Rapport BE

X-NUCLEO-IKO1A2

QIN Xiaotong

MAZARGUIL Marlon

2021/2022

M1 SME

Université Paul Sabatier

Table des matières

Introduction

Dans ce TP nous avons 2 partis, les TP de base et le BE. Dans les TP de base nous avons utilisé un capteur différent chacun. Deux capteurs de température.

Le but du TP de base est de se familiariser avec l'environnement STM32CubeIDE (nous avons utilisé version 1.8.0) et de comprendre comment programmer des capteurs grâce à la carte NUCLEO-L476RG. Pour ce faire, nous devons afficher la température sur un écran LCD pour le capteur de température.

Une fois que nous avons compris comment fonctionne le logiciel et la carte nous avons fait le BE en utilisant le même logiciel et la même carte.

Dans le TP BE chaque groupe à un projet dans lequel il doit utiliser un ou les capteurs de son choix. Notre projet est de visualiser l'orientation et le mouvement d'un objet en 3D dans un logiciel. Pour ce faire nous utiliserons le capteur X-NUCLEO-IKO1A2.

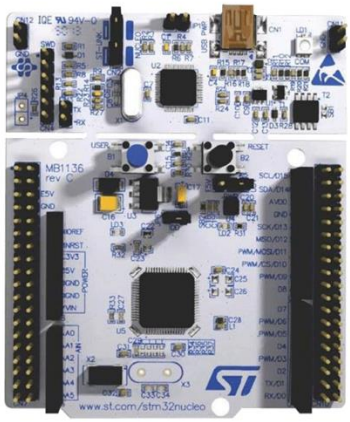


Figure 1 : NUCLEO-L476RG

Figure 2 : STM32CubeIDE 1.8.0

TP de Base

Capteur de Température I2C (PmodTMP2)

Dans ce projet de base nous utiliserons le capteur PmodTMP2. Le but est d'afficher la température sur un écran LCD. Pour cela nous utiliserons le logiciel STM32CubeIDE 1.8.0 et la carte NUCLEO-L476RG.

Figure 4 : PmodTMP2

Le capteur (PmodTMP2) et l'écran LCD ce commande en utilisant l'I2C. Or on n'a qu'une sortie I2C sur le NUCLEO-L476RG.

Nous avons donc besoin d'une carte connectique : Base Shield v2.1 qui permet d'avoir plus de branchement.

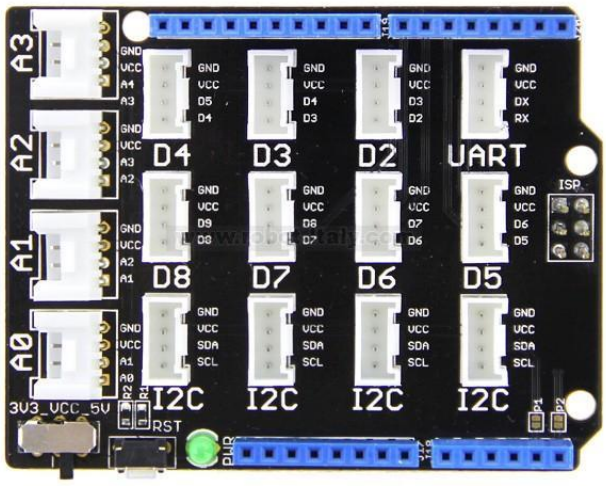


Figure 5 : Base Shield v2.1

On en déduit le schéma fonctionnel suivant :

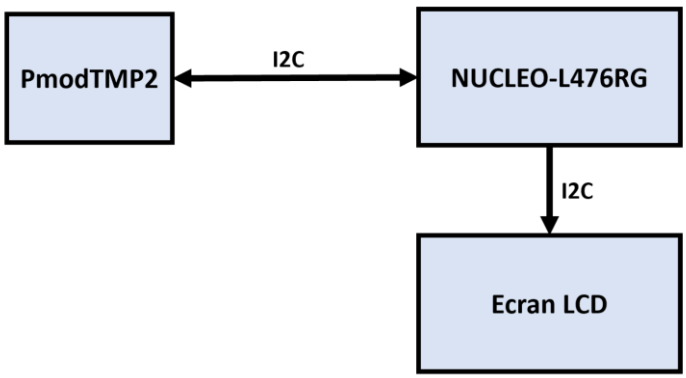


Figure 3 : Schéma fonctionnel

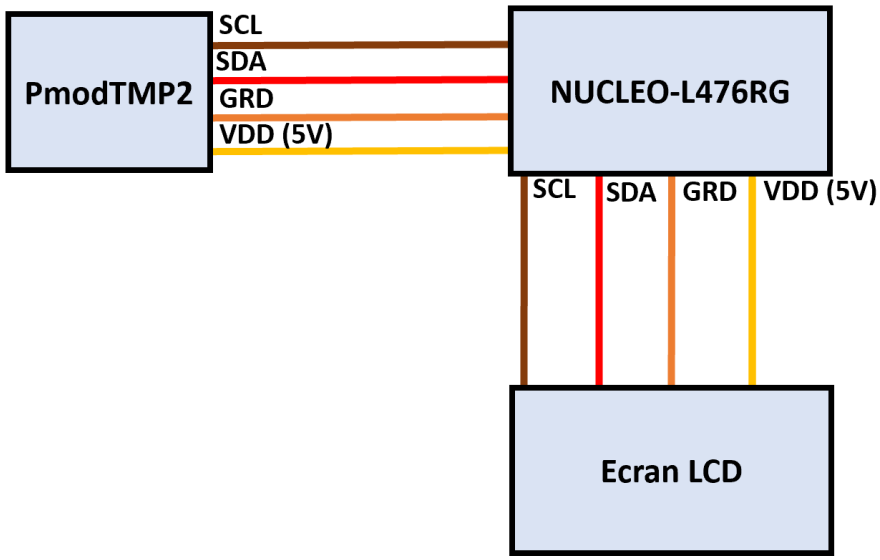


Figure 6 : Câblage

Datasheet

En lisant la datasheet nous avons vu que l'adresse est : 0x4B

Table 20. I²C Bus Address Options

Binary							
A6	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**Hex**
1	0	0	1	0	0	0	0x48
1	0	0	1	0	0	1	0x49
1	0	0	1	0	1	0	0x4A
1	0	0	1	0	1	1	0x4B

On peut voir le format de la data. la data correspond aux données pour connaître la température

Table 5. 13-Bit Temperature Data Format

Temperature	Digital Output (Binary) Bits[15:3]	Digital Output (Hex)
−40°C	1 1101 1000 0000	0x1D80
−25°C	1 1110 0111 0000	0x1E70
−0.0625°C	1 1111 1111 1111	0x1FFF
0°C	0 0000 0000 0000	0x000
+0.0625°C	0 0000 0000 0001	0x001
+25°C	0 0001 1001 0000	0x190
+105°C	0 0110 1001 0000	0x690
+125°C	0 0111 1101 0000	0x7D0
+150°C	0 1001 0110 0000	0x960

Nous pouvons voir que le capteur peut lire des températures allant de -40°C à +150°C.

Pour connaître la conversion pour passer de la data reçu au °C, nous pouvons regarder l'image ci-dessous :

TEMPERATURE CONVERSION FORMULAS

16-Bit Temperature Data Format

Positive Temperature = *ADC Code* (dec)/128

Negative Temperature = (*ADC Code* (dec) – 65,536)/128

where *ADC Code* uses all 16 bits of the data byte, including the sign bit.

Pour l'écriture des données :

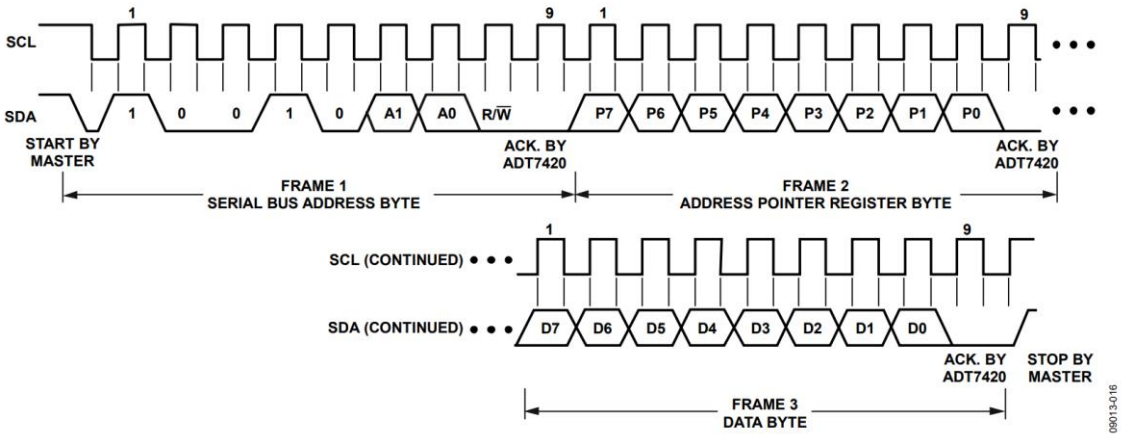


Figure 14. Writing to a Register Followed by a Single Byte of Data

Pour la lecture des données :

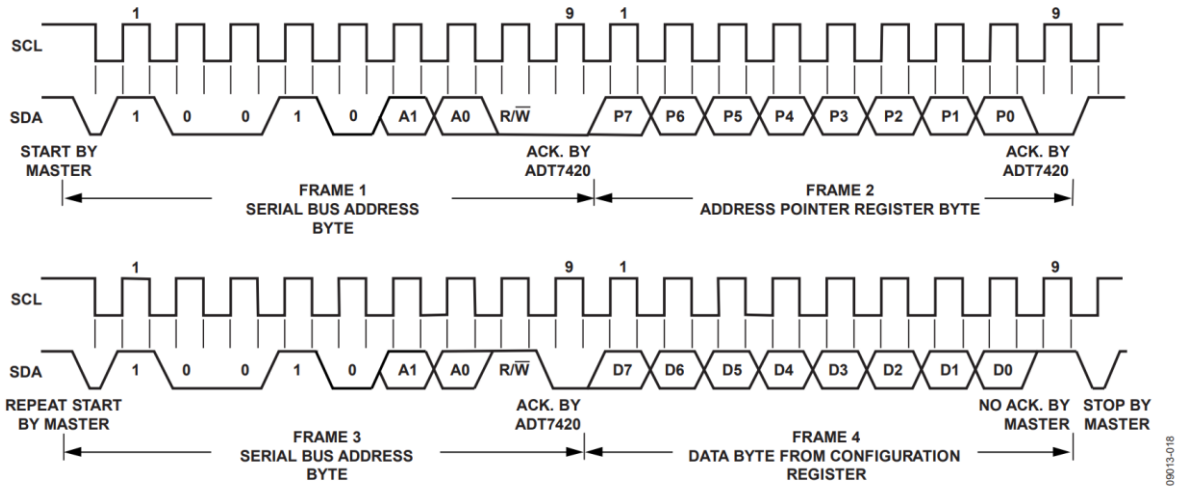
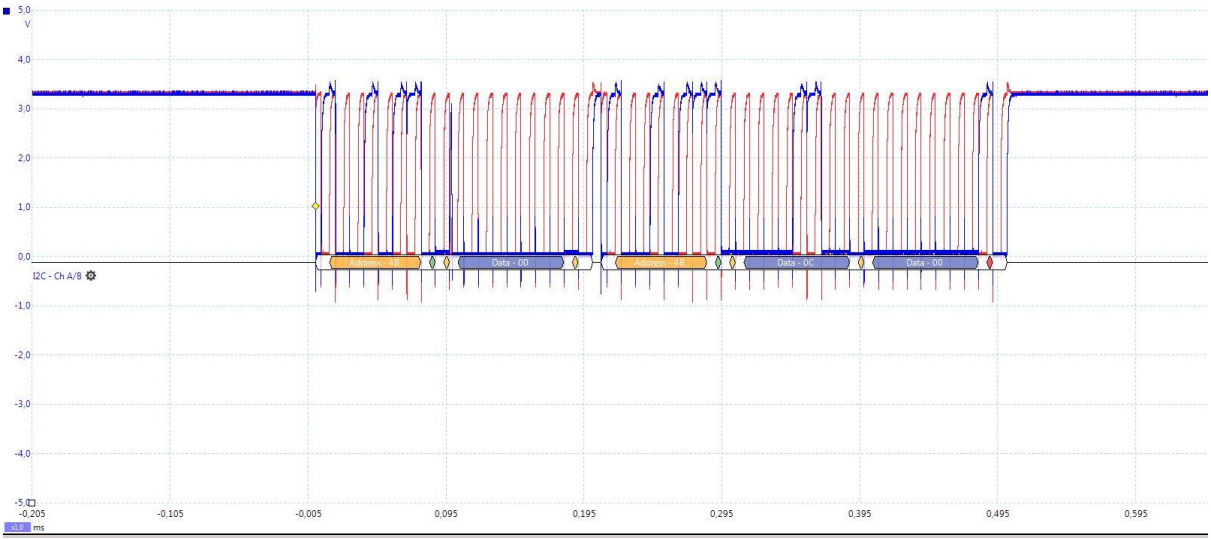


Figure 16. Reading Back Data from the Configuration Register

Lecture des données

Maintenant que nous avons les données que nous devons recevoir, on peut vérifier les traces que nous recevons via un picoscope.
Nous pouvons voir ces données via le graphique ci-dessous :



En rouge on voit l'horloge et en bleu des data

0x4B correspond à l'adresse de la DATASHEET

et on reçoit comme données pour la température : 0x0C00

ce qui correspond en décimal à : 3072

En appliquant le calcule de la datasheet pour convertire la DATA en °C (on à une température positif donc : DATA en décimal / 128)

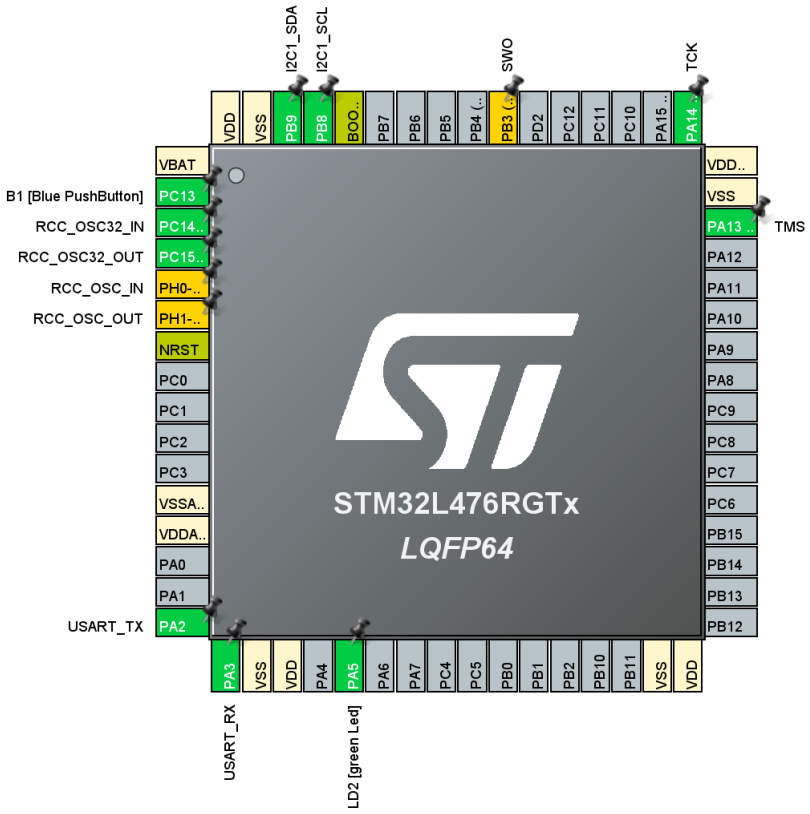
On obtient : 24°C

On a pris cette mesure en salle de classe, elle est donc dans le bon ordre de grandeur.

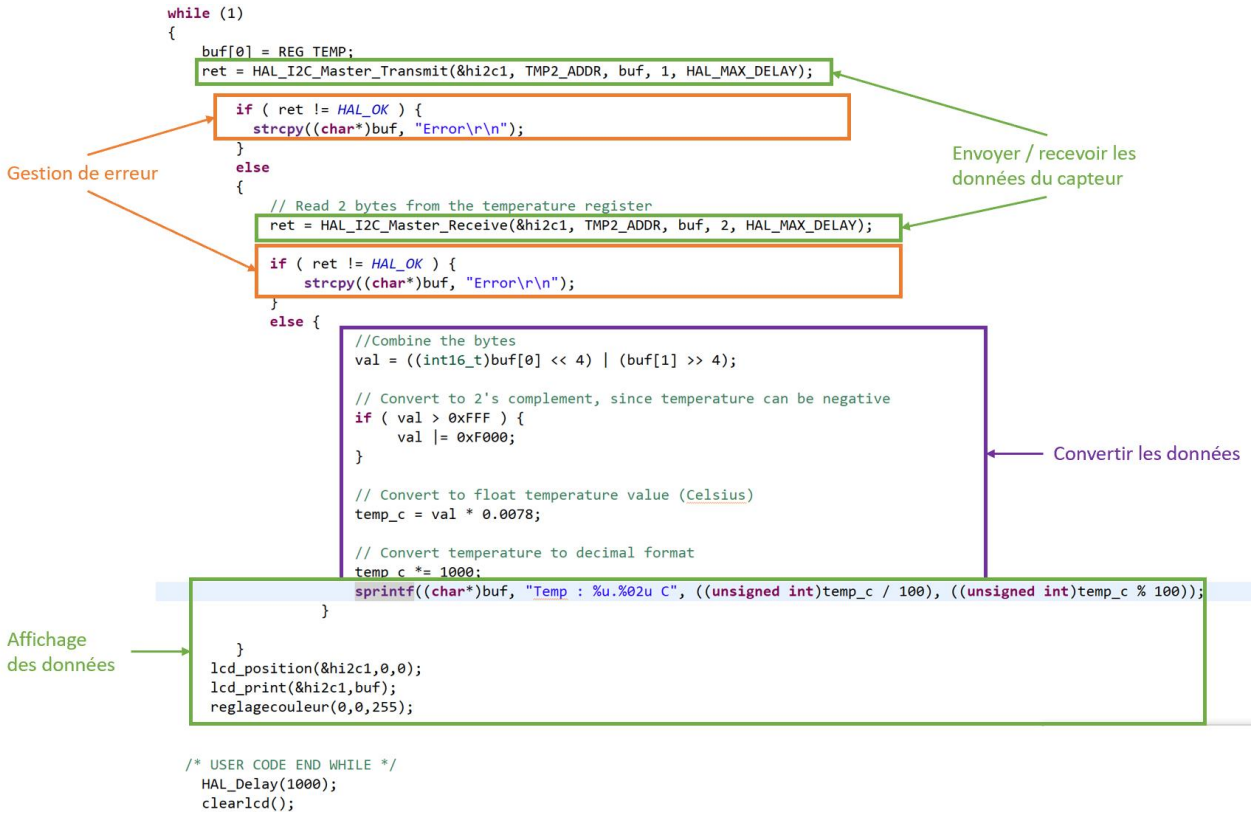
Pour être sûr que le capteur marche bien, nous avons mis notre doigt sur le capteur pour le chauffer et nous avons vu que la température augmentait.

Configuration et code du STM32

Nous avons configuré l'I2C pour envoyer et recevoir des données



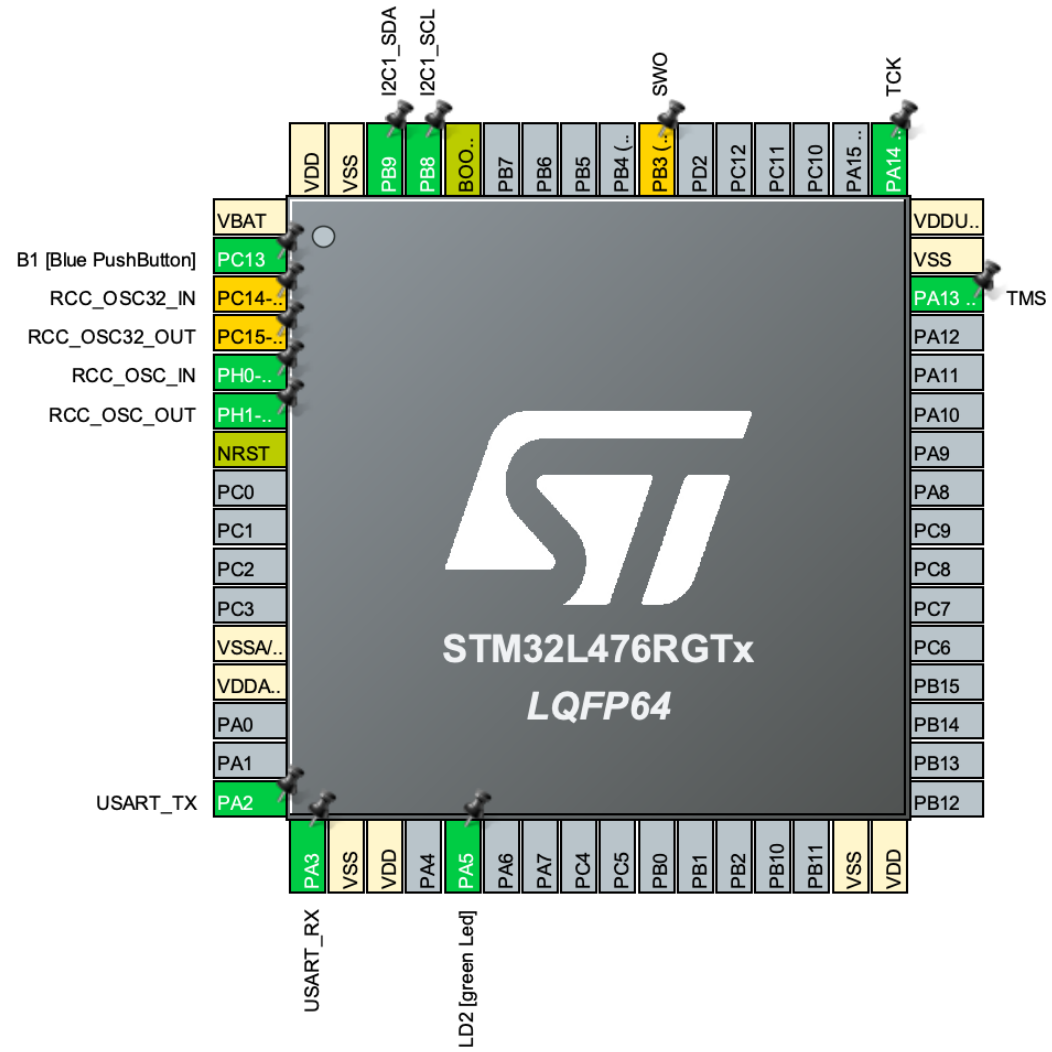
Nous pouvons voir le code ci-dessous pour, on comment par envoyer des données, une fois que le capteur reçoit les données, le capteur envoie les données de température.
Quand nous recevons les données de température nous les mettons dans une variable, on vérifie si la température est négative (négative → faire le complément à 2) puis on fait le calcule (diviser par 128).
Ensuite on peut afficher les données sur l'écran LCD
Et on attend pendant 1 sec avant de reprendre des mesure




```
while (1)
{
    buf[0] = REG_TEMP;
    ret = HAL_I2C_Master_Transmit(&hi2c1, TMP2_ADDR, buf, 1, HAL_MAX_DELAY);

      if ( ret != HAL_OK ) {
        strcpy((char*)buf, "Error\r\n");
      }
      else
      {
          // Read 2 bytes from the temperature register
          ret = HAL_I2C_Master_Receive(&hi2c1, TMP2_ADDR, buf, 2, HAL_MAX_DELAY);

          if ( ret != HAL_OK ) {
              strcpy((char*)buf, "Error\r\n");
          }
          else {
                  //Combine the bytes
                  val = ((int16_t)buf[0] << 4) | (buf[1] >> 4);

                  // Convert to 2's complement, since temperature can be negative
                  if ( val > 0xFFF ) {
                       val |= 0xF000;
                  }

                  // Convert to float temperature value (Celsius)
                  temp_c = val * 0.0078;

                  // Convert temperature to decimal format
                  temp_c *= 1000;
                  sprintf((char*)buf, "Temp : %u.%02u C", ((unsigned int)temp_c / 100), ((unsigned int)temp_c % 100));
              }

      }
    lcd_position(&hi2c1,0,0);
    lcd_print(&hi2c1,buf);
    reglagecouleur(0,0,255);

  /* USER CODE END WHILE */
    HAL_Delay(1000);
    clearlcd();
```

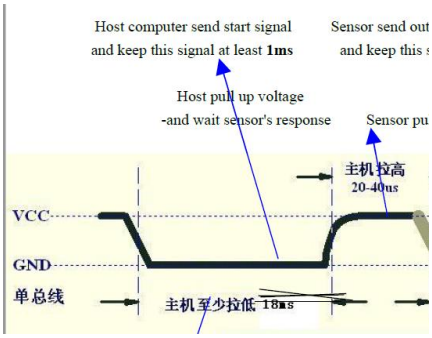
Capteur de Température (DHT22)

Dans ce TP de base nous voulons capter la température et l'humidité d'une pièce grâce au capteur de DHT22 et le microcontrôleur STM32L476rg, puis afficher leur valeur sur notre écran LCD.



Tout d'abord on a notre écran LCD qui fonctionne avec un protocole de communication I2C. Il faut donc paramétrer un port SDA et SCL sur le microcontrôleur pour permettre la communication avec l'écran LCD. On peut voir les ports sur le schéma au-dessus (PB8 & PB9).

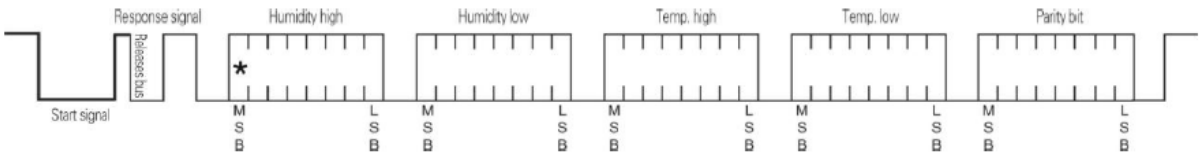
Ensuite on regarde la datasheet du DHT22 pour voir comment elle s'initialise



Sur la datasheet on peut savoir qu'il faut que la broche du signal envoyé du microcontrôleur vers le DHT22 soit mise à 0 pendant au minimum 1 ms et au maximum 18 ms, puis doit être mise à 1 pendant 20 à 40 µs. C'est aussi la raison pour laquelle on n'a pas défini le porte PA1 , car PA1 c'est input aussi que output.

Sur la datasheet les données seront sous la forme suivante :

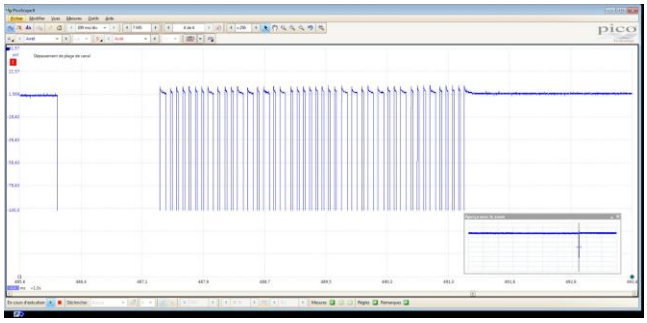
```
/*commence la communication avec le capteur*/

HAL_Delay(3000);
Data_Output(GPIOA, GPIO_PIN_1); //info vers le capteur
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
DWT_Delay_us(1200); //signal de commande
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
DWT_Delay_us(30); //signal de commande
Data_Input(GPIOA, GPIO_PIN_1); //info vers le microcontroleur
```

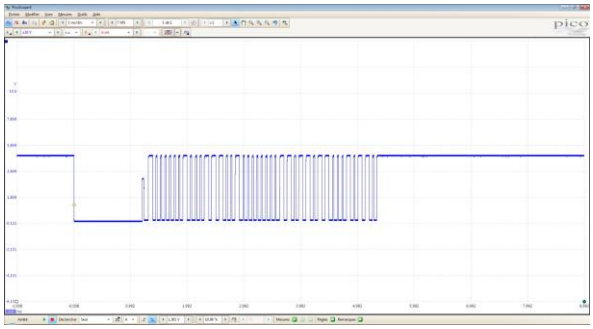


Pic5: AM2302 Single-bus communication protocol

Donc le 16 premier bits c'est pour la valeur de l'humidité et en suit 16 bit c'est la valeur de température et 8 bits de parité à la fin qui servira à vérifier qu'on n'ait pas d'erreur. Normalement la valeur de l'octet de parité est égale à la somme des 4 octets précédents.

Par l'oscilloscope on peut voir c'est même que la datasheet :





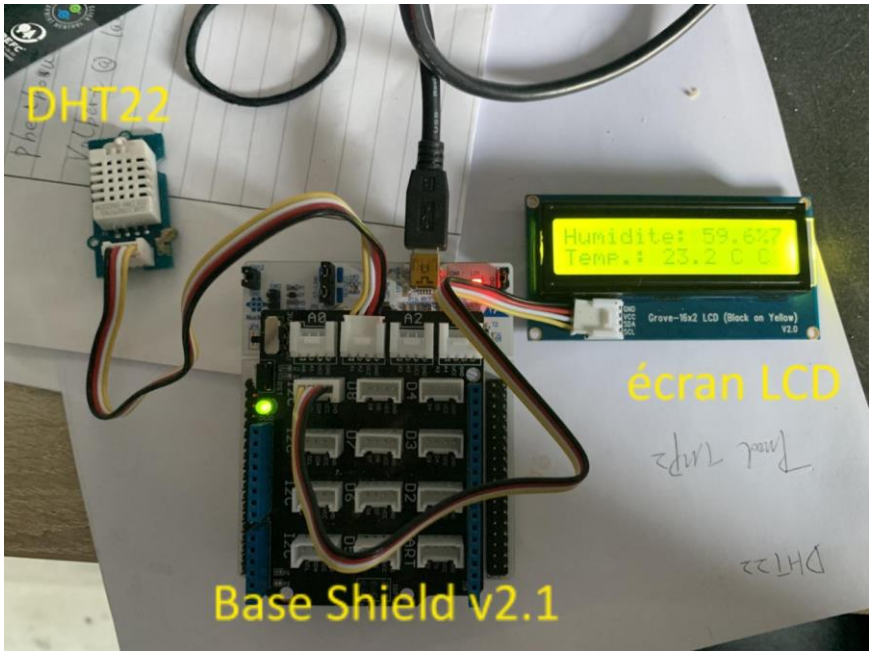
Pour la code on peut écrit comme ça pour lire les valeur:

```
Read_data(&dataH1); //dans la library HT.c
Read_data(&dataH2);
Read_data(&dataT1);
Read_data(&dataT2);
Read_data(&SUM);

check = dataH1 + dataH2 + dataT1 + dataT2; //pour verifier la lecture dans le IDE

RH = (dataH1<<8) | dataH2;
TEMP = (dataT1<<8) | dataT2;

Humidite = RH / 10.0;
Temperature = TEMP / 10.0;

HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET); //pour la prochaine lecture
```

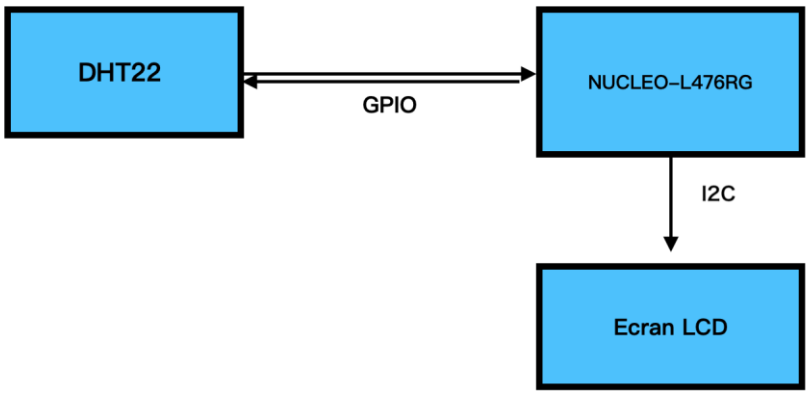
Enfin pour on peut voir les valeur on doit finir les partie de LED:

```
/*commence transmission vers LCD*/
clearlcd();
sprintf(bufRH,"Humidite: %.1f", Humidite);
sprintf(bufT, "Temp.: %.1f C", Temperature);
lcd_position(&hi2c1,0,0);
lcd_print(&hi2c1,bufRH);
lcd_print(&hi2c1,"%");
lcd_position(&hi2c1,0,1);
lcd_print(&hi2c1,bufT);
reglagecouleur(0,0,255);
```

Comme ça on a:



Pour le schéma fonctionnel on a :



TP BE X-Nucleo-IKSO1A2

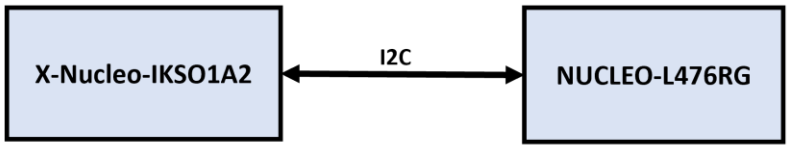
Le but de ce projet est de connaître l'orientation et le mouvement d'un objet et de l'afficher en 3D.

Nous utilisons la carte X-Nucleo-IKSO1A2 qui permet de faire cela. Cette carte a plusieurs capteurs :

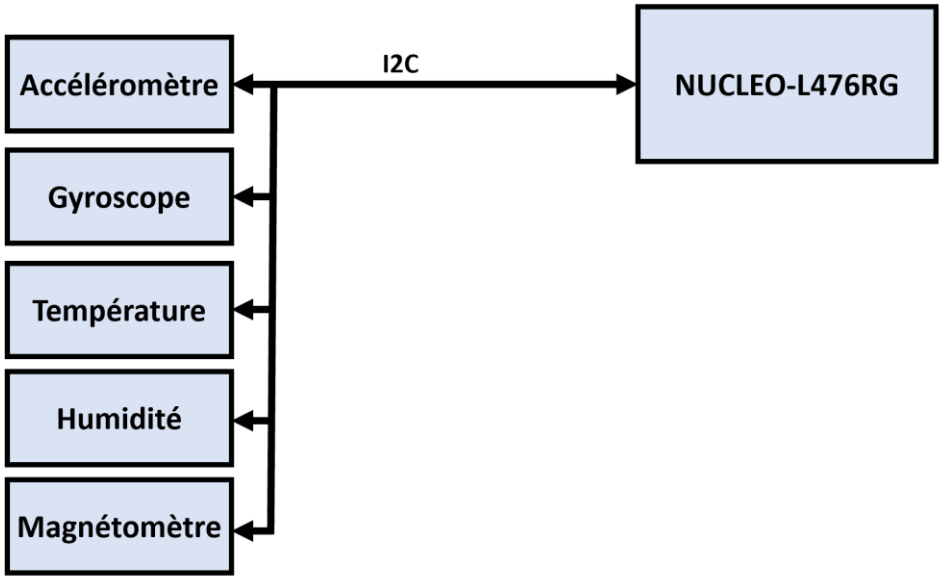
- 2 Accéléromètre (LSM303ARG et LSM6DSL)
- 1 Gyroscope (LSM6DSL)
- 1 Magnétomètre (LSM303ARG)
- 1 capteur d'Humidité (HTS221)
- 2 capteurs de Température (HTS221 et LPS22HB)
- 1 capteur de pression (LPS22HB)

Les deux capteurs qui nous intéressent sont l'accéléromètre et le gyroscope.

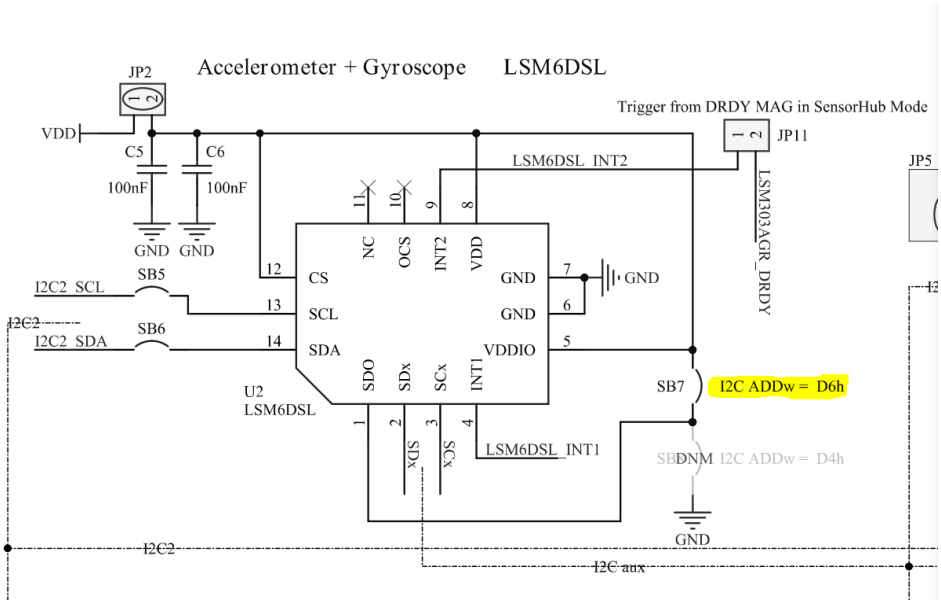
Pour communiquer avec les Capteur nous utilisons l'I2C, on peut voir le fonctionnement avec le schéma fonctionnel ci-dessous :



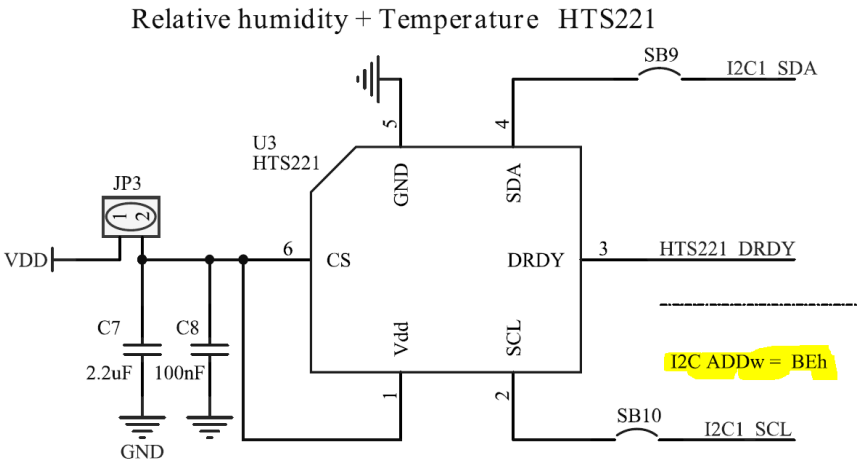
Pour communiquer avec les capteur il faut commencer par envoyer l'adresse du capteur avec lequel on veut communiquer puis lire les données car tous les capteurs sont relié ensemble, ci-dessous nous pouvons voir le schéma fonctionnelle plus détaillé avec les capteurs :



Comme nous nous intéressons principalement au gyroscope nous pouvons regarder son adresse dans la datasheet de X-Nucleo-IKSO1A2 qui : 0xD6



Pour analyser les données nous pouvons utiliser le capteur de température qui a pour adresse : 0xBE



Une des applications pourrais d'intégrer la carte dans le projet TER Skate électrique

Cela pourrait permettre de connaître sa position dans l'espace (degrés d'inclinaison de la pente) et l'accélération du skate.

Pour faire marcher le capteur nous avons fait quelques recherches sur les différentes méthodes pour le faire fonctionner. Nous en avons identifié 2 principales.

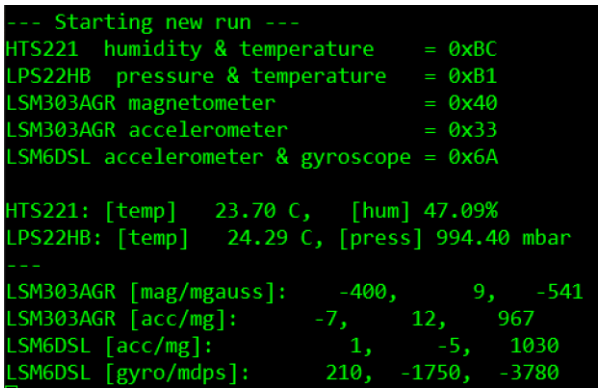
La première est d’utiliser le logiciel MBED,

La deuxième méthode est de le faire marcher avec STM32CubeIDE.

Faire fonctionner la carte avec MBED

MBED est un logiciel Open source en ligne gratuit (https://os.mbed.com/). On peut les compiler directement depuis leur site internet. Nous avons accès à toutes les librairies et des exemples de code pour faire marcher différents types de capteur, dans notre cas nous utiliserons les programmes en rapport avec notre carte :

Ce logiciel va nous permettre de tester rapidement tous les capteurs sur la carte. Via le lien suivant X-NUCLEO-IKS01A2 Motion MEMS and Environmental Sensor | Mbed , nous pouvons avoir des programmes pour tester notre carte. Ce programme va envoyer les données sur un port de l'ordinateur, nous avons donc pu récupérer les données avec le logiciel TeraTerm.
On a pu voir les données suivante :



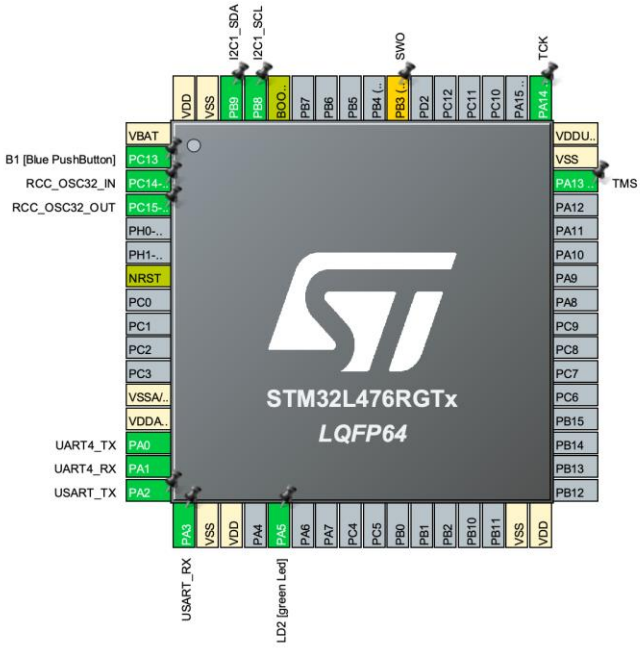
On a effectué quelques tests pour vérifier si les capteurs marchaient bien.
Pour le gyroscope, nous avons modifié son orientation
Pour l'accéléromètre, nous avons mis en mouvement la carte (accélérer et décélérer)
Pour le magnétomètre, nous pouvons approcher notre téléphone
Pour la température, nous avons mis notre doigt sur le capteur.

Faire marcher le capteur de Température

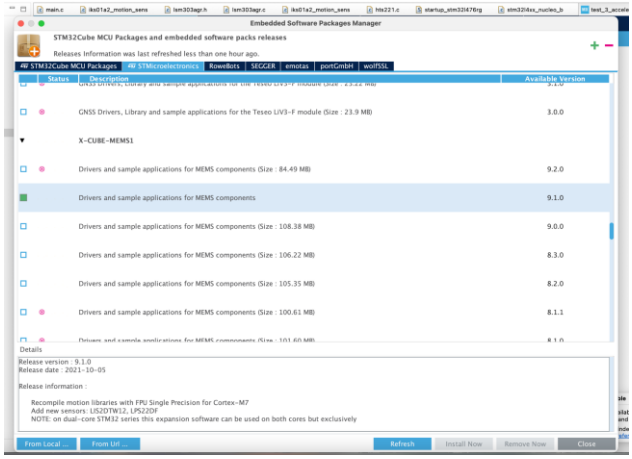
Après avoir affiché les valeurs de tous les capteurs de la carte X-Nucleo-IKSO1A2 avec MBED, nous pouvons travailler avec le logiciel STM32CubeIDE, nous commençons par afficher la valeur du capteur de température.

Avec l'aide de la vidéo suivante :https://www.youtube.com/watch?v=EZ9jxS5A9_s , cette vidéo nous montre comment afficher la valeur du capteur de température.

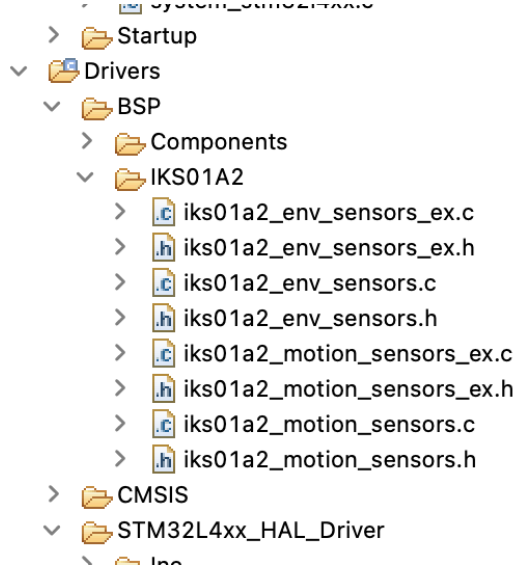
Nous allons vous montrer comment nous avons configuré le logiciel pour afficher la température.



Tout d'abord notre capteur fonctionne avec un protocole de communication I2C. Il faut donc paramétrer un port SDA et SCL sur le microcontrôleur pour permettre la communication avec le capteur. On peut voir les ports sur le schéma au-dessus (PB8 & PB9).



Après on doit télécharger le bibliothèque de notre capteur, la bibliothèque s'appelle : X-CUBE MEMS 1, puis on choisi la version que l'on souhaite, dans notre cas nous avons choisi la dernière version qui est :"Drivers and sample applications for MEMS components V 9.1.0".



Après avoir installé la bibliothèque on peut voir que de nouveau fichier ce sont ajouté dans notre projet, il y a deux types de fichier :

- Les fichiers pour les capteurs d'environnement (env_sensors), il correspond aux capteurs de température, humidité et magnétomètre.
- Et les fichiers pour les capteurs de mouvement (motion_sensors), il correspond à l'accéléromètre et au gyroscope.

Dans ces bibliothèques, nous pouvons utiliser des fonction qui nous servions à utiliser les capteurs :

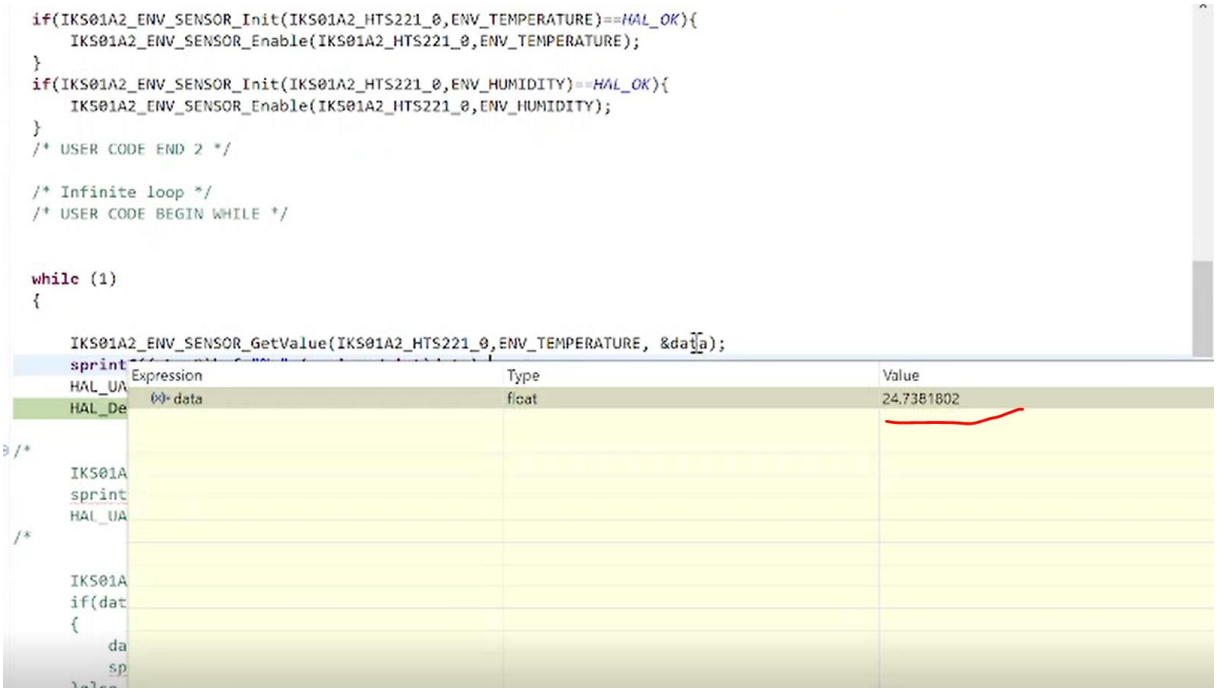
```
int32_t IKS01A2_ENV_SENSOR_GetValue(uint32_t Instance, uint32_t Function, float *Value)
{
  int32_t ret;

  if (Instance >= IKS01A2_ENV_INSTANCES_NBR)
  {
    ret = BSP_ERROR_WRONG_PARAM;
  }
  else
  {
    if ((EnvCtx[Instance].Functions & Function) == Function)
    {
      if (EnvFuncDrv[Instance][FunctionIndex[Function]]->GetValue(EnvCompObj[Instance], Value) != BSP_ERROR_NONE)
      {
        ret = BSP_ERROR_COMPONENT_FAILURE;
      }
      else
      {
        ret = BSP_ERROR_NONE;
      }
    }
    else
    {
      ret = BSP_ERROR_WRONG_PARAM;
    }
  }

  return ret;
}
```

Par exemple la fonction ci-dessous nous servira à récupérer les valeur envoyé par les capteur d'environnement, nous pouvons l'utiliser dans la fonction main :

```
/* USER CODE BEGIN 2 */
if(IKS01A2_MOTION_SENSOR_Init(IKS01A2_LSM6DSL_0,MOTION_GYRO)==HAL_OK){
    IKS01A2_MOTION_SENSOR_Enable(IKS01A2_LSM6DSL_0,MOTION_GYRO);
}

if(IKS01A2_MOTION_SENSOR_Init(IKS01A2_LSM303AGR_ACC_0,MOTION_ACCELERO)==HAL_OK){
    IKS01A2_MOTION_SENSOR_Enable(IKS01A2_LSM303AGR_ACC_0,MOTION_ACCELERO);
}

if(IKS01A2_ENV_SENSOR_Init(IKS01A2_HTS221_0,ENV_TEMPERATURE)==HAL_OK){
    IKS01A2_ENV_SENSOR_Enable(IKS01A2_HTS221_0,ENV_TEMPERATURE);
}
if(IKS01A2_ENV_SENSOR_Init(IKS01A2_HTS221_0,ENV_HUMIDITY)==HAL_OK){
    IKS01A2_ENV_SENSOR_Enable(IKS01A2_HTS221_0,ENV_HUMIDITY);
}
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */


while (1)
{

    IKS01A2_ENV_SENSOR_GetValue(IKS01A2_HTS221_0,ENV_TEMPERATURE, &data);

    /*
    sprintf((char*)buf,"%u",(unsigned int)data);
```

En faisant ce test on peut voir que tout fonctionne correctement, pour faire les tests nous avons utilisé la fonction débogage en pas à pas.

on peut visualiser le résultat avec l'image ci-dessous :



(Affiche environs 24 °C)

Faire marcher tous les capteurs et afficher les données

Maintenant que nous avons réussi à afficher la température (via le débogueur), nous devons afficher la valeur des autres capteurs et envoyer les données dans un port, ce qui nous permettra de traiter et d'analyser les données.

Dans notre cas, nous nous intéressons aux capteurs de mouvement, principalement au gyroscope.

Nous avons dû analyser les fichiers motion_sensors pour comprendre comment les fonctions fonctionnent (paramètre d'entrées et de sortie).

La fonction qui nous intéresse est la suivant :

```
int32_t IKS01A2_MOTION_SENSOR_GetAxes(uint32_t Instance, uint32_t Function, IKS01A2_MOTION_SENSOR_Axes_t *Axes);
```

- Instance : Permet de choisir le capteur que nous voulons,
- Function : Pour choisir quelle fonction de capteur nous voulons (accéléromètre oui gyroscope)
- *Axes : Nous donne les valeurs que le capteur va renvoyer (position selon les axes x,y,z …)

Une fois les données envoyées sur le port nous pouvons les lire avec le logiciel Arduino, nous aurions pu utiliser d'autre logiciel, mais Arduino permet d'afficher les données sur un graphique.

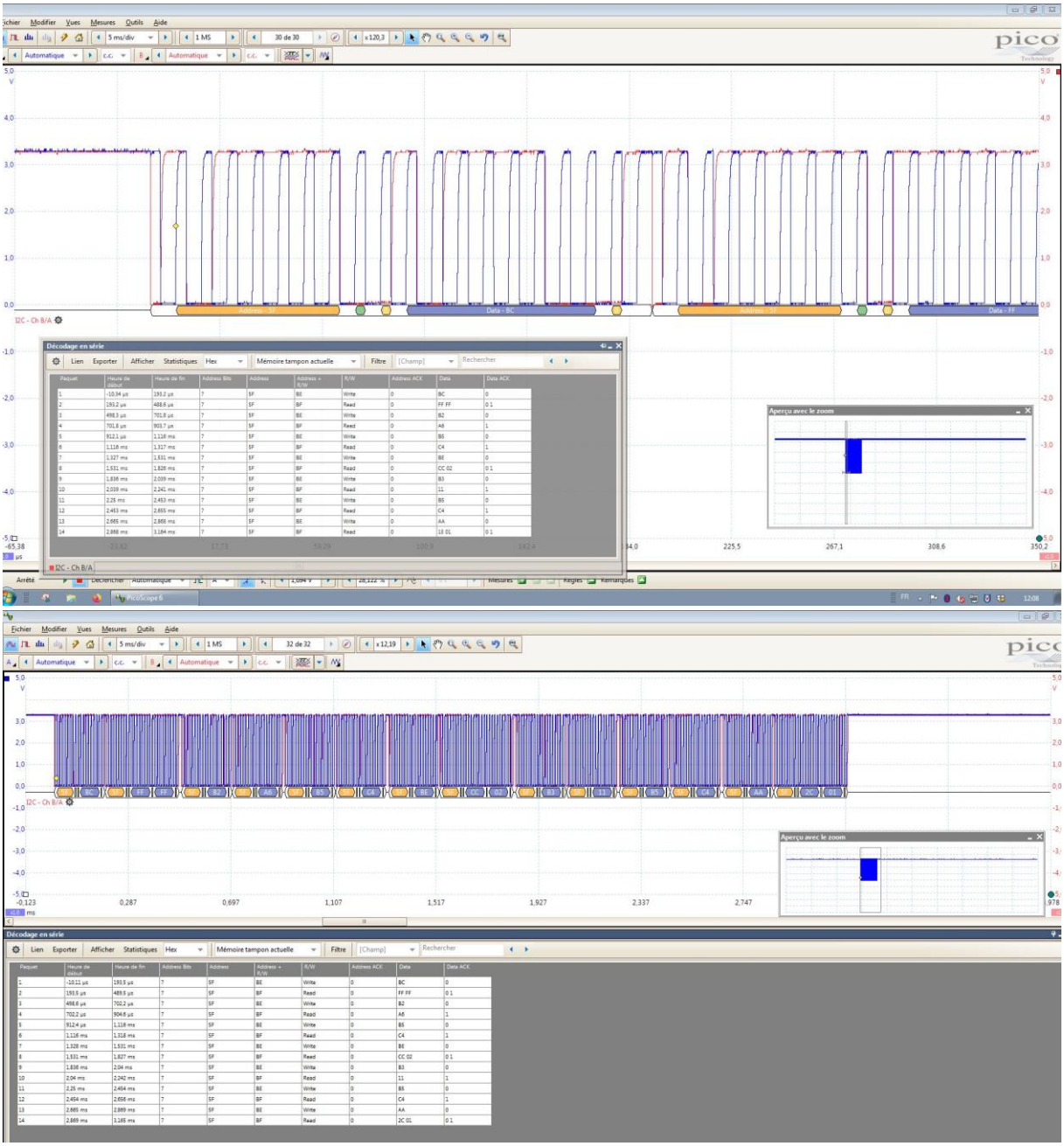
Nous allons commencer par analyser les données reçues (par octet) et vérifier que tout fonctionne comme nous le voulons.

Analyser les données avec un graphique

Pour analyser les données nous utilisons un picoscope.

Comme nous souhaitons vérifier le fonctionnement nous allons prendre le cas où nous utilisons le capteur de température car il y a moins de données à analyser.

Une fois le picoscope brancher au broche I2C de la carte nous pouvons voir les données suivante:

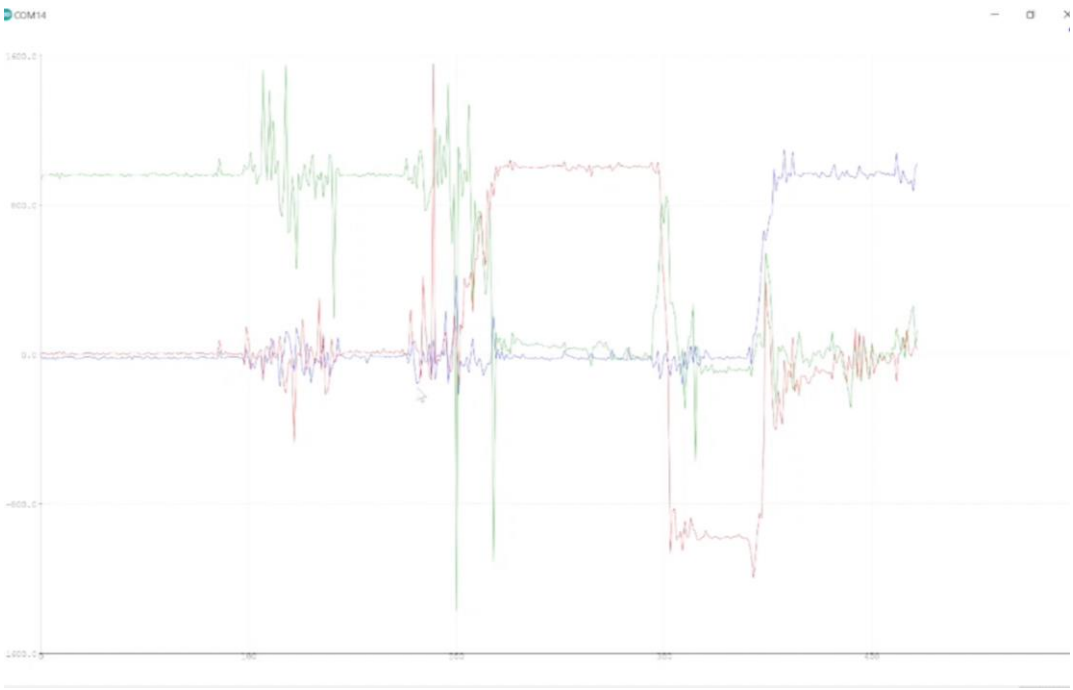


On peut voir que nous avons bien l'adresse du capteur de température, 0xBE (cf datasheet du capteur).

Il faut faire attention quand on lit des données car l'adresse s'écrit sur 8 bites et non sur 7 bits, car il inclut le Write/Read.

Nous avons donc le bon capteur qui communique avec la carte. Nous pouvons maintenant lire les données sur un graphique via le logiciel Arduino (les données que nous lirons sont celle du gyroscope)

Rar arduino Serial plot on a:



Ensuite nous aimerions afficher en 3D les mouvements de la carte (comme sur cette vidéo : https://www.youtube.com/watch?v=E84ROlwPnqM).

Nous avons commencé à faire quelques recherches pour avoir cet affichage et nous avons trouvé le logiciel Procissi 3 qui lit les sorties de arduino et montre les mouvements par modèle 3D.

Conclusion

Ce TP nous a permis d'apprendre à connaître et à comprendre la carte de développement STM32. Elle a renforcé notre compréhension du développement embarqué.

Nous avons pu voir comment fonctionnait le protocole de communication I2C. Avec l'envoie des adresses/données pour communiquer avec le composant et faire fonctionner le composant, et avec la réception des données (adresse et data).

Nous avons pu expérimenter le cas où nous avions des valeurs négatif (complément à 2)

Pour aller plus loin nous pouvons soit l'ajouter à d'autre projet déjà existant, par exemple l'ajouter à un Skate électrique, soit développer la partie mouvement de la carte en affichant en 3D les l'orientation de la carte et les accélération qu'elle subit.

Nous avons aussi la possibilité de connecter d'autres capteurs.

Bibliographies :

PmodTMP2

Datasheet PmodTMP2: https://www.mouser.fr/datasheet/2/609/ADT7420-878995.pdf

Exemple de code pour le PmodTMP2 : https://www.digikey.be/en/maker/projects/getting-started-with-stm32-i2c-example/ba8c2bfef2024654b5dd10012425fa23

Base Shield v2.1 information et achat : https://wiki.seeedstudio.com/Base_Shield_V2/

Base Shield v2.1 schéma : https://files.seeedstudio.com/wiki/Base_Shield_V2/res/Base%20Shield%20v2_SCH.pdf

TP BE X-Nucleo-IKSO1A2

MBED code et logiciel : X-NUCLEO-IKS01A2 Motion MEMS and Environmental Sensor | Mbed

Tuto MBED afficher toutes les données : https://youtu.be/Q0wA9UqgPQ0

Datasheet X-NUCLEO-IKSO1A2 : Motion MEMS and environmental sensor expansion board for STM32 Nucleo

Tuto installation Tera Term : Install Tera Term and connect CISCO Router through SSH Windows 10 2018 - YouTube