Notice d’application :

Insertion du mode 21 de la sonde HC2A

Projet P23B09 : Développement d’un capteur IOT LoraWAN

Etudiants : Enzo Lacheze et Alexis Fournier

Tuteur Polytech : Alexis Landrault

Client : Gael Godi

Tuteur industriel : Pascal Fickinger

Polytech Clermont GE5A 2023-2024

Sommaire

Table des figures

Glossaire

Fichier .c : « Un fichier .c est un fichier de code source écrit dans le langage de programmation C. Il contient des fonctions, des variables et d'autres constructions utilisées pour créer un programme qui peut être compilé en code machine pour l'exécution sur un ordinateur. Le fichier .c est généralement édité à l'aide d'un éditeur de texte ou d'un environnement de développement intégré (IDE) puis compilé à l'aide d'un compilateur C, tel que GCC, pour générer un fichier exécutable. »

Fichier .h : « Un fichier .h est un fichier d'en-tête en C. Il contient généralement des déclarations de variables, de fonctions et de structures de données utilisées par d'autres fichiers source dans un projet. Les fichiers d'en-tête sont généralement inclus dans les fichiers source C à l'aide de l'instruction #include. Les fichiers d'en-tête permettent de regrouper les déclarations communes dans un seul emplacement pour une utilisation répétée dans plusieurs fichiers source, ce qui facilite la maintenance et la modification du code.»

Driver : « Un "driver" (pilote en français) est un logiciel qui permet à un système d'exploitation de communiquer avec un périphérique matériel spécifique. Dans notre projet il s'agit de fichier .c et .h qui permettent de communiquer et d'utiliser un type de sonde particulier.

Main.c : « Le fichier "main.c" est un fichier qui contient généralement la fonction principale, appelée "main", qui est le point d'entrée du programme. Lorsqu'un programme en C est exécuté, l'exécution commence par la fonction "main". Les autres fonctions et programmes sont appelés à partir du main.

TTN : « TTN (The Things Network) est une initiative open source qui vise à créer un réseau mondial décentralisé pour l'Internet des objets (IoT) basé sur la technologie LoRaWAN (Long Range Wide Area Network). Elle se traduit par une plateforme gratuite qui permet de récupérer les données de capteurs afin de les stocker ou de les utiliser pour une utilisation ultérieure.

LoraWan : « LoRa est une modulation radio spécifique utilisée pour la transmission des données sans fil. C'est une technologie de modulation qui permet une communication à longue portée. Elle est conçue pour offrir une longue portée de communication (plusieurs kilomètres) et une faible consommation d'énergie. Elle est adaptée aux applications où les dispositifs doivent transmettre des petites quantités de données sur de longues distances tout en conservant une faible consommation d'énergie. LoraWan est un protocole de communication de réseau qui utilise la technologie LoRa pour permettre la communication à longue portée dans le domaine de l'IoT. »

CR : « En C, un retour chariot (Carriage Return) est représenté par la séquence d'échappement '\r'. C'est un caractère spécial qui, lorsqu'il est rencontré par le compilateur, fait passer le curseur au début de la ligne courante sur l'écran, écrasant tout ce qui se trouve à droite de cette position. Il est souvent utilisé conjointement avec '\n' pour créer des retours chariots et des sauts de ligne. Dans le driver, il sert à signifier à la sonde la fin de la commande. »

Introduction

Le dragino LSN50-V2 est un nœud de capteur LoRaWAN longue portée. C'est un dispositif conçu pour l'enregistrement de données en extérieur pour une utilisation à long terme et une transmission de données sécurisée. Il est conçu pour permettre aux développeurs de déployer rapidement des solutions LoRa et IoT au niveau industriel.

Figure 1: structure du réseau LoraWan

Il est défini par des modes de fonctionnements qui utilisent un ou plusieurs types de sondes différentes. Plusieurs modes de fonctionnement ainsi que plusieurs types de sondes différentes sont déjà pré-enregistrés dans l'appareil.

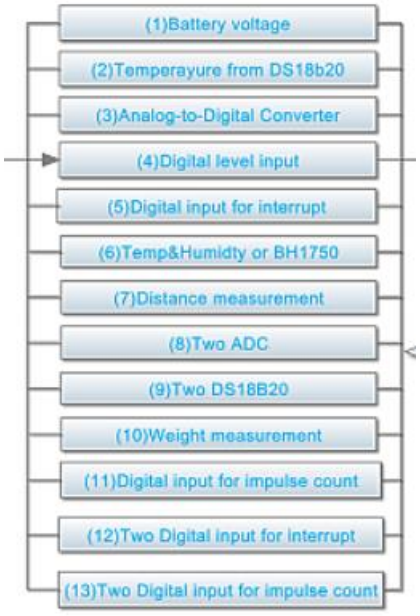

Figure 2 : Modes de fonctionnement du dragino

L'objectif du projet était de réaliser un nouveau mode de fonctionnement pour le dragino qui prendrait un nouveau type de sonde, la sonde HC2A. Une seule sonde était demandé pour le nouveau mode de fonctionnement.

Figure 3 : Sonde HC2A

Utilisation de la sonde HC2A

La sonde HC2A est une sonde développée par Rotronic pour la mesure de précision de température et d'humidité. Le dragino, l'appareil qui récupère les données de la sonde et les envois via protocole LoraWan sur TTN est connecté à la sonde via les broches du dragino.

La connexion de la sonde est représenté figure 2

Dragino	HC2A
VDD	VCC
GND	GND
PA9	RX
PA10	TX

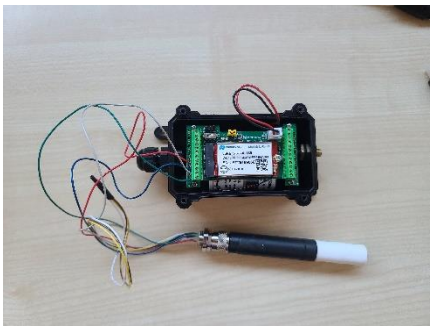

Figure 4 : Branchement de la sonde au dragino

Ecriture du driver de la sonde

Le driver est l'élément le plus important et le plus difficile à réaliser lors de la création d'un mode. C'est le fichier qui va s'occuper de récupérer les données de la sonde qui seront ensuite envoyé via protocole LoraWan sur TTN. La sonde possède plusieurs mode de communication mais notre client nous a conseillé d'utiliser une communication UART afin de communiquer avec la sonde.

La sonde HC2A est équipée d'une puce « airchip 3000 » qui s'occupe de récupérer les données de température et d'humidité et d'envoyer ces données sous forme de trame vers le destinataire.

La trame est envoyé lorsque la puce reçoit une commande spécifique :

3.4 RDD command: read values

Returns the measured and calculated values as well as the information necessary to interpret the data (calculated parameter type, engineering units, status, serial number and name of the device, etc.)

Command format:

{	ID	Adr	RDD	Chksum or }	CR

Answer format:

{	ID	Adr	rdd	Data	Chksum

Figure 5 : Commande de lecture des valeurs de la sonde

Exemple :

```
{F04RDD}
{F04rdd 001;  4.45;%RH;000;=; 20.07;°C;000;=;Fp;-19.94;°C;000;+;001;B2.8;0000000002;HyClp 2     ;006;J^M
```

Figure 6 : Exemple de trame reçue par la sonde

Dans notre cas, comme nous n'avons qu'une seule sonde à utiliser, on remplace l'identification ID et l'adresse Adr par respectivement un espace et 99. On a donc « { 99RDD}\r »

La trame de retour ne peut pas être récupérée de manière normale. Afin d'établir une communication UART entre le dragino et la sonde on a besoin de créer une liaison UART propre. Le dragino possède déjà une liaison série UART sur les broches PA2 et PA3 :

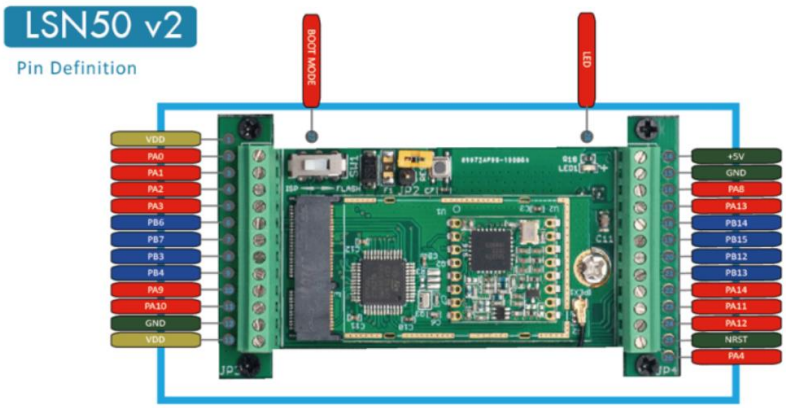

Figure 7 : Définition des broches du dragino

4	PA2	In/Out	Directly from STM32 chip, 10k pull up to VCC	Used as UART_TXD in LSN50 image
5	PA3	In/Out	Directly from STM32 chip, 10k pull up to VCC	Used as UART_RXD in LSN50 image

Figure 8 : Fonction des broches PA2 et PA3

Or ces broches sont déjà utilisés lors de la communication série entre l'ordinateur et le dragino, il faut donc créer une nouvelle liaison UART en créant un port virtuel pour la nouvelle communication.

Le dragino possède plusieurs modes déjà préconfigurés qui prennent en charge plusieurs capteurs différents avec plusieurs protocoles de communication spécifiques

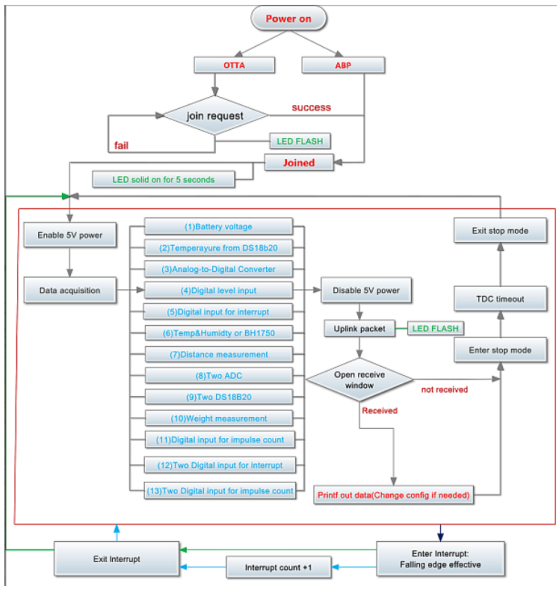

Figure 9 : Diagramme de fonctionnement du dragino

Parmi eux le mode 2, qui permet d'utiliser plusieurs types sondes de distance possède un capteur qui fonctionne uniquement en UART. Le TF-mini plus est un capteur de distance qui communique en liaison UART avec le dragino en créant un port virtuel pour la communication. Les fonctions à utiliser sont contenues dans les fichier vcom.c et vcom.h. Nous nous sommes donc servis de l'exemple du capteur de distance et avons créer nos propres fonctions d'initialisations et d'utilisation du port. Nous avons développés une fonction d'initialisation du capteur et de la liaison UART(BSP_hc2a_init) qui utilise des fonctions que nous avons crées dans le fichier vcom.c et qui mettent en place le port virtuel. Nous avons repris le même branchement que celui du capteur de distance TF-mini plus pour pouvoir ré-utiliser les fonctions d'initialisation des ports.

La fonction BSP_hc2a_init vérifie que l'on a bien une sonde qui communique avec le dragino.

```
void BSP_hc2a_init(void)
{
  uart41_init_uart41(); //creation du port virtuel de la liaison UART
  uart1_IoInit(); //initialisation des ports PA9 pour le RX et PA10 pour le TX
  if (HAL_UART_Init(&UartHandle1) != HAL_OK)
  {
    /* Initialization Error */
    PPRINTF("Error: HC2A UART bus Init failure\r\n");
    Error_Handler();
  }
  if (HAL_UART_Init(&UartHandle1) == HAL_OK)
  {
    PPRINTF("\r\nHC2A UART bus Init success\r\n");
  }
  uart1_IoDeInit();
}
```

Figure 10 : Fonction d'initialisation de la sonde

La fonction uart41_init_uart41() crée un port virtuel avec les spécificités données dans la documentation technique de la sonde.

```c
void uart4l_init_uart4l(void)
{
  UartHandle1.Instance        = USART1;

  UartHandle1.Init.BaudRate   = 19200;
  UartHandle1.Init.WordLength = UART_WORDLENGTH_8B;
  UartHandle1.Init.StopBits   = UART_STOPBITS_1;
  UartHandle1.Init.Parity     = UART_PARITY_NONE;
  UartHandle1.Init.HwFlowCtl  = UART_HWCONTROL_NONE;
  UartHandle1.Init.Mode       = UART_MODE_TX_RX;

  if(HAL_UART_Init(&UartHandle1) != HAL_OK)
  {
    Error_Handler();
  }
}
```

Figure 11 : Fonction d'initialisation de la liaison UART

La fonction uart1_IoInit() met les broches PA9 et PA10 en sortie et entrée de la liaison UART et les affecte à cette liaison.

La fonction de récupération des données de température et d'humidité appelle une fonction situé dans vcom.c

```c
void read_data_hc2a(hc2a_sensor *hc2a_t)
{
    PPRINTF("START MODE 21");
    uart1_IoInit(); // Initialize UART once before the loop

    hc2a_receive_data(&hc2a_t->temp, &hc2a_t->hum, 2000); // Assuming hc2a_receive_data takes pointers as arguments

    uart1_IoDeInit();
}
```

Figure 12 : Fonction de lecture des données de la sonde dans le fichier hc2a.c

La fonction est séparé en 3 parties, la transmission de la commande, la réception de la trame et l'extraction des données utiles.

```c
void hc2a_receive_data(float *tempe, float *humi,uint16_t delayvalue)
{
  //initialisation des variables
  uint8_t txdatas[10] = "{ 99RDD}\r"; //commande : { 99RDD}
  uint8_t rxdatatrame[200];
  uint8_t begin=0,datanumber=0;
  float humidite=0;
  float temperature=0;
  float k=10;
  num=0;
  uint32_t currentTime = TimerGetCurrentTime();
  //transmission de la commande de lecture
  HAL_UART_Transmit(&UartHandle1, txdatas, 10, 0xFFFF);//envoie la trame de lecture de données
  while(UartHandle1.gState != HAL_UART_STATE_READY) //test du temps de réponse de la liaison
  {
    if(TimerGetElapsedTime(currentTime) >= 5000)
    {
      PRINTF("\r\n depassement");
      break;
    }
  }
 //reception de la trame
 flags_command=1; //flag de réception de données
 HAL_UART_Receive_IT(&UartHandle1, (uint8_t *)aRxBuffer,1);
 HAL_Delay(delayvalue);
 flags_command=0;
 for(uint8_t number=0;number<sizeof(response);number++)
 {
   if(begin==1) //rxdatatrame prend le caractère receptionnée de la liaison UART.
   {
     rxdatatrame[datanumber++]=response[number];
     if(datanumber==150) //si la trame fait plus de 150 caractères arreter de recuperer la trame
     {
       begin=2;
     }
   }
   else if ((response[number]==0x7b)&&(begin==0)) //si le caractère de la trame est un "{" alors commencer la recuperation dans rxdata
   {
     rxdatatrame[datanumber]= response[number];
     begin=1;
   }
 }
 HAL_Delay(1000);
 PRINTF("\r\n");
 HAL_Delay(500);
 //extraction de la temperature et de l'humidite
 for (uint8_t i = 0; i < 2; i++)
       {
           humidite = humidite + ((rxdatatrame[12 + i] - 48) * k) + ((rxdatatrame[15 + i] - 48) * (k / 100));
           temperature = temperature + ((rxdatatrame[29 + i] - 48) * k) + ((rxdatatrame[32 + i] - 48) * (k / 100));
           k = k / 10;
       }
       *tempe=temperature;
       *humi=humidite;
}
```

Envoi de la commande

Réception de la trame

Extraction de la temperature et de l'humidité

Figure 13 : Programme d'envoi et de réception de la trame de la sonde

Pour transmettre et recevoir les données on utilise des fonctions déjà existantes dans le code du dragino. Comme la trame a une taille fixe, on connait la position des valeurs de températures et d'humidité dans la trame ce qui permet de les extraire. Comme on a pas besoin de récupérer toute la trame, on s'arrête à 150 caractères. La fonction retourne un pointeur ce qui permet de récupérer les valeurs dans le hc2a.c .

Dans le fichier hc2a.c, les valeurs sont stockées dans une structure hc2a_t qui contiendra les valeurs de température et d'humidité.

Modification du BSP

Les parties du BSP qui sont importantes sont l'initialisation des sondes en fonction du mode choisis et la récupération des données des sondes

L'initialisation de la liaison consiste en un simple appel de la fonction d'initialisation de la liaison UART qui va tester la liaison UART et mettre les broches de la sonde en entrée et sortie suivant la broche.

```
else if (mode == 21)
 {
   PRINTF("\r\nDEBUT INIT \r\n");
   BSP_hc2a_init(); //appel de la fonction initialisation de la sonde hc2a
   PRINTF("FIN INIT \r\n");
 }
```

Figure 14 : initialisation de la sonde dans bsp.c

La récupération des données de la sonde se fait en récupérant les valeurs de gain et d'offset afin de corriger la valeur de la température selon notre choix.

```
else if (mode==21)
{
  read_data_hc2a(&hc2a_t);
  sensor_data->hc2a_t.offset= offset_hc2a;
  sensor_data->hc2a_t.gain=gain_hc2a;
  sensor_data->hc2a_t.temp=hc2a_t.temp;
  sensor_data->hc2a_t.hum=hc2a_t.hum;
  float t=(sensor_data->hc2a_t.temp+sensor_data->hc2a_t.offset)*sensor_data->hc2a_t.gain;
  sensor_data->hc2a_t.temp=t;
  PPRINTF("HC2A_offset:%.2f, HC2A_gain:%.2f, HC2A_temp:%.2f,  HC2A_hum: %.2f\r\n" ,sensor_data->hc2a_t.offset,sensor_data->hc2a_t.gain,sensor_dat
}
```

Figure 15 : récupération des valeurs dans le BSP

Sensor_data est une variable de type structure qui possède toutes les valeurs de tous les capteurs disponible pour le dragino. A l'intérieur de cette structure, nous avons préalablement inséré notre propre structure de la sonde (hc2a_t).Les valeurs de température, humidité, gain et offset sont mis dans sensor_data. Puis on recalcule la température en fonction du gain et de l'offset, les valeurs sont ensuite affichées si on est connecté via l'ordinateur.

Modification du main.c

Le main est le programme qui prépare les trames qui seront envoyés via protocole LoraWan. La modification du main reprend la même structure que les autres modes de fonctionnement du dragino. Cela permet d'utiliser le fichier de décodage des trames (payload) d'origine qui sera mis sur TTN sans trop de modification. En faisant cela, une partie de la trame envoyée est inutile mais cela permet à celui qui utilise le dragino de ré-utiliser l'appareil pour les anciens modes sans modifier le payload.

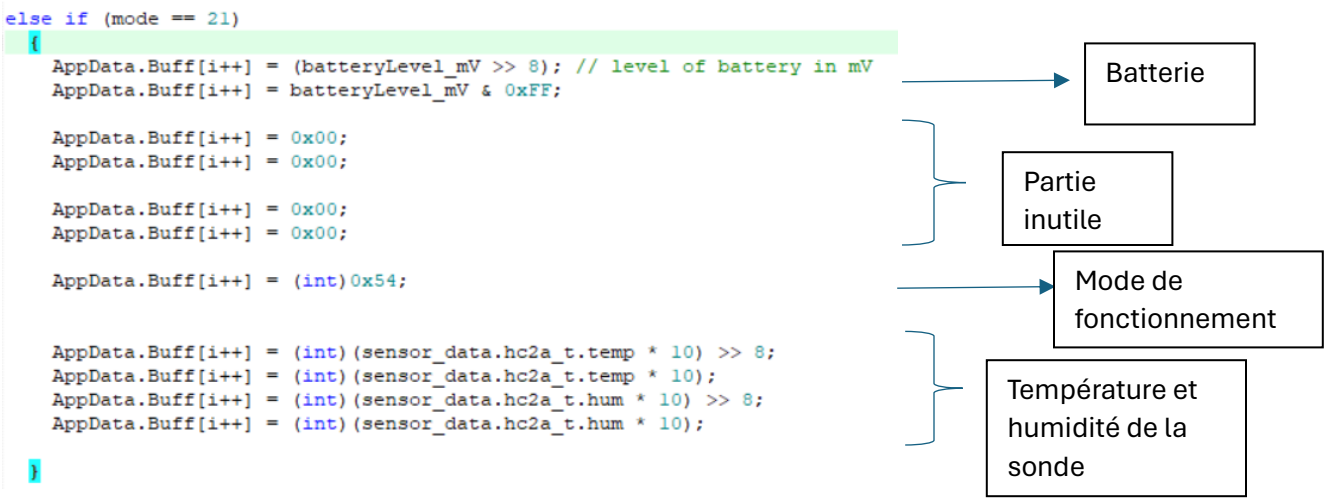

Figure 16 : préparation de la trame Lora à envoyer

Bibliographie

Page wiki du dragino :
http://wiki.dragino.com/xwiki/bin/view/Main/User%20Manual%20for%20LoRaWAN%20End%20Nodes/LSN50%20%26%20LSN50-V2%20-%20LoRaWAN%20Sensor%20Node%20User%20Manual/