14/11/2018

Florent Garrit - Henri Roques

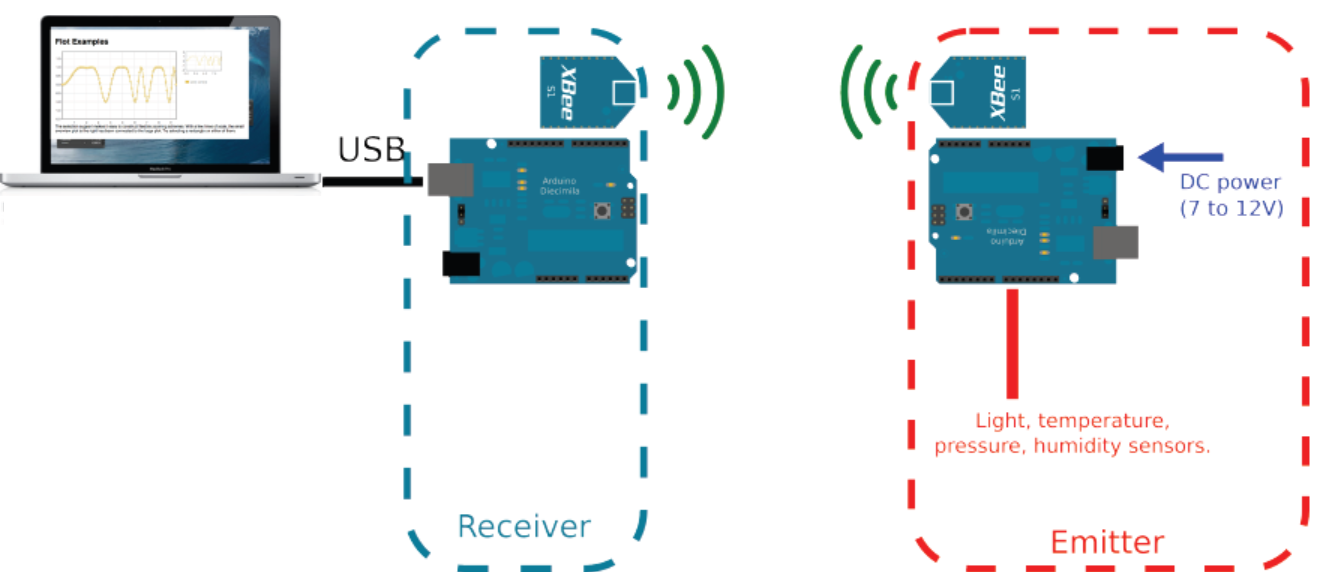
SEE08

EN326 – Capteurs pour l’embarqué

# Introduction

L’objectif de ce projet est de réaliser une station météo sans fil. Elle sera composée d’une partie émission distante munie de différents capteurs, et d’une partie réception locale reliée à un ordinateur.

L’architecture retenu imposée par le sujet est la suivante :



Chaque module se compose :

* d’une carte Arduino UNO avec microprocesseur ATMEGA328P ;
* d’un shield Wireless Proto ou SD ;
* d’un module xBee.

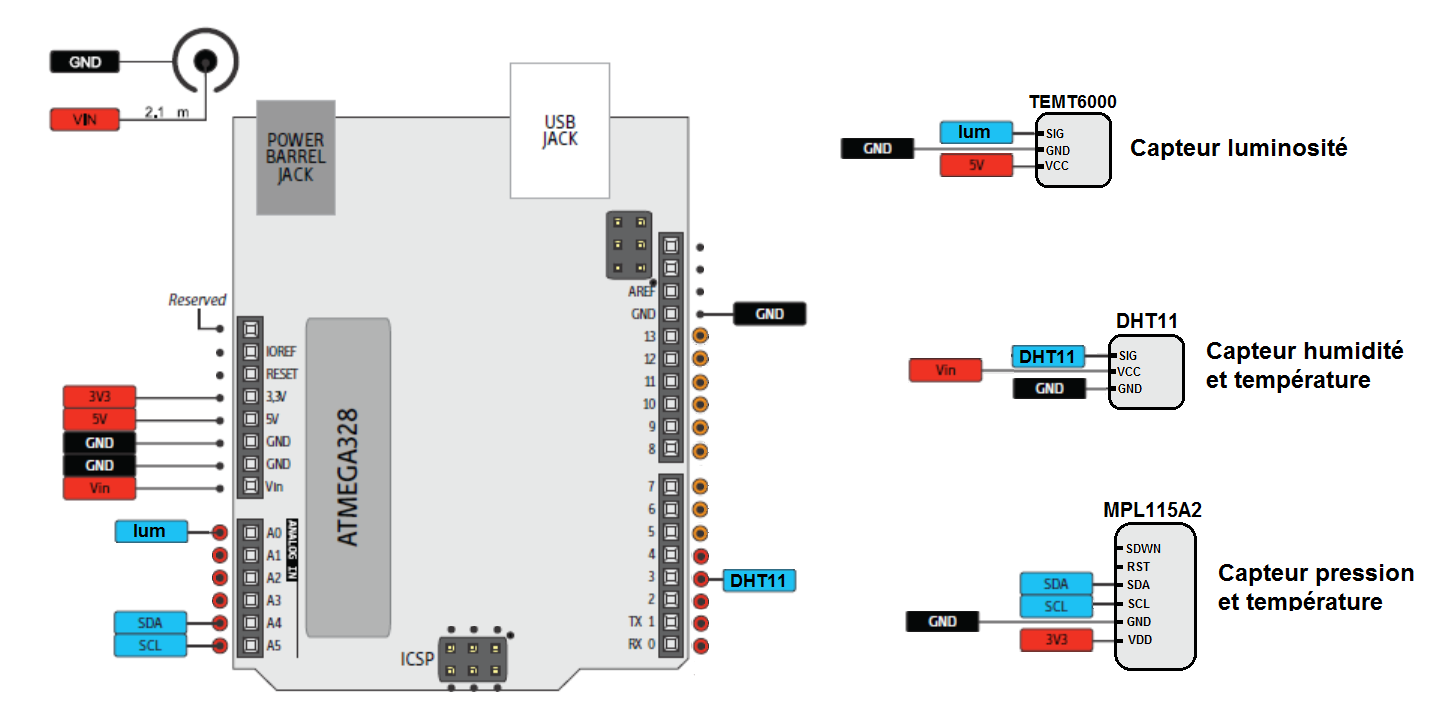
Les capteurs disponibles sont :

* un luxmètre Vishay TEMT6000 ;
* un baromètre/thermomètre Freescale MPL115A2 ;
* un hygromètre/thermomètre D-Robotics DHT11.

Chacun des capteurs utilise un protocole de communication différent. Un des objectifs est donc d’implémenter les codes dans le microcontrôleur pour acquérir les mesures correctement.

# Architecture globale

## Câblage carte



## Capteur de luminosité Vishay TEMPT6000

Le capteur de luminosité n’utilise pas de protocole de communication particulier. Il se compose d’un phototransistor couplé avec une résistance de 10kΩ qui nous renvoie une tension image de la luminosité captée par le phototransistor.

D’après le schéma du capteur associé à sa résistance on peut en déduire la valeur de la tension qu’il nous renvoie en fonction de la luminosité.  
La datasheet du capteur TEMPT6000 nous indique la loi suivante :

Figure 1 - Schéma TEMPT6000

On peut ainsi en déduire la valeur de la tension en entrée du CAN de la carte arduino :

Le CAN de la carte Arduino converti une tension 0/5V en une valeur numérique 0/1023. On peut donc en déduire la loi finale entre la valeur numérique reçu et la luminosité réelle :

Nous avons donc créé une fonction « float lecture\_lum(void) » qui vient lire la valeur du signal issu du capteur de luminosité connecté à la broche A0 de la carte Arduino et qui retourne la valeur de la luminosité réelle en lux.

Cette fonction est appelée dans notre main pour stocker la valeur de la luminosité dans la variable « luminosite ».

## Capteur de température et d’humidité Sensirion SHT15

## Capteur de température et de pression Freescale MPL115A2

### Présentation du capteur et de ses connexions

Le baromètre-thermomètre MPL11A52 communique grâce à une liaison I²C.

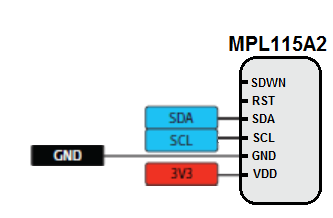
Il a une consommation faible : environ 5µA en fonctionnement. D’après la documentation technique il peut être alimenté en 3V3, nous le connecterons donc à la broche 3V3 de l’arduino pour VDD.

Le brochage de la carte portant le capteur est le suivant :

|  |  |  |
| --- | --- | --- |
| 1 | SDWN | *Shutdown* : connecter au GND pour mettre le capteur en veille. |
| 2 | RST | *Reset* : connecter au GND pour décastiver la communication I²C |
| 3 | SDA | *Serial Data I/O line* |
| 4 | SCL | *I²C Serial Clock Input* |
| 5 | GND | Ground |
| 6 | VDD | Broche d'alimentation générale |

RST et SDWN sont laissés non connectés car inutiles dans le cadre de notre projet.

Les connexions avec la carte arduino sont les suivantes :

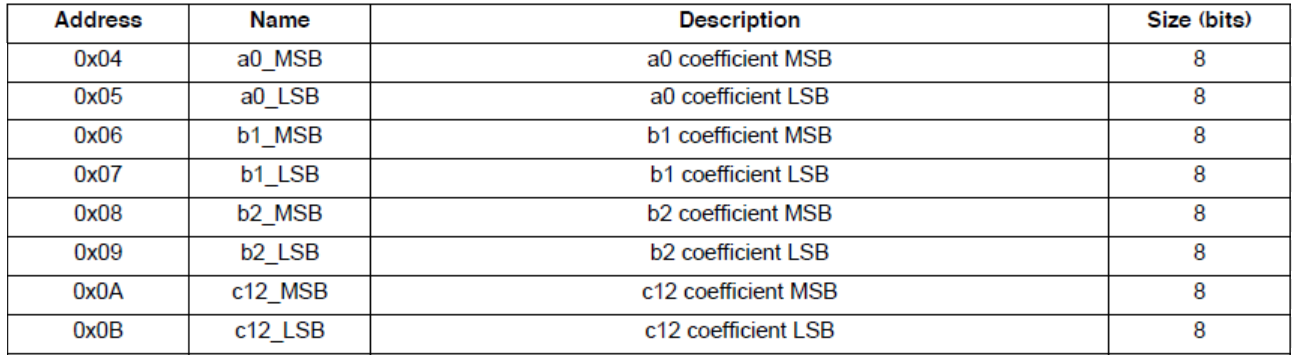


### Récupération des paramètres de correction

La calibration du capteur est faite en usine et les paramètres de corrections sont entrés en dur dans le capteur. Il suffit de venir lire ces paramètres au démarrage de la carte pour pourvoir les utilise ensuite.

Cette récupération est donc décrite dans la partie setup du code.

Il y a 4 paramètres : a0, b1, b2 et c12. Chacun est découpé en 2 parties de 8bits :



Le paramétrage de la connexion est le suivant :

Wire.begin(); //join i2c bus

**Serial**.begin(9600); //start serial for output

Wire.beginTransmission(0x60); //début transmission adresse esclave

Wire.write(0x04); //placement du curseur sur adresse 1er coeff

Wire.endTransmission(); //fin écriture

Wire.requestFrom(0x60, 8); //8 octets demandés

On récupère les trames octet par octet.

On devra donc regrouper les paramètres dans des variables de 16bits. Pour cela nous utilisons des structures dont le modèle est le suivant :

union {

int ia0;

char ca0[2];

} parametreA0 ;

Pour a0, b1 et b2 on récupère les données de la même façon :

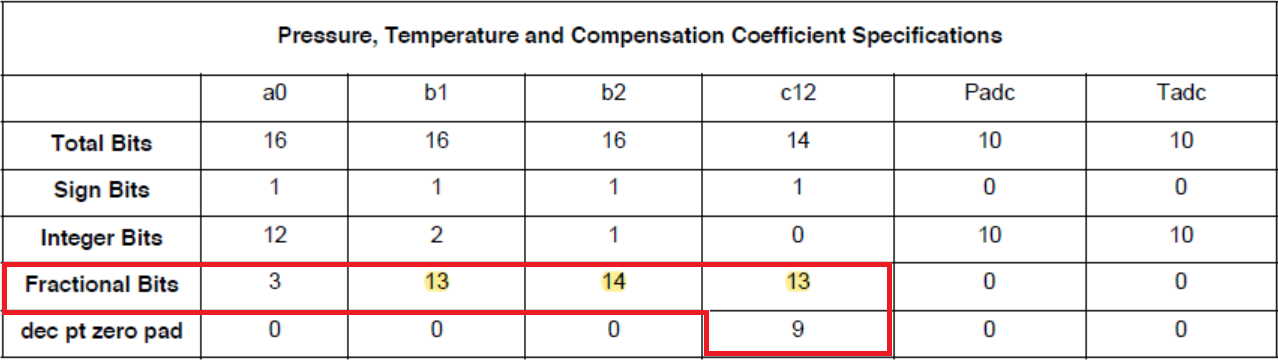
float fA0; //variable globale déclarée hors fonction

parametreA0.ca0[1] = Wire.read(); //lecture trame MSB

parametreA0.ca0[0] = Wire.read(); //lecture trame LSB

fA0 = (float)parametreA0.ia0 / X ; //division par 2^?

La division permet de décaler la virgule du flottant au bon endroit, pour déterminer X on doit se référer à la datasheet :



Pour a1 : X = 2^3 = 8.

Pour b1 : X = 2^13 = 8190.

Pour b2 X = 2^14 = 16384.

Pour c12 la récupération est un peu différente en raison de la longueur totale de 14 bits et non de 16 bits comme les 3 paramètres précédents, un décalage de 2 bits vers la droite est nécessaire :

float fC12; //variable globale déclarée hors fonction

parametreC12.cc12[1] = Wire.read(); //lecture trame MSB

parametreC12.cc12[0] = Wire.read(); //lecture trame LSB

parametreC12.ic12 = (parametreC12.ic12) >> 2; //décalage de 2bits

fC12 = (float)parametreC12.ic12 / 4194304.0; //div par 2^(13+9)

Notons que le décalage de virgule prend en compte les 9 bits de padding.

### Démarrage de la conversion (ADC)

Il faut signifier au capteur qu’il doit démarrer la conversion des valeurs dans l’ADC.

Cette partie est décrite au début de la fonction getPT() .

Pour cela on envoie la commande suivante :

Wire.beginTransmission(0x60); //adresse du capteur : 0x60

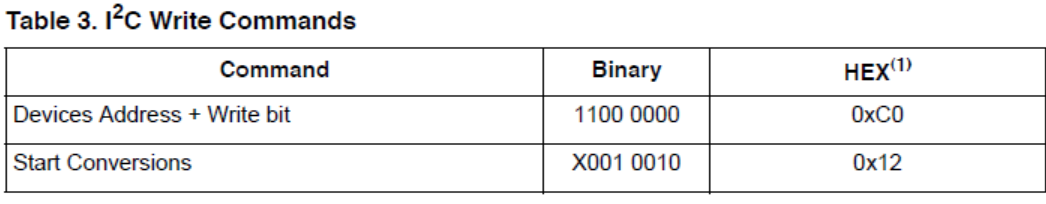
Wire.write(0X12); //start conversion ADC

Wire.write(0x00); //adresse MSB pression

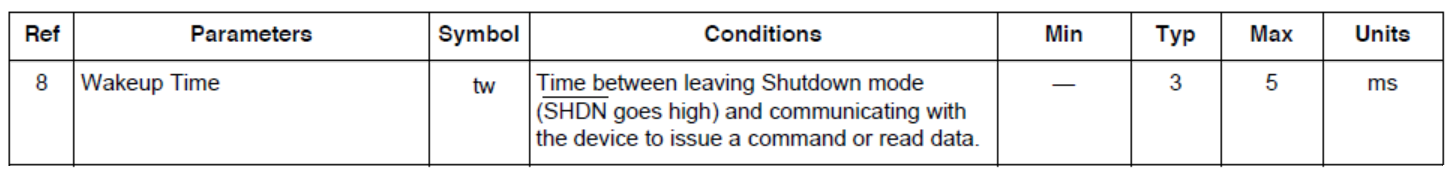
Wire.endTransmission();

delay(5); //attente de la fin de la conversion

La commande correspondant au signal « start conversion » est donnée dans la documentation :



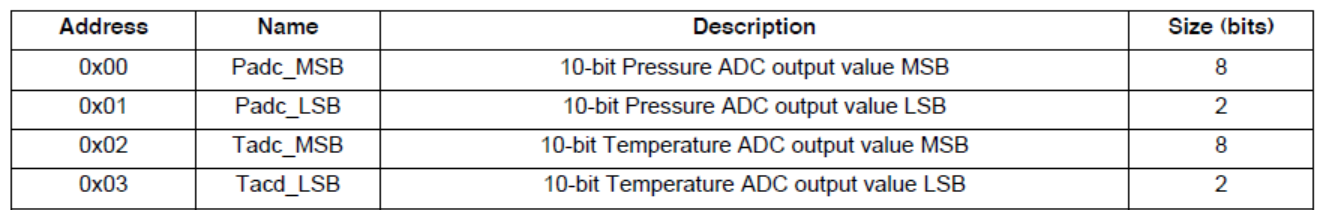
Le délai d’attente est nécessaire car la conversion n’est pas instantanée. On trouve dans la datasheet les durées moyenne et maxi :



### Acquisition des données brutes (pression et température)

L’ADC a maintenant rempli les registres, on peut donc venir lire les valeurs brutes de pression et de température.

On veut lire les 4 registres suivants :



Les deux grandeurs physiques sont sur 10 bits. Il faut donc acquérir chaque trame de 8 bits et réunir les MSB et LSB pour la pression et pour la température. Cette partie est décrite dans la suite de la fonction getPT() :

float pres, temp; //variables locales

Wire.beginTransmission(0X60); //adresse du capteur : 0x60

Wire.write(0x00);

Wire.endTransmission();

Wire.requestFrom(0x60, 4); //4 registres à partir de l'adresse 0x06

pres = (( Wire.read() << 8) | (Wire.read())) >> 6; //MSB=8b LSB=2b

temp = (( Wire.read() << 8) | (Wire.read())) >> 6; //MSB=8b LSB=2b

### Correction des données

Les données sont corrigées grâce aux paramètres précédemment acquis.

Le calcul est donné dans la documentation du capteur.

Cette partie est également décrite dans la fonction getPT().

float pressureComp; //variable locale

pressureComp = fA0 + (fB1 + fC12 \* temp ) \* pres + fB2 \* temp;

Il faut maintenant convertir la pression en kPa et la température en °C.

float P, T; //variable globale déclarée hors fonction

P = ((65.0 / 1023.0) \* pressureComp) + 50.0; // kPa

T = ((float) temp - 498.0) / -5.35 + 25.0; // °C

# Interface Homme-Machine