Rapport de projet individuel

Sujet 61 : Intégration de capteurs Arduino dans une plate-forme de mobile crowdsourcing

Projet de second semestre de Master Informatique

Janvier à Mai 2017

Auteur :

— Moncef AOUDIA

Encadrants :

— Romain ROUVOY
— Antoine VEUILLER

Remerciements

Je tiens à remercier Mr VEUILLER de m'avoir accueilli dans l'équipe SPIRALS et pour son suivi durant le projet.

Enfin, je tiens à remercier Mr ROUVOY pour ses explications, son suivi et son aide durant la réalisation de ce projet et la rédaction de ce rapport.

Résumé

Avec l'émergence de l'Internet des Objets , de nombreux équipements «connectés» ont vu le jour (bracelets, capteurs domotiques, etc.) et peuvent se connecter aux téléphones des usagers pour synchroniser leurs données. Le téléphone ne joue plus uniquement le rôle de capteur mais aussi de relais de l'information.

Dans le cadre de mes études, je participe à un projet visant étudier le cas de la technologie Arduino qui permet de concevoir ses propres objets connectés. En particulier, nous souhaitons offrir à la communauté Arduino la possibilité de publier les données issues de n'importe quel capteur (gaz, lumière, son, etc.) sur l'Internet via une connexion Internet qui serait mise à disposition par le téléphone (e.g., via son interface Bluetooth). L'objectif est donc de développer un kit de développement (SDK) pour Arduino qui permette de publier des données de manière passive (à la demande du téléphone) ou réactive (à l'initiative du capteur Arduino). Ce kit de développement sera conçu au dessus de la technologie JavaScript pour IOT et illustré sur le cas d'une application de surveillance de la qualité de l'air qui pourra être déployée à l'échelle de l'université.

Afin de développer une solution pour la publication des données collectées, je devrai implémenter, une librairie qui gère des capteurs de qualité de l'air de type MQ avec le langage C et la porter en JavaScript par la suite. Ensuite, je devrai coder un serveur web (Wifi et Ethernet) afin de l'utiliser pour envoyer les données collectées par les capteurs de qualité de l'air dans le but de les partager sous format de données JSON, dans un premier temps en utilisant le langage C et puis le JavaScript. La troisième tâche consistera à faire transiter les données collectées par les capteurs en utilisant la technologie Bluetooth, cette partie sera implémentée en langage C.

Pour cela, je devrai prendre en main le matériel mis à disposition et les différentes technologies liées aux composants. Je suis confronté à de nombreux problèmes liés au manque de documentation et au long processus de débogage, car quand on travaille sur du matériel, il n'y a pas de débogueur performant pour m'aider. Je devrai trouver une solution performante qui s'adapte aux microcontrôleurs de type Arduino et rendre mon projet modulable dans le but de faciliter la prise en main et l'intégration de nouveaux capteurs.

Table des matières

Table des figures

1 Introduction

Pour mes études de master informatique à l'université de Lille 1, je participerai à un projet proposé par l'équipe SPIRALS [1] de l'INRIA [2] de Lille de recherche afin de découvrir ce milieu. Étant intéressé par le domaine des systèmes distribués et intergiciels et les objets connecté, j'ai choisi ce projet afin mettre en pratique mes compétences théoriques acquises durant mon cursus universitaire et avoir une expérience professionnelle.

1.1 Présentation

Pour ce projet, je suis amené à développer une solution afin de collecter les données issues de capteurs de qualité de l'air et de pouvoir partager les données sur la plateforme APISENSE pour être exploitées par la suite. Cette solution a pour but de faciliter la collecte et le partage de données en utilisant la technologie Arduino via :

- Serveur Web Wifi : le microcontrôleur pourra collecter les données via ses capteurs et les partager avec une connexion Wifi.
- Serveur Web Ethernet : le microcontrôleur pourra collecter les données via ses capteurs et les partager avec une connexion Ethernet.
- Bluetooth : le microcontrôleur pourra collecter les données via ses capteurs et les partager avec une connexion Bluetooth.

Ce type de projet utilise différentes technologies et exige du matériel assez spécifique.

1.2 Contexte

Pour la réalisation de ce projet, je travaille avec l'équipe SPIRALS qui mène des activités de recherche dans les domaines des systèmes répartis et des sciences du logiciel. Ils ont pour but d'introduire plus d'autonomie dans les mécanismes d'adaptation des systèmes logiciels, en particulier, afin d'assurer la transition des systèmes adaptatifs vers les systèmes autoadaptatifs. Ils visent plus particulièrement deux propriétés : l'autoguérison et l'auto-optimisation. Avec l'autoguérison, ils ont pour but d'étudier et d'adapter des solutions de fouille de données et d'apprentissage à la conception et la mise en œuvre de systèmes logiciels, plus particulièrement en vue de la réparation automatique des systèmes logiciels. Avec l'auto-optimisation, ils ont pour but de partager, collecter et analyser les comportements dans un environnement réparti afin de continuellement adapter, optimiser et maintenir en fonctionnement des systèmes logiciels et d'aller vers l'obtention de systèmes distribués éternels.

Le projet sur lequel je travaille est effectué à l'INRIA de Lille et il s'étale sur la durée de tout le deuxième semestre du master informatique. Je suis supervisé par Mr VEUILLER à l'INRIA et Mr ROUVOY à l'université de Lille 1.

1. Self-adaptation for distributed services and large software systems
2. Institut national de recherche en informatique et en automatique

1.3 Problématiques

Le défi est lié au grand nombre de capteurs de gaz disponible et au manque de documentation.Le travail de recherche sera essentiel pour l'aboutissement du projet.

Il faut pour cela concentrer mes efforts sur les capteurs de type MQ afin de de réaliser une librairie Arduino qui les supportent. De plus, le projet doit être modulable afin de pouvoir intégrer de nouveaux capteurs facilement.

Au niveau de la partie transfert de données, les différentes technologies utilisées rendent le travail plus difficile, car entre la version Wifi, Ethernet et Bluetooth, c'est une implémentation avec deux approches différentes à cause des restrictions liées au Bluetooth.

Les langages de programmation utilisés demandent une prise en main particulière, car le langage C pour Arduino est une version allégée du C et ça s'applique aussi à la partie JavaScript.

L'optimisation mémoire est très importante dans la partie développement, on risque vite de saturer cette dernière si on ne fait pas attention à notre code et ça peut avoir des répercussions sur les performances du microcontrôleur sur le court ou le long terme.

L'écriture de la documentation du projet et des différents Readme [3] prend du temps et souvent elle contient des manipulations très poussées qui peuvent paraître difficile à réaliser dans un premier temps.

1.4 Objectifs du projet

Le premier objectif est d'implémenter une libraire Arduino qui gère les capteurs de qualité de l'air de type MQ d'une manière modulable, car elle supporte 3 capteurs au début (MQ2, MQ6, MQ8), par la suite on doit pouvoir ajouter d'autre capteurs facilement.
Le deuxième objectif consiste à développer un serveur Web qui pour Arduino, qui fera le lien entre les capteurs de gaz et la plateforme APISENSE, entre cette dernière et le serveur on peut utiliser un téléphone ou tout autre appareil compatible.
Le dernier objectif est d'implémenter les mêmes fonctionnalités du projet Arduino C en JavaScript, pour des raisons d'interopérabilité, le projet sera donc compatible pas avec Arduino seulement, mais avec tous les microcontrôleurs supportés par le moteur JavaScript.
Pour répondre aux différentes problématiques évoquées en haut, je devais d'abord voir le matériel mis à disposition pour la réalisation du projet et lire la documentation et les

3. Un fichier readme (en français, lisezmoi) est un fichier contenant des informations sur les autres fichiers du même répertoire

fiches de données des capteurs, ainsi que voir les différents projets réalisés sous Arduino disponible sur Internet.
Je détaillerai toutes les démarches pour résoudre les problématiques et j'expliquerai la solution finale que j'ai implémentée dans le projet.

2 Prérequis et environnement de développement

Dans cette partie, je présentai les outils et technologies utiliser durant le processus de développement du projet.

2.1 Matériels

2.1.1 Arduino Uno

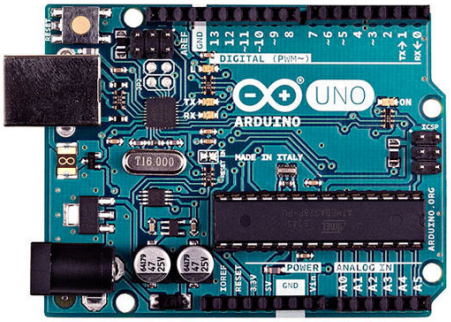


FIGURE 1 – Microcontrôleur Arduino Uno

Arduino/Genuino Uno est un microcontrôleur basé sur le soc ATmega328P. Il a 14 pins entrée/sortie pins (dont 6 qui peuvent être utilisés comme prise de courant 5v), 6 entrées analogiques, 16 MHz de fréquence, une connexion USB, une prise de courant, une tête ICSP et un bouton remise a zéro.

2.1.2 Shield WiFi Arduino

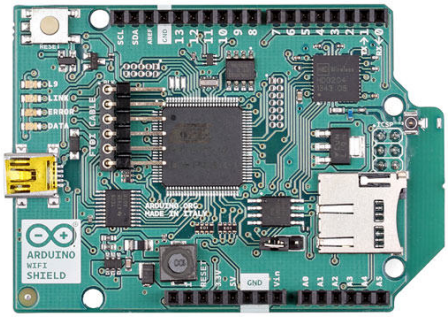


FIGURE 2 – Shield Wifi Arduino

L'Arduino WiFi Shield [4] permet de relié l'Arduino à Internet sans fil.
— Nécessite un Arduino pour fonctionner.
— Tension de service 5V (fournie par la carte Arduino).
— Arduino Due compatible.
— Connexion via : réseaux 802.11b/g.
— Types de cryptage : WEP et WPA2 personnels.
— Connexion avec Arduino sur le port SPI.
— Slot micro SD embarqué.
— En-têtes ICSP.
— Connexion FTDI pour le débogage en série du bouclier WiFi.
— mini-USB pour la mise à jour du micrologiciel du bouclier WiFi.

4. "Bouclier" en anglais, c'est une carte conçue pour et ajouter des fonctionnes se poser sur l'Arduino

2.1.3 Shield bluetooth Arduino

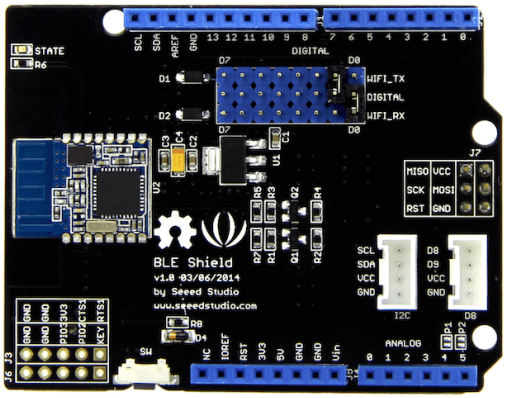


FIGURE 3 – Shield Bluetooth Arduino

Le Bluetooth Shield intègre un module Serial Bluetooth. Il peut être facilement utilisé avec Arduino pour une communication série sans fil transparente. on peut choisir deux broches d'Arduino D0 à D7 en tant que Ports Serial [5] Serial pour communiquer avec Bluetooth Shield (D0 et D1 est Hardware Serial Port). La shield comporte également deux connecteurs Grove (l'un est numérique, l'autre est analogique) pour installer des modules Grove.

2.1.4 Capteurs de gaz MQ2, MQ6, MQ8

FIGURE 4 – Capteurs de gaz MQ2,MQ6,MQ8

— Le capteur MQ2 peut être utilisé pour la détection de H2, LPG [6], CH4, CO, Alcool, fumée ou Propane.

5. Transmission série
6. Liquefied petroleum gas

— Le capteur MQ6 peut être utilisé pour la détection de LPG ou CH4.
— Le capteur MQ8 peut être utilisé pour la détection de H2.

2.1.5 ESP8266

FIGURE 5 – ESP8266-12E

L'ESP8266 est un circuit intégré à microcontrôleur avec connexion Wifi développé par le fabricant chinois Espressif.
Voici ses caractéristiques techniques :

— 32-bit RISC CPU : Tensilica Xtensa LX106, 80 MHz 64 KiB d'instructions RAM, 96 KiB de RAM de données.
— QSPI flash externe - 512 KiB jusqu'à 4 MiB (support jusqu'à 16 MiB), IEEE 802.11 b/g/n Wifi.
— TR switch intégré, balun [7], LNA, amplificateur de signal Wifi et compatibilité avec les réseaux WEP or WPA/WPA2, 16 GPIO pins ,SPI, I2C,I2S interfaces avec DMA (partage de pins avec GPIO).
— UART dans les pins dédies , 10-bit ADC.

J'ai présenté en haut les matériels le plus importants pour la réalisation du projet, il y a toute la connectique derrière que je ne présenterai pas dans ce rapport.

7. Un balun est un circuit électrique utilisé pour effectuer la liaison entre : une ligne de transmission symétrique (ligne bifilaire ou lignes imprimées parallèles) et une ligne de transmission asymétrique (câble coaxial ou ligne imprimée au-dessus d'un plan de masse)

2.2 Logiciels

2.2.1 Arduino IDE

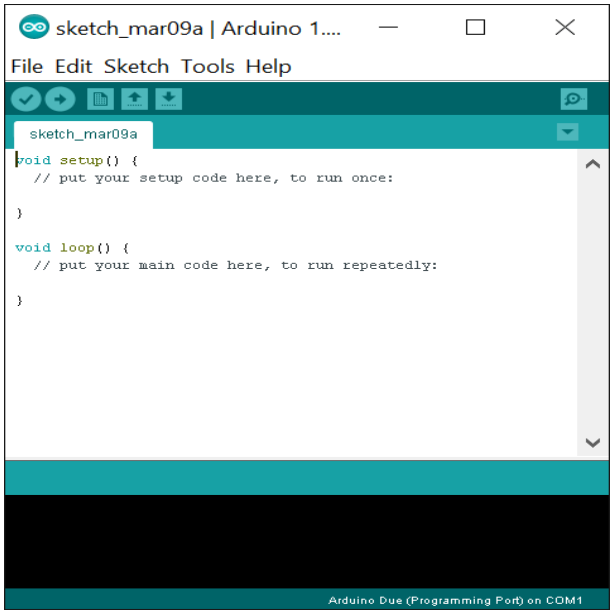


FIGURE 6 – Arduino Web IDE

L'IDE est bon pour un début mais ensuite, je me suis rendu compte qu'il est pas assez puissant lors de la phase de débogage. J'ai opté pour la solution Arduino Cmake qui est parfaite pour les besoins de mon projet.

2.2.2 Arduino CMake

Arduino est une plate-forme de développement formidable, facile à utiliser. Il a tout ce dont un débutant devrait avoir besoin. L'IDE Arduino simplifie beaucoup de choses pour l'utilisateur standard, mais si vous êtes un programmeur professionnel, l'IDE peut paraître simpliste et restrictif.

Un inconvénient majeur de l'IDE d'Arduino est que je ne peux rien faire sans celui-ci, ce qui pour moi est une destruction totale. C'est pourquoi j'ai utilisé un système de construction alternatif pour Arduino.

CMake est un excellent système de construction multiplateforme qui fonctionne sur pratiquement n'importe quel système d'exploitation. Avec cela, on n'est pas contraint à un seul système de construction. CMake me permet de générer le système de construction qui correspond à mes besoins. Il peut générer tout type de système de construction, à partir de simples Makefiles, pour compléter des projets pour Eclipse, Visual Studio, XCode, etc.

Le système de construction Arduino CMake s'intègre étroitement au SDK Arduino qui doit être installé au préalable (Si l'IDE Arduino est installé sur la machine implique

que le SDK est déjà installé sur votre système).

Pour pouvoir compiler sous Linux il faudra avoir les logiciel suivant sur la machine de développement :

— gcc-avr - AVR GNU GCC compilateur
— binutils-avr - AVR outils binaire
— avr-libc - AVR C librairie
— avrdude - utiliser pour charger le firmware.

2.2.3 Espruino IDE

Espruino est un interpréteur JavaScript pour les microcontrôleurs. Il est conçu pour les appareils dotés d'un flash de 128kB et d'une RAM de 8kb.

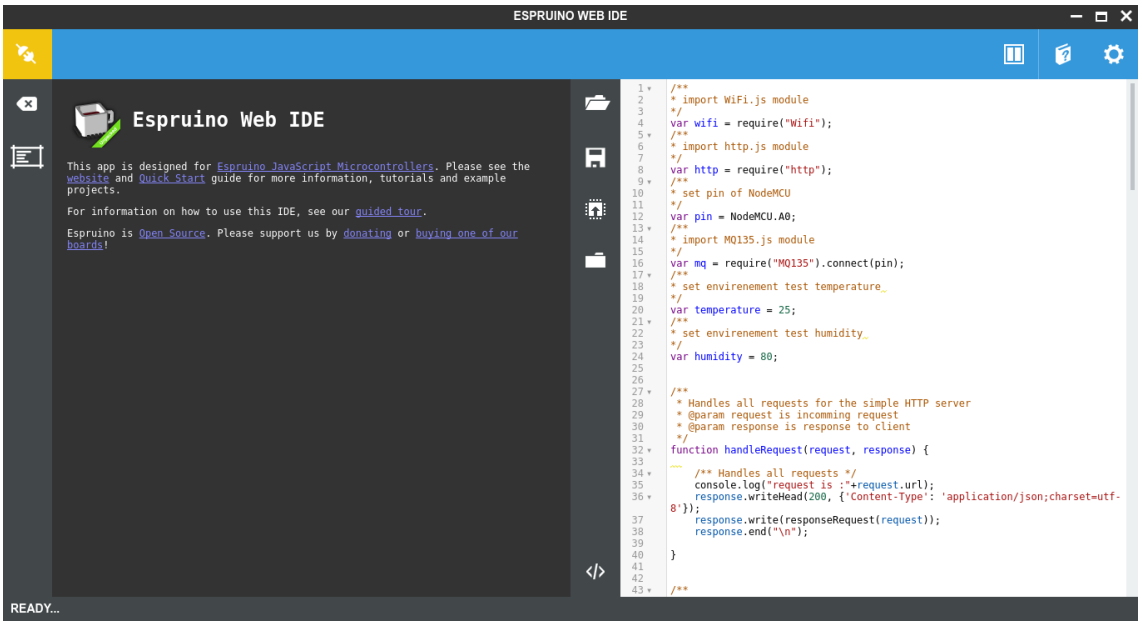


FIGURE 7 – Espruino Web IDE

L'IDE peut être intégré au navigateur Chrome ou Chromium (version open source de chrome) sous la forme d'une extension qu'on peut installer du Chrome web store ou à partir du dépôt GitHub officiel.

il est disponible également sous la forme d'un module Node.js pour les puristes de la ligne de commande.

```
USAGE: espruino ...options... [file_to_upload.js]

  -h,--help               : Show this message
  -j [job.json]           : Make or load options from JSON. See 'job file' section below
  -c,--color              : Color mode,
  -v,--verbose            : Verbose
  -q,--quiet              : Quiet - apart from Espruino output
  -m,--minify             : Minify the code before sending it
  -w,--watch              : If uploading a JS file, continue to watch it for
                              changes and upload again if it does.
  -p,--port /dev/ttyX
  -p,--port aa:bb:cc:dd:ee : Specify port(s) or device addresses to connect to
  -b baudRate             : Set the baud rate of the serial connection
                              No effect when using USB, default: 9600
  --no-ble                : Disable Bluetooth Low Energy (used by default if the 'bleat' module exists)
  --list                  : List all available devices and exit
  -t,--time               : Set Espruino's time when uploading code
  -o out.js               : Write the actual JS code sent to Espruino to a file
  -f firmware.bin         : Update Espruino's firmware to the given file
                              Espruino must be in bootloader mode
                              Optionally skip N first bytes of the bin file,
  -e command              : Evaluate the given expression on Espruino
                              If no file to upload is specified but you use -e,
                              Espruino will not be reset
#
If no file, command, or firmware update is specified, this will act
as a terminal for communicating directly with Espruino. Press Ctrl-C
twice to exit.
```

FIGURE 8 – Espruino tools sous ligne de commande

2.2.4 Esptools

Esptools est un outil utilisé pour flasher différentes versions d'ESP, le chargement du micrologiciel ESPRUINO permet l'utilisation sur le ESP8266.

```
$ esptool.py
esptool.py v1.3
usage: esptool [-h] [--port PORT] [--baud BAUD]
               {load_ram,dump_mem,read_mem,write_mem,write_flash,run,image_info,make_image,elf2image,read_mac,chip_id,flash_id,read_flash,verify_flash
,erase_flash,version}
               ...

esptool.py v1.3 - ESP8266 ROM Bootloader Utility

positional arguments:
  {load_ram,dump_mem,read_mem,write_mem,write_flash,run,image_info,make_image,elf2image,read_mac,chip_id,flash_id,read_flash,verify_flash,erase_flash,
version}
                        Run esptool {command} -h for additional help
    load_ram            Download an image to RAM and execute
    dump_mem            Dump arbitrary memory to disk
    read_mem            Read arbitrary memory location
    write_mem           Read-modify-write to arbitrary memory location
    write_flash         Write a binary blob to flash
    run                 Run application code in flash
    image_info          Dump headers from an application image
    make_image          Create an application image from binary files
    elf2image           Create an application image from ELF file
    read_mac            Read MAC address from OTP ROM
    chip_id             Read Chip ID from OTP ROM
    flash_id            Read SPI flash manufacturer and device ID
    read_flash          Read SPI flash content
    verify_flash        Verify a binary blob against flash
    erase_flash         Perform Chip Erase on SPI flash
    version             Print esptool version

optional arguments:
  -h, --help            show this help message and exit
  --port PORT, -p PORT  Serial port device
  --baud BAUD, -b BAUD  Serial port baud rate used when flashing/reading
```

FIGURE 9 – Esptools

Il rend possible l'utilisation de JavaScript avec tous les microcontrôleurs supportés. il est disponible également sous la forme d'un module Node.js pour les puristes de la ligne de commande.

2.2.5 APISENSE

La plate-forme APISENSE® permet de collecter une grande variété de données à partir des téléphones portables des usagers tout en garantissant le respect de leur vie privée.

FIGURE 10 – APISENSE

Utilisée dans de nombreux domaines, la plate-forme APISENSE® facilite la collecte, le stockage et le traitement des données renseignées par une foule d'usagers contributeurs. Basée sur les standards du web et déployée dans le Cloud, la plate-forme APISENSE® a notamment la capacité de traiter de gros volumes de données qui peuvent être restituées à différentes parties prenantes et sous différentes formes (OpenData, visualisations, etc.).

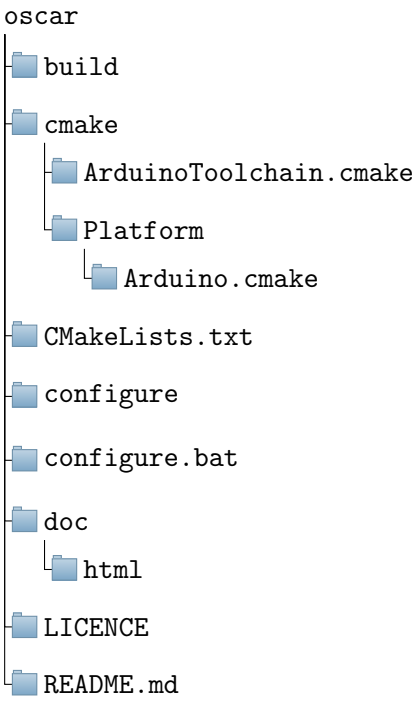
3 Implémentation

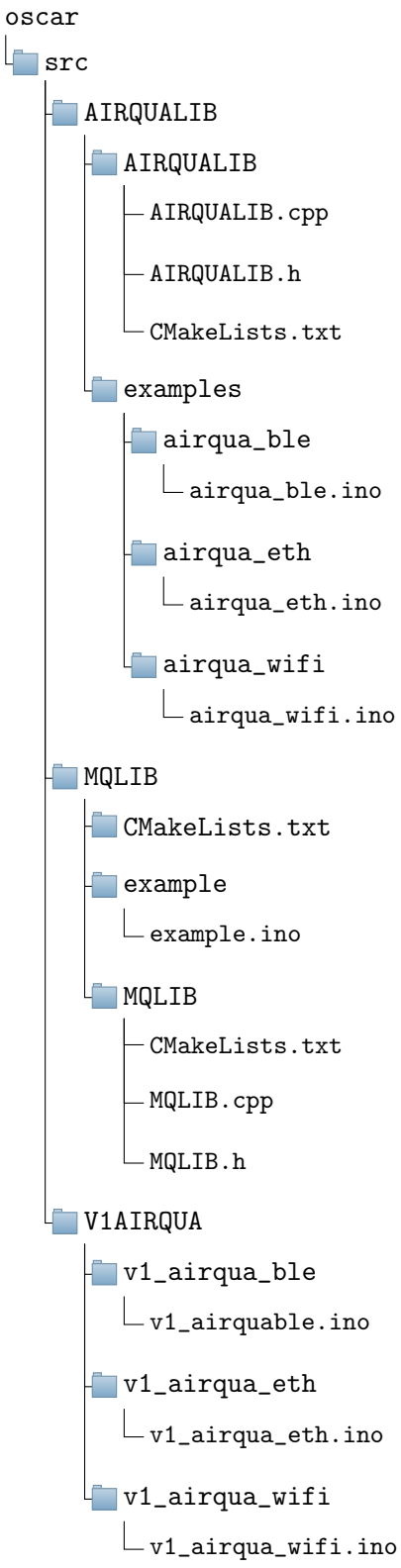
Dans cette partie, j'expliquerai en détail le travail réalisé durant le semestre, la première phase de développement en langage C a pris la majeure partie du temps, pour la partie JavaScript, c'était plutôt de l'adaptation de code.

Tout le code et la procédure de prise en main et d'installation sont disponibles sur le dépôt GitLab du projet et de l'archive jointe à ce rapport.

3.1 Partie Arduino langage C

3.1.1 Structure du projet





Langage	fichiers	espace	commentaire	code
Arduino Sketch	7	273	457	792
C++	2	43	110	149
C/C++ Header	2	66	150	70
CMake	5	38	46	60
TOTAL:	16	420	763	1071

Fichiers	espace	commentaire	code
src/V1_AIRQUA/v1_airqua_ble/v1_airqua_ble.ino	51	100	180
src/V1_AIRQUA/v1_airqua_wifi/v1_airqua_wifi.ino	53	100	164
src/V1_AIRQUA/v1_airqua_eth/v1_airqua_eth.ino	51	69	135
src/AIRQUALIB/examples/airqua_ble/airqua_ble.ino	36	69	107
src/AIRQUALIB/examples/airqua_wifi/airqua_wifi.ino	39	68	90
src/AIRQUALIB/AIRQUALIB/AIRQUALIB.cpp	22	49	88
src/AIRQUALIB/examples/airqua_eth/airqua_eth.ino	34	38	62
src/MQLIB/MQLIB/MQLIB.cpp	21	61	61
src/MQLIB/example/example.ino	9	13	54
src/MQLIB/MQLIB/MQLIB.h	47	111	50
src/AIRQUALIB/examples/CMakeLists.txt	13	14	20
src/AIRQUALIB/AIRQUALIB/AIRQUALIB.h	19	39	20
src/V1_AIRQUA/CMakeLists.txt	10	13	19
src/AIRQUALIB/AIRQUALIB/CMakeLists.txt	6	7	8
src/MQLIB/CMakeLists.txt	5	6	7
src/MQLIB/MQLIB/CMakeLists.txt	4	6	6
TOTAL:	420	763	1071

3.1.2 MQLIB

Dans cette première étape, j'ai commencé par la lecture des fiches techniques des capteurs et de l'Arduino Uno r3. Ensuite, j'ai continué la recherche sur les forums d'Arduino et des capteurs de gaz afin de récolter le maximum d'informations sur le matériels avant la phase de développement.

Le premier programme, appelé aussi croquis [8] Arduino, était très simple et consistait à lire une valeur analogique à partir des différents capteurs, afin de s'assurer de leurs bons fonctionnements.

```
/* Pin sur lequel le capteur est connecter */
int sensorPin = A0;

void setup() {
  /* UART reglages, baudrate = 9600bps.*/
  Serial.begin(9600);
}

void loop() {
  /* Lecture et affichage de la valeur du capteur */
  Serial.print("Value of sensor is: "+analogRead(sensorPin);
  Serial.print("\");
}
```

Ensuite, j'ai implémenté un programme qui permet de lire des valeurs en PPM [9], pour la liste des gaz supportés par les capteurs MQ (MQ2, MQ6, MQ8).
A la fin, j'ai intégré tout le code dans une libraire pour Arduino, qui facilite l'utilisation des capteurs avec un simple import.

```
/* Declaration de la librairie MQLIB pour Arduino */
#ifndef MQLIB_H
#define MQLIB_H

#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif
...
#endif
}
```

j'ai ajouté un croquis qui permet de tester MQLIB et faciliter le travail des développeurs qui veulent l'utiliser ou l'améliorer.

8. Un programme écrit avec IDE pour Arduino s'appelle un croquis
9. Partie par million, fraction valant un millionième

```
void loop() {

  /* Q2 Sensor */
  Serial.print("MQ2 SENSOR");
  Serial.print("LPG:");
  Serial.print(air.displayGasValue(air.sensorA, GAS_LPG));
  Serial.print("ppm");
  Serial.print("    ");
  Serial.print("CO:");
  Serial.print(air.displayGasValue(air.sensorA, GAS_CO));
  Serial.print("ppm");
  Serial.print("    ");
  Serial.print("SMOKE:");
  Serial.print(air.displayGasValue(air.sensorA, GAS_SMOKE));
  Serial.print("ppm");
  Serial.print("\n");

  /* Q6 Sensor */
  Serial.print("MQ6 SENSOR");
  Serial.print("LPGQ6:");
  Serial.print(air.displayGasValue(air.sensorB, GAS_LPGQ6));
  Serial.print("ppm");
  Serial.print("        ");
  Serial.print("CH4::");
  Serial.print(air.displayGasValue(air.sensorB, GAS_CH4));
  Serial.print("ppm");
  Serial.print("\n");

  /* Q8 Sensor */
  Serial.print("MQ8 SENSOR");
  Serial.print("H2:");
  Serial.print(air.displayGasValue(air.sensorC, GAS_H2));
  Serial.print("ppm");
  Serial.print("\n");
  Serial.print("\n");
  delay(200);
}
```

3.1.3 AIRQUA

J'ai implémenté un serveur web (REST) qui utilise MQLIB et peut être interrogé avec de simples requêtes GET, en fonction de la requête envoyée la réponse est renvoyée sous format JSON. Ensuite, j'ai intégré le serveur Web dans une librairie Arduino.

```
...
/** class declaration.*/
class AIRQUA {

...

/**
* constructor3
* @param mq_pin1 - analog input channel you are going to use
* @param mq_pin2 - analog input channel you are going to use
* @param mq_pin3 - analog input channel you are going to use
*/
AIRQUA(uint8_t mq_pin1, uint8_t mq_pin2, uint8_t mq_pin3);

/**
 *  display current sensor value to client.
 *  @param sensorName is String argument which set sensor name.
 *  @param gasTypes is String argument which set sensor gas type.
 *  @param value is String argument which set sensor value.
 *  @return string sensor value
 */
    String sensorValue(String sensorName, String gasTypes, int
   value);


/**
 *  display sensor informations to client in Json format.
 *  @param sensorName is String argument which set sensor name.
 *  @param model is String argument which set sensor model.
 *  @param reference is String argument which set sensor reference
   .
 *  @param unite is String argument which set sensor unite.
 *  @param gasTypes is String argument which set sensor gas type (
   write it in
* Json nested array format ex. "[\"LPG\",\"CO\",\"SMOKE\"]" ).
 *  @return string sensor informations
 */
    String sensorType(String sensorName, String model, String
   reference,String unite, String gasTypes);

/**
 *  display the response to the http request
 *  @param httpRequest is String argument which set http header
   content.
```

```
40  *  @return response is String
41  */
42     String responseRequest(String httpRequest);
43 };
44
45 #endif
```

3.1.3.1 Wifi : Cette version permet d'interroger AIRQUA en utilisant des requêtes GET via une connexion Wifi.

```
1  ...
2  void setup()
3  {
4      /* Initialize serial and wait for port to open: */
5      Serial.begin(9600);
6
7      /* check for the presence of the shield: */
8      if (WiFi.status() == WL_NO_SHIELD)
9      {
10         Serial.println("WiFi shield not present");
11
12     }
13
14     /* check firmware version */
15     String fv = WiFi.firmwareVersion();
16     if (fv != "1.1.0")
17     {
18         Serial.println("Please upgrade the firmware");
19     }
20
21     /* attempt to connect to Wifi network: */
22     while (status != WL_CONNECTED && attemps <= 3)
23     {
24         Serial.print("Attempting to connect to SSID: ");
25         Serial.println(ssid);
26         /* Connect to WPA/WPA2 network. Change this */
27         /* line if using open or WEP network: */
28         status = WiFi.begin(ssid, pass);
29         attemps++;
30         /* wait 10 seconds for connection: */
31         delay(10000);
32     }
33
34     /* start wifi server */
35     server.begin();
36     /* you're connected now, so print out the status: */
37     printWifiStatus();
38
```

```
39    ...
40
41 void printWifiStatus()
42 {
43     /* print the SSID of the network you're attached to: */
44     Serial.print("SSID: ");
45     Serial.println(WiFi.SSID());
46
47     /* print your WiFi shield's IP address: */
48     IPAddress ip = WiFi.localIP();
49     Serial.print("IP Address: ");
50     Serial.println(ip);
51
52     /* print the received signal strength: */
53     long rssi = WiFi.RSSI();
54     Serial.print("signal strength (RSSI):");
55     Serial.print(rssi);
56     Serial.println(" dBm");
57 }
```

3.1.3.2 Ethernet : C'est les mêmes fonctions que le serveur que Wifi sauf pour la partie connectique, il exploite un port Ethernet connecté à l'Arduino pour transférer les données.

```
1 ...
2 /** enter a MAC address and IP address for your controller below.
      */
3 byte mac[] = {
4   0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
5 };
6
7 /** the IP address will be dependent on your local network: */
8 IPAddress ip(192, 168, 1, 127);
9
10 /**
11  * initialize the Ethernet server library
12  * with the IP address and port you want to use
13  * (port 80 is default for HTTP):
14  */
15 EthernetServer server(80);
16 ...
```

3.1.3.3 Bluetooth : J'ai développé un programme qui permet une communication avec un périphérique doté de la technologie Bluetooth, pour mes essais j'ai utilisé un smartphone et un ordinateur. Il permet les mêmes fonctionnalités que les deux versions précédentes, mais la partie communication est différente car l'utilisation du Bluetooth

impose beaucoup de code en plus et davantage de restrictions pour le transfert des données.

```
...
/**
 * for Uno:
 * HM10 TX pin to Arduino Uno pin D6
 * HM10 RX pin to Arduino Uno pin D7
 * for Mega 2560:
 * HM10 TX pin to Arduino Mega 2650 pin D10
 * HM10 RX pin to Arduino Mega 2560 pin D11
 */
SoftwareSerial ble(6,7);

/** buffer to store response */
char buffer[BUFFER_LENGTH];
...

/**
 *  Loop() is where the code that runms ovewr and over goes (your program).
 *  Listen to client requests and returns resposes.
 */
void loop() {

  if (ble.available()) {

    char request[200] = {0};
    int i = 0;
    while (ble.available()){
      char c = ble.read();
      request[i++] = c;
      delay(200);
    }

    Serial.println(request);

    String response = airqua.responseRequest(request);
    /* display response for current client request */
    ble.println(response);

  }
}
```

Les possibilités sont les suivantes :
- Valeur actuelle du capteur.

```
curl --get http://192.168.0.21/sensors/co/value
{"name":"MQ2","gasTypes":"co","value":"0"}
```

- Type du capteur.

```
sparow@debian:~$ curl --get http://192.168.0.21/sensors/h2/type
{"name":"Gas","model":"MQ8","reference":"MQ-8","unite":"PPM","
    gasTypes":["H2"]}
```

- Toutes les valeurs retournées par les capteurs.

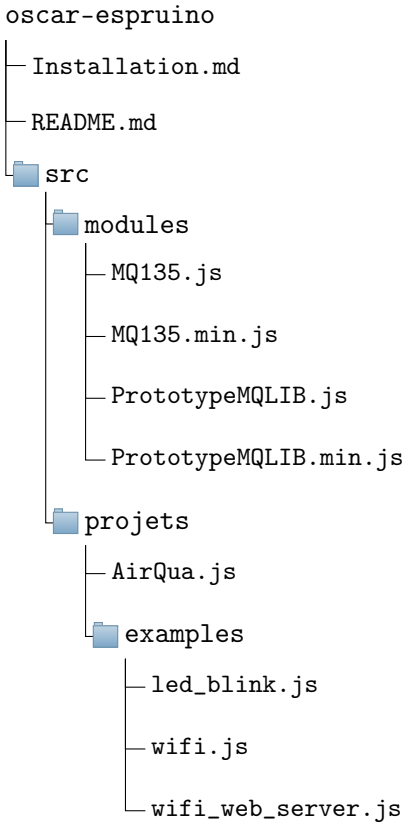
```
sparow@debian:~$ curl --get http://192.168.0.21/sensors/values
{"name":"MQ2","gasTypes":"co","value":"0"}
{"name":"MQ2","gasTypes":"smoke","value":"0"}
{"name":"MQ2","gasTypes":"lpg","value":"0"}
{"name":"MQ6","gasTypes":"lpg","value":"0"}
{"name":"MQ8","gasTypes":"h2","value":"0"}
{"name":"MQ6","gasTypes":"ch4","value":"0"}
```

- Tous les types disponibles.

```
sparow@debian:~$ curl --get http://192.168.0.21/sensors/types
{"name":"Gas","model":"MQ2","reference":"MQ-2","unite":"PPM","
    gasTypes":["LPG","CO","SMOKE"]}
{"name":"Gas","model":"MQ6","reference":"MQ-6","unite":"PPM","
    gasTypes":["LPG","CH4"]}
{"name":"Gas","model":"MQ8","reference":"MQ-8","unite":"PPM","
    gasTypes":["H2"]}
```

3.2 Partie ESP8266 JavaScript

3.2.1 Structure du projet



Langage	fichiers	espace	commentaire	code
JavaScript	8	100	244	274
TOTAL:	8	100	244	274

Fichiers	espace	commentaire	code
src/projects/AirQua.js	26	59	95
src/modules/PrototypeMQLIB.js	44	100	88
src/projects/examples/wifi_web_server.js	7	37	37
src/modules/MQ135.js	20	39	36
src/projects/examples/wifi.js	2	7	9
src/projects/examples/led_blink.js	1	2	5
src/modules/MQ135.min.js	0	0	3
src/modules/PrototypeMQLIB.min.js	0	0	1
TOTAL:	100	244	274

3.2.2 Prototype MQLIB

J'ai commencé par une longue documentation sur le moteur JavaScript ESPRUINO et le microcontrôleur ESP8266.Ensuite, j'ai développé les exemples suivants :

- ledblink.js : permet de de faire clignoté la LED de l'ESP8266 :

```
var ledIsOn = false;
/ * 1000 milliseconds = 0.5 seconds */
var interval = 1000;
setInterval(function(){
  /*D2 is the blue LED on the ESP8266 boards
  Assigns and flips the state on or off */
  digitalWrite(D2, ledIsOn = !ledIsOn);
}, interval);
```

- wifi.js : permet de se connecter à un réseau Wifi :

```
var wifi = require("Wifi");

/**
 * Handles the connection for the wifi.connect call.
 * @param {Error} err
 */
function onConnect(err) {
    if(err) {
        console.log("An error has occured :( ", err.message);
    } else {
        console.log("Connected IP Adress : ", wifi.getIP().ip);
    }
}
wifi.connect("YourNetworkSSID", { password: "NetworkPassword" },
    onConnect);
```

- wifiwebserver.js : c'est un serveur web qui affiche les réseaux Wifi disponibles autour :

```
...

var http = require("http");
...

/**
 * Handles all requests for the simple HTTP server
 *
 * @param {any} request - Incomming request
 * @param {any} response - Response to client
 */
function handleRequest(request, response) {
    //Handles all requests
    response.writeHead(200, {'Content-Type': 'text/html'});
    //Scans for accessible networks
    wifi.scan(function(networks){
        response.end(generatePage("Networks", generateTableForAPs(networks)));
    });
}
...
```

Enfin, j'ai implémenté un prototype en JavaScript de MQLIB en m'inspirant d'un module existant et qui à était tester proprement sur ESPRUINO.

```
exports.connect = function(mq_pin1, mq_pin2, mq_pin3) {
  return new MQLIB(mq_pin1, mq_pin2, mq_pin3);
}
/**
* constructor3
* @param mq_pin1 - analog input channel you are going to use
* @param mq_pin2 - analog input channel you are going to use
* @param mq_pin3 - analog input channel you are going to use
*/
function MQLIB(mq_pin1, mq_pin2, mq_pin3)
{
...
/**
* this gives the current gas value
* @param mq_pin - analog input channel
* @param gas_id - target gas type
* @return ppm of the target gas with the right sensor
*/
MQLIB.prototype.displayGasValue = function(mq_pin, gas_id)
{
    return (this.getGasPercentage(this.read(mq_pin) / this.tabR[mq_pin], gas_id));
}
```

3.2.3 AIRQUA

Suite à des restrictions matérielles liées ESP8266, qui contient un seul pin Analogique, j'ai implémenté un serveur avec un seul capteur (MQ135). le serveur offre les mêmes fonctions que la version Wifi implémentée en langage C.

```
...
/**
* import MQ135.js module
*/
var mq = require("MQ135").connect(pin);
...
/**
 *  display current sensor value to client.
 *  @param sensorName is String argument which set sensor name.
 *  @param is String argument which set sensor gas type.
 *  @param value is String argument which set sensor value.
 */
function sensorValue(sensorName, gasTypes , value)
{
    return ("{\"name\":\"" +sensorName+ "\",\"gasTypes\":\"" +
    gasTypes+ "\",\"value\":\"" +value+ "\"}\n");

}
/**
 *  display sensor informations to client in Json format.
 *  @param sensorName is String argument which set sensor name.
 *  @param model is String argument which set sensor model.
 *  @param reference is String argument which set sensor reference
    .
 *  @param unite is String argument which set sensor unite.
 *  @param gasTypes is String argument which set sensor gas type (
    write it in
 * Json nested array format ex. "[\"LPG\",\"CO\",\"SMOKE\"]" ).
 */
function sensorType(sensorName, model, reference,
    unite, gasTypes)
{
    return ("{\"name\":\"" +sensorName+ "\",\"model\":\"" +model+
    "\",\"reference\":\"" +reference+ "\",\"unite\":\"" +unite+ "
    \",\"gasTypes\":" +gasTypes+ "}\n");
}
...
```

3.2.4 Tableau comparatif

La liste des commandes et les résultats sont exactement les mêmes comparés à la version Arduino développée en langage C.

Arduino C	Espruino JavaScript
Avantages	
Gestion de la mémoire très efficace.	Rapidité d'exécution.
Faible consommation de ressources matériels	Code source court et compacte.
Support de plusieurs capteurs analogiques	Support de plusieurs capteurs numériques.
Support de la majorité des capteurs utilisés	Processeur puissant et efficace
Environnement de développement complet	Support natif de l'API RESTfull.
Inconvénients	
Non support natif de l'API RESTfull	Consommation d'avantage de ressources matériels que Arduino
Problème de latence comparé à la version JavaScript	Ajout obligatoire de Multiplexeur pour le support de plusieurs capteurs analogiques

4 Conclusion

La solution que j'ai apportée correspond aux attentes dans la partie implémentée en langage C.

Je suis parti des connaissances acquises durant mon cursus universitaire. Par la suite, j'ai effectué des recherches qui m'ont permis de mettre en place les différentes parties du projet.

Les objectifs que je me suis fixés au début du PJI n'ont pas tous été réalisés. J'ai bien réussi la première partie, mais pour la deuxième, j'ai eu trop de problèmes liés à la fois au matériel et logiciel. J'ai du changer le microcontrôleur, car Arduino Uno r3 n'était pas assez puissant pour supporter le moteur JavaScript et je l'es remplacé par le ESP8266 qui a ses avantages et ses inconvénients.

Ce PJI a été très intéressant et instructif. J'ai beaucoup appris sur le monde de la recherche, ainsi que sur les connaissances qui seront utiles dans mon parcours. Je suis très attiré par le monde de la recherche et parmi mes objectifs pour le PJI, je voulais me faire faire une idée plus précise de ce milieu. Cette expérience m'a permis de faire des rencontres avec des chercheurs dans le domaine qui m'intéresse. Grâce à ce PJI, j'ai pu travailler les connaissances que j'ai étudiées lors de mes cours du master. C'est grâce à ce projet que j'ai eu une vraie idée sur le travail dans une équipe de recherche, et surtout à devenir autonome sur un projet concret.

Références

[API] APISENSE : *Site Web APISENSE*. https://apisense.io

[Ard] Arduino : *Site Web Arduino*. https://www.arduino.cc

[Ct] China-total : *Site Web Capteurs MQ*. http://www.china-total.com/Product/meter/gas-sensor/Gas-sensor.htm

[ESP] ESP8266 : *Forum ESP8266*. http://www.esp8266.com