



Département EEA - Faculté Sciences et Ingénierie

Master SME - Année 2021-2022

BE STM32

TPs de base



Rédigé par AIT-BRAHIM Mohamed, KISSI Medhi

Table des matières

1	Introduction	3
2	Le capteur SHT31 (I2C)	4
3	Le capteur DHT22 (One-Wire)	7

1 Introduction

L'objectif de cette première partie du BE était de récupérer la température et le taux d'humidité mesuré par un capteur de température et l'afficher sur un écran LCD, grâce à une carte Nucleo STM32 (STM32-L476RG).

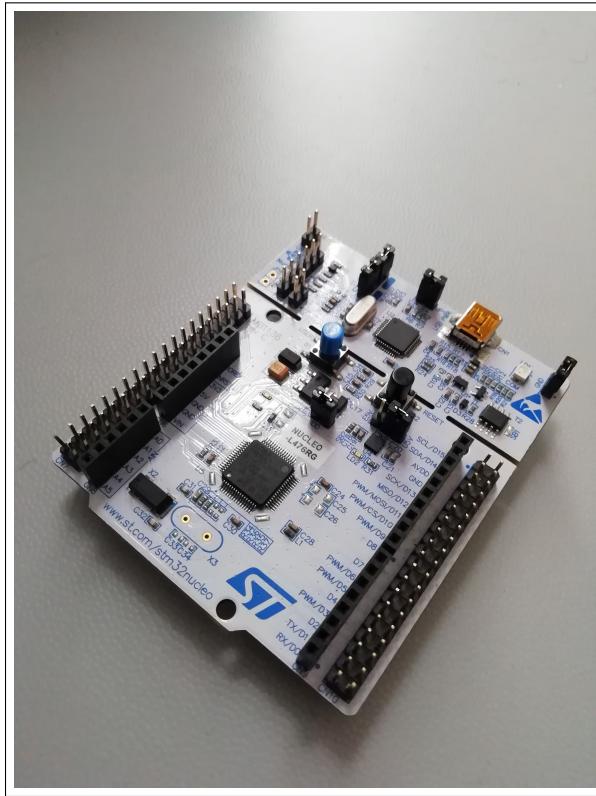


FIGURE 1 – Carte STM32 utilisée

Chacun des membres du binôme possédait une carte ainsi qu'un écran et un capteur différent. L'un des capteurs était un DHT22, qui utilise la communication one-wire, l'autre capteur était un SHT31 qui utilise la communication I2C, tout comme les écrans LCD.

2 Le capteur SHT31 (I2C)

Le capteur SHT31 est constitué de 4 broches : la masse (GND), la broche d'alimentation (5V), une broche SDA et une broche SCL pour faire la communication I2C.

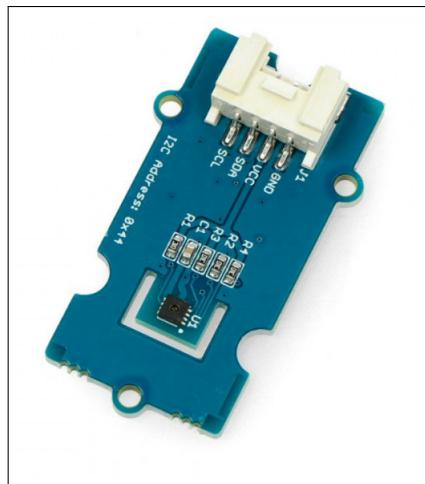


FIGURE 2 – Capteur SHT31

Ainsi il suffit de connecter le capteur sur une broche I2C du microcontrôleur :

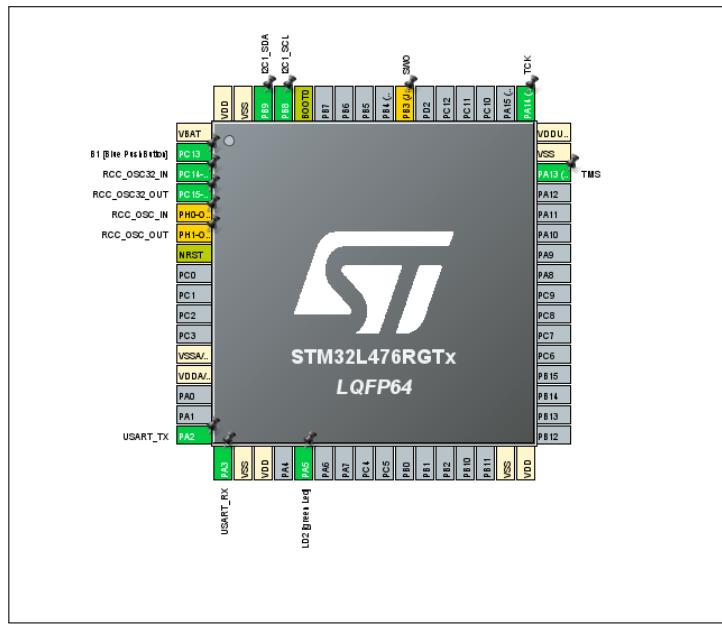


FIGURE 3 – Configuration des broches

Ici, on utilise la broche PB9 pour le SDA et la broche PB8 pour le SCL. On peut voir aussi dans cette partie graphique, qu'on va utiliser l'UART 2, qui permet de communiquer au PC afin de pouvoir afficher les valeurs sur un terminal. Le LCD et le SHT31, seront sur les mêmes broches I2C, en parallèle.

On réalise donc le montage suivant :

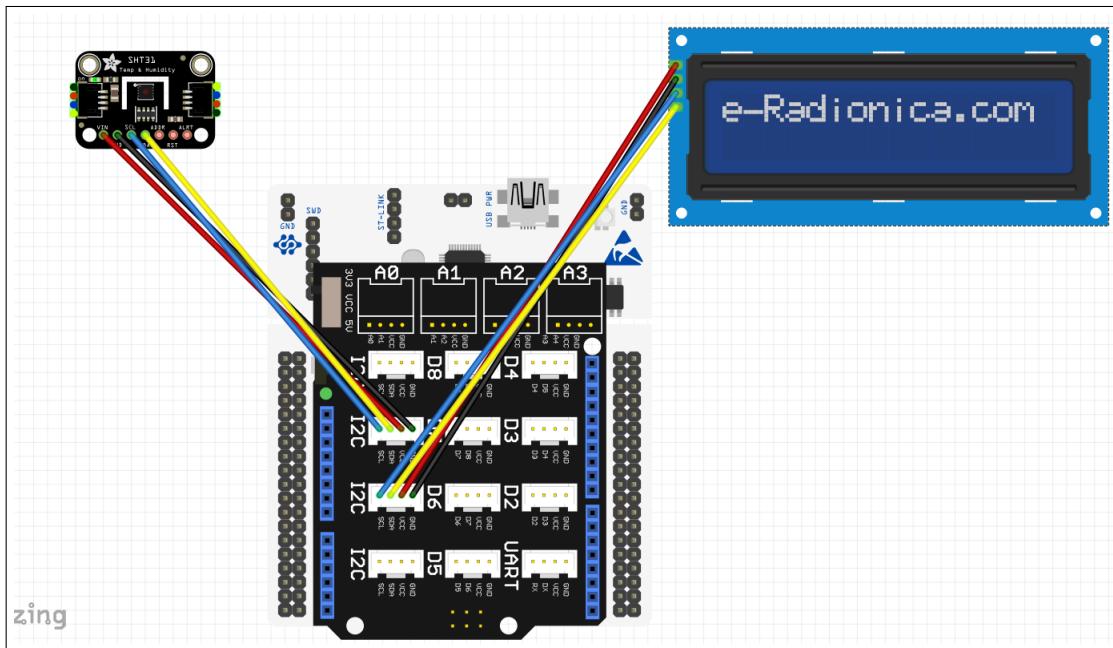


FIGURE 4 – Schéma de câble du SHT31 et du LCD

Il faut donc récupérer les données du SHT31 et l'afficher sur l'écran LCD. Voici le programme main.c :

```

while (1)
{
    /* USER CODE END WHILE */
    char res [10];
    Temp_read(&temp, &humidity);
    ftoat (temp,res,1);
    lcd_position(&hi2c1,12,0);
    lcd_print(&hi2c1,res);

    ftoat (humidity,res,1);
    lcd_position(&hi2c1,10,1);
    lcd_print(&hi2c1,res);
}

```

FIGURE 5 – Programme : main.c

Dans ce programme on peut voir qu'on utilise la fonction Temp_read, qui va nous renvoyer la valeur de la température et de l'humidité. On va ensuite grâce à la fonction ftoat, prendre cette valeur de type float, et la mettre sous forme de chaîne de caractère dans notre tableau "res". Puis nous allons l'afficher dans le LCD, grâce aux fonctions lcd_position et lcd_print.

Nous allons maintenant voir comment on récupère ces données grâce la fonction Temp_read :

Nous avons crée un tableau Data, avec initialement 0x24 et 0x00, ce sont les commandes à envoyer au SHT31, pour qu'il nous envoie la température et l'humidité. Grâce à la fonction TMP_Transmit, on envoie les deux commandes, puis avec la fonction TMP_Receive, on reçoit la réponse du capteur. Suite à cette réponse on traite ces informations avec des calculs mathématiques pour trouver les valeurs de température et d'humidité.

```

void Temp_read( float *temp, float *humidity)
{
    float temperature=0;

    uint8_t Data[6]=(0x24, 0x00);
    TMP_Transmit(ADRESSETEMP, Data, 2);
    HAL_Delay(50);
    TMP_Receive(ADRESSETEMP, Data, 6);

    temperature = Data[0] * 256 + Data[1];
    temperature = -45 + (175 * temperature / 65535.0);
    *temp=-45+175*(Data[0]<>8 | Data[1])/65535.0 ;
    *humidity=-45+175*(Data[3]<>8 | Data[4])/65535.0 ;
}

```

FIGURE 6 – Fonction lecture de données

A l'aide d'un picoscope nous allons visualiser les trames que le capteur envoie au microcontrôleur et que le microcontrôleur envoie au capteur :

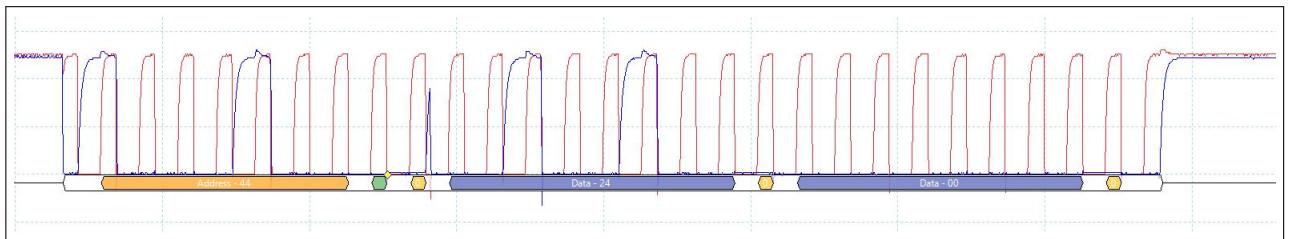


FIGURE 7 – Commande du microcontrôleur vers le SHT31

On peut voir que le premier octet nous avons 0x44 qui correspond à l'adresse de notre capteur I2C. Ensuite nous avons les deux commandes que nous allons envoyer au capteur (0x24, 0x00) qui va nous permettre de récupérer la température et l'humidité.

Nous allons donc voir maintenant ce que le capteur nous répond suite à cette commande :

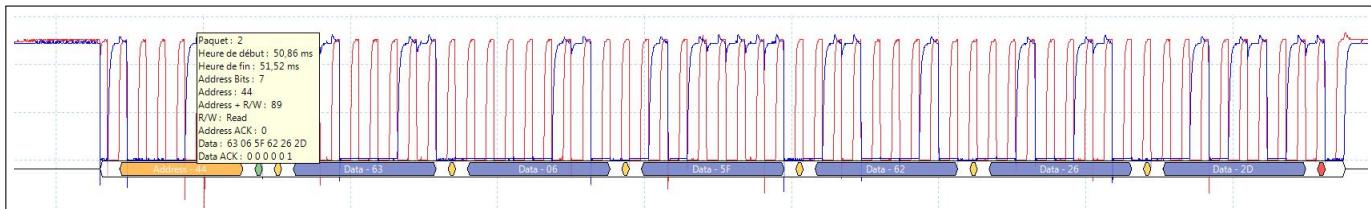


FIGURE 8 – Réponse du SHT31 vers le microcontrôleur

On retrouve dans le premier octet l'adresse de notre capteur (0x44). Nous retrouvons ensuite dans la deuxième (0x63) et troisième (0x06) valeur ce qui va correspondre à la température. Nous allons appliquer la formule afin de valider la véracité de ses valeurs. Le capteur était positionné dans la salle de classe, nous devrions avoir une température dans les 20°C.

$$\text{Température} = -45 + (175 * ((0x63 * 256) + 0x06)) / 65535 = 22.69^\circ\text{C}.$$

Nous avons donc une température qui est bonne. Nous allons ensuite passer au calcul de l'humidité, pour cela nous allons prendre en compte la cinquième (0x62) et la sixième (0x26) valeur.

$$\text{Humidité} = -45 + 175 * (0x62<>8 | 0x26) / 65535 = 22.09 \text{ pourcents}$$

Nous avons donc une humidité à 22.09 pourcents, ce qui est cohérent.

Nous avons donc grâce à ce programme, une manière de recueperer les données qui nous interesse de ce capteur en passant par un bus I2C. Et nous arrivons aussi à afficher cette information sur l'écran LCD, en passant toujours par le même bus I2C.

3 Le capteur DHT22 (One-Wire)

Le capteur DHT22 est constitué de 3 broches : la masse (GND), la broche d'alimentation (5V), ainsi que d'une broche pour communiquer avec le microcontrôleur.



FIGURE 9 – Capteur DHT22

Ainsi il suffit de connecter le capteur sur un des ports analogiques du SHIELD car on utilise la broche en GPIO pour communiquer entre le capteur et le microcontrôleur.

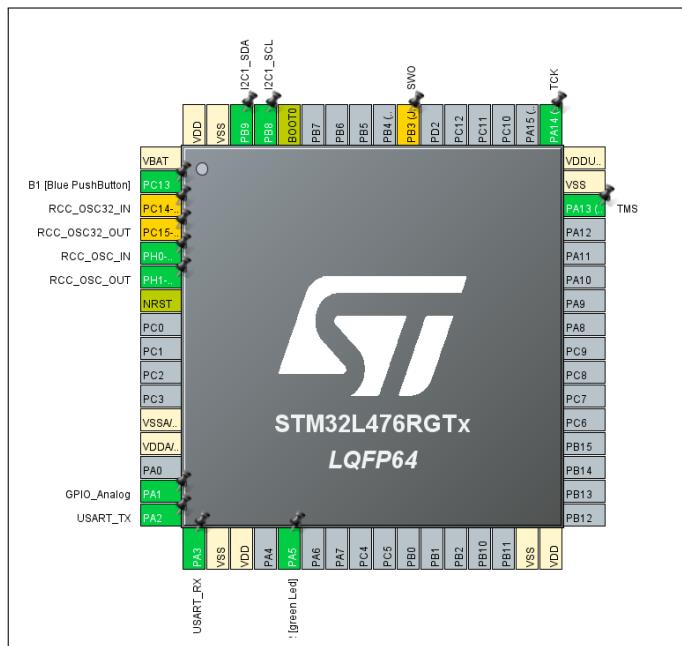


FIGURE 10 – Configuration des broches de la carte sur CubeIDE

On utilise ici la broche PA1 en tant que GPIO analog. On a aussi configuré les broches PB8 et PB9 en I2C (SDA et SCL) pour afficher les données sur l'écran LCD, qui sera donc relié à un port I2C du Shield. On réalise ainsi le montage suivant :

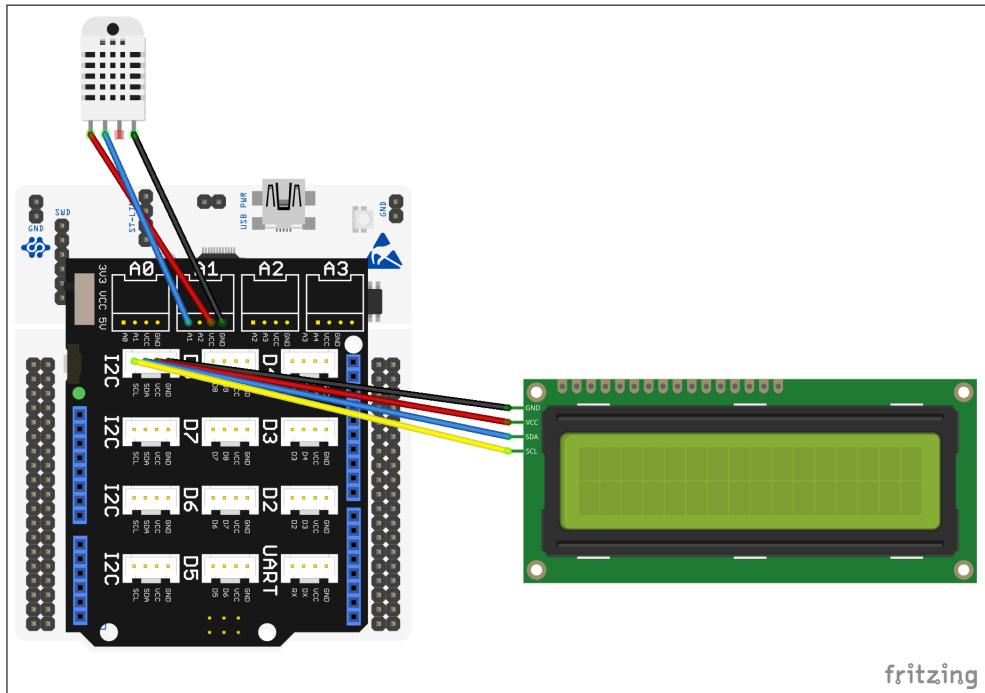


FIGURE 11 – Schéma de câblage du DHT22 sous Fritzing

Pour faire fonctionner notre projet, c'est à dire récupérer les données du capteur DHT22 et les afficher sur le LCD, il reste à rédiger le programme (main.c).

```

/* USER CODE BEGIN PV */

float Temperature = 0, Humidite = 0;
uint16_t RH = 0, TEMP = 0;
uint8_t dataH1;
uint8_t dataH2;
uint8_t dataT1;
uint8_t dataT2;
uint8_t SUM;
uint8_t check;
char bufRH[20]; //pour stocker une valeur et l'afficher sur le LCD
char bufT[20]; //pour stocker une valeur et l'afficher sur le LCD

/* USER CODE END PV */

/* Private function prototypes -----
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

```

```

104  /* Infinite loop */
105  /* USER CODE BEGIN WHILE */
106  while (1)
107  {
108      /* USER CODE END WHILE */
109
110     /* USER CODE BEGIN 3 */
111
112     /*commence la communication avec le capteur*/
113
114     HAL_Delay(3000);
115     Data_Output(GPIOA, GPIO_PIN_1); //info vers le capteur
116     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
117     DWT_Delay_us(1200); //signal de commande
118     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
119     DWT_Delay_us(30); //signal de commande
120     Data_Input(GPIOA, GPIO_PIN_1); //info vers le microcontroleur
121
122     /*commence la reception de donnees*/
123
124     while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1)));
125
126     for (k=0;k<1000;k++)
127     {
128         if (HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_1) == GPIO_PIN_RESET)
129         {
130             break;
131         }
132     }
133
134     while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1)));
135     DWT_Delay_us(40);
136
137     Read_data(&dataH1); //dans la library HT.c
138     Read_data(&dataH2);
139     Read_data(&dataT1);
140     Read_data(&dataT2);
141     Read_data(&SUM);
142
143     check = dataH1 + dataH2 + dataT1 + dataT2; //pour verifier la lecture dans le IDE
144
145     RH = (dataH1<<8) | dataH2;
146     TEMP = (dataT1<<8) | dataT2;
147
148     Humidite = RH / 10.0;
149     Temperature = TEMP / 10.0;
150
151     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET); //pour la prochaine lecture
152
153     /*commence transmission vers LCD*/
154     clearlcd();
155     sprintf(bufRH,"Humidite: %.1f", Humidite);
156     sprintf(bufT, "Temp.: %.1f C", Temperature);
157     lcd_position(&hi2c1,0,0);
158     lcd_print(&hi2c1,bufRH);
159     lcd_print(&hi2c1,"%");
160     lcd_position(&hi2c1,0,1);
161     lcd_print(&hi2c1,bufT);
162     reglagecouleur(0,0,255);
163 }
164 /* USER CODE END 3 */
165 }
```

FIGURE 12 – Programme main.c du projet

A l'aide d'un picoscope, nous avons pu lire la trame que le capteur envoie au microcontrôleur, qui est la suivante :

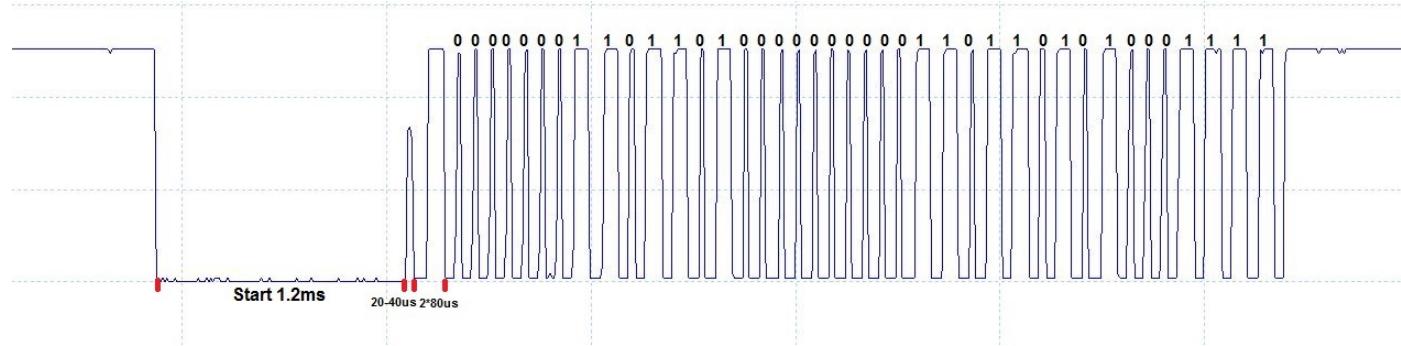


FIGURE 13 – Trame du DHT22

Sur cette trame les 2 premiers bits correspondent au signal de départ et au signal de réponse, on peut ensuite lire sur les 2 octets suivant l'humidité, puis sur les 2 qui viennent après la température, le dernier octet correspond aux bits de parités.

Ainsi sur cette trame on peut lire que l'humidité mesurée est 00000001 10110100 ce qui nous donne 436 donc 43.6% d'humidité.

Pour ce qui est de la température les 2 octets lu sont 00000000 11011010 ce qui nous 218 donc 21.8 degré.

Ainsi, on a pu vérifier que notre projet fonctionnait correctement, et les bonnes informations sont affiché sur le LCD.

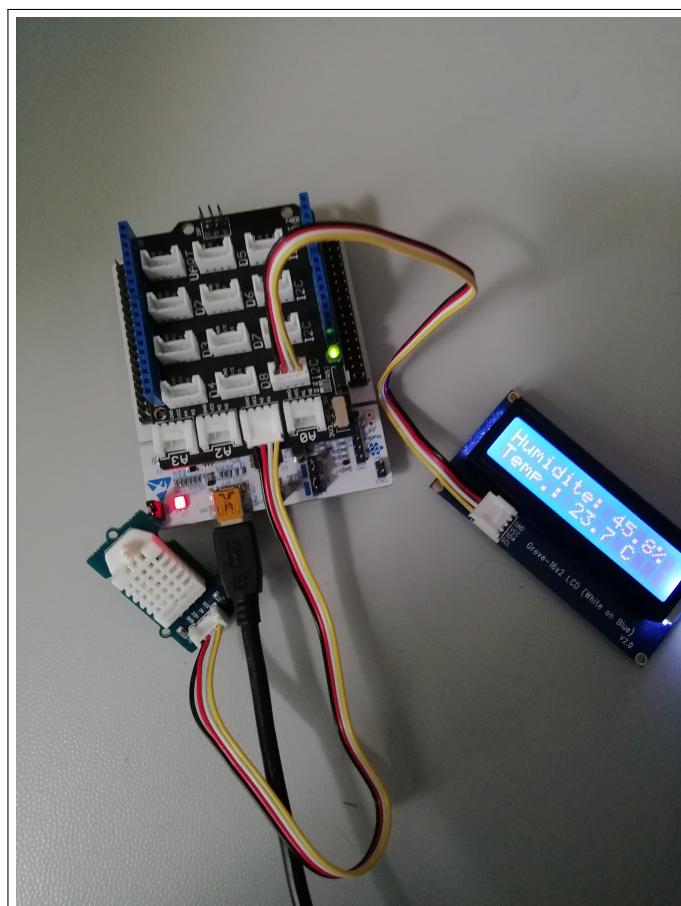


FIGURE 14 – Photo du montage final des TP de base pour le DHT22