

Pager Mobile de Taches

Rapport de projet



Bts snir 2017 - 2018

RAPHAËL Junior CHRISTOPHE Yannis BORDELAIS Axell

Table des matières

Table des matières

[1. Présentation du projet 2](#_Toc515794170)

[2. Répartition des tâches 3](#_Toc515794171)

[3. Planning Mensuel 4](#_Toc515794172)

[4. Gestion Batterie/Aide (Junior Raphaël) 5](#_Toc515794173)

[4.1. Présentation du nodeMCU ESP8266 5](#_Toc515794174)

[4.1.1. Matériel utiliser 6](#_Toc515794175)

[4.1.2. Découverte de l'écran OLED ssd1306 8](#_Toc515794176)

[4.2. Environnement de travail 9](#_Toc515794177)

[4.3. Environnement de développement 9](#_Toc515794178)

[4.4. Test 11](#_Toc515794179)

[4.4.1. Recherche & documentation 12](#_Toc515794180)

[4.4.4. Test 13](#_Toc515794181)

[4.4.5. Test 14](#_Toc515794182)

[4.5. Problème rencontrés / solutions 17](#_Toc515794183)

[4.6. Synthèse 17](#_Toc515794184)

[5. Documents 18](#_Toc515794185)

[5.1. ANALYSE UML 18](#_Toc515794186)

[5.2. NOTICE D'UTILISATION 22](#_Toc515794187)

[5.3. NOTICE D'INSTALLATION 22](#_Toc515794188)

[6. CONCLUSION 24](#_Toc515794189)

[7. Annexes 25](#_Toc515794190)

[7.1. Code source de la gestion batterie/aide 25](#_Toc515794191)

# Présentation du projet

Le système a pour vocation à être utilisé à l'hôtel Salako du Gosier afin de pouvoir joindre des employés "mobiles" présents sur le site (comme par exemple du personnel d'entretien) et de leurs envoyer de cours message texte. Le système se compose d'une base " le coordonnateur ", reliée à un ordinateur, ou une IHM permettra à un Superviseur de composer le texte à envoyer vers un ou plusieurs agents " End device ", en passant éventuellement par un ou plusieurs routeurs. Un " bip sonore " sur le récepteur de poche porté par la personne que l'on cherche à appeler lui indiqueras l'arrivée du message de service. La personne aura ainsi les nouvelles consignes qui lui sont assignées.

Le projet PMT (Pager Mobile de Tâches) à pour objectif d'apporter un nouveau moyen de dynamiser un personnelle d'entretien au sein d'une entreprise par le biais de pager mobile pouvant être mis dans la poche qui fera office de carnet de tâches.

Depuis son poste le superviseur en charge pourras facilement gérer le personnel qui par ce fait enverra des messages qui contiendras les ou la tâche a exécuter par l'agent en quelque seconde l'agent auras pris connaissance de son objectif en appuyant sur son bouton de validation pour lire son message car son buzzer s'étant activer il devras d'abord l'éteindre pour ensuite prendre compte du message reçue et par la suite procédé a la réalisation de cette tâche.

Il possède aussi un moyen de prévenir le superviseur si besoin d'aide si le travaille devient compliqué, il n'auras qu'à appuyer sur son autre bouton qui est pour faire une demande d'assistance auprès du superviseur qui lui a son tour recevra une alerte qui sera son buzzer et par la suite, observera le message d'aide de l'agent ainsi que sa position actuelle et, à son tour le superviseur informera un agent d'aller rejoindre l'agent qui aurait fait une demande d'assistance après du superviseur et répondra cet agent par un message de confirmation par exemple " Un agent est sur le chemin ".

# Répartition des tâches

Pour se faire ce travail a été répartie en 3 parties une pour chaque étudiant de ce groupe qui est :

- Étudiant 1 CHRISTOPHE Yannis :

- Transmettre les messages du superviseur à l'agent

- Confirmer réception de l'agent au superviseur

- Indiquer mise en route au superviseur

- Indiquer arriver nouveau message

- Étudiant 2 BORDELAIS Axell :

- Indiquer couverture ondes radios à l'agent

- Indiquer charge batterie à l'agent

- Capter les ondes radios

- Étudiant 3 RAPHAËL Junior :

- Indiquer charge batterie à superviseur

- Demander assistance au superviseur

- S'adapter à la batterie

# Planning Mensuel

|  |
| --- |
| Mois de Janvier 2018 |
| Junior Raphaël | - Étude du projet  - Recherche des librairies necessaires  - Réalisation de la tâches "Demander Assistance"  - Test effectué  - Câblages de l'écran OLED |
| Axell Bordelais |  |
| Yannis Christophe |  |

|  |
| --- |
| Mois de Février 2018 |
| Junior Raphaël | - Préparation revue  - Préparation ccf  - Recherche des librairies nécessaire au fonctionnement de l'écran  - Affichage de texte sur l'écran OLED  - Utilisation d'un bouton poussoir |
| Axell Bordelais |  |
| Yannis Christophe |  |

|  |
| --- |
| Mois de Mars 2018 |
| Junior Raphaël | - Rapport de projet  - Préparation examens  - Finalisation de la tâche "Demander Assistance"  - Test effectué  - Câblages de l'écran OLED  - |
| Axell Bordelais |  |
| Yannis Christophe |  |

|  |
| --- |
| Mois d’Avril 2018 |
| Junior Raphaël | - Rapport de projet  - Recherche des moyens nécessaires a la récupération de la tension  - Réalisation de la tâches "Informer Superviseur"  - Test effectué  - Préparation revue |
| Axell Bordelais |  |
| Yannis Christophe |  |

|  |
| --- |
| Mois de Mai 2018 |
| Junior Raphaël | - Préparation revue  - Examens  - Assemblage des codes dans un seul fichier  - Test concernant la récupération du pourcentage restant  - Rapport de projet |
| Axell Bordelais |  |
| Yannis Christophe |  |

# Gestion Batterie/Aide (Junior Raphaël)

## Présentation du nodeMCU ESP8266

L’ESP8266 est un circuit intégré à microcontrôleur qui a été développée en 2014 par la société Chinoise Espressif … d’où le nom ESP ! .

Ce circuit intégré produite par une entreprise tierce, AI-Thinker. Celui-ci, de taille réduite, permet de connecter un microcontrôleur à un réseau WiFi et d’établir des connexions TCP/IP avec des commandes Hayes.

Il existe à ce jour 12 versions de ce composant. Chaque version est identifiée par une nomenclature sous la forme : ESP-01, ESP-02 ou ESP-12E mais j'ai opté pour celui que je connaissais déjà l'ESP-12E.

Ce composant embarque un module Wifi, de la mémoire, une liaison série et gère des ports GPIO. Tout cela en quantité différente en fonction de la version.

Pour faciliter son utilisation des cartes ont été créées. Ces cartes permettent de connecter le microcontrôleur à un PC en USB, gèrent l’énergie et fournissent un firmware permettant de programmer le microprocesseur. On parle de carte NodeMCU ou Wemos.

En ce qui concerne NodeMCU, ce terme désigne à la fois le firmware permettant de programmer le microcontrôleur en LUA et aussi la carte complète. La carte NodeMCU est basée le microcontrôleur ESP-12E..

Programmation

L’ESP8266 peut se programmer de plusieurs façons :

- Avec des scripts Lua, interprétés ou compilés, avec le firmware NodeMCU

- En C++, avec l’IDE Arduino

- En JavaScript, avec le firmware Espruino

- En MicroPython, avec le firmware MicroPython

- En C, avec le SDK d’Espressif ou avec le SDK esp-open-sdk3

### Matériel utiliser

Tout d'abord pour cela nous aurons besoin de plusieurs éléments qui sont :

- Un esp8266



- Un afficheur OLED



- Des boutons poussoir



-Une batterie externe



Nous avons utilisé l'IDE (Integrated Development Environement) qui en français (Environnement de Développement Intégrée) qui va nous permettre de réaliser les la programmation de cet esp8266 pour pouvoir réaliser le programme permettant à réaliser ce projet.



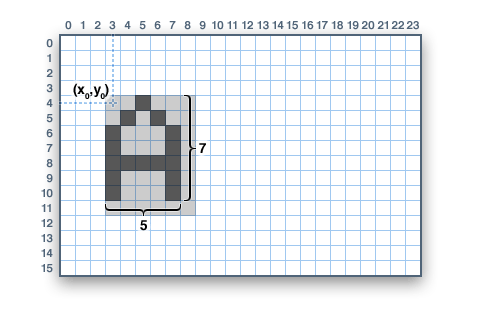
### Découverte de l'écran OLED ssd1306

#### Écran OLED ssd1306

L'écran OLED que nous allons utiliser comme afficheur pour l'agent est un ssd1306 de dimension 128x64 ayant un bus de communication SPI (Serial Peripheral Interface ou Interface Série pour Périphérique

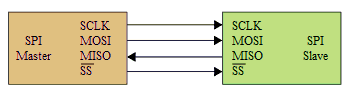
#### Librairie Adrafruit\_GFX / Adafruit\_SSD1306 et SPI



La librairie Adrafruit\_GFX est disponible sur le site du constructeur cette librairie a pour but de gérer la coordination des pixels ainsi que les couleurs et contient toute les fonctions permettant d'afficher les caractères sur l'écran.

La librairie Adrafruit\_SSD1306 est disponible sur le site du constructeur cette librairie a pour but de gérer les écran OLED de petite taille ainsi que ces paramètre au niveau des broches



La librairie SPI permet de communiquer avec les périphérique utilisant le bus SPI avec une carte électronique en tant que "maitre" (Serial Peripheral Interface ou Interface Série pour Périphérique) ce qui lui permet commander les périphériques

Lien : <https://learn.adafruit.com/monochrome-oled-breakouts/arduino-library-and-examples>

## Environnement de travail

Durant ces 5 mois de développement, j’ai eu à ma disposition un nodeMCU ESP8266, un écran OLED, un bouton poussoir.

J’ai décidé de travailler sur un ordinateur de la section pour notre projet. Étant donné que les machines sont pas nôtre et peuvent être formater du jour au lendemain, donc toutes les sauvegardes que j'ai eu a faire ont été faite sur une clef usb.

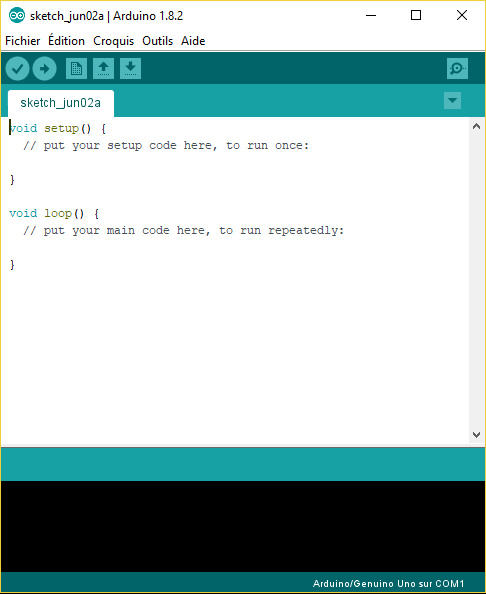
Ce qui était une grosse perte de temps de toujours devoir tout réinstaller.

De plus, les machines des salles utilisées, j’étais dans l’obligation d’installer les drivers du microcontrôleur car les machines ne réconnaissait pas le microcontrôleur brancher en usb.

## Environnement de développement

Le nodeMCU ESP8266 peut être programmer de diverse méthodes mais j'aiopter pour l'IDE Arduino un éditeur qui propose de compiler pour ce microcontrôleur avec du code en C++.

Arduino est un IDE qui permet de réaliser des programmes en C++ dans deux méthodes une setup() et l'autre loop(), le setup() est plus utiliser pour réaliser la partie initialisation des composant et le loop() pour répétez en boucles les actions comme par exemple envoyer une mesure a un serveur.



## Test

|  |  |
| --- | --- |
| Type de test | Test de vérification |
| But | - Envoyer un message /afficher sur l'écran superviseur  - Recevoir une confirmation |
| Organisation | Logiciels :    - Arduino |
| Deroulement | - Configuration du client en TCP/IP  - Création du message a envoyer / afficher  - Récupération de la réponse et affichage sur l'écran |
| Conclusion | Test valide : l'envoie du message ainsi que son affichage sont réussi, la réponse est reçue et ensuite afficher a son tour |

|  |  |
| --- | --- |
| Type de test | Test de vérification |
| But | - Envoyer un message /afficher sur l'écran superviseur  - Passage en mode économie |
| Organisation | Logiciels :    - Arduino |
| Déroulement | - Configuration de la condition de détection  - Création du message a envoyer / afficher  - Passage en mode économie pendant un temps fixée |
| Conclusion | Test valide : la détection de l'état de charge est reconnue l'envoie du message ainsi que son affichage sont réussi, le passage en mode économie est bien enclenchée |

### Recherche & documentation

#### Commande AT

Les commandes AT sont des commandes utilisables pour pouvoir récupérer les informations d'un réseau Wi-Fi auquel il est connecté et qui gère toute la structure réseau de la carte que j'aurais utilisé pour récupérer la position de l'agent via ces commandes AT.

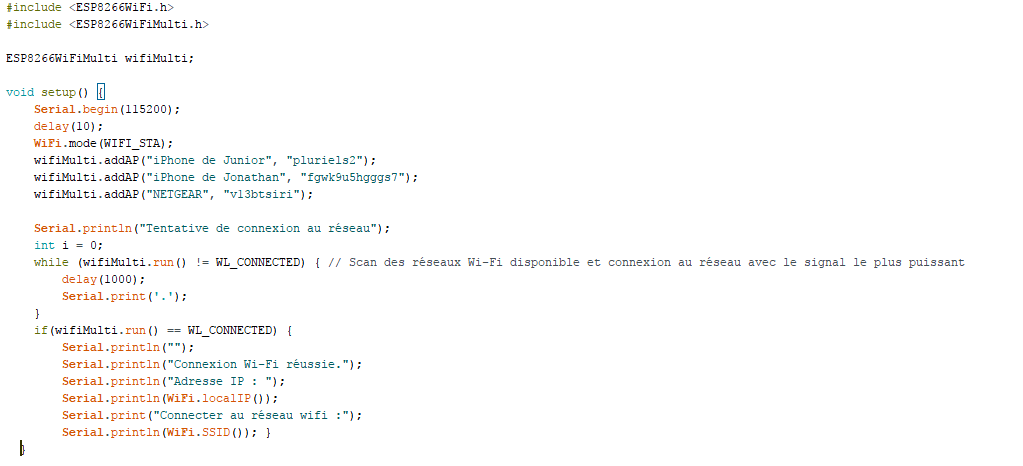
Après avoir recherché longuement sur ces commandes qui m'ont pris un bon bout de temps sans jamais réussir a les utilisées je me suis aperçue quelle étais pas fonctionnel sur mon modèle d'esp8266 mais sur un autre qui est un module Wi-Fi avec une configuration faite pour fonctionner avec ces commandes.

#### Libraire ESP8266WiFiMulti/ESP8266WiFi

Après avoir perdu énormément du temps sur les commande AT j'ai trouvé cette alternative qui est beaucoup plus pratique et permet de regrouper plusieurs réseau Wi-Fi dans une liste et se connecte au réseau avec le signal le plus puissant et offre une plus grande maniabilité de la gestion.

J'ai donc décidé d'utiliser cette librairie pour pouvoir récupérer la position de l'agent qui n'est d'autre que le nom du point d'accès auquel il sera connecté dans l'entreprise.

### Test

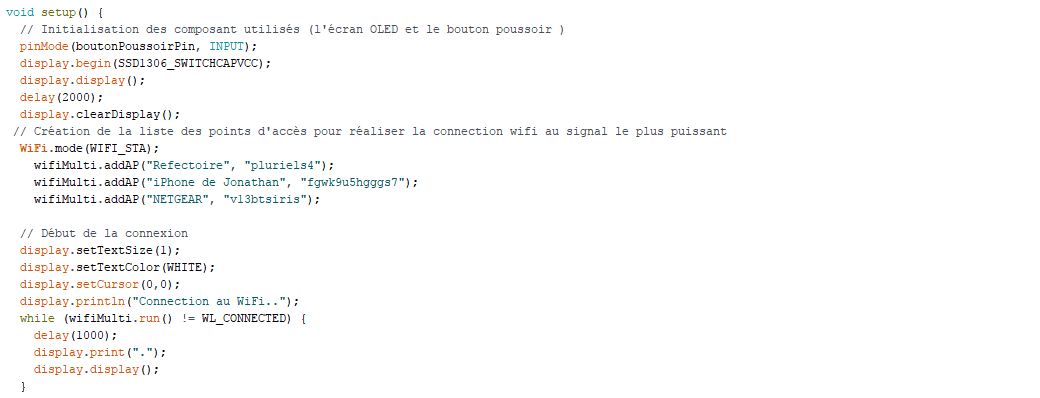


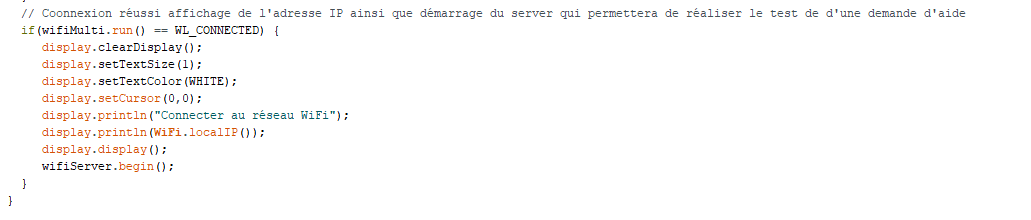
Après avoir rédiger un code permettant de récupérer la position ou plutôt l'SSID du point d'accès auquel il sera relié

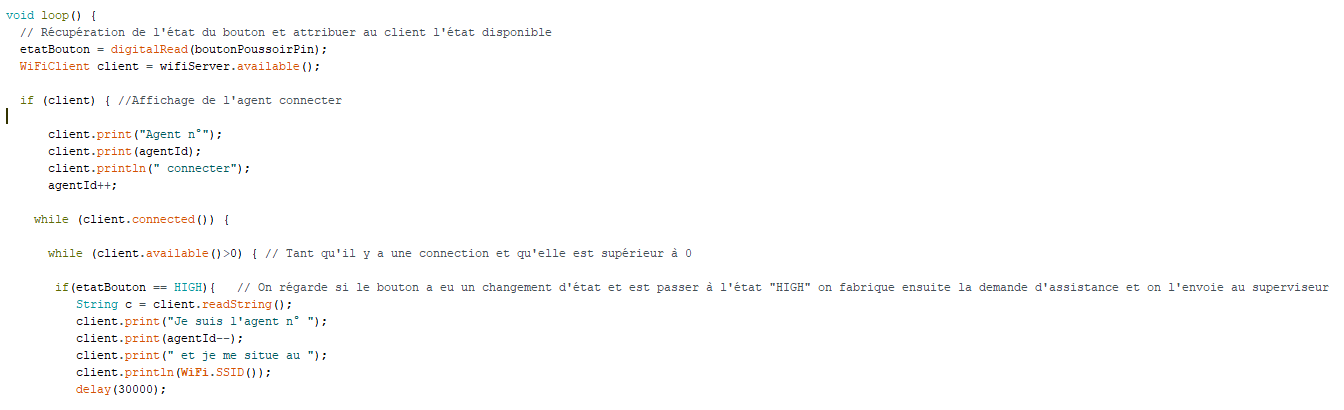
Ce code à bien réussit à nous faire récupérer l'SSID du point d'accès auquel il était connecté

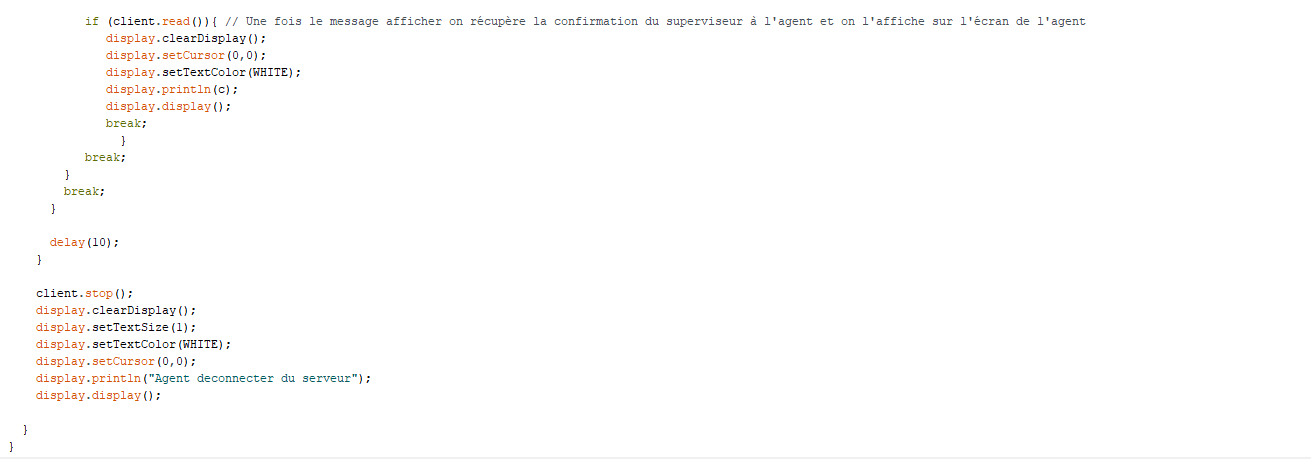
### Test



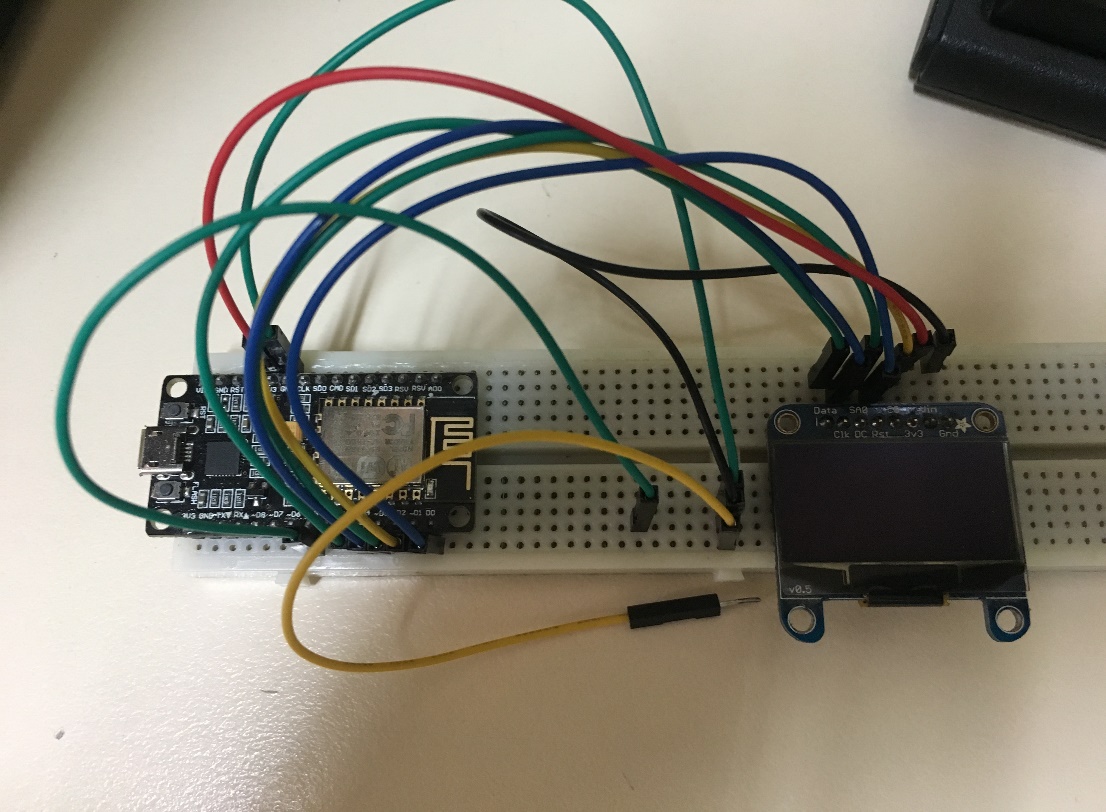








Ceci est le code utilisé pour pourvoir réaliser la demande d'assistance au superviseur par l'agent et par la suite recevoir une confirmation par le superviseur, ainsi que quelque image du montage.



## Problème rencontrés / solutions

La première difficulté que j’ai rencontrée a été de m'être orientée dans la mauvaise voix j'ai perdu beaucoup de temps à mettre en place des commandes AT sur l'esp8266 que j'utilisais c'est après quelque semaine que j'ai compris qu'elle n’était pas compatible voir pas du tout nécessaires avec mon esp8266 qui la version 12E du nodeMCU.

De ce fait j’ai passé beaucoup de temps à chercher une alternative à ces commandes AT. Au bout de quelque heure de recherche, j’ai finalement trouvé la manière dont sera configuré le nodeMCU pour que je puisse le faire agir en tant que "client" et communiquer avec le superviseur.

Le second problème que j’ai rencontré a été de retrouver la bonne valeur qui consistait à lire une valeur et la convertir en un seuil.

Pour résoudre ce problème j’ai utilisé une un autre moyen que d'utilisa la méthode ESP.getvcc() qui elle donne des valeurs j'ai donc opté pour la broche VU du nodeMCU qui elle représente la tension émise par la batterie externe reçue en usb.

## Synthèse

La gestion de batterie et d'aide respecte le cahier des charges.

J’ai ajouté des fonctionnalités comme par exemple le fait de pouvoir modifier l’adresse

IP et le port du serveur, sur lequel l’application souhaite accéder, grâce au fichier.

J’ai aussi ajouté une ligne disant que pour la connexion au poste superviseur soit relancer quand on est déconnecter.

Cela permet de ne pas attendre indéfiniment une réponse du superviseur si celui-ci n’est pas lancé.

Les modifications apportées n’étaient pas comprises dans le cahier des charges.

Ayant terminé assez rapidement la gestion d'aide (environ 3 mois), j’ai commencé à développer la partie information pour pourvoir informer le superviseur une fois après avoir détecté avec le seuil de charge très bas ( 20%).

J’ai utilisé le langage C++ et continuez sur l'IDE Arduino pour la programmation de cette fonction.

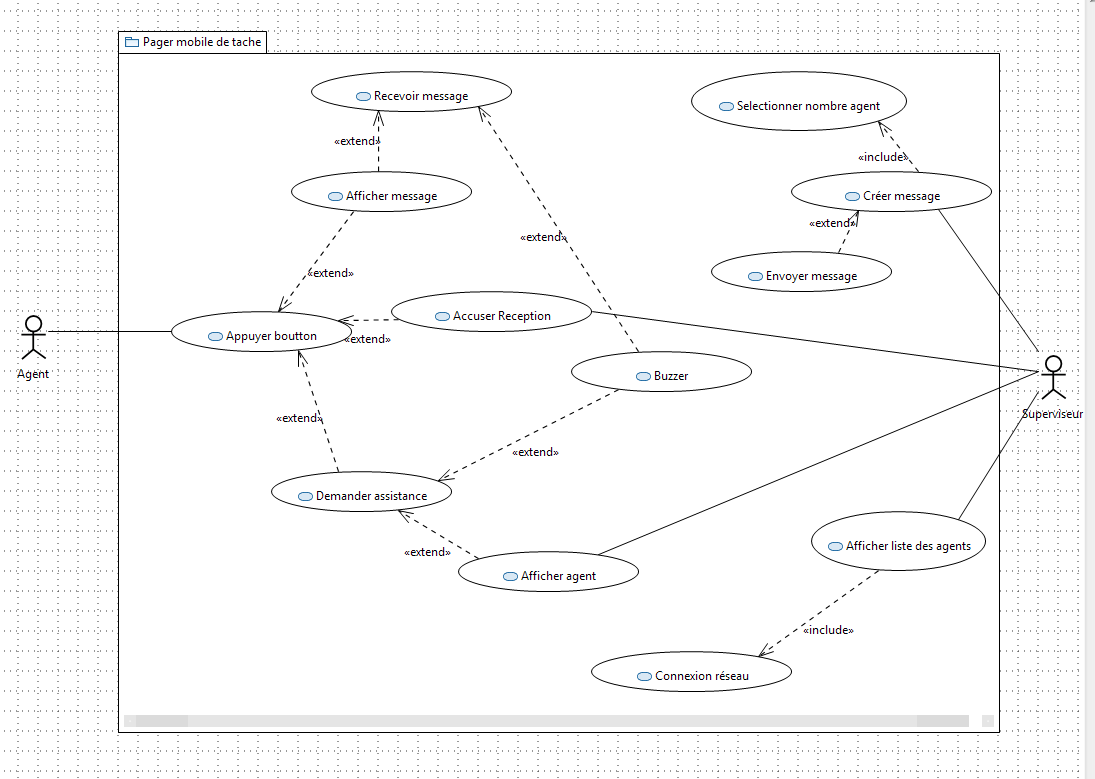
# Documents

Vous trouverez ici tous les documents nécessaire pour pouvoir comprendre le fonctionnement et comment le faire fonctionner

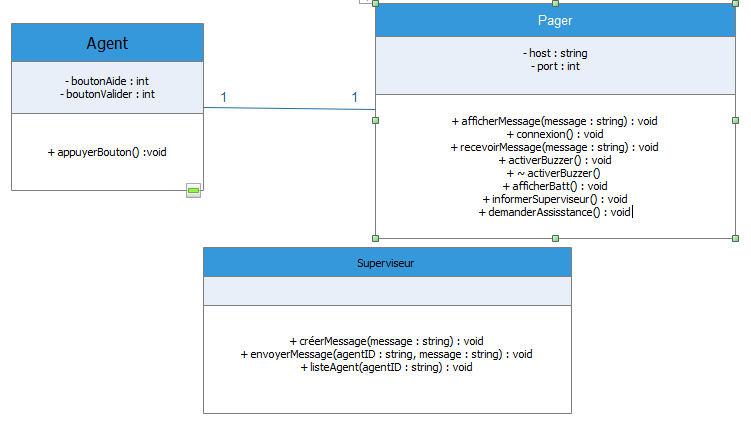
## ANALYSE UML

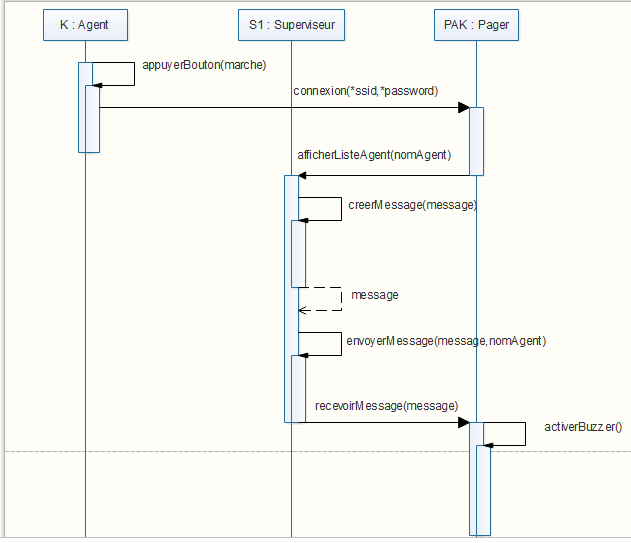
Dans cette partie vous pouvez y voir les 3 diagrammes nécessaires et utiliser mis a disposition pour pouvoir comprendre le fonctionnement du projet et comment il va fonctionner

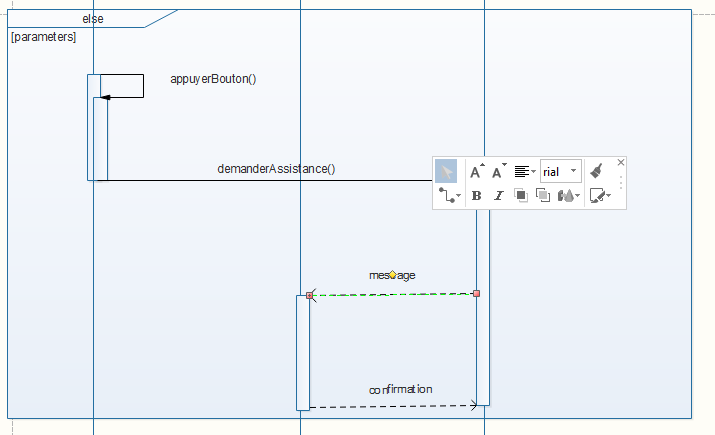
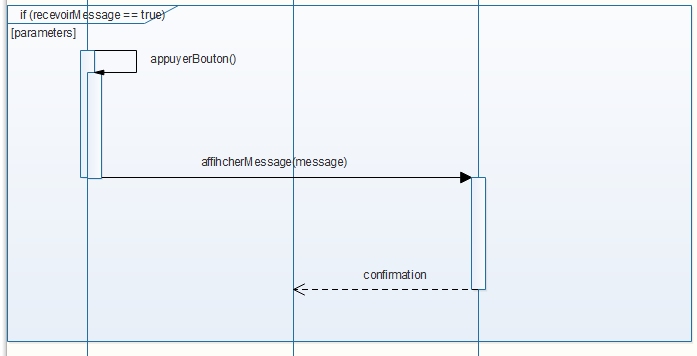
-Diagramme de cas d'utilisation :

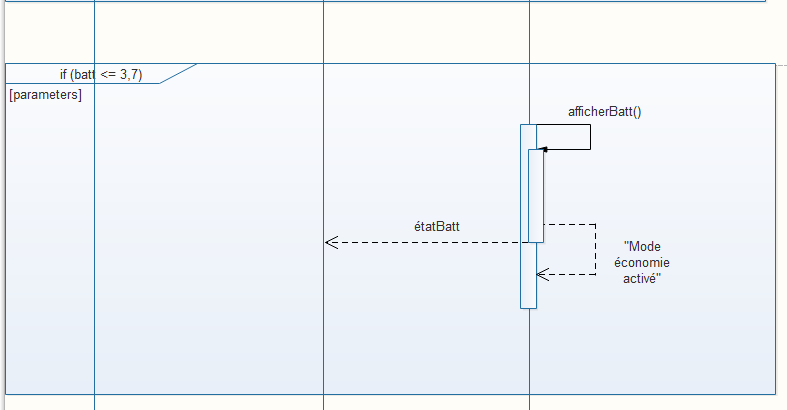


- Diagramme de classe :



- Diagramme de séquence :





## NOTICE D'UTILISATION

* + Gestion Batterie/Aide (Junior Raphaël)

Vérifier que le nodeMCU ne soit pas en mode économie (éteint) et que vous soyez connecter au réseau entreprise.

Appuyer sur bouton poussoir d'assistance pour pouvoir enclenchée une demande d'aide au superviseur.

Une confirmation de lecture est renvoyée si le message a bien été reçu et lue.

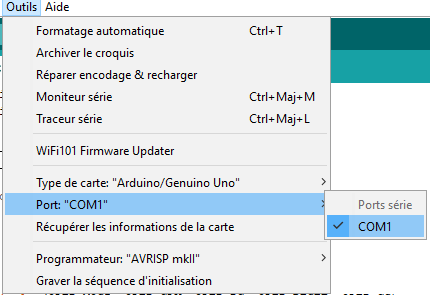
## NOTICE D'INSTALLATION

* + Gestion Batterie/Aide (Junior Raphaël)

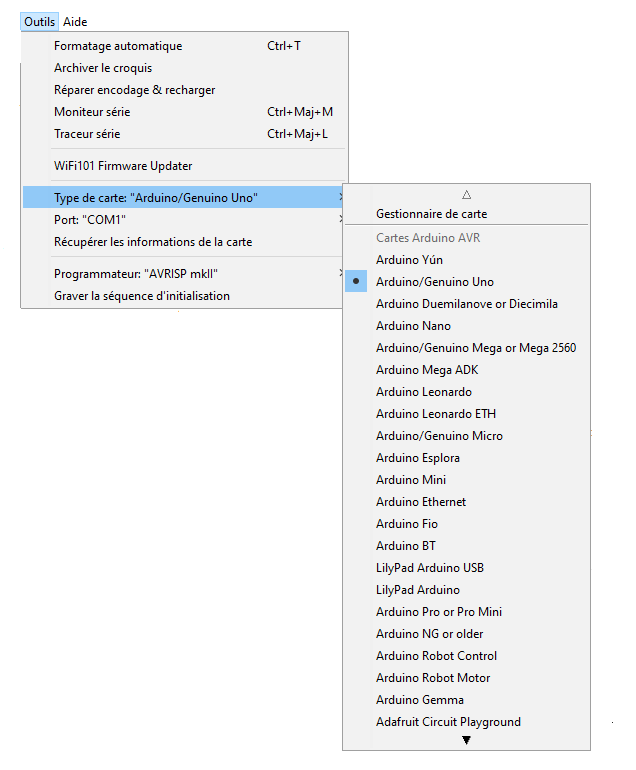
Téléverser le fichiez PMT\_Assemblage\_Code\_Final\_e3.ino sur le nodeMCU ESP8266 qui est reliée au port usb de l'ordinateur par le biais de l'IDE Arduino

- Vérifié que la broche A0 soit bien connecter à la broche VU du nodeMCU

- Vérifié que le port a bien été sélectionner sur l'IDE Arduino



- Vérifié que la bonne carte soit sélectionnée sur l'IDE Arduino



- Vérifié que le câble a bien été branché

- Survoler sur le bouton en forme de flèche sur lequel vous verrez apparaître le mot téléverser puis clicker pour téléverser le code sur le nodeMCU



Une fois le téléversement le programme tournera sur le nodeMCU correctement.

# CONCLUSION

Au terme de ces 5 mois de projet, la plupart des tâches attendues ont été accomplies.

La de mande d'assistance sur le nodeMCU ESP8266 est terminer et fonctionnelle et réalise l’intégralité des fonctions demandées dans le cahier des charges.

Cependant, certaines fonctionnalités ne sont pas encore mise en œuvre à la date de la remise du dossier mais devrait l’être au moment de la présentation.

Ce projet nous a permis tout d’abord d’apprendre à travailler en équipe sur un même projet à partir d’un cahier des charges dont le référentiel a été respecté.

De plus, le développement sur le nodeMCU, m’a permis (Junior Raphaël) de mettre en œuvre mes connaissances et d’apprendre à utiliser une librairie spécifique.

Voici les tâches qu’il nous reste à faire avant l’orale de projet :

* Le mode économie 100% fonctionnel

# Annexes

## Code source de la gestion batterie/aide

#include <ESP8266WiFi.h>

#include <ESP8266WiFiMulti.h>

#include <SPI.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

// Définir les broches de l'écran permettant l'affichage des messages

#define OLED\_MOSI D4

#define OLED\_CLK D3

#define OLED\_DC D2

#define OLED\_CS D1

#define OLED\_RESET D0

Adafruit\_SSD1306 display(OLED\_MOSI, OLED\_CLK, OLED\_DC, OLED\_RESET, OLED\_CS);

void setup() {

display.begin(SSD1306\_SWITCHCAPVCC);

display.display();

delay(2000);

display.clearDisplay();

// Création de la liste des points d'accès pour réaliser la connexion wifi au signal le plus puissant

WiFi.mode(WIFI\_STA);

wifiMulti.addAP("iPhone de Junior", "pluriels2");

wifiMulti.addAP("iPhone de Jonathan", "fgwk9u5hgggs7");

wifiMulti.addAP("NETGEAR", "v13btsiris");

// Début de la connexion

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0,0);

display.println("Connection au WiFi..");

while (wifiMulti.run() != WL\_CONNECTED) {

delay(1000);

display.print(".");

display.display();

}

// Connexion réussi affichage de l'adresse IP ainsi que démarrage du server qui permettra de réaliser le test de d'une demande d'aide

if(wifiMulti.run() == WL\_CONNECTED) {

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0,0);

display.println("Connecter au réseau WiFi");

display.println(WiFi.localIP());

display.display();

}

}

void loop() {

demanderAssistance();

informerSuperviseur();

}

void demanderAssistance(){

String agentId = WiFi.macAddress();

const int boutonAide = 5;

const char \* host = "172.20.10.1";

const uint16\_t port = 1026;

WiFiClient client;

boolean etatBouton;

pinMode(boutonAide, INPUT);

etatBouton = digitalRead(boutonAide);

if (!client.connect(host,port)) // Connexion au serveur si la connexion n'est pas encore établie

client.connect(host,port);

else { //Affichage de l'agent connecté

client.print("Agent n° ");

client.print(agentId);

client.println(" connecté");

}

if(etatBouton == HIGH){ // On regarde si le bouton a eu un changement d'état et est passeé à l'état "HIGH" on fabrique ensuite la demande d'assistance et on l'envoie au superviseur

client.print("Je suis l'agent n° ");

client.print(agentId);

client.print(" et je me situe au ");

client.println(WiFi.SSID());

delay(5000);

String c = client.readString();

display.setCursor(0,0);

display.setTextColor(WHITE);

display.println(c);

display.display();

}

}

void informerSuperviseur() {

unsigned int tension= analogRead(A0);

float volts= (tension/1024.0) \* 4.2;

if (volts <= 3.70){

// Envoie des informations concernants l'état de charges de l'agent au superviseur

client.print("La charge de l'agent n°");

client.print(agentId);

client.println(" est en dessous des 20% ");

display.println("Mode économie activée.");

// Passage en mode économie pendant 20 secondes

ESP.deepSleep(20e6);

}

}