Université Paul Sabatier

Compte Rendu de TP :

BE TP de base

Auteurs :

Abderrahmane AMOUR
Mohammed LATRECHE

Encadrant :

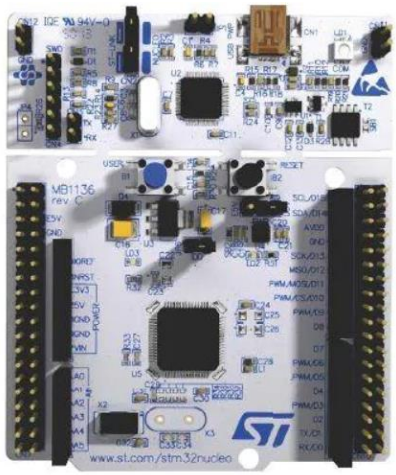
Mr. Thierry PERISSE

Introduction :

Dans ce TP nous allons configurer les deux capteurs pour mesurer : la température, l'humidité, et la pression, ensuite, nous allons afficher les mesures sur un écran LCD.

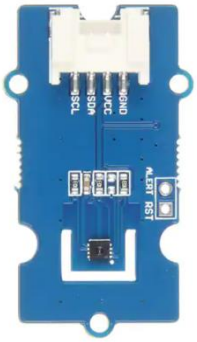
Matériel utilisé :

- Une carte STM32L152RE
- Un capteur SHT31
- Un capteur BMP280
- Un écran LCD JHD1802M1
- Shield Arduino



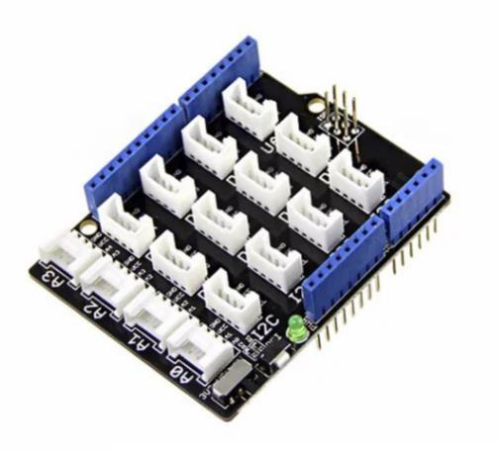
Carte STM32L152RE

Capteur BMP280



Capteur SHT31

Écran LCD JHD1802M1



Shield Arduino

1. Définir les périphériques utilisés et identifier ses pins sur la carte :

Pour configurer les deux capteurs et l'écran LCD nous allons utiliser le protocole I2C (Inter-Integrated Circuit) qui est un protocole de communication série entre un périphérique maître et un esclave.

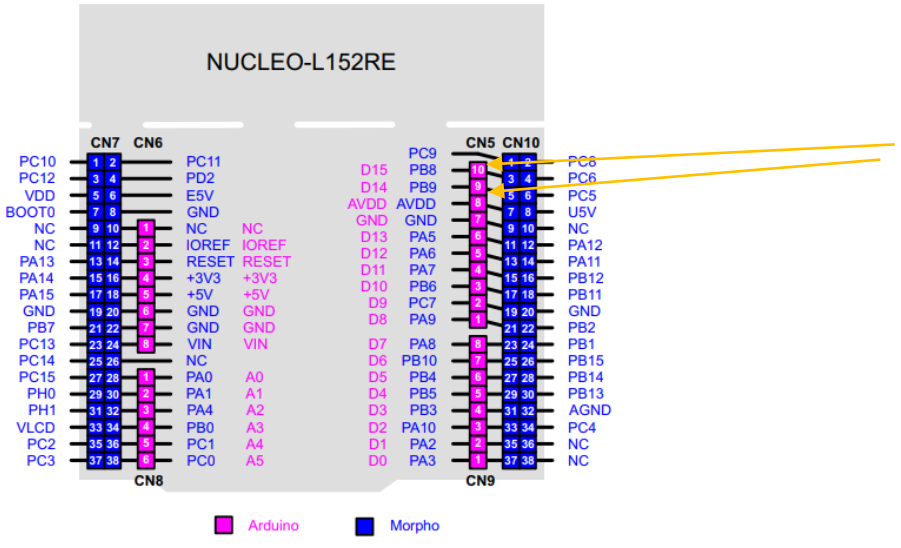
Sur le guide Nucleo de la carte, on trouve que le I2C est disponible sur les pins PB8 et PB9 de la carte STM32 ou sur les broches D15 et D14 d'Arduino.

ON	PB9 and PB8 (I2C) are connected to A4 and A5 (pin 5 and pin 6) on ARDUINO® connector CN8 and ST morpho connector CN7 as I2C signals. Thus SB56 and SB51 must be OFF.

Pour afficher les mesures sur une fenêtre en PC, nous allons utiliser le protocole UART2 et le logiciel Realterm, le module uart2 est relié avec les deux pins PA2 et PA3, il est relié au port USB pour la communication série avec le PC.

A l'aide de guide Nucleo, on peut identifier les deux pins :

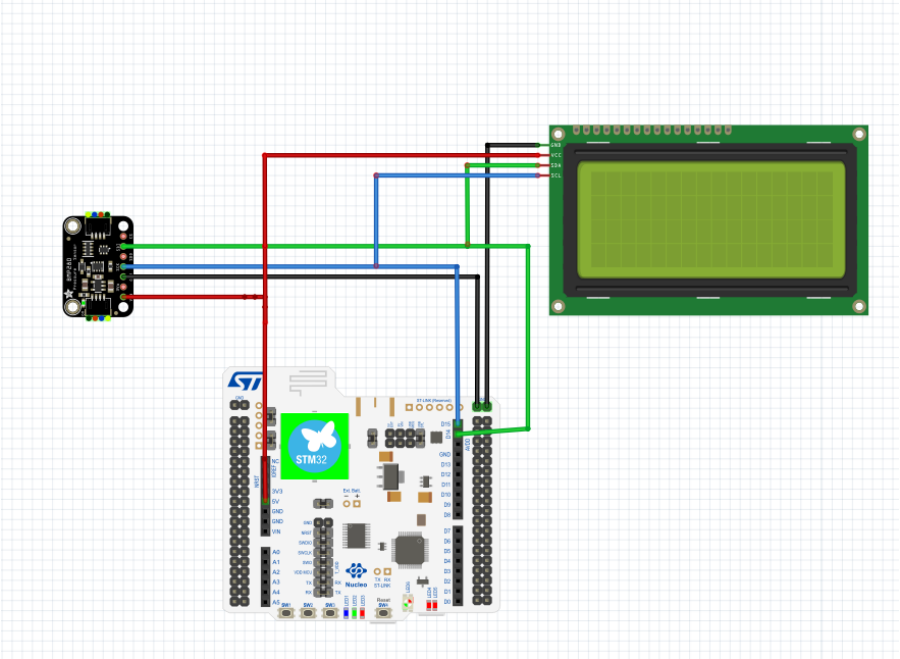
Figure 22. NUCLEO-L152RE



2. Schémas de câblage :

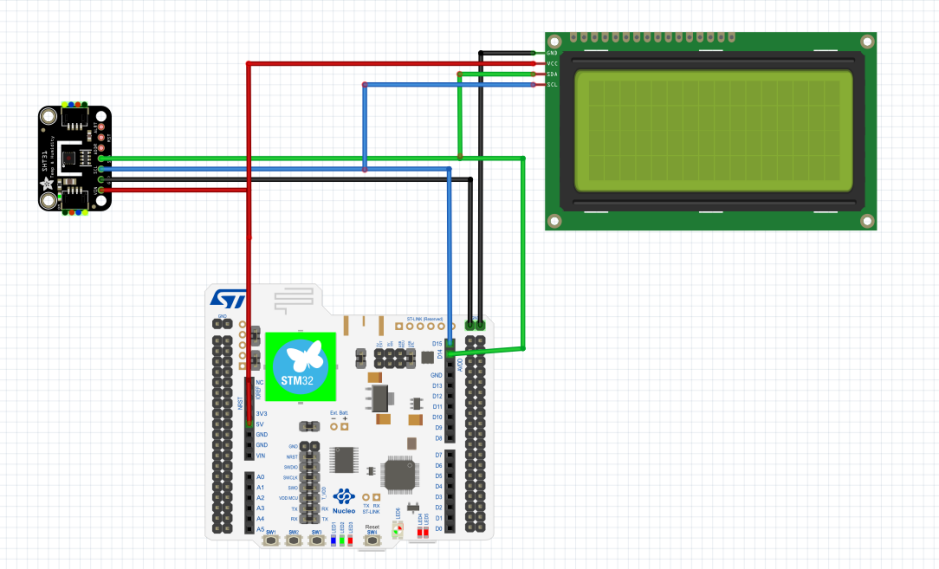
I. BMP280 :

A l'aide de logiciel Fritzing, on présente le schéma de câblage, le capteur utilise 4 câbles, SDA pour les données, SCL pour l'horloge, 5v et GND pour l'alimentation :



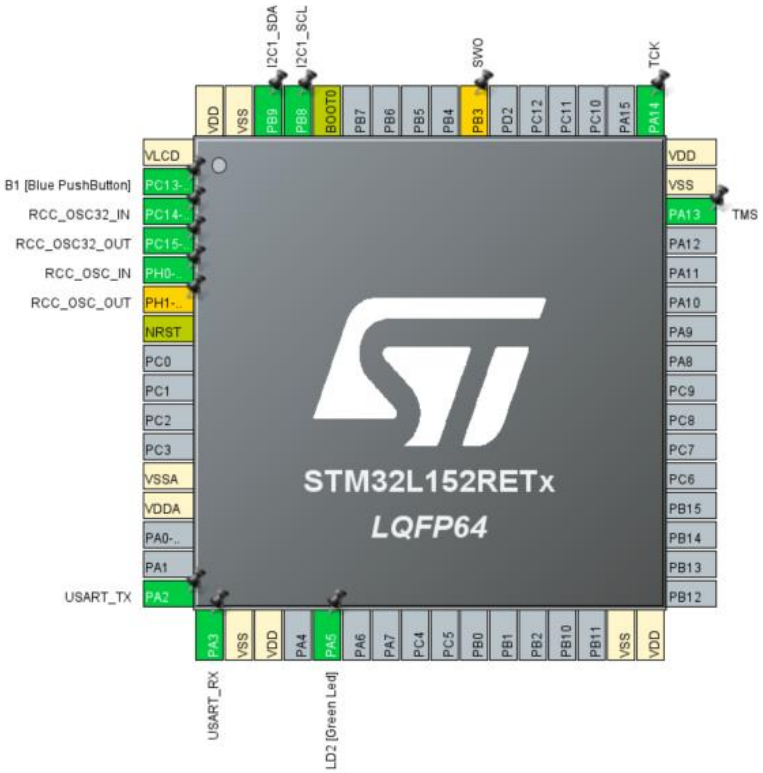
II. SHT31 :

le SHT31 est un capteur d'humidité et de température de haute précision, compact et peu coûteux, avec une excellente stabilité à long terme et une faible consommation d'énergie.



3. Configuration du projet à l'aide de logiciel CubeMX :

On crée un projet BMP280 avec STM32CubeIDE, et le configure à l'aide de CubeMX :



A l'aide de CubeMX on active i2c sur les broches PB8 et PB9, on active aussi les paramètres par défaut.

4. Programme :

I. Capteur BMP280 :

Pour pouvoir utiliser le capteur BMP280, on utilise une bibliothèque prête crée par « **Sheinz »,** on effectue des modifications sur la bibliothèque :

- On enlève tous les lignes de code qui ont relation avec la mesure d'humidité, car notre capteur ne la mesure pas

-Cette bibliothèque supporte deux capteurs : BMP280, BME280, donc on enlève tous les lignes qui concernent le BME280.

-Sur notre main : on initialise les paramètres du capteur, on définit l'adresse du capteur et on sélectionne le module i2c utilisé, dans notre cas il s'agit du module i2c1 avec les lignes du code :

```
bmp280_init_default_params(&bmp280.params);
bmp280.addr = BMP280_I2C_ADDRESS_1;
bmp280.i2c = &hi2c1;
```

-Avec les lignes suivantes on peut lire la valeur de la température et la pression :

```
while (!bmp280_read_float(&bmp280, &temperature, &pressure)) {
size = sprintf((char *)Data,"Temperature/pressure reading failed\n\r");
HAL_UART_Transmit(&huart2, Data, size, 1000);
for(int i=0;i<100000;i++);
}
```

-Dans le cas où la lecture ne peut pas être effectuée (capteur débranché, capteur défectueux…), cette boucle va réessayer la lecture en affichant un message pour debugger.

II. Sht31:

Pour pouvoir utiliser le capteur on crée une bibliothèque nommée sht31 (code source + fichier entête), on crée une fonction nommée : **humidity_read_value. On utilise les fonction de la bibliothèque HAL, pour pouvoir communiquer avec le capteur :**

1. D'abord on envoie l'adresse du capteur du capteur avec la ligne :

```
ret = HAL_I2C_Master_Transmit( hi2c1, CAPTEUR_ADRS, buf, 2, HAL_MAX_DELAY);
```

On vérifie bien que le capteur a bien reçu, et l'adresse envoyée correspond bien à l'adresse du capteur, dans le cas contraire on affiche une erreur à l'aide de USART2.

```
if ( ret != HAL_OK)
{
strcpy((char*)buf, "erreur_T!!\r\n");
HAL_UART_Transmit(huart2, buf, sizeof(buf), 1000);
}
```

2. On reçois la réponse du capteur avec la commande :

```
ret = HAL_I2C_Master_Receive(hi2c1, CAPTEUR_ADRS, buf, 6, HAL_MAX_DELAY);
```

On vérifie bien qu'on a bien reçu sinon on affiche une erreur.

3. On récupère les valeurs de la température et l'humidité avec les lignes du code :

```
value  =   buf[1] | buf[0] << 8;

temp = -45 + 175 * ( (float)value / 65535);
Entier_part = (int) temp;
Decimal_part = temp;
Decimal_part *= 100;
Decimal_part = Decimal_part - (Entier_part * 100);
value = buf[4] | buf[3] << 8;

umid = -49 + 315 *( (float)value / 65535);
```

III. Ecran LCD :

Pour l'écran LCD, on utilise la bibliothèque fournie par un ancien étudiant en SME, on rajoute le code source et le fichier entête à notre projet pour pouvoir utiliser la bibliothèque.

Sur notre main :

- On initialise d'abord l'écran, on met le curseur sur la première ligne et colonne et on règle la couleur :

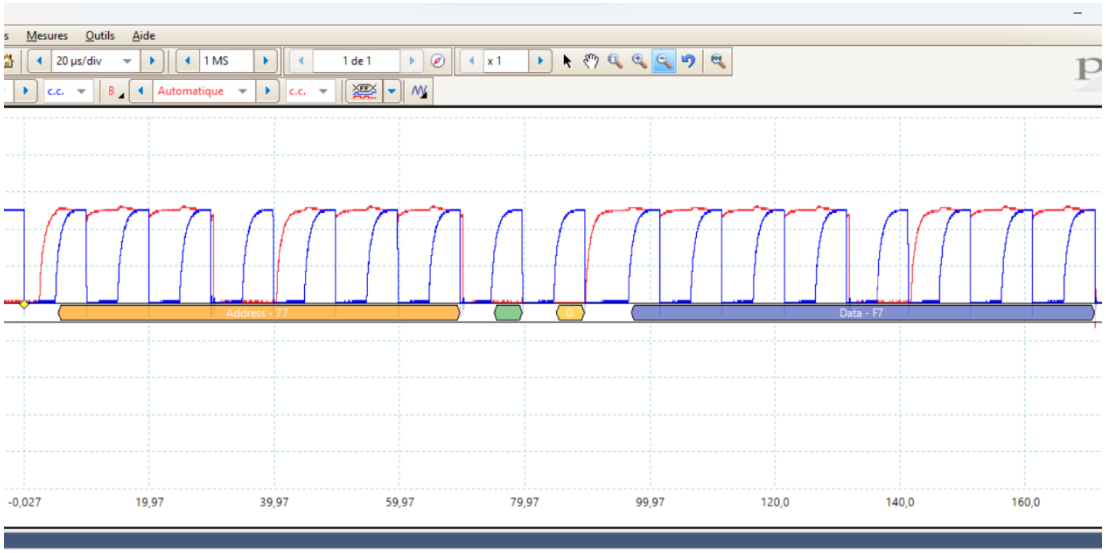
```
lcd_init(&hi2c1, &lcdData); // initialise le lcd
lcd_position(&hi2c1,0,0);
reglagecouleur(0,0,255);
```

-On affiche les valeurs de la température, humidité, pression avec les lignes du code :

```
lcd_position(&hi2c1,0,0);
lcd_print(&hi2c1,"Pres:");
lcd_position(&hi2c1,7,0);
lcd_print(&hi2c1,Data_pres);
lcd_position(&hi2c1,0,1);
lcd_print(&hi2c1,"Temp: ");
lcd_position(&hi2c1,7,1);
lcd_position(&hi2c1,0,1);
lcd_print(&hi2c1,"Hum: ");
lcd_position(&hi2c1,7,1);
lcd_print(&hi2c1,&buf[10]);
lcd_print(&hi2c1,Data_temp);
```

5. Analyse de la trame i2c :

I. BMP280 :



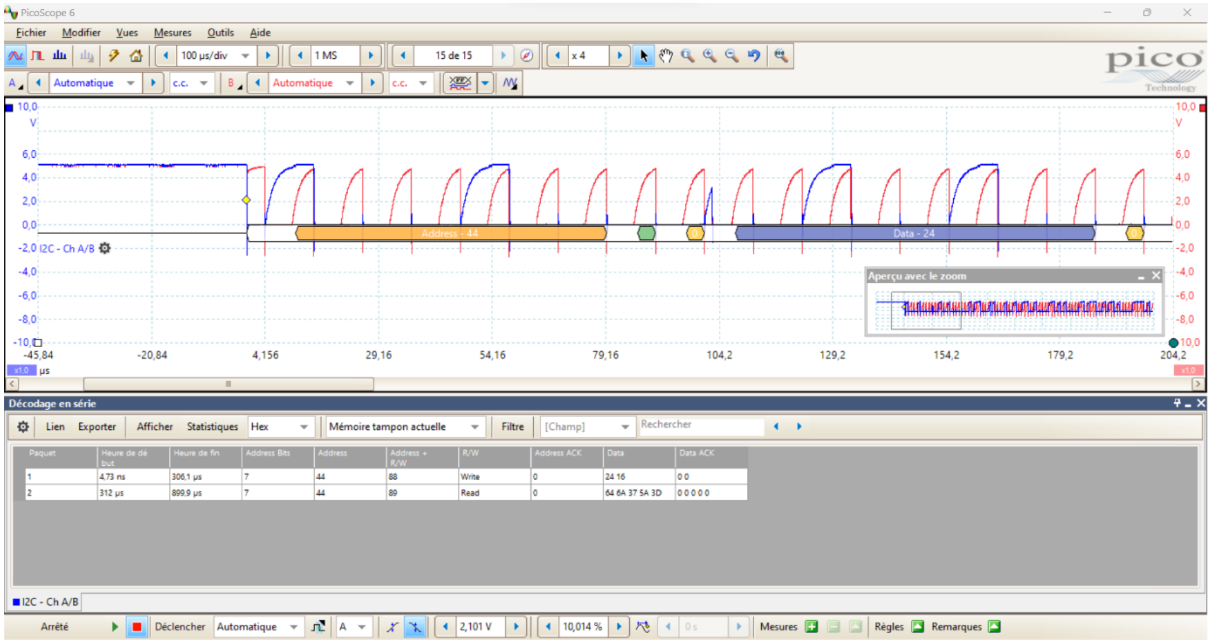
Avec un Pico scope on analyse la trame I2C :

- Sur la trame, on a en bleu l'horloge et en rouge les données du capteur.

- On voit bien que l'adresse du capteur est **0x77**, ce qui correspond bien à l'adresse du capteur affichée dans la datasheet du capteur.

- La commande F7 correspond à la partie **msb** de la valeur de la pression.

II. SHT31 :



Avec un Pico scope on analyse la trame I2C :

- Sur la trame, on a en bleu l'horloge et en rouge les données du capteur.

- On voit bien que l'adresse du capteur est **0x44**, ce qui correspond bien à l'adresse du capteur affichée dans la datasheet du capteur.

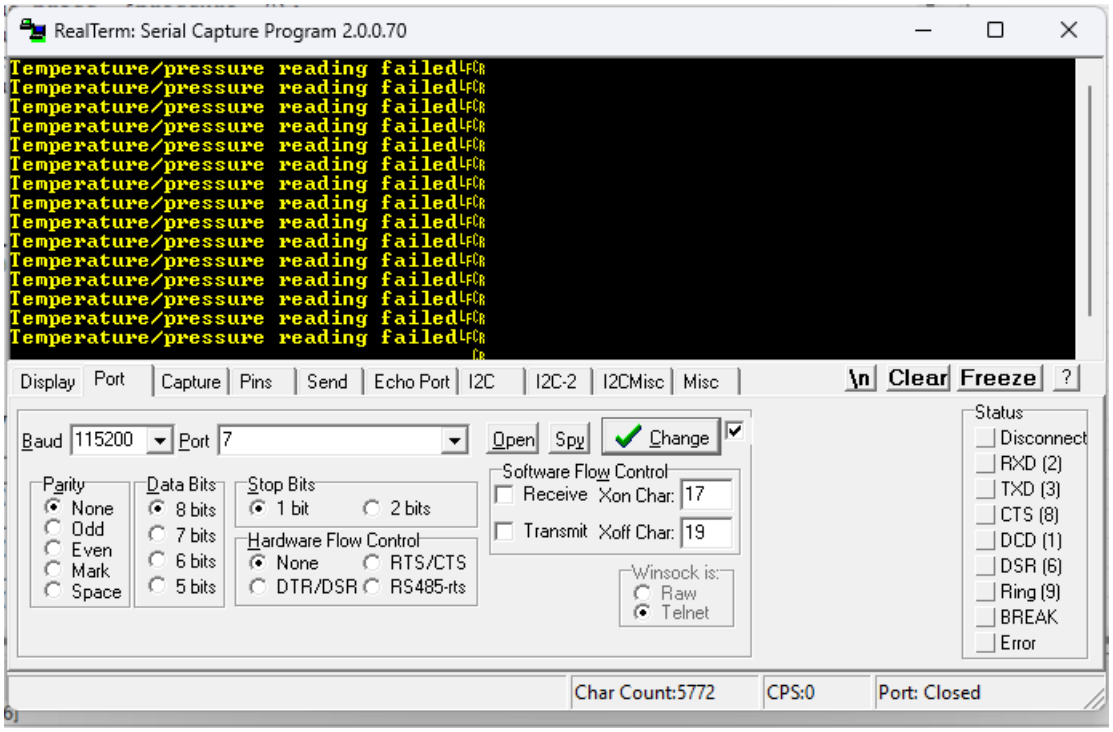
- La commande 0x24 correspond à la partie **MSB** de la valeur de la pression.

6. Debugger :

Pour débugger on a utilisé deux méthodes :

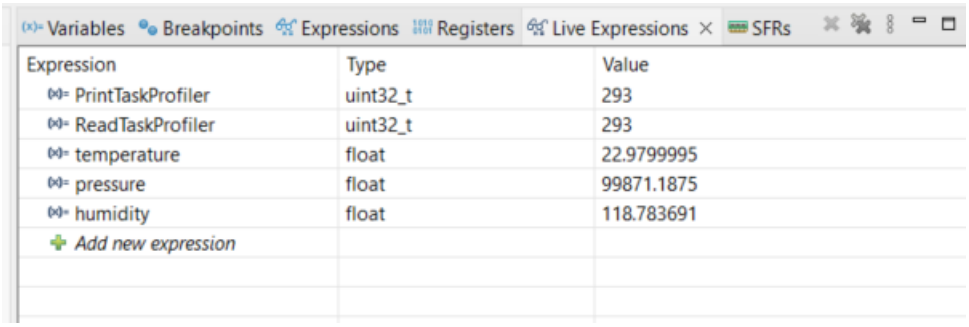
I. la communication série avec le pc :

On a utilisé le logiciel Realterm qui fournit un terminal pour recevoir les messages des erreurs, pour afficher les valeurs de certaines variables pour débugger et fixer certains bugs



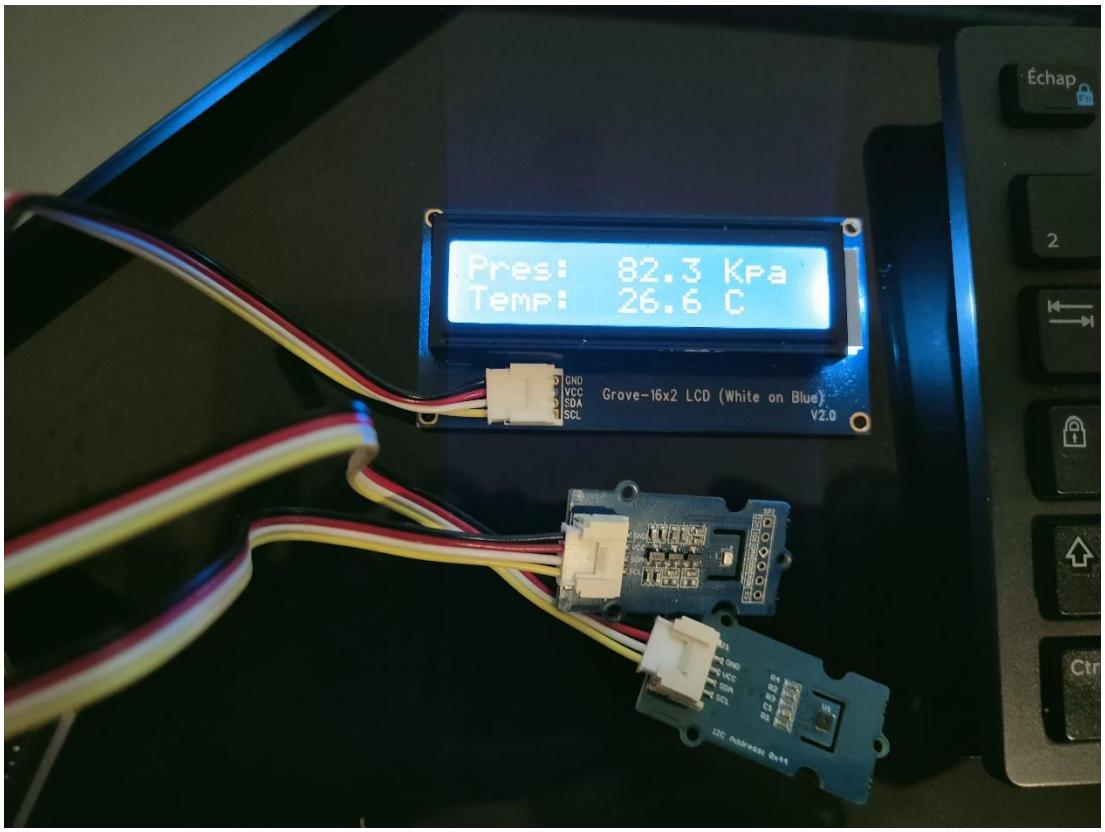
II. La fenêtre Live Expression de CubeIDE :

En mode Debugger, on peut aussi suivre certaines variables à l'aide de la fenêtre Live Expressions, cela nous a beaucoup aider pour fixer certains erreurs :



7. Résultats :

Après avoir fixé toutes les erreurs on affiche les grandeurs mesurés sur l'écran LCD :



8. Conclusion :

Dans cette première partie nous avons éffectué la configuration des deux capteurs SHT31 & BMP280 et voir les données en écran LCD par deux carte SMT32, cette partie nous a permis de renforcer nos compétences en matière de cartes STM32 et en utilisant divers composants et protocoles de communication grâce à cette expérience.