BUREAU D'ETUDE :

STATION METEO

Parcours : M1 SME

Compte Rendu

TP de Base

Binôme :

- Anis Mourad BENNACER
- Hichem Ibrahim Salah

Responsable TP :

* Thierry PERISSE

2022/2023

Introduction :

Durant le 2ème semestre nous avons été amenés à réaliser un projet dans le cadre d'un bureau d'étude, ce projet a pour but d'améliorer notre façon d'appréhender une problématique liée à notre domaine d'étude et trouver des solutions grâce à diverse documentations tout en travaillant de manière autonome. Pour ce bureau d'étude, on doit réaliser une station météo à l'aide de la carte STM32L152RE, deux capteurs (Dht22 & BMP280) et deux modules de communication Xbee pro. La réalisation de notre projet va se deviser en deux parties.

La première partie comporte les TP de base qui ont pour but de familiariser avec la carte, nos deux capteurs ainsi que actionneurs dont on dispose.

La deuxième partie est la partie réalisation, ou on doit assembler les différents composants et créer un système qui doit remplir une certaine tache.

Matériels utilisés :

Carte STM32L152RE :

Le STM32L152RE de la famille STM32 de STMicroelectronics. Il fait partie de la série STM32L1 est un microcontrôleur basse consommation avec un processeur ARM Cortex-M3, qui est spécifiquement conçue pour les applications nécessitant une faible consommation d'énergie. Il offre diverses interfaces de communication, notamment l'I2C, l'UART, le SPI et le CAN. Ces interfaces permettent de communiquer avec d'autres périphériques et de connecter le microcontrôleur à des capteurs, des actionneurs ou d'autres systèmes.

Il propose différents modes de gestion de l'énergie, tels que les modes veille, stop, standby et shutdown, permettant d'optimiser la consommation d'énergie en fonction des besoins de l'application, ce qu'il le rend idéal pour les applications où la gestion de l'énergie est essentielle.

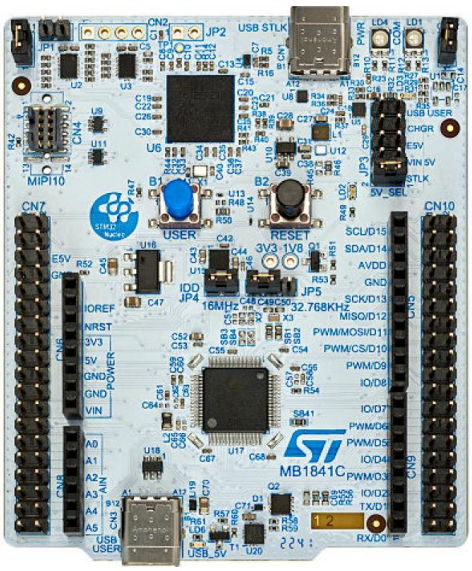


Figure 1 : STM32 NUCLEO-L152RE

Capteur BMP280 :

Le capteur BMP280 est un capteur de pression atmosphérique et de température précis et compact. Il est largement utilisé dans diverses applications telles que la météorologie. Il mesure la pression atmosphérique et la température ambiante avec une grande précision.
Le capteur BMP280 communique via une interface de bus série, généralement I2C ou SPI, ce qui le rend compatible avec une large gamme de microcontrôleurs et de plateformes de développement. Il dispose également d'une faible consommation d'énergie, ce qui en fait un choix populaire pour les applications alimentées par batterie.

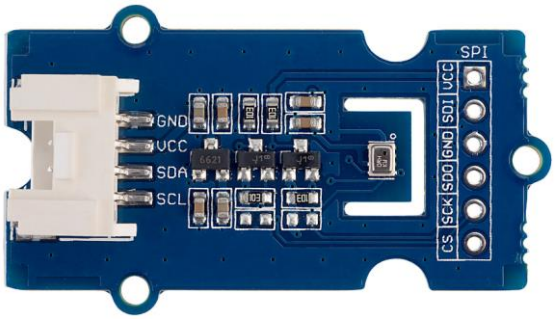


Figure 2: Capteur BMP280

Capteur DHT22 :

Le capteur DHT22 est un capteur d'humidité et de température numérique utilisé dans de nombreuses applications. Il est souvent utilisé dans les dispositifs météorologiques.
Il est doté d'un élément de capteur à semi-conducteur qui mesure à la fois l'humidité relative et la température ambiante. Il est conçu pour offrir une grande précision et une plage de mesure étendue. Le capteur DHT22 communique avec d'autres appareils via un signal numérique. Il utilise une interface à un seul fil (One-Wire) pour transmettre les données de température et d'humidité. Cela facilite son intégration avec une variété de microcontrôleurs et de plateformes de développement.

L'un des avantages de ce capteur est sa fiabilité et sa stabilité à long terme. Il est conçu pour compenser automatiquement les variations de température et d'humidité, ce qui permet de maintenir des mesures précises et cohérentes sur une période prolongée.
Il est généralement alimenté par une tension de 3,3 V et consomme très peu d'énergie, ce qui en fait un choix idéal pour les projets à faible consommation d'énergie.

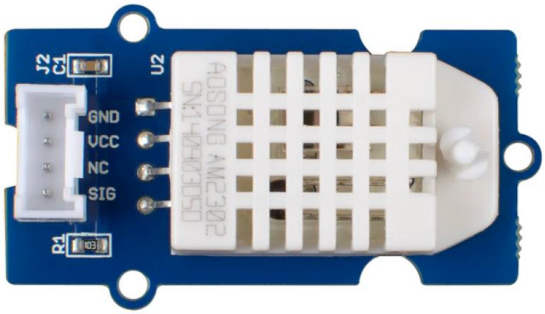


Figure 3 :Capteur de température et d'humidité DHT22

Afficher LCD I2C :

L'écran LCD I2C se compose d'un écran à cristaux liquides et d'un module d'interface I2C intégré. Le module d'interface I2C permet de contrôler l'affichage et de transférer les données via le bus I2C, simplifiant ainsi la connexion avec le microcontrôleur.

Au lieu d'utiliser plusieurs broches pour le contrôle et la transmission des données, l'écran LCD I2C utilise seulement deux broches : SDA (Serial Data Line) et SCL (Serial Clock Line), qui sont utilisées pour la communication I2C.

Figure 4: 16x2 LCD I2C (White on blue)

Base Shield :

Le module Base Shield est une carte d'extension conçue pour faciliter le prototypage et l'utilisation de modules et de capteurs avec des microcontrôleurs ou des cartes de développement. Il est généralement associé à des plates-formes telles que STM32, permettant une connexion facile et rapide de différents composants électroniques.

Le module Base Shield fournit une interface standardisée et un agencement de broches préconfiguré qui simplifie grandement le câblage et l'interconnexion des composants, il peut également comporter des connecteurs pour des broches d'alimentation (5V, 3.3V, GND) permettant d'alimenter les différents composants connectés.

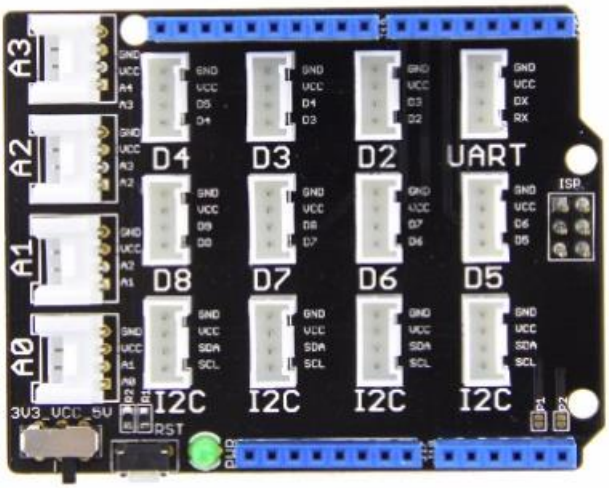


Figure 5: Base Shield V2

Les TP de Base

1. BMP280 :

Ce TP de base consiste à mesurer la température et l'humidité à l'aide du capteur BMP280 dans n'importe quel milieu et afficher les valeurs mesurées sur un afficheur LCD I2C.

1.1 Schéma de câblage :

À l'aide du logiciel Fritzing nous avons pu réaliser le câblage de notre premier TP de base comme il est indiqué sur la figure 6 et 7 ci-dessous :

BMP280 :

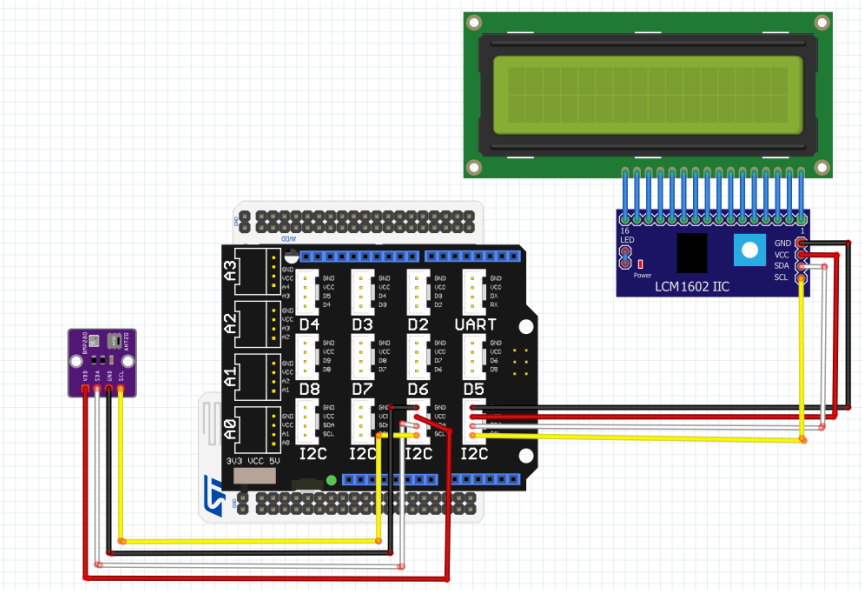


Figure 6 : Schéma de câblage avec Fritzing

DHT22 :

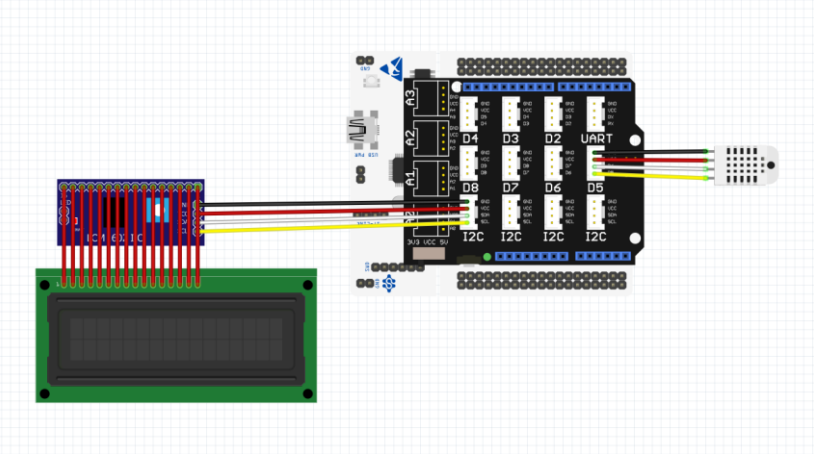


Figure 7 :Schéma de câblage avec Fritzing

1.2 Configuration sur STM32CubeIDE :

BMP280 :

A présent nous allons créer notre projet BMP280 sur le logiciel STM32CubeIDE, après nous allons configurer les bronches nécessaires à savoir PB8 et PB9 en I2C à l'aide de STM32CubeMX afin de générer le code en C.

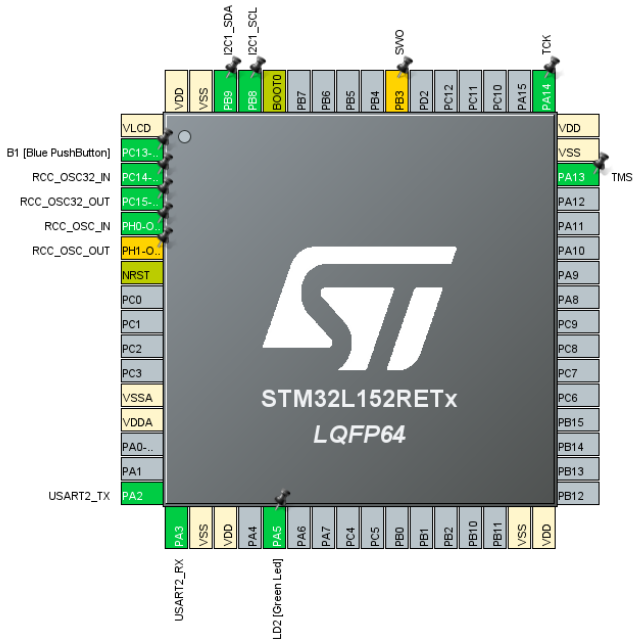


Figure 8 : Configuration des broches (BMP280)

DHT22 :

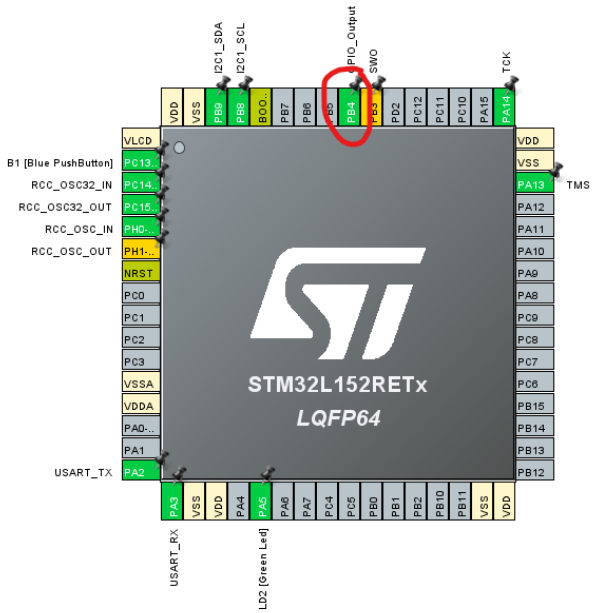


Figure 9 : Configuration des broches (dht22)

On configure la broche PB4 (one wire) qui sert de communication entre le microcontrôleur et le capteur Dht22.

1.3 Programmation

BMP280 :

Dans cette partie nous allons utiliser une bibliothèque qui est prête et nous allons apporter quelques modifications au niveau des lignes de codes :

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 :

```
/* USER CODE BEGIN 2 */
bmp280_init_default_params(&bmp280.params);
bmp280.addr = BMP280_I2C_ADDRESS_1;
bmp280.i2c = &hi2c1;
```

Pour lire la valeur de la température et la pression :

```
   /* USER CODE END WHILE */
while (!bmp280_read_float(&bmp280, &temperature, &pressure,)) {
   size = sprintf((char *)Data,"Temperature/pressure reading failed\n\r");
        HAL_UART_Transmit(&huart1, Data, size, 1000);
        HAL_Delay(2000);
                  }
            pressure=pressure/100.;
```

Affichage sur LCD I2C :

```
        lcd_position(&hi2c1,0,0);
        lcd_print(&hi2c1,"Temp:  ");
        lcd_position(&hi2c1,6,0);
        lcd_print(&hi2c1,Data);
        lcd_position(&hi2c1,0,1);
        lcd_print(&hi2c1,"Pres: ");
        lcd_position(&hi2c1,6,1);
        lcd_print(&hi2c1,Data);
```

DHT22 :

La fonction Read_data permet de lire les donnes du capteur DHT22.

```
void Read_data (uint8_t *data)
  {
    int i, k;
    for (i=0;i<8;i++)
    {
        if (HAL_GPIO_ReadPin (GPIOB, GPIO_PIN_4) == GPIO_PIN_RESET)
        {
            (*data)&= ~(1<<(7-i)); //data bit is 0
            while(!(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_4)));
            DWT_Delay_us(40);
        }
        else                        //data bit is 1
        {
            (*data)|= (1<<(7-i));
            for (k=0;k<1000;k++)
            {
                if (HAL_GPIO_ReadPin (GPIOB, GPIO_PIN_4) == GPIO_PIN_RESET)
                  {
                    break;
                  }
            }
            while(!(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_4)));
            DWT_Delay_us(40);
        }
     }
  }
```

Programme principale

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

        /* USER CODE BEGIN 3 */

        /*commence la communication avec le capteur*/

    HAL_Delay(1000);// Effectuer des mesures chaque seconde
        Set_Pin_Output(GPIOB, GPIO_PIN_4); //Configuration du pin en sortie
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET); //Mise à 0 du pin
        DWT_Delay_us(1000); //Envoi de la premiere partie du signal de commande
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET); // Mise a 1 du pin
        DWT_Delay_us(30); //Envoi de la deuxieme partie du signal de commande
        Set_Pin_Input(GPIOB, GPIO_PIN_4); //Configuration du pin en entrée

        // Lecture des données

        while(!(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_4)));

        for (k=0;k<1000;k++)
        {
            if (HAL_GPIO_ReadPin (GPIOB, GPIO_PIN_4) == GPIO_PIN_RESET)
            {
              break;
            }
        }

        while(!(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_4)));
        DWT_Delay_us(40);

        Read_data(&dataH1); //Lecture du premier octet d'humidité
        Read_data(&dataH2); //Lecture du deuxieme octet d'humidité
        Read_data(&dataT1); //Lecture du premier octet de temperature
        Read_data(&dataT2); //Lecture du deuxieme octet de temperature
        Read_data(&SUM);  //Lecture d'octet de sécurité

        check = dataH1 + dataH2 + dataT1 + dataT2;
```

```
RH = (dataH1<<8) | dataH2;  //Concatenation des octets d'humidité
TEMP = (dataT1<<8) | dataT2; //Concatenation des octets de temperature

Humidite = (int) RH / 10.0; //Les valeurs recues sont egales à fois les valeurs reelles
Temperature = (int) TEMP / 10.0;

HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET); //Preparation pour lecture suivante
```

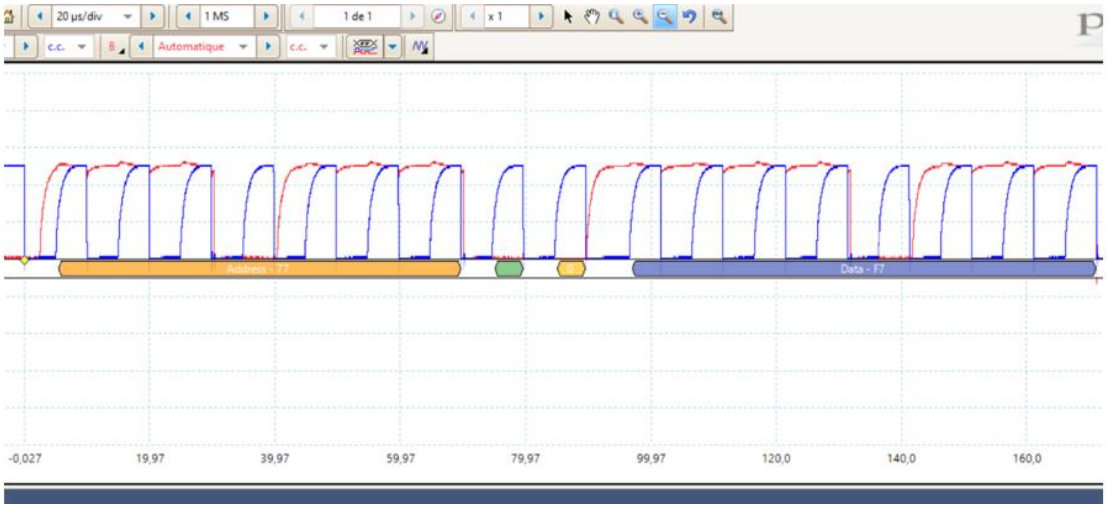
Affichage LCD :

```
  size=sprintf((char *)Data,"HUMIDITE : %.1f", Humidite);

  lcd_position(&hi2c1,0,0);
  lcd_print(&hi2c1,(char*)Data);
  lcd_print(&hi2c1,"%");
  size=  sprintf((char *)Data, "TEMP :    %.1f C", Temperature);

  lcd_position(&hi2c1,0,1);
  lcd_print(&hi2c1,(char*)Data);
}
```

1.4 Analyse des trames :



Dans cette partie nous allons voir les données sur la trame I2C en utilisant un picoscope. L'objectif est de comparer ces résultats aux informations fournies par le fabricant sur la datasheet.

Au niveau de la trame, on remarque que le chronogramme bleu correspond à l'horloge et celui en rouge aux données du capteur.

On voit bien que l'adresse du capteur affichée dans la datasheet correspond bien à l'adresse du capteur sur la trame qui est de 0X77.

Réalisation et manipulation :

Après réaliser le câblage et modifier le code, nous avons téléverser ce dernier vers la carte pour tester notre projet et voir les mesures obtenues illustrés sur la figure ci-dessous :

BMP280 :

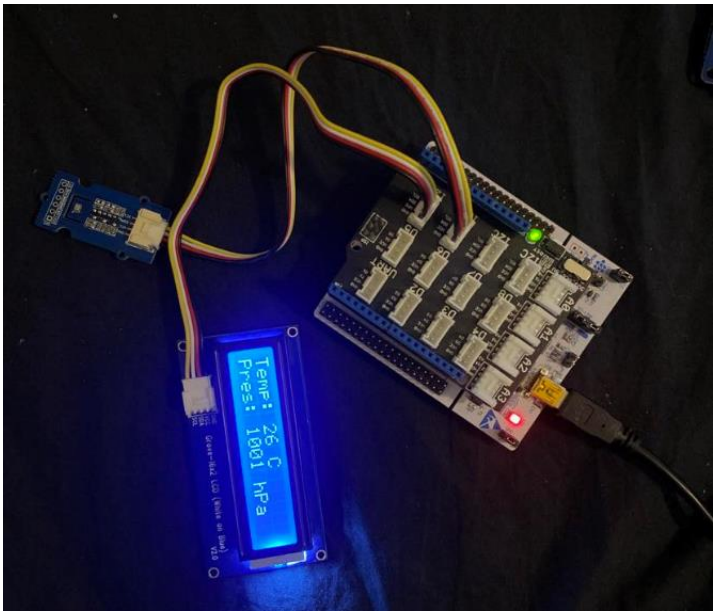


Figure 10: Test et affichage des mesures du capteurBMP280

DHT22 :

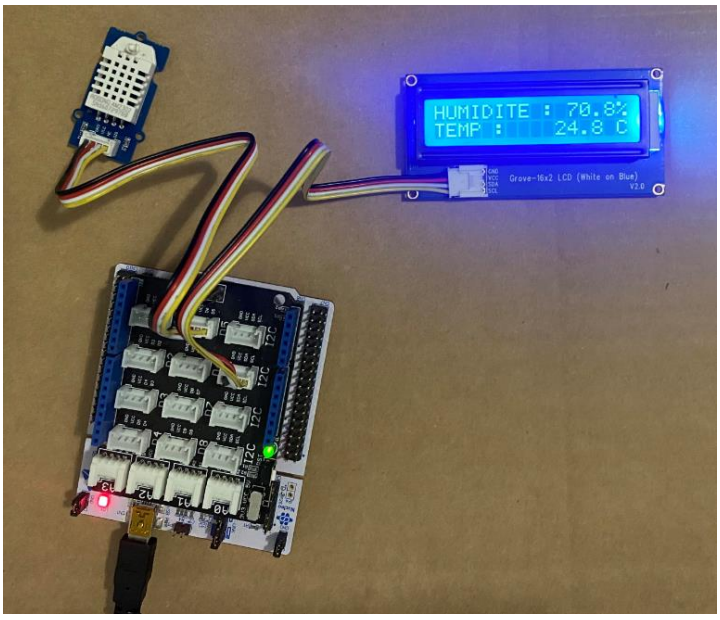


Figure 11: Test et affichage des mesures du capteur DHT22

Conclusion :

Dans cette première partie nous avons vu comment configurer les broches sur la carte STM32, nous avons pu lire les donnes des capteurs et afficher les mesures sur un écran LCD, cette partie nous a permis de nous familiariser avec les cartes Nucleo STM32 ainsi que le logiciel CubeIDE, et d'apprendre plus sur les protocoles de communication.

Dans la partie 2 du BE nous allons voir comment communiquer entre les deux cartes grâce au protocole de communication Xbee, en utilisant les module Xbee pro et en effectuant u affichage croisé.