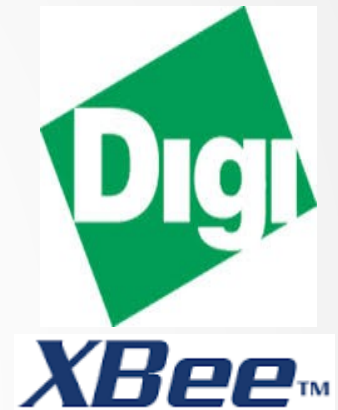
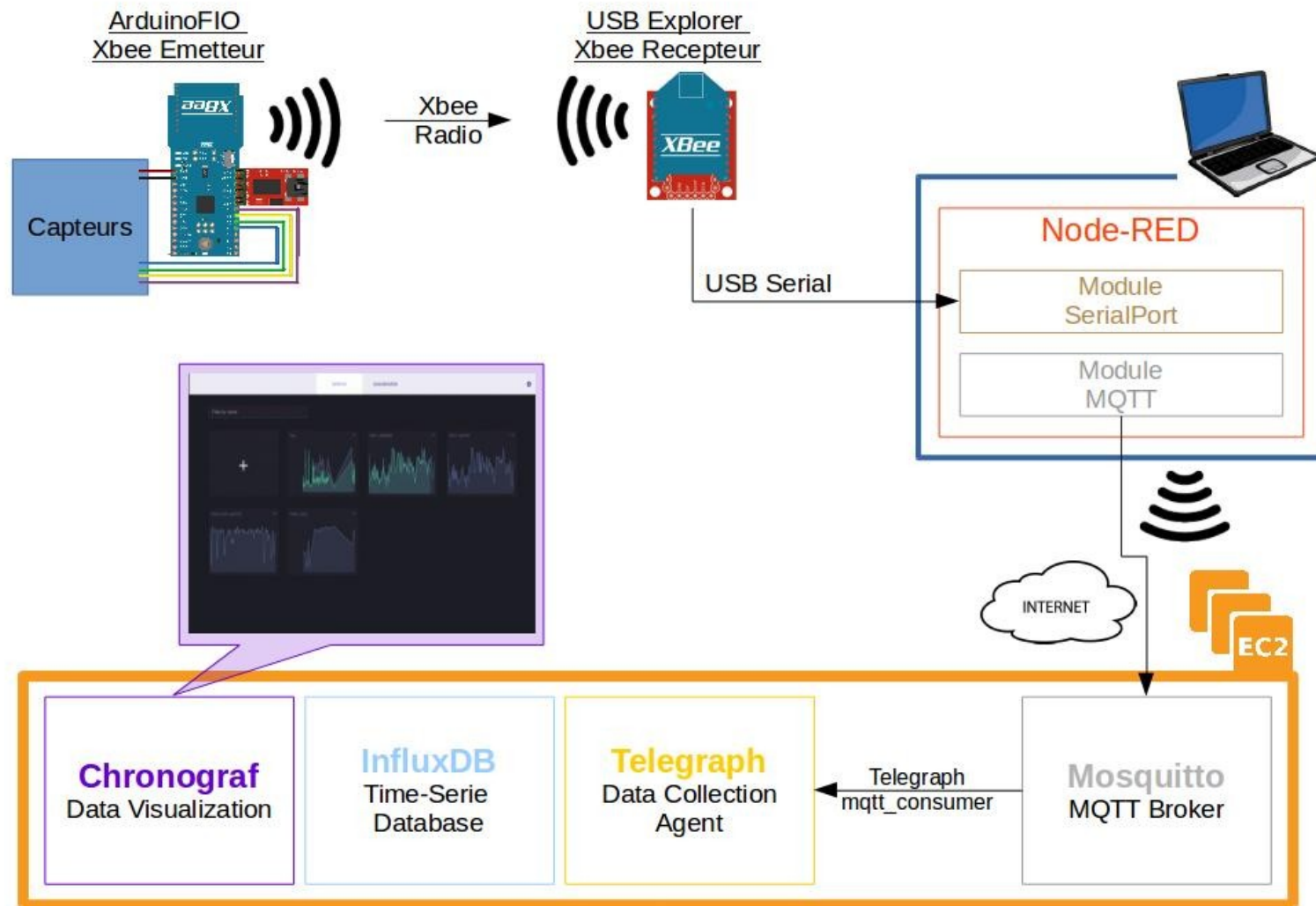


Xbee - Monitoring



Monitoring de votre jardin
(basé sur les module Xbee)

Architecture



Architecture – Conception générale

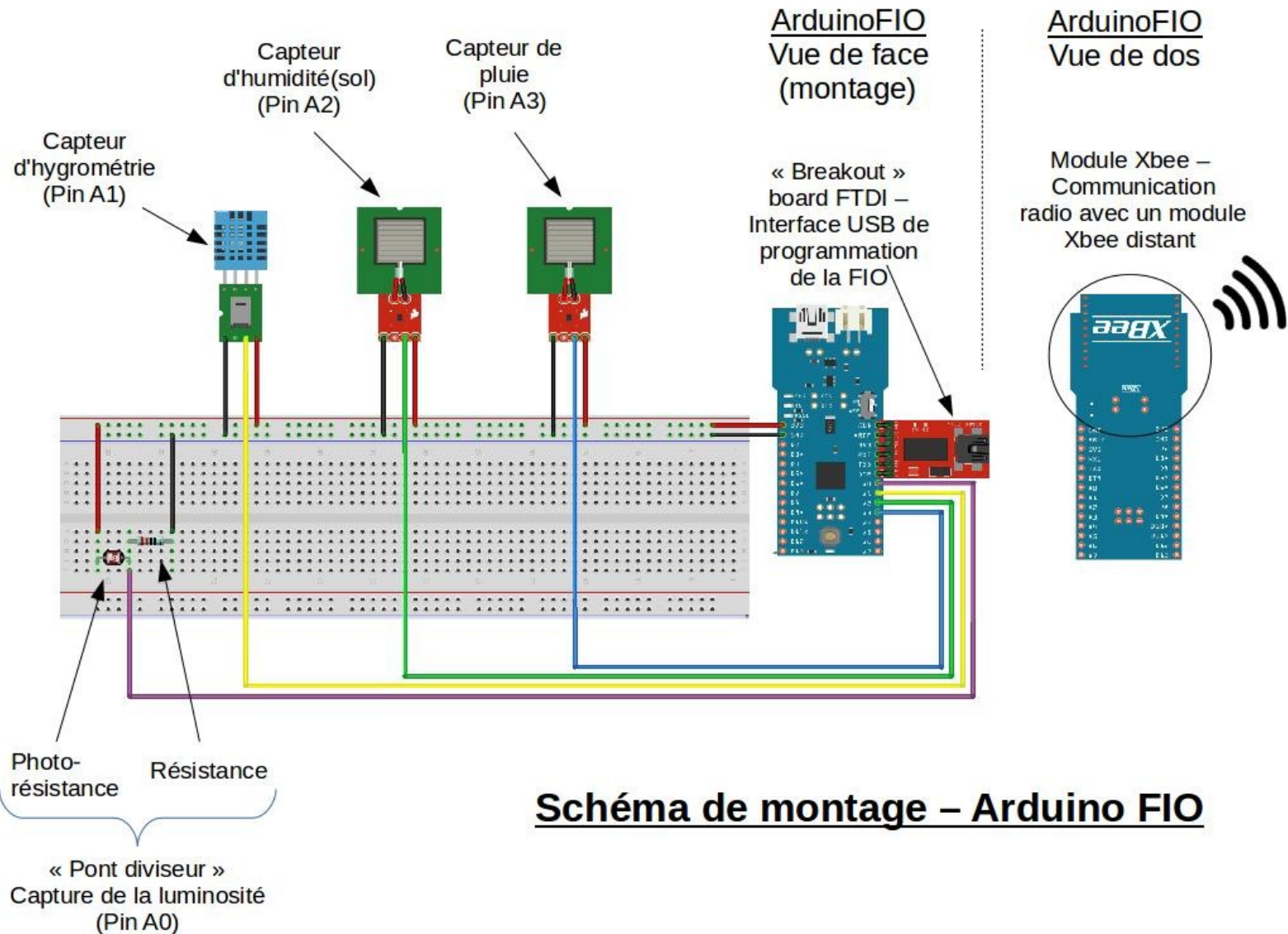
Xbee - Configuration

- Terminal Série (Minicom) ou X-CTU
 - baud rate : 57600
 - flow control: none
 - data bit: 8
 - parity: none
 - stop bits: 1

Xbee - Configuration

	Emetteur	Récepteur
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

ArduinoFIO - Montage



Machine locale

- Installation de Mosquitto
- Installation de Node-RED
- Installation de node-red-serialport
- Lancement : node-red

Machine Locale

The screenshot displays the Node-RED web interface in a browser at `localhost:1880/#`. The interface includes a sidebar with node categories (input, output, function, mqtt, http, websocket, tcp, udp, serial) and a main workspace showing a flow named "Flow 1".

The flow consists of the following nodes:

- XbeeReceiver** (input node, connected): Receives data from a serial port.
- json** (input node): Parses the received data into a JSON object.
- Formating payload** (function node): Processes the JSON data into a specific format.
- MQTT - topic/m2pgi** (mqtt node, connected): Publishes the formatted data to the MQTT topic `topic/m2pgi`.

Three configuration panels are open:

- Edit serial-port config node**: Shows the serial port configuration for the XbeeReceiver node. The serial port is `/dev/ttyUSB0`, baud rate is `57600`, data bits are `8`, parity is `None`, and stop bits are `1`. The input is split on the character `\n` and delivered as `ascii strings`.
- Edit mqtt out node**: Shows the MQTT output configuration for the MQTT node. The server is `EC2-Public-Address:1883`, the topic is `topic/m2pgi`, and the name is `MQTT - topic/m2pgi`.
- Edit function node**: Shows the JavaScript code for the "Formating payload" function. The code is as follows:

```
1 var light = msg.payload.light;
2 var rain = msg.payload.rain;
3 var grdhum = msg.payload.grdhum;
4 var airhum = msg.payload.airhum;
5
6 msg.payload = "{ \"light\": \"+light;
7 msg.payload+=\", \"rain\": \"+rain;
8 msg.payload+=\", \"grdhum\": \"+grdhum;
9 msg.payload+=\", \"airhum\": \"+airhum+\"}\"";
10
11 return msg
```

On the right side of the interface, there is a text area with the following text:

message is passed in as a JavaScript object called `msg`. By convention it will have a `msg.payload` property containing the body of the message.

Logging and Error Handling

If you want to log any information, or report an error, the following functions are available:

- `node.log("Log")`
- `node.warn("Warning")`
- `node.error("Error")`

Catch node can also be used to handle errors. To invoke a Catch node, pass `msg` as the first argument to `node.error`.

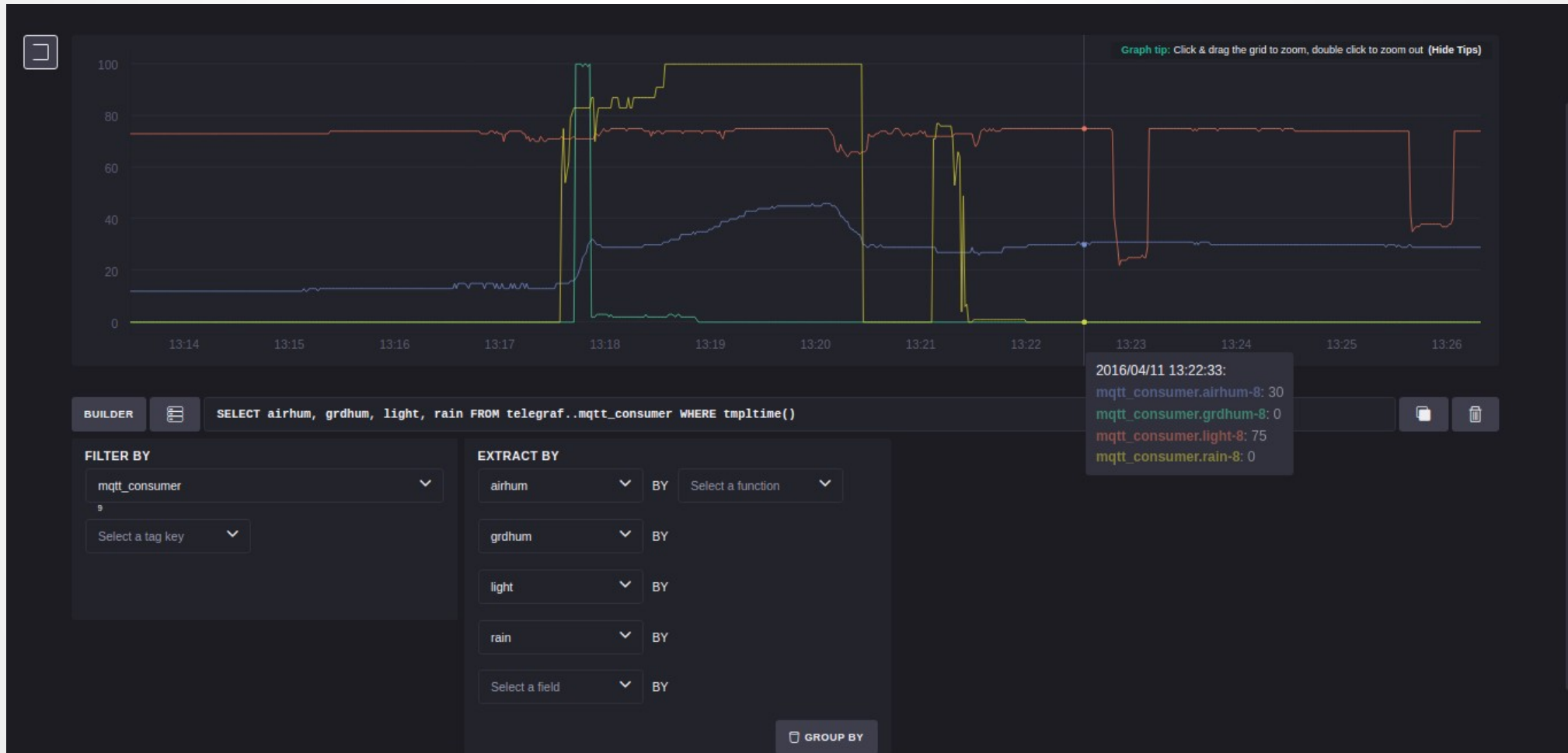
Instance EC2 - AWS

- Instance EC2 : « Security Group »
- Lancement
- Installation de Mosquitto
- Installation Telegraf-Influx-Chronograf(-K)
- Configuration de Telegraf et Chronograf
- Lancement des services

Visualisation - Chronograf



Visualisation - Chronograf



Démonstration