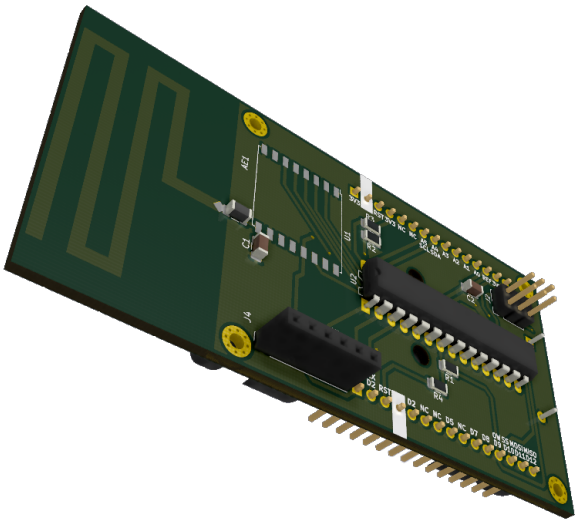
Rapport de stage

DUT GEII

Capteurs connectés pour l'écologie



Titouan Soulard

Semestre 4
avril à juillet 2020

Mes remerciements à mon tuteur de stage, M. Rebaudo
Un grand merci également à Mme. Fraysse, référente de stage à l'IUT
Mes remerciements aussi à Mme. Egloffe ainsi qu'aux enseignants & personnels de l'IUT
Pour finir, je remercie les membres du laboratoire EGCE pour leur accueil (même très bref)

0.1 Résumé

1. Les insectes ravageurs de cultures sont responsables de nombreuses pertes de récolte. La lutte contre ces insectes joue un rôle central sur la sécurité alimentaire des populations, il est donc important de mettre en place des modèles permettant de prédire la phénologie de ces insectes nuisibles afin d'intervenir au bon moment.
2. Certaines informations sont disponibles via les stations météorologiques, mais elles ne sont pas suffisantes : un maillage plus fin est nécessaire pour l'étude des conditions locales. En particulier, le couvert végétal influe beaucoup les valeurs de température, d'humidité relative, et sur la luminosité perçue par les insectes.
3. Le projet présenté vise à développer des unités locales, basés sur un microcontrôleur, permettant de mesurer divers paramètres des conditions de vie des insectes. Ces unités locales communiquent avec un serveur central pour l'analyse ultérieure des données.
4. Les composants nécessaires pour la réalisation du projet représentent moins de 20 €, un ordre de grandeur bien inférieur aux unités commerciales, ne proposant pas par ailleurs la transmission des données à distance. Cette baisse de coût, en plus d'une flexibilité accrue, permet la mesure des conditions associées à un microenvironnement, laissant un maillage beaucoup plus fin pour l'étude des insectes.
5. Au vu de la grande disponibilité de ces composants aujourd'hui, et vu la simplicité d'assemblage de la carte, ne nécessitant aucune qualification particulière, ces capteurs de terrains verront une augmentation importante dans les prochaines années.
6. En parallèle, un serveur permettant la centralisation des données est proposé, afin de permettre une étude à plus grande échelle et sur différents continents des insectes nuisibles.

0.2 Mots-clefs

microcontrôleur, Arduino, ATMega, serveur, LoRa, Raspberry Pi, agriculture, microclimat, microenvironnement

Table des matières

0.1 Résumé 2

0.2 Mots-clefs 2

1 Contexte du stage **6**

1.1 Le stage de DUT GEII : une année exceptionnelle 6

1.1.1 Objectifs &c. 6

1.1.2 Les entités de la recherche 7

1.1.3 Les personnels de la recherche 8

1.2 Objectifs du stage 10

1.2.1 Présentation du projet PI2P 10

1.2.2 Vue d'ensemble du projet de recherche 11

1.2.3 Répartition des tâches 12

1.3 Établissement d'un cahier des charges & organisation 14

1.3.1 Cahier des charges pour les deux systèmes 14

1.3.2 Protocoles de transport & modèles de menace 14

1.3.3 Organisation du stage – Diagramme de Gantt 17

2 Données locales : un transfert fiable et efficace **18**

2.1 Réponse à la problématique : utilisation d'une carte de terrain 18

2.1.1 Simplicité d'utilisation : une carte Arduino ? 19

2.1.2 Sur la communication sans-fil : un système LoRa . . . 20

2.1.3 Visualisation globale du code embarqué 21

2.2 Aspects de sécurité et de fiabilité 21

2.2.1 Chiffrement pour l'embarqué : état de l'art 21

2.2.2 Étude critique du système mis en place 24
2.2.3 Conservation locale des données de terrain 25
2.3 Prototype de passerelle et vulgarisation au public 26
2.3.1 Réalisation d'un prototype de passerelle 26
2.3.2 Documentation sur la réalisation de l'unité locale . . . 27

3 Côté serveur : réception, stockage et restitution de l'information 29

3.1 Choix techniques préliminaires 29
3.1.1 Les API REST 29
3.1.2 Choix du langage 30
3.1.3 Gestion de la base de données 31
3.2 Arborescence du projet et modèles de données 31
3.2.1 Côté serveur : une architecture MVC 32
3.2.2 Gestion du Javascript client 34
3.2.3 Base de données : ORM, modèles et migrations 34
3.3 Gestion des accès et sécurisation des données 35
3.3.1 Les clefs API 35
3.3.2 Séparation en groupes 36

Table des figures

1 Organigramme global de la recherche 8
2 Organigramme interne du laboratoire 9
3 Représentation des tâches à effectuer 13
4 Représentation en couches des protocoles utilisés 16
5 Algorigramme de principe du code de l'unité locale 22
6 Contenu d'un paquet et processus de chiffrement 24
7 Schéma relationnel pour la base de données 32
8 Représentation du modèle MVC 33

Liste des tableaux

1 Tableau de contraintes pour les unités locales 15
2 Tableau de contraintes pour le protocole local 17
3 Diagramme de Gantt du stage 18

1 Contexte du stage

> Une intelligence incapable d'envisager le contexte et le complexe planétaire, rend aveugle, inconscient et irresponsable.
>
> *Edgar Morin*

1.1 Le stage de DUT GEII : une année exceptionnelle

Les bousculements de l'époque ont quelque peu modifié l'organisation normale du DUT : banalisation de l'unité d'enseignement associée, annulation des stages de nombreux étudiants, etc. Malgré tout, le projet dont il est question dans ce rapport a pu être mené de manière convenable ; cette partie décrit l'organisation ayant rendu cela possible.

1.1.1 Objectifs &c.

Le stage est une étape importante du DUT GEII, tant d'un point de vue pédagogique que professionnel et remplit à ce titre un triple objectif. D'abord, ce stage à permis la découverte d'un nouveau milieu : celui de la recherche publique. La section suivante (1.1.2) de ce rapport lui est dédiée, et vise à démêler les différentes entités cohabitant pour rendre possible un projet d'ampleur.

Un second rôle important du stage est de préparer à la poursuite d'études ; un stage dans un laboratoire de recherche est à ce titre une bonne occasion de se faire une idée du fonctionnement de la recherche. Cette année, le contexte particulier à quelque peu empêché la communication avec tous les différents acteurs, mais donne une idée des possibilités de poursuite de carrière dans ce domaine.

En plus des objectifs mentionnés, le stage doit contribuer à l'apprentissage de nouvelles notions, et la mise en œuvre des apprentissages dans un projet

concret ; les parties 2 et 3, soit une majorité de ce rapport — à vocation technique — sont consacrées à cette mise en œuvre.

1.1.2 Les entités de la recherche

La recherche est un domaine pour le moins complexe, avec de multiples acteurs. Globalement, l'entité névralgique est l'Unité Mixte de Recherche (UMR). Le stage dont il est question dans ce rapport s'est déroulé au sein de l'UMR Évolution, Génomes, Comportement, Écologie (EGCE).

Une UMR est issue de l'association de deux types d'organismes : les Établissements Publics à caractère Scientifique et Technologique (EPST) ou les Établissements Publics à caractère Industriel et Commercial (EPIC) et les établissements d'enseignement supérieur.

Ici, l'UMR EGCE dépend de l'Université Paris-Saclay (enseignement supérieur), qui est une université française créée en 2019 visant à regrouper trois grandes universités : Paris-Sud, Versailles-Saint-Quentin et Évry-Val-d'Essonne, ainsi que des établissements non-universitaires annexes.

Côté EPST, l'UMR (ou laboratoire) dépend de l'Institut de Recherche de le Développement (IRD) — visant à apporter une aide au développement des pays du Sud, en particulier sur les domaines des sciences humaines et sociales et de la santé — ainsi que du Centre National de la Recherche Scientifique (CNRS), institution publique regroupant une grande partie des laboratoires de recherche français.

En interne, le laboratoire est divisé en plusieurs pôles ; ils permettent de spécialiser le travail et de mener en parallèle des projets de recherche sur des thématiques différentes. Ce stage est organisé au sein du pôle *Évolution et écologie*, ayant pour objet de comprendre l'impact des changements globaux sur l'écologie des insectes, dans les contextes particuliers de la sécurité alimentaire et de la santé publique.

Par ailleurs, le laboratoire EGCE mène (conformément aux missions de l'IRD) des partenariats internationaux, notamment avec deux instituts étrangers qui seront mentionnés à de multiples reprises dans ce rapport : ICIPE au Kenya et PROINPA en Bolivie. La figure 1 récapitule les différentes informations de cette partie, de manière schématique.

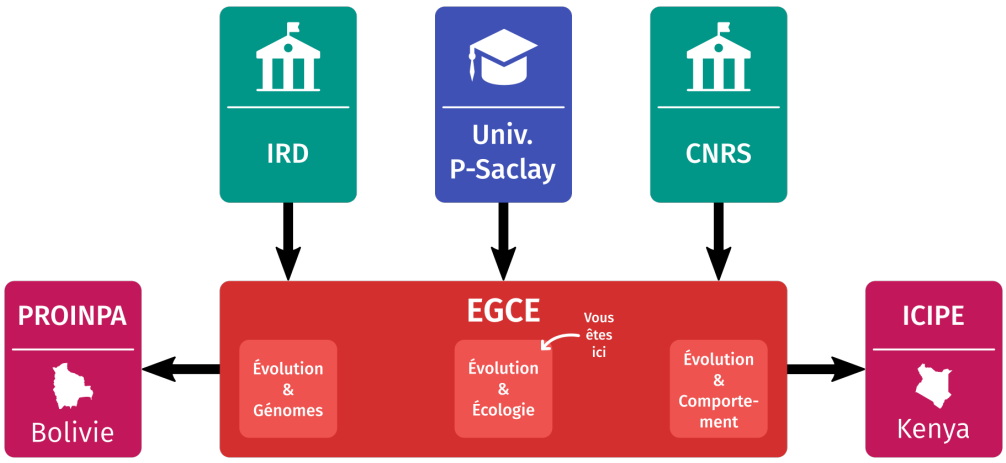


FIGURE 1 – Organigramme global de la recherche

1.1.3 Les personnels de la recherche

Chaque pôle est animé par un ou une responsable de pôle (RP). Ici, le pôle évolution et écologie du laboratoire est dirigé par *Myriam Harry* et *Paul-André Calatayud*.

Sous leur direction, on retrouve les personnels du laboratoire ; parmi eux, nous nous focaliserons sur les chercheur·euse·s titulaires, qui se divisent globalement en deux catégories : chercheur·euse·s (ingénieur·e de recherche, chargé·e de recherche, directeur·ice de recherche), enseignant-chercheur et enseignante-chercheuse (maître de conférence, professeur des universités, etc), rattaché·e·s à une des universités de tutelle (Paris-Saclay ici) et donnant des cours en parallèle de leur activité de recherche.

Enfin, ces personnels de recherche peuvent encadrer des personnels non-titulaires (stagiaires, doctorants, CDD), c'est ici que je me trouve, ainsi qu'un autre stagiaire de master, Aghiles Ben Sider et un doctorant en écologie, Baptiste Regnier. La figure 2 résume la situation à l'intérieur du pôle (en incluant seulement **une petite partie** des personnels pour illustrer le propos).

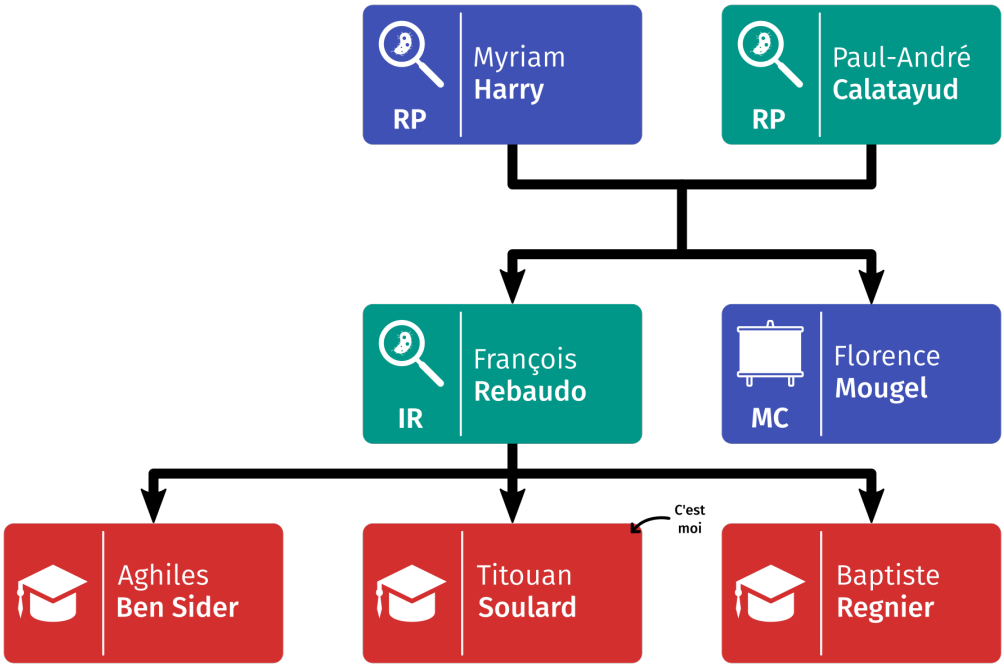


FIGURE 2 – Organigramme interne du laboratoire

1.2 Objectifs du stage

Après présentation des acteurs du stage, il est important d'en bien comprendre les objectifs, par une présentation d'abord globale de l'intérêt des recherches menées, et leur mise en perspective par rapport à l'état actuel de la recherche dans le domaine, puis par une présentation plus approfondie du sujet particulier traité pendant ces trois mois de stage.

1.2.1 Présentation du projet PI2P

Les modèles phénologiques prennent une grande importance ces dernières années ; en effet, dans le monde, les insectes nuisibles sont responsables, dans certains cas, de plus de 80 % de pertes de récoltes [1]. Par ailleurs, la connaissance des conditions de vie des insectes pourrait permettre l'utilisation plus efficace et raisonnée de produits phytosanitaires, et ainsi que d'éviter les pertes financières liées à la perte de récoltes, supérieure à 70 milliards de dollars [2].

La mise en œuvre de ces nombreux modèles nécessite des données, desquelles dépend largement leur précision ; ces données sont usuellement prises en laboratoire, à chaque stade de la vie de l'insecte, en ajustant différents paramètres (en particulier la température). Il serait plus aisé de prendre des mesures directes dans le milieu de vie des insectes afin d'obtenir des modèles plus fiables [3]. Par ailleurs, la mise en place de nouveaux types de capteurs (humidité du sol, humidité de l'air, luminosité, etc), permettrait d'affiner les modèles existants, majoritairement basés sur la température [4].

C'est un des objectifs du projet PI2P, et l'aspect majoritairement abordé dans ce rapport : la mise en place de capteurs de terrain. À la suite de ce stage, le projet devrait s'orienter vers une analyse des données collectés, afin d'affiner

1. Oerke E-C (2006) Crop losses to pests. J Agric Sci 144 :31–43. doi :10.1017/S0021859605005708

2. Bradshaw CJA, Leroy B, Bellard C, et al (2016) Massive yet grossly underestimated global costs of invasive insects. Nat Commun 7 :12986. doi :10.1038/ncomms12986

3. Campbell A, Frazer B, Gilbert N, et al (1974) Temperature requirements of some aphids and their parasites. J AnimEcol 11 :431–438. doi :10.2307/2402197

4. Rebaudo F, Rabhi V-B (2018) Modeling temperature-dependent development rate and phenology ininsects : review of major developments, challenges, and future directions. Entomol Exp Appl. doi :10.1111/eea.12693

les modèles phénologiques existants, puis, à terme, d'établir une évaluation de l'impact des différents scénarios possibles du changement climatique pour les pays partenaires du projet (Kenya et Bolivie), sur le plan de l'agriculture et de l'économie.

1.2.2 Vue d'ensemble du projet de recherche

Le projet PI2P est un projet long, s'étendant sur plus de 3 ans ; le stage étant de seulement quelques mois, il ne traitera que quelques aspects du projet. Le découpage préliminaire fait état de deux groupes de tâches distincts : une approche basée sur la littérature scientifique, et concernant particulièrement les aspects écologiques et biologiques du projet ; cette approche, loin des attendus du DUT GEII, n'est pas traitée dans ce rapport.

La seconde approche est basée sur des études de terrain ; elle combine la biologie et les sciences de l'ingénieur, en particulier le génie électrique — pour la réalisation des capteurs de terrain — et le génie informatique — pour la mise en place du serveur, et de la station de transfert sur site.

Ces deux approches sont menées en parallèle, et ce stage se focalise sur la seconde ; son objectif est simple : récolter puis analyser des données au plus proche des conditions de vie des insectes.

Cette approche est également découpée en plusieurs sous-projet, dans un ordre chronologique :

1. d'abord, construire un système permettant d'accéder à distance à des données locales, et de stocker ces données pour analyse ultérieure ;
2. ensuite, analyser rétrospectivement les données obtenues, et les corréler avec les données de piégeages phéromonaux — capteurs de « la quantité » d'insectes présents sur le site — mis en place sur le terrain, afin de dégager des tendances quant aux conditions climatiques favorables au développement des insectes ;
3. en parallèle avec le premier aspect du projet, il faudra tenter de construire des modèles permettant de quantifier les tendances observées, puis confronter ces modèles aux données collectées pour les ajuster ;
4. le dernier objectif consiste à ouvrir le projet, en tentant d'appliquer les modèles développés aux différents scénarios du réchauffement climatique

proposés par le GIEC, pour étudier la prolifération des insectes dans les prochaines décennies.

Le premier point est l'objet de ce stage, ainsi que du stage d'Aghiles Ben Sider, qui mènera d'autres tâches en parallèle.

1.2.3 Répartition des tâches

Avant de continuer sur la répartition des tâches, voyons le vocabulaire utilisé pour le projet ; les éléments couverts par ce vocabulaire sont basés sur le prototype de François Rebaudo, qui a constitué une base de travail pour le stage. L'ensemble du projet est basé sur quatre éléments physiques : d'abord, des unités locales, qui centralisent plusieurs capteurs sur une zone géographique étroite. Ils doivent être sans-fils, et fonctionner de manière autonome, sans intervention extérieure. Il faut ensuite une passerelle ou *gateway*, recevant les informations d'un ou plusieurs capteurs de terrain, et transmettant cette information via un lien Internet.

Pour la partie informatique, il faut un serveur, s'occupant de la réception et la centralisation des données, et enfin un ordinateur en bout de chaîne pour la visualisation des données. Pour l'organisation, une liste générale des tâches a été effectuée ; cette liste est à vocation indicative, mais permet d'avoir une vision des choses à effectuer ; la partie électronique-informatique du projet est découpée en sept tâches comme suit :

1. détermination des variables (Baptiste & François) : à partir modèles existants, il s'agit de définir les variables à mesurer, et d'établir une première liste de capteurs répondant au besoin ;
2. comparatif des capteurs (Aghiles) : depuis la liste préliminaire de capteurs, il s'agit de les comparer et de définir les plus intéressants, ainsi que d'écrire un morceau de code Arduino permettant leur exploitation ;
3. réalisation de l'unité locale (Titouan) : réalisation d'une unité locale flexible permettant de connecter différent types de capteurs ; cette unité doit être autonome et envoyer les données vers la passerelle ;
4. transfert de l'information (Titouan & François) : (a) gestion de la communication unité locale-passerelle ; (b) une fois l'information arrivée au niveau de la passerelle, il faut gérer l'envoi de l'information vers le serveur ;

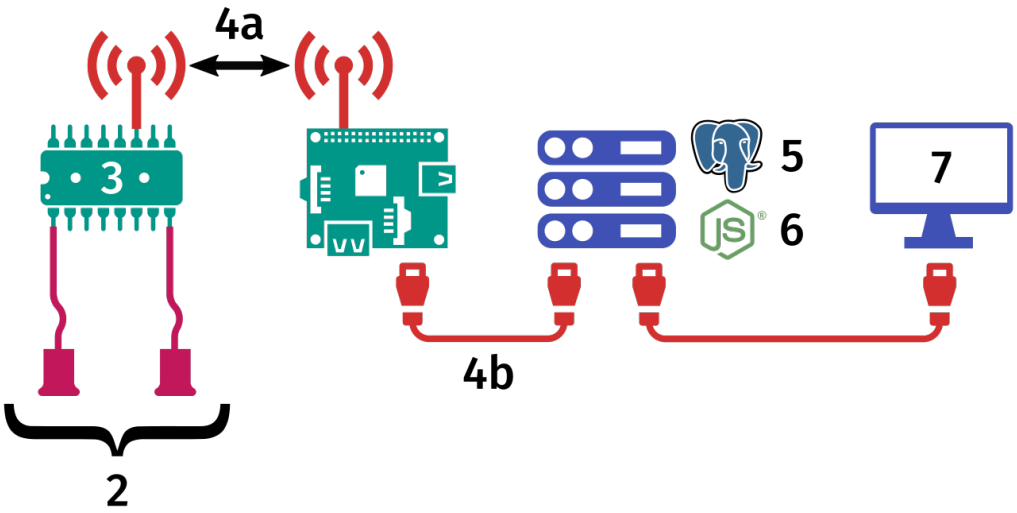


FIGURE 3 – Représentation des tâches à effectuer

5. stockage des données (Titouan) : gestion de la réception des données côté serveur, et du stockage des informations, en vue d'une restitution ultérieure ;
6. restitution des données (Titouan) : mise en place, côté serveur, d'une procédure pour la récupération sous différentes formes de l'information stockée ;
7. visualisation des données (François) : réalisation de graphiques interactifs à partir des données de différents sites, et de différents capteurs ; également, génération de rapports sur le bon fonctionnement des capteurs.

Ce découpage des tâches est utilisé à de nombreuses reprises dans le projet, il est donc très important ; la figure 3 schématise les différentes tâches, pour une meilleure compréhension. Ce rapport sera par la suite découpé en deux grandes parties : un côté électronique, avec la réalisation et la programmation de l'unité locale (tâche 3, partie 2), et un prototype du transfert passerelle-serveur (tâche 4) et un côté informatique, orienté sur la réalisation du serveur pour la réception, le stockage et la restitution des données (tâches 5 & 6, partie 3).

1.3 Établissement d'un cahier des charges & organisation

Comme mentionné ci-dessus, ce rapport de stage se focalise sur la réalisation de deux systèmes distincts ; pour chacun de ces deux systèmes, il est important, à partir des informations disponibles, et des contraintes évidentes, d'établir un cahier des charges. Par la suite, nous étudierons les contraintes des systèmes de communication, très importants pour le projet ; enfin, la réalisation des différents éléments sera organisée dans le temps via un diagramme de Gantt sur la durée du stage.

1.3.1 Cahier des charges pour les deux systèmes

Les contraintes pour l'unité locale sont données dans le tableau 1 ; elles ont été établies à partir de réflexions sur les lieux de déploiement des capteurs, ainsi que des contraintes explicites du projet. Pour le serveur, les contraintes sont moins strictes ; on se contentera d'une liste globale des besoins :

— globalement, on souhaite une très bonne sécurité, car le serveur sera accessible depuis l'extérieur ;
— par ailleurs, on voudra récupérer les données au format JSON, pour pouvoir les travailler ensuite en Javascript ;
— le serveur doit être en mesure de répartir la charge lorsqu'arrivent plusieurs connexions simultanées ;
— en ce qui concerne le contrôle d'accès, il est nécessaire d'avoir deux niveaux : un filtrage grossier, pour les permissions évidentes (ex. seul un administrateur peut supprimer une passerelle) et un filtrage fin pour les accès passerelle (ex. seule la passerelle n° 1 peut ajouter des données aux capteurs de la passerelle n° 1).

1.3.2 Protocoles de transport & modèles de menace

Trois vecteurs de communication sont *a priori* associés au système : entre l'unité locale et la passerelle, entre la passerelle et le serveur et entre le serveur et le client. Pour chacun d'entre eux, il faut définir des contraintes, ainsi qu'un modèle de menace, qui permettra de déterminer le niveau de sécurité

Contrainte	Critères	Niveaux
Utilisateur	– facile à concevoir – facile à utiliser	– peut être conçu par un néophyte en électronique – changement facile des piles – raccordement des capteurs aisé – facile à reprogrammer
Autonomie	Tenue sur piles	Au moins un mois
Robustesse	– température – altitude (pression) – précipitations	– de -30 à +50 °C – de 0 à 5000 mètres – quelques projections d'eau
Communication	∅	Communique de manière fiable avec la passerelle.
Compatibilité	Protocoles disponibles pour les capteurs	– I^2C – SPI – OneWire – quelques GPIOs
Fiabilité	∅	Pas de débranchements intempestifs.

TABLE 1 – Tableau de contraintes pour les unités locales

nécessaire.

Le premier choix technique a été d'unifier les deux derniers protocoles : il n'y a pas de raison apparente de séparer la communication client-serveur de la communication passerelle-serveur, en particulier car dans les deux cas, le besoin de sécurité est le même (très important), et la fiabilité demandée est plus ou moins similaire ; en choisissant un protocole unique, on simplifie également la démarche de conception.

Contrairement à la partie précédente, où les choix techniques pouvaient être faits *a posteriori*, les outils mis en œuvre pour la réalisation de la partie serveur peuvent être fondamentalement différents selon le protocole choisi ; le choix est donc effectué en amont du projet, et basé sur les critères suivants :

— (globalement) on souhaite une consultation facile des informations côté client, et ne pas réinventer la roue. L'outil le plus standard pour la

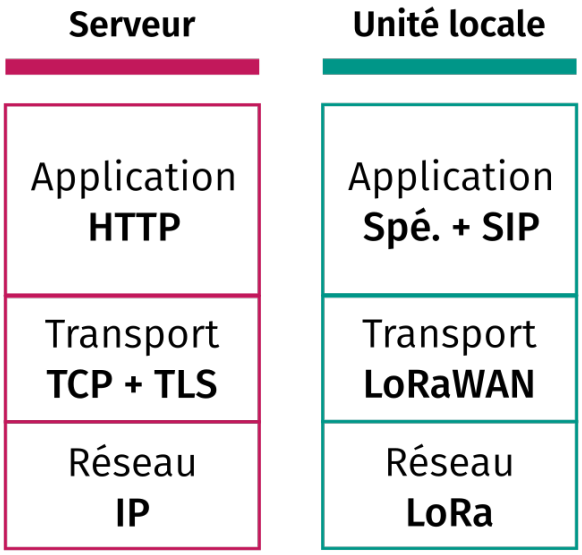


FIGURE 4 – Représentation en couches des protocoles utilisés

transmission de ce type d'information aujourd'hui est Internet, on veut donc un protocole Internet ;

— (couches réseau & transport) on choisira de se baser sur une pile TCP+IP ; la raison du choix est très simple : aucune limite pour la taille des données, ou rapidité de la transmission et besoin de fiabilité important ;

— (sécurisation des données) contrairement au transfert depuis l'unité locale, la sécurisation des données est ici un enjeu majeur : aucun accès ne doit pouvoir être détourné, on se basera donc sur du chiffrement fort et éprouvé, avec *Transport Security Layer* (TLS) ;

— (couche application) TLS est aujourd'hui implémenté presque partout, mais son usage majeur est dans le texte enrichi via **HTTPS** ; ce protocole convient parfaitement au besoin ici : côté client, présentation de pages dynamiques et côté serveur, réception de données avec accusé de réception.

Ces contraintes donnent une pile réseau comme indiquée figure 4, avec la pile de l'unité locale, détaillée section 2.1.2. Du côté de l'unité locale, le besoin en sécurité est moins fort, mais la puissance de calcul déployée doit être la plus faible possible — pour limiter la consommation électrique ainsi que pour

Contrainte	Critère	Explications
Communicativité	(1) à distance (2) international	Deux contraintes basiques mais très importantes : pas de lien filaire possible, et déploiement dans plusieurs pays.
Portée	Moyenne distance : quelques kilomètres	La distance entre la passerelle et l'unité peut être grande.
Légèreté	Fonctionnel sur un Arduino Uno	Certains protocoles modernes sont trop lourds pour un petit microcontrôleur, il faudra donc s'adapter.
Sécurité	Relative, pour prévenir l'intrusion facile	La communication entre les unités locales et la passerelle sont globalement de faible portée, inutile donc d'introduire des niveaux de sécurité aberrants, juste de quoi décourager les bidouilleurs locaux.

TABLE 2 – Tableau de contraintes pour le protocole local

des raisons de coût. Le choix s'effectuera par la suite, selon les critères établis dans le tableau 2.

1.3.3 Organisation du stage – Diagramme de Gantt

Un diagramme de Gantt du stage est proposé dans le tableau 3. On y voit les quatres tâches traitées en priorité — bien que quelques réalisations relèvent des autres, ainsi qu'un « Divers » regroupant la réalisation du cahier des charges la première semaine, et la restitution la dernière (rédaction de la documentation, présentation du fonctionnement de la carte en présentiel, etc).

On distingue bien deux parties pour le stage, mises en évidence par une séparation dans le tableau. La première moitié du stage s'est orientée vers la réalisation du serveur, tandis que la seconde moitié était consacrée à l'unité locale.

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
T3								●	●	●	●			
T4a												●	●	
T5		●	●			●								
T6				●	●									
T7							●							
Div.	●													●

TABLE 3 – Diagramme de Gantt du stage

2 Données locales : un transfert fiable et efficace

> Il est aisé de prouver qu'aucune opération locale sur un des membres d'une paire intriquée peut influer sur l'autre membre.
>
> *Juan Maldacena*
> *Leonard Susskind*

L'unité locale a pour objectifs de centraliser plusieurs capteurs de terrain (température, humidité, etc), ainsi que de communiquer avec la passerelle. Comme mentionné précédemment, le système mis en place doit être le plus simple possible d'utilisation, et, dans la mesure du possible, de conception ; nous étudierons donc d'abord la meilleure manière de répondre aux contraintes techniques, avant d'aborder les aspects sécuritaires du protocole de communication, et enfin de mentionner le prototype de passerelle mis en place pour les tests du projet.

2.1 Réponse à la problématique : utilisation d'une carte de terrain

Deux des principales difficultés concernant les contraintes pour la carte de terrain sont la simplicité d'utilisation et la transmission de données sans-fil. L'étude de ces deux facteurs pourra nous permettre d'établir un algorigramme rapide du fonctionnement du système, pour une visualisation simple de son fonctionnement.

2.1.1 Simplicité d'utilisation : une carte Arduino ?

Avant de réfléchir à la conception d'un système spécifique, il est intéressant de voir les solutions techniques existantes, et qui permettraient de répondre à la problématique. Dans le cadre du projet, deux systèmes semblent chacun intéressants pour un besoin spécifique : d'abord, les cartes Arduino présentent des facilités très intéressantes pour les néophytes, qui semblent répondre au besoin de simplicité de montage et d'utilisation du projet.

D'un autre côté, ces cartes ne disposent pas d'un système de communication sans-fil, et leur autonomie est limitée, à cause de l'utilisation d'un Quartz externe, ainsi que de l'interface USB — alimentée en permanence ; sauf à s'orienter vers la gamme professionnelle, très chère, ces cartes ne correspondent donc pas entièrement au besoin ; de même, l'ajout d'un système sans-fil devient plutôt coûteux.

Par ailleurs, la facilité de montage apparente peut se traduire par un manque de fiabilité des connexions à long-terme, ne répondant pas parfaitement aux besoins du projet. Les autres systèmes sur lesquels s'appuyer sont les enregistreurs de données industriels (*dataloggers*), présentant une fiabilité très importante, mais souffrant d'un coût très élevé, en particulier pour les modèles sans-fil.

On souhaite donc obtenir un système ayant une fiabilité proche des enregistreurs industriels, ou au moins avec des contacts résistants, et une simplicité proche de celle d'Arduino pour la programmation et l'interfaçage, en plus d'un module de communication sans-fil, le tout en conservant le coût le plus bas possible.

Le choix fait ici est la réalisation d'une carte électronique basée sur un microcontrôleur AtMega328 — le même que sur l'Arduino Uno, entre autres — et possédant un module LoRa intégré RFM95 pour la communication sans-fil (voir 2.1.2 pour justification de ce choix).

Cette carte est rendue programmable dans l'IDE Arduino, en utilisant les mêmes bibliothèques, ce qui permet de conserver une certaine simplicité. Une perte évidente est la facilité de conception, puisque la carte doit maintenant être réalisée manuellement ; l'idée est que cette complexification soit faite au prix d'une bien meilleure robustesse des connexions — au module LoRa notamment — ainsi que d'un coût bien inférieur, puisque le coût de la main

d'œuvre est internalisé (assemblage, tests, etc) ; l'ensemble revenant à environ 20 €.

2.1.2 Sur la communication sans-fil : un système LoRa

Sur le protocole sans-fil, et hormis les contraintes de sécurité, qui seront traitées dans la section suivante, la contrainte la plus importante est la portée de l'unité locale. Dans certaines zones, la passerelle peut en être assez éloignée, jusqu'à quelques kilomètres. Les systèmes de communication à basse-énergie classiques tels que de le Bluetooth, ZigBee, ou même le WiFi, ne disposent pas d'une portée suffisante.

Pour obtenir des portées de ce type, en conservant une très faible consommation d'énergie, il est possible de se tourner vers les solutions de type *Low-Power Wide-Area Network* (LPWAN) ; en cherchant, on trouve particulièrement les trois grands noms du domaine : Sigfox, LoRa et NB-IoT.

En procédant par élimination, il est possible de réduire la liste à un seul candidat ; en effet, NB-IoT demande l'achat d'une bande de spectre radiofréquence, et ne dispose que de modules à prix déraisonnables [5] ; de même, Sigfox mise sur la connectivité à son réseau propriétaire pour le fonctionnement du matériel, et ne permet donc pas les réseaux privés (souhaités ici), ne reste donc que LoRa, qui semble cocher toutes les cases.

Comme indiqué ci-dessus, la carte embarquée intègre un module LoRa RFM95, qui est une référence peu chère permettant l'utilisation à l'international (ou presque) de l'unité locale ; en effet, elle communique sur la bande Industrielle, Scientifique et Médicale (ISM) de 866 MHz, utilisable sans licence dans la plupart des pays du monde (25 mW de PAR en France, et coefficient d'utilisation de 1 %).

En plus du module et du microcontrôleur, la carte de l'unité locale contient quelques composants passifs — pour le pull-up du bus I^2C par exemple —, des broches de raccordement pour les capteurs, ainsi qu'une antenne et un réseau (minimal) d'adaptation d'impédance. La réalisation d'antennes est un domaine complexe, et il n'est pas question d'une telle sophistication ici ;

5. Kais Mekkia, Eddy Bajica, Frederic Chaxela, Fernand Meyer, A comparative study of LPWAN technologies for large-scale IoT deployment. ICT Express 5-1. doi :10.1016/j.icte.2017.12.005

l'antenne a été réalisée selon un processus simple en deux temps :

1. réalisation naïve d'une antenne sur la largeur disponible, en respectant une longueur totale d'une moitié de la longueur d'onde — soit $\frac{c}{2\lambda} \approx 173\text{mm}$;
2. ajustement des espacements entre spires pour obtenir une impédance d'environ 50Ω — adaptation d'impédance pour éviter la réflexion à hautes-fréquence ; pour rappel l'impédance est donnée par le rapport $Z_{ant} = \sqrt{\frac{L_{ant}}{C_{ant}}}$.

2.1.3 Visualisation globale du code embarqué

Un algorigramme général de fonctionnement du code de l'unité locale peut être trouvé page 22. On y trouve le stockage local, expliqué partie 2.2.3 ainsi que le protocole de transmission très simple, attendant une réponse pour savoir si la donnée à bien été reçue par la passerelle. La mise en sommeil profond permet l'économie d'énergie lorsqu'aucune donnée n'est transmise, elle est pour le moment de 15 minutes ; juste après, le programme reprend son exécution au niveau de « Boucle infinie ».

2.2 Aspects de sécurité et de fiabilité

Afin de rajouter un niveau de protection supplémentaire, les paquets envoyés par l'unité locale sont chiffrés à l'envoi, puis déchiffrés par la passerelle au moment de leur arrivée. Cette sécurité ne vise non pas à éviter l'interception des données (ayant vocation à être publiques), mais bien à éviter l'insertion de fausses données par un attaquant. Par ailleurs, en cas d'absence de réponse de la passerelle à l'envoi d'un paquet, l'unité locale est capable de conserver des données, puis de les renvoyer plus tard ; c'est le sens du stockage local dans l'algorigramme proposé.

2.2.1 Chiffrement pour l'embarqué : état de l'art

Le système de chiffrement utilisé par le serveur, TLS, ajoute du chiffrement à la couche de transport ; c'est un protocole robuste et très sécurisé, mais aussi

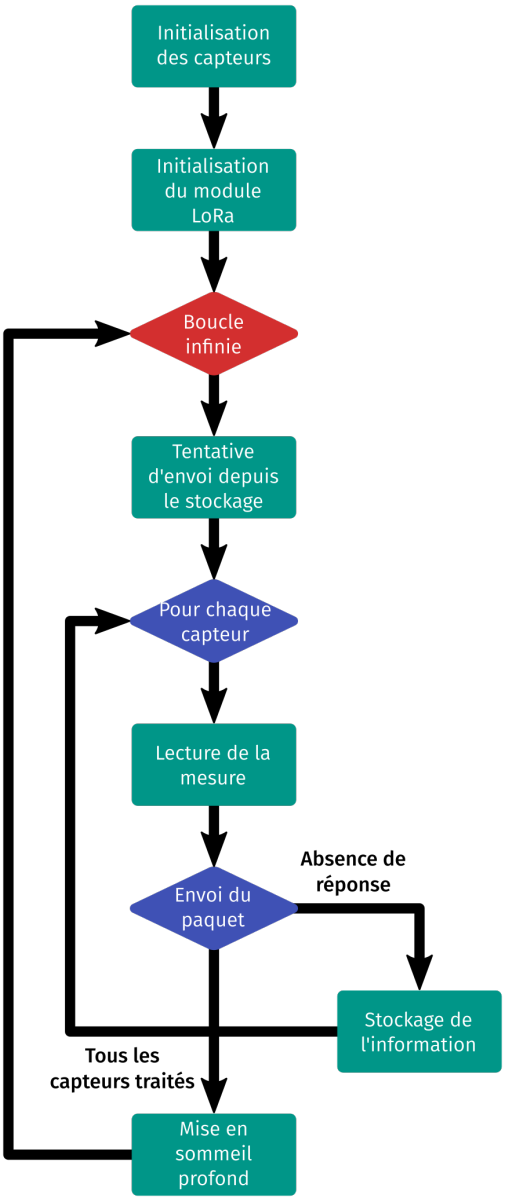


FIGURE 5 – Algorigramme de principe du code de l'unité locale

assez lourd. Une des plus légères implémentations, MbedTLS, dans sa version minimale, pèse près de 50 kilooctets en Flash, sans compter le programme autour ; c'est beaucoup trop pour un microcontrôleur avec 32 kilooctets de mémoire programme.

Par ailleurs, on pourrait tenter d'utiliser un algorithme simple, sans toute la couche de transport ; dans le cas qui nous intéresse, du chiffrement symétrique serait parfait, car la passerelle et l'unité locale peuvent partager un secret préalablement à la communication. On se tourne alors naturellement vers AES, le standard le plus connu ; même s'il est vrai que cet algorithme est fonctionnel sur les processeurs AVR, son fonctionnement est lent — plus de 40 000 cycles ou 10 millisecondes à 4MHz [6] et également gourmand en ressources, ce qui pourrait restreindre inutilement la taille du programme utile.

Heureusement, certains protocoles de chiffrement symétriques sont prévus spécifiquement pour les applications d'électronique embarquée. L'un des plus connus est Skipjack, mais de plus performants ont été conçus depuis ; c'est en particulier le cas de SIT [7], un algorithme récent pour le chiffrement embarqué.

Le projet SIT n'est pas en l'état un standard industriel ; aucun standard n'a d'ailleurs pour le moment émergé pour les puces de très faible puissance. La première implémentation publique de SIT en C pourrait donc être un plus pour son adoption dans d'autres domaines ; le code du projet côté embarqué — contenant donc également SIT — pourra être trouvé sur Github [8] pour plus de détails.

6. Biswas K., Muthukkumarasamy V., Wu XW., Singh K. (2016) Performance Evaluation of Block Ciphers for Wireless Sensor Networks. In : Choudhary R., Mandal J., Auluck N., Nagarajaram H. (eds) Advanced Computing and Communication Technologies. Advances in Intelligent Systems and Computing, vol 452. Springer, Singapore. doi :10.1007/978-981-10-1023-1_44

7. Muhammad Usman, Irfan Ahmed, M. Imran Aslam, Shujaat Khan and Usman Ali Shah, "SIT : A Lightweight Encryption Algorithm for Secure Internet of Things" International Journal of Advanced Computer Science and Applications(IJACSA), 8(1), 2017. doi :10.14569/IJACSA.2017.080151

8. `https://github.com/frareb/pi2p/tree/master/Task03`

2.2.2 Étude critique du système mis en place

Par identification avec le protocole de réseau, on peut établir un diagramme en couche de la pile de communication entre l'unité locale et la passerelle. Cette représentation, disponible figure 4, permet de bien visualiser les protocoles utilisés.

Il faut insister sur le fait que SIT n'est qu'un algorithme de chiffrement, pas une couche complète, on l'utilisera dans ce projet sur **la couche applicative**, ce qui n'est pas vraiment la tendance de ces dernières années — car la protection des métadonnées est réduite sur les couches hautes en particulier, ce qui amène à descendre le chiffrement le plus possible dans la pile.

La justification est très simple : la couche transport, LoRaWAN, ne supporte que le chiffrement AES, on doit donc nécessairement avoir du chiffrement sur la couche supérieure, sauf à vouloir réimplémenter la couche transport avec. Le besoin de sécurité étant relatif ici, on peut largement se contenter d'avoir un chiffrement applicatif.

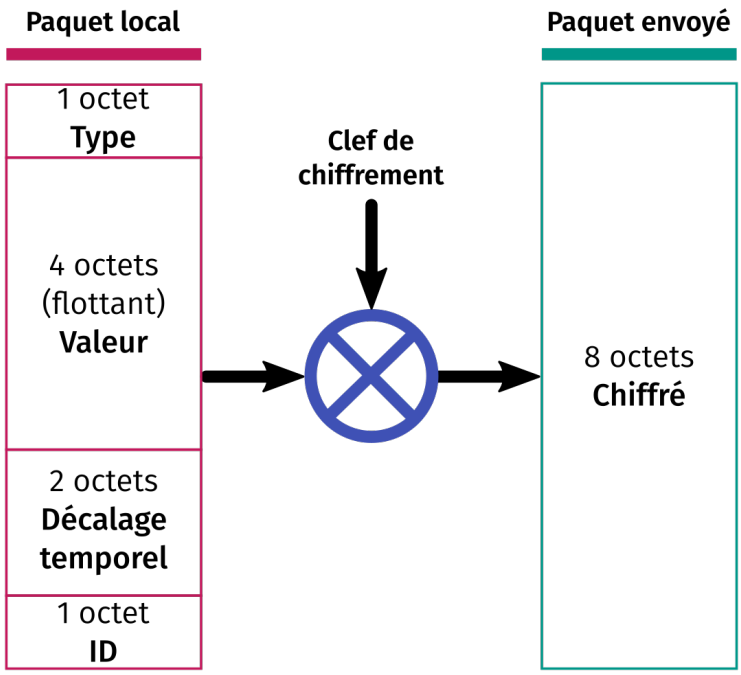


FIGURE 6 – Contenu d'un paquet et processus de chiffrement

La couche applicative complète est un simple paquet comme illustré figure 6 ; ce paquet est ensuite chiffré en utilisant l'algorithme SIT et envoyé à la passerelle, qui se chargera de le déchiffrer. Bien que le chiffrement assure une sécurité quant à l'insertion de fausses données, les plus attentifs pourront remarquer que le niveau de sécurité proposé reste vulnérable à (au moins) deux vecteurs d'attaque :

— la force brute — c'est-à-dire l'envoi de paquets aléatoires — d'habitude jamais mentionnée, est ici non-négligeable ; on remarque en effet que seuls $2^{16} = 65k$ paquets sont distinguables selon l'identifiant ou le type de capteur. Si trois type de capteurs différents sont reconnus, il reste une chance sur 20 000 de faire passer un paquet malicieux. Ce vecteur d'attaque peut être atténué **très facilement** en envoyant une réponse quel que soit le résultat du déchiffrement au niveau de la passerelle ; un attaquant n'aura ainsi aucun indice sur la validité de son paquet, et ne pourra pas augmenter le ratio ;
— l'attaque par rejeu, qui consiste à intercepter un paquet envoyé à la passerelle, et à le réutiliser. Cette attaque n'est malheureusement pas facilement atténuable, et pourrait causer des dégâts si une personne mal intentionnée souhaitait compromettre les résultats. Des méthodes de protection par jeton unique permettent de se protéger de ce type d'attaque, mais sont généralement complexes à mettre en œuvre ; quoi qu'il en soit, ce pourrait être un point d'amélioration dans le futur.

Un protocole de sécurité n'est de toute manière jamais parfait, mais on peut considérer ici qu'il remplit sa simple mission d'éviter que le système ne puisse être *trop facilement* détourné. Pour les personnes extérieures au projet, il est impossible de connaître ces vecteurs d'attaque, et des moyens importants devraient donc être mis en œuvre pour l'attaque, ce qui est improbable, la mission est donc remplie.

2.2.3 Conservation locale des données de terrain

Dans certains pays, la distribution de l'électricité n'est pas très fiable, et la passerelle pourrait donc être coupée, parfois pendant plusieurs heures. On souhaiterait pallier à ce problème via un système simple de stockage temporaire d'informations au niveau de l'unité locale. Ce stockage pourrait conserver des données pendant quelques heures, et tout envoyer par la suite,

en tenant compte du décalage temporel.

Pour cela, on commence par envoyer, conjointement avec la donnée, une information de décalage temporel, qu'on initialisera à zéro. Il suffit ensuite de s'arranger pour que la passerelle envoie une réponse lors de la réception de données, puis l'unité locale attendra quelques secondes cette réponse. Si elle n'est pas reçue, alors on suppose que la donnée n'a pas été traitée par la passerelle.

Au lieu d'envoyer immédiatement la donnée suivante (ou de se mettre en mode sommeil), l'unité locale va alors enregistrer l'information dans son EEPROM — qui n'était pas utilisée jusqu'à maintenant, c'est pourquoi elle a été choisie, au lieu de la RAM déjà bien remplie. Au prochain tour de boucle, les informations stockées vont être renvoyées, après leur avoir rajouté un décalage temporel approximatif de quinze minutes.

Ce décalage temporel est un problème car en mode sommeil, le microcontrôleur ne conserve aucune notion du temps. On doit donc supposer que nos quinze minutes sont exactes ; quoi qu'il en soit, cette supposition est exacte à 5 % d'après la fiche technique, ce qui est une précision largement suffisante pour la conservation, qui reste un cas exceptionnel.

2.3 Prototype de passerelle et vulgarisation au public

Cette section traite de deux meta-aspects liés à l'unité locale : la réalisation d'un prototype de passerelle, afin de tester les unités et le serveur, et la documentation réalisée concernant l'unité, en particulier concernant le montage d'icelle par des néophytes.

2.3.1 Réalisation d'un prototype de passerelle

Afin de pouvoir tester les unités locales réalisées — ainsi que le serveur de la partie suivante, il était nécessaire d'avoir une petite passerelle ; le prototype réalisé par François Rebaudo ne convenait pas, car il utilisait d'autres protocoles que ceux proposés ici. Il fallait donc repartir sur un nouveau prototype, qui conserve toutefois les technologies matérielles du premier (Raspberry Pi 3 avec connexion Ethernet).

L'idée d'un prototype est de pouvoir évoluer facilement par la suite ; le langage Python à donc été choisi sur cette partie, car l'équipe travaillant sur le projet est familière avec le langage, ce qui permettra son évolution dans le futur. Ce choix a demandé une étape supplémentaire avant l'écriture du code à proprement parler : la gestion du déchiffrement en Python.

Comme mentionné précédemment, l'algorithme SIT est implémenté en C pour le projet ; deux choix était donc possible pour l'intégration en Python : réécriture de l'algorithme dans le langage ou utilisation de *Foreign Function Interface* (FFI) pour appeler les fonctions C en Python. La deuxième solution a été choisie, car elle permet de ne pas avoir à maintenir en double l'algorithme de chiffrement.

Une fois cette complication gérée, il n'y a plus de grandes surprises sur l'algorithme final : réception d'un paquet, envoi d'une réponse, déchiffrement en utilisant la FFI mise en place, et envoi au serveur si le paquet est valide. Pour déterminer un paquet valide, on se base sur l'identifiant, initialisé à une valeur connue, et le type de mesure, qui ne peut prendre que trois valeurs valides.

2.3.2 Documentation sur la réalisation de l'unité locale

Le fait de devoir réaliser une carte électronique spécifique pour l'unité locale dégrade la volonté de simplicité d'assemblage ; le soudage des composants étant une compétence spécifique. Dans l'optique d'aider les personnes assemblant la carte par eux-mêmes, une documentation est fournie, permettant à la fois de comprendre comment souder la carte, la faire fonctionner, et également la modifier pour répondre à des besoins courants.

La première étape a été de préciser comment fabriquer la carte ; bien que certaines cartes devraient être fournies directement pré-soudées aux partenaires, il est bon pour le développement du réseau de permettre la fabrication de ces cartes. Le support de communication le plus simple pour cela est la vidéo [9] ; une vidéo détaille donc le processus de soudage de la carte ; elle est très simple pour le moment, et nécessitera quelques ajustements de forme, mais elle détaille tout le processus.

9. `https://nc.sld.ovh/s/tjcDZe4BaHHoLzG`

Le second vecteur de communication est la documentation [10], donnant les composants à acheter ainsi que quelques indications sur le montage et la programmation de la carte. Cette documentation n'est pas l'équivalent textuel de la vidéo, mais apporte au contraire des informations supplémentaires, sur la génération des clefs de chiffrement par exemple.

10. `https://frareb.github.io/pi2p_docu/docs/docClim02`

3 Côté serveur : réception, stockage et restitution de l'information

> On a beau dire, le vol ne rapporte jamais mais la restitution non plus.
>
> *Noël Audet*

Comme indiqué dans le cahier des charges, le serveur s'occupe d'une part de recevoir les données de la passerelle, et d'autre part de les renvoyer aux clients sur demande ; pour cela, et comme déterminé précédemment, on utilisera le même format de données à l'envoi et à la réception.

3.1 Choix techniques préliminaires

Au vu des contraintes sur la passerelle, en particulier des fortes discontinuités de réseau dans certains pays en voie de développement, la communication vers le serveur doit impérativement s'effectuer sans conservation d'un identifiant de connexion, qui pourrait être perdu lors de la coupure réseau. Il est impensable de recourir à une méthode de « connexion permanente » entre le serveur et la passerelle.

3.1.1 Les API REST

Une architecture logicielle répondant à ce besoin est l' API *REpresentational State Transfer* (REST) pour sa communication avec l'extérieur. Une API de ce type est basée sur un principe simple : une requête HTTP(S) équivaut à une action, et contient toutes les informations nécessaires à l'action : le jeton de connexion, les métadonnées, etc.

Une API REST permet également d'unifier l'accès aux ressources : chaque donnée disponible possède un point d'accès unique qu'on utilisera à la fois pour récupérer la ressource, mais aussi pour la modifier ou même la supprimer. Elle regroupe donc les tâches 4 (communication avec la Rasperry Pi), 5 (stockage

des informations en base de données) et 6 (récupération des informations par le client).

Du point de vue de HTTP, cela implique une *URL* unique pour l'accès à une même donnée (un même capteur par exemple), et le type d'opération à effectuer sera codé dans le type de requête — GET pour récupérer des informations, POST pour en ajouter et DELETE pour en supprimer, par exemple.

On distingue dans ces API deux types de structure de données : les ressources, qui sont des entités pouvant être identifiées de manière unique, et les collections, qui, comme leur nom l'indique, sont un regroupement de plusieurs ressources autour d'un thème commun.

En ce qui concerne les capteurs, situés sur les unités locales, on peut par exemple identifier un capteur comme une ressource, et l'ensemble des capteurs se situant en France comme une collection, et il existe une multitude d'autres collections tout aussi valides.

3.1.2 Choix du langage

Le choix du langage de programmation utilisé est toujours une décision partisane ; toutefois, je vais tenter ici de donner des arguments les plus objectifs possibles afin de justifier le choix de Node.js, qui est une plateforme fournissant les bibliothèques nécessaires afin de faire fonctionner du Javascript sur un serveur. Dans le cadre du projet, on pourra lui trouver les intérêts suivants :

— il est de conception asynchrone, ce qui permet de gérer plusieurs sessions parallèlement sans aucun souci ;
— le format JSON (*JavaScript* Object Notation), utilisé pour la plupart des API REST, émane directement de la représentation des objets utilisée dans le langage ; parser un JSON est une fonction native, ce qui n'est pas le cas dans certains autres langage, requérant une bibliothèque externe ;
— les structures de données sont équivalentes à celles trouvables sur du Javascript côté client, on pourra donc très facilement manipuler les représentations de types (et en particuliers les dates) sans conversion aucune du côté du client.

Enfin, l'écosystème est très développé, et on trouve des outils pouvant servir de base stable à la création d'un projet comme celui traité ici ; la bibliothèque choisie pour le projet est express.js [11], qui dispose d'une structure très simple et facilement compréhensible pour un intervenant externe.

3.1.3 Gestion de la base de données

Le choix de la base de données, comme pour le langage, est une décision plus ou moins arbitraire ; de plus, l'utilisation d'un ORM, comme ce sera le cas dans ce projet, abstrait totalement les spécificités des différents fournisseurs ; PostgreSQL, déjà utilisé sur le prototype, a donc été choisi pour la tâche.

Plutôt que de travailler en direct avec la base de données PostgreSQL, et comme évoqué ci-dessus, on utilisera un *Object-Relational Mapping* (ORM), permettant de gérer plus facilement la base de données. Le principe est que la base est abstraite en modèles, qui sont compatibles avec les représentations de données Javascript. Dans le cadre du projet, l'ORM utilisé est Sequelize [12].

La gestion des ressources est classique, avec un type de données = une table, exception faite des données de capteurs, qui seront stockées dans une table unique, **quel que soit le type de donnée** : ainsi, par exemple, température et pression sont dans un même table.

Pour distinguer les types de données, la table listant les capteurs — liée à la table de données par une relation — sera utilisée ; elle permet d'associer à la donnée son unité, ainsi qu'une description expliquant ce qu'elle mesure. Cette organisation étant un peu difficile à conceptualiser, un schéma est proposé figure 7.

3.2 Arborescence du projet et modèles de données

L'ensemble du code source en Javascript (pouvant être trouvé ici [13]) se décompose globalement en trois parties : un code côté serveur, caché des utilisateurs finaux, et s'occupant de toute la machinerie interne — formatage

11. `https://expressjs.com/en/starter/hello-world.html`
12. `https://sequelize.org/`
13. `https://github.com/frareb/pi2p/tree/master/Task04`

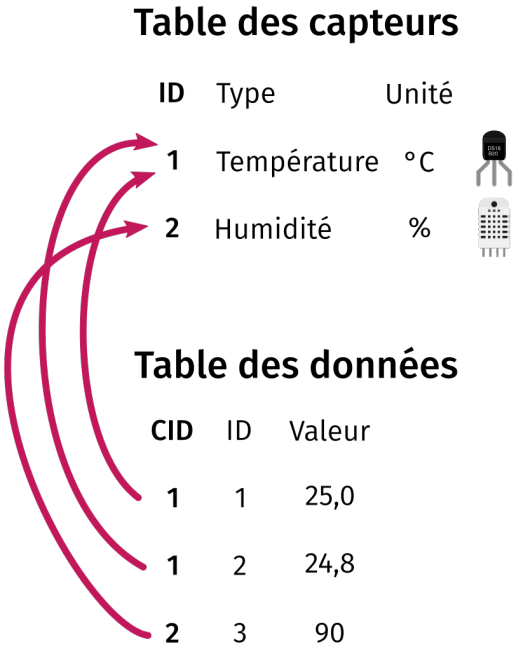


FIGURE 7 – Schéma relationnel pour la base de données

des réponses API, aiguillage des requêtes, etc — un système de gestion de l'affichage côté client (en particulier les graphiques de la tâche 7), et un ORM pour l'abstraction de la base de données.

3.2.1 Côté serveur : une architecture MVC

En ce qui concerne la partie purement serveur, l'architecture mise en place est de type Modèle-Vue-Contrôleur. Ce modèle, qu'on retrouve partout sur le Web [14], impose une séparation stricte du code, ce qui assure un certain degré de réusabilité.

Les objets du monde réel n'ont aucune signification pour les machines, il est donc nécessaire de leur expliquer comment ils fonctionnent ; on peut faire cela par une définition abstraite et l'ajout de propriétés de type connus autour de cette définition : c'est un modèle. Par exemple, un institut est une entité

14. Parmis les framework implémentant ce modèle, on notera Ruby on Rails, Symfony ou ASP.NET MVC

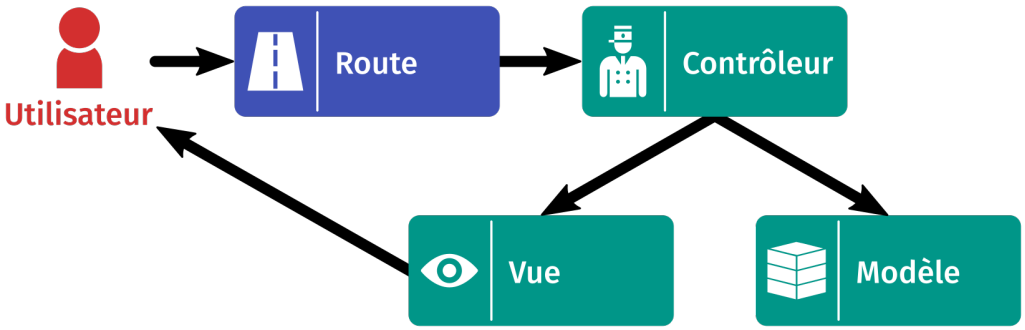


FIGURE 8 – Représentation du modèle MVC

abstraite possédant les propriétés suivantes : le pays où elle est située et son nom.

Les vues, quant à elles, sont des portions de code interagissant directement avec l'utilisateur final ; dans le cas d'une API, leur intérêt reste très limité, mais ce projet nécessitera des vues pour la réalisation des graphiques ou l'affichage de la bibliographie par exemple. Ici, un moteur de *templates* est utilisé, ce qui permet d'avoir des vues simples, proches de HTML, avec une souplesse ajoutée, permettant de charger dynamiquement certains éléments de la page (liste des instituts, des capteurs, etc).

Enfin, les contrôleurs sont les plus importants dans le cas qui nous intéresse ici ; en général, ils agissent en tant que tampon entre le modèle et la vue, recevant les informations de l'utilisateur et les convertissant vers des informations saines pour la base de données, représentée par les modèles. Ils se chargent en particulier de vérifier la conformité des informations entrées, et de construire une réponse cohérente à l'initiateur de la requête. Les points d'accès API s'adressent presque directement aux contrôleurs, mais il faut entre les deux une couche de gestion permettant de distinguer les différentes URLs ainsi que de décoder les différentes valeurs envoyées au format JSON par l'utilisateur ; tous les accès Web passent par une route, qui sont regroupées dans le dossier `routes/`.

3.2.2 Gestion du Javascript client

La distinction entre le Javascript client et serveur n'est pas nécessairement évidente pour qui n'en a jamais entendu parler ; le principe est simple : on exécute un maximum de choses au niveau du serveur, en particulier pour la gestion des informations et les requêtes gourmandes en énergie, et on se permet d'intégrer au niveau du navigateur certaines fonctionnalités ayant attrait à l'interactivité (zoom sur un graphique, visualisation du réseau de capteurs).

Cette séparation permet d'épargner le serveur, en lui évitant de devoir générer dynamiquement de nouvelles pages à chaque fois. La presque entièreté du rapport mentionne uniquement la partie serveur, car une grande partie du JS client a été écrit par François Rebaudo. La jointure entre les deux est l'objet de cette section.

Deux grandes parties ont permis l'intégration du côté client : la transformation des fichiers HTML statiques en fichiers générables dynamiquement et la génération du Javascript. Le problème est le suivant : on utilise sur la partie de Javascript client des fichiers externes, qu'on appelle **dépendances**, et qui visent à simplifier le développement en ne réinventant pas la roue. Ces dépendances sont à chaque fois un fichier de plus à charger, ce qui créé un double problème :

— dépendance à des fournisseurs externes pour les fichiers ;
— nécessité de multiples requêtes HTTP·S, alourdissant la page à chaque nouvelle dépendance.

Une solution simple à ce double problème est de créer des fichiers Javascript contenant les dépendances **et** le code local. L'intégration proposée utilise Webpack [15] et réalise cette « compilation » des dépendances.

3.2.3 Base de données : ORM, modèles et migrations

La base de données utilisée, PostgreSQL, est interfaçable via la langage SQL ; formater des requêtes SQL est peu pratique, d'autant plus qu'il faut vérifier la conformité des informations aux modèles. On peut résoudre ce

15. `https://webpack.js.org/`

problème en utilisant un outil appelé *Object-Relational Mapping* (ORM), qui se chargera pour nous d'écrire les requêtes SQL, en fournissant une interface adaptée autour.

Les modèles sont généralement proches de la représentation en base de données, on peut donc se contenter de créer des tables équivalentes aux modèles. C'est le rôle des migrations : elles permettent d'intervenir sur la base de données via l'ORM choisi, en créant de nouvelles tables, en ajoutant des champs, etc. Ces migrations permettent également de versionner la base avec des outils comme Git, garantissant que la base donnée correspond toujours au code fourni avec.

L'ORM s'assure également du lien entre les modèles ; une passerelle appartenant à un institut va ainsi disposer de l'identifiant de l'institut, qui sera lié directement au besoin, sans avoir à réfléchir au type de jointure à utiliser.

3.3 Gestion des accès et sécurisation des données

Un des intérêts d'une API REST est que, contrairement à un accès web classique, aucune variable de session ou autre état volatile n'est enregistré côté serveur ; chaque requête vers le serveur est donc indépendante.

3.3.1 Les clefs API

Cette caractéristique facilite grandement la gestion des accès aux ressources. Dans le cas où une authentification (c'est-à-dire la vérification d'une identité, réelle ou numérique) n'est pas nécessaire, on peut identifier uniquement les entités ayant le droit d'accéder ou d'ajouter des données sur chaque point d'accès à partir d'une clef d'accès.

L'idée est que chaque acteur dispose d'une clef unique aléatoire ; cette clef permet à la fois d'identifier l'entité à l'origine de l'action (la passerelle au Kenya par exemple), et de s'assurer que sa requête est légitime (contrôle d'accès). L'idée est que les clefs générées sont uniques et suffisamment longues pour ne pas pouvoir être trouvées de l'extérieur.

Ces clefs d'API sont générées aléatoirement et cryptographiquement sûres ; pour être prise en compte, la clef doit être ajoutée dans l'entête 'Authorization'

de la requête, sous forme de ‘Bearer‘. Une clef invalide ou une absence de clef fera automatiquement basculer l'utilisateur dans le groupe par défaut, qui dispose de très peu de permissions (accès en lecture uniquement).

3.3.2 Séparation en groupes

Plutôt qu'une gestion d'accès par utilisateur, le contrôle d'accès mis en place propose une séparation en groupes, avec un affinage possible au sein des groupes pour l'accès à certaines URL. Chaque groupe dispose d'une ou plusieurs clefs d'API, lui donnant accès aux ressources de son groupe, et éventuellement à des ressources spécifiques.

Les autorisations sont gérées globalement dans le groupe (par exemple : tous les administrateurs peuvent créer de nouveaux instituts), puis certains paramètres des URLs sont spécifiques à certaines propriétés du groupe (par exemple : une passerelle ne peut ajouter des données qu'a ses capteurs).

Pour conclure, ce stage semble avoir apporté aux deux parties. Du point de vue du laboratoire, la réalisation d'une carte d'électronique embarqué ouvre de nouvelles perspectives : c'est une ouverture relativement récente dans le cadre de la recherche en écologie, qui pourrait grandement se développer dans les prochaines années.

Par ailleurs, les possibilités ouvertes par ce stage s'inscrivent pleinement dans un projet de poursuite d'études dans la recherche : contacts professionnels, découverte du milieu, mise en avant de la participation à un projet, etc. Pour terminer ce rapport, quelques pistes d'amélioration sont proposées sur les aspects traités du projet ; elles peuvent constituer un sujet de stage pour un étudiant suivant, ou être réalisés parallèlement aux aspects futurs du projet, mentionnés section 1.2.2.

On propose d'abord l'étude sérieuse des possibilités en matière d'optimisation du lien radio ; la technique proposée pour l'antenne de l'unité locale n'est en rien professionnelle, bien qu'elle permette d'obtenir un résultat correct, et rien n'est proposé au niveau de la passerelle. De même, quelques améliorations en termes de sécurité sont sans doute possibles, l'ajout d'une somme de contrôle par exemple serait souhaitable ; on peut également envisager, si c'est nécessaire, la mise en place d'une sécurité par jeton unique.

Ensuite, il faudra évidemment revoir la passerelle presque complètement ; il est bien précisé que ce qui a été réalisé jusqu'à présent est *un prototype*, présentant de nombreux défauts ; en particulier, il n'est pas robuste face aux erreurs dans les paquets reçus, ou lors d'une indisponibilité du serveur.

Enfin, en ce qui concerne le serveur, quelques améliorations pourraient être portées sur la qualité du code, et sa généricité — c'est-à-dire éviter les répétitions. Le code écrit est en effet assez brut par endroit, et quelques optimisations sont sans doute possibles, mais il n'en reste pas moins que le serveur est dans l'ensemble robuste, et surtout très sécurisé, puisqu'un grand soin a été apporté à la partie gérant l'authentification.