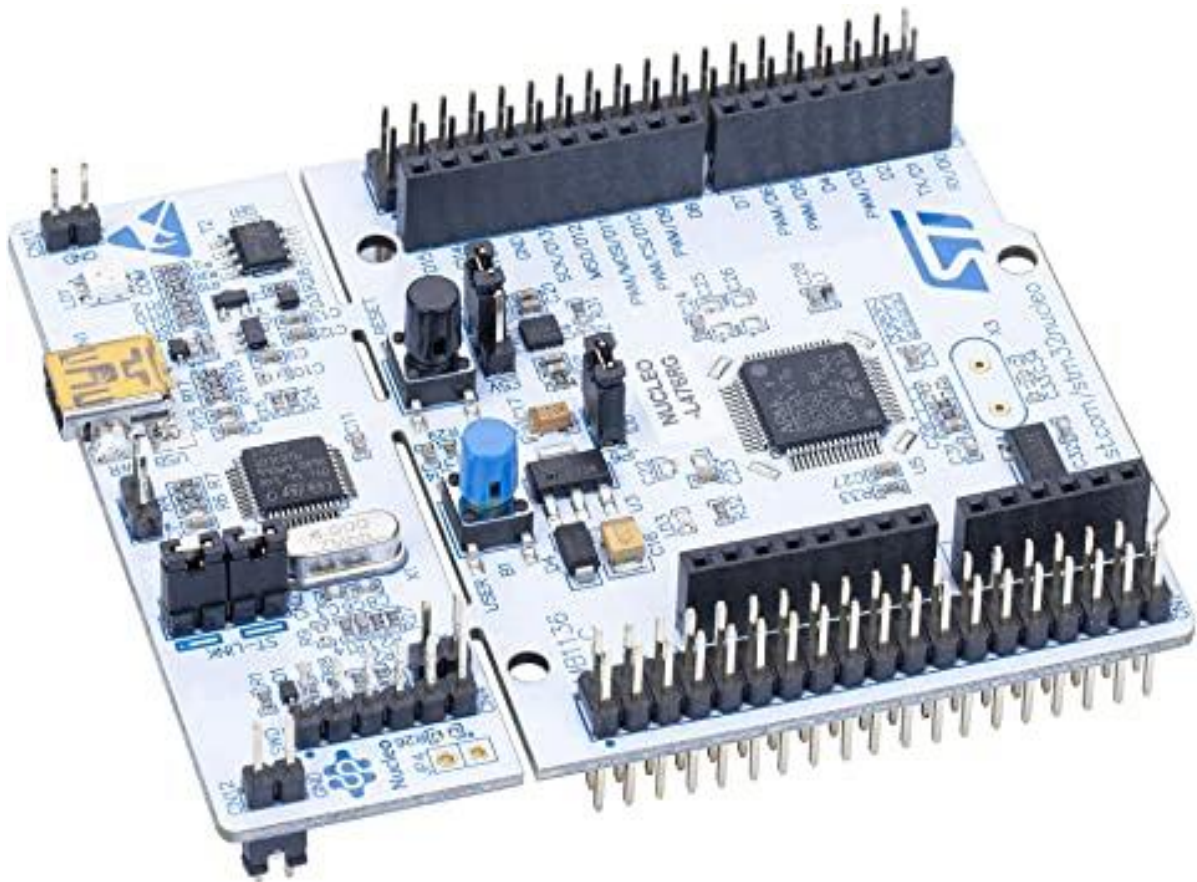


## BE : STM 32



Réalisé par :

**ACLASSATO Francesca**

**CAMARA Cheikh**

Encadré par :

**PERISSE Thierry**

## Table des matières

Introduction .....	3
I. TP DE BASE .....	3
1) Cahier des charges.....	3
1.2) SHT31.....	5
1.3) Le bus i2c [4].....	5
a) La transmission d'un octet :.....	6
b) La transmission d'une adresse : .....	6
c) Lecture d'une donnée :.....	6
2) TP de base STM32 + Capteur .....	7
3) TP de base STM32 + Capteur + Afficheur LCD I2C .....	9
4) Conclusion.....	11
II. Bureau d'Etude .....	12
1) Cahier des charges.....	12
2) Le module WIFI X-NUCLEO-IDW04A1 .....	13
3) Configuration du module WIFI.....	13
4) Envoie des données par WIFI .....	16
III. Conclusion.....	18

## Introduction

Le BE avait pour but de nous familiariser avec le microcontrôleur STM32 Nucléo, mais aussi les différents protocoles de communication de ce dernier (I2C, UART, SPI...).

Il était divisé en deux grandes parties. La première était sous forme de TP de base où on devait travailler avec des capteurs de température et d'humidité, la seconde c'était le BE. Pour ce dernier, on devait définir un cahier des charges avec l'encadrant et le réaliser étape par étape.

D'abord nous allons parler du TP de base, expliquer les protocoles, leurs fonctionnements et commenter les résultats obtenus et enfin nous allons faire la même chose pour la partie du BE.

### I. TP DE BASE

Je voulais vous prévenir que sur le rapport il y'a que la partie SHT31 car ma binôme (Francesca ACLASSATO) n'a pas pu faire sa partie du rapport.

#### 1) Cahier des charges

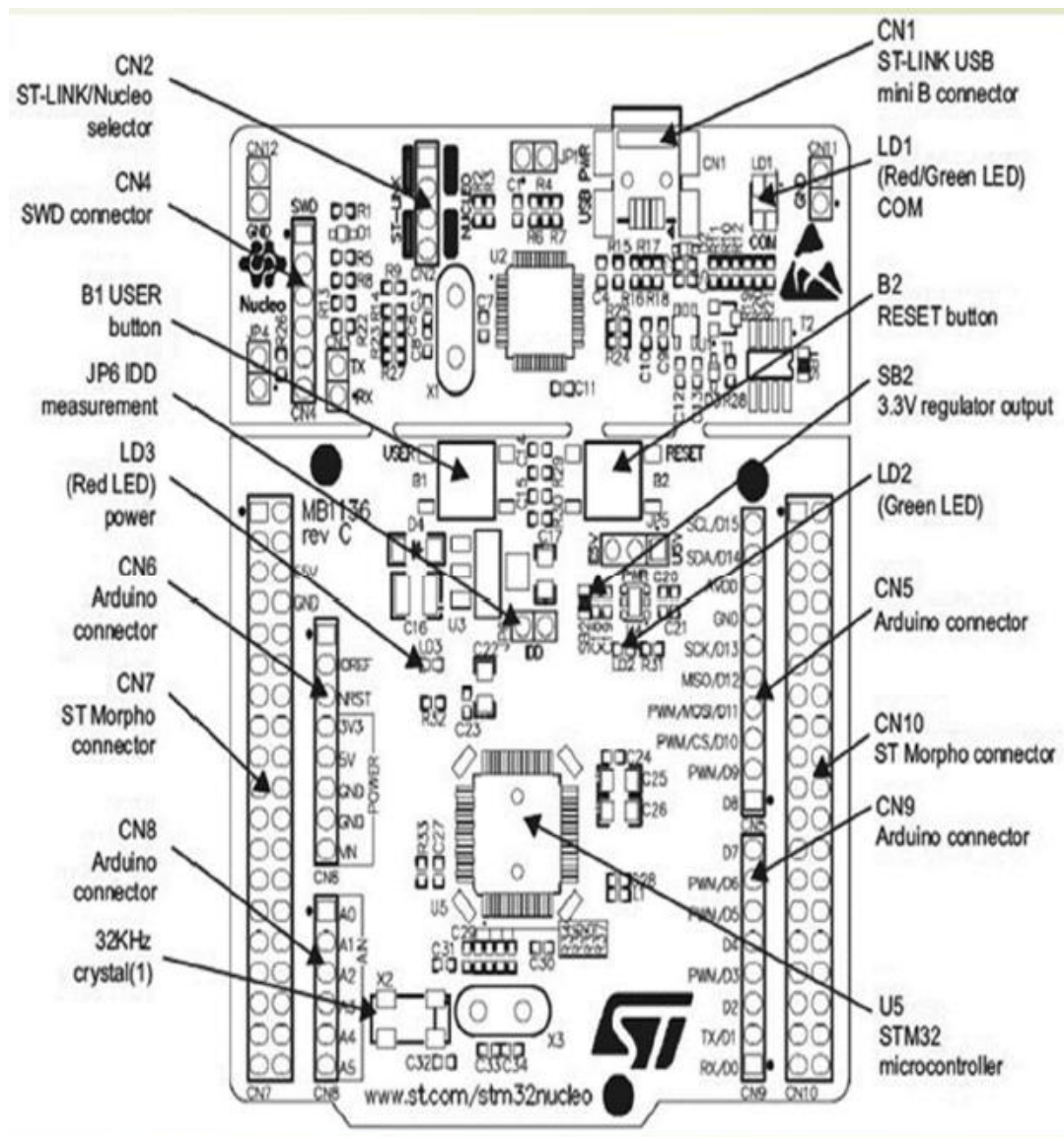
Le TP se déroule en deux parties. La première consiste à recueillir la température et l'humidité d'une pièce grâce au capteur SHT31, l'autre c'est d'afficher les résultats sur un écran LCD.

##### 1.1) STM32 Nucleo-64

La carte STM32 Nucleo-64 offre aux utilisateurs un moyen abordable et flexible d'essayer de nouveaux concepts et de construire des prototypes en choisissant parmi les diverses combinaisons de fonctionnalités de performance et de consommation d'énergie, fournies par le microcontrôleur STM32. La prise en charge de la connectivité ARDUINO® Uno V3 et les en-têtes morpho ST permettent d'étendre facilement les fonctionnalités de la plate-forme de développement ouverte STM32 Nucleo avec un large choix de boucliers spécialisés.

La carte STM32 Nucleo-64 ne nécessite aucune sonde séparée car elle intègre le débogueur / programmeur ST-LINK.

La carte STM32 Nucleo-64 est livrée avec les bibliothèques complètes de logiciels gratuits STM32 et des exemples disponibles avec le package MCU STM32Cube. [1]



La carte nucleo comprend :

2 connecteurs :

- Arduino compatible headers
- Morpho headers

Un programmeur-linker (générateurs de liens) de type ST-LINK

3 LEDs :

- LD1 pour indiquer que le port USB est opérationnel ;
- LD2 qui peut être utilisée dans l'application ;
- LD3 qui indique la présence de l'alimentation.

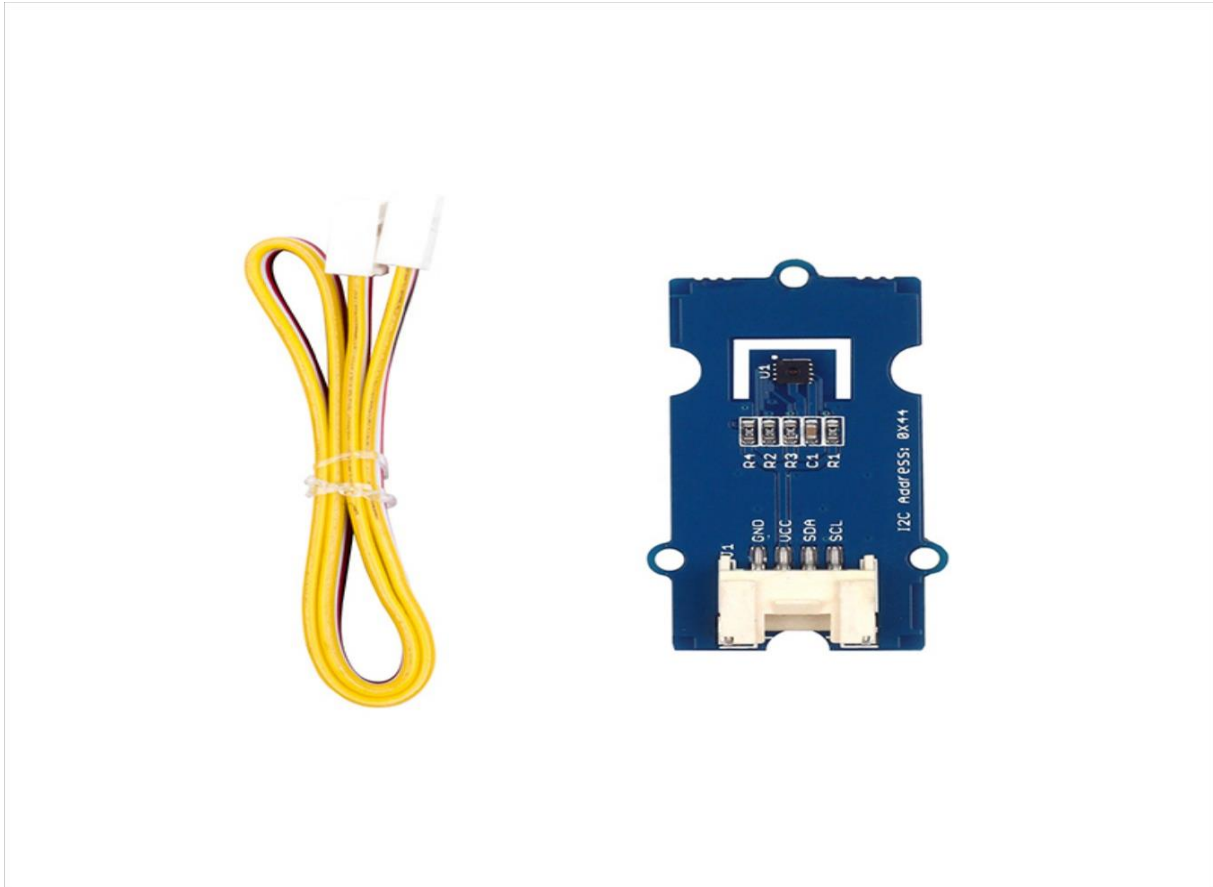
2 BPs :

- B1 pour l'application si nécessaire ;
- B2 pour initialiser la carte.

Les cartes Nucleo64 contiennent un oscillateur externe à 32 768 kHz utilisé par le STM32. [2]

## 1.2) SHT31

Les capteurs de température / humidité Sensirion sont parmi les appareils les plus fins et les plus précis sur le marché. Ils ont des interfaces I2C pour une lecture facile. Le capteur SHT31 a une excellente humidité relative de  $\pm 2\%$  et une précision de  $\pm 0,3\text{ }^{\circ}\text{C}$  pour la plupart des utilisations. [3]



Le capteur SHT31 a 4 ports : 2 pour l'alimentation (VCC et GND) et les 2 autres sont les ports i2c (SDA et SCL)

## 1.3) Le bus i2c [4]

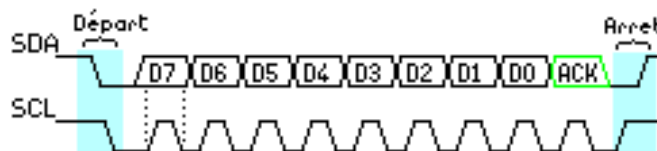
Le protocole I2C définit la succession des états logiques possibles sur SDA et SCL, et la façon dont doivent réagir les circuits en cas de conflit.

Pour prendre le contrôle du bus, il faut que celui-ci soit au repos (SDA et SCL à '1'). Pour transmettre des données sur le bus, il faut donc surveiller deux conditions particulières :

- La condition de départ. (SDA passe à '0' alors que SCL reste à '1')
- La condition d'arrêt. (SDA passe à '1' alors que SCL reste à '1')

Lorsqu'un circuit, après avoir vérifié que le bus est libre, prend le contrôle de celui-ci, il en devient le maître.

C'est toujours le maître qui génère le signal d'horloge.



Exemple de condition de départ et d'arrêt.

#### a) La transmission d'un octet :

Après avoir imposé la condition de départ, le maître applique sur SDA le bit de poids fort D7. Il valide ensuite la donnée en appliquant pendant un instant un niveau '1' sur la ligne SCL. Lorsque SCL revient à '0', il recommence l'opération jusqu'à ce que l'octet complet soit transmis. Il envoie alors un bit ACK à '1' tout en scrutant l'état réel de SDA. L'esclave doit alors imposer un niveau '0' pour signaler au maître que la transmission s'est effectuée correctement. Les sorties de chacun étant à collecteurs ouverts, le maître voit le '0' et peut alors passer à la suite.

#### b) La transmission d'une adresse :

Le nombre de composants qu'il est possible de connecter sur un bus I2C étant largement supérieur à deux, il est nécessaire de définir pour chacun une adresse unique. L'adresse d'un circuit, codée sur sept bits, est défini d'une part par son type et d'autre part par l'état appliqué à un certain nombre de ces broches. Cette adresse est transmise sous la forme d'un octet au format particulier.

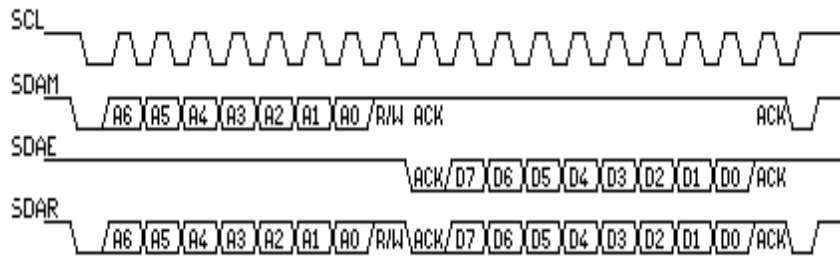


On remarque ici que les bits D7 à D1 représentent les adresse A6 à A0, et que le bit D0 et remplacé par le bit de R/W qui permet au maître de signaler s'il veut lire ou écrire une donnée. Le bit d'acquittement ACK fonctionne comme pour une donnée, ceci permet au maître de vérifier si l'esclave est disponible.

#### c) Lecture d'une donnée :

La lecture d'une donnée par le maître se caractérise par l'utilisation spéciale qui faite du bit ACK. Après la lecture d'un octet, le maître positionne ACK à '0' s'il veut lire la donnée suivante (cas d'une mémoire par exemple) ou à '1'le cas échéant. Il envoie alors la condition d'arrêt.



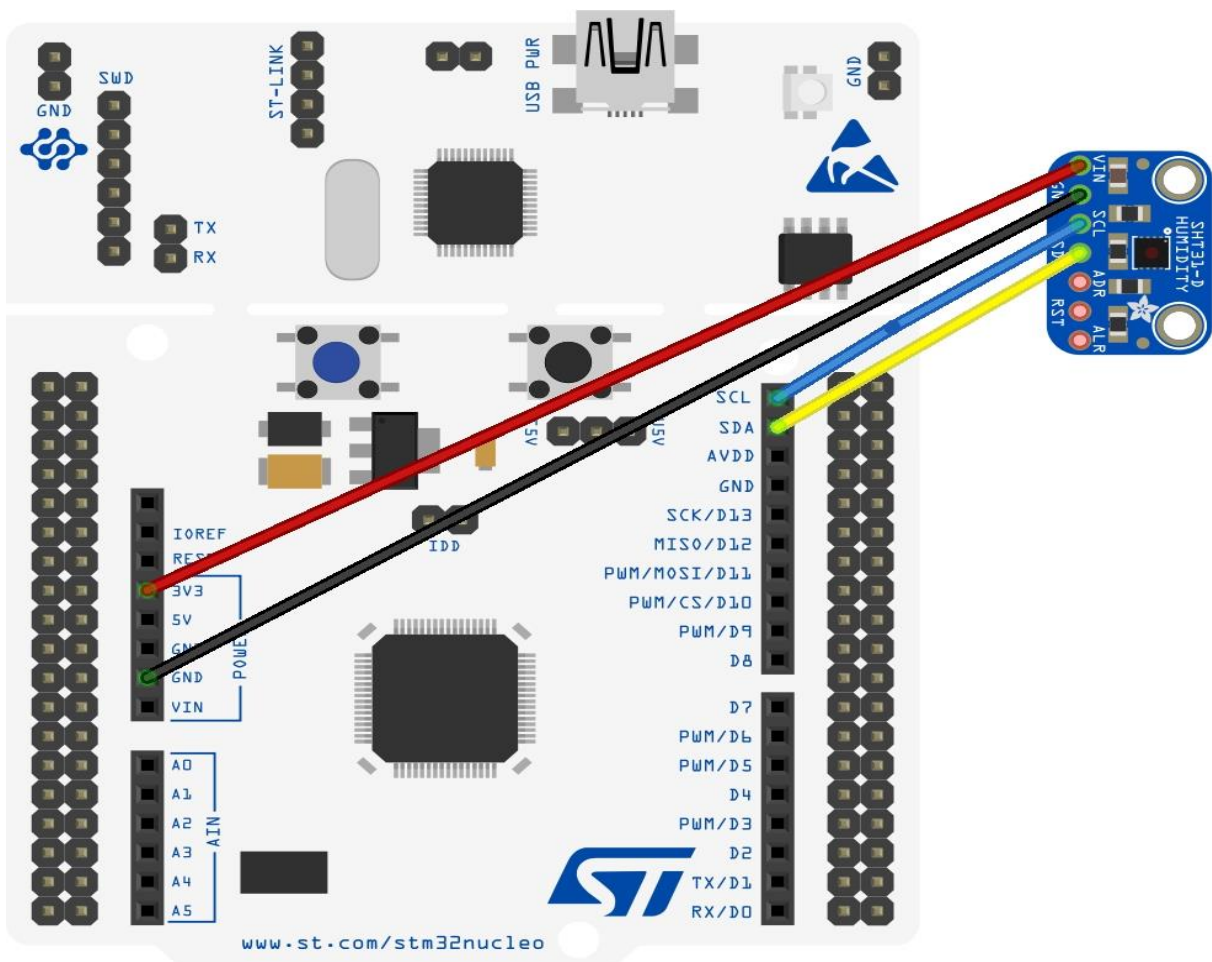


Exemple de lecture d'une donnée.

- SCL : Horloge imposée par le maître.
- SDAM : Niveaux de SDA imposés par le maître.
- SDAE : Niveaux de SDA imposés par l'esclave.
- SDAR : Niveaux de SDA réels résultants.

## 2) TP de base STM32 + Capteur

Tout d'abord, faisons un schéma de câblage du capteur SHT31 avec la carte nucleo-64.



fritzing

Le port SDA du capteur doit être relié au port i2c de la carte nucleo. De même que les ports SCL et l'alimentation.

Après avoir fini le câblage, on commence la programmation en utilisant un IDE. Pour notre part, on a utilisé CubeIDE. (Le programme se trouve sur GitHub)

En observant la communication entre le STM32 (maître) et le capteur SHT31(esclave) avec un oscilloscope, on peut lire les données suivantes.

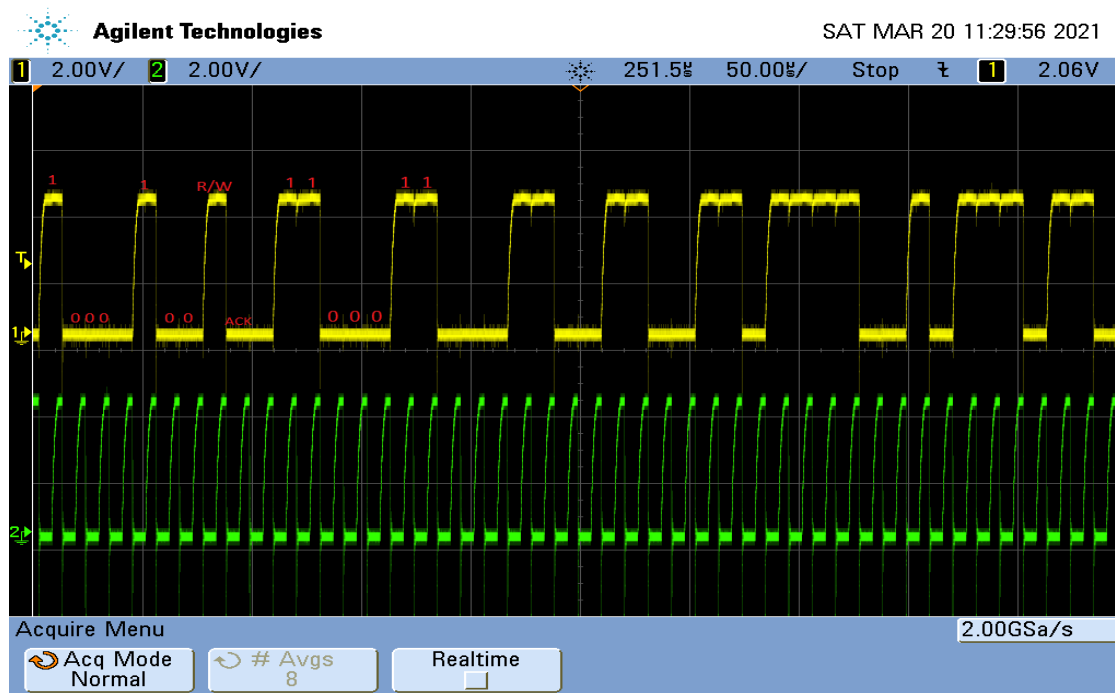


Figure1 : Trame i2c entre le STM32 et le SHT31

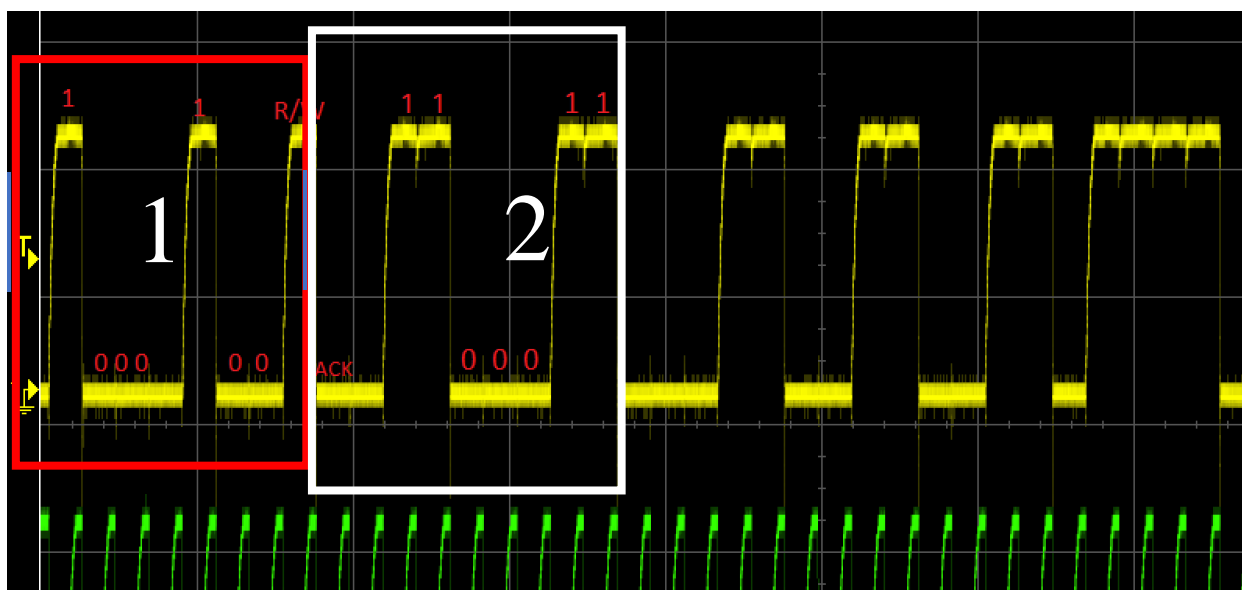


Figure 2 : Trame i2c entre le STM32 et le SHT31



En observant la trame i2c échangée entre le maître (STM32) et l'esclave (SHT31), on voit que le premier octet, qui est envoyé par le maître (STM32), correspond à l'adresse de l'esclave et au bit R/W.

Dans la datasheet du SHT31 on peut voir l'adresse de l'esclave (le capteur) qui est : **0x44**.

En convertissant l'adresse en binaire on obtient :

**0x44 = 1000100.**

Donc sur la trame i2c obtenue avec l'oscilloscope, on voit bien que le premier octet (1000100) correspond bien à l'adresse du capteur. Cette adresse permet au maître de dire avec quel composant il veut communiquer.

Le deuxième octet correspondant à l'information envoyée par le capteur au STM32. Le premier bit correspond au ACK et les 7 autres, à l'information recueillie par le capteur.

On reçoit **1100011** qui est égal à **99** en décimal. Dans la datasheet, on a la formule pour calculer la valeur de la température et de l'humidité.

**Température = (- 45 + 175 \* (valeur/65535)) \* 100 ;**

En faisant le calcul et en remplaçant « valeur » par **99**, on trouve une température égale à **26.23°C**. C'est la même température qu'on avait observée avec Tera Term.

Pour avoir l'humidité, on fait la même chose avec la formule de l'humidité qui se trouve dans la datasheet.

Grace à la trame observée avec l'oscilloscope, on peut en déduire que la communication entre le maître et l'esclave fonctionne très bien, et que le maître reçoit les données envoyées par l'esclave (la température et l'humidité).

### **3) TP de base STM32 + Capteur + Afficheur LCD I2C**

Tout d'abord, faisons le schéma de câblage du projet :

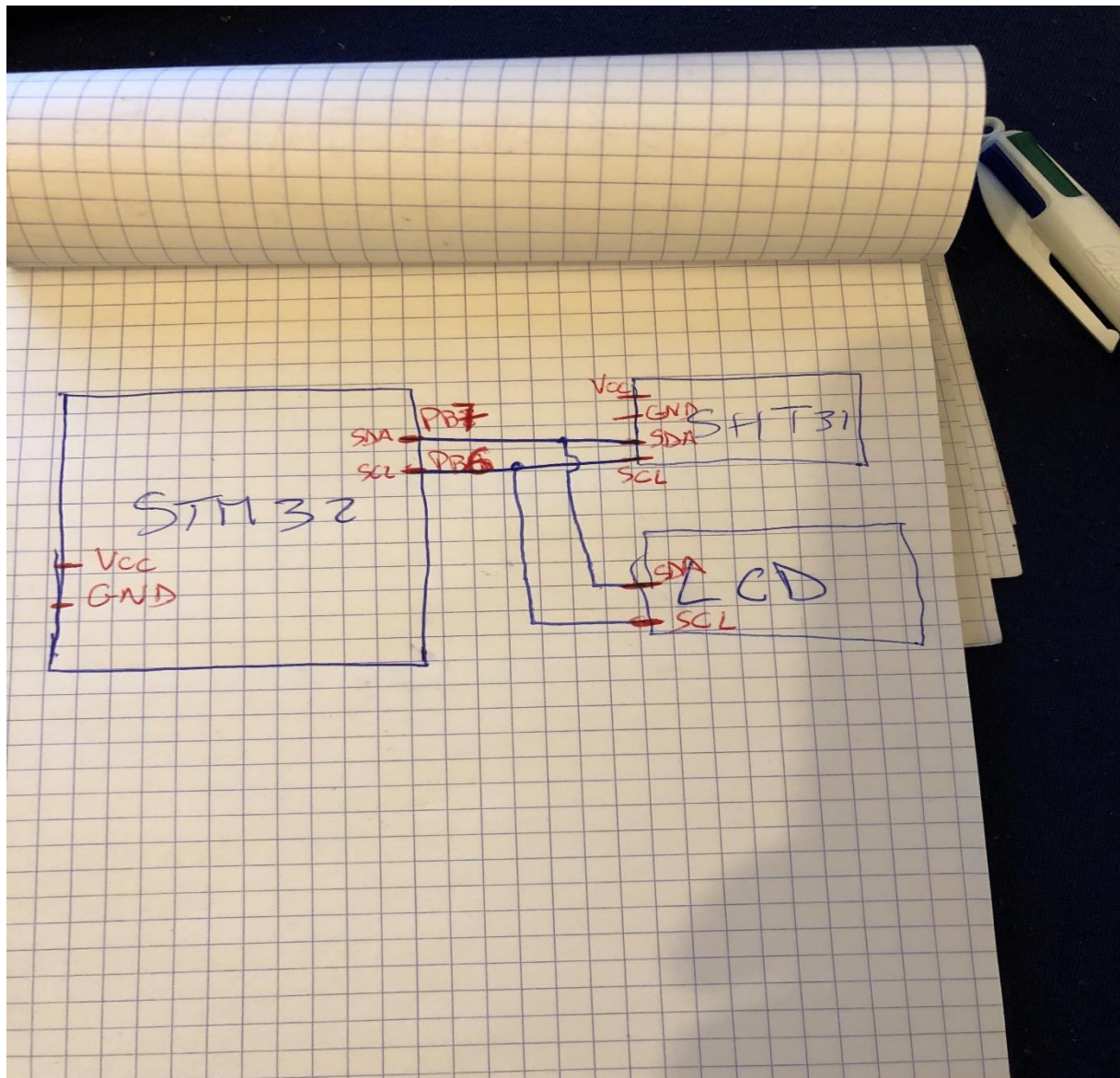


Figure 3 : Schéma de câblage STM32+LCD+SHT31

Pour pouvoir afficher la température et l'humidité sur un écran LCD grâce au bus I2C, on a utilisé les bibliothèques et les fonctions que nous a donnée Mr Perisse.

Après avoir fini la programmation, on a réussi à afficher la température et l'humidité sur l'écran LCD et on obtient le résultat suivant :

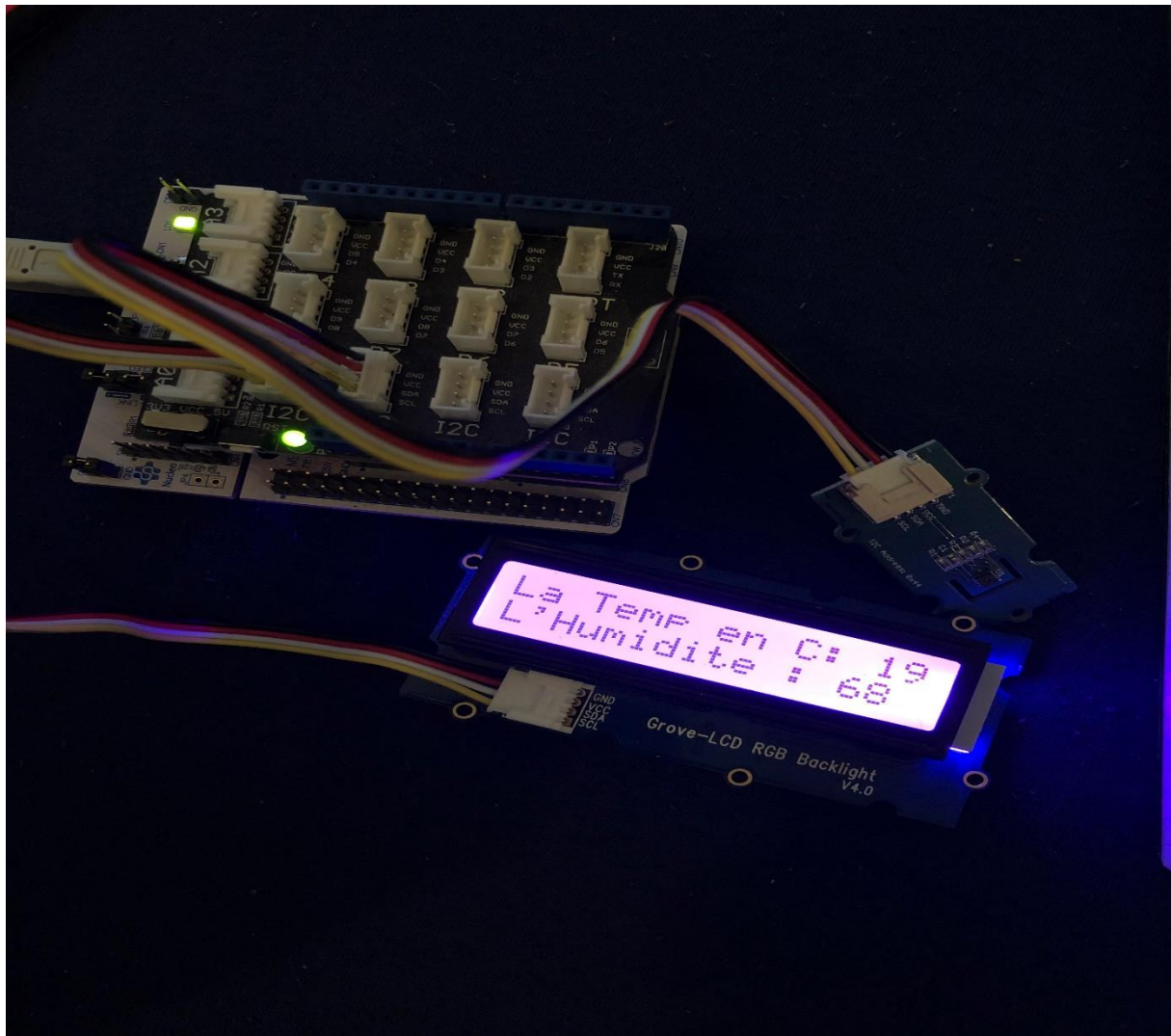


Figure 4 : Affichage température et humidité sur le LCD

On a réussi à avoir récupérer et afficher les résultats du capteur sur le LCD.

#### 4) Conclusion

Ce TP de base a été un bon début pour manipuler et comprendre les microcontrôleurs STM32. En effet, on a eu à utiliser les bus I2C, GPIO, USART... et cela nous a mieux comprendre le fonctionnement et comment programmer ces bus.



## II. Bureau d'Etude

### 1) Cahier des charges

Pour le BE, on a défini le cahier des charges suivant :

Après avoir récupérer les données (températures et humidités) des deux capteurs (DHT22 et SHT31), on envoie ces données via un module WIFI et on les récupère avec un PC.

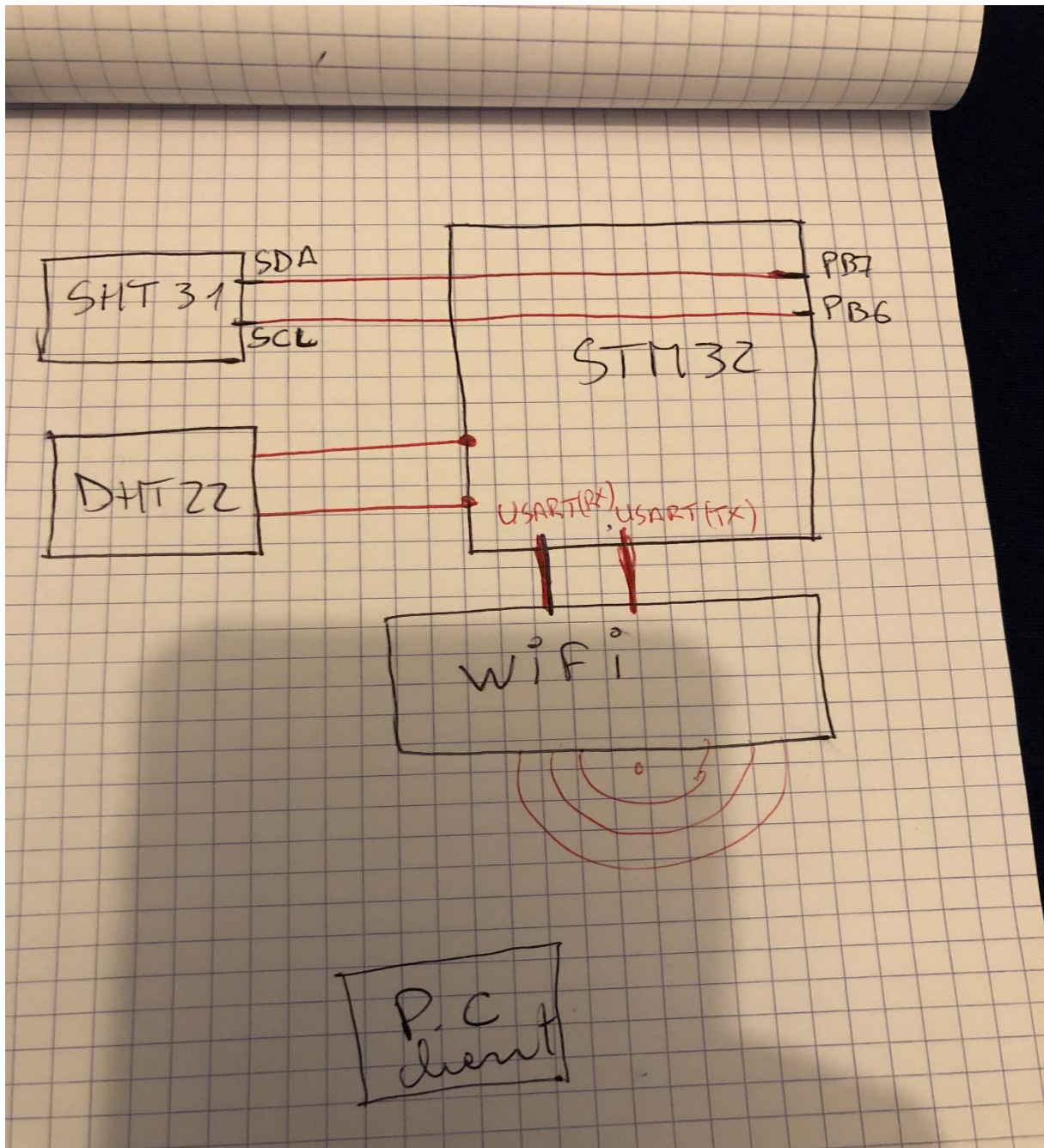


Figure 5 : Schéma de câblage du BE

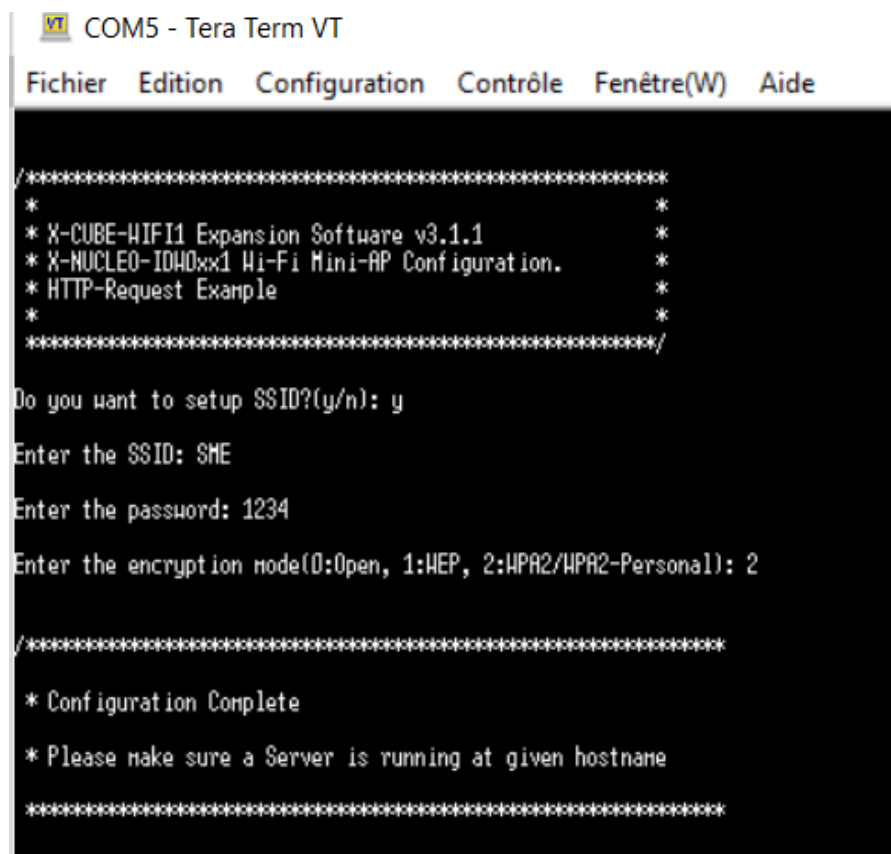
## 2) Le module WIFI X-NUCLEO-IDW04A1

La carte d'évaluation Wi-Fi X-NUCLEO-IDW04A1 est basée sur le module SPWF04SA et élargit la gamme STM32 Nucleo. Le module SPWF04SA dispose d'un microcontrôleur STM32 intégré, d'un SoC Wi-Fi b / g / n basse consommation avec amplificateur de puissance et gestion de l'alimentation intégrés, et d'une antenne SMD. Le module SPWF04SA repose sur 2 Mo de mémoire Flash interne MCU. 1 Mo de mémoire flash interne sert à stocker le système de fichiers utilisateur et à effectuer une mise à jour sécurisée du micrologiciel par liaison radio (FOTA). Une interface matérielle permet l'utilisation de la mémoire externe pour étendre la capacité de stockage du système de fichiers sans limite de taille.

## 3) Configuration du module WIFI

Le module WIFI utilise le protocole UART, donc on doit le relier avec la carte STM32 à l'aide du port USART1 car le port USART2 est uniquement pour debugger la carte.

On a configuré le module avec les commandes AT :



```
COM5 - Tera Term VT
Fichier  Edition  Configuration  Contrôle  Fenêtre(W)  Aide

/*****
*
* X-CUBE-WIFI1 Expansion Software v3.1.1
* X-NUCLEO-IDW04A1 Hi-Fi Mini-AP Configuration.
* HTTP-Request Example
*
*****/

Do you want to setup SSID?(y/n): y

Enter the SSID: SME

Enter the password: 1234

Enter the encryption mode(0:Open, 1:WEP, 2:WPA2/WPA2-Personal): 2

/*****

* Configuration Complete

* Please make sure a Server is running at given hostname

*****/
```

La configuration a réussi avec succès.

Pour que le réseau WIFI du module soit visible, on doit l'initialiser avec les commandes AT. On obtient le résultat suivant :

```
Initializing the wifi module...

Build Configuration:
Nucleo: NUCLEO-L476RG
X-Nucleo: X-NUCLEO-ID404A1
Interface: UART

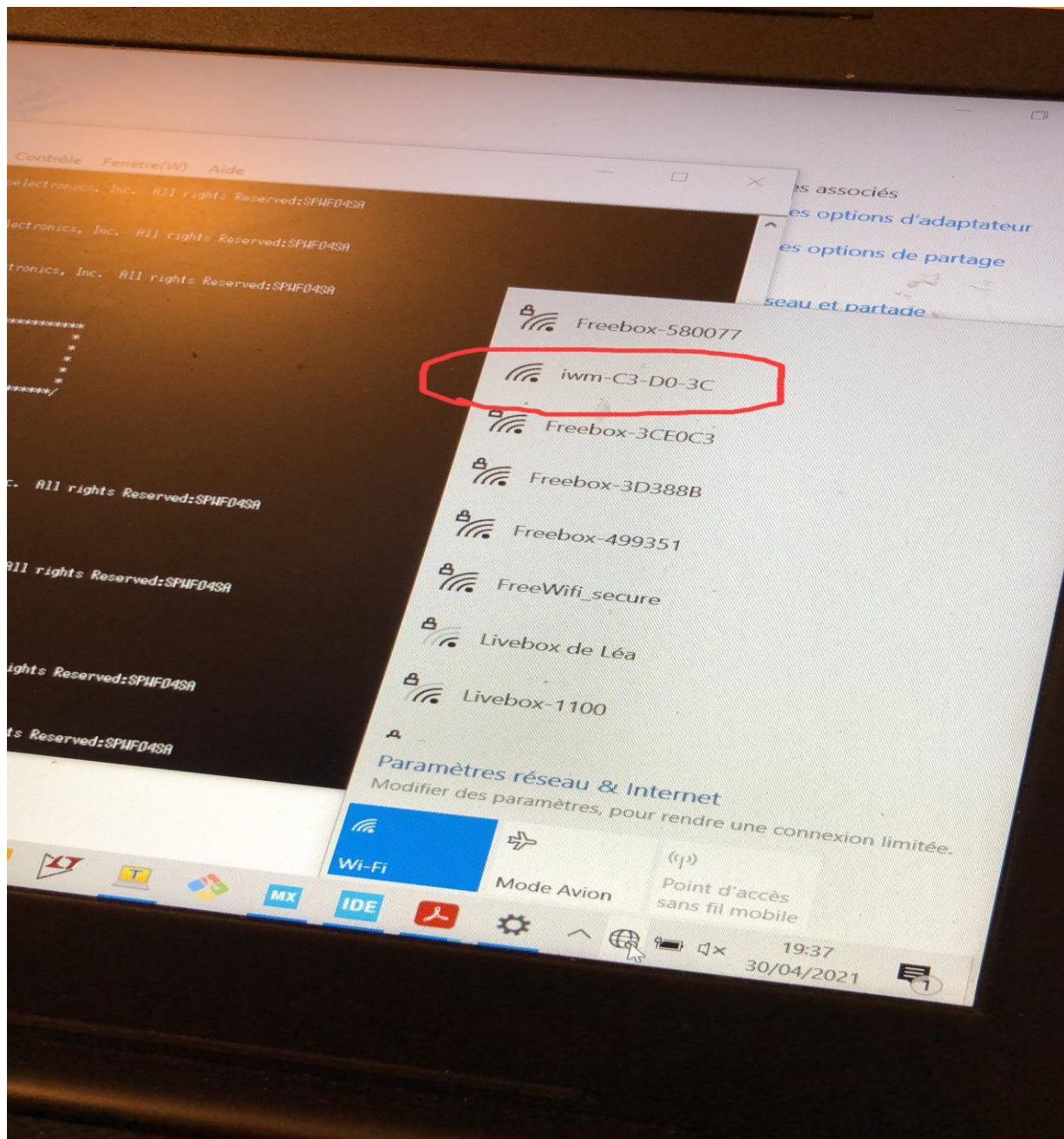
/*****
 *
 * X-CUBE-WIFI1 Expansion Software v3.1.1
 * Console Application
 * Send AT commands to SPWF module directly
 *
 *****/

Please wait...
.MCFG
AT-S.O.K
AT+S.RESET
+HIND:2:Reset
+HIND:1:Poweron:170216-fd39c59-SPWF04S
+HIND:13:Copyright (c) 2012-2017 STMicroelectronics, Inc. All rights Reserved:SPWF04SA
+HIND:0:Console active
+HIND:3:Watchdog Running:20
Console Ready...
at
AT-S.O.K
```

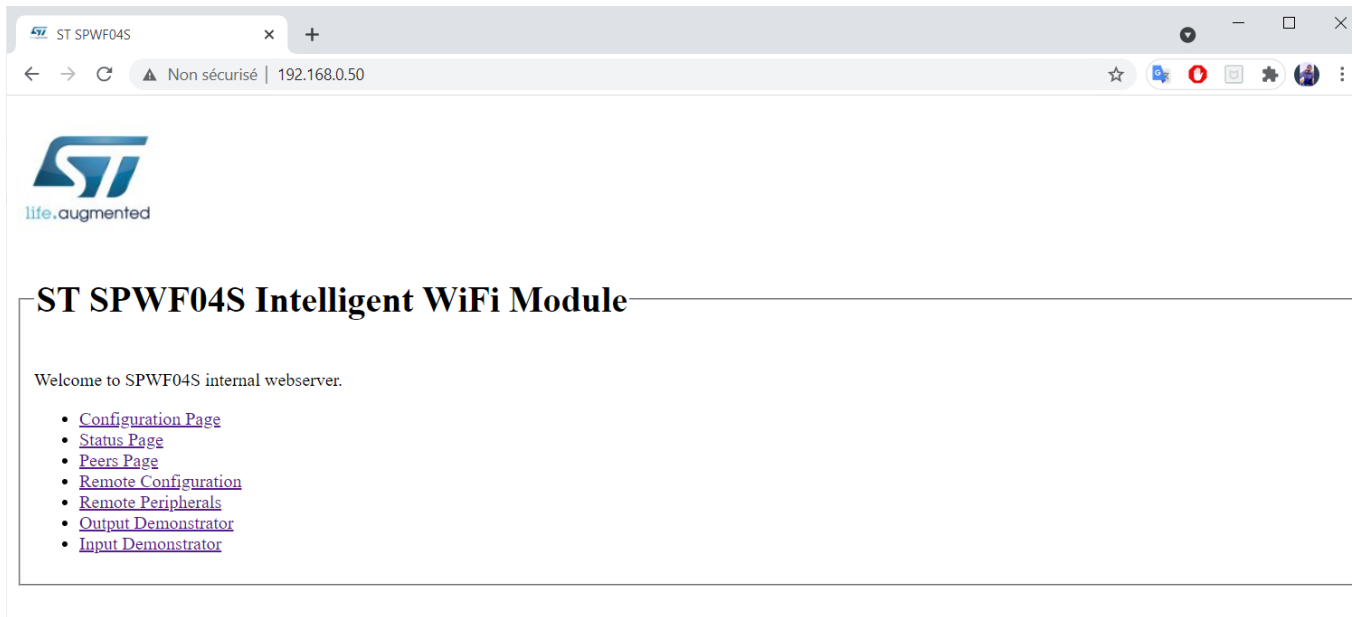
En tapant la commande AT, le module nous répond OK ce qui veut dire que tout s'est bien passé.

Maintenant on peut voir notre réseau WIFI sur la liste des réseaux disponible :





Une fois connecter au réseau WIFI, si on tape l'adresse IP du module (192.168.0.50) sur un navigateur web, on peut avoir accès à l'interface du module comme suit :



Avec l'interface graphique, on peut configurer le module wifi.

#### 4) Envoie des données par WIFI

Malheureusement, on n'a pas réussi à envoyer la température et l'humidité par WIFI.

Pour savoir si le problème vient du module ou de la carte STM32, on a décidé de faire un test en envoyant à partir du STM32 et avec le protocole UART, une lettre (la lettre a) pour voir si on réussi à le recevoir à la sortie du STM32. Pour ce faire, on a visualisé les ports USART avec un oscilloscope pour voir si on reçoit correctement la lettre envoyée.

La lettre « a » correspond à 97 en ASCII. Sur l'oscilloscope, on peut voir la trame suivante :

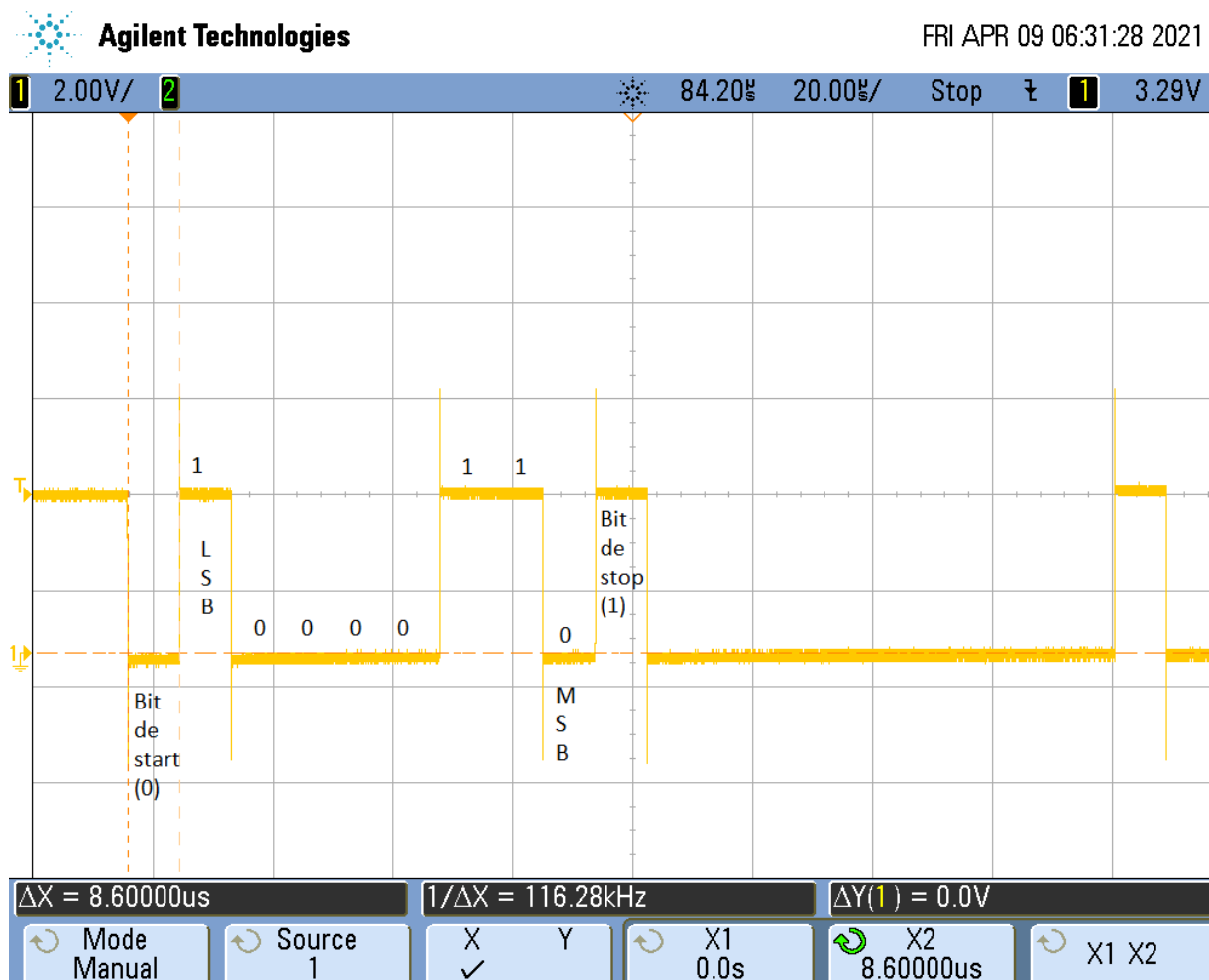


Figure 5 : Trame UART de la lettre « a »

Ici entre le LSB et le MSB on obtient : **1 0 0 0 0 1 1 0**

**1 0 0 0 0 1 1 0 = 0x61 (en hexa) ;**

**0x61 = 97 en décimal ce qui correspond « a » en ASCII.**

Donc là on voit bien que la lettre « a » envoyé par le STM32 à travers le port USART a bien été reçu par l'oscilloscope.

On peut conclure que notre STM32 envoie bien les données par UART, ce qui veut dire que notre problème vient du module WIFI.

Malheureusement on n'a pas réussi à résoudre le problème.

### III. Conclusion

Ce BE fut très enrichissant car cela nous a permis de travailler avec des microcontrôleurs, des capteurs, des écrans lcd, des oscilloscopes entre autres. Ça nous a aussi permis de faire de travailler en autonomie et en équipe. Nous avons travaillé avec des protocoles comme I2C, UART, SPI... qui sont largement utilisé maintenant.

Malheureusement nous n'avons pas pu réaliser le cahier des charges du BE car on a eu des problèmes pour envoyer les données par le module wifi.

[1] <https://www.st.com/en/evaluation-tools/nucleo-l476rg.html>

[2] [ftp://thierryperisse@ftpperso.free.fr/STM32/BE\\_STM32\\_vers2021.pdf](ftp://thierryperisse@ftpperso.free.fr/STM32/BE_STM32_vers2021.pdf)

[3] <https://www.adafruit.com/product/2857>

[4]

<http://electro8051.free.fr/I2C/busi2c.htm#:~:text=Le%20protocole%20I2C%20d%C3%A9finit%20la,circuits%20en%20cas%20de%20conflit.&text=Lorsqu'un%20circuit%2C%20apr%C3%A9s%20avoir,il%20en%20devient%20le%20ma%C3%AAtre.>