

INTERNET OF THINGS
Agricultural monitoring system
Master HES-SO

Émilie GSPONER, Grégory EMERY

13 juin 2016
version 1.0

Table des matières

1	Introduction	3
2	Communication Bluetooth	4
2.1	Matériel à disposition	4
2.2	Séquence d'acquisition des données	4
2.2.1	Caractéristiques utilisées	5
2.3	Seuils de génération d'alarme	6
2.4	Attente de 5 minutes	6
3	Communication LoRaWAN	6
3.1	Tests de la communication	6
4	Application client : agroMQTTClient	8
5	Conclusion	10

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.

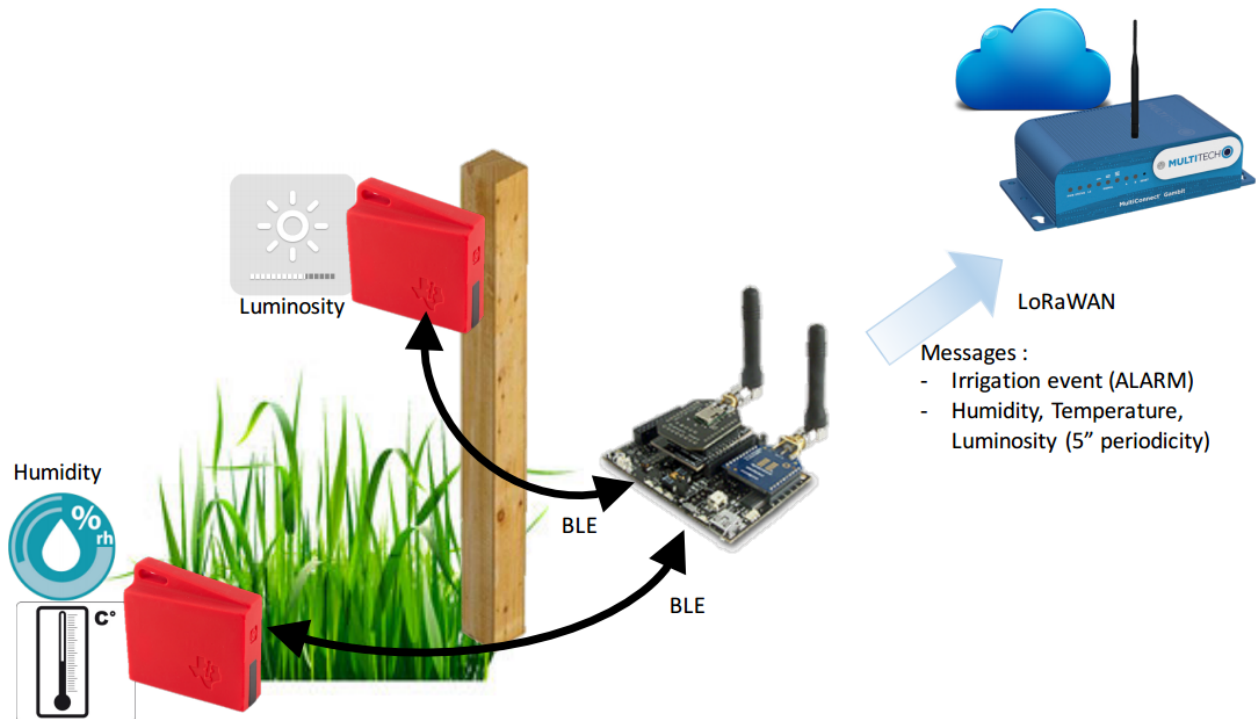


FIGURE 1 – Représentation du système

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 Wasp mote :
 - 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 1 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 pendant les 5 minutes d'attente, cela permet d'économiser leur batterie.

2.2.1 Caractéristiques utilisées

En nous basant sur le site http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/tearDown.html, nous avons utilisé le service présenté ci-dessous pour collecter les données du capteur d'humidité. L'avantage est que cette caractéristique donne également la température. On peut avoir les deux valeurs en une seule lecture.

0x27	39	0x2800	GATT Primary Service Declaration	F000AA20-0451-4000-B000-00000000	R	Humidity Service
0x28	40	0x2803	GATT Characteristic Declaration	12:29:00:00:00:00:00:00:00:00:B0:00:40:51:04:21:AA:00:F0	R	Humidity Data
0x29	41	0xAA21	Humidity Data		RN	TempLSB:TempMSB:HumidityLSB:HumidityMSB
0x2A	42	0x2902	Client Characteristic Configuration		RW	Write "01:00" to enable notifications, "00:00" to disable
0x2B	43	0x2803	GATT Characteristic Declaration	0A:2C:00:00:00:00:00:00:00:00:B0:00:40:51:04:22:AA:00:F0	R	Humidity Config
0x2C	44	0xAA22	Humidity Config		RW	Write "01" to start measurements, "00" to stop
0x2D	45	0x2803	GATT Characteristic Declaration	0A:2E:00:00:00:00:00:00:00:00:B0:00:40:51:04:23:AA:00:F0	R	Humidity Period
0x2E	46	0xAA23	Humidity Period		RW	Period = [Input*10] ms, (lower limit 100 ms), default 1000 ms

FIGURE 2 – Caractéristique pour la température et l'humidité

Le service suivant est celui utilisé pour l'acquisition de la luminosité.

0x3F	63	0x2800	GATT Primary Service Declaration	F000AA70-0451-4000-B000-00000000	R	Luxometer Service
0x40	64	0x2803	GATT Characteristic Declaration	12:41:00:00:00:00:00:00:00:00:B0:00:40:51:04:71:AA:00:F0	R	Luxometer Data
0x41	65	0xAA71	Luxometer Data		RN	LSB:MSB
0x42	66	0x2902	Client Characteristic Configuration		RW	Write "01:00" to enable notifications, "00:00" to disable
0x43	67	0x2803	GATT Characteristic Declaration	0A:44:00:00:00:00:00:00:00:00:B0:00:40:51:04:72:AA:00:F0	R	Luxometer Config
0x44	68	0xAA72	Luxometer Config		RW	Write "01" to start Sensor and Measurements, "00" to put to sleep
0x45	69	0x2803	GATT Characteristic Declaration	0A:46:00:00:00:00:00:00:00:00:B0:00:40:51:04:73:AA:00:F0	R	Luxometer Period
0x46	70	0xAA73	Luxometer Period		RW	Period = [Input*10]ms (lower limit 1000ms), default 2000ms

FIGURE 3 – Caractéristique pour la luminosité

Remarque : Sur le site cité plus haut, il y a une erreur, le service pour avoir la luminosité qui est présenté n'est pas le bon. Il faut aller dans l'onglet *Full GATT Table* pour voir le vrai luxometer service.

Le wiki, http://processors.wiki.ti.com/index.php/CC2650_SensorTag_User's_Guide, a été utilisé pour convertir les données brutes des capteurs dans la bonne unité.

Remarque : Nous avons remarqué que toutes nos données converties n'avaient aucun sens. Cela est dû à Libellium qui rajoute un byte (le premier, 0) aux données lues par Bluetooth

pour indiquer la taille des données à lire. Il faut donc commencer au byte 1 des données reçues pour les convertir. Les valeurs obtenues ont tout de suite plus de sens.

2.3 Seuils de génération d'alarme

Le paramètre alarme a une valeur de 0 si tout va bien et de 1 si le champ a besoin d'irrigation. Nous avons fixé trois seuils déclenchant l'alarme :

- Si la température est supérieure à 30 degrés
- Si le taux d'humidité passe en dessous de 30%
- Si la luminosité est supérieure à 30000 lux. Ce dernier seuil a été fixé selon Wikipédia qui dit que cela correspond à une exposition en plein soleil.

2.4 Attente de 5 minutes

Nous avons remarqué que les SensorTag n'arrivent pas à rester 5 minutes en advertisement, ils s'éteignent après environ 2 minutes. Cela est très embêtant car nous ne pouvons plus nous connecter avec eux.

Afin de résoudre ce problème, nous avons réglé une boucle d'attente qui va se connecter puis se déconnecter à chacun des deux capteurs toutes les minutes pendant 5 minutes. Cela permet de garder les SensorTag éveillés.

3 Communication LoRaWAN

Pour la communication LoRaWAN, nous nous sommes basés sur l'exemple *LoRaWAN_08_join_otaa_send_unconfirmed.pde*. Le join est fait une seule fois au début du programme. Les packets envoyés ne sont pas confirmés. Nous avons configuré notre *Device EUI* comme étant 11-12-13-14-15-16-17-18

Une fois les données des capteurs collectées, elles sont regroupées dans une seule trame au format JSON. La trame créée est ensuite convertie en sa représentation ASCII hexadécimale, car le récepteur n'accepte pas d'autre format. Puis on termine par l'envoi de la trame et l'on recommence le tout après 5 minutes.

3.1 Tests de la communication

Voici une capture d'écran du moniteur série de Waspnote. On voit que l'on rejoint le réseau, collecte des données et qu'un envoi est fait.

```

Application started
1. Switch ON OK
2. Device EUI set OK
3. Application EUI set OK
4. Application Key set OK
5. Save configuration OK
JOINTOTAA: Module answer error
JOINTOTAA: Module answer error
JOINTOTAA: Message sent
OTAA joined
Lum: 83
hum: 43
temp: 26
Message sent

```

FIGURE 4 – Moniteur série de Wasmote

Toutes les données envoyées sont visibles dans la fenêtre de debug (http://192.168.32.4/debug_window.php). Les packets que nous avons envoyés sont encadrés en rouge. On voit que les cinq minutes entre les trames sont bien respectées, on est à 5 minutes et environ 14 secondes entre chaque envoi.

-- LoRaWAN Debug Window --

Timestamp	Device Address	Payload
2016-06-09T08:08:20.532986Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:606;hum:48;temp:25 - irr:0;id:1;lum:569;hum:47;temp:26
2016-06-09T08:05:43.650467Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:196;hum:48;temp:25 - irr:0;id:1;lum:195;hum:47;temp:26
2016-06-09T08:05:24.066209Z	11-12-13-14-15-16-17-18	["temp":26,"hum":47,"lum":40,"irr":0]
2016-06-09T08:04:25.728395Z	01-02-03-04-05-06-07-30	temp:26, hum:51, lum:268, irr:1
2016-06-09T08:04:25.495721Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:215;hum:48;temp:25 - irr:0;id:1;lum:214;hum:47;temp:26
2016-06-09T08:03:07.181987Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:751;hum:48;temp:25 - irr:0;id:1;lum:736;hum:47;temp:26
2016-06-09T08:01:48.936078Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:805;hum:48;temp:25 - irr:0;id:1;lum:794;hum:47;temp:26
2016-06-09T08:01:45.480332Z	de-ad-be-ef-de-ad-be-ef	(temp:27)
2016-06-09T08:00:59.689206Z	de-ad-be-ef-de-ad-be-ef	(temp:27)
2016-06-09T08:00:30.702480Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:275;hum:48;temp:25 - irr:0;id:1;lum:271;hum:47;temp:26
2016-06-09T08:00:10.238332Z	11-12-13-14-15-16-17-18	["temp":25,"hum":46,"lum":289,"irr":0]
2016-06-09T07:59:53.629948Z	de-ad-be-ef-de-ad-be-ef	(temp:27)
2016-06-09T07:57:54.063359Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:243;hum:48;temp:25 - irr:0;id:1;lum:245;hum:47;temp:26
2016-06-09T07:56:35.677062Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:307;hum:48;temp:25 - irr:0;id:1;lum:304;hum:47;temp:26
2016-06-09T07:55:24.886071Z	de-ad-be-ef-de-ad-be-ef	(temp:0)
2016-06-09T07:55:16.962457Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:374;hum:48;temp:25 - irr:0;id:1;lum:377;hum:48;temp:26
2016-06-09T07:55:05.432592Z	de-ad-be-ef-de-ad-be-ef	(temp:0)
2016-06-09T07:54:55.677187Z	11-12-13-14-15-16-17-18	["temp":25,"hum":47,"lum":132,"irr":0]
2016-06-09T07:53:58.564711Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:404;hum:48;temp:25 - irr:0;id:1;lum:412;hum:48;temp:26
2016-06-09T07:52:40.076937Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:448;hum:48;temp:25 - irr:0;id:1;lum:388;hum:48;temp:25
2016-06-09T07:51:37.201427Z	01-02-73-74-05-06-07-08	["temperature":00,"humidity":00,"luminosity":128,"irrigation":1]
2016-06-09T07:51:21.681054Z	01-02-03-04-05-06-07-99	irr:0;id:0;lum:531;hum:48;temp:25 - irr:0;id:1;lum:529;hum:48;temp:25
2016-06-09T07:50:12.169925Z	01-02-73-74-05-06-07-08	["temperature":00,"humidity":00,"luminosity":118,"irrigation":1]
2016-06-09T07:49:38.445950Z	11-12-13-14-15-16-17-18	["temp":25,"hum":47,"lum":55,"irr":0]

FIGURE 5 – Réception des packets

4 Application client : agroMQTTClient

Un petit client a été codé pour le pur plaisir d'utiliser le langage Go, souvent appelé Golang, de chez Google¹. 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.

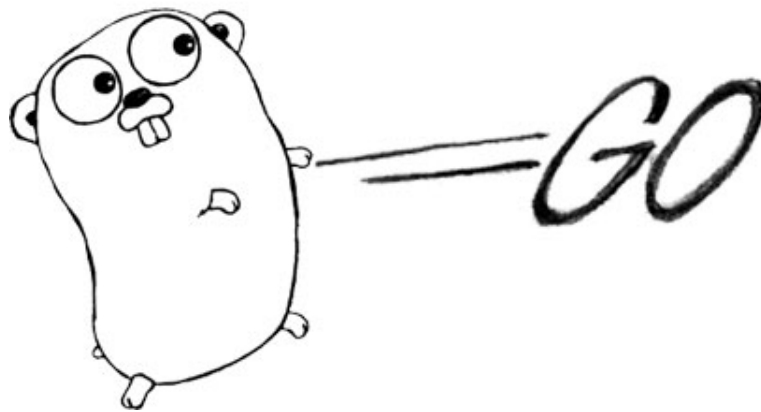


FIGURE 6 – Logo du langage de Google, Golang

Le projet de communication M2M *paho*² 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/>

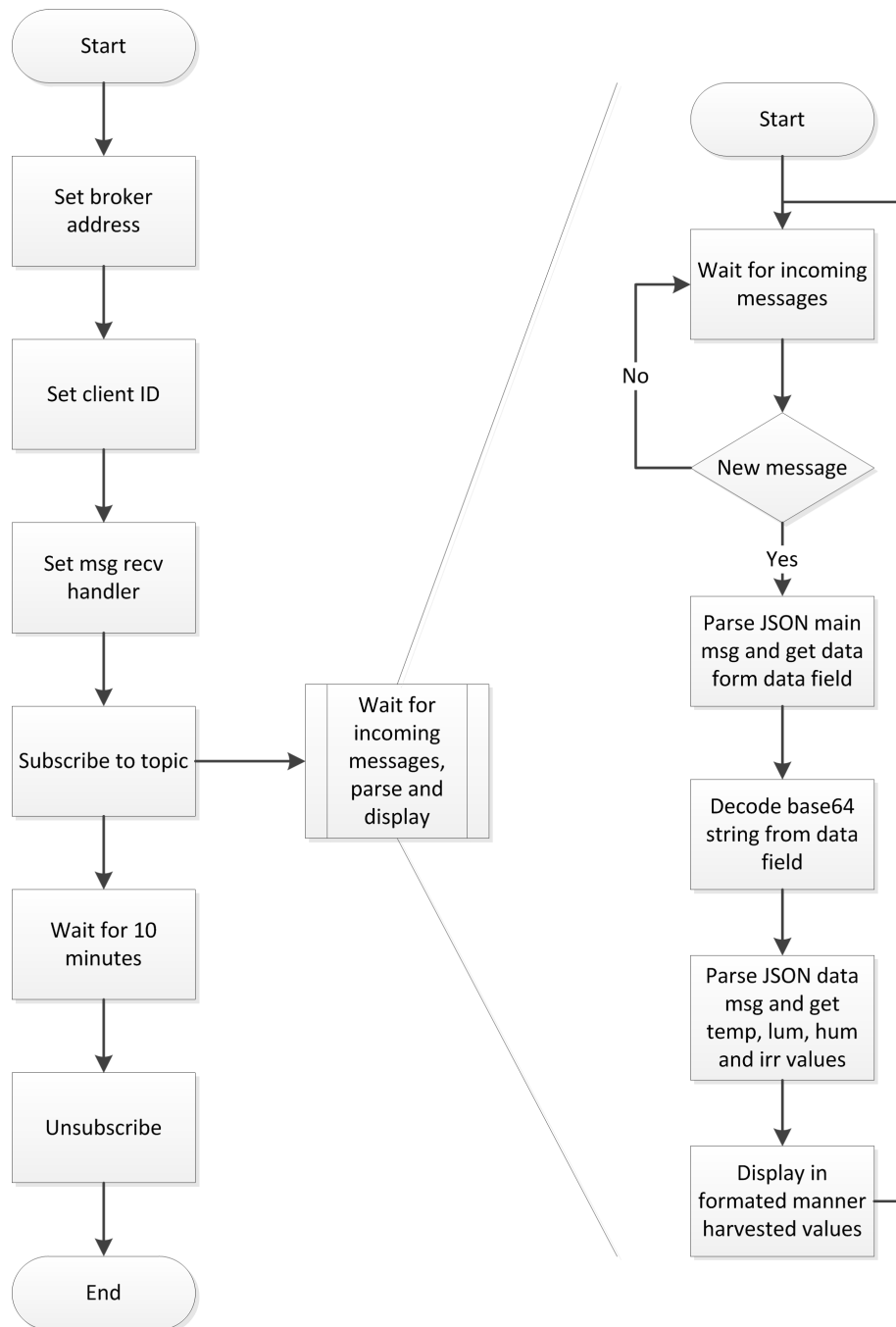


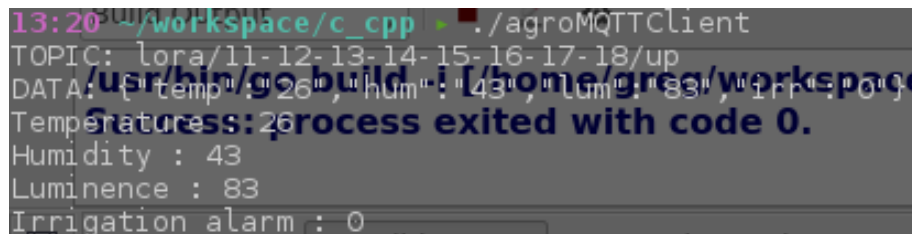
FIGURE 7 – Étapes réalisées par l'application

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.

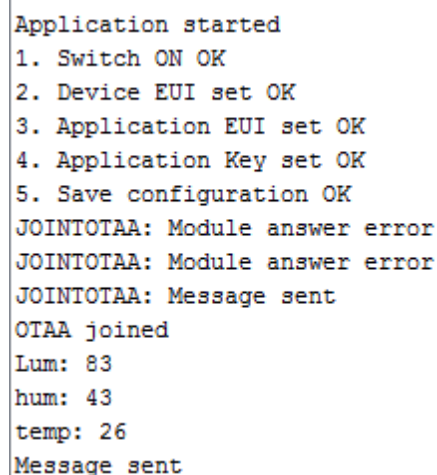
On récupère ces données et on les affiche simplement dans la console.



```
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 8 – Réception et affichage d'un message

On voit que ces données correspondent à celles envoyée par notre carte.



```
Application started
1. Switch ON OK
2. Device EUI set OK
3. Application EUI set OK
4. Application Key set OK
5. Save configuration OK
JOINTOTAA: Module answer error
JOINTOTAA: Module answer error
JOINTOTAA: Message sent
OTAA joined
Lum: 83
hum: 43
temp: 26
Message sent
```

FIGURE 9 – Moniteur série Waspnote

5 Conclusion

Nous avons pu implémenter et tester tous les points demandés par le cahier des charges. Nous avons même fait la partie client, optionnelle.