Wattebled Arnaud
Lamarche Dorian
Zanin Florian

UNIVERSITÉ PARIS SUD

IUT DE CACHAN

université PARIS-SACLAY

Rapport Station météo

LP MECSE 2017/2018

Professeur : Xavier Mininger

Introduction

Ce document a pour but de vous présenter le projet de la station et comment nous l'avons réalisé.
Le projet de la station météo a donc pour but de récupérer diverses informations météorologiques (température, pression atmosphérique, humidité, vitesse du vent,...), de les envoyer via bluetooth à une raspberry pi qui servira de serveur web afin d'afficher les données sur une page web.
Au début nous avions une station météo de Lacrosse Technology mais étant donné que la transmission de la station vers le pc ne fonctionnait pas nous avons décidé de créer notre station météo à l'aide d'une carte keil et de divers capteurs. Le schéma synoptique se trouve ci-dessous.

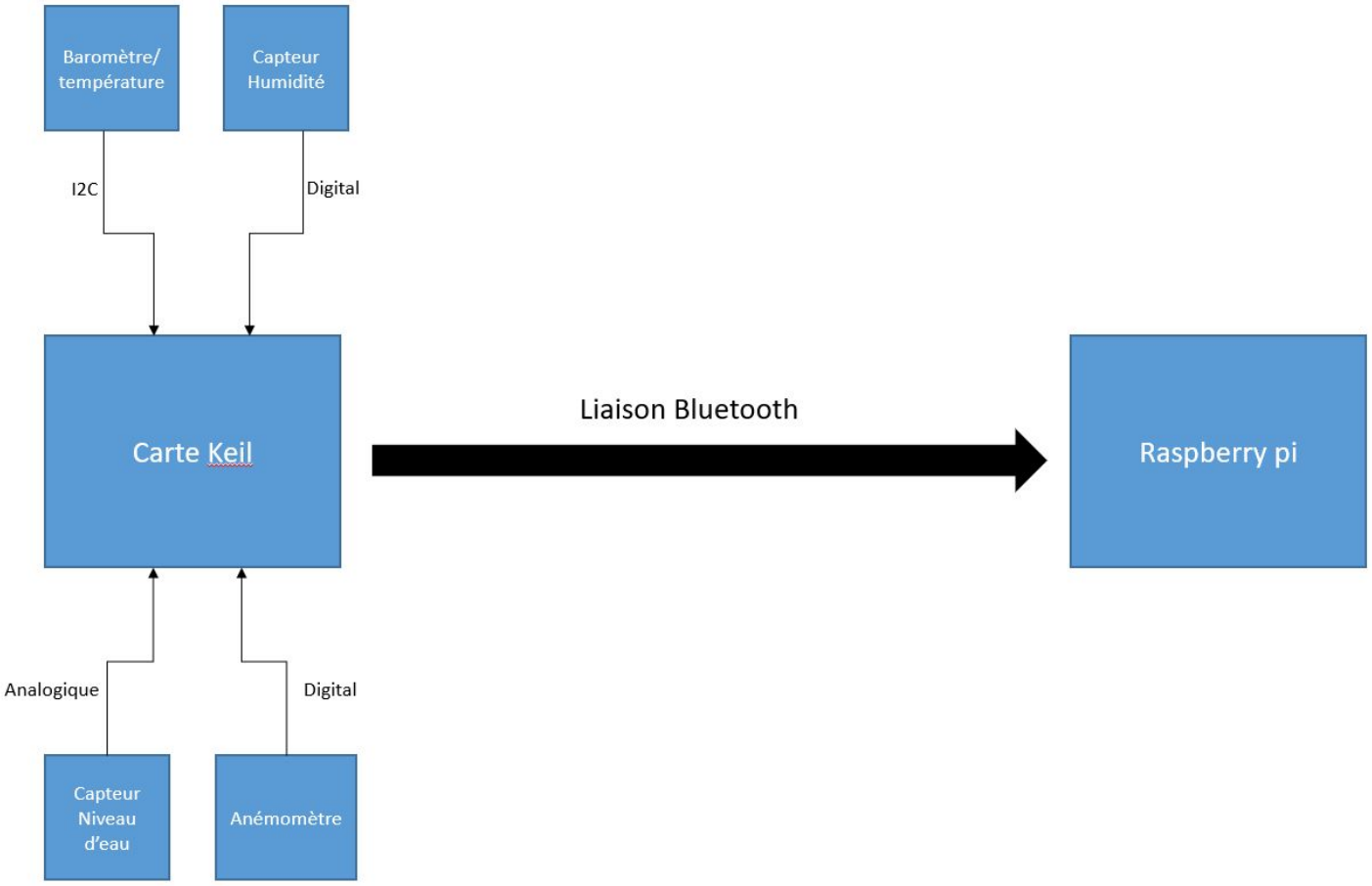

figure 1 : schéma synoptique de la station météo

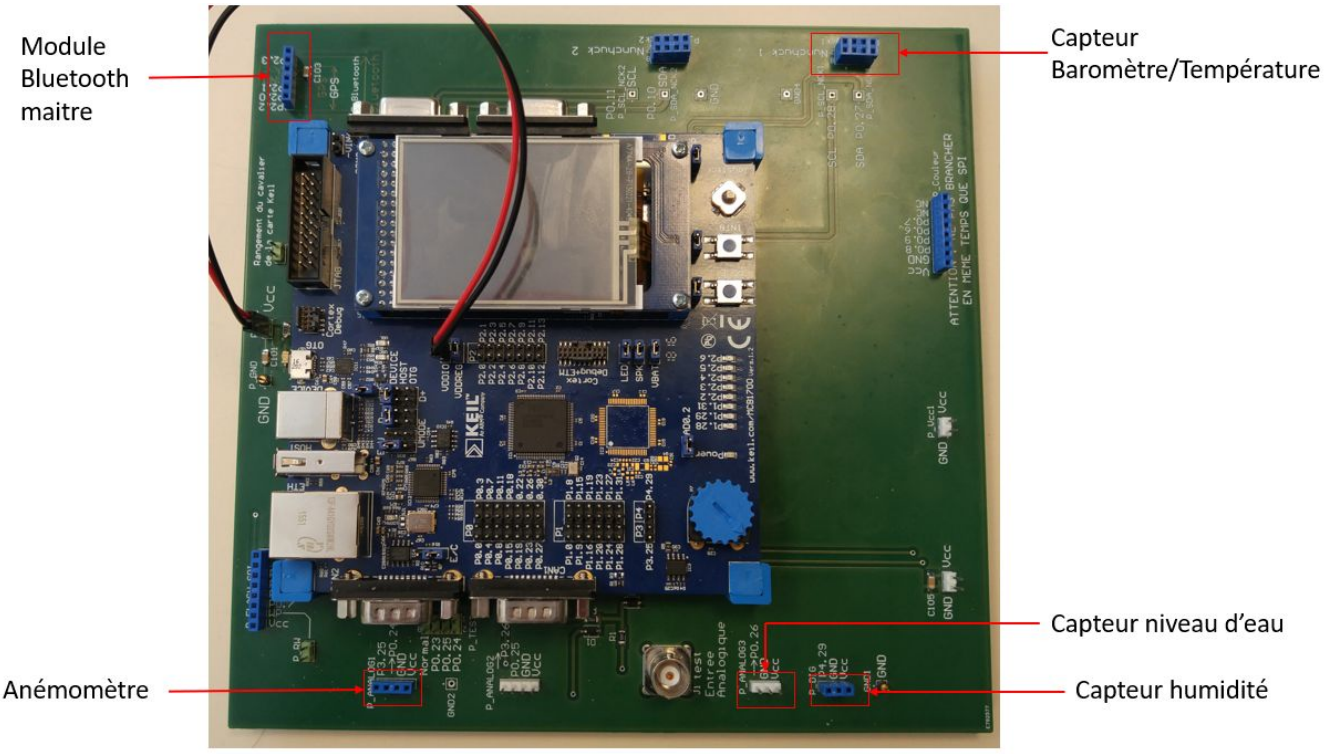

Figure 2 : Photo du branchement des capteurs sur la Keil

Sommaire

Partie 1 Les différents capteurs

Pour pouvoir créer une station météo fonctionnelle, plusieurs capteurs ont été choisis afin de réaliser cette partie .
Nous nous sommes dirigé vers un capteur de niveau d'eau, un capteur de pression et température ainsi qu'un capteur d'humidité puis un anémomètre coupler avec une girouette. Tous ces capteurs ont dû être programmés individuellement.

Capteur de niveau d'eau

Le capteur de niveau d'eau est un capteur analogique, branché sur la carte Keil sur les ports *P ANALOG3*. On peut récupérer les données sur le Port "*P0.26*" avec une plage de 4096 valeurs.

```
int main (void)
{

  int16_t niveau;
  //int tour;int valVT;
  char affichage[50];

  ADC_Initialize();     //initialisation des ports analogiques

  GLCD_Initialize();    //initialisation de l'ecran LCD
  GLCD_ClearScreen();
  GLCD_SetFont(&GLCD_Font_16x24);

  while(1) {

  ADC_StartConversion();    //se connecte au capteur Analogique

  while(ADC_ConversionDone() !=0);
  niveau = ADC_GetValue();      //recupere la valeur entre 4096 et 0
```

figure 3 : Programme capteur d'eau

figure 4 : capteur de niveau d'eau

Le programme ci-dessus permet d'initialiser les ports ainsi que de récupérer les valeurs dans la variable "niveau".

Capteur Anémomètre

L'anémomètre est un capteur dédié à la mesure de la vitesse du vent. Celui-ci délivre à plusieurs reprises des signaux qui permet au programme de compter le nombre de tours.

L'anémomètre est branché à l'entrée analogique "*P_ANALOG1*". Nous récupérons une valeur à l'état haut ou à l'état bas. Elle va nous servir à compter le nombre de tours.

Une girouette est accrochée au capteur, qui délivre une tension en continu variant en fonction de l'axe (de 0v à 5 v).

figure 5 : Anémomètre

Le capteur d'humidité

Comme le dit son nom, ce capteur nous permet de connaître le pourcentage d'humidité dans l'air. Il fonctionne en digital, nous le branchons donc sur la broche *P4.29* prévu pour ce fonctionnement.

Afin de faire fonctionner ce capteur, il faut tout d'abord lui envoyer une tension sur son pin de data, nous configurons donc tout d'abord notre broche en tant qu'entrée. Une fois la tension envoyée, il faut immédiatement configurer la broche en sortie afin de lire ce que nous renvoie le capteur.

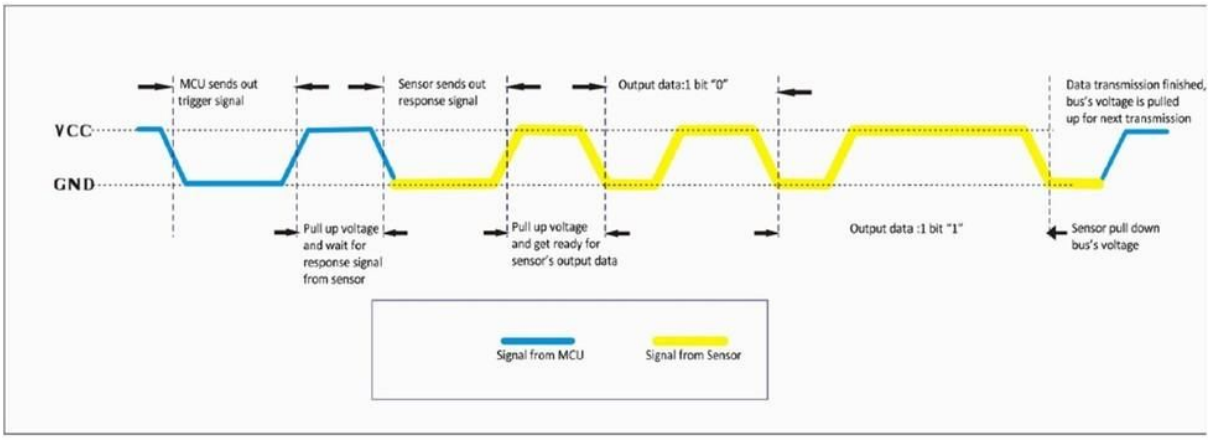

figure 6 : signal d'émission et réception du capteur d'humidité

Ce capteur nous renvoie une trame qui commence toujours l'enchaînement suivant: haut, bas, haut, bas. La suite de la trame est aussi constituée d'une succession d'état haut et d'état bas, mais ce qui nous intéresse ici est la durée d'un état haut (court ou long). Si un état haut à une longue durée, cela veut dire que le capteur nous envoie un bit à 1, par conséquent si la durée est courte elle cela correspond à un 0. Seulement les 16 premiers bits nous intéressent pour l'humidité, les 24 bits restant correspondent à la température (que l'on récupère à l'aide d'un autre capteur) et au checksum.

Capteur baromètre/température

Ce capteur va nous permettre d'avoir 2 informations, il nous délivre la valeur de la température ambiante ainsi que la valeur de la pression atmosphérique. Il fonctionne en *I2C*, nous le branchons sur les port *P0.27* et *P0.28* de la carte Keil qui correspondent à *I2C0*.

Afin de recevoir les différentes valeurs du capteur nous avons besoins de l'initialiser, pour effectuer cela, on envoie une trame *I2C* au capteurs d'adresse *0x76*, cette trame contient le commande *0x48*.

Une fois que le capteur est initialisé, on vient lui demander de nous envoyer les valeur de pression/température. Pour lire une valeur en *I2C* il faut d'abord envoyer une requête au capteur.

```
Driver_I2C0.MasterTransmit(SLAVE_I2C_ADDR,tabEnv,1,false);
while (Driver_I2C0.GetStatus().busy == 1);
Driver_I2C0.MasterReceive(SLAVE_I2C_ADDR,I2C,6,false); //Re
while (Driver_I2C0.GetStatus().busy == 1);
```

figure 7 : échange I2C pour récupérer la donnée de température

Pour faire la requête au capteur on lui envoie la commande *0x10*, suite à cette commande le capteur va nous renvoyer 6 octets (3 octets pour la pression et 3 octets pour la température). On vient donc lire le capteur est stocker les 6 octets envoyé dans un tableau (ici *I2C*), ce tableau est rempli du MSB de la température jusqu'au LSB de la pression.

```
temperature = I2C[2];
temperature += I2C[1]<<8;
temperature += I2C[0]<<16;
temperature /= 10;
```

```
pression = I2C[5];
pression += I2C[4]<<8;
pression += I2C[3]<<16;
pression /= 10;
```

Il nous reste plus qu'à mettre les valeurs du tableau dans 2 variables distinctes.

Partie 2 : Communication entre cartes

Une fois toutes les informations récupérés, on vient les ajouter dans une chaîne de caractère. Nous transmettons cette chaîne de caractère par liaison Bluetooth via UART de la carte Keil à la carte Raspberry Pi. Sur la carte Keil, il s'agit d'un module Bluetooth maître qui y est connecté, ici il est configuré à 9600 bauds.

```
32  //------------------INITIALISATION UART-------------//
33  void Init_UART(void){
34          Driver_USART1.Initialize(NULL);
35          Driver_USART1.PowerControl(ARM_POWER_FULL);
36          Driver_USART1.Control( ARM_USART_MODE_ASYNCHRONOUS |
37          ARM_USART_DATA_BITS_8 |
38          ARM_USART_STOP_BITS_1 |
39          ARM_USART_PARITY_NONE |
40          ARM_USART_FLOW_CONTROL_NONE,
41          9600);
42          Driver_USART1.Control(ARM_USART_CONTROL_TX,1);
43          Driver_USART1.Control(ARM_USART_CONTROL_RX,1);
44  }
```

figure 8 : initialisation de l'UART sur la carte Keil

Nous réglons donc les paramètres de l'UART de la carte Keil à 9600 bauds.

```
sprintf(trame,"#%3d%3d%5d%3d%3d",temperature,humidite,pression,vent,niveau_eau);
GLCD_DrawString(10,150,trame);
while(Driver_USART1.GetStatus().tx_busy == 1);
Driver_USART1.Send(trame,18);
```

figure 9:code d'envoie de la chaîne Bluetooth

Cette chaîne est constitué de 18 caractères, elle commencera toujours avec le symbole '#' comme premier caractère suivis des valeurs des différents capteurs, dans l'ordre suivant : Température, Humidité, Pression, Vitesse du vent et Niveau d'eau.

Passons maintenant du côté de la réception, la Raspberry Pi, pour récupérer cette chaîne de caractère nous branchons un module Bluetooth esclave à cette carte.
Le programme de réception de trame se fait aussi en langage C.
Le but de ce programme est de recevoir en continue la chaîne ci-dessus afin de la sauvegarder dans un fichier texte qui sera utile pour la page web.
Etant donné que la Raspberry n'utilise pas les même drivers que la carte Keil, nous utilisons une méthode différente de celle utilisé pour l'envoie afin d'effectuer la réception.

```
20    //---------------Ouverture du port faisant la réception Bluetooth---------------/
21    fd = open("/dev/serial0", O_RDWR | O_NOCTTY);
22    if (fd == -1) {
23      perror("open_port: Unable to open /dev/ttyAMA0 - ");
24      return(-1);
25    }
26
27                  struct termios options;
28                          tcgetattr(fd, &options);
29                          options.c_cflag = B115200 | CS8 | CLOCAL | CREAD;               //<Set baud rate
30                          options.c_iflag = IGNPAR;
31                          options.c_oflag = 0;
32                          options.c_lflag = 0;
33                          options.c_cc[VMIN]=18;
34                          tcflush(fd, TCIFLUSH);
35                          tcsetattr(fd, TCSANOW, &options);
36
37    // Turn off blocking for reads, use (fd, F_SETFL, FNDELAY) if you want that
38    fcntl(fd, F_SETFL, 0);
```

figure 10: initialisation de l'UART sur la Raspberry Pi

Tout d'abord, nous ouvrons le port permettant d'effectuer la transmission Bluetooth et nous définissons les différents paramètres UART de la carte Raspberry Pi, ici on vient brancher le module Bluetooth esclave au niveau de Serial0 sur la carte. Contrairement au module Bluetooth maître, celui-ci ne fonctionne pas en 9600 bauds mais en 115200 bauds.

```
//--------------------Lecture de la trame reçu en Bluetooth--------------------/
n = read(fd, (void*) buf, sizeof(buf));
if (n < 0) {
        perror("Read failed - \n");
        return -1;
                        }
else if (n == 0)
        printf("No data on port\n");

else
{
        buf[n] = '\0';
}
```

figure 11: code de réception de la trame Bluetooth

Une fois le port initialisé, nous venons le lire et stocker ce qu'il que nous recevons dans un tableau, il s'agit de *buf* ici. Cependant, le tableau ne se rempli pas tout le temps dans l'ordre des caractères envoyé.

```
//---------------------------------Triage du tableau---------------------------------/

for(i = 0;i<n;i++)
{
        if(buf[i]=='#')
        {
                k=i;
                for(j=0;j<n;j++)
                {

                        if(k==18)
                        {k=0;}
                        tab[j]=buf[k];
                        k++;
                }
        }
}
```

figure 12: tri de la trame reçue

Pour régler ce problème nous effectuons un algorithme afin de trier le tableau et donc avoir nos valeurs dans le bon ordre. C'est à ce moment que le caractère '*#*' utilisé dans la trame d'envoie à son importance, en effet, nous venons scruter *tab* à la recherche de ce caractère. Une fois ce caractère trouvé, nous le copions dans un nouveau tableau de même taille ainsi que les caractère qui suivent afin d'avoir le tableau rempli dans le bon ordre.

```
//---------------------Création des différentes chaîne des caractères---------------------/

sprintf(tabtp,"%c%c,%c ",tab[1],tab[2],tab[3]);
sprintf(tabhum,"%c%c%c ",tab[4],tab[5],tab[6]);
sprintf(tabpres,"%c%c%c%c,%c ",tab[7],tab[8],tab[9],tab[10],tab[11]);
sprintf(tabvent,"%c%c%c ",tab[12],tab[13],tab[14]);
sprintf(tabeau,"%c%c,%c ",tab[15],tab[16],tab[17]);

fdp=fopen ("/var/www/html/donnees.txt","w+"); //Création du document texte

//------------------Remplissage du document texte avec les valeurs reçu------------------/

if (fdp != NULL)
{
        printf("%c",tabvent[0]);
        printf ("ouvre\n");
        fputs("Température : ",fdp);
        fputs(tabtp,fdp);
        fputs("°C \nHumidité : ",fdp);
        fputs(tabhum,fdp);
        fputs("% \nPression : ",fdp);
        fputs(tabpres,fdp);
        fputs("hPa \nVent : ",fdp);
        fputs(tabvent,fdp);
        fputs("km/h \nPrécipitation : ",fdp);
        fputs(tabeau,fdp);
        fputs("mm \n",fdp);
}

else
{
        printf ("ouvre pas\n");
}
```

figure 13: transformation de la trame reçue en plusieurs trames

On vient ensuite mettre les différentes valeurs dans différentes chaînes de caractères afin de remplir notre fichier texte.

Partie 3 : Interface serveur web

La partie interface du serveur web de la Raspberry permet donc d'afficher sur une page web les données reçue. Le serveur web *Apache* est hébergé sur la Raspberry Pi. Cette page sera consultable depuis chaque PC présent dans le même sous réseau que la Raspberry.
Le code de la page web est donc réalisé avec 3 langages : html, css et php.

```
<style>


body {
    background-image: url("Volcano.jpg"); /*change image background*/

}

h1 {
    font-size: 56px; /*taille de police*/
    text-align: center; /*poistionnement du texte*/
    color : red; /*couleur du texte*/
    font-family: Papyrus; /*style de police du texte*/
    text-shadow: #000000 1px 1px, #000000 -1px 1px, #000000 -1px -1px, #000000 1px -1px;/* application d'effet d'ombre sur le texte*/
    }

p{
    color: white;
    font-size: 24px;
    text-align:bottom;
    position:absolute;/* permet de positionner plus precisement le texte*/
    right:10px;
    bottom:10px;
    text-shadow: #000000 1px 1px, #000000 -1px 1px, #000000 -1px -1px, #000000 1px -1px;
}
.text{
    font-size: 52px;
    font-family: impact;
    text-shadow: #000000 1px 1px, #000000 -1px 1px, #000000 -1px -1px, #000000 1px -1px;
}
.bas{
    position: absolute;
    left: 1%;
    bottom: 1%;
}
.vroum{
    position: absolute;
    left: 47%;
    bottom: 0%;
}

</style>
```

figure 14 : code css de la page web

Sur la figure ci-dessus, on peut voir le code css, compris entre les deux balises html <style> et </style>, de la page qui sert à positionner les textes, changer leurs polices et leurs couleurs. Le css permet aussi de mettre un background à la page.

```
<body>


    <h1>Station meteo de l'Apocalypse LPRO SESAM 2017/2018</h1><!--met un titre sur la page web-->
    <p2 class="text">
    <div align="center">

    <?php

        $file=fopen("donnees.txt","r");//ouvre le fichier texte et donne les droits de lecture
        if($file)//si on reussi à ouvrir le fichier
        {
            while (($line = fgets($file,4096))!== false)//on vérifie si on a atteint la fin du fichier
            {
                echo '<span style ="color:green;">'.$line.'</span>';//affichage du texte
                echo '<br>';//retour à la ligne
            }

            fclose($file);//fermeture du fichier
        }
        ?>
    </p2>
</div>
<div2 class="bas">
<img src="Logo_IUT.png" width="200" height="200"><!--place une image sur la page web-->
</div2>
<div3 class="vroum">
<img src="Vroum_arnaud.png" width="250" height="300">
</div3>
<p>réalisé par Dorian Lamarche, Arnaud Wattebled et Florian Zanin</p>
</body>
```

figure 15 : code html de la partie body

La partie body du code html permet de structurer la page web en plaçant le titre, les paragraphes et les images.

```php
<?php

    $file=fopen("donnees.txt","r");//ouvre le fichier texte et donne les droits de lecture
    if($file)//si on reussi à ouvrir le fichier
    {
        while (($line = fgets($file,4096))!== false)//on vérifie si on a atteint la fin du fichier
        {
            echo '<span style ="color:green;">'.$line.'</span>';//affichage du texte
            echo '<br>';//retour à la ligne
        }

        fclose($file);//fermeture du fichier
    }
    ?>
```

figure 16 : code php

Le code php va donc permettre d'ouvrir le fichier texte et va afficher le contenue de ce fichier texte sur le serveur web. On utilise les balises <?php et ?>.