Université d’été

Smart Windows

Raed Abdennadher – Orphée Antoniadis – Steven Liatti

ITI1 Septembre 2016

Table des matières

[Introduction 3](#_Toc461715079)

[Waspmote 4](#_Toc461715080)

[Communication avec les Capteurs 4](#_Toc461715081)

[Envoi des données par WiFi 4](#_Toc461715082)

[Interprétation des résultats 4](#_Toc461715083)

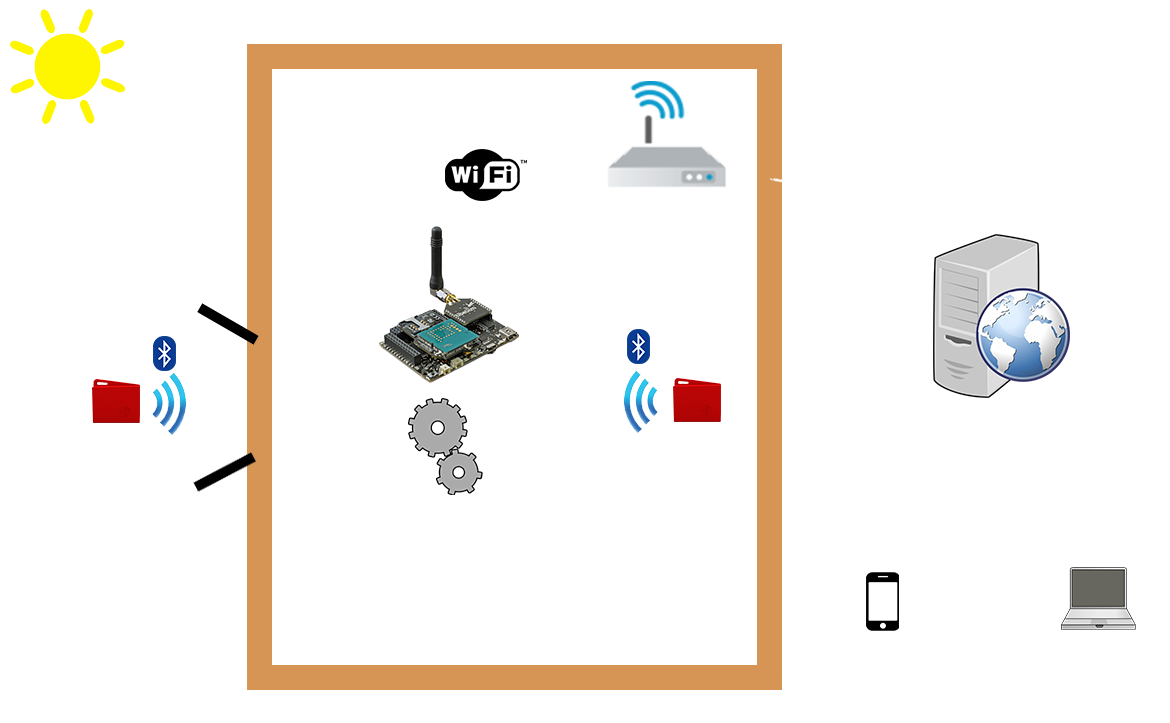
[Schéma électrique 5](#_Toc461715084)

[Serveur 5](#_Toc461715085)

Introduction

Dans le cadre de notre Université d’été, nous avons eu à développer un projet sur l’Internet des objets. Pour ce faire, nous avons eu à disposition une carte Waspmote ainsi que plusieurs modules de communication se connectant sur la carte. Nous avons rapidement imaginé une application dans la domotique. L’idée était de mettre en place tout un système d’ouverture automatique des fenêtres et des stores pour l’aération et la luminosité d’une pièce.

Nous avons donc imaginé placer un SensorTag servant de capteur de température à l’extérieur et un autre à l’intérieur servant aussi de capteur de température mais aussi de capteur de luminosité. Vient s’ajouter aux deux SensorTags, un anémomètre pour la capture de la vitesse du vent. Les SensorTags communiquent par Bluetooth, il a donc fallu utiliser un module BLE connectable directement sur la Waspmote. L’anémomètre est directement branché sur un port analogique de la carte. Les données reçues sont ensuite envoyées par wifi sur un serveur distant sur lequel un utilisateur peut se connecter et voir les données sous la forme de graphiques. L’utilisateur peut aussi choisir de mettre le système en mode manuel et contrôler l’ouverture des fenêtres et des stores directement depuis une interface web.



Fermer

Ouvrir

Calcul…

Figure 1: Schéma de la situation

Waspmote

Communication avec les Capteurs

Pour récupérer les données reçues par les SensorTags, nous avons d’abord utilisé le code d’exemple de l’éditeur Waspmote de connexion Bluetooth par le module BLE. Nous avons juste eu à retrouver les adresses MAC des senseurs pour pouvoir se connecter dessus. Nous avons ensuite utilisé un autre code d’exemple pour récupérer les données par notifications. Cette méthode permet de récupérer une donnée par seconde et non en continu pour éviter les erreurs. Les données reçues étaient en hexadécimal mais un code de conversion est disponible sur l’API de Waspmote. Toutes ces fonctions mises ensemble nous ont permis de lire les valeurs de température de luminosité envoyées par les senseurs. De plus nous avons décidé de faire la moyenne des valeurs lues par la Waspmote afin de n’envoyer qu’une seule valeur au serveur.

L’anémomètre étant directement connecté à la carte nous avons juste eu à lire le signal analogique sur le bon pin. Nous avons quand même eu à convertir cette donnée car c’est une valeur de tension. Une simple règle de 3 permet de convertir la valeur de tension en résultat exploitable ().

Envoi des données par WiFi

Pour la communication par WiFi entre la Waspmote et le serveur, nous avons été obligés d’utiliser un module WiFly. Ce module communique par UART avec la carte. Nous avons donc eu à configurer le WiFly sur TeraTerm et ouvrir les ports UART de la carte. Un code permet donc d’écrire sur les ports UART de la carte puis le WiFly vient les lire et les envois en TCP/IP sur le réseau. Un serveur connecté sur le réseau peut ensuite venir lire les données envoyées par le WiFly (dans notre cas le serveur est un Raspberry Pi).

Interprétation des résultats

Les données étant lues et envoyées, nous avons eu à penser à un algorithme d’ouverture automatique des fenêtres et des stores. Une simple comparaison entre la température intérieure et extérieure a finalement suffi. Si la température intérieure est supérieure à la température extérieure mais aussi supérieure ou égale à 25°C, la fenêtre s’ouvre. De plus, nous avons rajouté la condition du vent. S’il y a du vent, la fenêtre s’ouvre, même si la température extérieure est plus élevée.

Pour l’ouverture des stores nous avons fixé un seuil d’intensité lumineuse. Si l’intensité lumineuse est trop basse comparée à ce seuil (plus de 10 lux de différence), les stores s’ouvrent jusqu’à ce que le seuil soit atteint. Si le seuil n’est jamais atteint, les stores s’ouvrent complètement. La logique est inversée si l’intensité lumineuse est trop élevée. Nous avons-nous même fixé ce seuil à 150 lux en utilisant les résultats recueillis et en estimant quelle intensité lumineuse est agréable dans une pièce. Nous voulions faire en sorte que l’utilisateur puisse choisir ce seuil sur l’interface web mais nous ne l’avons pas fait faute de temps.

Schéma électrique

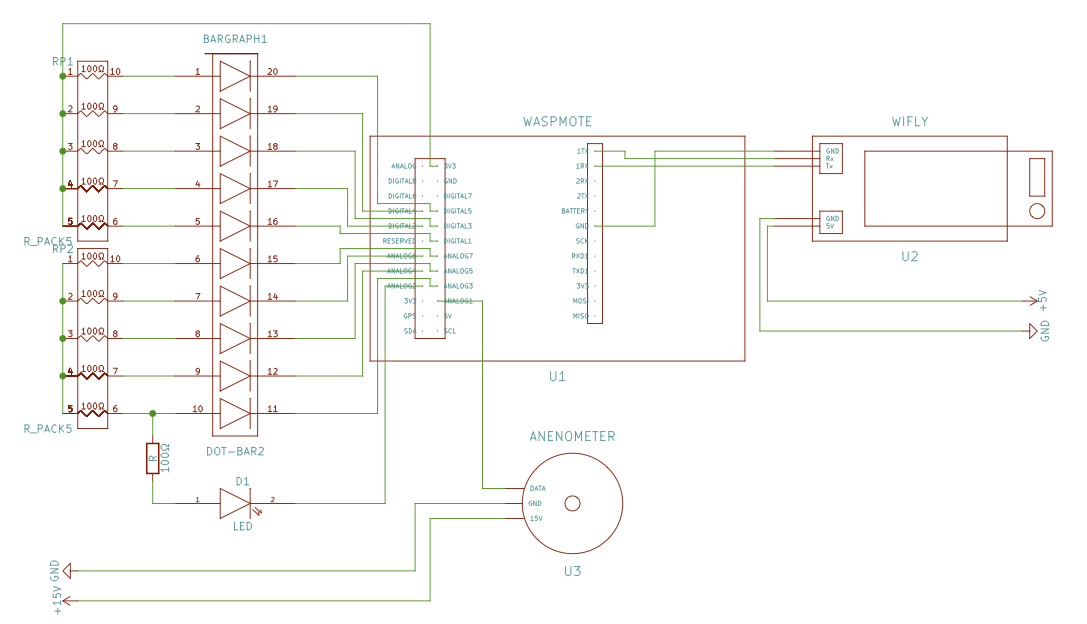


Figure 2: Schéma électrique

Serveur

Notre serveur est un Raspberry Pi 3 avec Raspbian (une variante de Debian) installée dessus. Plusieurs étapes s’exécutent sur le serveur : réception des données provenant du WiFly, enregistrement de ces données en base de données, serveur Apache pour notre interface web et envoi de données en retour au WiFly.

Code Python

Nous avons utilisé python pour ouvrir les sockets communicants avec la Waspmote et le serveur web. Pour celui de la Waspmote, nous avions initialement essayé en PHP, mais le socket ne recevait rien (sûrement un mauvais code de notre part). Le code python se divise en 3 threads : un thread principal enregistrant les valeurs de température, lumière et vent toutes les 10 minutes, un autre thread ouvrant un socket écoutant la Waspmote et un dernier lié à la base de données. Le socket qui écoute la Waspmote fonctionne ainsi : si le mode automatique est actif et qu’un changement d’état (fenêtre, store) est détecté, il enregistre en base de données les changements à la date et heure donnée. Un autre socket