

INTERNET OF THINGS  
**Agricultural monitoring system**  
Master HES-SO

Émilie GSPONER, Grégory EMERY

10 juin 2016  
version 1.0

## Table des matières

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Introduction . . . . .</b>                        | <b>3</b> |
| <b>2</b> | <b>Communication Bluetooth . . . . .</b>             | <b>4</b> |
| 2.1      | Matériel à disposition . . . . .                     | 4        |
| 2.2      | Séquence d'acquisition des données . . . . .         | 4        |
| 2.3      | Seuils de génération d'alarme . . . . .              | 5        |
| <b>3</b> | <b>Application client : agroMQTTClient . . . . .</b> | <b>5</b> |
| <b>4</b> | <b>Conclusion . . . . .</b>                          | <b>8</b> |

# 1 Introduction

Le projet choisi est celui du système de monitoring pour l'agriculture. Son but est de collecter des données de plusieurs capteurs répartis dans un champ et de les transmettre à l'agriculteur peu importe où il se trouve sur la parcelle.

Dans le cadre de ce projet nous allons utiliser deux capteurs SensorTag qui communiquent en Bluetooth Low Energy (BLE) pour collecter des informations sur l'humidité, la température et la luminosité. Les données sont ensuite transmises à un contrôleur via BLE qui va analyser les informations et générer une alerte si les champs ont besoin d'être irrigués. Toutes les données collectées ainsi que l'alerte sont transmises au travers d'un réseau LoRaWAN.

Optionnellement nous pouvons implémenter une petite application permettant de voir les données et alarmes collectées par le réseau.

L'image ci-dessous représente notre système.

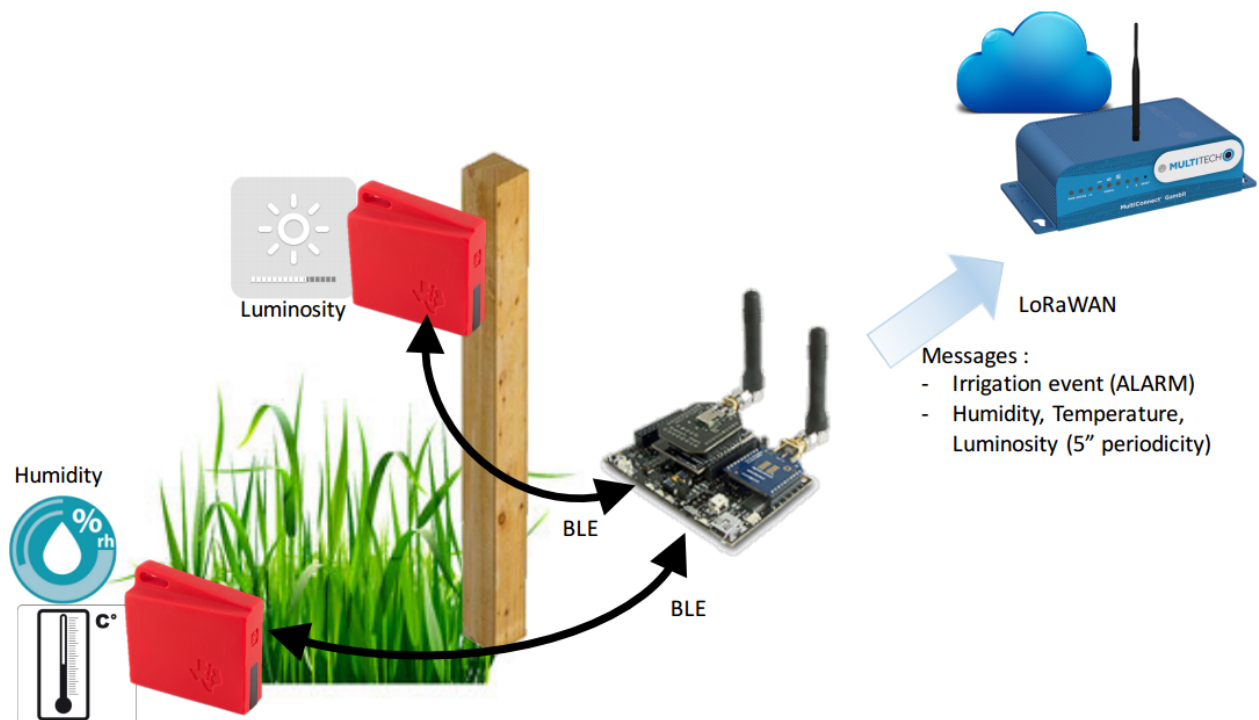


FIGURE 1 – Représentation du système

## 2 Communication Bluetooth

### 2.1 Matériel à disposition

Pour ce projet, nous avons pu avoir deux boîtes complètes à disposition. Voici quelques caractéristiques de ce matériel :

- SensorTag 1 :
  - Numéro du capteur : 9
  - Mac adresse : b0 :b4 :48 :c9 :b3 :85
  - Données collectées : Luminosité
- SensorTag 2 :
  - Numéro du capteur : 16
  - Mac adresse : b0 :b4 :48 :c9 :ba :01
  - Données collectées : Température, Humidité
- Carte Bluetooth du Waspote :
  - Numéro du socket : 1

### 2.2 Séquence d'acquisition des données

Nous avons décidé de collecter les données selon les étapes suivantes dans la boucle principale :

1. Au lancement de l'application, le module Bluetooth est connecté au socket 0 une seule fois dans la méthode setup.
2. L'application commence par tenter la connexion avec le capteur 1. Si la connexion échoue, l'application passe à la connexion avec le capteur 2 (étape 5).
3. Sinon, une fois le capteur connecté, la période d'acquisition du capteur de luminosité est configurée à 100ms et la mesure est activée.
4. On attend ensuite 1 seconde, le temps que le capteur fasse quelques mesures.
5. On va ensuite lire la valeur de la caractéristique et convertir les données. La mesure sur le capteur est ensuite désactivée et l'on se déconnecte du capteur.
6. Les étapes 2 à 5 sont ensuite répétées mais avec le capteur 2 pour récupérer l'humidité et la température.

Toutes les données sont stockées dans des variables globales. Si la connexion avec un des capteurs échoue, ce sont les valeurs précédentes du capteur qui sont envoyées au réseau LoRaWan. L'irrigation d'un champ n'a pas besoin de se faire à la minute près, le fait qu'une des données n'est pas mise à jour pour 5 minutes n'est donc pas dramatique. C'est également pour cela que nous n'avons pas choisi d'utiliser la notification pour recevoir les données des capteurs. L'acquisition des valeurs toutes les 5 minutes est suffisante.

Les capteurs sont désactivés pour ne plus faire de mesure

## 2.3 Seuils de génération d'alarme

# 3 Application client : agroMQTTClient

Un petit client a été codé pour le pur plaisir d'utiliser le langage Go, souvent appelé Golang, de chez Google<sup>1</sup>. Go a plusieurs particularités :

- Simplicité de syntaxe : le Go se lit sans problème sans même avoir d'expérience dans ce langage.
- Gestion du parallélisme : Go inclut nativement la gestion du multi-threading. Il faut juste écrire *go* avant l'appel d'une fonction pour que celle-ci soit lancée en parallèle du programme principal.
- Librairie standard : Go a une librairie standard qui couvre beaucoup de domaines, de la crypto jusqu'à la manipulation d'images et à l'encodage/décodage CSV.

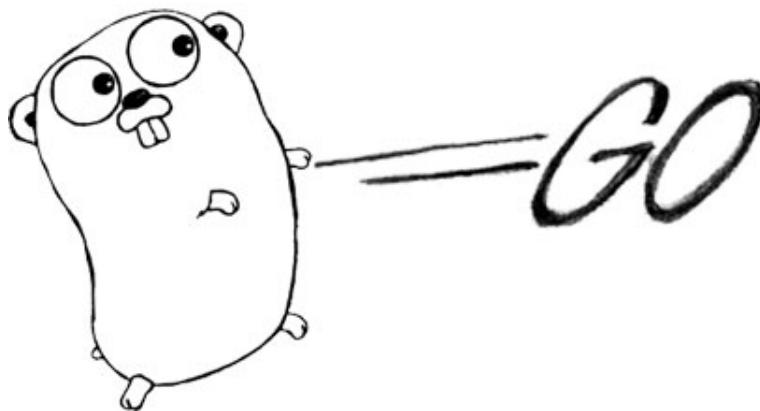


FIGURE 2 – Logo du langage de Google, Golang

Le projet de communication M2M *paho*<sup>2</sup> fournit une librairie client Go qui est très simple d'utilisation. Un exemple est d'ailleurs donné sur la page <https://eclipse.org/paho/clients/golang/> et c'est lui qui a servi de base pour la mini-app.

Voici d'ailleurs ce que l'application fait :

- 
1. <https://golang.org/>
  2. <http://www.eclipse.org/paho/>

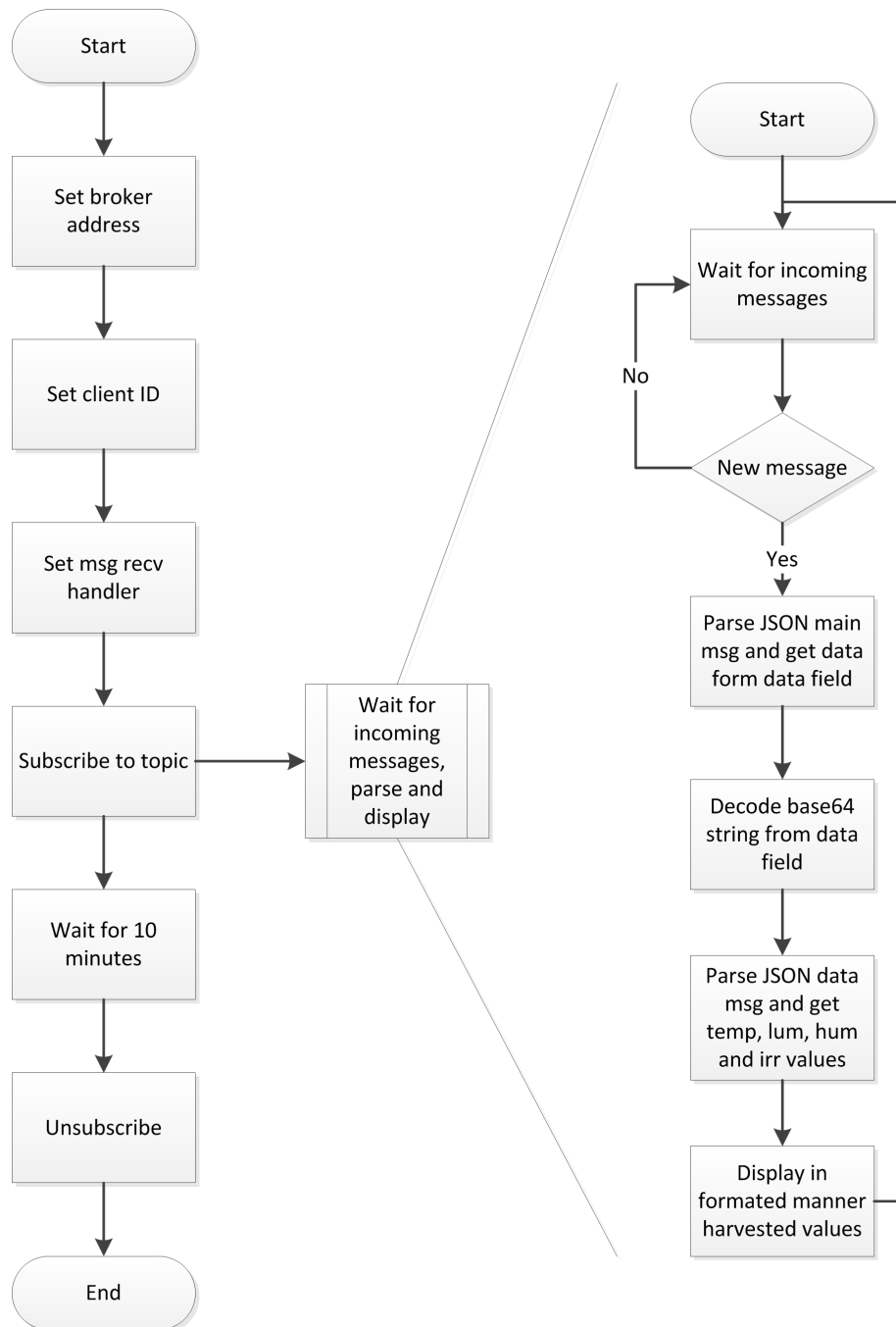


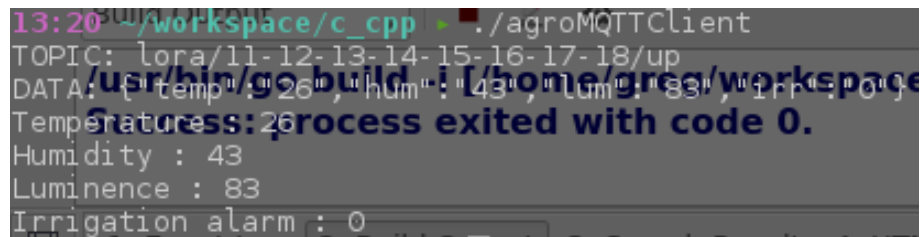
FIGURE 3 – Étapes réalisées par l'application

Les différents champs requis par le client sont remplis. Et on inscrit un callback sous forme de fonction anonyme comme récepteur des messages produits par le *broker*. Dès qu'on a souscrit au bon *topic*, le handler est lancé et s'exécute dès qu'un message arrive. Pendant ce temps, le programme principal attend, ici une période de 10 minutes, avant de se désinscrire du *topic*. Le programme principal s'arrête et stoppe le thread de réception des messages.

La fonction de réception n'est appelée qu'en cas de réception. Elle ne fait pas de polling elle-même sur l'arrivée de messages. Elle même reçoit en paramètre un message encodé en

JSON. On parse ce JSON pour en récupérer les données, elles-mêmes encodées en base64. Go dispose, dans sa librairie standard, de tout ce qu'il faut pour décoder ces formats. Une fois le base64 décodé en string, on décode à nouveau un JSON, qui a cette fois servi à transporter les données importantes pour notre application, la température, la luminosité, l'humidité et s'il y a lieu de s'inquiéter à propos de l'irrigation des cultures.

On récupère ces données et les affichent simplement dans la console.



```
13:20 ~/workspace/c_cpp • ./agroMQTTClient
TOPIC: lora/11-12-13-14-15-16-17-18/up
DATA: {"temp":"26","hum":"43","lum":"83","irr":"0"}
Temperature: 26
Humidity : 43
Luminence : 83
Irrigation alarm : 0
Success: process exited with code 0.
```

FIGURE 4 – Réception et affichage d'un message

## 4 Conclusion