M1 EEA SME 2021/22

Compte Rendu de TP :
UE RS BE - TP de base SHT31

Marine MARRAGOU

Matériel : Ce TP de base fait appel à une Nucleo STM32L476RG, un Grove LCD RGB Backlight V4.0 (I2C) et Grove Temp and Humi Sensor SHT31 (I2C)

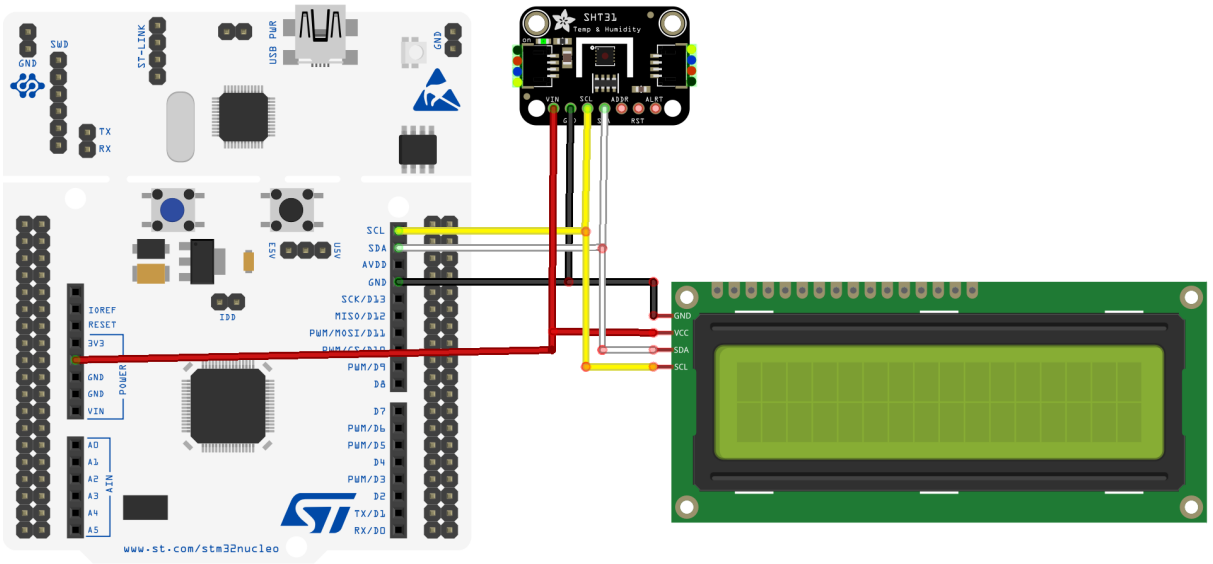
Logiciel : Ce TP utilisera STM32 Cube IDE V1.8.0

But : L'objectif de ce TP est d'afficher l'humidité et la température fournies par le SHT31 sur l'écran LCD

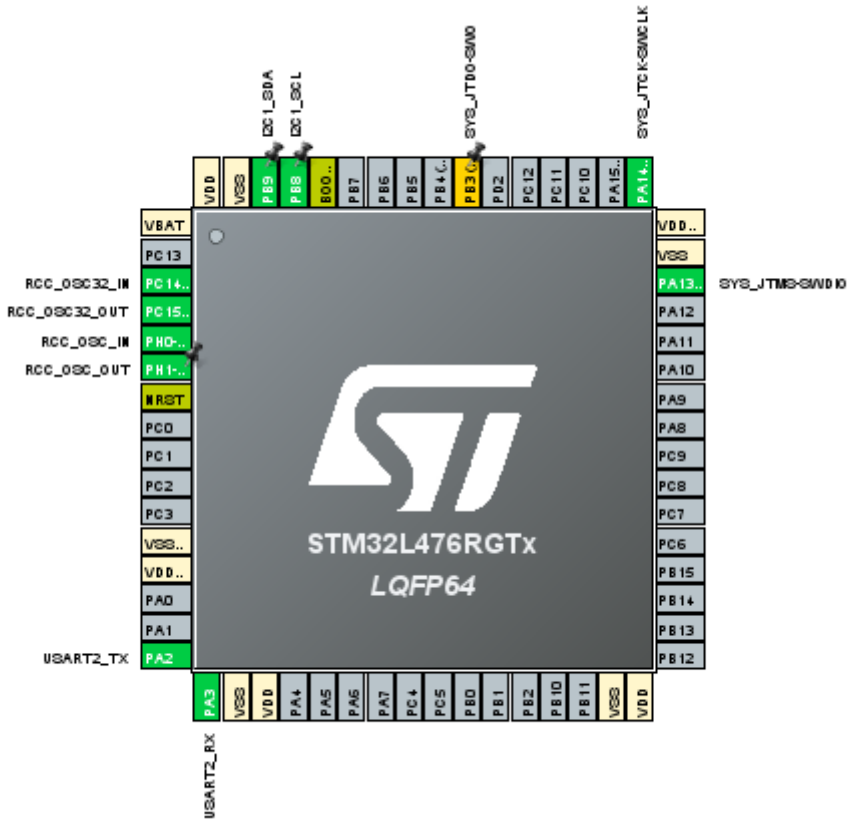
1. Schéma de câblage

Seule l'utilisation de l'écran LCD est traitée ici afin de vérifier la véracité des valeurs trouvées via le capteur SHT31. Toute l'initialisation ayant été réalisée avec l'ensemble du groupe de TP pendant l'une des séances.

Fritzing n'offrant pas la Nucleo STM32L476RG, l'écran LCD Grove et le capteur de température SHT31 de Grove, ces deux éléments ont été remplacés le plus proche équivalent disponible. L'écran LCD et le capteur conversent en I2C donc ceux-ci sont reliés au SDA et SCL de la carte ainsi qu'alimentés. En réalité, nous utilisons un Base Shield V2 de Groove, non représenté, afin de relier proprement notre carte.



2. Initialisation des ports



Tout d'abord, je demande au logiciel d'initialiser les ports par défaut. Ensuite, je paramètre PB8 et PB9 comme mon SCL et SDA. J'aurais également besoin d'un timer en ms donc j'initialise TIM2 à cet effet.

3. Codage

Tout le code du projet de base est disponible sur Github.
Je commence par faire le point sur mes librairies. Au code généré, j'ajoute quelques fichiers .c et .h trouvés en ligne :

- i2c.c/h
- stm32l4xx_it.c/h
- stm32l4xx_hal_conf.c/h

ainsi que caracter.c/h, fourni par un autre étudiant de la promotion, qui me permet de changer le type de mes données pour le futur affichage.
Quelques librairies de base sont également ajoutées au besoin comme math.c/h.

Je travaille maintenant uniquement sur main.c. Tout le code réalisé pour l'écran LCD est intégré au niveau projet ainsi que HAL_Delay_micro.
Afin d'ajouter mon capteur, je crée une fonction dédiée à celui_ci que j'appelle Mesure_Temp_Humi_SHT31. Elle est ensuite appelée et suivie d'un delay à l'intérieur du while(1) de la fonction main :

```
while (1)
{
  /* USER CODE END WHILE */

    Mesure_Temp_Humi_SHT31(Tab);//Appel de la fonction
    HAL_Delay(2000);

  /* USER CODE BEGIN 3 */
}
```

Ma fonction Mesure_Temp_Humi_SHT31 va me servir à dialoguer avec le capteur pour récupérer une donnée, la traiter, la convertir puis l'afficher. Je veux utiliser les fonctions disponibles dans HAL_I2C afin de converser avec mon SHT31 donc je réalise mes initialisations conformément aux indications de la librairie :

```
HAL_StatusTypeDef Ready;//Afin d'utiliser les fonctions de HAL_I2C

uint8_t MSB_comm = 0x24;//Registre de commande cf Datasheet
uint8_t LSB_comm = 0x00;
uint8_t Info[2] = {MSB_comm,LSB_comm};//Groupement des registres
uint8_t Converted[6] = {0};//Réceptionne la réponse du capteur (MSB, LSB, CRC, MSB, LSB, CRC)

uint16_t Sensor_Adress = 0x44<<1;//Adresse propre du capteur
```

Les tableaux et registres sont initialisés en fonction des informations fournies par la datasheet du composant.
Je peux maintenant utiliser mes fonctions :

```
//Adressage
Ready = HAL_I2C_Master_Transmit(&hi2c1, Sensor_Adress, Info, 2, 1000);//Commande

HAL_Delay(20); //Attente entre mesures

//Réponse du capteur
Ready = HAL_I2C_Master_Receive(&hi2c1, Sensor_Adress, Converted, 6, 1000);//Réception des données
```

Ces fonctions requièrent l'adresse de l'I2C utilisé, l'adresse du capteur, un tableau contenant ou réceptionnant des données retournées/de commande, le nombre d'octets des données précédentes et un timeout.
Je passe donc à la conversion des données récupérées dans Converted.
Cette étape nécessite quelques initialisation supplémentaires :

```
uint16_t Conca_T = 0;//Tableau pour le traitement de la valeur capteur
uint16_t Conca_H = 0;
```

Ensuite, je suis scrupuleusement les indications de la datasheet pour les formules de conversion des données :

```
//Conversion des données du capteur
Conca_T = (Converted[0] << 8) | (Converted[1]);//cf Datasheet
Val[0] = -45 + (175 *(Conca_T))/(pow(2,16) - 1);//cf Datasheet

Conca_H = (Converted[3] << 8) | (Converted[4]);
Val[1] = 100 * (Conca_H)/(pow(2,16) - 1);
```

Val[2] est de type double. Or, le format nécessaire à l'affichage sur l'écran LCD est string donc il faut procéder à une conversion à l'aide des librairie précédemment ajoutées :

```
FloatToStr(Val[0],T_unit,2);//Conversion de type pour l'affichage
FloatToStr(Val[1],H_unit,2);
```

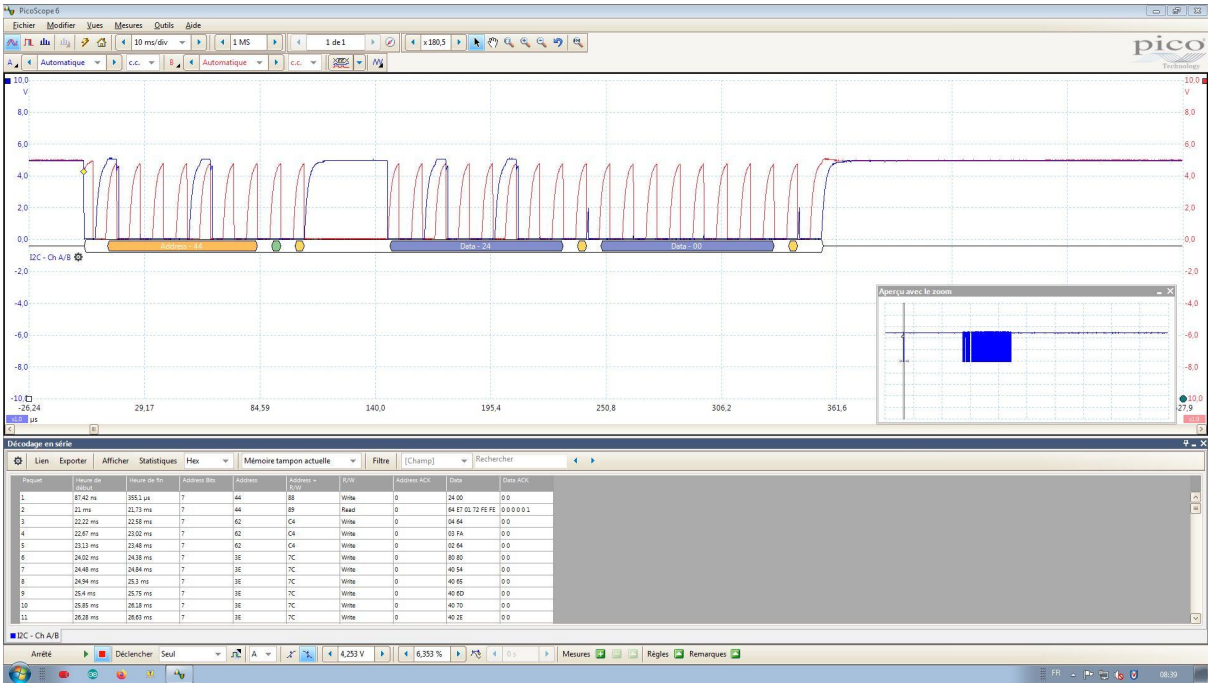
Je passe maintenant à l'affichage de mon information avec les fonctions utilisées lors de l'initiation à l'utilisation de l'écran LCD :

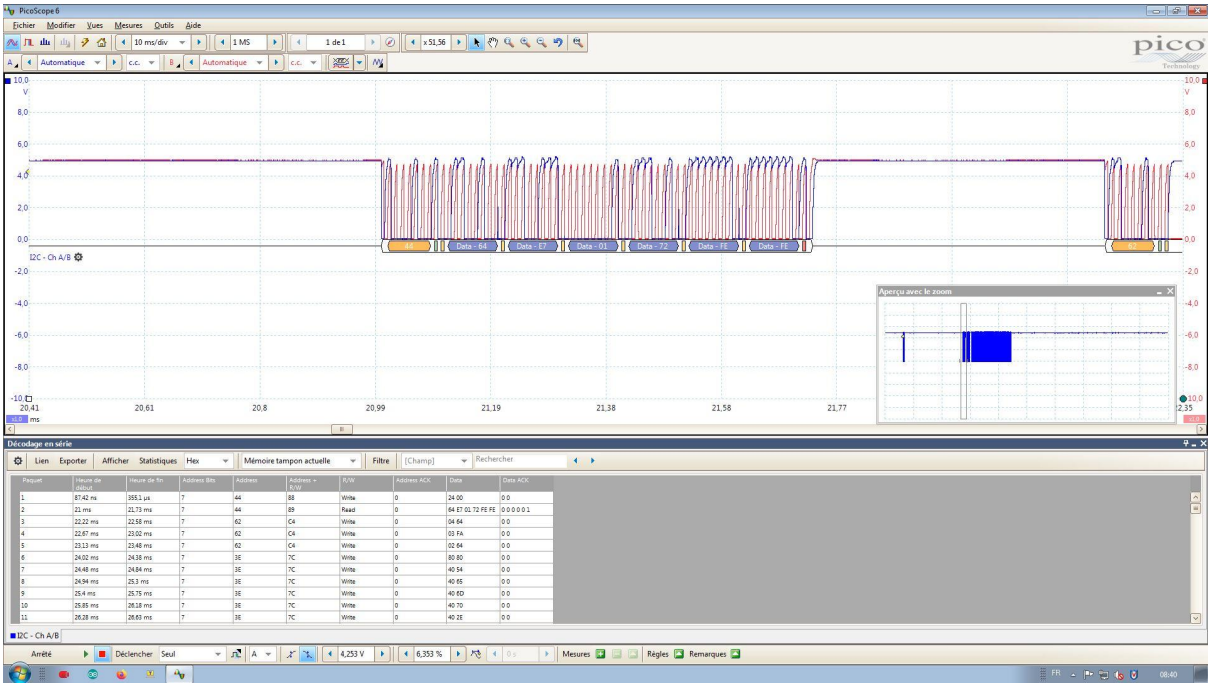
```
//Affichage de la température
lcd_position(&hi2c1,0,0); //Ligne 1
lcd_print(&hi2c1,"Temp. = ");//Texte
lcd_print(&hi2c1,T_unit);//Valeur finale
lcd_write(&hi2c1,0xDF); //Correspond au ° en ASCII étendu
lcd_print(&hi2c1,"C");//Texte

//Affichage de l'humidité
lcd_position(&hi2c1,0,1); //Ligne 2
lcd_print(&hi2c1,"Hum. = ");
lcd_print(&hi2c1,H_unit);
lcd_print(&hi2c1," %");
```

4. Résultats

Je compile et transfère mon code vers la carte. Je branche le picoscope pour observer mon échange de signaux :





J’observe correctement toutes mes adresses, mes registres et mes données et l’affichage sur l’écran s’effectue correctement :