CAZALIS Pauline – VON TOLKACZ Karol – JORET Eddy

Ville connectée

Pour une amélioration de la qualité de vie dans la ville de Pau

Rapport projet tutoré



Remerciements

Nous remercions Fabien LAMAS, Nicolas DUBOIS, Mikaël LEFEVRE, Guillaume FAUVET, Thomas ANSELMI, Lamia AMRANI et Lucas PAUZIES, encadrants de ce projet, pour l’aide qu’ils nous ont apportée.

Table des matières

[Introduction 3](#_Toc71474705)

[I) Présentation du projet 4](#_Toc71474706)

[1-1 – Contexte 4](#_Toc71474707)

[1-2 – Problématique 4](#_Toc71474708)

[1-3 – Description du projet 5](#_Toc71474709)

[II) Analyse du projet 6](#_Toc71474710)

[2-1- Cahier des charges 6](#_Toc71474711)

[2-2 – Analyse des données 7](#_Toc71474712)

[III) Présentation du matériel 9](#_Toc71474713)

[3-1 – Arduino 9](#_Toc71474714)

[3-2 – Capteurs 10](#_Toc71474715)

[3-3 – Logiciels et environnement de travail 11](#_Toc71474716)

[IV) Déroulement du projet 13](#_Toc71474717)

[4-1 – Planning via les diagrammes de Gantt 13](#_Toc71474718)

[4-2 – Répartition des tâches 13](#_Toc71474719)

[V) Présentation du travail réalisé 14](#_Toc71474720)

[5-1 – Gestion des poubelles 14](#_Toc71474721)

[5-2 – Gestion de l’éclairage public 17](#_Toc71474722)

[VI) Difficultés rencontrées 20](#_Toc71474723)

[6-1 – Retard sur le planning initial 20](#_Toc71474724)

[6-2 – Difficultés techniques 20](#_Toc71474725)

[6-3 – Changements apportés au projet 20](#_Toc71474726)

[Conclusion 22](#_Toc71474727)

[Annexes 23](#_Toc71474728)

# Introduction

Lors du Master Technologie de l’Internet à l’Université de Pau et des Pays de l’Adour, nous avons réalisé un projet tutoré dans le cadre de notre module « Gestion et réalisation de projet ».

Notre groupe est composé de :

* Pauline CAZALIS
* Eddy JORET
* Karol VON TOLKACZ

Ce projet a été proposé et encadré par Fabien LAMAS, Nicolas DUBOIS, Mikaël LEFEVRE, Guillaume FAUVET et avec l'aide de Thomas ANSELMI, Lamia AMRANI et Lucas PAUZIES, des ingénieurs logiciels de Capgemini (entreprise de services du numérique français basé à Pau).

En partant d’une solution déjà existante d’agence connectée, le but de celui-ci est d’étendre ce système à la ville de Pau pour améliorer la qualité de ses services ainsi que sa dépense énergétique.

Deux groupes ont travaillé autour de ce projet. Le premier a concentré son travail sur l’urbanisme de la ville et le nôtre, s’est orienté sur l’environnement.

Pour la mise en place de notre solution, nous avons respecté la problématique suivante : « Quels changements pouvons-nous apporter pour améliorer la qualité de vie de Pau ? »

Pour structurer notre écrit, nous allons dans un premier temps faire une présentation globale de notre projet, puis la description du matériel utilisé et enfin de façon détaillée le travail réalisé et les difficultés rencontrées.

# Présentation du projet

## 1-1 – Contexte

Chaque année en France, un habitant produit en moyenne plus de 500kg de déchets.

Ce chiffre ne cesse d’augmenter au fil des années.

À cause de la crise sanitaire, la vente de repas à emporter a considérablement explosé.

Les déchets s’entassent dans les poubelles de la ville, obligeant les passants à trouver de la place pour jeter les leurs. Certains n’hésitent pas à les poser à côté des poubelles ce qui a pour effet de gâcher l’environnement.

À l’heure actuelle aucune solution n’est mise en place pour remédier à ce problème.

Pour l’éclairage public de la ville de Pau, en 2009 la consommation annuelle était de l’ordre de 2,9 mégawatts[[1]](#footnote-1).

En 2017, la ville a décidé de réduire sa consommation électrique en changeant les ampoules actuelles par des LED.

Il est donc primordial de pouvoir gérer ces deux problèmes en temps réel afin de pouvoir améliorer la qualité de vie des habitants tout en réduisant les coûts pour la ville.

## 1-2 – Problématique

Ce projet, réparti entre les deux groupes, avait pour objectif la mise en place d’une « Ville Connectée » ou « Ville intelligente » c’est-à-dire une ville où l’utilisation de capteurs permet la récupération de données. Son objectif est l’amélioration de la qualité des services urbains et la réduction de leurs coûts.

Notre groupe, avait deux tâches principales à réaliser :

* Gestion des déchets : détecter la quantité de détritus dans la poubelle.
* Gestion de l’éclairage public : allumer ou éteindre des lampadaires en fonction de la luminosité extérieure

## 1-3 – Description du projet

Nous avons repris le projet de l’année dernière à savoir, la création d’une agence, équipée d’une solution de surveillance de données, se voulant être les prémices de l’élaboration d’une solution à l’échelle d’une ville.

Nous avons construit la base de notre projet en apportant des améliorations pour répondre à notre problématique.

Pour pouvoir extraire et interpréter les données souhaitées, nous avions à disposition une carte *Arduino* ainsi que différents capteurs et logiciels, servant à récupérer, traiter et stocker ces données.

# Analyse du projet

## 2-1- Cahier des charges

Contexte :

Ce projet est un POC (proof of concept) ayant pour objectif futur de pouvoir répondre à des demandes de collectivitées pour qu’elles deviennent des « Ville Connectée » comme la ville de Pau ou des alentours.

Objectifs :

* Récupérer les données des différents capteurs
* Mettre en place une architecture pour analyser les données récupérées
* Afficher les données analysées sur une page web

Contraintes :

* Contraintes techniques :

Tous les mardis, nous avions une réunion en ligne avec nos encadrants. Celle-ci nous permettait de discuter de l’avancée du projet et des problèmes rencontrés. À l’issu de ces réunions un planning pour la semaine suivante était mis en place.

À cause de la crise sanitaire nous n’avons pas pu travailler ensemble mais à distance, individuellement avec notre propre matériel.

* Contraintes de temps :

Le projet et le rapport sont à rendre pour le 12 mai et la soutenance orale pour le 25 Mai.

## 2-2 – Analyse des données

Dans cette partie nous allons présenter nos deux tâches principales à l’aide de « diagrammes de cas d’utilisation ».

* Gestion des poubelles :

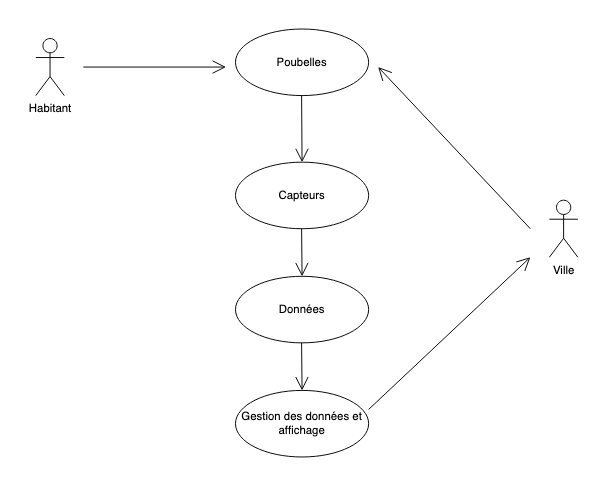


Diagramme de cas d'utilisation des poubelles

Nous avons deux acteurs principaux :

* Les habitants de la ville qui mettent leurs déchets dans les poubelles
* La ville, qui gère les poubelles et consulte les données affichées.

Chaque poubelle possède un capteur de pression attitré, qui communique le poids des déchets contenus à l’intérieur et permettra à la ville d’agir en conséquence.

* Gestion des lampadaires :

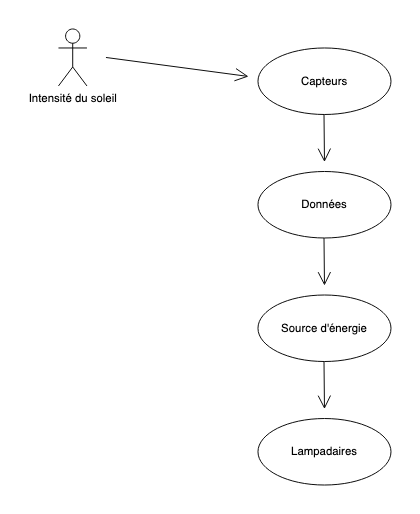


Diagramme de cas d'utilisation de l'éclairage public

Dans ce diagramme les capteurs analysent l’intensité lumineuse ambiante et permettent l’allumage à l’entrée de la nuit ou l’extinction des lampadaires à l’aube.

# Présentation du matériel

Dans le but de répondre au mieux aux besoins techniques de notre projet, l’utilisation de plusieurs outils a été nécessaire. En voici la présentation.

## 3-1 – Arduino

Nous disposons d’une carte *Arduino NodeMCU ESP8266* de la marque *Velleman*, avec son câble d’alimentation USB/Micro-USB.

C’est une carte avec circuit intégré à un microcontrôleur avec connexion Wi-Fi, composé de 10 broches GPIO (General Purpose Input/Output), que l’on utilisera avec le langage C++ avec l’IDE Arduino.

Chaque GPIO peut être PWM (Pulse Width Modulation), I2C (Inter-Integrated Circuit) ou encore 1-Wire (adressage et utilisation de plusieurs capteurs à communication série sur une seule et même broche numérique).

Elle est aussi composée de deux boutons poussoirs rst (reset) et flash (permet de lancer une séquence de flasharge de la mémoire) et d’une antenne PCP qui permet d’améliorer la portée.



Arduino NodeMCU ESP8266 Velleman

Cette carte nous permet de recevoir des informations fournies par un ou plusieurs capteurs et de les transmettre vers un logiciel externe.

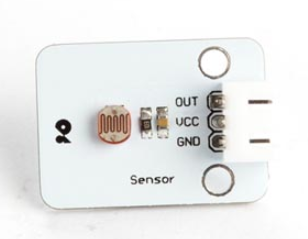
L’*Arduino ESP8266* est équipée d’un module Wi-Fi lui permettant d’envoyer les données reçues sans utiliser une connexion filaire.

Nous disposons aussi de 21 câbles pour la connexion de l’*Arduino* aux capteurs.

Cette carte va nous servir pour la gestion de l’éclairage public.

## 3-2 – Capteurs

3 capteurs nous ont été fournis pour notre projet :

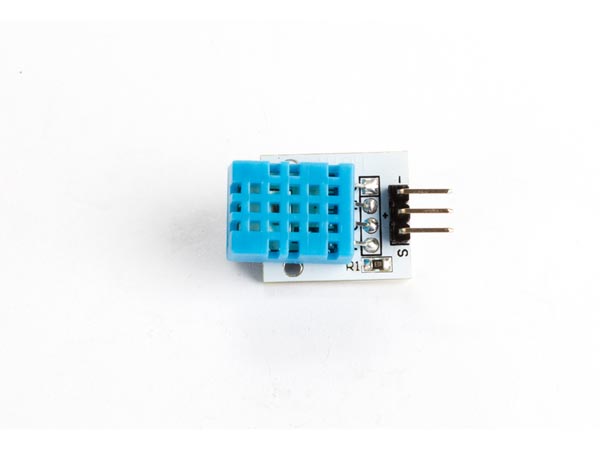
* 1x capteur photosensible qui utilise une photorésistance pour produire une tension de sortie.

Capteur photosensible Velleman

Une photorésistance est une résistance dont la valeur change en fonction de la lumière qu’elle reçoit.

Grâce à ce capteur, la gestion de l’éclairage public pourra être optimisée.

* 2x capteurs de température (NTC) et d’humidité (DHT11).



Capteur de température et d'humidité digital DHT11 Velleman

Ils permettent de récupérer la température et l’humidité présente dans l’air de la ville.

Nous ne les avons pas utilisés pour notre projet mais ils auraient pu servir pour la gestion de l’arrosage public par exemple.

## 3-3 – Logiciels et environnement de travail

Pour exploiter et afficher nos résultats, nous avons utilisé plusieurs logiciels :

Arduino IDE : c’est un logiciel de programmation pour la carte *Arduino*. Il va nous permettre de créer le code pour la faire fonctionner avec les différents capteurs.



Broker MQTT Mosquitto : c’est un protocole de messagerie de type publish-subscribe. Il va permettre à notre *Arduino* d’envoyer et/ou de recevoir des messages en se connectant à un « topic » pour éviter que ceux-ci ne se mélangent.



Node-RED : c’est un outil de programmation, basé sur un navigateur, qui permet de développer des objets connectés à l’aide de flux et de simuler les données d’un capteur fictif.



MongoDB Atlas : c’est une base de données. Elle nous permet de stocker les données reçues par les différents capteurs.

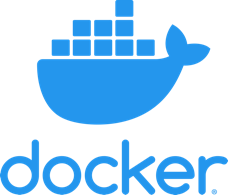


Express JS : C’est un framework (permet de simplifier le travail des développeurs en offrant une architecture « prête à l’emploi ») qui fournit les fonctionnalités d’applications web et mobile fondamentales. Il est basé sur *NodeJs* (environnement d’exécution qui permet d’utiliser JavaScript côté serveur).

Il va nous permettre de recevoir les données des capteurs et les traiter pour pouvoir les envoyer à la fois sur notre page web et sur la base de données.



Angular : C’est un Framework, écrit en JavaScript, qui permet la création d’applications web. C’est grâce à lui que nous allons pouvoir afficher nos données pour qu’elles soient consultables par l’utilisateur.



Docker : C’est une plateforme logicielle qui permet de concevoir, tester et déployer des applications rapidement grâce à des conteneurs logiciels. Ces conteneurs rassemblent tous les éléments nécessaires (bibliothèques, outils systèmes…) afin de pouvoir les exécuter dans n’importe quel environnement.

Ainsi, nous avons pu regrouper notre broker MQTT, *Node-RED* ainsi que *Express* pour qu’ils puissent s’exécuter simultanément.

Pour pouvoir organiser notre projet, nous avons utilisé deux logiciels :



Github : c’est un service web d’hébergement et de gestion de développement de logiciels. Il nous a permis de stocker nos différents fichiers en ligne afin que tous les membres du groupe puissent y avoir accès.



Trello : c’est un outil de gestion de projet en ligne. Nos encadrants ont pu ainsi lister les différentes tâches que nous avions à faire.

Nous avons pu, grâce à cet outil, nous organiser pour sa réalisation.

# Déroulement du projet

## 4-1 – Planning via les diagrammes de Gantt

Pour réussir notre projet tutoré, il nous fallait absolument passer par une planification efficace. Celle-ci a été mise en place via un planning prévisionnel.

Le planning prévisionnel est un outil permettant de visualiser la manière dont ce projet va s’organiser dans le temps (Annexe 1).

En parallèle, nous avons réalisé un planning réel nous permettant de comparer l’avancée de nos travaux par rapport au prévisionnel (Annexe 2).

Pour les modéliser, nous avons utilisé le logiciel *GanttProject*, qui nous permet de créer des diagrammes de Gantt.

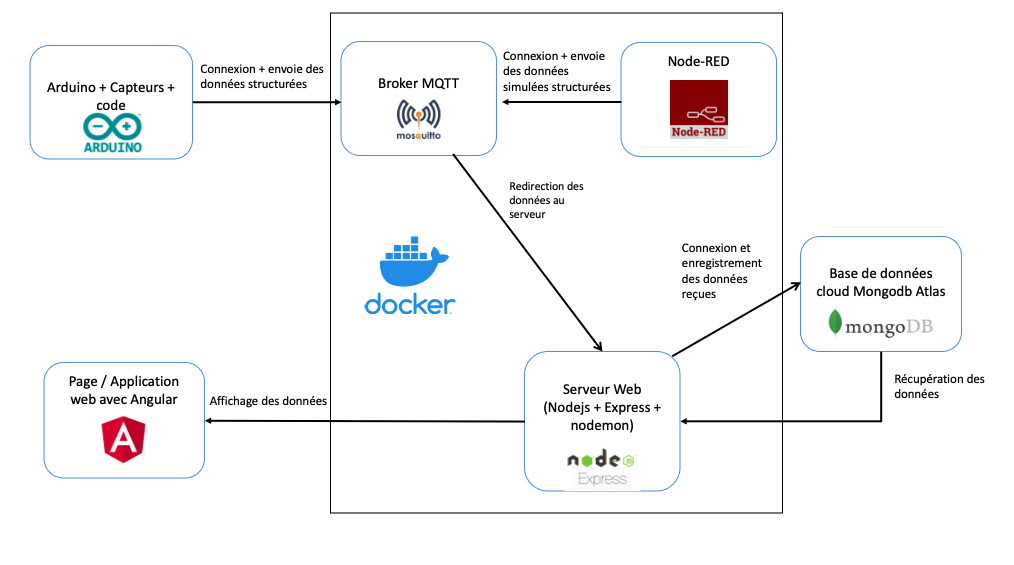
## 4-2 – Répartition des tâches

Après avoir établi ce planning, nous avons réparti ces différentes tâches entre nous trois :

* Réflexion sur le sujet, recherche et proposition de solutions (Pauline CAZALIS, Karol VON TOLKACZ, Eddy JORET)
* Écriture du code pour les différents capteurs (Eddy JORET)
* Mise en place du MQTT sur l’*Arduino* et *Node-RED* (Eddy JORET)
* Simulation des données sur *Node-RED* (Eddy JORET)
* Création du conteneur *Docker* (Pauline CAZALIS, Karol VON TOLKACZ, Eddy JORET)
* Création du serveur *NodeJs* via *Express* (Pauline CAZALIS)
* Récupération des données de MQTT au serveur (Pauline CAZALIS, Eddy JORET)
* Envoie des données du serveur à la base de données *MongoDB* (Pauline CAZALIS, Eddy JORET)
* Récupération des données du serveur vers *Angular* (via socket) (Pauline CAZALIS)
* Création du dashboard sur *Angular* (Pauline CAZALIS, Karol VON TOLKACZ)
* Rédaction du rapport (Pauline CAZALIS, Karol VON TOLKACZ, Eddy JORET)

# Présentation du travail réalisé

Le schéma ci-dessous représente le parcours des données allant du capteur jusqu’à l’enregistrement et leur affichage :



## 5-1 – Gestion des poubelles

Nous devions trouver une solution simple et efficace à cette demande. Nous avons donc décidé de placer un capteur de pression directement à l’intérieur de la poubelle.

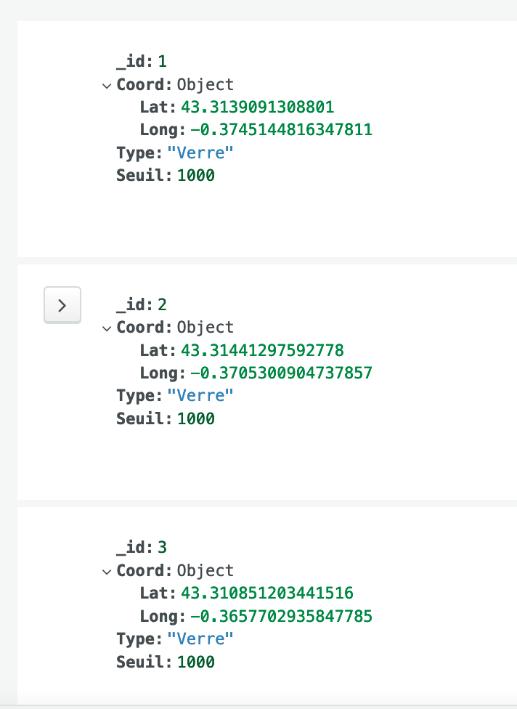
Ne possédant pas le capteur de pression pour l’*Arduino*, nous avons donc dû le simuler avec Node-RED.

🡪 Création de la base de données

La ville possède 3 types de poubelles : verre, déchets communs et recyclables.

Les données qui nous intéressent sont les coordonnées, le type, le seuil, la pression ainsi qu’un moyen d’identification pour chaque poubelle. Pour cela nous avons créé deux types de collections :

* poubelle\_id : contient les données globales des poubelles de Pau. Chaque poubelle possède un id, ses coordonnées, un type et un seuil.



Exemple du contenu de poubelle\_id

* poubelle\_hist : contient les données en temps réel de chaque poubelle et est relié à la première collection grâce à la clé Poubelle\_id.



Exemple du contenu de poubelle\_hist

🡪 Création du code de simulation

Pour cette partie, nous n’avons pas utilisé la carte *Arduino* car nous ne possédons pas le capteur adéquat. Nous l’avons donc simulé grâce au logiciel *Node-RED*.

Pour simuler sur *Node-RED*, nous avons créé un flux spécifique aux données liées à la gestion des poubelles. Dans ce flux, nous avons utilisé des variables nous permettant d’y accéder depuis n’importe quelle fonction.

Nous utilisons des nodes « inject » qui envoient une information au bout d’un temps donné et répètent cet envoi régulièrement. Avant la première utilisation, nous devons configurer les données initiales des poubelles via deux fonctions :

* La première concerne les données de localisation (Coord), de type (Type) et de seuil (Seuil) pour un identificateur (\_id) de poubelle.
* La seconde concerne la pression initiale avec son heure de création en y associant l’id de la poubelle.

Ces données créées seront envoyées ensuite au topic correspondant : poubelle\_id ou poubelle\_hist.

Une fois cela fait, nous utilisons d’autres nodes « inject » pour créer une boucle permettant de mettre à jour la pression de chaque poubelle, et envoyer cette nouvelle valeur sur le topic voulu sous le format JSON défini lors de la création de la pression initiale.

🡪 Création du serveur

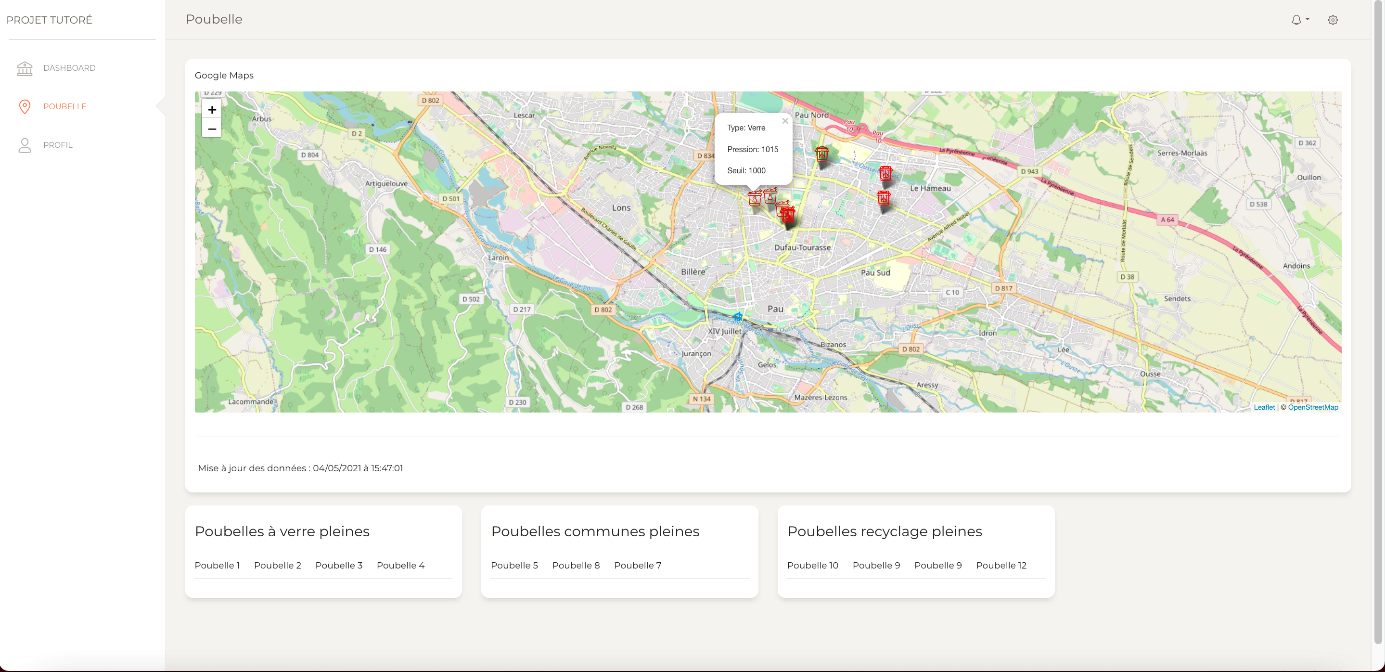
Notre serveur nous permet de récupérer nos données dans un message via le broker MQTT reçu sous format texte. Il ne nous reste plus qu’à le convertir au format JSON pour le sauvegarder dans la base de données.

Le serveur pourra aussi récupérer les informations de la base de données et l’enverra sur notre page web *Angular* via une socket.

Une socket est un protocole réseau permettant la création d’un canal de communication à double sens entre un serveur et un navigateur.

🡪 Création de la page web

Après avoir reçu les données depuis la socket, *Angular* va stocker ces données au bout d’un temps défini pour pouvoir les afficher. Nous avons décidé d’afficher le type, le seuil et la pression de chaque poubelle de façon séquentielle sur une carte (créée par la bibliothèque *Leaflet*). L’opérateur pourra donc surveiller en temps réel la situation des poubelles connectées.



Notre "dashboard" indiquant l'état en temps réel des poubelles de la ville de Pau

## 5-2 – Gestion de l’éclairage public

Notre deuxième tâche : chercher une solution pour réduire la consommation électrique et le coût de l’éclairage public de la ville de Pau

Nous avons opté pour la solution la plus simple : le capteur photosensible va détecter s’il fait jour ou nuit de manière à activer ou non le lampadaire.

🡪 Création de la base de données

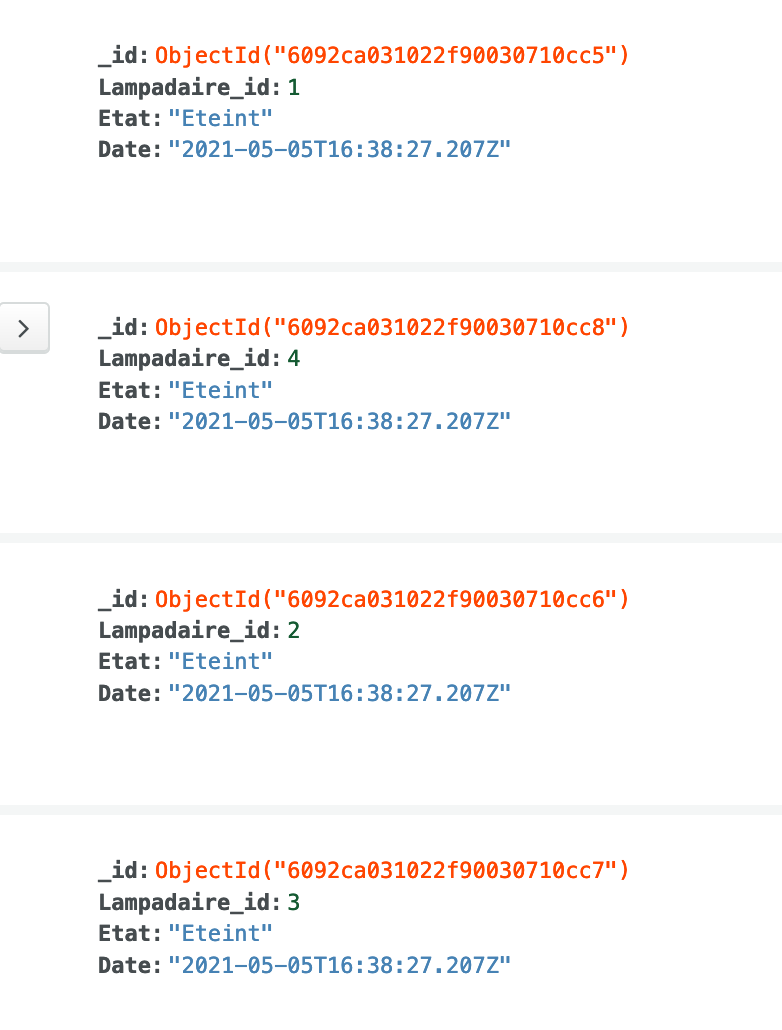
Notre base de données comporte deux collections :

* lampadaire\_id : contient l’identificateur et les coordonnées du lampadaire.



Exemple du contenu de lampadaire\_id

* lampadaire\_hist : contient les données lors du passage jour/nuit avec l’heure de changement. La base est reliée à la première collection grâce à la clé Lampadaire\_id.



Exemple du contenu de lampadaire\_hist

🡪 Création du code

Nous avons utilisé deux moyens différents :

Le premier avec un véritable capteur et un *Arduino,* et le second avec une simulation sur *Node-RED* via différentes fonctions codées en JavaScript permettant la simulation de plusieurs lampadaires.

Pour pouvoir utiliser le capteur et l’*Arduino*, nous avons utilisé *Arduino IDE*.

Nous avons dû la connecter à une connexion internet via Wi-Fi avec le module correspondant, pour pouvoir accéder au broker MQTT et s’y connecter.

Ainsi, une fois connecté à celui-ci, il ne nous reste plus qu’à attendre l’envoi des données par le capteur. La transmission du message, avec l’état voulu, sur le topic du broker ne se fera uniquement qu’au moment du changement jour/nuit.

Sur Node-RED, lors de la création des données initiales, nous n’envoyons que l’ID du lampadaire ainsi que ses coordonnées sur le topic lampadaire\_id. Pour les données historiques nous envoyons l’état initial (éteint) ainsi que l’ID du lampadaire avec son heure de création sur le topic lampadaire\_hist.

Ensuite, nous simulons via des variables, les changements d’état jour/nuit et envoyons le message correspondant sur le topic voulu.

Une variable de vérification permet de savoir si un message a déjà été envoyé sur le topic concernant le dernier changement effectué. Dans ce cas, il n'y a pas d'envoi jusqu'au prochain changement.

# Difficultés rencontrées

## 6-1 – Retard sur le planning initial

Le plus gros problème que nous avons rencontré avec le planning initial est celui de la gestion des poubelles.

Il y a eu aussi un problème de communication entre les logiciels.

À cause des différents soucis techniques et de changement, deux semaines de plus nous ont été nécessaires pour clore le projet.

## 6-2 – Difficultés techniques

La première difficulté que nous avons rencontrée a été la connexion MQTT sur l’*Arduino*. Sans passer par *Docker*, nous n’avions pas accès au MQTT par une adresse IP mais uniquement par le nom « localhost ». Malheureusement avec l’*Arduino* nous n’avons pas pu nous connecter à ce protocole via ce nom.

Le second problème a été de revoir la base de données qui était initialement composée d’une seule collection. De ce fait la base de données n’était pas assez générique.

Le dernier problème a été de transmettre les données simulées par *Node-RED* au Framework *Express* de *NodeJs* sans passer par *MongoDB Atlas*. Pour cela nous avons essayé avec des requêtes HTTP et WebSockets. Malheureusement nous n’avons pas pu réussir à faire fonctionner les requêtes HTTP et nous avons abandonné les WebSockets pour incompatibilité (la socket du serveur est de type « socket.io » et non de type WebSockets).

## 6-3 – Changements apportés au projet

Le problème de la connexion MQTT sur *Arduino* a été résolu une fois que nous avons défini un conteneur MQTT accessible par le nom défini dans le fichier dockerfile.

Pour pallier la base de données qui n’était pas assez générique, nous avons utilisé deux collections et par conséquent nous avons dû changer les codes de simulations pour que les données correspondent à celles voulus par la base de données.

Pour réussir à envoyer les données de *Node-RED* au framework *Express* de *NodeJs*, nous avons décidé d’utiliser le protocole MQTT, utilisé aussi par les *Arduino* pour les transmettre.

Nous avons décidé de créer un conteneur *Docker* global composé de plusieurs conteneurs : *Node-RED*, MQTT et *NodeJs* avec le module nodemon (pour relancer le serveur automatiquement lors d’une modification). Le fait de les mettre dans un seul et même conteneur permet de les lier et de les faire reconnaître entre eux pour établir des connexions de communications.

# Conclusion

Ce projet nous a permis de découvrir plusieurs solutions possibles d’utilisation de l’IOT pour améliorer ou créer des techniques en rapport avec l’écologie.

Pour mettre en œuvre ce projet, nous avons dû passer par des phases de réflexions, conceptions et créations suivant différents points de vue :

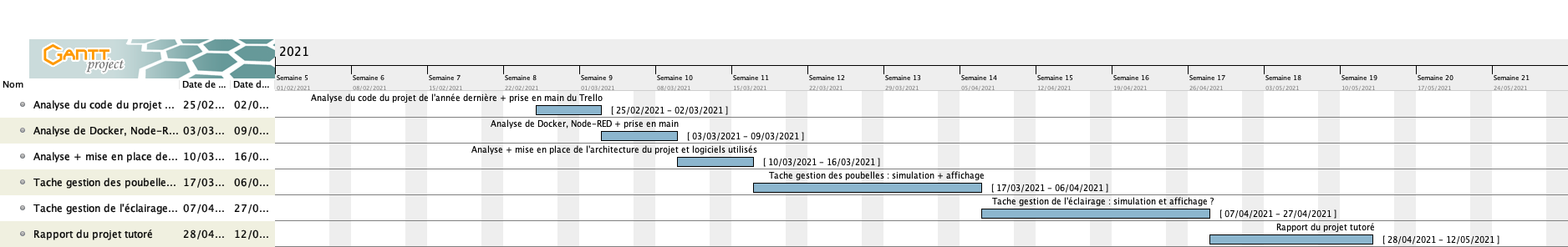
* Travail d’équipe : une bonne cohésion du groupe a permis la répartition des tâches.
* Communication : une bonne communication dans le groupe est nécessaire pour faire avancer le projet dans les meilleures conditions.
* Délai : cela nous a permis de voir qu’il est difficile de respecter un planning compte tenu des diverses problématiques rencontrées.
* Nouvelles technologies : pour faire fonctionner notre application, nous avons exploré et utilisé de nouvelles technologies comme *NodeJs* et *Angular*.

Afin d’optimiser l’application nous avons eu quelques idées supplémentaires :

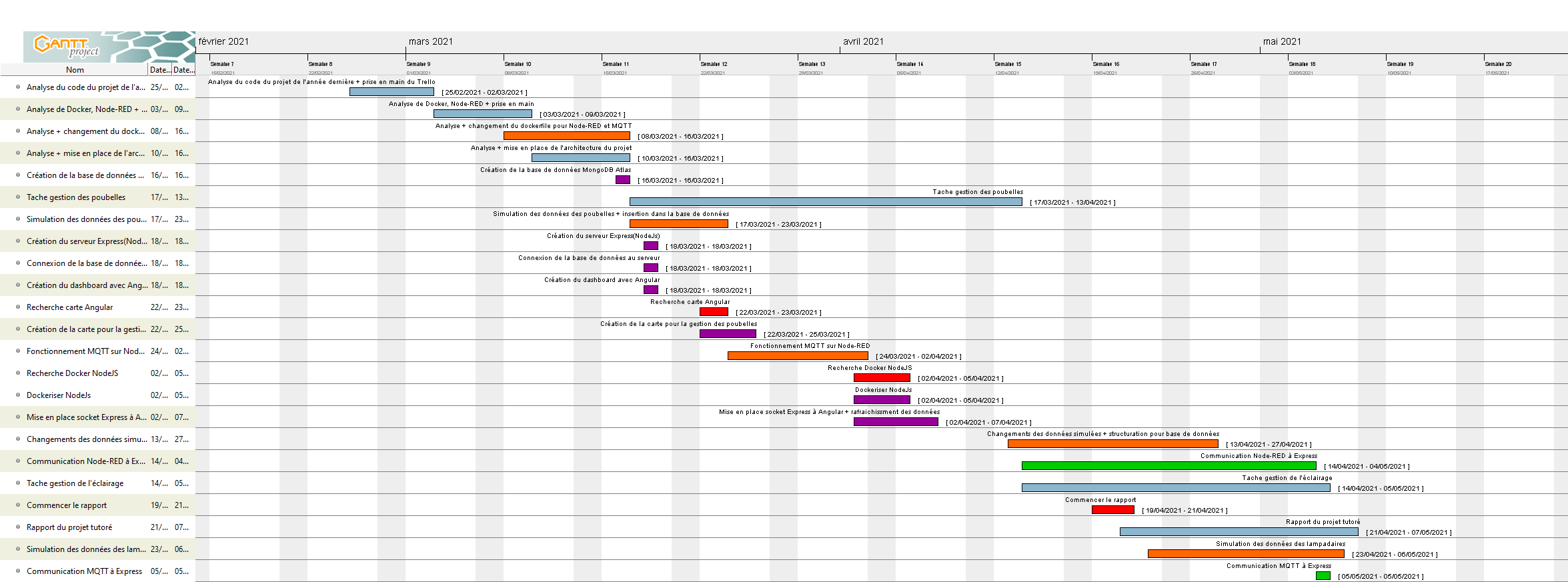
* Gestion des poubelles : faire une liste des poubelles triées par type. Un bouton permettra d’afficher sur la carte, l’état de chaque type de poubelles.
* Gestion de l’éclairage : réduire encore la consommation d’électricité en optimisant l’allumage par la présence de capteurs de mouvement. Cela permettra aux lampadaires situés en aval d’être avertis d’une présence (effet domino : passage de l’éclairage en mode normal en mode réduit et vice-versa).

# Annexes

Annexe 1 : Planning prévisionnel



Annexe 2 : Planning réel



Légende :

Couleurs des taches :

* Bleu : groupe
* Vert : Eddy JORET et Pauline CAZALIS
* Orange : Eddy JORET
* Violet : Pauline CAZALIS
* Rouge : Karol von Tolkacz

1. Source : <https://www.larepubliquedespyrenees.fr/2010/11/13/un-meilleur-eclairage-pour-une-consommation-moindre,165788.php> [↑](#footnote-ref-1)