# Projet SEE – Capteurs pour l'embarqué

# Station météo sans-fil

## 1 – Présentation globale

L'objectif de ce projet SEE 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. À l'issu des séances, vous devrez être en mesure de présenter un système capable d'afficher sur l'écran de l'ordinateur mis à disposition les valeurs instantanées des capteurs embarqués.

L'architecture retenue pour cette station est présentée Figure 1.

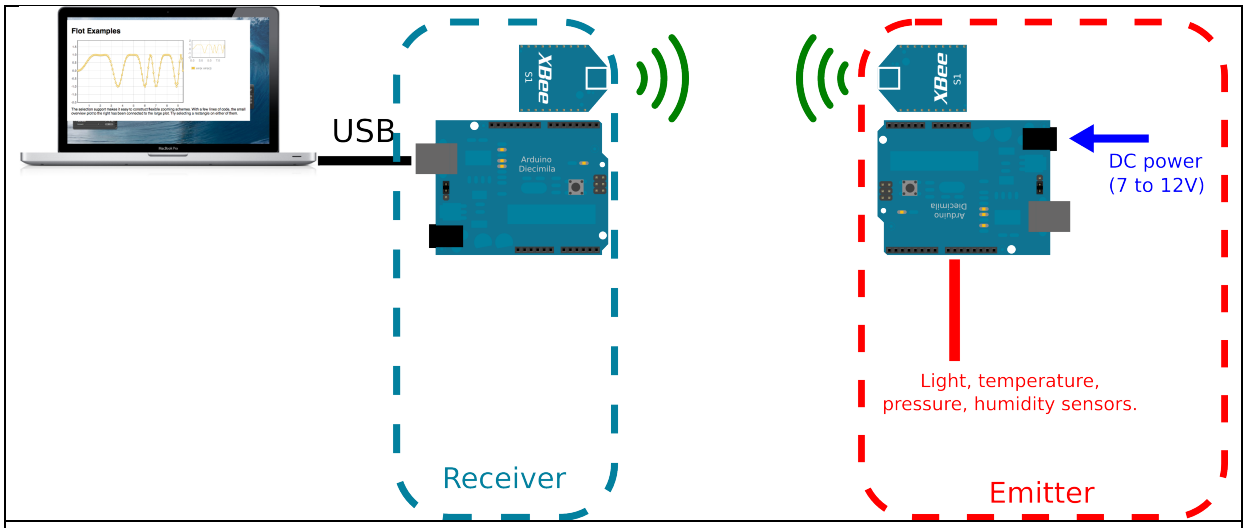


Figure 1 - Architecture de la station météo

Chaque module se compose :

- D'une carte Arduino UNO avec microcontroleur ATMEGA328P.
- D'un shield Wireless Proto ou SD.
- D'un module xBee.

Le module récepteur doit être relié au PC par USB et tirera son alimentation directement par le port. Le module émetteur sera alimenté par une alimentation stabilisée externe (tension d'entrée de 7 à 12V). C'est ce dernier qui embarquera les capteurs.

## 2 – Capteurs mis à disposition

Vous disposerez de trois capteurs dont les réponses devront être affichées sur le PC par l'intermédiaire d'une liaison sans fil. Pour chacun de ces capteurs, la datasheet est mise à votre disposition : vous y trouverez les informations nécessaires pour obtenir les mesures de ces capteurs.

- Un capteur de luminosité Vishay TEMT6000.
- Un capteur de pression Freescale MPL115A2, donnant également une mesure de température.
- Un capteur d'humidité/température Sensirion SHT15 ou RHT03.

Chacun de ces capteurs utilise un protocole différent pour communiquer avec son environnement. Étudiez attentivement les datasheets pour trouver les protocoles à mettre en œuvre et programmer le microcontrôleur en conséquence.

## 3 – Arduino, Processing et XBee

Pour mener ce projet à bien, vous aurez à programmer les cartes Arduino UNO afin d'interagir avec les capteurs. La programmation se fera par le biais de l'environnement Arduino (langage proche du C). Les exemples contenus dans le logiciel permettront aux étudiants non expérimentés de progresser rapidement dans cet environnement. Au vu des objectifs de ce TP, il vous est demandé de ne pas avoir recours à des librairies extérieures sur la partie Arduino.

Il vous est également demandé de développer l'Interface Homme Machine, sous l'environnement Processing. Ce logiciel vous permettra de créer rapidement un affichage graphique des données issues des capteurs. Là encore les exemples fournis avec le logiciel serviront de base pour appréhender cet environnement, dont le langage est basé sur le Java. Pour vous aider sur la partie tracé de graphique, la librairie giCentre Utils sera mise à votre disposition.

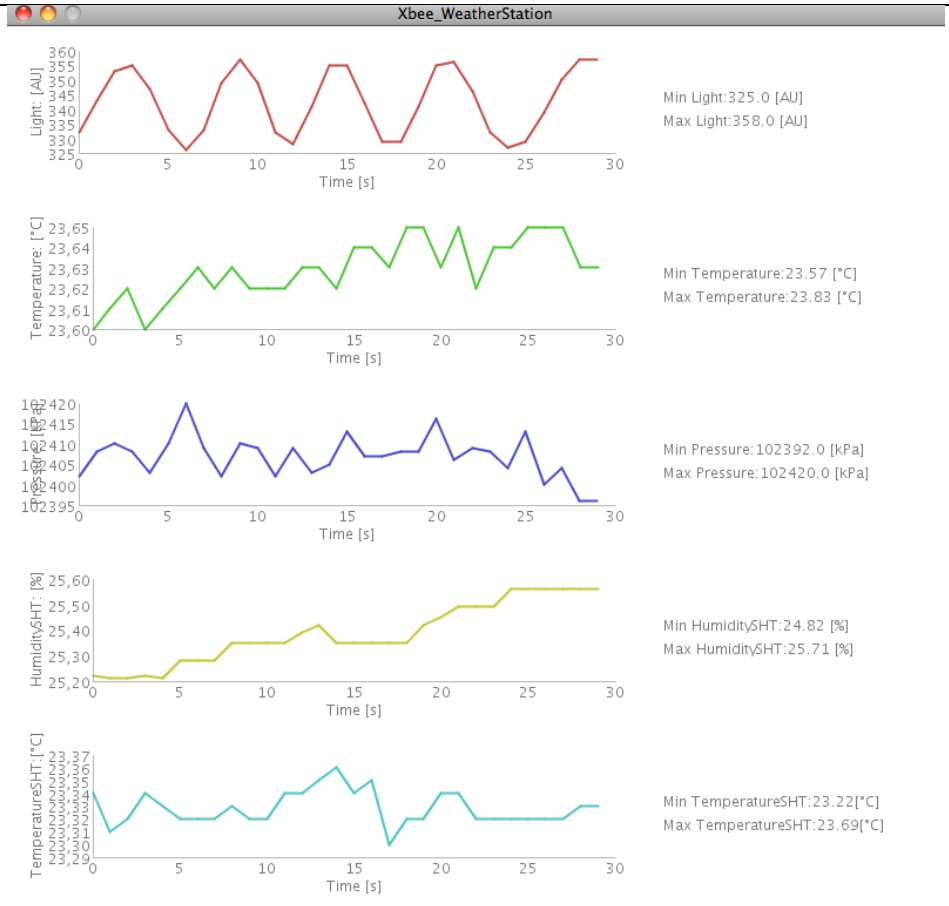


Figure 2 – Exemple d'affichage graphique des mesures.

### 3.1 – Arduino UNO

Le projet Arduino a été développé afin de démocratiser l'utilisation d'un microcontrôleur. C'est un projet Open Source et Open Hardware, qui grâce à des fonctions avancées permet à des personnes non formées à la programmation de pouvoir rapidement effectuer des taches telles que le pilotage d'entrées/sorties, l'acquisition de signaux analogiques, la communication vers des périphériques par le biais de bus TWI/**I2C** ou SPI, ou vers l'ordinateur par le biais d'une liaison série… Le nom Arduino associe un Environnement de Développement Intégré (EDI) et des cartes microcontrôleurs dotées de caractéristiques spécifiques.

La programmation s'effectue en langage C/C++, et malgré la couche d'abstraction rajoutée pour simplifier le codage pour les novices, les fonctions avancées du microcontrôleur sont accessibles. Par exemple, pour mettre la broche 8 de l'Arduino UNO au niveau logique 1 (=5V), les deux méthodes suivantes sont valides :

```
// Configure digital pin
pinMode(8, OUTPUT);           // Sets pin 8 as an output.
digitalWrite(8, HIGH);        // Sets pin 8 level high.
...
...
DDRB = DDRB|B00000001;        // Sets PB0 (Arduino UNO pin 8) as an output.
PORTB = B00000001;            // Sets PB0 (Arduino UNO pin 8) level high.
```

Code 1 - Configuration de la broche 8 en sortie, puis écriture d'un 1 logique.

Il vous appartient dans ce projet de choisir les méthodes qui vous sont le plus familières.

La structure de base d'un programme Arduino comporte deux fonctions, nommées *setup()* et *loop()*. La fonction *setup()* est appelée une seule fois à l'exécution du code ; la fonction *loop()* est une boucle infinie qui est appelée à la suite de la fonction *setup()*. C'est la fonction *loop()* qui contient généralement le programme principal.

```
int ledPin = 13;                // A LED is connected to pin 13 on the Arduino UNO.

// Executed once on startup.
void setup(){
        // Initialisation.
        pinMode(ledPin, OUTPUT);        // Set pin 13 as output.
}

// Infinite loop.
void loop(){
        // Main program.
        digitalWrite(ledPin, HIGH);     // Set pin 13 level high.
        delay(1000);                    // Wait for 1000 ms.
        digitalWrite(ledPin, LOW);      // Set pin 13 level low.
        delay(1000);                    // Wait for 1000 ms.
}
```

Code 2 – Structure d'un programme dans l'EDI Arduino.

L'environnement Arduino n'inclut pas de débogueur, mais il possède un moniteur série, qui bien que basique permet en adaptant son code de faire du débogage en commentant les portions de codes. Il vous permettra également de vérifier les paquets de données que vous ferez transitez sur le port série.

Le fichier « *Arduino_uno_Pinout.pdf* » représenté Figure 2 permet de faire la correspondance entre les broches du microcontrôleur ATMEGA328P et les broches accessibles sur la carte Arduino UNO. Ce document permet également de repérer les broches possédant des fonctions spécifiques comme les sorties PWM, des ports de communication **I2C** ou autres. Prêtez attention aux fonctionnalités de ces broches afin d'utiliser les broches adapter lorsque vous câblerez vos capteurs.

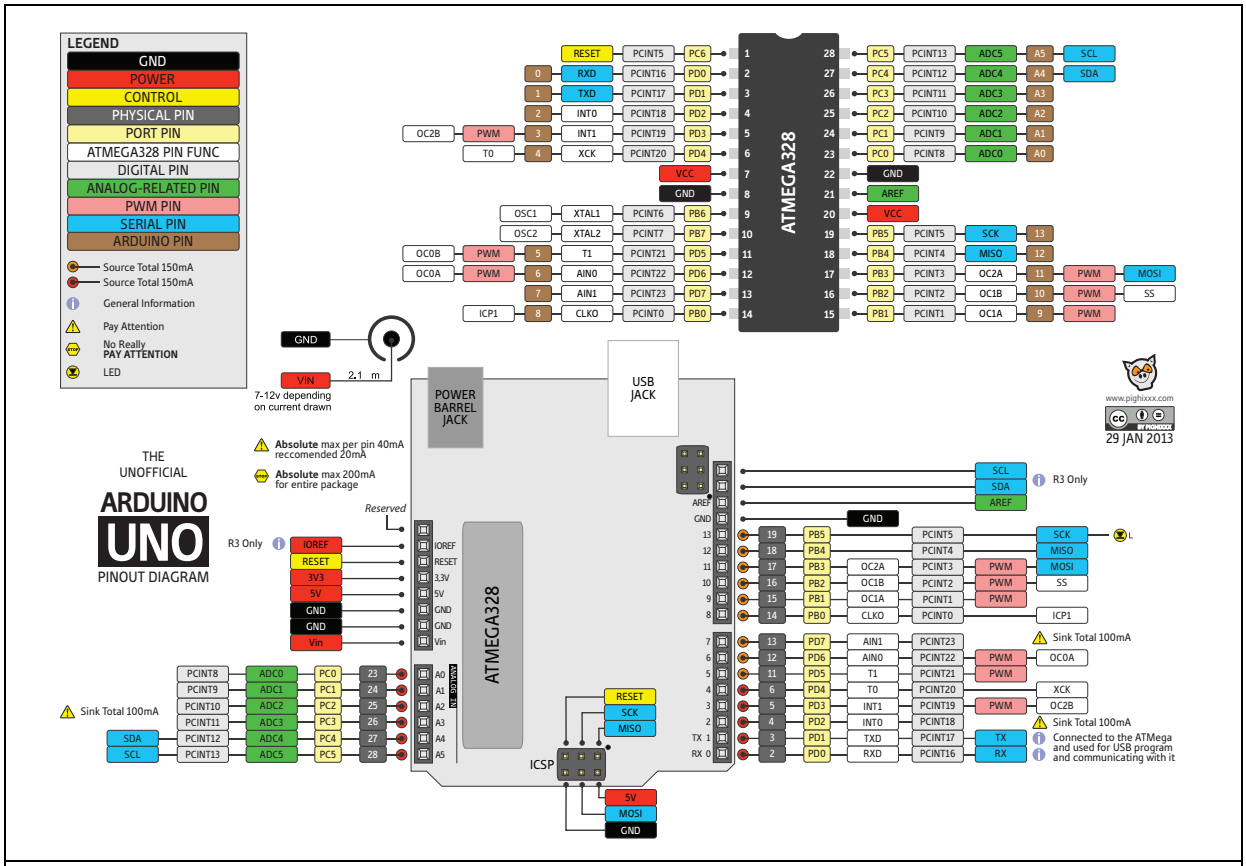


Figure 2 – Brochage de l'Arduino UNO.

### 3.2 – Environnement Processing

Cet Environnement de Développement Intégré (EDI) permet de créer de manière simplifiée des dessins en 2D et 3D à partir d'un langage proche du Java. C'est cet EDI qui a servi de base à l'EDI Arduino, ce qui explique leurs ressemblances en termes d'agencement.

La structure d'un programme Processing possède également des similitudes avec un programme Arduino, avec une fonction d'initialisation appelée au lancement du programme et nommée *setup()*, et une fonction principale qui est une boucle infinie nommée *draw()*.

```
// Executed once on startup.
void setup(){
        // Initialisation.
        size(400, 300); // Main window size in pixels.
background(0);          // Black background.
}

// Infinite loop.
void draw(){
        // Main program.
        text(« Hello World », 100, 100);
}
```

Code 3 – Structure d'un programme dans l'EDI Processing.

L'environnement Processing va vous permettre de réaliser l'affichage des données issues des capteurs. Le flux des données des capteurs sera le suivant :

- Acquisition des données capteurs sur la carte émettrice.
- Transmission RF des données de la carte émettrice vers la carte réceptrice.
- Réception des données par RF depuis la carte réceptrice.
- Transmission série des données de la carte réceptrice vers le PC pour traitement par Processing.

Il est possible de créer à partir du jeu d'instruction de base de Processing des tracés graphiques en 2D. Il est toutefois plus facile d'utiliser une librairie adaptée pour réaliser cette opération. La librairie

« *giCentre Utils* » sera ainsi mise à votre disposition pour effectuer les tracés grâce à la classe « *XYChart* ». Vous trouverez ci-dessous un exemple de code utilisant cette classe.

```
import org.gicentre.utils.stat.*; // Import the gicentre utils stat library for chart classes.

XYChart myXYchart;                      // Declare an XYChart object.
float[] xData = new float[360];          // x data table.
float[] yData = new float[360];          // y data table.

void setup() {
  size(600, 400);
  noLoop();        // Perform the loop() just once.

  // XY chart data initialisation.
  for(int i=0; i<xData.length; i++) {
    xData[i]=i;
    yData[i]=sin(radians(xData[i]));// The sin() function takes angles in radians as an input.
  }

  myXYchart = new XYChart(this);      // Create an XYChart object.
  myXYchart.setData(xData, yData);    // Assign data to the XYChart object.

  // Chart parameters settings
  myXYchart.setPointSize(0);
  myXYchart.setLineWidth(2);
  myXYchart.setLineColour(color(200,80,80));

  myXYchart.showXAxis(true);
  myXYchart.setXAxisLabel("X axis");
  myXYchart.showYAxis(true);
  myXYchart.setYAxisLabel("Sin(X)");
  myXYchart.setMaxY(1.5);
  myXYchart.setMinY(-1.5);
}

void draw() {
  background(255);

  myXYchart.draw(10,10,width-20, height-20);

  // Draw a title over the top of the chart.
  fill(120);
  textSize(20);
  text("Example of a sinus plot using giCentre XYChart", 70,30);
  textSize(11);
  text("Help files are in the Processing/libraries/gicentreUtils/reference folder", 70,45);
}
```

Code 4 - Exemple de code utilisant la librairie *giCentre utils* et la classe associée *XYChart*.

Vous pourrez toutefois si vous le souhaitez utiliser les fonctions plus bas niveau de Processing.

### 3.3 – Modules XBee

La communication sans fil entre les deux cartes Arduino UNO mises à votre disposition va s'effectuer grâce à des modules XBee de chez Digi International. Ces modules radio sont basés sur le standard IEEE 802.15.4, et sont adaptés pour les applications impliquant une faible communication et une faible portée.

La bande utilisée pour les modules fournis est la bande 2.4GHz, et la puissance de sortie de 1mW. La portée dans des conditions optimales est d'environ 100m.

Les modules ont été préconfigurés au préalable afin d'éviter les conflits. L'adresse du réseau utilisé par chaque paire de modules XBee est indiquée sur ces derniers.

Les shields XBee possèdent un interrupteur permettant de sélectionner le mode *Micro* ou le mode *USB* :

- Lorsque l'interrupteur est en position *Micro*, la broche DOUT du shield XBee est connectée à la broche RX du microcontroleur, et la broche DIN à la broche TX. Cela implique que les données

émises par le microcontroleur sur la ligne TX seront transmises sans fil par le module XBee. De façon analogue, les données reçues sans fil par le module XBee seront présentes sur la broche RX du microcontroleur, permettant ainsi de lire les données depuis la carte Arduino UNO. Il ne sera pas possible de reprogrammer le microcontroleur dans ce cas.

- Lorsque l'interrupteur est en position *USB*, les broches DOUT et DIN sont connectées aux broches RX et TX du convertisseur USB-série présent sur l'Arduino UNO. Ce mode vous permettra de reprogrammer le microcontroleur.

Pour résumer, en position *Micro*, l'émission/réception de données par XBee depuis l'Arduino UNO se fera à l'aide du jeu d'instructions de communication série dont vous trouverez une description dans l'aide Arduino.

## Question et remarques :

vincent.raimbault@ims-bordeaux.fr