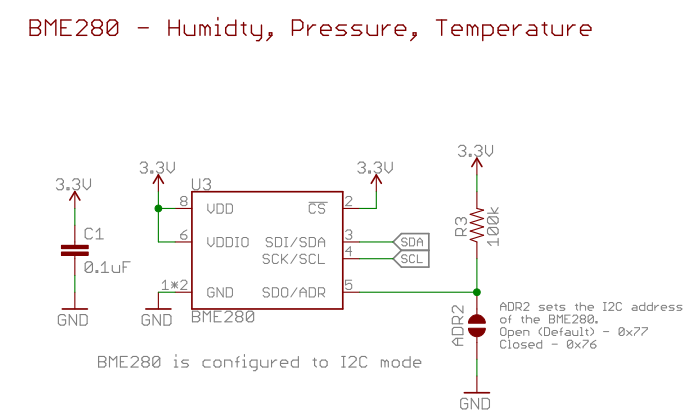
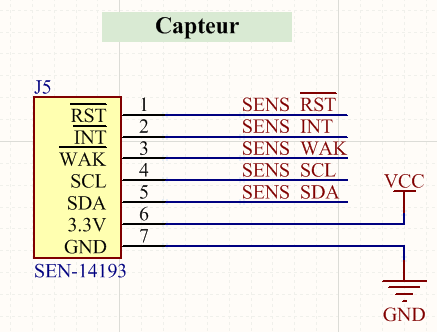
Chacunes des cathodes de la LED RGB est connectées à une pattes OC, ce qui me permettre de mieux gérer les couleurs.



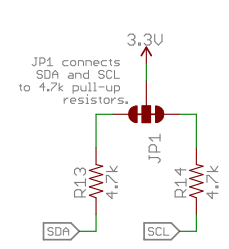
## Capteur

Lors d'une discussion avec M. Castoldi, nous avons discutés de prendre le kit de Sparkfun qui à déjà été utilisé par l'EMTL-ES, il contient en plus du BME280, le CCS811 (capteur de CO2). Il à l'avantage de pouvoir être pluggé directement sur des connecteurs 2.54mm.

Voici le schéma pour mon capteur dans son ensemble, détails des pins et du dimensionnement des composants après.



***Symbole schématique Schéma interne au module***

**PIN 4, SCL PIN 5, SDA**

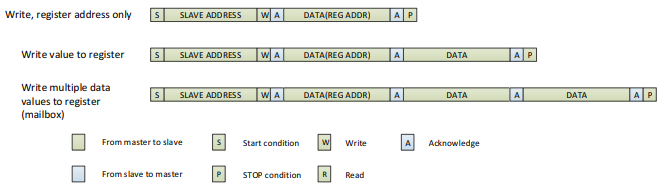
Les pins SCL et SDA seront utilisées pour la communication I2C.

Grâce au module tout les résistance de pull-up, sont déjà intégrées.

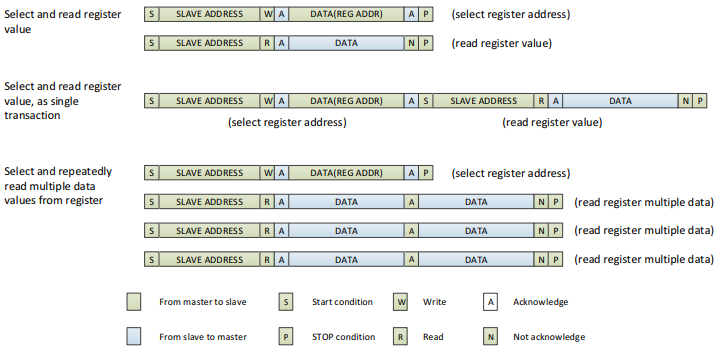


**Construction d'une trame I2C**

Voici comment construire une trame pour l'écriture d'un ou plusieurs registre(s) du capteur.

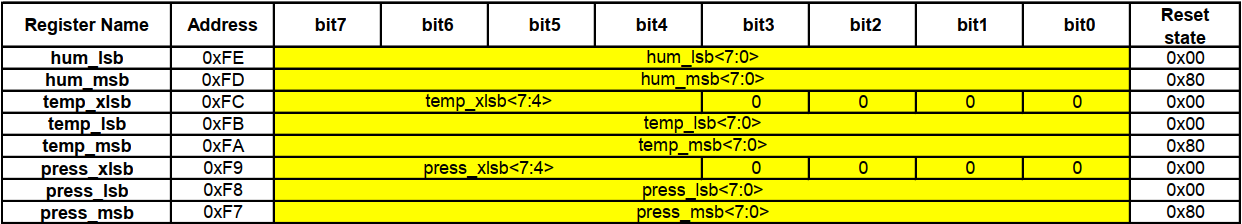


Voici comment construire une trame pour la lecture d'un ou plusieurs registre(s) du capteur.



Le BME280 peut avoir deux adresses, la première est 0x76 si la pin ADDR est à LOW ou 0x77 si la pin est à l'état HIGH.

Les registres qui me seront utiles lors des mesures de température et d'humidité et pression.

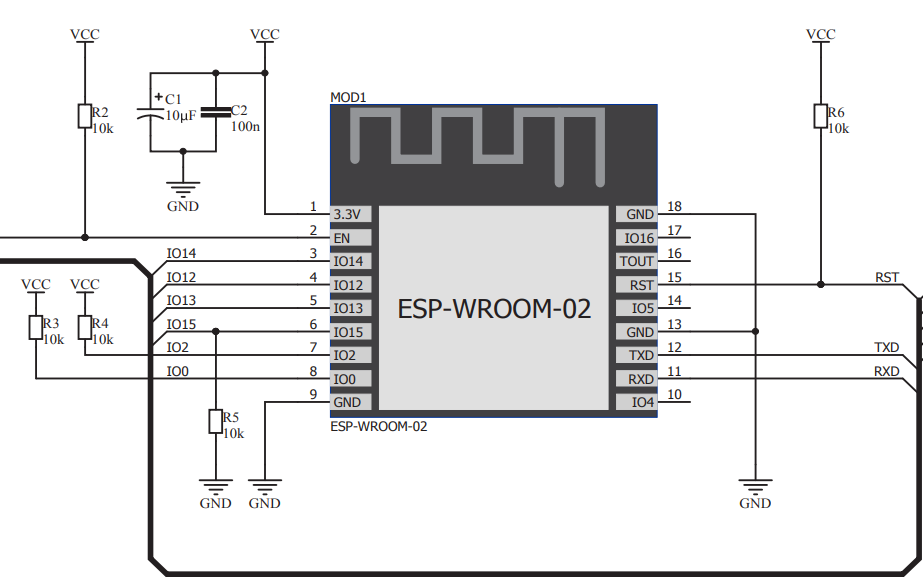


Nous pouvons voir que les données d'humidité sont sur deux bytes et celles pour la température et la pression se trouvent sur 3 bytes.

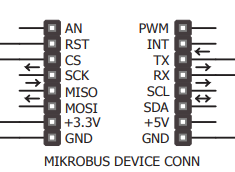
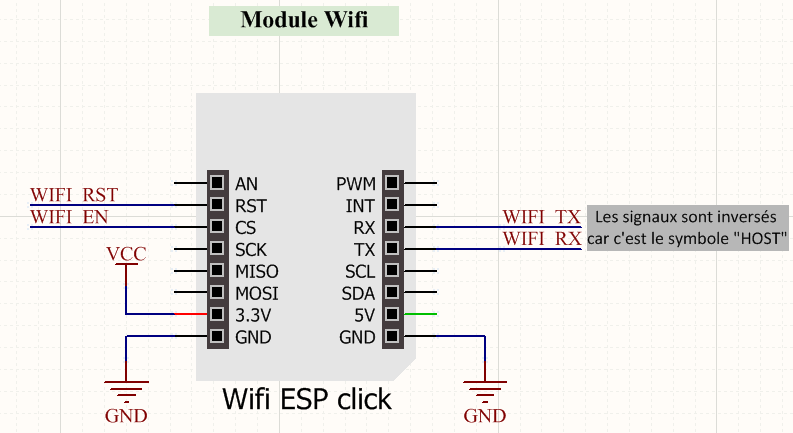
## MikroE Click Boards Wireless Connectivity WiFi ESP click frontModule Wifi

Comme demander dans mon cahier des charges, le module Wifi que je dois utiliser est le Wifi ESP click. Ce module est équipé d'un ESP-WROOM-02 qui integre un ESP8266EX.

Voici le schéma interne au module wifi. On peut voir que les que les pull-ups et les condensateur de découplages sont déjà intégrés et donc pas besoins de les mettre sur le PCB.



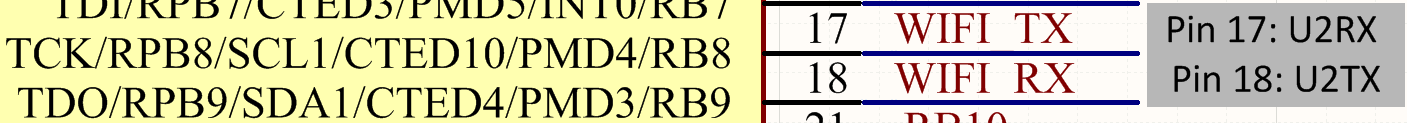
Voici le schéma de mon module Wifi, les pins RX et TX ont été inversées car c'est le symbole HOST (où viendra se connecter le Wifi ESP click).



**Symbole schématique HOST Symbole schématique DEVICE**

**PIN TX PIN RX**

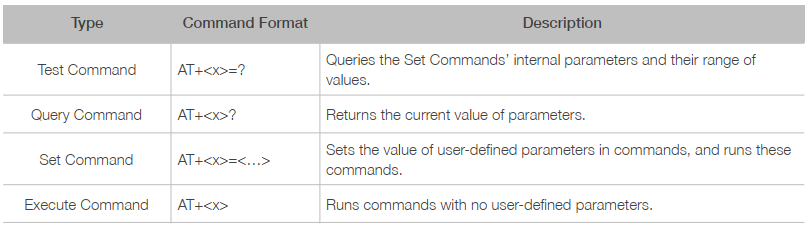
Les pins TX et RX seront utilisées pour la communication UART entre le microcontrôleur et le module.



**PIN RST PIN EN**

Le reset du module est activé lorsque l'état de la pin passe LOW. Le chip enable est active à l'état haut.

Il existe plusieurs commandes fournit par le fabriquant qui me permetteront de communique avec le module Wifi, elles commencent toute par "AT+..." quelques que soit le type de la commande.



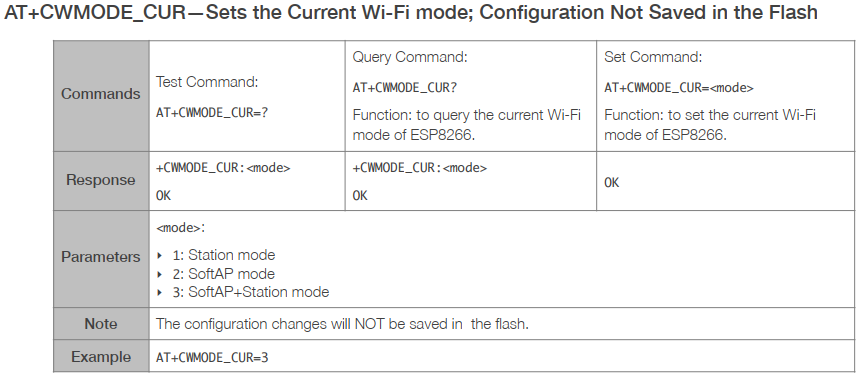
J'ai listé quelques commandes qui me sembleront utiles pour mon projet :



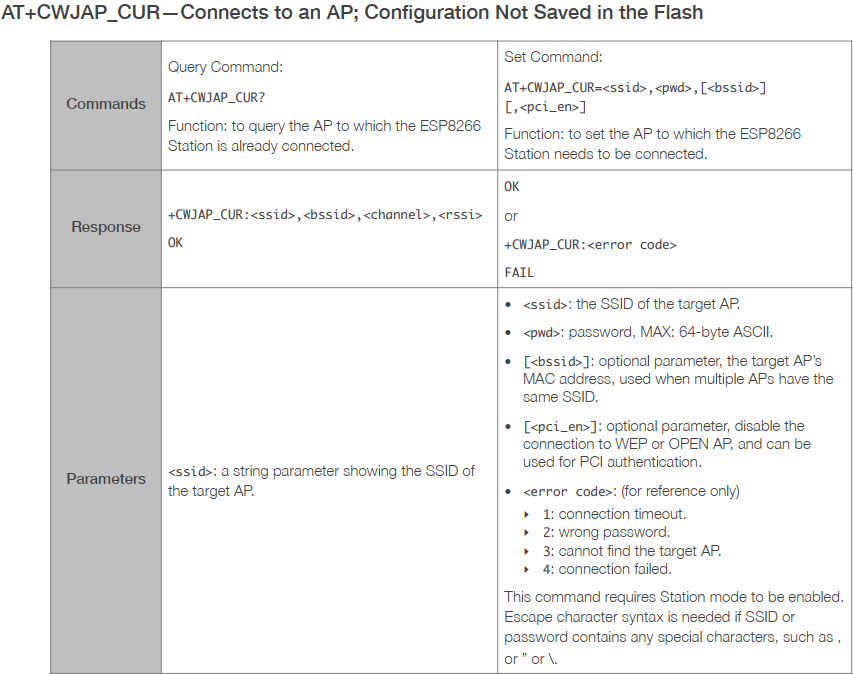
Cette commande me permettra de vérifié si le module wifi est bien connecté si je reçois en retour un "OK".



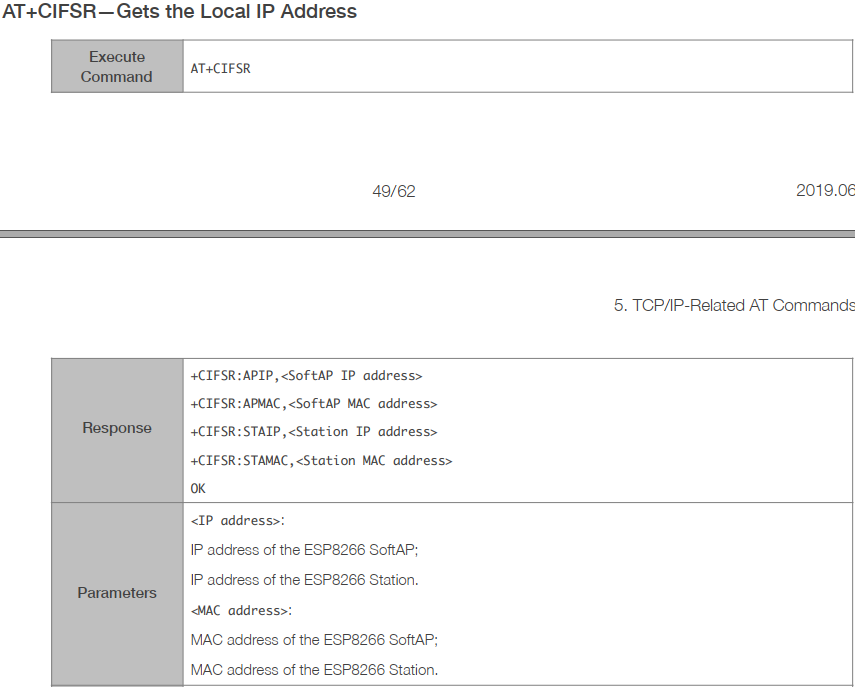
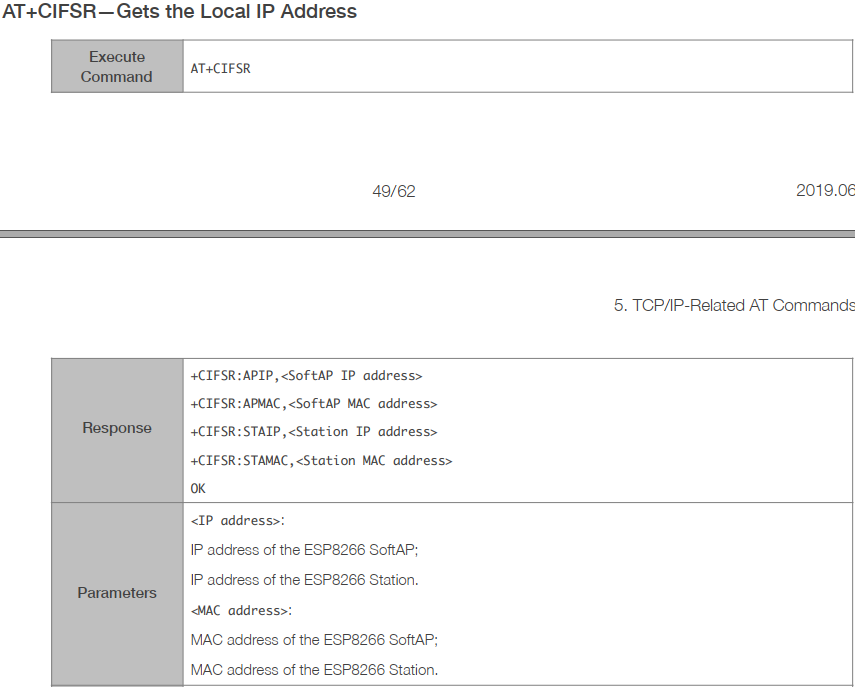
Cette commande me permettra de reset le module wifi et je recevrai un "OK" en retour.



Cette commande permettera de selectionner un mode de fonctionnement du module Wifi, Station mode, SoftAP mode et SoftAP+Station mode. Dans mon cas, je serai dans le premier mode car il faut que le module transfers les données. En retour on reçois un "OK" si ça à fonctionné.



Cette commande permet au module Wifi de se connecter à un réseau en mettant en paramètre le SSID et le mot de passe. En retour on reçois un "OK" si ça à fonctionné.

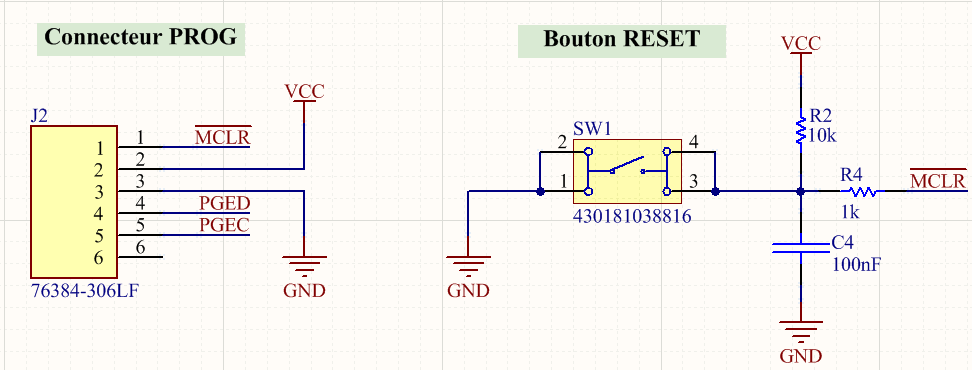


Cette commande permet de recevoir l'addresse IP et MAC du module Wifi.

## Reset et connecteur de programmation

Le schéma de reset grâce au bouton poussoir et le schéma du connecteur de programmation a été tiré du schéma du kit PIC32MX795F512L fourni par l'ETML-ES.

Voici le schéma de mon reset et bouton reset et le détails des pins après.



Lorsque l'on presse cela met un 0V sur la pin /MCLR du microcontrôleur ce qui permet de le reset.

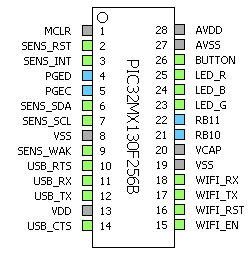
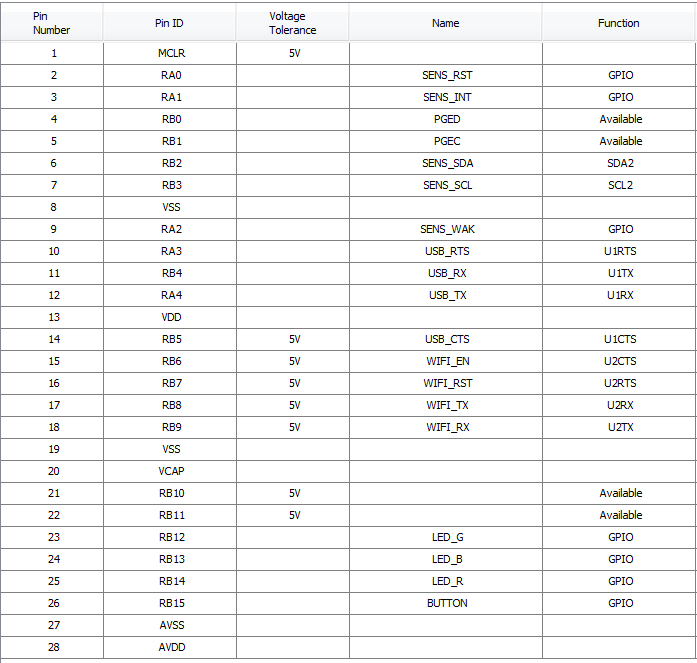
**PIN 4, PGED PIN 5, PGEC**

Les pins de programmation sont directement reliées aux pâtes PGED1 et PGEC1 (pin 4 et pin 5) du microcontrôleur.

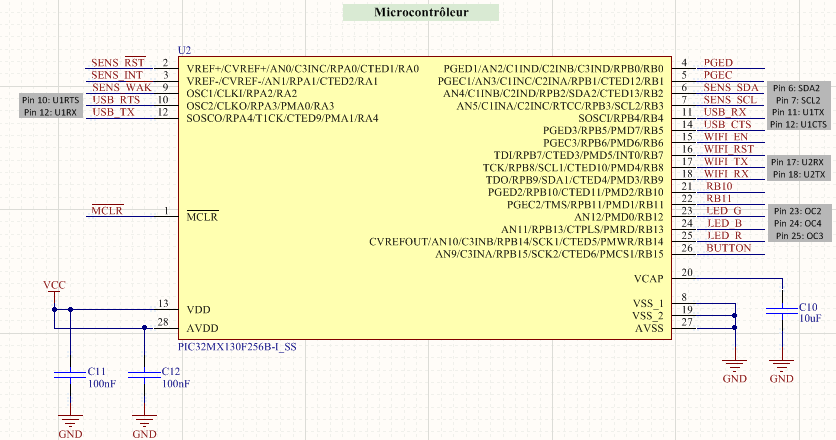


## Microcontrôleur

Le microcontrôleur que j'utiliserai sur cette carte est le PIC32MX130F256B. Grâce à l'IDE MPLAB X, j'ai pu directemment faire le pinout. Ci-dessous, on peut voir le listing de toutes les pins utilisées et quelles fonctions elles auront.

Finalement, voici le schéma de mon PIC32.



Deux connecteurs ont été ajoutés afin de toujours pouvoir faire des mesures sur tout les signaux.

Diagramme d'état



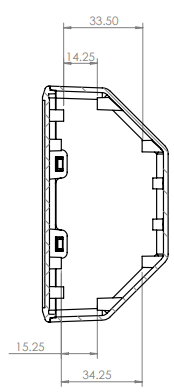
## Liste des composants

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Designator** | **Descriptor** | **Value** | **Manufacturer** | **Manufacturer Part Number** | **Supplier** | **Supplier Part Number** | **Price / unit** | **Quantity** | **Price** |
| 1 | C1 | 0805 Condensator | 10nF | Wurth Electronik | 885382207006 | Mouser | 710-885382207006 | 0.09 | 1 | 0.09 |
| 2 | C2, C3 | 0805 Condensator | 47pF | Wurth Electronik | 885012007014 | Mouser | 710-885012007014 | 0.09 | 2 | 0.18 |
| 3 | C4, C6, C7, C11, C12 | 0805 Condensator | 100nF | Wurth Electronik | 885382207007 | Mouser | 710-885382207007 | 0.12 | 5 | 0.6 |
| 4 | C5, C8 | 0805 Condensator | 4.7uF | Wurth Electronik | 885012107013 | Mouser | 710-885012107013 | 0.13 | 2 | 0.26 |
| 5 | C10 | 0805 Condensator | 10uF | Wurth Electronik | 885012107004 | Mouser | 710-885012107004 | 0.17 | 1 | 0.17 |
| 6 | D1 | LED standard - CMS RGB 0603 SMD | - | Dialight | 598-8710-307F | Mouser | 645-598-8710-307F | 1.99 | 1 | 1.99 |
| 7 | GND, VCC, VUSB | Test point smd | - | Keystone Electronics | 5019 | Mouser | 534-5019 | 0.36 | 3 | 1.08 |
| 8 | J1 | Connecteurs micro USB top mount | - | Molex | 105164-0001 | Mouser | 538-105164-0001 | 0.79 | 1 | 0.79 |
| 9 | J2 | Connecteur de programmation | - | Amphenol FCI | 76384-306LF | Mouser | 649-76384-306LF | 0.88 | 1 | 0.88 |
| 10 | J3, J4 | Pin Header, pitch 2.54mm, Single Row, 14pin | - | Wurth Electronik | 61301411121 | Mouser | 710-61301411121 | 0.68 | 2 | 1.36 |
| 11 | J5 | CCS811 + BME280 kit | - | SparkFun | SEN-14348 | Mouser | 474-SEN-14348 | 33.60 | 1 | 33.6 |
| 12 | JP1 | Jumper | - | - | - | - | - |  | - | - |
| 13 | L1 | Chip Bead for Power Filtering, 0805, RoHS | - | Laird Performance Materials | MI0805K601R-10 | Mouser | 875-MI0805K601R-10 | 0.09 | 1 | 0.09 |
| 14 | R1, R3 | 0805 Resistor | 27R | Yageo | RT0805FRE0727RL | Mouser | 603-RT0805FRE0727RL | 0.11 | 2 | 0.22 |
| 15 | R2, R8 | 0805 Resistor | 10k | Yageo | RT0805FRE1310KL | Mouser | 603-RT0805FRE1310KL | 0.11 | 2 | 0.22 |
| 16 | R4 | 0805 Resistor | 1k | Yageo | RT0805FRE101KL | Mouser | 603-RT0805FRE101KL | 0.11 | 1 | 0.11 |
| 17 | R5 | 0805 Resistor | 68R | Yageo | RT0805FRE0768RL | Mouser | 603-RT0805FRE0768RL | 0.10 | 1 | 0.1 |
| 18 | R6, R7 | 0805 Resistor | 5.1R | Yageo | RT0805DRD075R1L | Mouser | 603-RT0805DRD075R1L | 0.46 | 2 | 0.92 |
| 19 | SW1, SW2 | WS-TASV SMD Tact Switch 4.5X4.5 mm | - | Wurth Electronik | 430181038816 | Mouser | 710-430181038816 | 0.46 | 2 | 0.92 |
| 20 | U1 | USB to Basic UART IC | - | FTDI | FT230XS-U | Mouser | 895-FT230XS-R | 1.87 | 1 | 1.87 |
| 21 | U2 | Microcontrôleurs 32 bits - 256KB FL | - | Microchip Technology | PIC32MX130F256B-I/SS | Mouser | 579-MX130F256BISS | 2.53 | 1 | 2.53 |
| 22 | U3 | Régulateurs de tension 3.3V, 1A | - | Maxim Integrated | MAX1793EUE33+T | Mouser | 700-MAX1793EUE33T | 3.41 | 1 | 3.41 |
| 23 | Wifi ESP click | mikroBUS Host Socket | - | Mikroe | MIKROE-2542 | Mouser | 932-MIKROE-2542 | 13.73 | 1 | 13.73 |
|  |  |  |  |  |  |  |  |  | **TOTAL** | 65.120 |

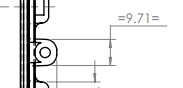
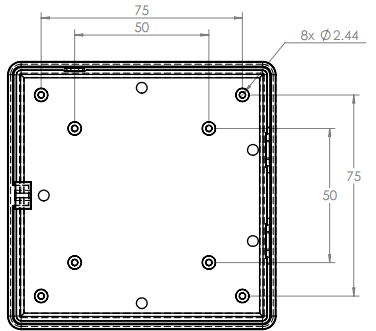
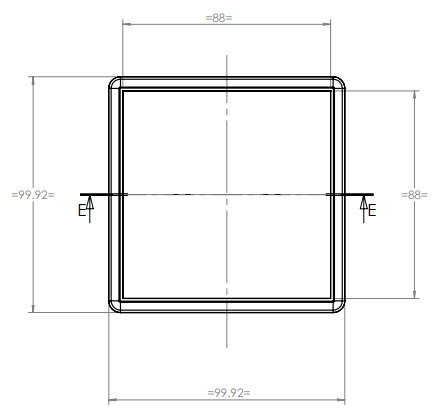
# PCB

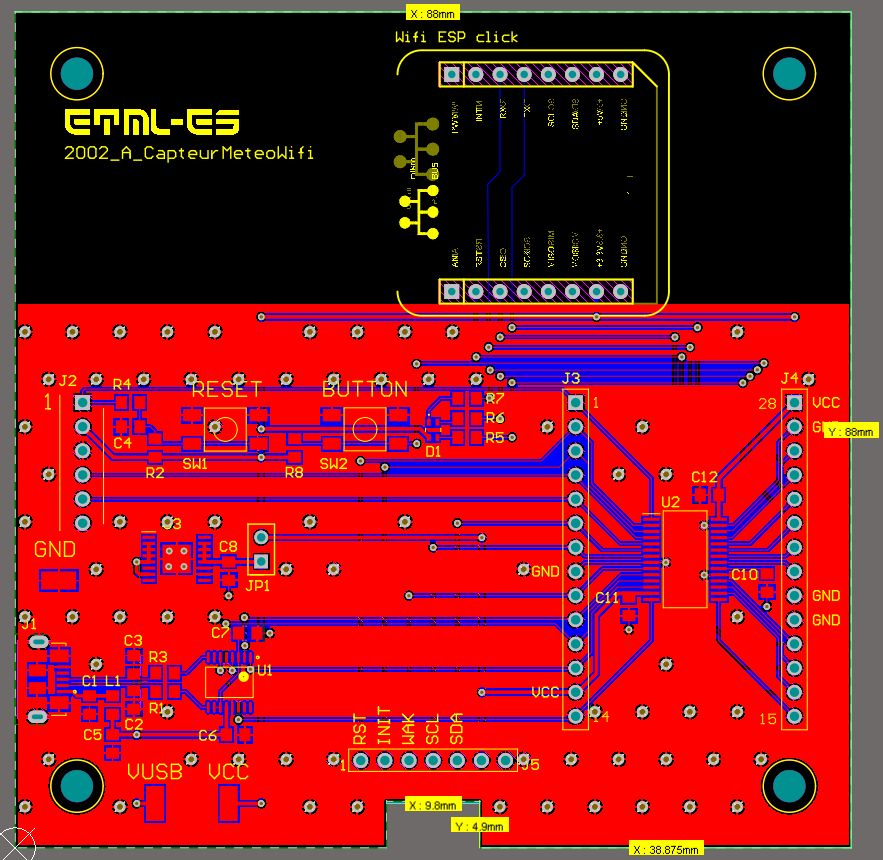
## https://www.camdenboss.com/products/img/CB1500-10SBK-KIT.jpg?mode=pad&width=405&height=333&bgcolor=fff&rnd=637281612040000000Contraintes

Ma première contrainte a été celle du boitier. En discutant avec mon client, nous avons pu trouver un boitier adapté à la taille du module Wifi qui à une hauteur de 20mm. Le boitier **CB1500-10 SBK-KIT** de chez Camdenboss.



Après avoir choisis le boitier, j'ai donc fait un PCB de 88mm sur 88mm avec une encoche de 9.71mm (9.8mm afin d'avoir un peu de marge).





## Routage

Routage en XY Largeur des pistes : 0.2mm Largeur des pistes d'alimentation : 0.5mm

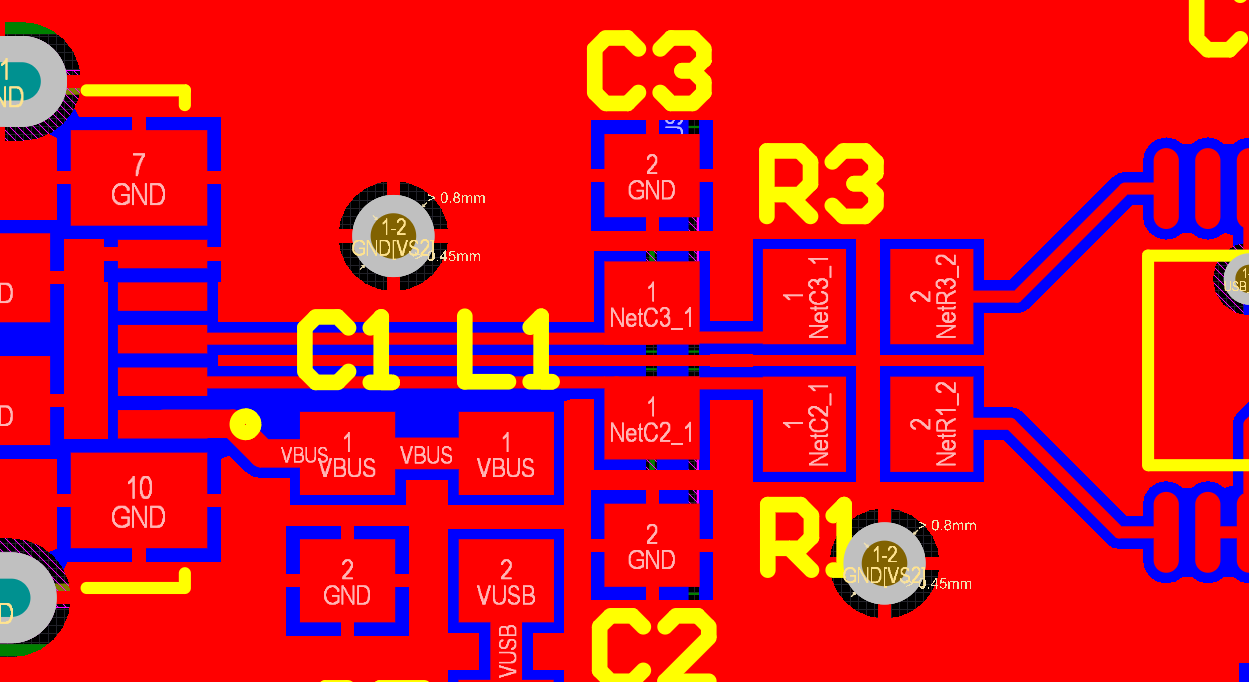
Vias signaux : diamètre total : 0.8mm, diamètre du trou : 0.45mm

Vias stitching : diamètre total : 1.27mm, diamètre du trou : 0.711mm

Les vias ont été mis en tented. Plan de masse sur le top et sur le bottom, avec des vias de stiching.

## Particularités de routage

Lors du routage des signaux USB, sachant qu'ils sont différenciel il ma fallu faire attention de garder les pistes les plus proche possible est avec la même longueur.



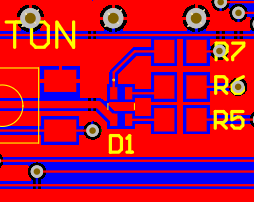
**D+**

**D-**

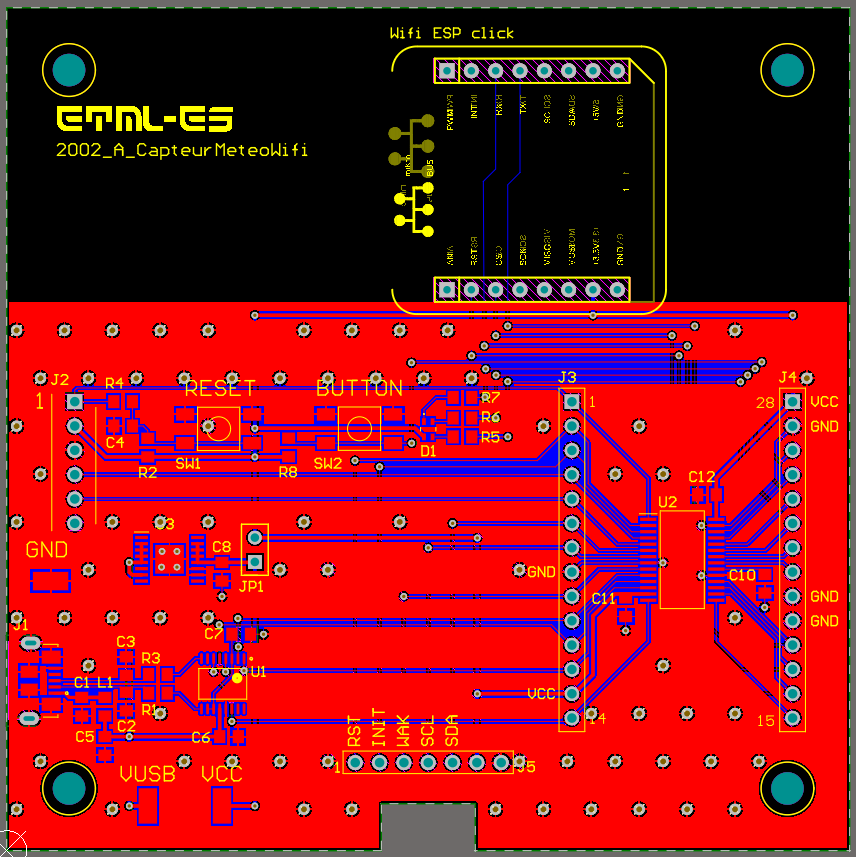
Pour le module Wifi, j'ai essayé de l'isolé le plus possible des autres composants et des plans de masse afin d'éviter le plus possible les perturbations sur le signal Wifi.

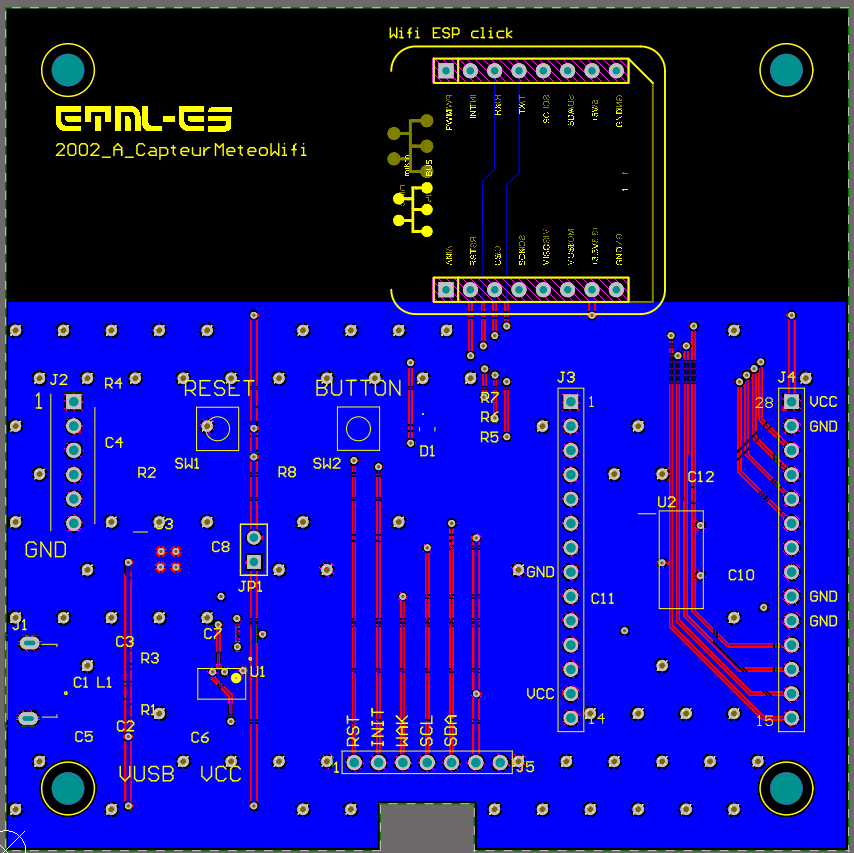


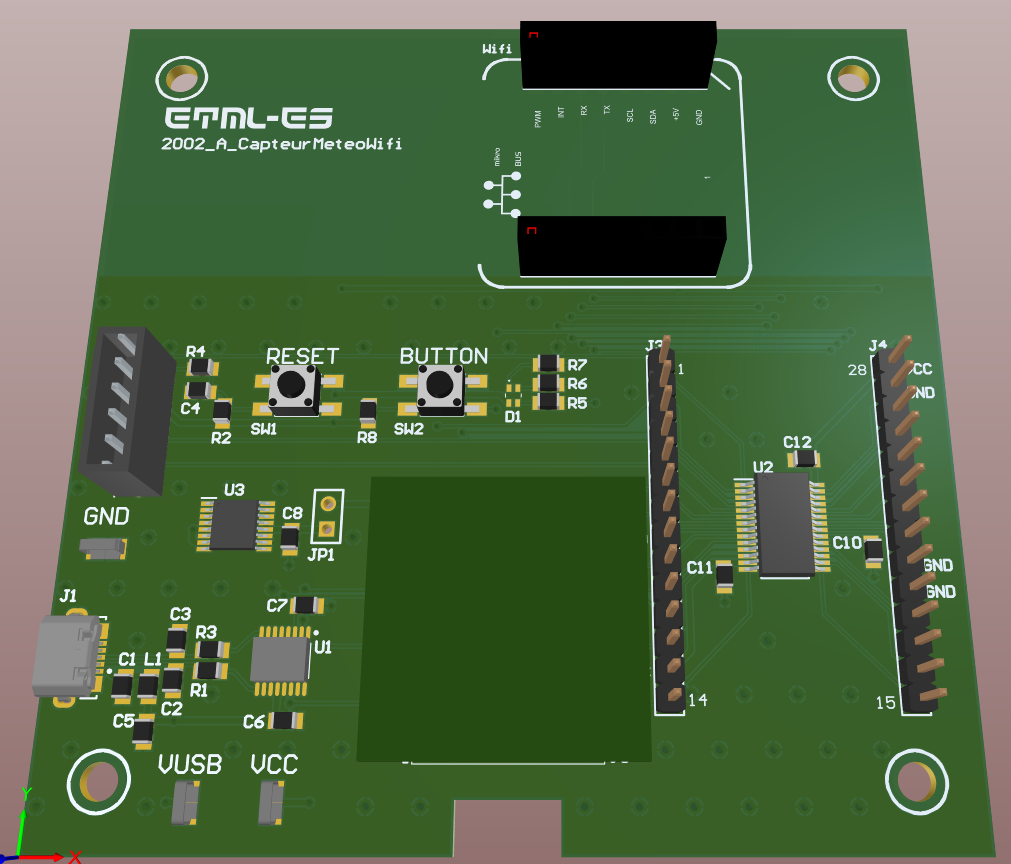
Il m'a été demandé de mettre la LED au centre de la carte afin d'y mettre un light pipe pour conduire la lumière



Voici donc le PCB (top, bot et une vue 3D).







**Module Capteur**

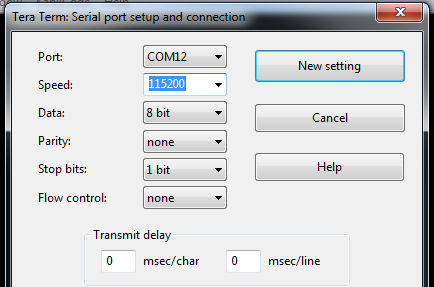
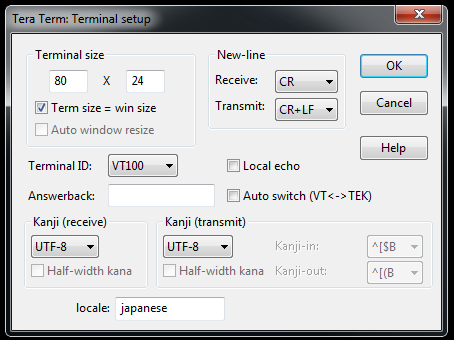
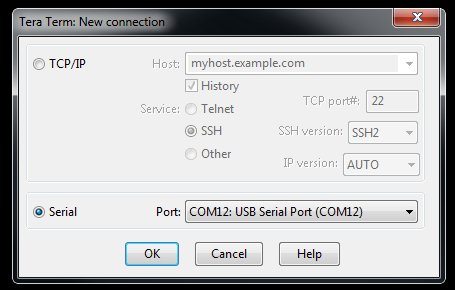
# Montage et tests

En attendant de recevoir mes composants ainsi que mon PCB, j'ai pu tester mon module Wifi depuis mon ordinateur. Grâce au câble TTL-232R-3V3 de chez FTDI, j'ai pu envoyer certaines commandes depuis le programme Tera Term au module Wifi.

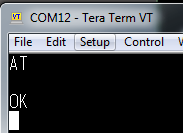
## Domoticz

**Réglages Tera Term**

Lorsque j'ai ouvert Tera Term, j'ai choisis sur quel port COM communiquer (COM12: USB Serial Port) puis dans les réglages du terminal, j'ai modifier les caractère à ajouter lorsque j'envoie (CR+LF) puis dans les réglages du port sériel, la vitesse de communication (115200 baud).



Pour commencer j'ai tester les commandes que j'ai listé lors du Design du module Wifi. La première (AT) m'a permis de vérifier la présence du module et de vérifier si la communication est fonctionnelle. Une commande mal fromulée est répondu par le message "ERROR".

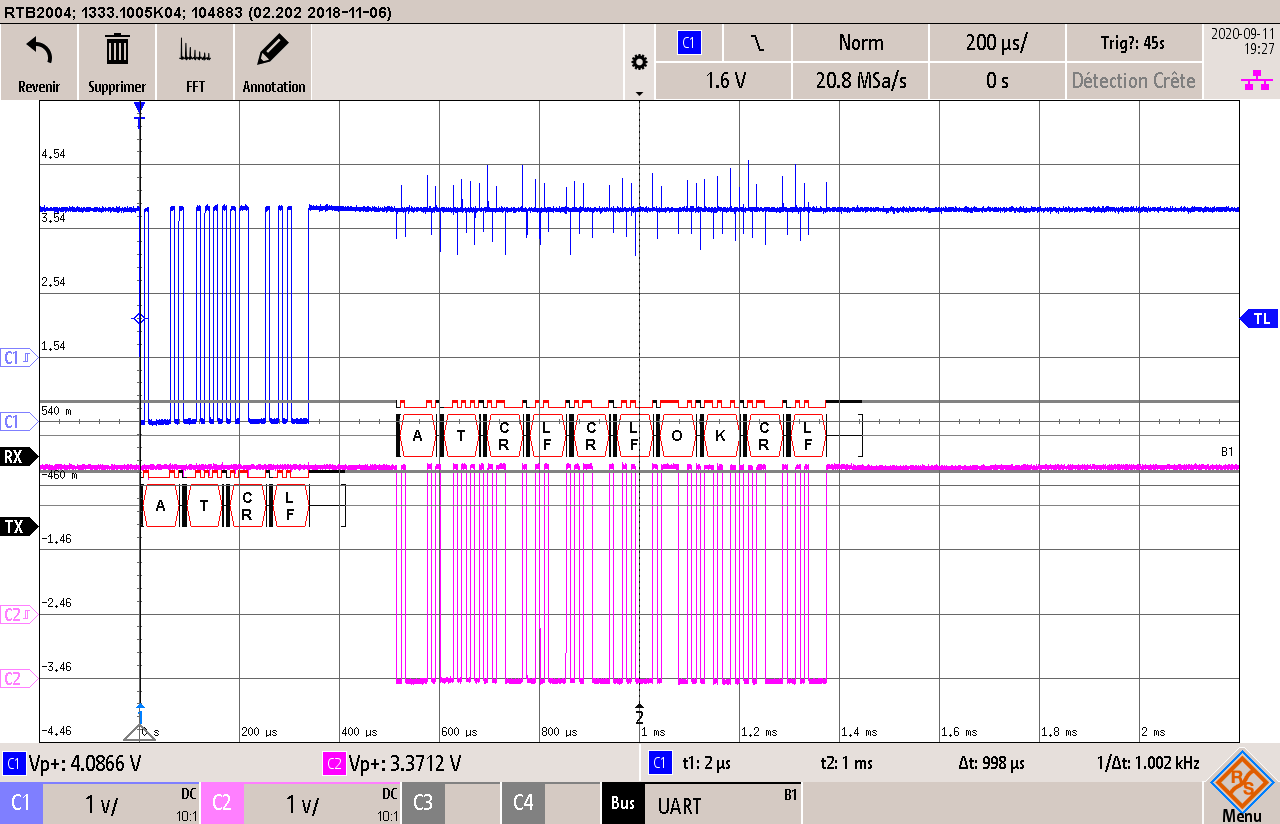
 

Commande

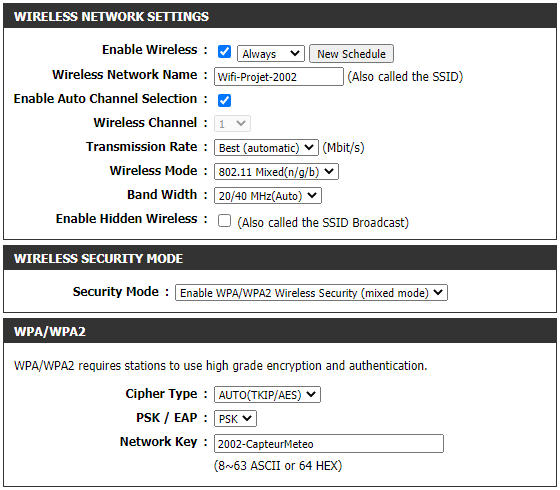
Réponse

Commande

Réponse



On constate que lorsqu'on envoie une commande, le module nous réponds en renvoyant la commande puis "**OK**". C'est le mode echo qui est activé et qu'il est possible de désactiver, mais pour mes tests, celà me sera utiles.

Un routeur m'a été fournit par l'ETML-ES et je l'ai utiliser afin de faire un point d'accès au réseau de l'école. En utilisant une Virtual Machine (Windows 10) connecté au même réseau que le routeur afin d'y accéder et j'ai pu régler l'SSID et le mot de passe du Wifi.

J'ai pu constater sur mon smartphone que le Wifi est bien visible et qu'il est possible de s'y connecter.

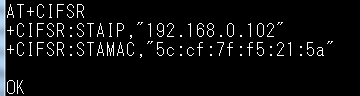
Maintenant que nous avons un Wifi sur lequel nous pouvons nous connecter, j'ai suivi la suite de commande donnée dans le datasheet d'exemple de commandes. Cette commande permet de programmer le module Wifi en mode Station.



Suite à ça, grâce à la commande suivante, j'ai pu connecter le module Wifi sur le réseau Wifi créer précédement. Et l'on peut voir les réponses renvoyées "WIFI CONNECTED", "WIFI GOT IP" et "OK" qui confirment que la commande à bien fonctionnée.

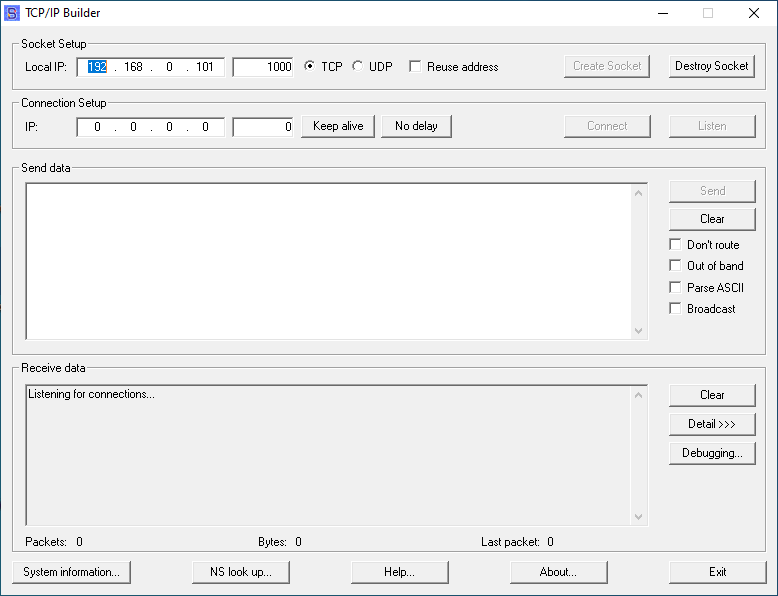


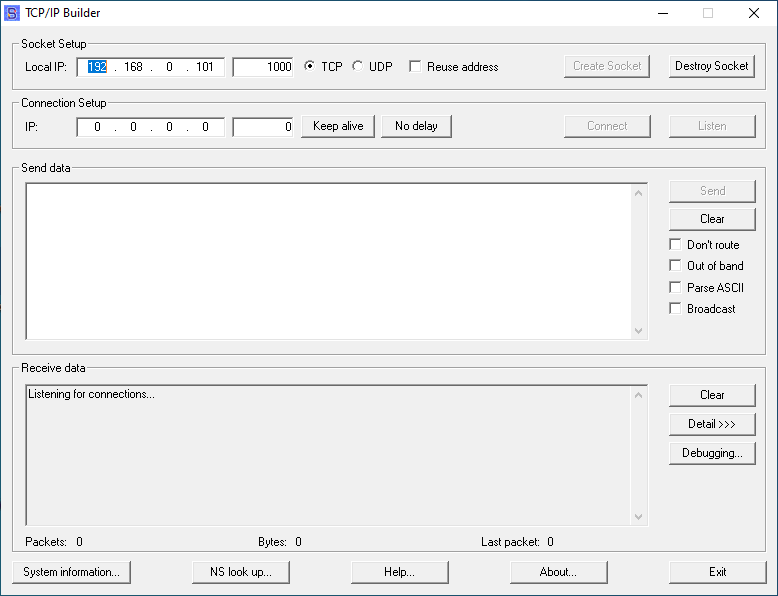
Avec cette commande, il est possible de récuperer l'adresse IP ainsi que l'adresse MAC du module Wifi.



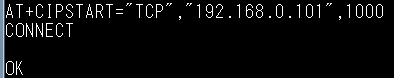
Les commande suivantes permettent de communiquer à une serveur, pour ce faire j'ai installer le programme TCP/IP Builder qui permet de créer un serveur sur lequel je vais me connecter avec mon module Wifi.

Afin de créer ce serveur grâce au programme, il faut configurer l'adresse IP ainsi que le port sur lequel se connecter. Pour ma part j'ai chosisis l'addresse IP de ma machine et le port 1000. Suite à cela il faut presser sur "Create Socket" puis "Listen".

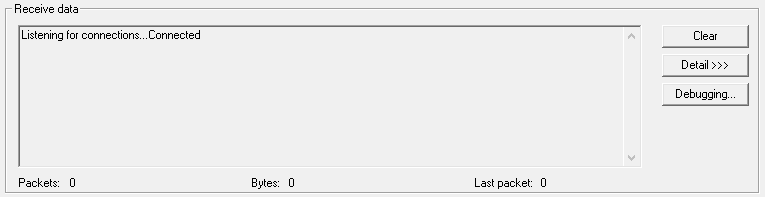




Maintenant que le serveur est créer, la commande suivante permet de nous y connecter et nous pouvons voir que sur le programme l'adresse IP du module Wifi ainsi que dans Receive data que le module c'est connecter.



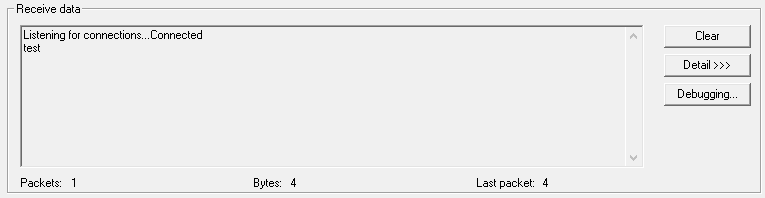




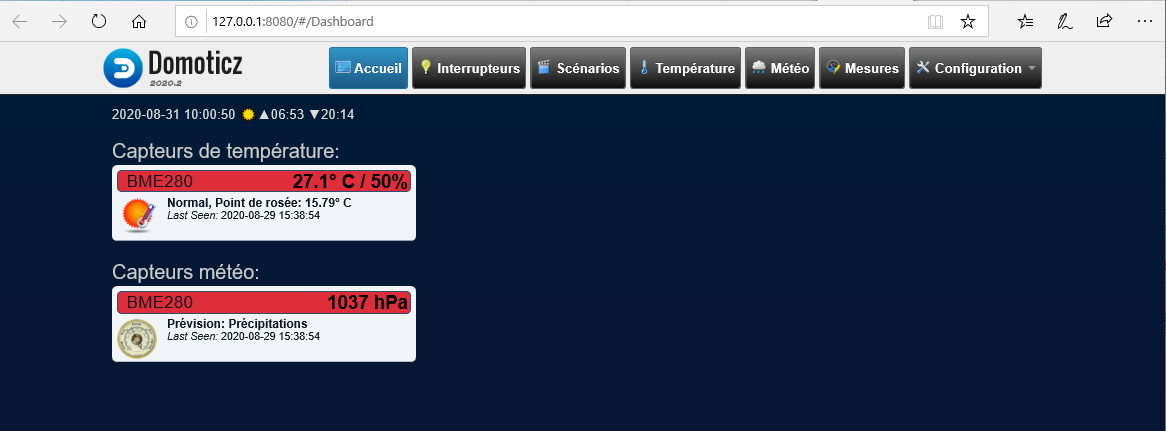
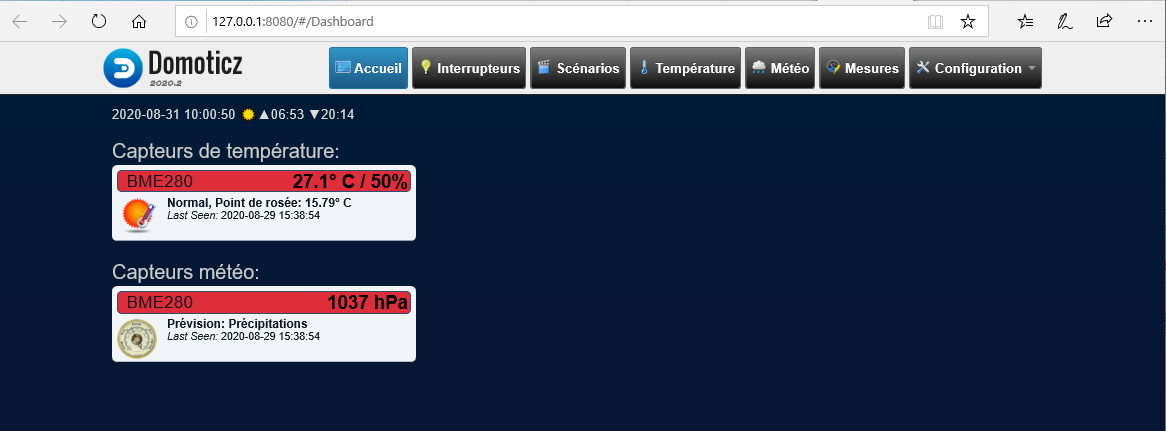
Pour finir, la commande suivante permet d'indiquer le nombre de byte que l'on veut envoyer puis d'écrire ce que l'on veut envoyer. Dans mon cas, j'ai envoyé le mot "test" et nous pouvons voir qu'il a bien été reçu par le serveur.



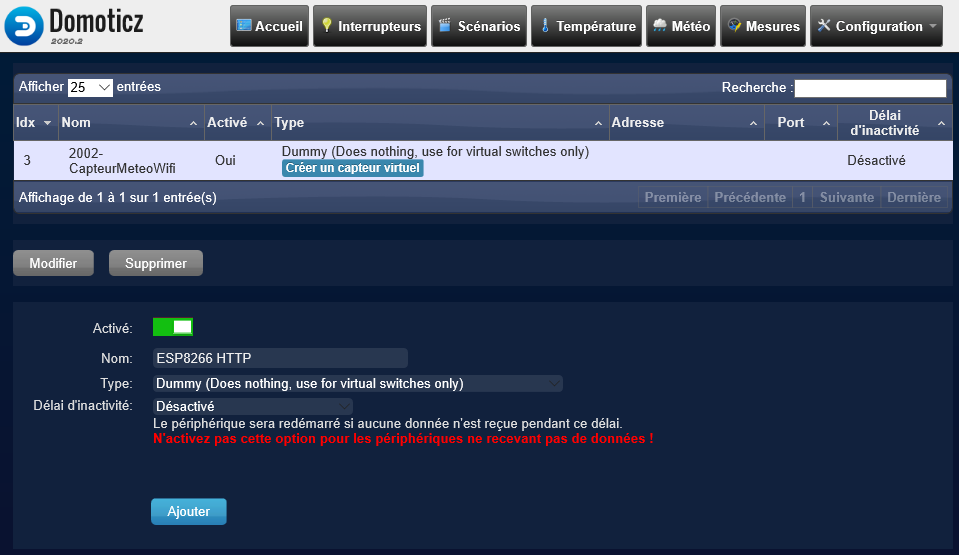
test

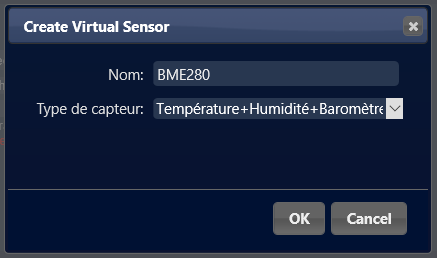


Maintenant que j'ai pu tester toutes les commandes qui me seront utilses, j'ai installer Domoticz sur mon ordinateur. Il est possible d'accéder à l'interface graphique en me connectant à l'adresse IP 127.0.0.1 sur le port 8080 dans un navigateur.



Afin d'ajouter un capteur, il faut aller dans "Configuration"puis "Matériel". Pour ma part, j'y ai ajouté un nouveau matériel nomé "2002-CapteurMeteoWifi" dans lequel j'ai ajouté un capteur virtuel, le BME280 en indiquant le type de capteur (Température, Humidité et pression).





Une fois le capteur créer, dans l'onglet "Configuration" puis "Dispositifs", on retrouve le capteur récemment créé. On peut voir dans la colonne "Données", que pour l'instant il fait 0°C, 50% d'humidité et 1010hPa (valeurs par défaut).



Afin de modifier ces valeurs, il suffit de créer une adresse URL qui envoie une requête HTTP écrite en JSON. Il est possible de la trouver sur le wikipédia[[1]](#footnote-1). L'URL se compose de l'adresse IP Domoticz suivit du port (127.0.0.1:8080), **idx=1** (le numéro du capteur dans notre cas le numéro 1 comme on peut le voir sur l'image précédente), **value=27.10;31.10;0;1024.03;4** (la première valeur représente la température, la deuxième l'humidité, la troisième le niveau d'humidité, la quatrième la pression et la cinquième la prévision météorologique).



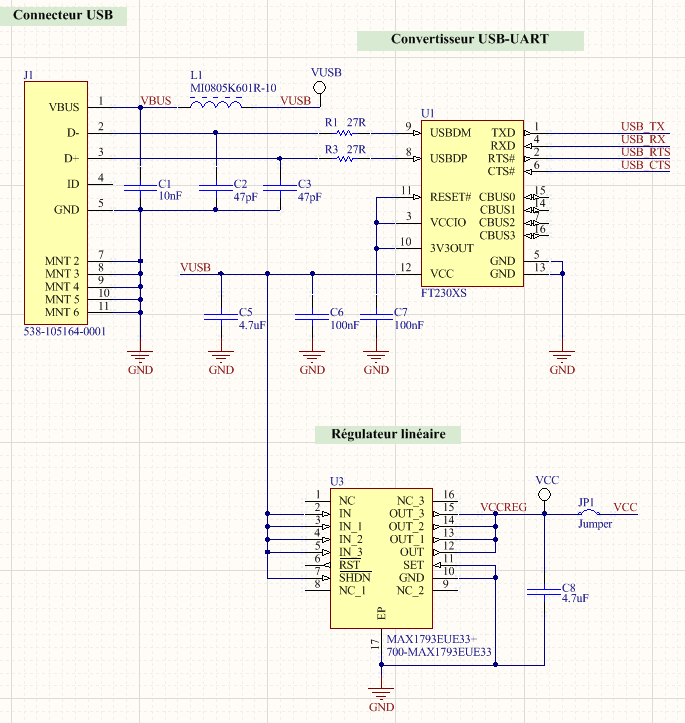
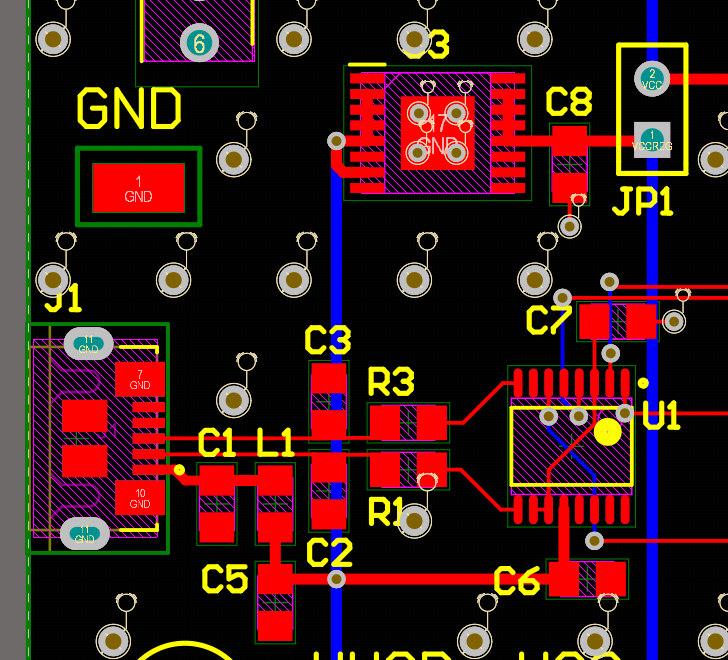




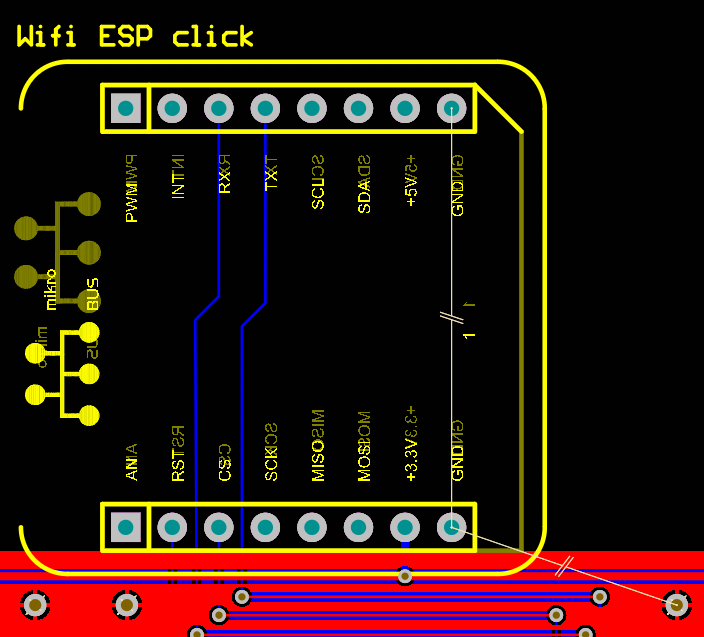
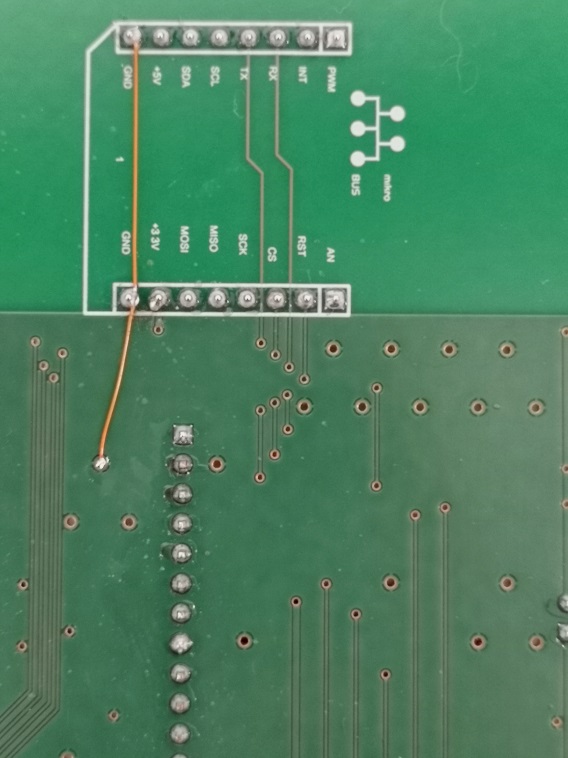
J'ai fait tout ces tests car dans mon projet fonctionnera sous le même principe. Il aura un serveur sur lequel "Domoticz" sera héberger et grâce à des requettes HTTP, je pourrai envoyer les données de mon capteur.

## Alimentation

Une fois le montage terminé, j'ai commencé mes tests sur la partie alimentation, j'ai commencé en alimentant la carte avec un connecteur microUSB. Grâce aux test points que j'ai placé après du connecteur micro USB et le régulateur linéaire, j'ai pu vérifier si la tension d'entrée est bien à 5V et la sortie du régulateur est bien de 3.3V.

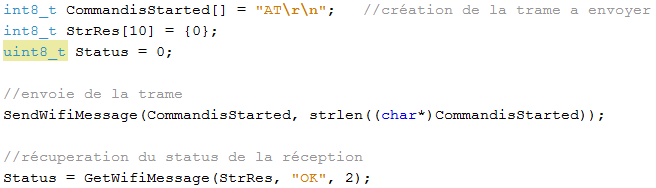


Une fois le bon fonctionnement de mon alimentation vérifié, j'ai placé le jumper afin que tous les modules soient alimenté. Lors de ces tests j'ai remarqué que les pins GND du module Wifi n'étaient pas relié au plan de masse. J'ai les ai donc reliées à la masse à l'aide d'un file de couivre.

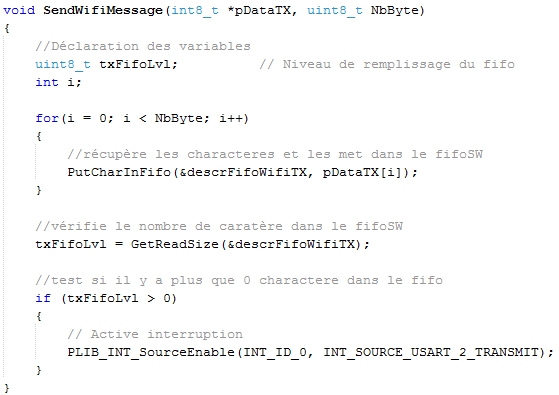
 

## Test de la communication UART

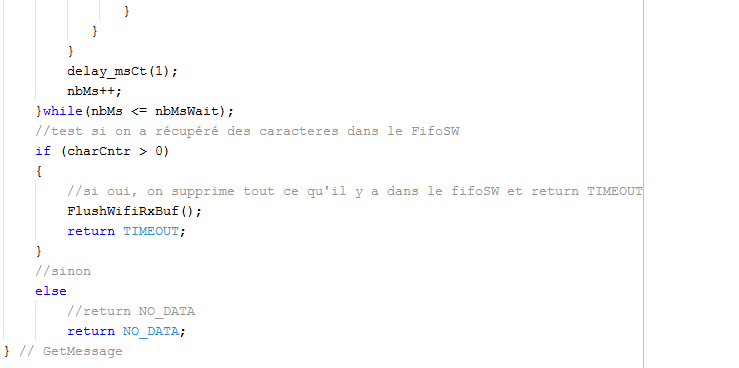
Une fois tout les composants monté, j'ai testé ma communication UART entre le module Wifi et microcontrôleur. Pour ce faire j'ai pluggé le module Wifi ESP click aux broches sur mon board puis dans le software, j'ai tester d'envoyer la commande "AT", qui pour rappel permet de détecter la présence du module.



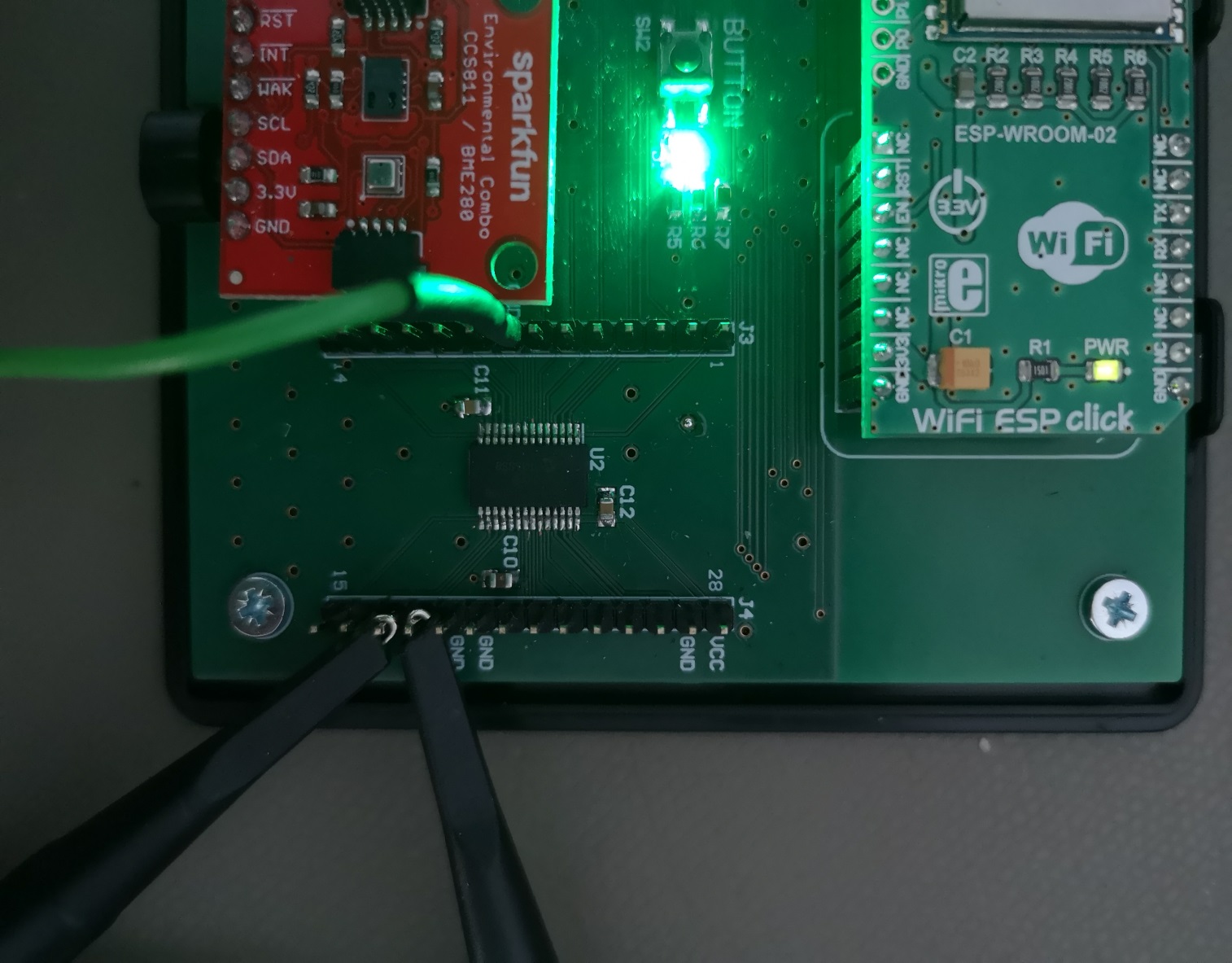
Pour la fonction d'envoie de message, c'est une fonction que l'on avait déjà utilisé auparavant. Elle permet simplement de récuperer les charactère de CommandisStarted[] et de les mettre dans le fifoSW.



La fonction de réception est aussi une fonction que l'on a déjà utilisé mais qui à été un peu modifiée. Elle récupère les caractères reçus dans le premier paramètre, vérifie si on trouve le mot voulu et le troisième paramètre est un timeout.



### Schéma de mesure



**CH1**

**Pin 18, UTX2**

**Trig : CH1**

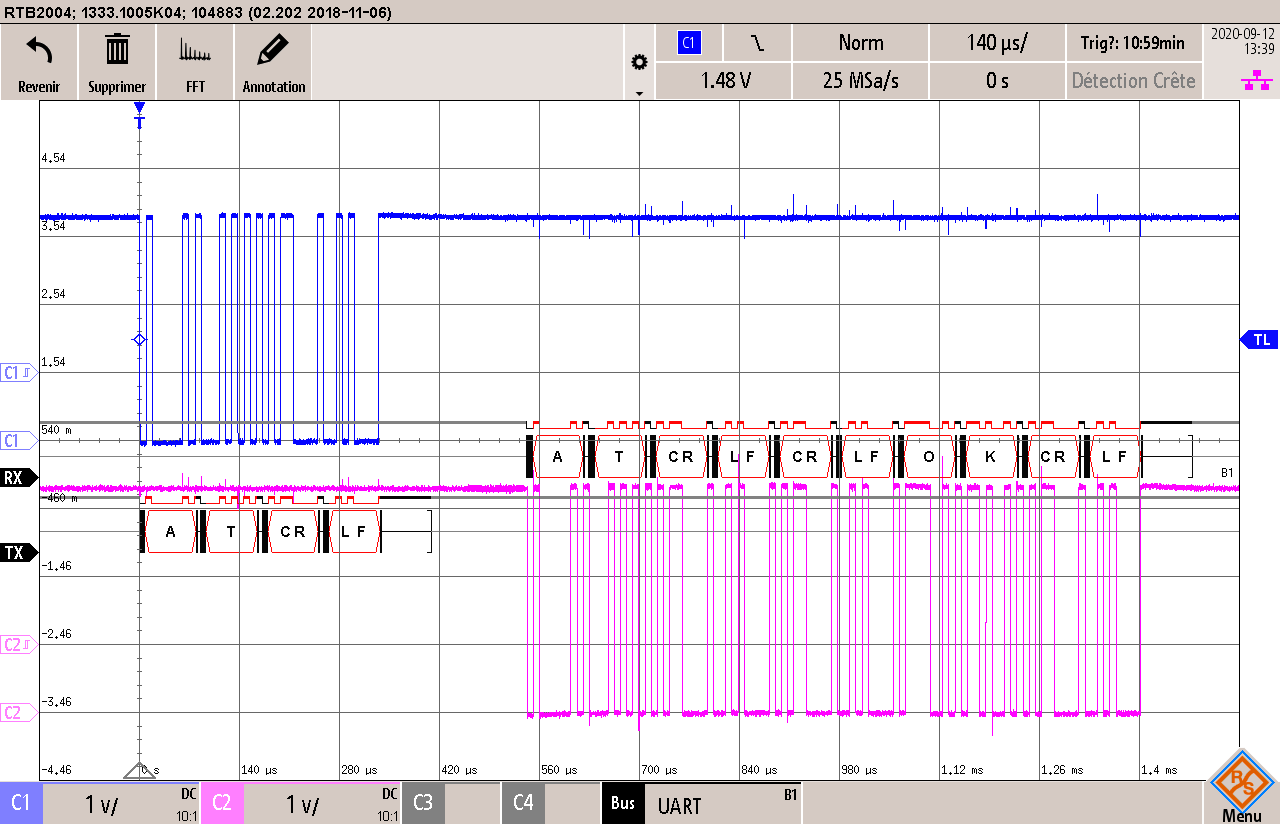
**Flanc descendant**

**Protocole : UART**

**CH2**

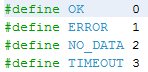
**Pin 17, URX2**

### Mesure



On peut voir sur l'oscillogramme que la trame crée est correctement envoyée et que la trame de réponse est l'echo de ce que j'ai envoyé avec un "OK" comme nous avons pu le voir lors de mes tests sur Tera Term.



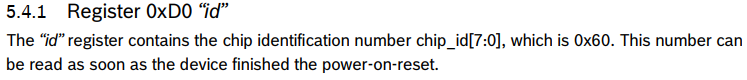
Et ici on voit bien aussi que la valeur de retour de la fonction de réception correspond à 0. Ce qui confirme qui confirme que le "OK" à correctemetn été reçu.

## Test de la communication I2C

Une fois tout les composants monté, j'ai testé ma communication I2C. Pour ce faire j'ai pluggé le module SEN-14348 aux broches sur mon board puis dans le software, j'ai tester le lire un registre en particulier le 0xD0, celui du chip ID.



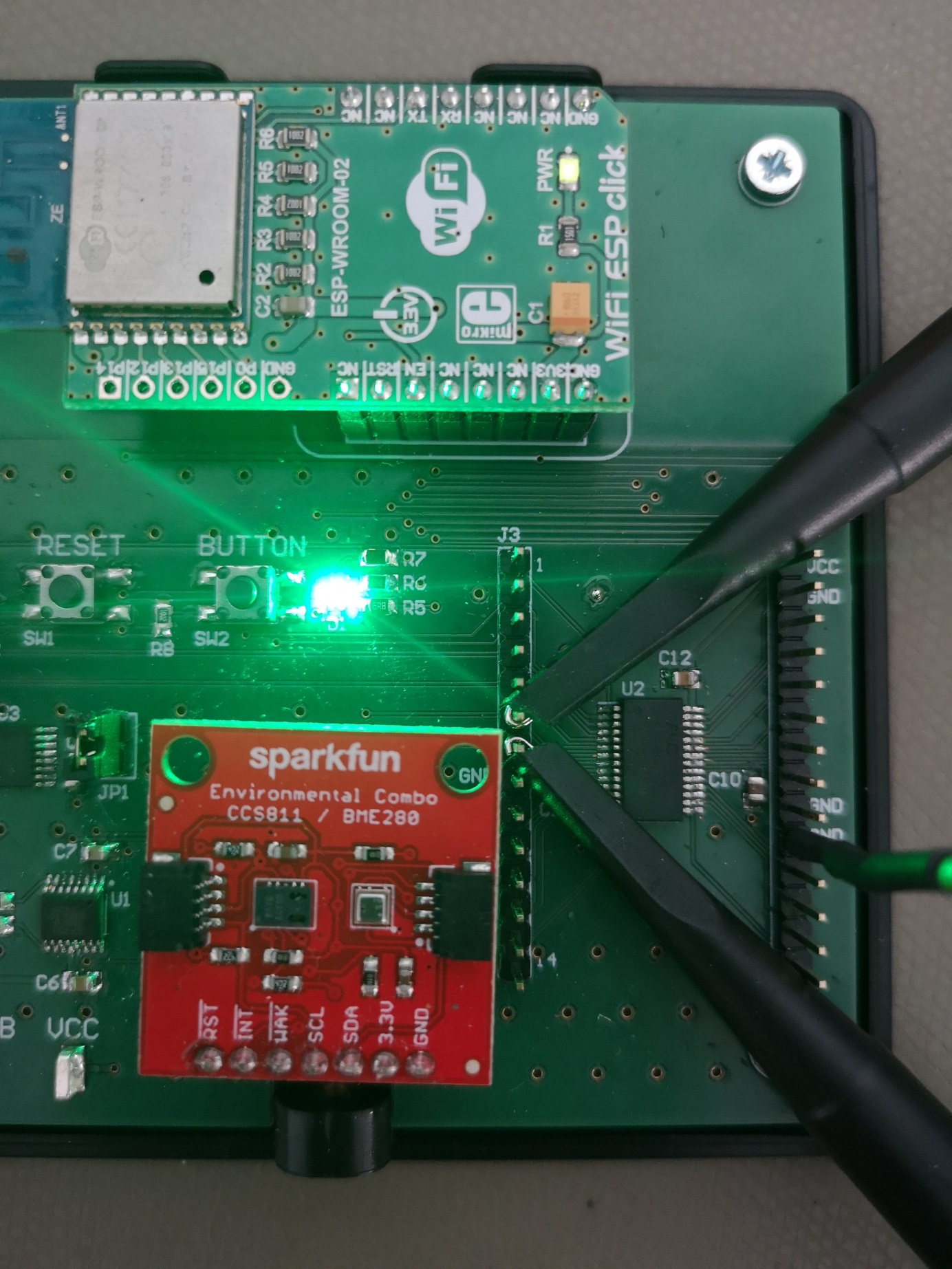
L'avantage de lire ce registre, c'est que l'on connais ça valeur et donc vérifier tout de suite si la communication fonctionne de plus sachant que lorsque vient lire un registre en I2C on doit forcémment écrire, on peut tester directement les l'écriture et la lecture.



J'ai utilisé une fonction de lecture d'un registre de 8bits avec en paramètre l'adresse du BME280 (0x77) et l'adresse du registre (0xD0).



### Schéma de mesure



**Trig : CH1**

**Flanc descendant**

**Protocole : I2C**

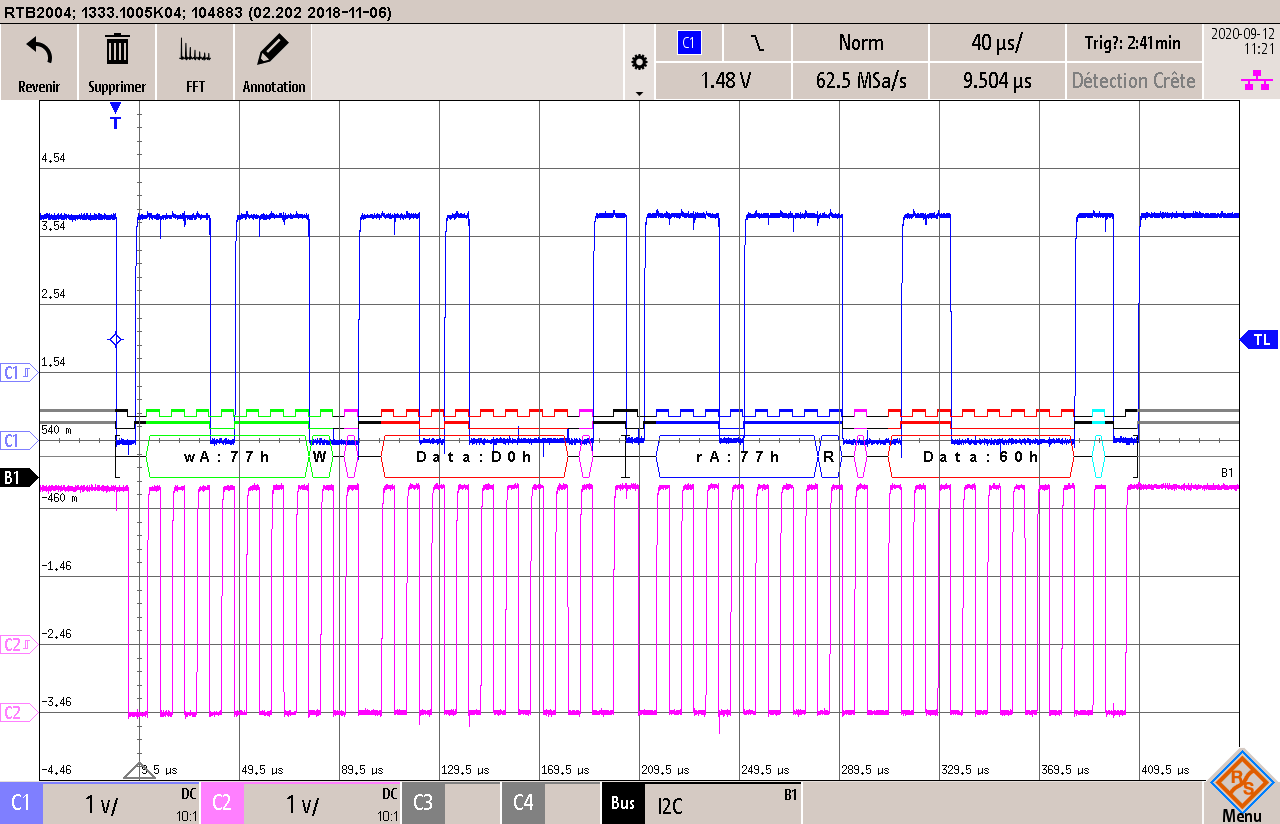
**CH2**

**Pin 7, SCL2**

**CH1**

**Pin 6, SDA2**

### Mesure



Voici le résultat des signaux I2C, on peut voir que l'on envoie bien l'adresse du device (0x77) suivis d'un bit à 0 pour spécifier que l'on veut écrire, puis l'adresse du registre (0xD0). On envoie ensuite à nouveau l'adresse du BME280 mais cette fois suivis d'un 1 et l'on voit que l'on reçois bien la valeur 0x60, qui correspond bien à ce que l'on attendait.



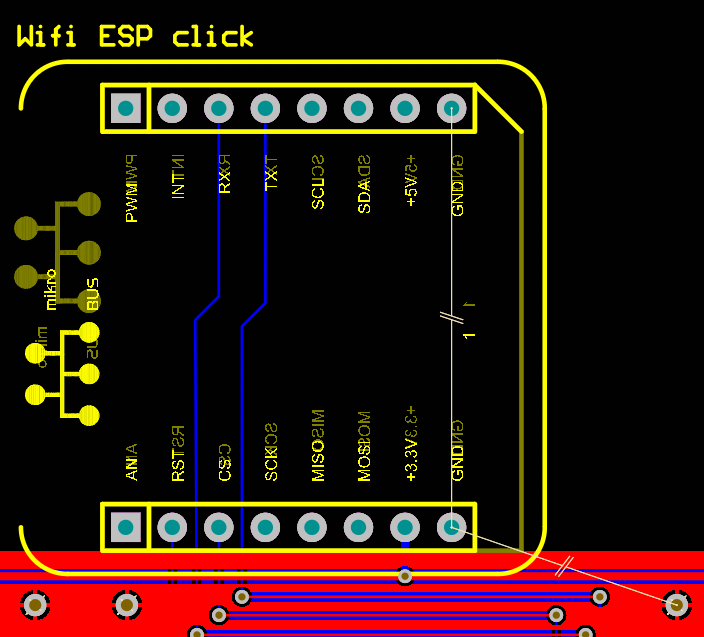
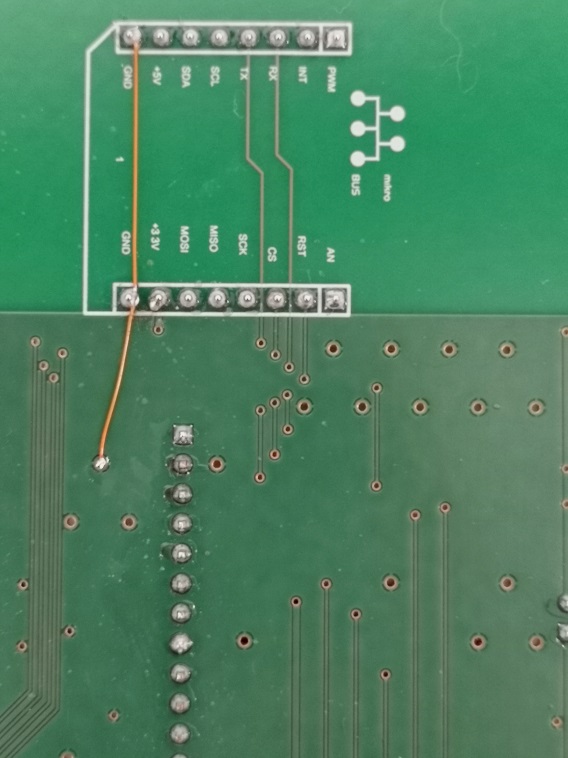
Et voici on voit bien aussi que la valeur reçu par la fonction est bien la même. Ce qui confirme qu'elle fonctionne correctement.

# Erreurs

Durant la création de mon PCB, j'ai fait plusieurs erreurs. Voici donc mes erreurs et comment je les ai corrigés.

## Première erreur

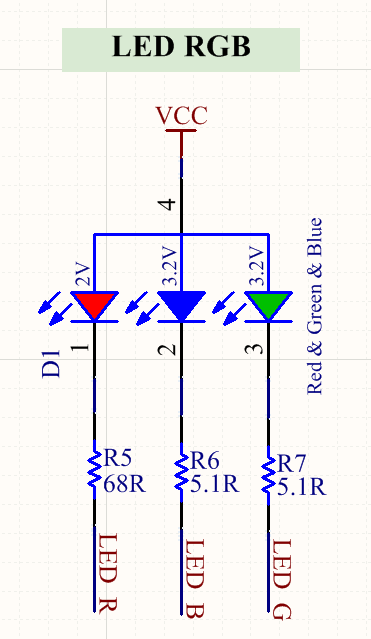
Comme j'en ai parlé durant mes tests de l'alimentation, lors de le routage du PCB, j'ai oublié de connecter les pins GND du module Wifi au plan de masse. J'ai donc corrigé cette erreur en connectant un fil de cuivre au plan de masse.

## Deuxième erreur

Lors de l'étape du Design, j'ai récupéré le schématique de la LED RGB d'un ancien projet que j'avais fait. Mais lors de la commande de composant, la LED en question n'était plus disponible et j'en ai pris une équivalente, sans vérifier si le schématique/footprint correspondait.

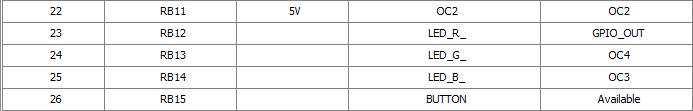
**Ancien schématique Nouveau schématique**

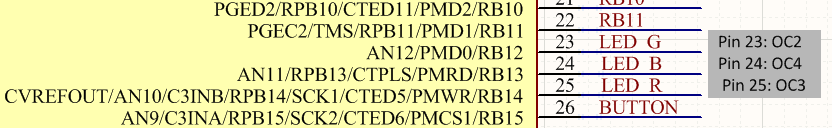
Heureusement la pin 4 est l'anode pour les deux schématique, ce qui au final n'a pas posé trop de problème (réoganisation des pins au niveau software).

## Troisième erreur

Lors de la selection des pins durant la partie Design, j'ai pensé connecter chaque catodes sur des sorties OC du microcontrôleur, ce qui permetterait d'avoir une meilleure gestion de l'allumage des LEDs. Malheurseusement, la pin 23 du microcontrôleur n'est pas un OC2, par contre la pin 22 l'est.



**OC2**

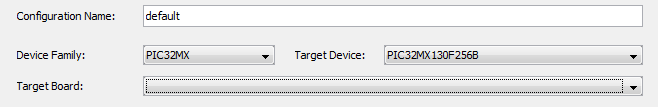


Finalement afin de corriger toutes ces erreurs, j'ai créé une version B du board (schematique et PCB). Permettant à l'avenir de ne plus faire ces erreurs.

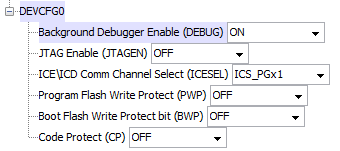
# Software

## Réglages Harmony

Pour cette partie, j'ai commencé par créer un projet "Microchip Embedded" "32-bit MPLAB Harmony Project" compatible avec le PIC32MX130F256B.

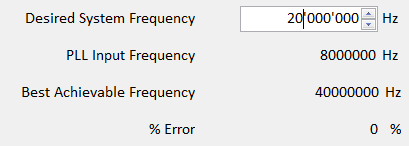
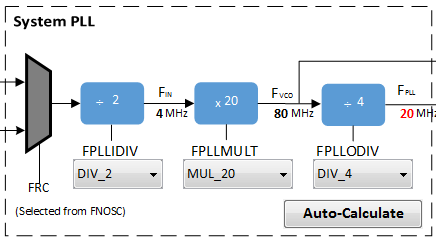


Directement dans Harmony Configuration, j'ai commencé par modifier le DEVCFG0 afin d'activer le mode Debug.



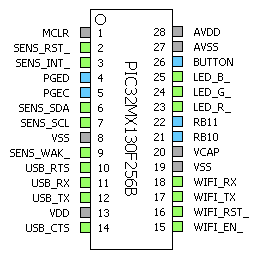
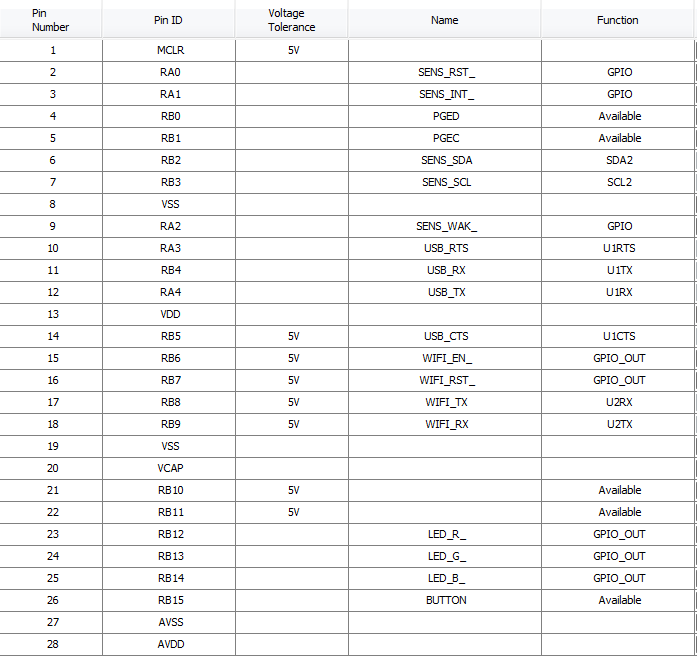
### Clock diagram

Puis j'ai réglé la fréquence de la clock, ce micro peut aller jusqu'à 50MHz sous certaines conditions, mais j'ai choisis de le faire fonctionner à une fréquence de 20MHz avec le bouton "Auto-Calculate" (cette fréquence peut être changer à tout moment surtout en fonction de la consommation de notre carte).

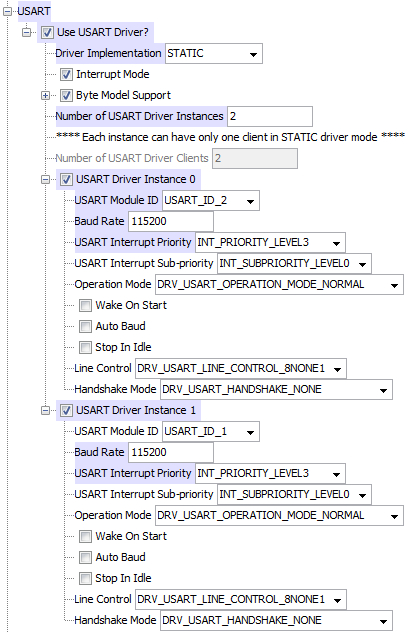
### Pin settings

Je suis ensuite allé dans le Pin Settings afin d'assigner le nom des pins ainsi que leurs fonctions.

### Configuration des périphériques

Pour la communication en I2C, nous n'allons pas la configurer grâce aux drivers de Harmony



Pour les UARTs, voici les réglages effectués.

La première instance représente l'UART utilisé pour la communication en le microcontrôleur et le module wifi.

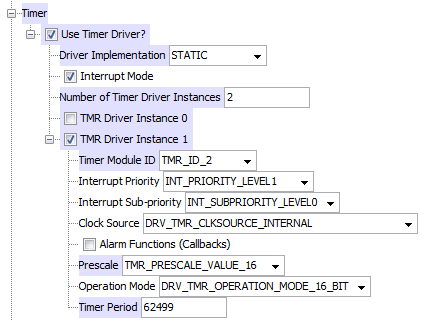
La deuxième instance représente la communication entre le microcontrôleur et l'ordinateur.

Les deux interruptions fonctionnent exactement de la même manière.

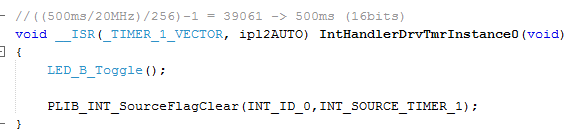




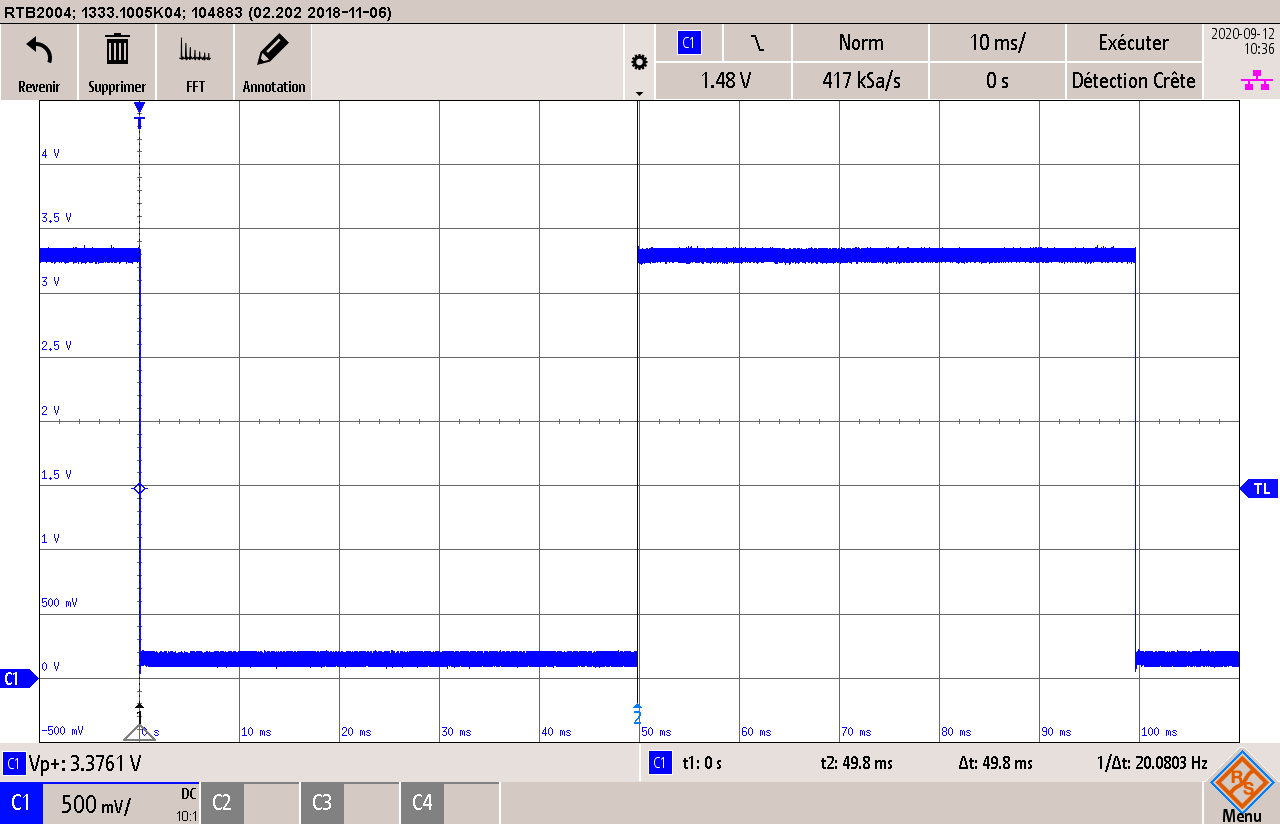
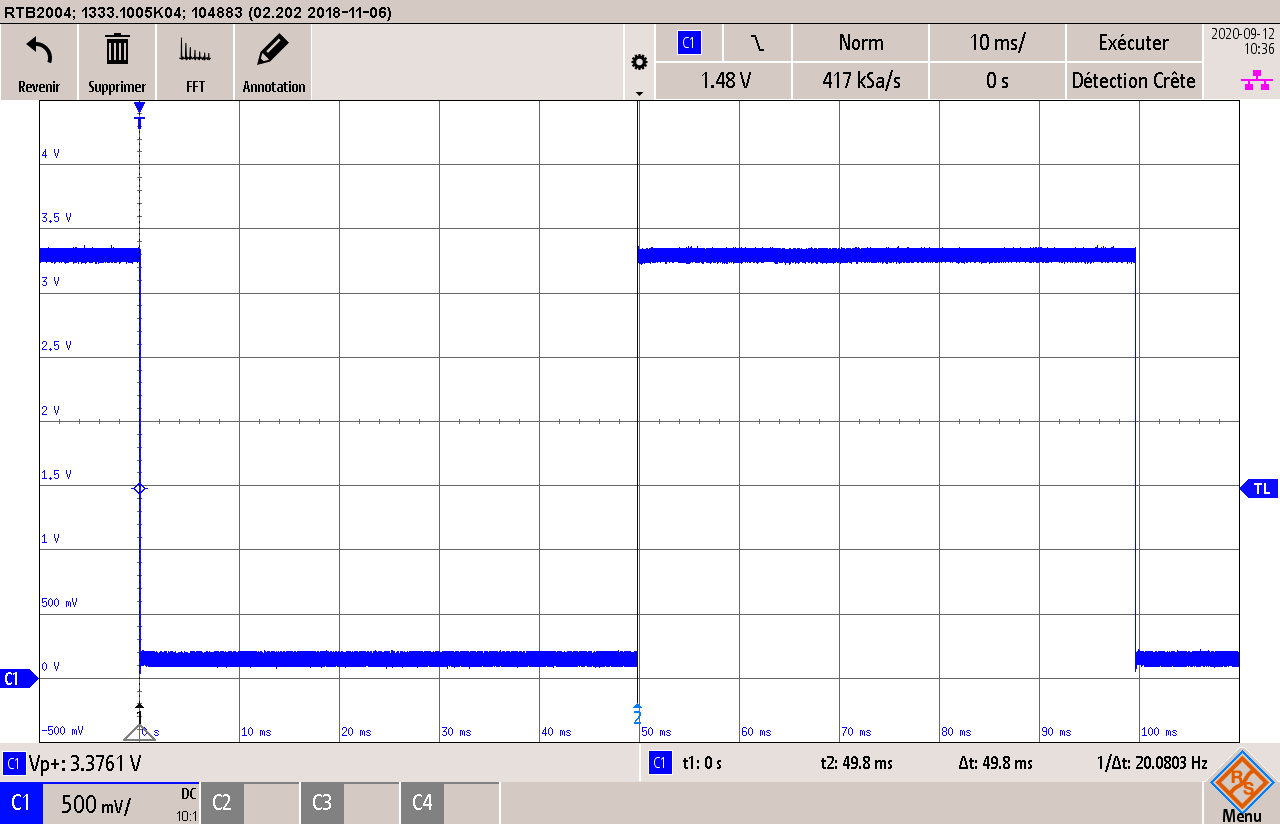
### Timer et interruption

J'ai activé le Timer et son interruption (toute les 50ms). La valeur de recharge ainsi que le prescaler ont étés calculé de la manière suivante :

Dans l'interrpution de ce même timer, je fais toggle la LED.



On constate que l'on a bien une période de 100ms, ce qui correspond bien au faite que l'on toggle la LED toutes les 50ms.



## Organigrammes du programme



## Description des fonctions

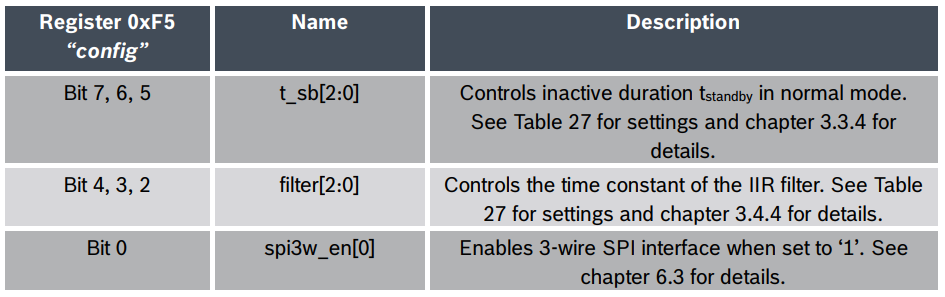
Pour chacune des fonctions crées, une explication est suivie mais pour plus de détail sur le fonctionnement de ces dernière, se fier aux annexe concercant les fichiers ".c".

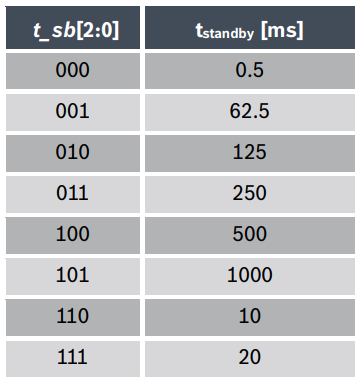
### BME280.c

bool BME280\_Init(void)

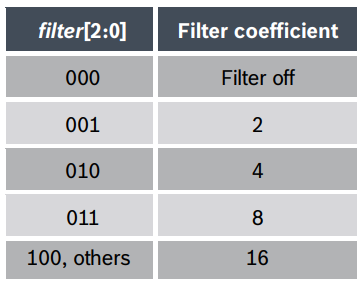
Cette fonction permet d'initialiser le BME280.

Pour initialiser le BME280, il y a 3 registres à écrire afin d’avoir les mesures voulues.



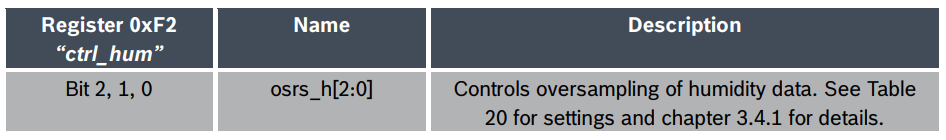
Ce premier registre permet de configurer le temps de standby du BME280, du niveau de filtrage et d’activer la communication SPI.

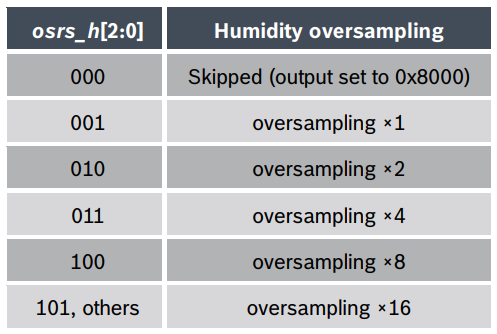
Sur ce premier tableau, nous pouvons voir les temps de standby, dans mon cas j’ai choisis de mettre un temps de 1 seconde et donc écrire les trois dernier bits du registre à 101. Le BME280 faira ces mesures toutes les secondes et se mettra en standby, mais je ne viendrai lire les registres que toutes les 10 minutes.



Pour le deuxième tableau, nous pouvons voir le niveau de filtrage. Dans mon cas, j'ai désactivé le filtrage des données et j'ai donc mis les trois bits à 0.

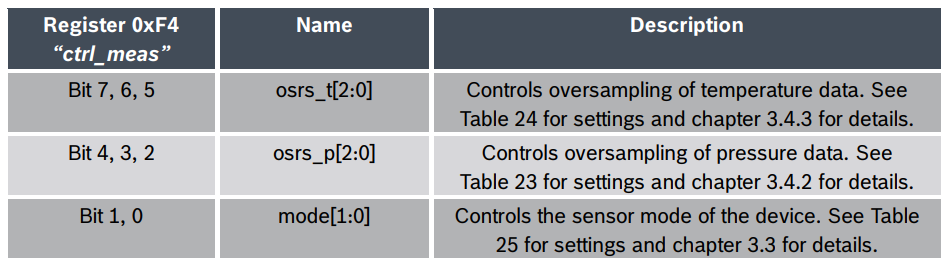
Pour le premier bit du registre, dans mon cas vu que je n’ai pas besoin de la communicatin SPI, j'ai donc mis le bit à 0.

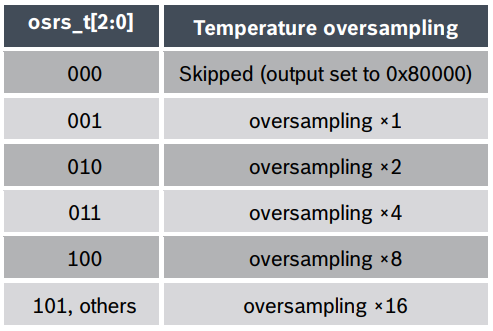


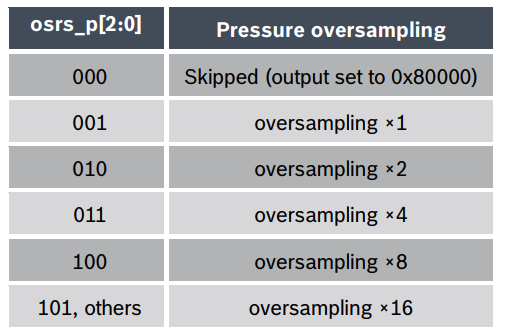


Ce deuxième registre permet de controler l'oversampling pour les données de l'humidité. Dans mon cas, j'ai d'avoir un oversampling x1, j'ai donc mis les trois bits à 001.

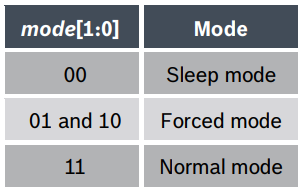
Il est impératif d'écrire le registre d'écrire ce registre avant d'écrire le registre ctrl\_meas.



Ce dernier registre permet de configurer l'oversampling de la température et de la pression ainsi que le mode de fonctionnement du BME280.

Comme pour l'oversampling de l'humidité, j'ai choisis de prendre x1 et j'ai donc mis les trois derniers bits à 001.

Pour la pression, j'ai décidé de mettre les bits 001, pour avoi run oversampling x1.

Le BME280 a trois modes de fonctionnement, le mode sleep, le mode forced et le mode normal.

Lorsque le mode sleep est activé, le device plonge en standby continu et aucune mesure n'est faite.

Lorsque le mode forced est activé, le device fait une première mesure puis se met en sleep mode jusqu'à se qu'on lui renvoye le mode forced.

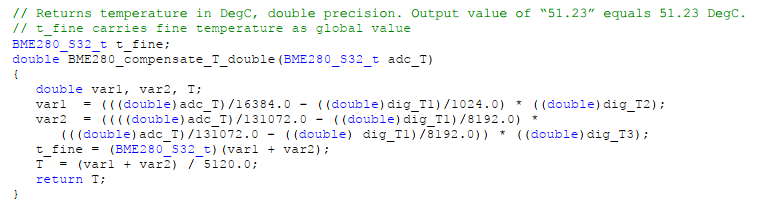
Losrque le mode normal est activé, le device fait des mesures sycliquement en fonction du temps de stanby.

void BME280\_ReadCoefficents(void)

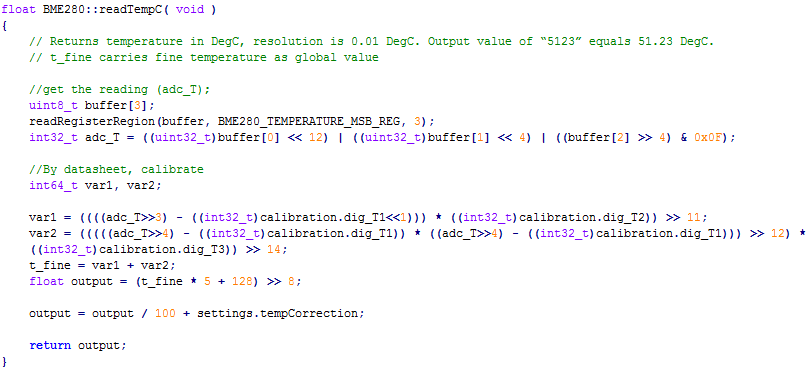
Cette fonction permet de lire les coéfficents de calibration interne au BME280 (obligatoire si l'on veut avoir la température, l'humidité et la pression).

float BME280\_readTemperature(void)

Cette fonction permet de lire les valeurs dans les registres de température puis de les convertir en température grâce aux coefficents. Pour faire cette fonction, je me suis aidé de l'expemple donné dans le datasheet du BME280 ainsi que celui fournit par SparkFun.



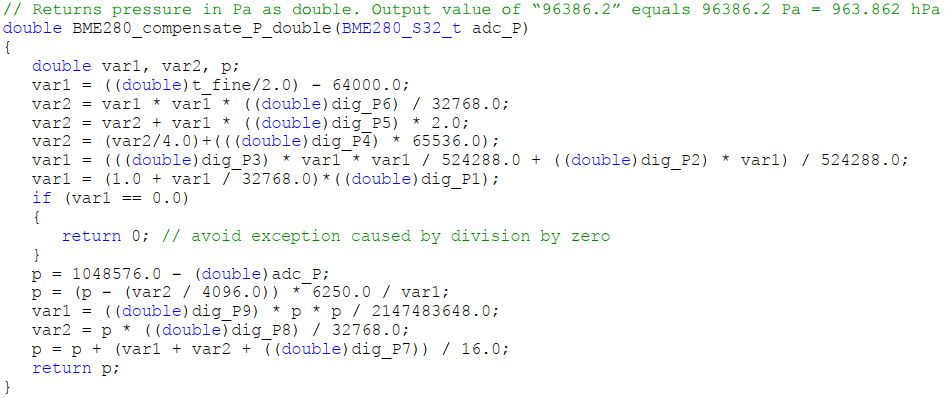
*Exemple donné par Bosch Sensortech*



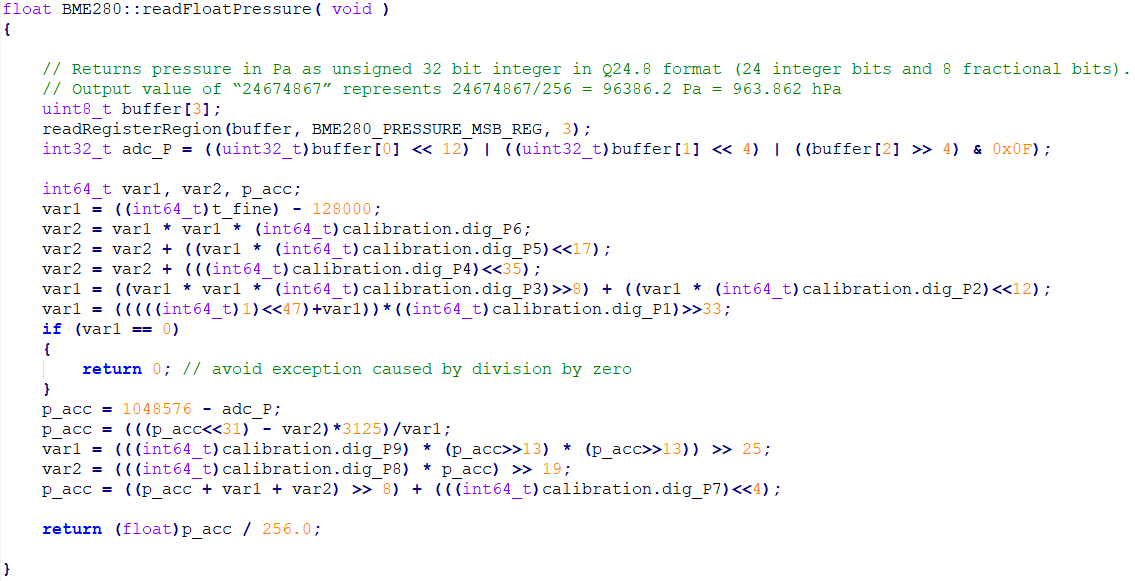
*Exemple donné par SparkFun*

float BME280\_readPressure(void)

Cette fonction permet de lire les valeurs dans les registres de pression puis de les convertir en pression grâce aux coefficents de calibration. Pour faire cette fonction, je me suis aidé de l'expemple donné dans le datasheet du BME280 ainsi que celui fournit par SparkFun.



*Exemple donné par Bosch Sensortech*

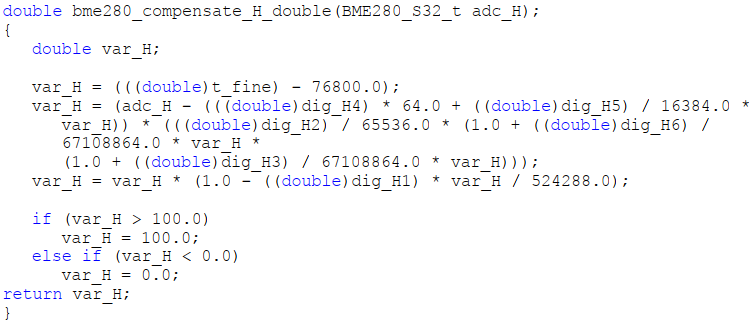


*Exemple donné par SparkFun*

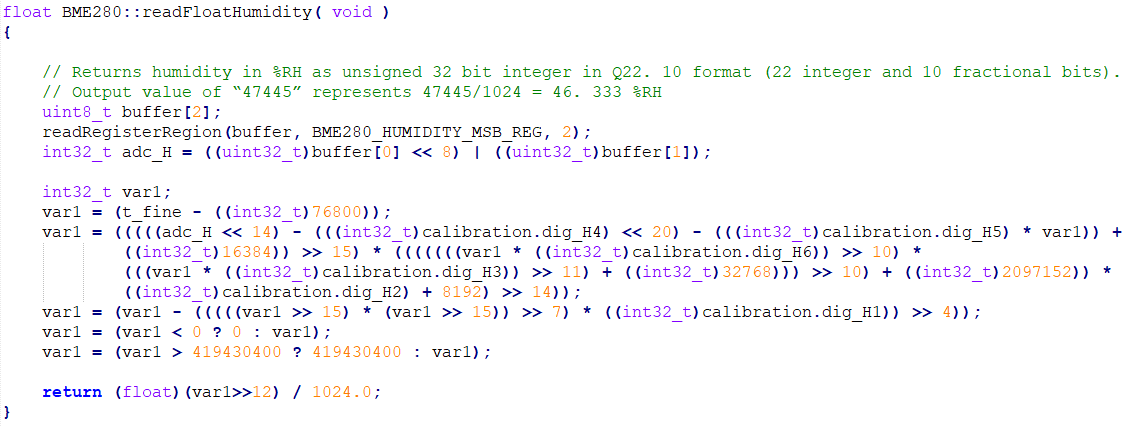
float BME280\_readHumidity(void)

Cette fonction permet de lire les valeurs dans les registres de l'humidité puis de les convertir en pression grâce aux coefficents.

Pour faire cette fonction, je me suis aidé de l'expemple donné dans le datasheet du BME280 ainsi que celui fournit par SparkFun.



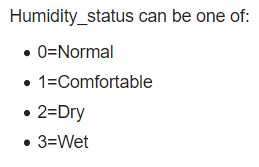
*Exemple donné par Bosch Sensortech*



*Exemple donné par SparkFun*

void BME280\_HumidityStatus(float Hum)

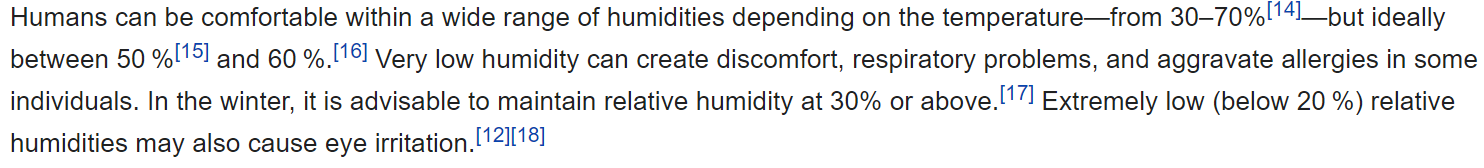
Cette fonction permet de déterminer le status de l'humidité (sec, normal, confortable et humide) en fonction du taux d'humidité en paramètre.



Les niveaux d’humidités, m’ont étés donnés directement sur le wikipédia de Domoticz.

*Source : https://www.domoticz.com/wiki/Domoticz\_API/JSON\_URL%27s*

Pour ce faire je me suis aider de ce que j’ai trouvé sur internet.



*Source :* [*https://en.wikipedia.org/wiki/Relative\_humidity*](https://en.wikipedia.org/wiki/Relative_humidity)

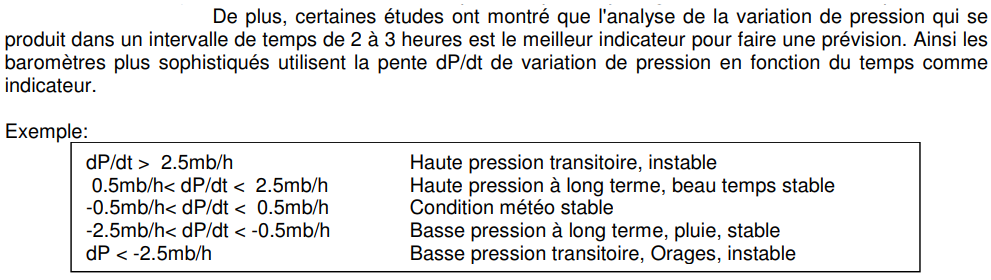
On constate que le taux l’humdité confortable est entre 30% et 70% mais idéalement entre 50% et 60%, j’ai donc décidé de mettre que lorsqu’on se trouve en dessous de 30%, on considère que c’est un niveau « sec », quand on se trouve entre 30 et 50, on considère que le niveau est « normal », quand on se trouve entre 50 et 70, on considère que le niveau est « confortable » et quand on se trouve au dessus de 70% le niveau est « humide ».

|  |  |
| --- | --- |
| Humidité | Status humidité |
| <=30% | **Sec (Dry)** |
| >30% et <=50% | **Normal (Normal)** |
| >50% et <70% | **Confortable (Confortable)** |
| >=70% | **Humide (Wet)** |

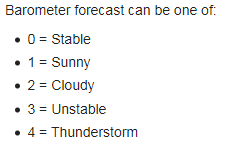
void BME280\_Forecast(float Press)

Cette fonction permet de déterminer les prévisions météorologique en prenant des mesures toutes les 20 minutes durant 3 heures.

Pour ce faire, je me suis aidé de l'AN501 de chez intersema. Dans leurs document, ils expliquent que les prévisions météorologique se font en analysant les variations de pression toutes les 20 minutes pendant 3 heures.



Lorsqu'on envoie la prévion à Domoticz, il existe que 5 prévision possible comme pour les niveau d'humidité. En envoyant chacunes des prévisons, je me suis rendu compte que les termes utilisés sur l'image de gauche sont faux.

Barometer forecast can be one of:

* 0 = No Info
* 1 = Sunny
* 2 = Cloudy
* 3 = Partly Cloudy
* 4 = Rain

C'est pour j'ai décidé de faire un tableau de la valeur à envoyer à Domoticz en fonction de la variation de pression [mba = hPa].

|  |  |
| --- | --- |
| Variation de pression | Prévison météorologique |
| >=2.5hPa | **beau temps (Sunny)** |
| >0.5hPa | **beau temps (Sunny)** |
| <=0.5hPa et >=-0.5hPa | **temps stable**  **En fonction de l'état précédent, on le garde.**  **Si il n'y avait pas encore d'état précédent, en dessus de 965hPa : beau temps (Sunny) et en dessous de 965hPa : nuageux (Cloudy).** |
| <-0.5hPa | **nuageux (Cloudy)** |
| <=-1.5hPa | **plutot nuageux (Partly Cloudy)** |
| <=2.5hPa | **pluie (rain)** |

### ESP8266.c

bool ESP8266\_Init(void)

Cette fonction permet d'initialiser le module Wifi.

uint8\_t ESP8266\_isStarted(void)

Cette fonction permet de vérifier la présence du module wifi.

uint8\_t ESP8266\_Reset(void)

Cette fonction permet de faire un reset software du module wifi.

uint8\_t ESP8266\_Mode(uint8\_t mode)

Cette fonction permet de mettre l'ESP8266 en un mode de fonctionnement. Il existe trois mode :

* Station
* Acces point
* Station + Acces point

uint8\_t ESP8266\_ConnectWifi(char \*SSID, char \*PWD)

Cette fonction permet au module de se connecter à un réseau Wifi.

uint8\_t ESP8266\_IP(char\* StrIP, char\* StrMAC)

Cette fonction permet de demander l'adresse IP et MAC du module Wifi.

uint8\_t getIpAndMac(char\* inputStr, char\* IpStr, char\* MacStr)

Cette fonction permet de récupérer l'adresse IP et MAC après les avoir demandés.

uint8\_t ESP8266\_ConnectToTCPServer(char \*IP, char \*Port)

Cette fonction permet de connecter le module Wifi à un serveur en donnant l'adresse IP et son port.

uint8\_t ESP8266\_DisconnectTCPServer(void)

Cette fonction permet de déconnecter le module Wifi au serveur auquel il est connecté.

uint8\_t SendDomoticz(uint8\_t Index, float Temp, float Hum, uint8\_t Hum\_, float Press, uint8\_t Press\_)

Cette fonction permet d'envoyer un requête HTTP au serveur Domoticz avec les données de température, humidité, humidité status, pression et prévision.

Comme lors des test avec Tera Term, il faut d'abords se connecter à domoticz grâce à la fonction **ESP8266\_ConnectToTCPServer()** puis envoyer la commande "**AT+CIPSEND=**" suivis du nombre de byte que représente la requête. Et finalement envoyer la requette HTTP, qui se compose comme telle :

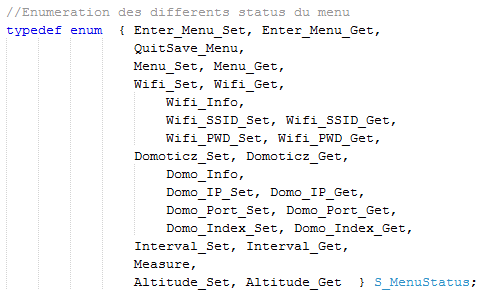


### MenuGen.c

void GestMenu (void)

Cette fonction permet de gérer l'affichage du menu.

Voici les différents états du menu:



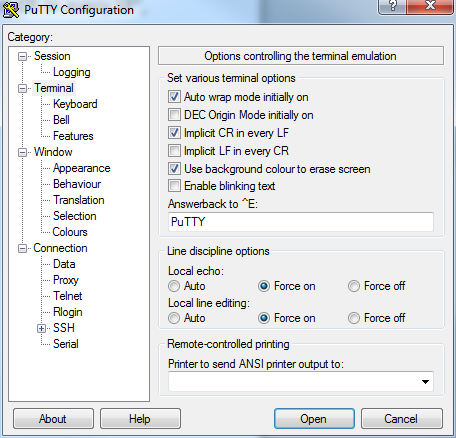
# Fonctionnement du menu

Comme demander dans le cahier des charges, j'ai créer un menu pouvant être naviguer et permettant d'afficher/modifier certains paramètre (Wifi, mesures, ...).

Tous mes tests ont été fait sur PuTTY, car c'est le programme que je connais le mieux.

## Réglages PuTTY

Tout dabords, il faut aller dans les réglages "**Terminal**" et cocher "**Implicit CR in every LF**", Local echo : "**Force on**" et Local line editing : "**Force on**".

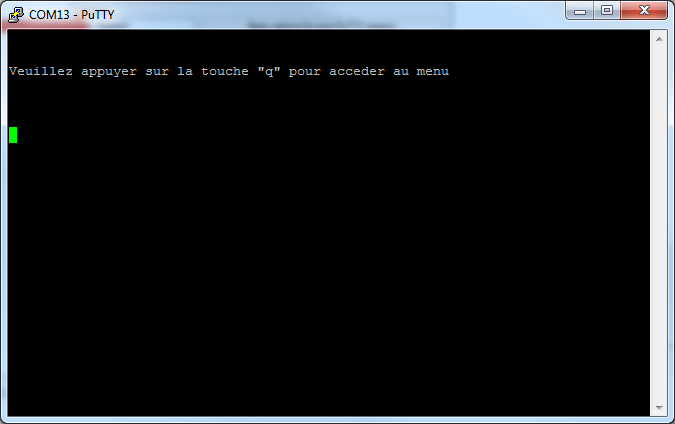


Puis aller dans les réglages de "**Session**", cocher "**Serial**" puis entrer votre "**Serial line**" (Dans mon cas c'est le COM13) et la vitesse de communication qui est de **115200** Baud. Et finalement, presser sur le bouton "**Open**".

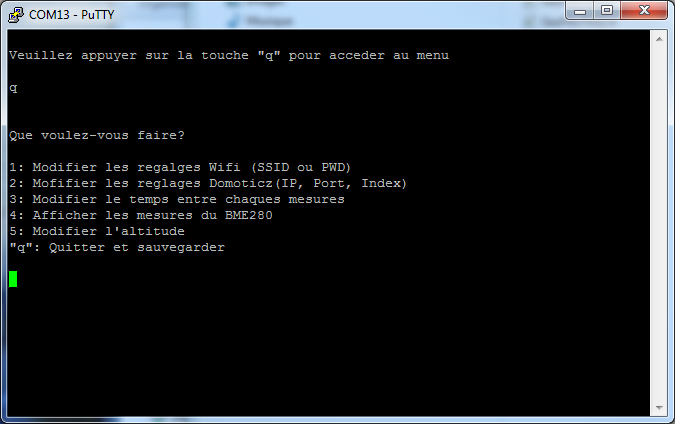


## Gestion du menu

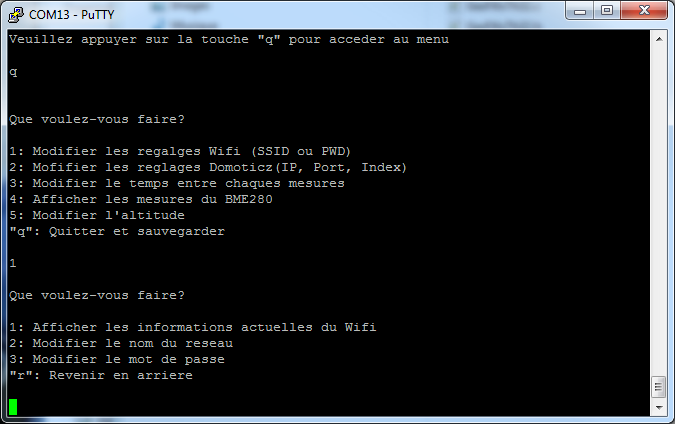
Maintenant que nous sommes connecté, le message "**Veuillez appuyer sur la touche "q" pour acceder au menu**", doit apparaitre cycliquement toutes les 5 secondes sur le terminal.



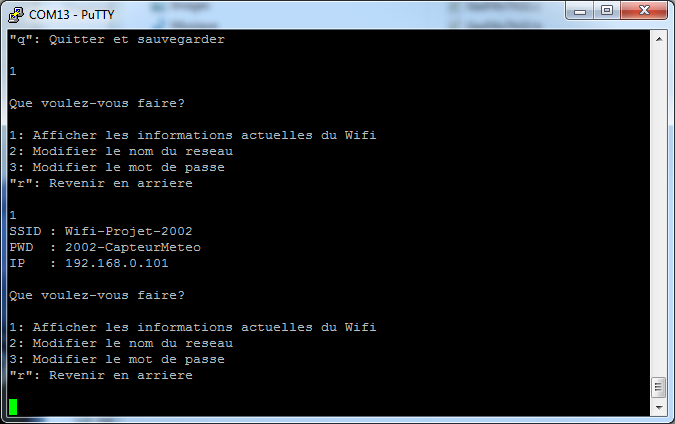
Une fois la touche "**q**" pressé, nous nous trouvons dans le menu principal (la LED clignotante verte doit être toujours bleue). Afin d'accéder à un sous-menu, il suffit de presser sur la touche "**1, 2, 3, 4 ou 5**" en fonction d'où l'on veut aller. En pressant sur la touche "**q**", nous pouvons quitter le menu et retourner dans en mode mesures.



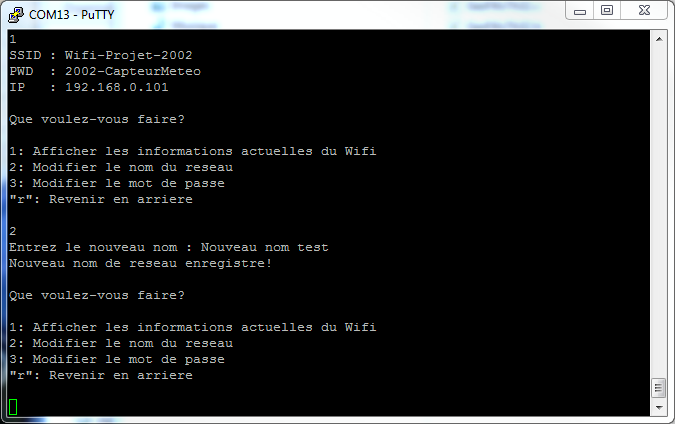
Dans cet exemple, nous sommes allez dans le sous-menu "**1**", qui permet de gérer les informations du Wifi.



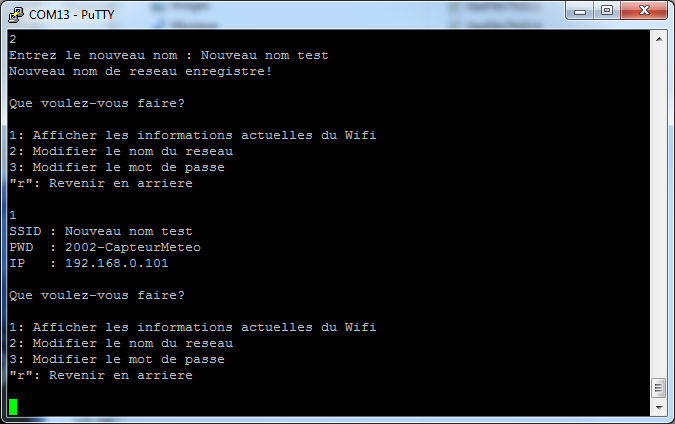
Ici nous sommes allés dans le sous-sous-menu "**Afficher les informations actuelles du Wifi**" et nous pouvons constater que le le programme nous affiche son **SSID**, son **PWD** et son **IP**. Pour revenir en arrière, il suffit de presser sur la touche "**r**".



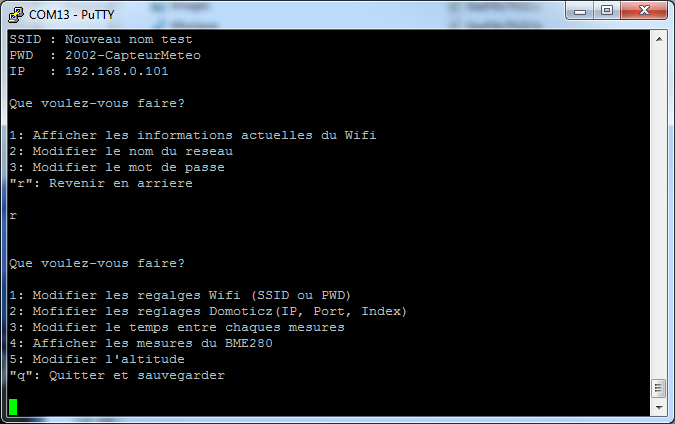
Comme exemple, j'ai décidé de vous montrer comment (par exemple) changer le nom de son Wifi. Pour celà, il suffit de presser sur la touche "**2**" et ensuite d'entrer le nouveau nom.



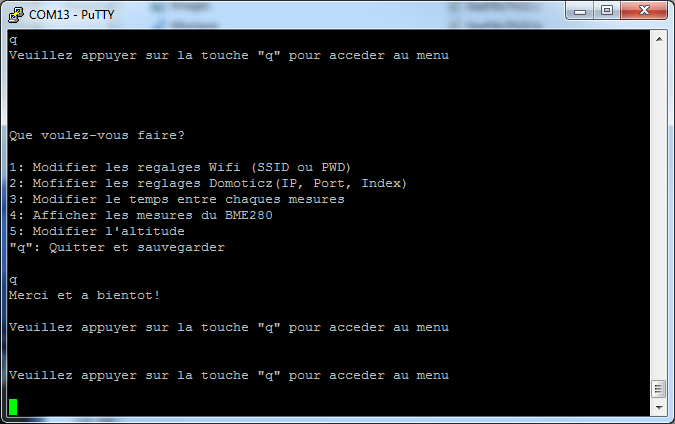
Maintenant que nous avons changé le nom de notre Wifi, nous pouvons à nouveau afficher les informations actuelles du Wifi afin de vérifier si le nom que nous venons d'entrer est correct, en appuyant sur la touche "**1**".



En appuyant sur la touche "**r**", nous sommes revenu au menu principal et pour quitter le mode menu, il suffit d'appuyer sur la touche "**q**".



Une fois le mode menu quitter, les modifications sont sauvegardées dans la mémoire flash du microcontrôleur et une initialisation de tous les modules est faite avec les nouvelles données.



# Etat d’avancement

Dans mon cahier des charges, il m'a été demandé de faire :

* Lecture des mesures du capteur
* Algorithme de prévision météo simple, basé sur l'évolution de la pression
* Menu de lecture et configuration en mode texte via port série virtuel créé par le FTDI. Le menu permettera notamment : lecture mesures, prévision, statut aisni que configuration wifi, entervalle réveil, méthode mise à jour valeurs.
* Envoi des mesures+prévison via des requêtes HTTP sur Wifi

Concernant la lecture des mesures du capteur, tout est fonctionnel.

Concernant l'algorithme de prévision météo, tout est fonctionnel.

Concernant le menu, tout est fonctionnel.

Concernant l'envoi des requêtes HTTP, tout est fonctionnel.

# Améliorations

# Conclusion

J'ai eu beaucoup de plaisir à travailler sur ce projet. Avoir un client au lieu d'un enseignant change complétement les choses. De ne pas pouvoir parler directement avec et de devoir téléphoner lors de doutes fut intéressant. Par chance, il a toujours pu répondre à mes questions. Merci à M.Roduit qui a toujours répondu sympathiquement à mes demandes.

Je souhaite aussi remercier M.Castoldi qui a pu m'aider et me rediriger lors de certains doutes, ou certains problèmes.

J'ai perdu du temps à essayer de trouver des solutions à des bugs, notamment sur mon ancienne carte, malheureusement sans succès. J'ai également passé beaucoup de temps à corriger des erreurs Hardware qui auraient facilement pu être évitée. La partie programmation du microcontrôleur, quant à elle, a aussi pris beaucoup de temps.

# Datasheets utiles

## BME280

bme280-ds002.pdf

https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/#documents

SEN-14348\_eng\_tds.pdf

https://www.sparkfun.com/products/14348

## FT230X

FT230X.pdf

https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS\_FT230X.pdf

AN\_146\_USB\_Hardware\_Design\_Guideline\_for\_FRDI\_ICs.pdf

https://www.ftdichip.com/Support/Documents/AppNotes/AN\_146\_USB\_Hardware\_Design\_Guidelines\_for\_FTDI\_ICs.pdf

## MAX1793

MAX1793.pdf

https://datasheets.maximintegrated.com/en/ds/MAX1793.pdf

## PIC32MX130F256B

PIC32MX1XX2XX 283644-PIN\_Datasheet\_DS60001168L.pdf

http://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX1XX2XX%20283644-PIN\_Datasheet\_DS60001168L.pdf#

## Prévision Météo

An501\_fr.pdf

En annexe.

## Wifi ESP click

MIKROE-2542\_WifiESPClick\_eng\_tds.pdf

https://download.mikroe.com/documents/add-on-boards/click/wifi-esp/wifi-esp-click-schematic-v100.pdf

DS\_TTL-232R\_CABLES.pdf

https://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS\_TTL-232R\_CABLES.pdf

4a-esp8266\_at\_instruction\_set\_en.pdf

https://www.espressif.com/sites/default/files/documentation/4a-esp8266\_at\_instruction\_set\_en.pdf

4b-esp8266\_at\_command\_examples\_en.pdf

https://www.espressif.com/sites/default/files/documentation/4b-esp8266\_at\_command\_examples\_en.pdf

# Annexes

## Planning

## Schéma

2002\_A\_CapteurMeteoWifi.SchDoc

2002\_B\_CapteurMeteoWifi.SchDoc

## Listing du programme

app.c / app.h

system\_interrupt.c

BME280.c / BME280.h

ESP8266.c / ESP8266.h

MenuGen.c / MenuGen.h

Mc32gest\_RS232.c / Mc32gest\_RS232.h

Mc32gest\_I2C.c / Mc32gest\_I2C.h

## Procès verbaux

## Journal de travail

1. https://www.domoticz.com/wiki/Domoticz\_API/JSON\_URL%27s [↑](#footnote-ref-1)