Département EEA – Faculté Sciences et Ingénierie

Master 1 SME

Projet BE : Abaque de température/humidité

Année 2022-2023

Rapport de projet

Rédigé par HAMDAN Feras et ZAIER Aymen

Réalisation de systèmes : BE-PROJET

Introduction

Dans le cadre de notre bureau d'étude, nous avons été amenés à mettre en place un système de détection de température humidité en fonction de la distance qui nous permettra par la suite d'établir des abaques d'évaluation du comportement de chaque caractéristique en fonction de la distance du point à étudier.
L'objectif de ce bureau d'étude est la mise en marche des notions des cours que nous avons eu précédemment sur les capteurs et les microcontrôleurs et la confrontation à une situation d'ingénieur dans l'étude, analyse et conception d'un projet à implémenter.

Dans ce rapport, vous trouverez, dans une première partie, l'étude technique des composants que nous avons utilisés pour réaliser notre projet.
Ensuite, nous allons se focaliser sur le type de communication sur lequel nous avons basé notre transmission de donnée.
Enfin, une partie pour évoquer le scénario de mise en situation de notre système.

Capteur de distance (HC-sr04)

Étude doc constructeur

En lisant la datasheet du capteur HC-sr04, les informations importantes à retenir sont :

- Il faut envoyer un pulse de 10 us sur le trigger
- On reçoit un signal proportionnel à la distance sur l'echo
- La formule pour calculer la distance est donné : Distance (cm) = temps (us)/58

Nous commencerons donc par créer le pulse que recevra le trigger

Création du pulse

Pour créer un pulse de 10 us, nous devons créer une fonction qui nous permettras d'avoir un delay en us, ce qui n'est pas possible avec la fonction fournis par l'IDE car celle-ci permet d'avoir un delay en ms.
Nous commençons par ajouter le timer 2 et initialiser son pre-scaler à 31 ce qui divise la fréquence du timer par 32, or ayant une fréquence initiale de 32 Mhz, nous nous retrouvons avec une fréquence de 1 Mhz soit une incrémentation du timer chaque 1 us.
La fonction delay_us est la suivante :

```
void delay_us(uint16_t us)
{
    __HAL_TIM_SET_COUNTER(&htim2,0);
    while(__HAL_TIM_GET_COUNTER(&htim2)<us);
}
```

Figure 1: fonction delay_us

__HAL_TIM_SET_COUNTER(&htim2,0) permet d'initialiser le timer 2 à 0.
__HAL_TIM_GET_COUNTER(&htim2) permet d'obtenir la valeur du timer 2.
La fonction commence donc par initialiser le timer à 0, puis elle attend que le timer atteigne la valeur désirée, à ce moment on sort de la fonction et le temps désiré c'est écoulé.

Nous devons créer à présent une fonction qui crée le pulse :

```
void distance()
{
	HAL_GPIO_WritePin(GPIOA,GPIO_PIN_0,GPIO_PIN_SET);
	delay_us(10);
	HAL_GPIO_WritePin(GPIOA,GPIO_PIN_0,GPIO_PIN_RESET);
	HAL_Delay(1000);
}
```

Figure 2: fonction distance

Nous commençons par mettre le pin 6 à 3,3V et 10 us plus tard nous le mettons à 0 grâce à la fonction delay_us crée précédemment. Nous avons maintenant un pulse de 10 us que nous envoyons sur le trigger. On rajoute un delay de 1 s pour assurer que le capteur reçoit l'entièreté de l'echo avant que le microcontrôleur lui envoie un autre pulse.
Pour valider le fonctionnement du capteur, on regarde sur le pico scope les signaux du trigger et de l'echo :

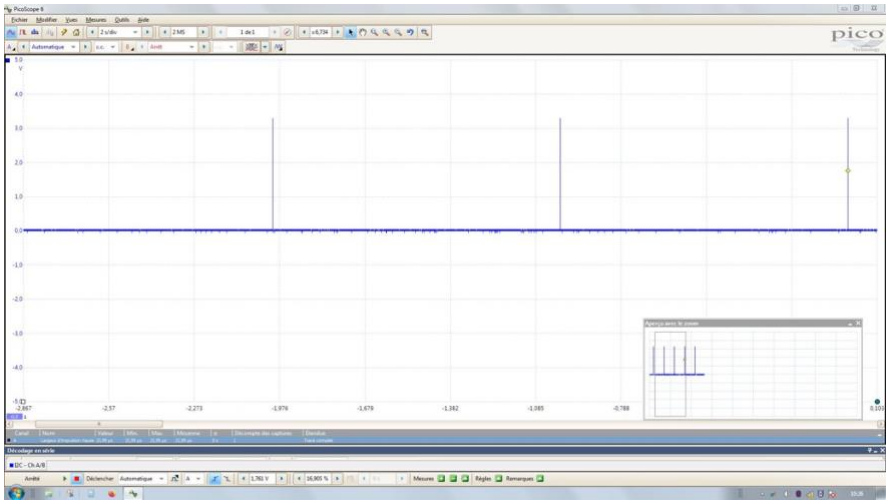


Figure 3: Signal Trigger

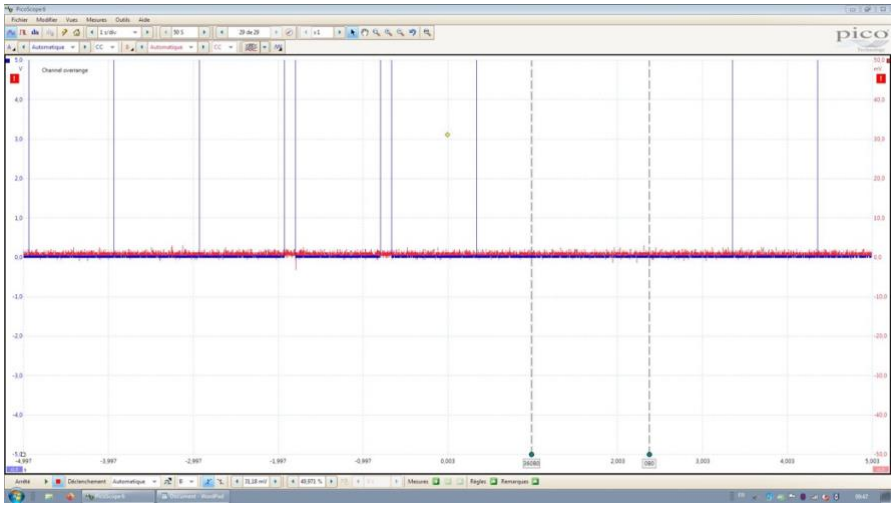


Figure 4: Signal Echo

On remarque sur la figure 3 que nous avons bien des pulses de 10 us à 1 seconde d'intervalle. On remarque sur la figure 4 que nous bien le signal désiré sur l'echo.

Nous décidons que le traitement du signal echo se feras par une fonction d'interruption puisque nous ne pouvons pas prédire l'instant précis où nous recevrons ce signal.

On utilise le timer 2 pour obtenir la durée de l'echo en microsecondes, on calcule la distance grâce à la formule donnée par la datasheet du composant. Lorsque la distance n'est pas dans l'intervalle de mesure, le capteur renvoie un signal d'une durée largement supérieure à l'équivalent de 5 m. On test donc si la distance obtenue est supérieure à 500 cm, dans ce cas on affiche erreur sur l'écran lcd. Dans le cas contraire on affiche la distance.

Affichage du résultat

Pour afficher le résultat nous utiliserons un écran LCD (sa programmation est décrite dans le rapport du Tp initiale). Nous allons créer une fonction pour simplifier le code.

```
void lcd(int a,int i,char *s)
{

     lcd_position(&hi2c1,a,i);
     lcd_print(&hi2c1,s);
}
```

Figure 5: fonction lcd

Cette fonction rend le code plus lisible et sera utilisé pour tous les affichages qui suivent.

Schéma électrique

Grace à la datasheet du capteur, nous savons qu'il doit être alimenter par une tension de 5 volts. Nous avons initialisé le Pin 0 en GPIO_Output et le Pin 1 en GPIO_EXTI1. Nous avons programmé le signal du trigger sur le Pin 0 et l'interruption sur le Pin 1. On connecte donc Vcc au Pin 5v, le GND au GND, l'echo au Pin 1 et le trigger au Pin 0.

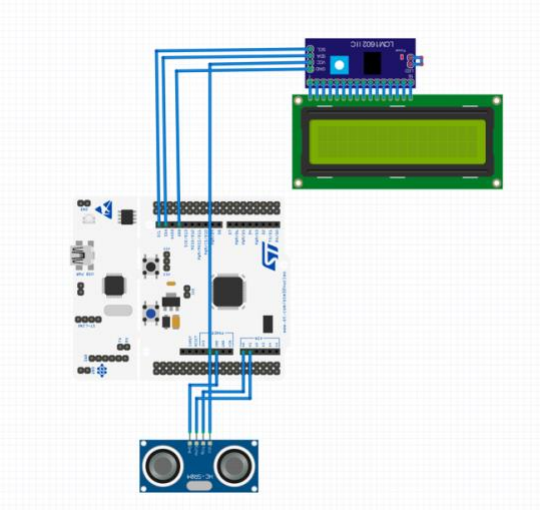


Figure 6: Schéma du montage de mesure de distance

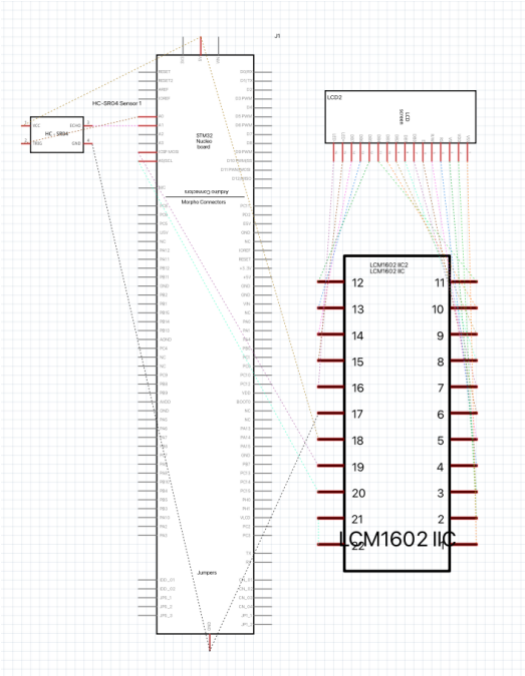


Figure 7: schéma électrique équivalent

Capteur de température et d'humidité (DHT22)

La communication du capteur DHT22 avec l'environnement de travail qui dans notre cas la carte nucléo est basée principalement sur une relation maitre esclave.
L'idée est d'émettre un signal du microcontrôleur avec des caractéristique spécifique à ce capteur et recevoir une réponse du capteur qui envoie à son tour au microcontrôleur toujours avec un signal spécifique.
Pour détailler un peu ce principe de fonctionnement, en s'appuyant sur la datasheet du capteur DHT22 et en raisonnant par rapport à notre mise en marche de ce capteur,
Tout d'abord nous avons créé une fonction sur le code du microcontrôleur qui permet de

définir un PIN sous forme d'entrée et sortie étant donné que notre capteur ait une communication bidirectionnelle.

```
void temperature_humidite()
{

      HAL_Delay(500);
      Data_Output(GPIOA, GPIO_PIN_8);
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
      DWT_Delay_us(1200);
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
      DWT_Delay_us(30);
      Data_Input(GPIOA, GPIO_PIN_8);


      while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8)));

      for (int k=0;k<1000;k++)
      {
          if (HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_8) == GPIO_PIN_RESET)
          {
            break;
          }
      }

      while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8)));
      DWT_Delay_us(40);


      Read_data(&dataH1);
      Read_data(&dataH2);
      Read_data(&dataT1);
      Read_data(&dataT2);
      Read_data(&SUM);

      check = dataH1 + dataH2 + dataT1 + dataT2;

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

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

      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);


      sprintf(bufRH,"H %.1f", Humidite);
      lcd(0,1,bufRH);
      sprintf(bufT, "T %.1f", Temperature);
      lcd(7,1,bufT);

}
```

Figure 8: fonction mesurant la température et l'humidité

Ensuite nous avons généré un signal maitre répondant aux caractéristiques fournit sur la datasheet.
En mettant le Pin 8 en Ouput, On envoie 0 pendant 1,2 ms puis 1 pendant 30 us.
En mettant le Pin 8 en Input, on attend que le signal passe à 1 puis de nouveau à 0 (signal de départ donné par la datasheet). On lit la valeur des données avec la fonction Read_data ;

```
void Read_data (uint8_t *data)
  {
    int i, k;
    for (i=0;i<8;i++)
    {
        if (HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_8) == GPIO_PIN_RESET)
        {
            (*data)&= ~(1<<(7-i)); //data bit is 0
            while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8)));
            DWT_Delay_us(40);
        }
        else                        //data bit is 1
        {
            (*data)|= (1<<(7-i));
            for (k=0;k<1000;k++)
            {
                if (HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_8) == GPIO_PIN_RESET)
                  {
                    break;
                  }
            }
            while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8)));
            DWT_Delay_us(40);
        }
    }
  }
```

Figure 9: fonction Read_data

Read_data lit la valeur d'un octet bit par bit, on lit la valeur du Pin 8, on le met dans le pointeur *data et on attend le prochain bit.

```c
void Data_Output (GPIO_TypeDef *PORT, uint16_t PIN) //direction vers le capteur
  {
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    GPIO_InitStruct.Pin = PIN;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP; //pushpull
    HAL_GPIO_Init(PORT, &GPIO_InitStruct);
  }
```

Figure 10: fonction Data_Output

```c
void Data_Input (GPIO_TypeDef *PORT, uint16_t PIN) //direction vers le microcontroleur
  {
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    GPIO_InitStruct.Pin = PIN;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    HAL_GPIO_Init(PORT, &GPIO_InitStruct);
  }
```

Figure 11: fonction Data_Input

La fonction Data_Output permet de configurer un Pin en Output et la fonction Data_Input permet de le configurer en Input.

Les fonction Read_data, Data_Output et Data_Input se trouve dans la librairie PortINOUT.h

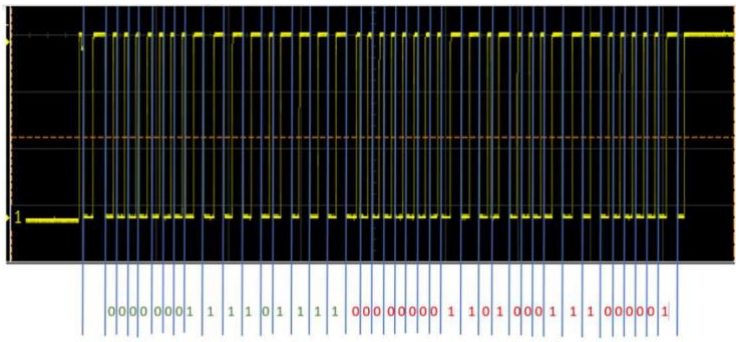
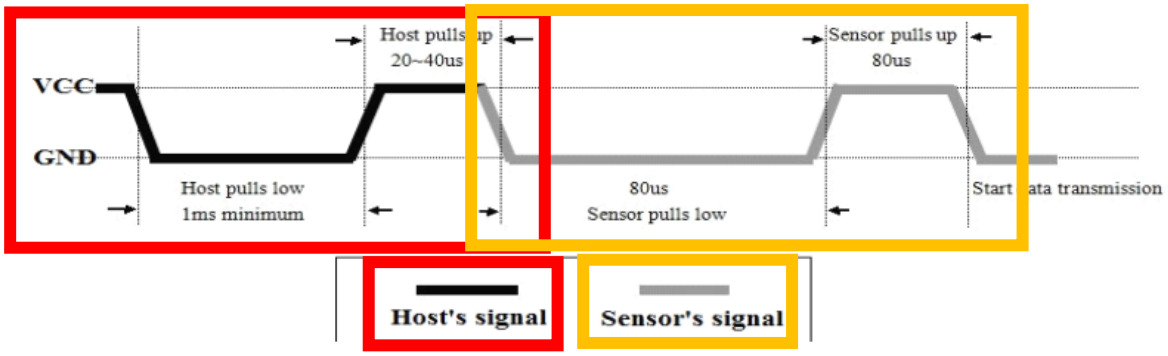


Figure 12:vérification sur l'oscilloscope du capteur DHT22

Sur la figure 12, on observe sur la trame :

- Le signal de départ
- 2 octet correspondant à la valeur d'humidité
- 2 octet correspondant à la valeur de température
- 1 octet de vérification



Afin de mieux respecter les contraintes temporelles inclus dans la génération du signal maitre, nous avons utilisé la fonction delay_us présenté précédemment.

Calcule et affichage du temps

On souhaite afficher également le temps(relatif au commencement des mesures). Pour cela nous allons utiliser le timer 3. De même que pour le timer 2 nous allons modifier la fréquence associée, cependant cette fois nous allons initialiser son pre-scaler à 31999 pour diviser la fréquence par 32000, elle passe alors de 32 Mhz à 1 Khz. Le timer s'incrémente alors chaque milliseconde.

```
while(__HAL_TIM_GET_COUNTER(&htim3)<1000*max_time);

time = time + __HAL_TIM_GET_COUNTER(&htim3)/1000;
sprintf(buftim, "t %.1f", time);
lcd(7,0,buftim);
__HAL_TIM_SET_COUNTER(&htim3,0);
```

Figure 13: section du programme permettant calculer le temps

Le code ci-dessus correspond au calcul et à l'affichage du temps et se trouve dans la boucle while(1) du programme. __HAL_TIM_SET_COUNTER(&htim3,0) permet de réinitialiser le timer et est déjà appeler avant la boucle while. max_time est le temps en secondes que l'on souhaite avoir entre chaque mesure.
Ici on attend que le timer arrive à la valeur souhaitée, puis on additionne le temps en seconde à la variable time. On transforme la variable time en string, on l'affiche et on remet la valeur du timer à 0. Les mesures sont effectuées directement à la suite.

Cette section de code permet également d'afficher les données pour la durée max_time, de façon à nous donner le temps de lire où relever les valeurs (de la même façon qu'un delay). Si nous souhaitons ne pas avoir l'effet delay on peut simplement supprimer la ligne de code while(__HAL_TIM_COUNTER(&htim3)<1000*max_time ;
Il est également important de noter qu'un max_time trop élevé retardera la réception des données par Bluetooth.

Bluetooth HC-05 (Arduino)

Commande AT sur Arduino Due

Pour connecter les deux STM32 par Bluetooth, nous utiliserons deux HC-05 (il est également possible d'utilisé un HC-05 et un HC-06). Il faut commencer par les configurer grâce à des commandes AT (il s'agit d'un protocole de communication). Les commandes qui nous intéressent sont :

- AT pour vérifier la connexion entre l'Arduino et HC-05
- AT + ADDR ? pour obtenir l'adresse de l'esclave
- AT + ROLE = 0 pour donner le rôle d'esclave
- AT + UART = 9600 pour définir le baud rate à 9600
- AT + ROLE = 1 pour donner le rôle de maitre

- AT + BIND = « Adresse de l'esclave » pour permettre au deux HC-05 de se connecter automatiquement (pairing)

Figure 14: Configuration de l'esclave

On observe sur la figure 14 les réponses du HC-05. Ici on configure l'esclave. On test la connexion, il renvoie OK pour signaler que la connexion est établie. On définit son rôle, demande son adresse et configurer le baud rate à 9600.
Il est important que les deux modules ai le même baud rate et les mêmes mots de passe (Le mot de passe est initialement 1234 et peut être modifié avec la commande AT + PSWD = « nouveau mot de passe».

Nous avons utilisé un Arduino Due (car c'est la seule carte que nous avons mais il possible d'utiliser un autre).

Schéma électrique

Pour envoyer les commandes AT sur le module Bluetooth, il faut brancher le Vcc à 5 V pour l'alimenter et le Key à 3.3 V pour le forcer en mode AT (si on utilise un HC-06, il n'est pas nécessaire de connecter le Pin Key car ce module est par default en mode AT). Pour savoir si le module est bien en mode AT, on regarde sa led, si elle clignote rapidement on n'est pas en dans le bon mode, si elle s'allume une fois chaque deux secondes, on est en mode AT . Il faut ensuite brancher le Pin Tx du Hc-05 au Pin Rx de l'Arduino et le Pin Rx au Pin Tx. Sur l'Arduino Due il existe plusieurs Pins Rx et Tx, il faut que les Pins choisis soit en accord avec le programme téléversé, dans notre cas on utilise Rx1 et Tx1 (Pin 19 et 18 respectivement).

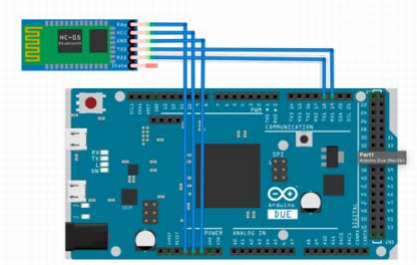


Figure 15: Montage permettant la configuration des HC-05

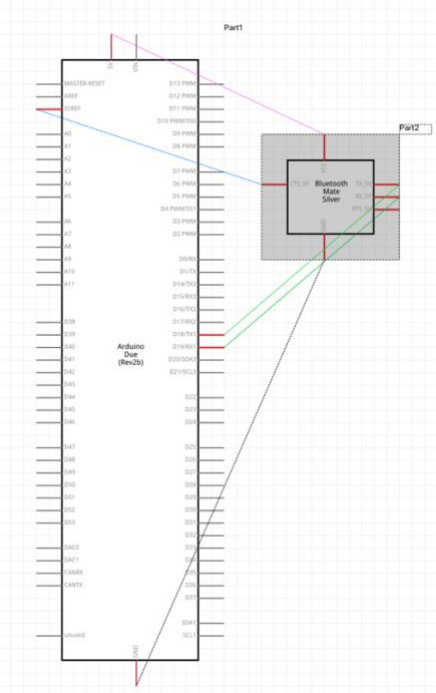


Figure 16: Schéma électrique équivalent

Programmation de l'Arduino

```
void setup()
{
    Serial.begin(38400);
    Serial1.begin(38400);
    Serial.print("begin");
}

void loop()
{

  while(Serial1.available()) {
    Serial.write(Serial1.read());
  }
  delay(100);


  while(Serial.available()) {
    Serial1.write(Serial.read());
  }
}
```

Figure 17: Programme Arduino permettant l'envoie des commandes AT

Bluetooth HC-05 (STM32)

Receive/transmit (interruption)

Pour transmettre et recevoir les données par Bluetooth, nous utiliserons Usart 1 pour communiquer entre STM32 et Hc-05, nous initialisons le Pin 9 en Usart1_Tx et le Pin 10 en Usart_Rx. Il est nécessaire d'initialise Usart 1 en mode asynchrone, d'autorisé global interrupt et de configurer le baud rate à celui du HC-05 (9600 dans notre cas).

Pour la transmission il suffit d'utiliser la fonction.
HAL_UART_Transmit(&huart1,txData,sizeof(txData),100)
&huart1 : indique qu'on utilise Usart 1
txData est la donnée qu'on veut transmettre
sizeof(txData) est la taille de la donnée
100 est le timeout, si la donnée n'est pas envoyée avant le timeout on passe à la prochaine ligne du code (ce qui permet de ne pas être bloqué en cas d'erreur)

Pour la réception on utilise une interruption :

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{

if(huart->Instance==USART1)
  {

    HAL_UART_Receive_IT(&huart1,rxData,100);

    lcd_position(&hi2c1 ,0, 0);
    lcd_print(&hi2c1,(char*)rxData);

  }
}
```

Figure 18: Fonction d'interruption permettant la réception de données

Lorsqu'un signal est reçu on rentre dans la fonction d'interruption où si le signal est sur l'Usart 1 on le récupère avec la fonction HAL_UART_Receive_IT avant de l'afficher sur l'écran LCD.

Un problème rencontré avec cette approche est que le programme rentre dans la fonction d'interruption une seule fois. D'après community.st ceci pourrais être relié à un bit dans le registre d'interruption qui ne passerais pas à 1 lorsque la fonction est terminée. Nous avons décidé de prendre une autre approche pour la communication Bluetooth.

Pour savoir si les deux HC-05 sont connectés, il suffit d'observer leurs leds. Si le module est en attente de connexion sa led clignote rapidement. Si les deux modules sont connectés les leds clignote lentement et de manière synchrone.

Receive/transmit (DMA)

La transmission et la réception des données par DMA (Direct Memory Access) qui un procédé qui permet l'envoi direct de données d'un périphérique connecté vers la mémoire.

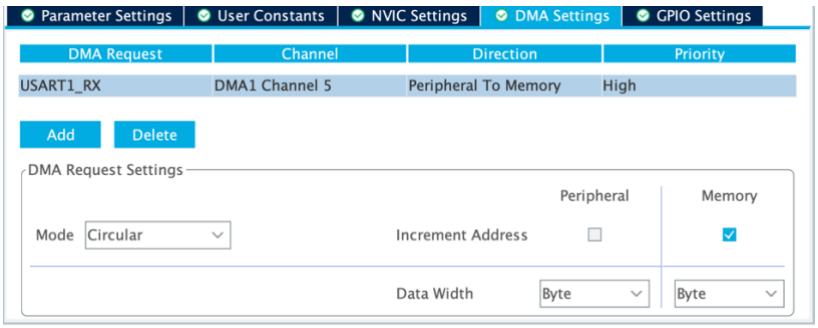


Figure 19: Initialisation du protocole DMA dans CubeMX

Pour l'initialisation, dans DMA settings on rajoute un DMA request et on le met en mode circular.

Il suffit à présent d'utiliser la fonction HAL_UART_Transmit_DMA pour la transmission et HAL_UART_Reception_DMA pour la réception (ici aucune fonction d'interruption n'est nécessaire).

Schéma électrique

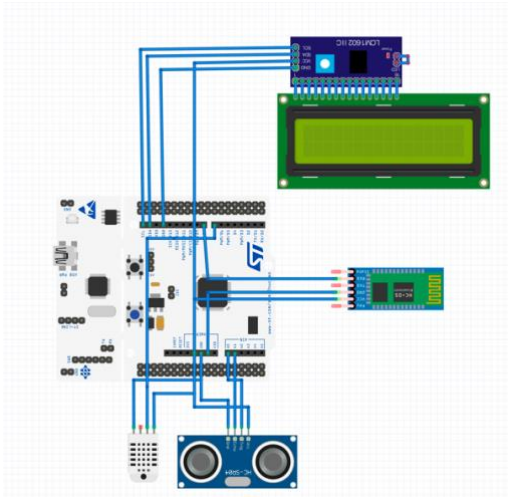


Figure 20: Pour la transmission

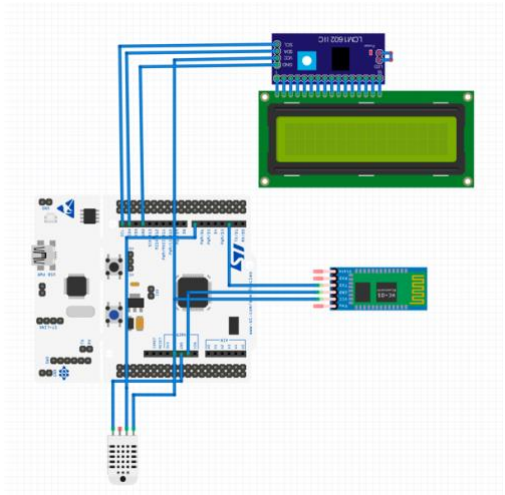


Figure 21: Pour la réception

On rajoute sur les montages déjà effectué les modules HC-05. La figure 20 montre le circuit qui vas transmettre les données, on connecte donc uniquement le Pin Rx du HC-05 au Pin 9 (Tx). On ne connecte pas le Pin Tx du HC-05 car aucune réception n'est demandée par ce microcontrôleur. Inversement pour la figure 21 où on connecte uniquement le Pin Tx du HC-05 au Pin 10 (Rx).

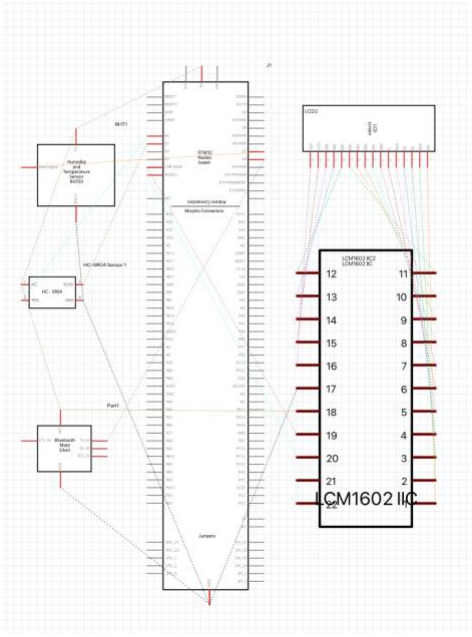


Figure 22: Pour la transmission

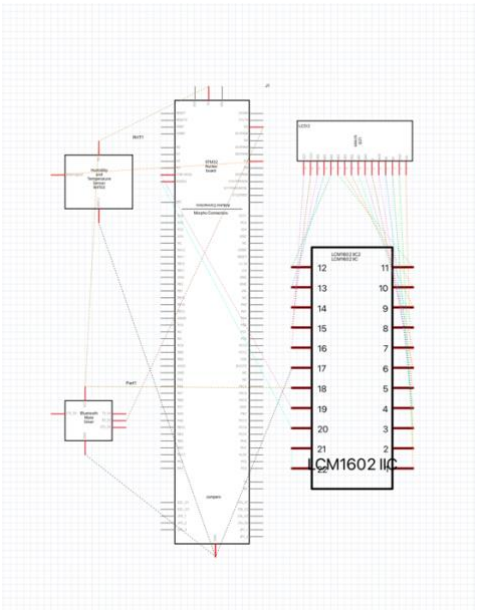


Figure 23: Pour la réception

Présentation de notre projet BE

Lors d'un stage en laboratoire, l'utilité des abaques nous a été démontré en raison de la quantité de situation où elle été nécessaire à nos travaux. Nous avons donc choisi de travailler sur un abaque de température en fonction de la distance et du temps. C'est dans ce but que nous utiliserons deux STM32, le premier sera fixe près du point chaud et mesura la température. Le deuxième sera mobile et se déplacera et mesurera la température et la distance. Le microcontrôleur mobile enverra la température mesurée par Bluetooth. Le point chaud dont nous faisons la mesure sera simuler par un radiateur.

Prise de mesures

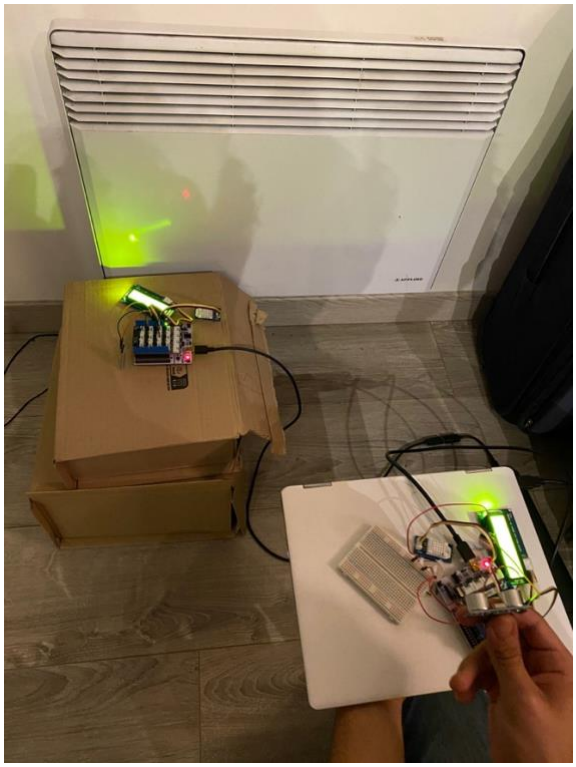


Figure 24: Photo montrant la prise de mesure

On place le STM32 responsable de la réception de données au plus proche du radiateur et on déplace le STM32 responsable de la transmission de 20 cm de façon à relever les données chaque 5 secondes pour une durée de 1 minutes.

Affichage des données

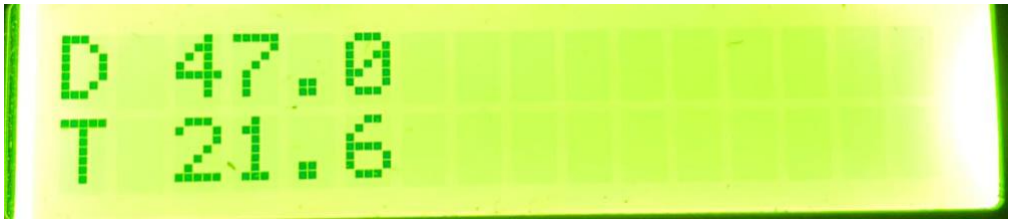


Figure 25: écran LCD (transmission)

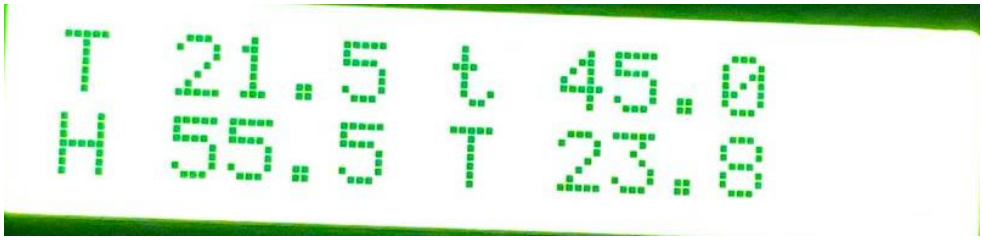


Figure 26: écran LCD (réception)

Sur la figure 25 on observe l'écran LCD lié au STM32 responsable de la transmission. On peut lire la distance en haut et la température en bas.

Sur la figure 26 on observe l'écran LCD lié au STM32 responsable de la réception. On peut lire la température reçue par Bluetooth en haut à gauche, le temps en haut à droite, l'humidité en bas à gauche et la température en bas à droite.

L'Abaque

On récupère alors les données que nous mettons dans un tableau :

Distance	20		40		60		80	
Temps	Temperature	Temperature Reçu 1	Temperature 2	Temperature Reçu 2	Temperature 3	Temperature Reçu 3	Temperature 4	Temperature Reçu 4
0	25,7	25,1	26,1	25	26,4	25,1	26,7	24,9
5	25,7	25,2	26,1	25,1	26,5	25,1	26,7	24,9
10	25,7	25,2	26,1	25,1	26,5	25,1	26,6	24,8
15	25,8	25,3	26,1	25,1	26,5	25	26,7	24,8
20	25,8	25,3	26,1	25,2	26,5	25	26,7	24,8
25	25,8	25,4	26,1	25,2	26,5	25	26,7	24,9
30	25,8	25,4	26,2	25,3	26,5	25,1	26,7	24,9
35	25,9	25,6	26,2	25,3	26,5	25,1	26,7	24,9
40	25,9	25,6	26,3	25,4	26,6	25,1	26,7	24,9
45	25,9	25,6	26,3	25,4	26,6	25,1	26,7	24,8
50	25,9	25,7	26,3	25,4	26,6	25,1	26,7	24,8
55	26	25,7	26,3	25,5	26,6	25,1	26,8	24,8
60	26	25,8	26,4	25,5	26,6	25,1	26,8	24,8

Figure 27: tableau de donnée

On dessine alors l'abaque :

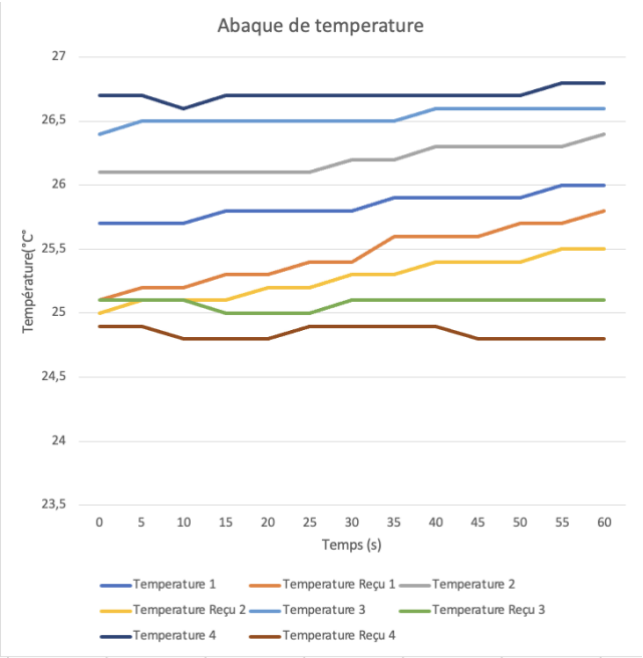


Figure 28: Abaque de température

Pour avoir plus d'information on décide d'étudier la différence de température entre la température mesurée et la température reçu par Bluetooth.

Distance	Temperature - Temperature reçu	
20	0,2	
40	0,9	
60	1,5	
80	2	

Figure 29: Différence de température en fonction de la distance

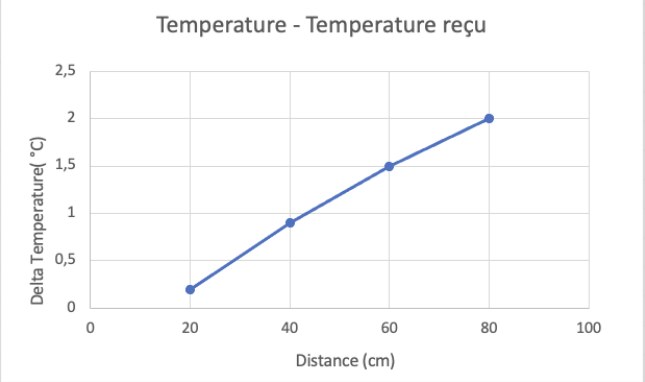


Figure 30: Différence de température en fonction de la distance

Bien évidement la différence augmente avec la distance.

Il est important de noter que nous avons arrêté les mesures à une distance de 80 cm car nous avions atteint la température de la pièce (24,8) et donc la température aurait été constante après ce point.

On relève aussi l'humidité. Les valeurs dans le tableau correspondent à l'humidité mesuré en même temps que température 1.

Temps	Humidité
0	60,8
5	60,7
10	60,4
15	60,1
20	59,7
25	59,7
30	59,5
35	59,3
40	59,1
45	58,9
50	58,8
55	58,9
60	58,8

Figure 31: Humidité en fonction du temps

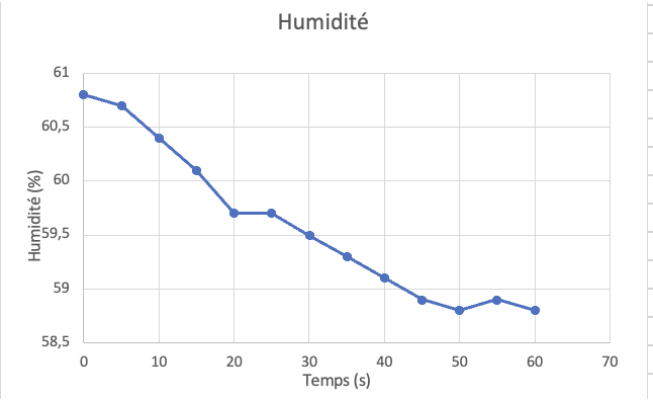


Figure 32: Humidité en fonction du temps

Pour aller plus loin

Pour améliorer notre projet, nous pourrions concevoir des PCB type shield sur lesquels on pourrait insérer notre STM32 et nos composants ainsi qu'une pile qui alimenterait le microcontrôleur, ce qui nous permettrai d'utiliser nos montages sans avoir à le connecter à l'ordinateur.

Il est également possible de faire l'abaque en traçant la température en fonction de la distance (opposé au temps).

Annexe

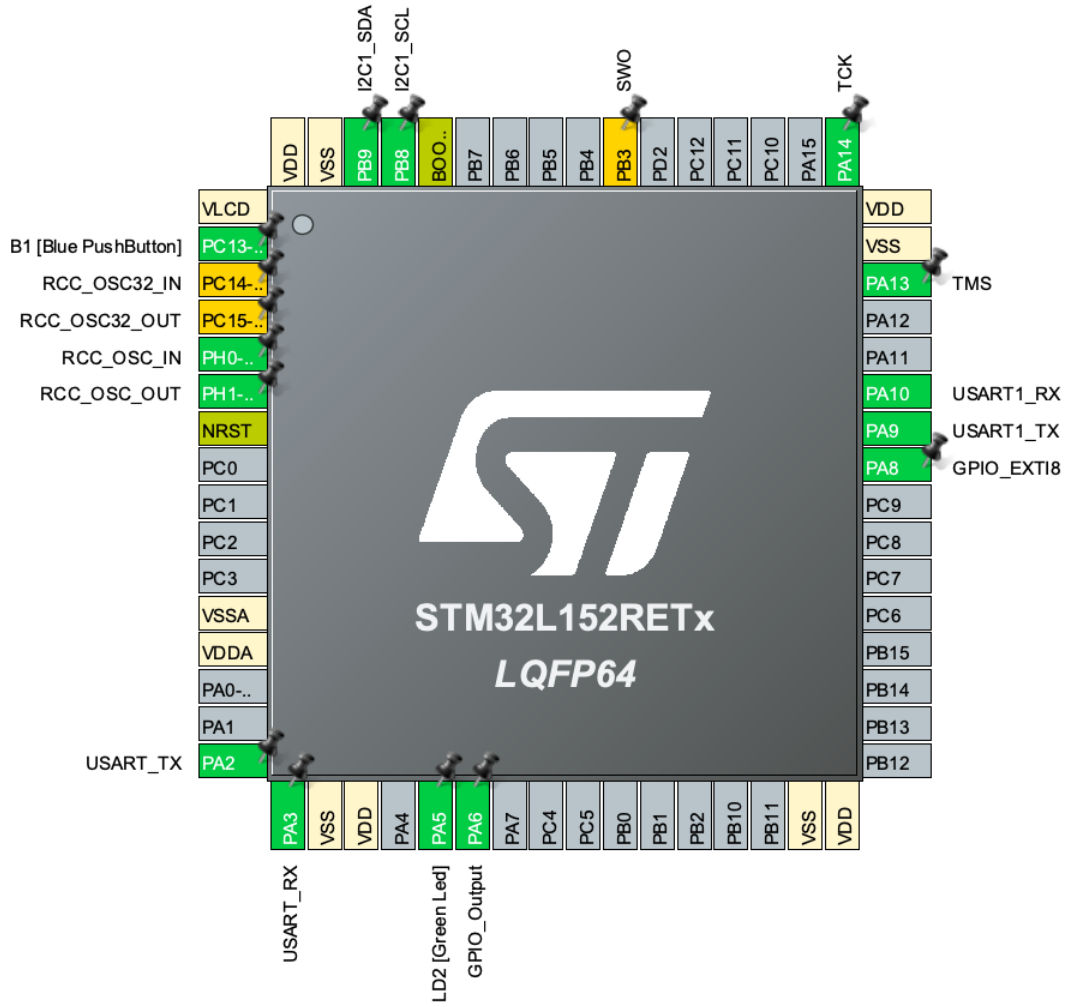


Figure 33: Configuration des Pins (Réception)

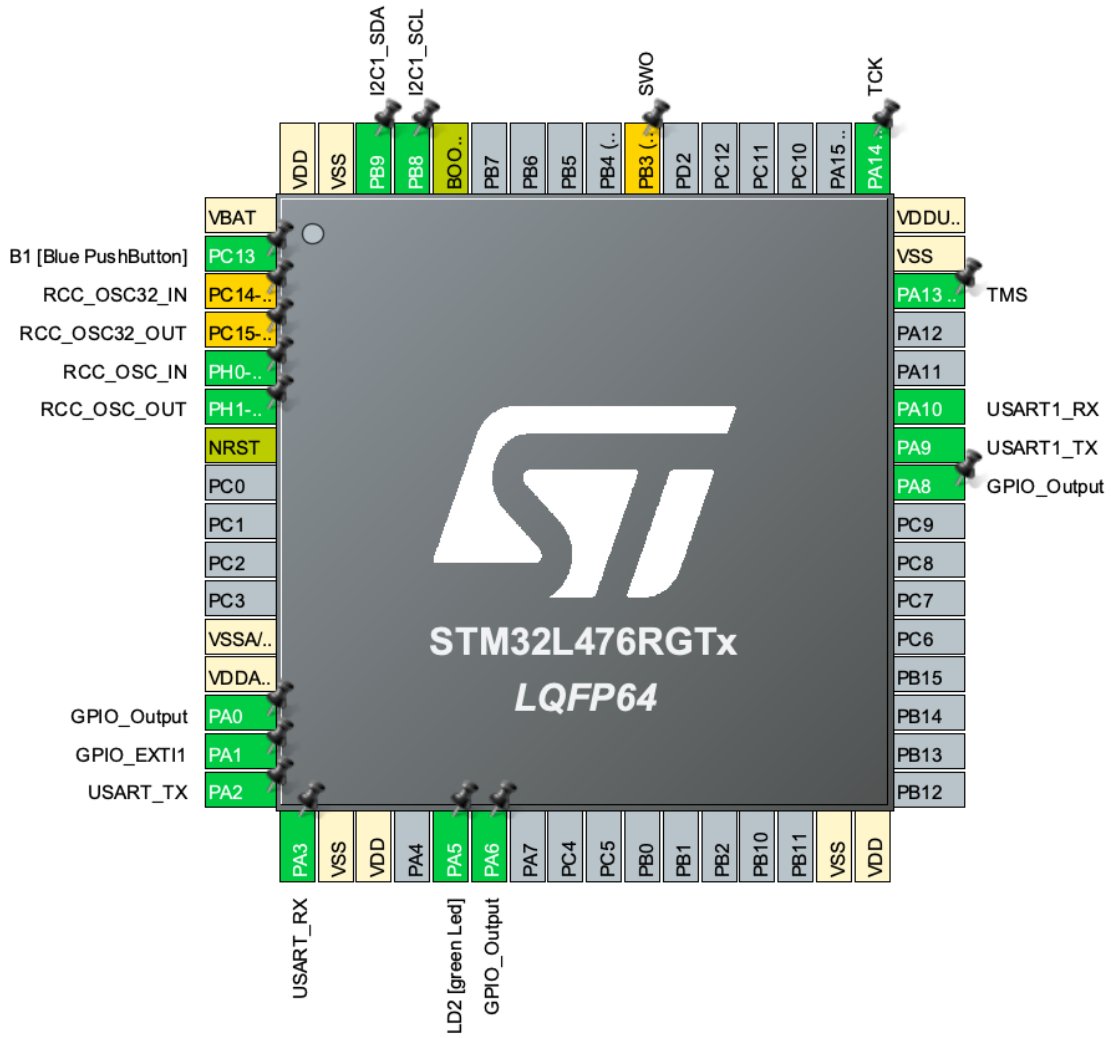


Figure 34: Configuration des Pins (Transmission)