Projet M2M

2015-2016

Capteur XBEE

Binone :
ZAKARI TOURE Ismael
DIAGNE el hadji Malick

I- Contexte

Dans le cadre de notre formation Master 2 Pro. Génie Informatique, nous sommes amenés à concevoir un système informatique d'interaction "Machine -to- Machine". Ce système mettra en œuvre l'exploitation de capteurs, embarqués sur un système distant alimenté par une batterie. Ce système devra par conséquent être minimal et transmettra les données capturées à un autre système plus important via les modules radios Xbee fournis pour réaliser ce projet.

II- Objectif du projet

Nous avons choisi de mettre en œuvre un système de surveillance (des plantes) d'un jardin. Ce système mettra en œuvre plusieurs capteurs listés ci-dessous. Il permettra d'avoir un aperçu des niveaux de luminosité, d'humidités (du sol et de l'air) et pluviométrie de notre jardin. L'objectif est de réduire au minimum le développement/écriture de code, au profit de la coopération de systèmes existants à travers leur configuration. A cet effet, nous utiliserons les technologies sous-mentionnées, notamment la plat-forme InfluxData pour la collecte, le stockage et le rendu de données séries capturées.

III- Matériel utilisé

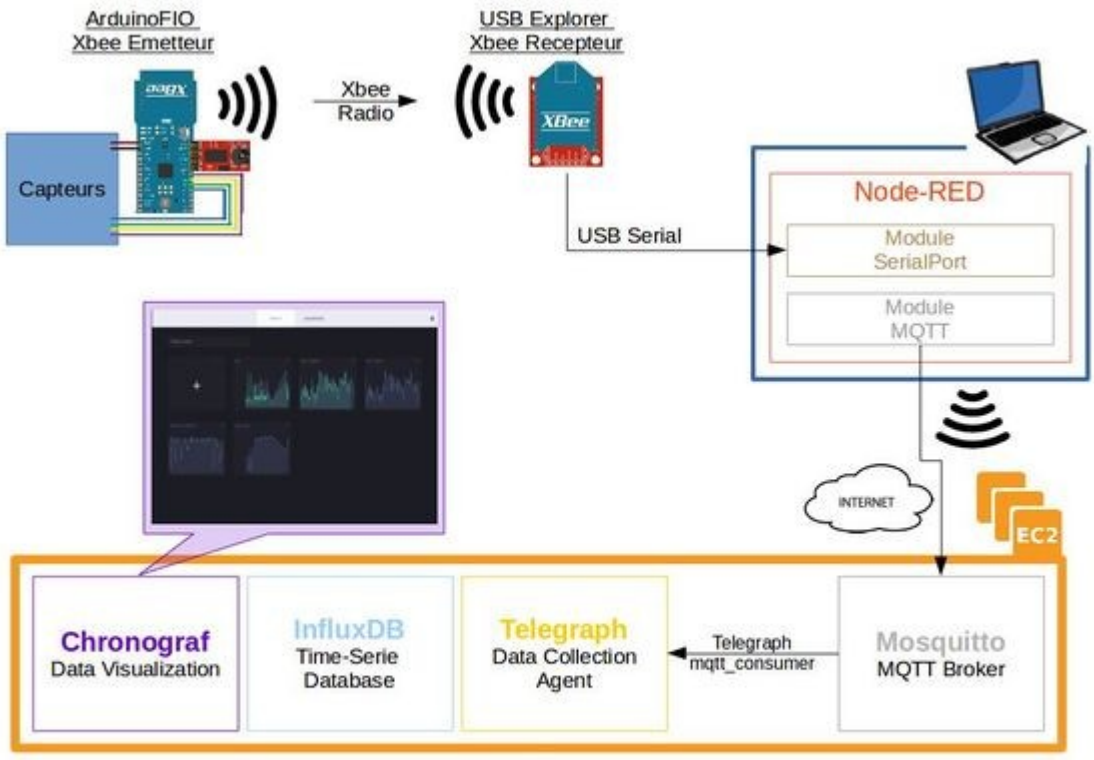
- 2 modules XBee: 1x module S1 et 1x module Pro
- 1x "Breakout board" Xbee USB Explorer pour programmer les Xbee
- 1x Arduino FIO
- 1x "Breakout board" FTDI pour programmer la FIO
- 1x photo-résistance, 2x capteurs d'humidité (air et sol) et 1x capteur de pluie

IV- Technologies utilisées

- Mosquitto, un broker MQTT
- Node-RED, avec le nœud node-red-node-serialport (à installer) pour lire les données reçues par le module Xbee "local"
- AWS EC2
- Telegraf
- InfluxDB
- Chronograf

V- Plan de développement

1.Architecture - Conception générale



Architecture – Conception générale

2. Système embarqué

Configuration des modules Xbee

Afin que les modules puissent communiquer entre eux, nous devons les configurer. Pour ce faire, nous disposons d'une "breakout board" USB Explorer Xbee.
Les 2 modules utiliseront les paramètres suivants:

- baud rate : 57600
- flow control: none
- data bit: 8
- parity: none
- stop bits: 1

Voici les paramètres que nous avons configuré pour chacun de nos modules:

	Xbee Emetteur	**Xbee Recepteur**
Adresse du réseau	1111	1111
Adresse du module dans le réseau	1	0
Adresse du destinataire dans le réseau	0	1
Autoriser l'émission des I/O	1	0
Autoriser la réception des I/O	0	1

Nous pouvons configurer les modules soit en utilisant un terminal série comme minicom ou screen, soit avec X-CTU.
En bas de page, quelques références qui sont d'une grande aide à la configuration de modules.

Montage

Ci-contre, voici le schéma de montage de notre ArduinoFIO:

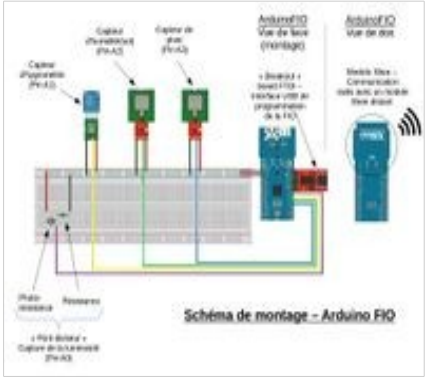


Schéma de montage - Arduino FIO

Branchements:

-Pin A0: Capteur de luminosité

-Pin A1: Capteur d'hygrométrie

-Pin A2: Capteur d'humidité du sol

-Pin A3: Détecteur de pluie

3- Programmation du ArduinoFIO

Nous allons maintenant programmer notre ArduinoFIO afin de lire les valeurs mesurées sur les pins A0, A1, A2 et A3, les mettre en forme et les envoyer via les modules Xbee à notre machine "locale". Le code arduino compilé dans notre ArduinoFIO est le suivant:

```
#include <SoftwareSerial.h>
SoftwareSerial xbee(2, 3);
int count;
int lightPin = A0;
int rainPin = A1;
int grdHumPin = A2;
int airHumPin = A3;
float lightValue = 0;
float rainValue = 0;
float grdHumValue = 0;
float airHumValue = 0;
float maxR = 0;
float maxG = 0;
float maxA = 0;
float minR = 1024;
float minG = 1024;
float minA = 1024;

void setup()
{

     xbee.begin(57600);
     Serial.begin(57600);
     count = 0;
}

/*************************************
*  MAIN LOOP
```

```
28.*************************************/
29.
30.void loop()
31.{
32.
33.  // Lecture des valeurs des capteurs
34.  lightValue = analogRead(lightPin);
35.  rainValue = analogRead(rainPin);
36.  grdHumValue = analogRead(grdHumPin);
37.  airHumValue = analogRead(airHumPin);
38.
39.  // Mise à jour des max et min
40.  if(rainValue < minR) minR=rainValue;
41.  if(grdHumValue < minG) minG=grdHumValue;
42.  if(airHumValue < minA) minA=airHumValue;
43.
44.  if(rainValue > maxR) maxR=rainValue;
45.  if(grdHumValue > maxG) maxG=grdHumValue;
46.  if(airHumValue > maxA) maxA=airHumValue;
47.
48.  // Calcul des pourcentages des mesures
49.  float a = 0;
50.  // Pourcentage de luminosité assez simple:
51.  lightValue = lightValue/10.24;
52.  //Pourcentage de pluviométrie
53.  a = 100/(minR-maxR);
54.  rainValue = a*rainValue-maxR*a;
55.  //Pourcentage de l'humidité du sol
56.  a = 100/(minG-maxG);
57.  grdHumValue = a*grdHumValue-maxG*a;
58.  //Pourcentage de l'humidité de l'air
59.  a = 100/(minA-maxA);
60.  airHumValue = a*airHumValue-maxA*a;
61.
62.  // Création de la donnée mesurée ("JSONobject string")
63.  String Light_Str = "\"light\": ";
64.  String Rain_Str = "\"rain\": ";
65.  String GrdHum_Str = "\"grdhum\": ";
66.  String AirHum_Str = "\"airhum\": ";
67.
68.  Light_Str += (int)lightValue;
69.  Rain_Str += (int)rainValue;
70.  GrdHum_Str += (int)grdHumValue;
71.  AirHum_Str += (int)airHumValue;
72.
73.  String phrase = "{"+Light_Str+", "+Rain_Str+", "+GrdHum_Str+", "+AirHum_Str+"}";
74.
75.  // Affichage et Envoi de la donnée via Xbee
76.  Serial.println(phrase);
77.  xbee.println(phrase);
78.
79.  delay(1000);
80.}
```

V.3 Serveur Node-RED ("local")

Installation de Node-RED

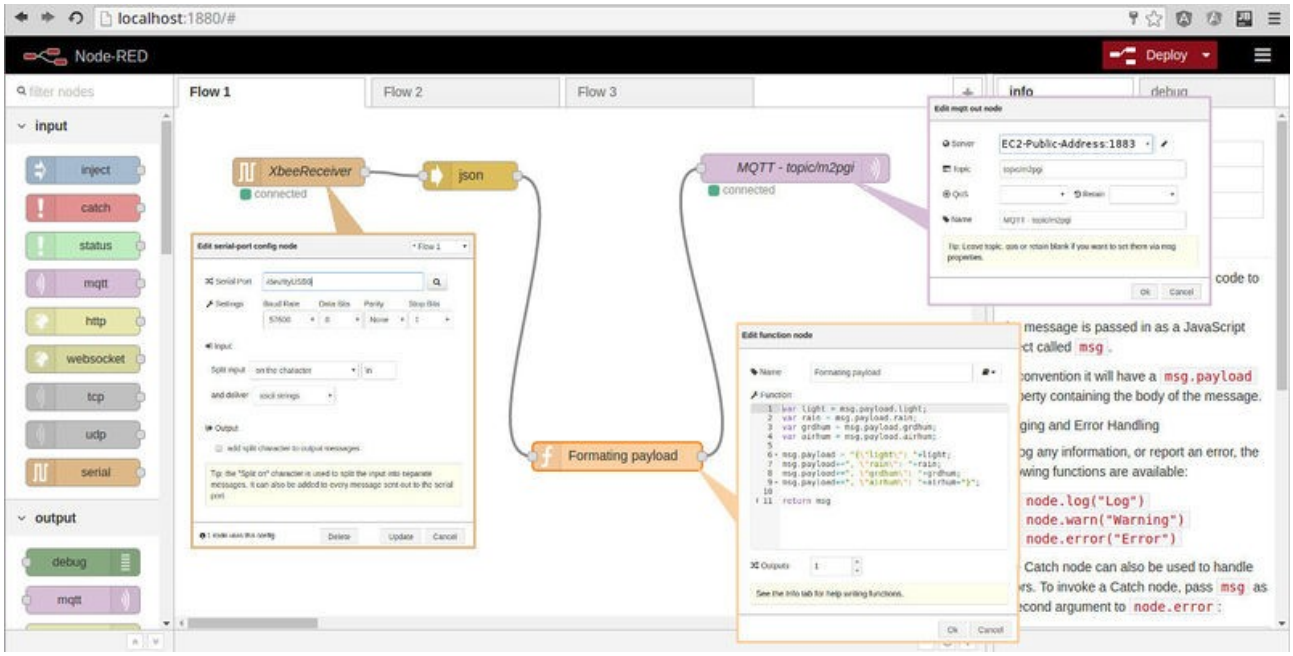
Nous utiliserons npm pour installer Node-Red sur notre machine locale. Dans un terminal, entrer:

```
sudo npm install node-red
```

Nous aurons besoin de créer un noeud "serialport" pour recevoir les données du module Xbee distant. Nous aurons besoin du module node-red-serialport

```
cd ~/.node-red
sudo npm install node-red
```

Création des noeuds Node-RED



Lancer le serveur node-red. Dans un terminal, entrer:

```
node-red
```

Il est maintenant accessible sur le port 1880 de notre machine locale ("localhost"). Ci-contre, une capture d'écran de nos noeuds Node-RED et les configurations qui leurs sont associées.

V.4 Serveur AWS ("distant")

Pour la suite nous avons créé un compte AWS ainsi qu'une instance EC2-Linux.
Après avoir lancé notre instance EC2, nous pouvons maintenant nous y connecter et y installer et configurer les services TICK ("Telegraf-InfluDB-Chronograf-Kapacitor").

Mais avant cela, commençons par installer le broker MQTT Mosquitto.

Installation de Mosquitto
Dans le terminal connecté à l'instance EC2, entrer:

```
// Installation de Mosquitto
sudo apt-get install mosquitto

// Lancer mosquitto
mosquitto
```

Mosquitto est ainsi actif sur notre instance EC2 (port 1883).

Installation de InfluxDB, Telegraf et Chronograf

Pour installer ces services, entrer dans le terminal (celui connecté à l'instance EC2):

```
// Téléchargement et installation de Telegraf
wget http://get.influxdb.org/telegraf/telegraf_0.12.0-1_amd64.deb
sudo dpkg -i telegraf_0.12.0-1_amd64.deb

// Téléchargement et installation de InfluDB
wget https://s3.amazonaws.com/influxdb/influxdb_0.12.1-1_amd64.deb
sudo dpkg -i influxdb_0.12.1-1_amd64.deb

// Téléchargement et installation de Chronograf
wget https://s3.amazonaws.com/get.influxdb.org/chronograf/chronograf_0.12.0_amd64.deb
sudo dpkg -i chronograf_0.12.0_amd64.deb
```

Une fois ces services installés, nous allons maintenant les configurer.

Configuration de Telegraf

Nous allons commencer par créer un fichier de configuration pour notre service.

```
telegraf -sample-config -input-filter mqtt_consumer -output-filter influxdb >
telegraf.conf
```

Avec cette commande, nous avons créé un fichier "telegraf.conf" à partir de la configuration par défaut de telegraf mais uniquement avec les sections qui nous intéressent :

- en "input", le service "mqtt_consumer" pour intercepter les entrées émises via le broker MQTT Mosquitto de notre instance EC2
- en "output", le service "influxdb" pour mettre en forme ces entrées et les stocker dans InfluxDB. Il faut noter que Telegraf y créera par défaut une table "telegraf" que nous exploiterons.

La section "INPUT PLUGINS" sera modifiée de la sorte:

```
###############################################################################
#                            INPUT PLUGINS                                    #
###############################################################################


###############################################################################
#                            SERVICE INPUT PLUGINS                            #
###############################################################################

# Read metrics from MQTT topic(s)
[[inputs.mqtt_consumer]]
  servers = ["EC2-Public-Address:1883"]
  ## MQTT QoS, must be 0, 1, or 2
  qos = 0

  ## Topics to subscribe to
  topics = [ "topic/m2pgi" ]

  # if true, messages that can't be delivered while the subscriber is offline
  # will be delivered when it comes back (such as on service restart).
  # NOTE: if true, client_id MUST be set
  persistent_session = false
  # If empty, a random client ID will be generated.
  client_id = ""

  ## username and password to connect MQTT server.
   username = "m2pgi"
   password = "m2pgi"

  ## Optional SSL Config
  # ssl_ca = "/etc/telegraf/ca.pem"
  # ssl_cert = "/etc/telegraf/cert.pem"
  # ssl_key = "/etc/telegraf/key.pem"
  ## Use SSL but skip chain & host verification
  # insecure_skip_verify = false

  ## Data format to consume.
  ## Each data format has it's own unique set of configuration options, read
  ## more about them here:
  ## https://github.com/influxdata/telegraf/blob/master/docs/DATA_FORMATS_INPUT.md
  data_format = "json"
```

Nous n'avons pas besoin de modifier la section "INPUT PLUGINS" car telegraf est configuré par défaut pour stocker les entrées séries dans InfluxDB (notamment dans la table "teleagraf").

Note : Il ne serait pas inutile de se documenter un peu plus sur la configuration détaillée de telegraf.

Configuration de InfluxDB

Nous n'avons pas besoin de configuration particulière à effectuer pour InfluxDB, la configuration par défaut suffit amplement.

Chronograf
La configuration et l'utilisation de Chronograf n'est pas compliquée. Elles sont d'autant plus simple que nous avons gardé les configuration par défaut de Telegraf et InfluxDB.

Petit rappel : nous avons gardé la configuration initiale de Telegraf. Nous exploiterons la database "telegraf" de InfluxDB, créée par défaut par Telegraf.**Note :** Pour accéder à chronograf depuis l'extérieur de notre instance EC2, penser à modifier son fichier de configuration à cet effet (par défaut "/opt/chronograf/config.toml").

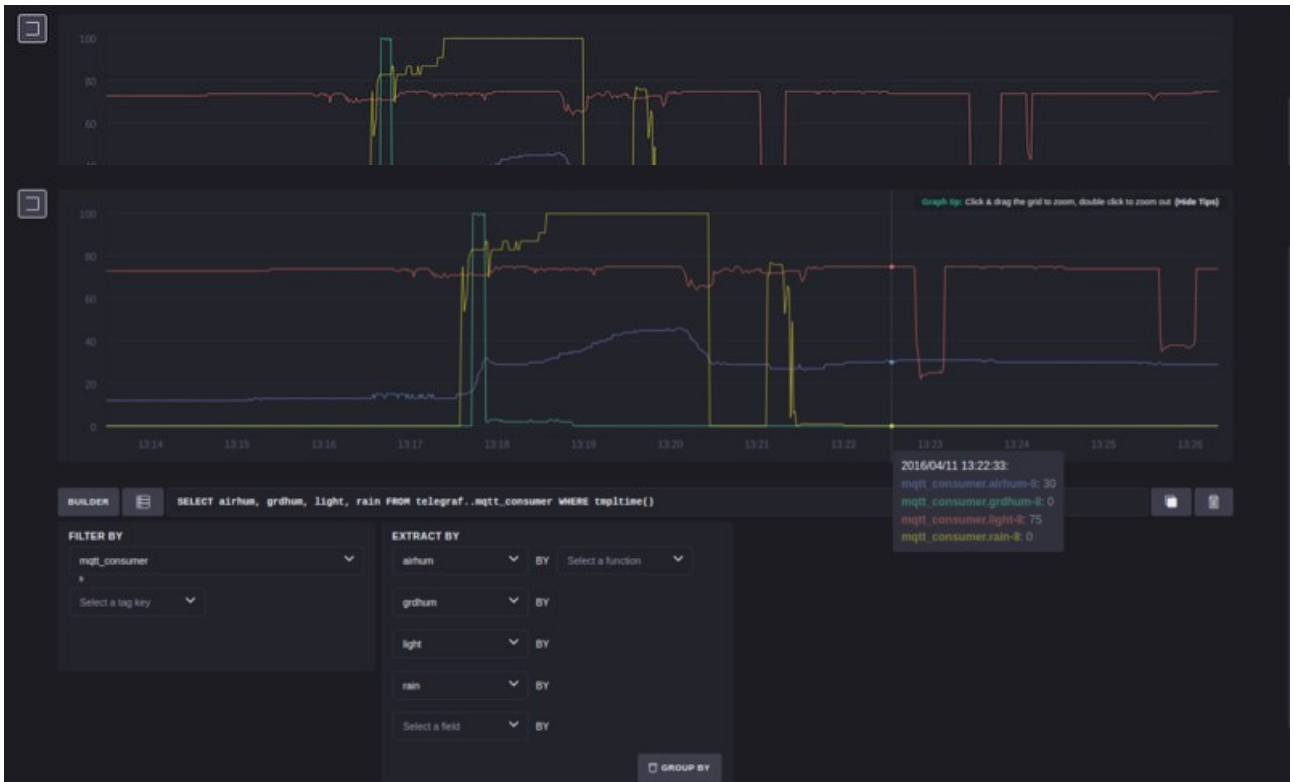
Lancement de nos services
Il ne nous reste plus qu'à lancer nos services Telegraf, InfluxDB et Chronograf:

```
// Lancer InfluDB
sudo service influxdb start

// Lancer Chronograf
sudo service chronograf start

// Lancer Telegraf
sudo service influxdb start

// Lancer la capture des entrées Xbee par Telegraf (en utilisant notre fichier de configuration édité précédemment)
telegraf -config myTelegraf.conf
```

Une fois les services lancés, la capture des données commence. Il ne nous reste plus qu'à visualiser ces données avec Chronograf. Chronograf est accessible sur le port :10000 de notre instance EC2.

Expérimentations et Résultats

Voici quelques captures d'écran de notre dashboard:



Captures du matériel utilisé.