RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

École Nationale Polytechnique

Université Paris Saclay

Département d'électronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

---

Étude et conception d'un système de ruche connectée

BOUCHEFER Yousra
ROUIZI Lilia

Présenté et soutenue le 12/07/2021

Composition du jury :

Président	**M. Cherif LARBES**	**Prof.**	**ENP**
Examinateur	**M. Rabah LOUALI**	**Dr.**	**Ecole Militaire Polytechnique**
Promoteur	**M. Samir BOUAZIZ**	**Prof.**	**Université Paris Saclay**
Promoteur	**M. Mourad ADNANE**	**Prof.**	**ENP**

ENP 2021

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

École Nationale Polytechnique

Université Paris Saclay

Département d'électronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

---

Étude et conception d'un système de ruche connectée

BOUCHEFER Yousra
ROUIZI Lilia

Présenté et soutenue le 12/07/2021

Composition du jury :

Président	**M. Cherif LARBES**	**Prof.**	**ENP**
Examinateur	**M. Rabah LOUALI**	**Dr.**	**Ecole Militaire Polytechnique**
Promoteur	**M. Samir BOUAZIZ**	**Prof.**	**Université Paris Saclay**
Promoteur	**M. Mourad ADNANE**	**Prof.**	**ENP**

ENP 2021

Dédicaces :

On dédie ce travail simplement et sincèrement à nous-mêmes, à nos parents, nos frères et sœurs, et à toute personne chère à notre cœur.

Remerciements :

Ce mémoire marque un tournant décisif dans notre vie, la fin d'un long périple éducatif, et le début de nouvelles aventures. Il est le fruit de notre dur labeur, concrétisé grâce au travail d'équipe irréprochable dont nous avons fait preuve ma binôme et moi, mais aussi à l'aide indéniable que nous a apporté notre entourage. C'est pourquoi nous tenons à remercier toute personne ayant contribué de près ou de loin à la réussite de ce travail.

Nous exprimons dans un premier temps notre reconnaissance à nos deux promoteurs le Professeur **BOUAZIZ Samir** de l'université **Paris Saclay** et le Professeur **ADNANE Mourad** de l'**Ecole Nationale Polytechnique**, pour leur encadrement, leur disponibilité et leurs judicieux conseils qui ont alimenté notre réflexion.

Nous témoignons toute notre gratitude à tous nos enseignants du département électronique qui nous ont suivi durant ces années d'études, et ce jusqu'à l'aboutissement de ce projet et tout particulièrement les Professeurs **LARBES Cherif** et **HADDADI Mourad.**

Nous remercions également nos camarades d'études des promos 2020, 2021 et 2022, en particulier **SADOUN Sarah, NENNOUCHE Mohamed, ATCHI Malik, BELMOUMENE Hakim** et **BOUCHELLAL Mohamed Larbi,** qui de par leurs interventions, leurs paroles, leurs conseils et leurs critiques nous ont apporté une aide considérable et fondamentale à l'achèvement de cet écrit.

Nous désirons remercier de façon très particulière un de nos anciens enseignants très cher à notre coeur pour l'assistance et le soutien incontestable qu'il nous a fourni sans quoi notre projet ne serait qu'une ébauche.

Enfin, et pas des moindres, nous rendons grâce à nos parents, nos frères et sœurs et à toute notre famille, pour leurs contributions et leurs constants encouragements.

ملخص

تتسلل التكنولوجيا إلى أسلوب حياتنا اليوم ولا قطاع يُستثنى منها ، بما في ذلك تربية النحل. المشقات التي يواجهها مربي النحل متنوعة ومتطلبة، ونظام خلية النحل المصمم يعمل على تخفيف الوزن على كتفيه وتسهيل عمله.

يتكون مشروعنا من إنتاج نظام مُجهز يتكون من شبكة من أجهزة الاستشعار وبطاقة برمجة وجهاز اتصال يختتم بمراقبة فورية للبيانات المتعلقة بحالة الخلية التي يتم توفيرها لمربي النحل عبر لوحة القيادة.

الكلمات المفتاحية : نظام خلية النحل المتصلة، شبكة أجهزة الاستشعار، بروتوكول MQTT، لوحة القيادة.

Abstract

Technology is creeping into our way of life today and no sector is spared, including beekeeping. The constraints faced by the beekeeper are diverse and demanding, and the connected beehive system designed serves to lighten the weight on his shoulders and make his work easier.

Our project consists of realising an instrumented system made up of a sensors network, a treatment card and a communication device, concluding with instantaneous monitoring of data relating to the state of the hive made available to the beekeeper via a dashboard.

Key words : Connected Beehive, sensors network, MQTT protocol, Dashboard.

Résumé

La technologie s'immisce aujourd'hui dans notre train de vie, et aucun secteur n'est épargné, y compris celui de l'apiculture. Les contraintes affrontées par l'apiculteur sont diverses et éprouvantes, et le système de ruche connectée conçu sert à alléger le poids sur ses épaules et lui faciliter son travail.

Notre projet consiste à réaliser un système instrumenté, constitué d'un réseau de capteurs, d'une carte de traitement et d'un dispositif de communication concluant par un suivi instantané des données relatives à l'état de la ruche mis à disposition de l'apiculteur via un dashboard.

Mots clés : Ruche connectée, réseau de capteurs, protocole MQTT, Dashboard.

Table des matières :

Liste des Tableaux :

Liste des Figures :

Liste des Acronymes :

CO2 : Dioxyde de Carbone
WiFi : Wireless Fidelity
MQTT : MQ Telemetry Transport
I2C : Inter Integrated Circuit
SPI : Serial Peripheral Interface
UART : Universal Asynchronous Receiver Transmitter
USB : Universal Serial Bus
CAN : Controller Area Network
ADC : Analog Digital Converter
DAC : Digital Analog Converter
PWM : Pulse Width Modulation
PPM : Partie Par Million.
AH : Ampère Heure
DT : Digital OUTput
SCK : Serial ClocK
3D : 3 Dimensions
A/N : Analogique/Numérique
SDA : Serial DAta
SCL : Serial CLock
GND : Ground
VCC : Voltage at the Common Collector
DC : Direct Current
CSB : Chip Select Bar
TVOC : Composants Volatiles Organiques Totaux
MOX : Oxyde Métallique
COV : Composants Volatiles Organiques
PPB : Partie Par Billion
DIP : Dual Inline Package
RAM : Random Access Memory
E/S : Entrées/Sorties
MCU : Microcontroller Unit
GPIO : General Purpose Input Output
SDK : Software Development Kit
IDE : Integrated Development Environment
OS : Operating System
SOC : System On Chip
IP : Internet Protocol
TCP : Transmission Control Protocol

Tx : Transmit
Rx : Receive
CH-PD : CHip Power-Down
I/O : Input/Output
Rst : Reset
FTDI : Future Technology Devices International
M2M : Machine to Machine
IOT : Internet Of Things
IBM : International Business Machines
API : Application Programming Interface
QoS : Quality of Service
LWT : Last Will and Testament
DNS : Domain Name System
TSDB : Time Series DataBase
NTP : Network Time Protocol
HTTP : Hypertext Transfer Protocol
PCB : Printed Circuit Board.

Introduction générale :

Nées il y a 65 millions d'années, les abeilles ont un rôle indispensable dans notre environnement, ces insectes sont essentiels pour le maintien de la biodiversité végétale et œuvrent fortement à conserver la reproduction des plantes et des fleurs. [64]

Du fait de leur mission principale qui est la pollinisation, les abeilles constituent un maillon essentiel de la chaîne qui contribue à sauvegarder l'équilibre des écosystèmes, ceci consiste en un transport de grains de pollen d'une fleur à une autre permettant ainsi la fécondation et la reproduction de nombreuses espèces végétales. [64]

Ainsi, les abeilles aident à la survie et à l'évolution de 80% des espèces de plantes à fleurs, ce qui rend leur valeur inestimable aux humains comme aux animaux, c'est pourquoi il est primordial de s'intéresser à ces dernières et à leur milieu de vie pour mieux les préserver. [64]

L'apiculture qui est une technique d'élevage spécifique aux abeilles, permet depuis des milliers d'années la collecte du miel, source de santé et de bien-être. Elle contribue aussi à la gestion saine des populations d'abeilles domestiques, néanmoins cette dernière doit être conforme à certaines exigences qui assurent un environnement de qualité aux abeilles, quotidiennement contrôlé, ainsi que la disponibilité de vastes espaces de butinage exempts de substances toxiques. [64]

C'est dans ce cadre que rentre le rôle de l'apiculteur qui doit se charger de l'entretien de la ruche, de la surveillance de l'activité des abeilles, des soins nécessaires à la survie de la colonie et bien sûr de la récolte du miel. Pour ce faire, ce dernier doit maîtriser les différentes techniques nécessaires en apiculture, assurer un suivi au quotidien et reconnaître les cas d'alerte pour y remédier. [64]

Pour toutes ces raisons, nous nous sommes intéressées à la conception d'un système qui permettra de faciliter la tâche à l'apiculteur mais aussi de mieux protéger les abeilles dans la ruche, qui consiste en "une ruche connectée". [64]

Ce système comporte un réseau de capteurs pour la collecte de différentes données environnementales, entres autres poids, température, humidité, pression, luminosité et taux de qualité d'air autour de la ruche, pour s'assurer que ces différents paramètres sont conformes aux normes et alerter l'apiculteur dans le cas contraire.

Ce réseau sera lié à une carte de traitement mbed NXP LPC1768 qui permettra de communiquer ces données via un réseau wifi permettant à l'apiculteur d'accéder aux différents changements à travers un dashboard réalisé sur le web.

Chapitre 1 : Notions d’apiculture

1.1 Les abeilles et leur mode de vie :

Depuis très longtemps, l'abeille nous subjugue par ses facultés à produire une essence naturelle qui exalte notre palais et nourrit les légendes : le miel. La vie de cet hyménoptère est fascinante et indispensable à l'équilibre des écosystèmes. [59]
L'abeille est un insecte social qui vit en interdépendance avec ses congénères. Elles forment une colonie composée de plusieurs dizaines de milliers d'ouvrières, de quelques centaines de faux bourdons et d'une seule reine, donnant ainsi une société réputée pour son organisation et sa discipline. [9,64]
L'existence de l'abeille peut être schématisée de façon immuable, et le rang de chaque insecte est prédéfini à sa ponte : reine, ouvrière ou faux-bourdon.

La reine :

La reine est pondu au printemps, pour succéder au règne de sa mère en fin de vie ou pour l'essaimage, elle se distingue des ouvrières par une corpulence plus importante et une raréfaction des poils. Son principal rôle est de s'accoupler avec les faux-bourdons et assurer ainsi la pérennité de l'espèce. [58]

Figure 1.1 La reine.

Les faux-bourdons :

Les faux-bourdons, plus gros que les ouvrières et dépourvus de dard, dépendent entièrement de ces dernières pour leur alimentation et leur rôle se résume à la fécondation de la reine.
Ceux qui s'accouplent meurent peu de temps après, le reste est chassé par les ouvrières à l'arrivée de la saison automnale. [9, 58,64]

Figure 1.2 Un faux-bourdon.

Les ouvrières :

Le maillon essentiel de la ruche, elles se répartissent en deux catégories : ouvrières d'été et ouvrières d'hiver. [9,58,64]
Leur évolution suit un véritable plan de carrière qui commence dès le premier jour de leur vie :

- Agent d'entretien : elle nettoie les alvéoles avant que la reine re-ponde un nouvel œuf et réchauffe le couvain, elle garde la ruche propre et en bonne santé.
- Nourrice : au 5ème/6ème jour de sa vie, l'ouvrière devient une nourrice et alimente les larves avec de la gelée royale qu'elle fabriquera et régurgitera. Les nourrices prodiguent des soins pointilleux aux larves qui sont alimentées régulièrement et reçoivent plus de 7000 visites de contrôle.
- Architecte : âgées de 8 jours, les maçonnes s'attèlent à la confection des rayons de la ruche et à leur entretien grâce aux glandes à cire de leur abdomen, c'est une tâche ardue qui nécessite un travail collectif et coordonné.
- Magasinière : du 10ème au 20ème jour de leur vie, les ouvrières mettent en réserve la récolte de nectar et de pollen dans les alvéoles.
- Ventileuse : en moyenne âgées de 18 jours, les ventileuses battent des ailes pour aérer la ruche et contrôler sa température, ses taux d'humidité et de gaz carbonique. La ventilation sert également à la contribution à l'évaporation de l'eau excédentaire contenue dans le nectar. Lors de l'essaimage, les ventileuses ont pour autre mission de battre le rappel pour permettre le regroupement de l'essaim.
- Gardienne : vigile postée à l'entrée de la ruche, la gardienne empêche de rentrer tout intrus comme les guêpes, les papillons et même les faux-bourdons à l'arrivée de l'automne.

- Butineuse : à partir du 20è jour, elle adoptera la fonction de butineuse jusqu'à la fin de sa courte existence, qui ne dépassera pas six semaines. Elle approvisionne la ruche en nectar, en pollen, et propolis, en miellat ou en eau. Généralement elle mourra d'épuisement lors d'un dernier vol d'approvisionnement. [9,58,64]

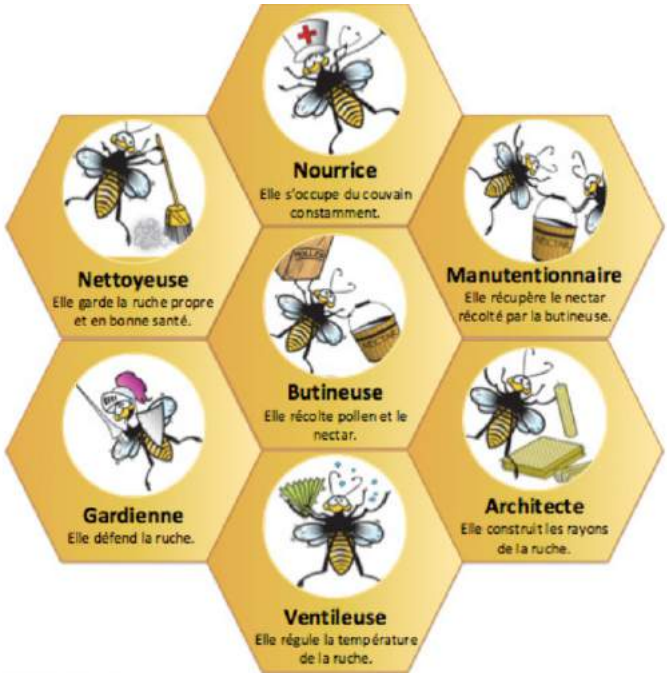

Figure 1.3 Les différents métiers de l'ouvrière.

L'ouvrière d'hiver a des fonctions identiques à celles de ses sœurs estivales pendant les premiers jours de sa vie, mais, à compter du vingtième jour, au lieu de devenir butineuse, elle reste confinée dans la ruche. Elle œuvrera pendant cinq à six mois à maintenir l'essaim en bon état pour lui permettre de franchir sans problèmes la saison froide. Enfin, un peu avant l'arrivée du printemps, elle s'appliquera à la préparation de la naissance des nouvelles générations.

Figure 1.4 L'ouvrière.

1.2 Habitat et environnement des abeilles :

L'habitat des abeilles est divers et varié. Il peut être un abri naturel, une cavité, une saillie dans une roche, un arbre creux, mais celui qui nous intéresse le plus n'est autre que "la ruche".
La ruche est une construction artificielle créée par l'homme dans le but d'attirer un essaim en vue de le faire produire et donc de récolter le miel. Elle représente un compromis entre les besoins de la colonie et les attentes de l'apiculteur en termes de quantité de miel produit.[10,64]

Figure 1.5 La ruche.

1.2.1 Structure de la ruche :

La ruche est composée de différents compartiments qui ont chacun une fonction bien précise [11] :

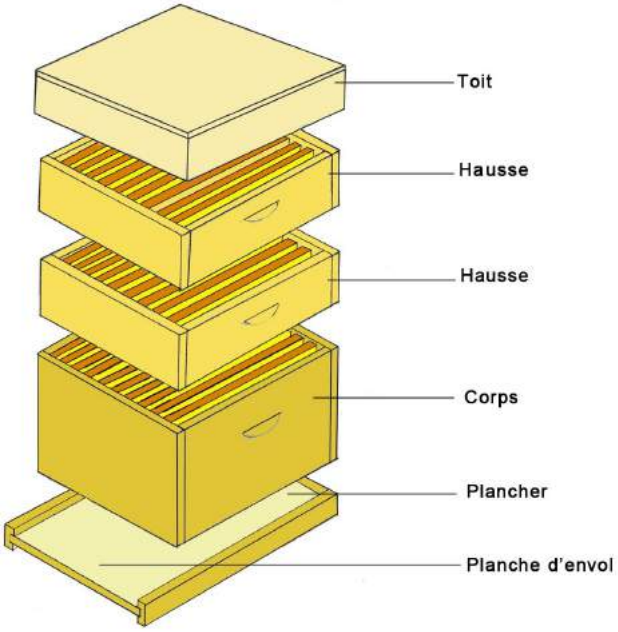

Figure 1.6 Structure de la ruche.

- **La planche d'envol :** cette partie située tout en bas représente la zone aéroportuaire de la ruche, au-dessus de cette dernière se trouve un trou de vol qui est un espace laissé libre pour différentes raisons telles que le passage des abeilles qui entrent et sortent du rucher, la respiration de la colonie, ou même l'évacuation des déchets et cadavres en dehors de la ruche. Ces raisons font que cette partie est très importante et permet d'alerter l'apiculteur en cas de souci dans le rucher.
- **Le corps de la ruche :** c'est dans cette partie que vit la colonie d'abeilles, elle comprend un nombre de cadrans (allant de 5, 8, 10 ou 12) selon la taille de la ruche qui servent de chambre à couvain. Ces cadrans sont bâtis de rayons de cire, c'est là que la reine pond ses œufs et que les ouvrières s'occupent des larves et emmagasinent miel et pollen.
  En général, une grille à reine est disposée juste au-dessus du corps, conçue de façon à ce qu'elle soit de taille suffisante pour faire passer les ouvrières mais trop étroite pour que la reine puisse s'y frayer un chemin. Elle permet de fermer l'accès des hausses à la reine et de récolter le miel sans aucun risque pour elle. Cela aide à ne pas troubler la récolte puisque le couvain est cantonné dans le corps. Le miel récolté est aussi exempt de couvain ce qui veut dire qu'il ne contient ni de spores ni de bactéries.
- **La hausse :** vient juste au-dessus du corps de la ruche et est deux fois moins haute que ce dernier, elle est placée par l'apiculteur au moment de la miellée et joue le rôle de magasin de miel pour l'extraction de ce dernier. Selon l'efficacité de la colonie et l'abondance du nectar, il est possible de placer plusieurs hausses.
  Un cadre de hausse plein à bloc peut peser jusqu'à deux kilogrammes.
- **Le couvre cadre et le toit :** Le couvre cadre délimite la partie dédiée aux abeilles et joue le rôle de bouclier thermique, il est aussi utilisé pour alimenter ces dernières. Le toit, quant à lui, est en tôle et permet de protéger la ruche du vent, des courants d'air et de tous genres d'intempéries (pluie, neige…), mais aussi des prédateurs qui peuvent nuire aux abeilles en quête de miel.

Une coupe à l'intérieur de la ruche donne ce qui suit :

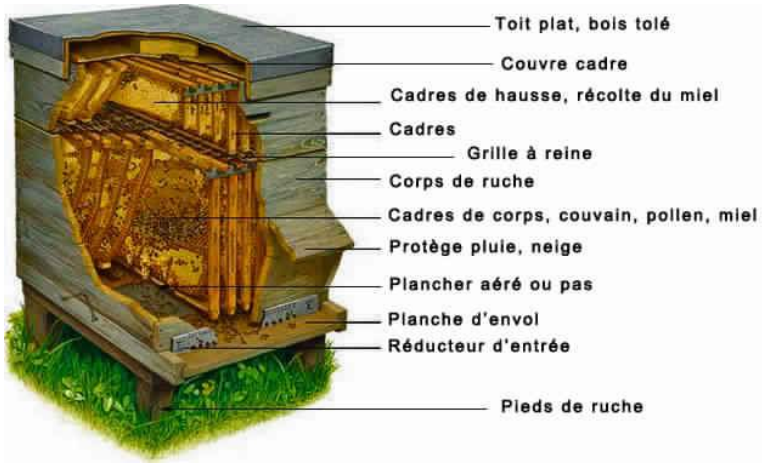

Figure 1.7 Coupe au niveau des différents compartiments de la ruche.

Au sein d'une ruche, des milliers d'abeilles cohabitent et s'entraident pour un seul but commun: la survie de la colonie. Dans ce cadre entrent plusieurs facteurs qui font qu'une ruche soit dans un environnement sain et propice à la vie de ses abeilles.

1.2.2 Paramètres physiologiques d'une ruche :

- **La température :**

Une température idéale au sein de la ruche est de 34-35°C**,** elle ne doit jamais dépasser ces seuils. Le couvain ou corps de la ruche doit rester au chaud pour se développer convenablement.
Pour produire de la chaleur, les abeilles utilisent les muscles qui font bouger leurs ailes mais en maintenant celles-ci immobiles. Cette tension musculaire sans mouvement permet de réchauffer la partie de la ruche réservée au couvain.
En cas contraire, lorsque la température augmente, une thermo-régulation est assurée par l'évaporation d'eau dans la ruche ainsi qu'une ventilation. Les abeilles assurent une ventilation dans la ruche afin de faire circuler l'air de l'intérieur vers l'extérieur. Cette opération a pour objectif de rafraîchir l'air dans la ruche. [12,64]

- **L'humidité :**

L'humidité dans la ruche est tout aussi importante que la température, sa valeur doit être maintenue entre 50 et 70%, en cas de baisse ou hausse de cette dernière, les abeilles opèrent, tout comme dans le cas de la température, une évaporation ou une ventilation, ce qui permet de rétablir l'humidité nécessaire à la survie de la colonie.
Il est à noter que ce taux d'humidité n'a rien avoir avec le taux nécessaire à la conservation du miel dans la ruche qui est tout autre. [12,64]

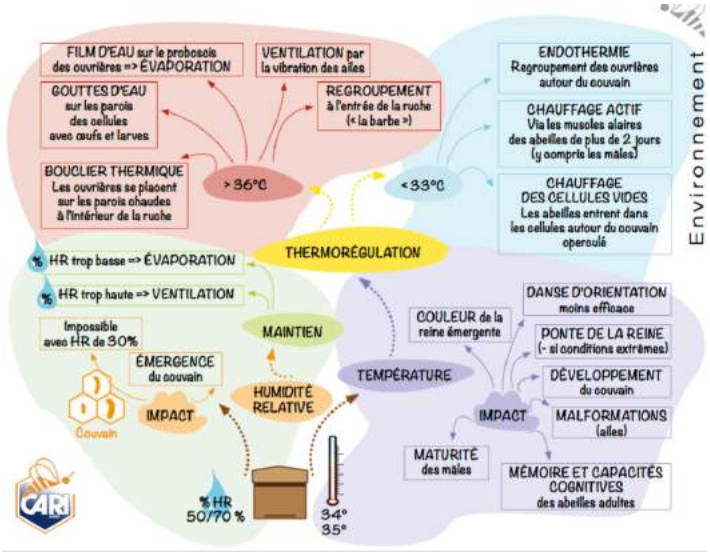

Figure 1.8 La thermorégulation dans une colonie d'abeilles.

- **La lumière du soleil :**

Elle est primordiale dans l'activité de butinage des abeilles, ces dernières sont dotées de différents organes qui leur permettent de détecter la direction et les modifications de la clarté. Ces variations de l'intensité de la lumière aident les abeilles à se diriger hors et dans la ruche. La position du soleil dans le ciel et l'intensité de son rayonnement permettent aux abeilles de se repérer dans la journée et autour de la ruche. [14,64]

- **Taux de CO2 :**

Les abeilles sont très sensibles au taux de CO2 se trouvant dans la ruche [64]. Plusieurs études ont établi différents seuils de ce gaz que les abeilles peuvent supporter en fonction de leur âge, les plus jeunes sont beaucoup plus touchées par ce facteur, néanmoins dans un cas général, elles sont toutes affectées lorsque la dose de ce gaz dépasse une certaine limite, ce qui fait que la population d'abeilles soit anesthésiée, et a beaucoup de mal par la suite à récupérer ses sens.

- **Le poids :**

La prise ou la baisse de poids d'une ruche traduit l'interaction de la colonie d'abeilles avec son environnement. Les apiculteurs s'approprient généralement cet outil pour mieux prévoir les miellées en saison apicole ou les famines en période froide ou sèche. Le praticien doit savoir interpréter ces données de poids pour en tirer des informations utiles à des fins médicales.
Il est possible de comprendre une colonie d'abeilles en la pesant, par exemple les ouvrières butineuses récoltent principalement de l'eau, du nectar et du pollen qu'elles rapportent à la ruche. Ces apports permettent de satisfaire aux besoins vitaux de la colonie, à savoir se nourrir, réguler la température dans la ruche, maintenir en vie le couvain, ... Les entrées versus les sorties (ou les apports versus les consommations) varient donc en fonction de l'activité de la colonie. L'évolution de la masse d'une colonie entre deux instants traduit ainsi le bilan comptable de ces échanges. [16,64]
Les points les plus importants à suivre dans les variations de poids d'une ruche sont les suivants:

1. Suivi des miellées :

Suivre quotidiennement la masse d'une ruche permet de savoir si son poids progresse ou s'il diminue. En saison apicole, une augmentation importante de la masse, régulière et continue, annonce donc une miellée, la donnée de poids indique directement le départ ou l'arrêt de cette miellée, son intensité et finalement la quantité de miel produit. [16,64]

2. Evolution des réserves hivernales :

En dehors des miellées, les suivis de poids permettent de suivre le niveau des réserves dont dispose la colonie. [16,64] Cette information devient vitale en hiver : dans sa lutte contre le froid, la colonie doit produire de la chaleur. Cette thermogenèse s'appuie sur la consommation des sucres stockés, donc des réserves de miel, qui peut parfois dépasser 2 kg par mois. Si ces réserves atteignent un seuil trop bas, en particulier en fin d'hiver ou en période pré-hivernale, la survie de la colonie est compromise. En connaissant le poids de la ruche, l'apiculteur aura une information globale utile sur l'état des réserves moyennes de son rucher, ceci lui permettra de prévoir à l'avance si des apports de nourrissements seront utiles et quand ils seront nécessaires.

3. Suivi de la santé et de l’évolution de la colonie :

Les données collectées par l’apiculteur sur le poids peuvent lui permettre d’en tirer plusieurs informations sur l’état de santé de ses colonies.

Une variation trop importante de poids, ou trop rapide à des périodes sans activité importante peuvent alerter l’apiculteur sur un état anormal de la ruche. Toute anomalie devrait donc susciter une visite de la colonie par ce dernier pour un check-up. [16,64]

Quelques exemples de conditions qui causent une anomalie de poids :

- La présence de frelons asiatiques dans les environs ou autres spécimens dangereux pour les abeilles.
- Un problème au sein de la ruche qui fait que les abeilles n’y pénètrent ou n’en sortent plus...etc

1.3 Rôle de l’apiculteur :

Au gré des saisons et de l’activité des abeilles, les ruches ont plus ou moins besoin de soins et d'attention. Le travail de l'apiculteur est de bien connaître la cadence de la colonie et être attentif à son développement et à son évolution sans pour autant l’importuner. [17,64]

Après la saison froide, l’apiculteur doit vérifier la taille et la qualité du couvain, Un mauvais état du couvain peut être signe de la défaillance ou de la vieillesse de la reine qui nécessiterait un remplacement. La colonie peut être affaiblie exigeant un renforcement ou un rassemblement à une autre colonie pour former une plus forte, capable de subsister. [17,64]

Il faut également vérifier si les provisions de miel sont suffisantes ou si la colonie a besoin d'un complément : sirop, miel...

Aux beaux jours, l'apiculteur a de nouvelles préoccupations : il doit éviter que les colonies essaiment, ce qui suscite une surveillance quotidienne de la colonie. Si la saison et la floraison s'annoncent bonnes, il doit ouvrir plus largement les portes et placer des hausses pour laisser toute la place nécessaire à une production abondante de miel. [17,64]

En été, il faut s'assurer que la ruche ait un point d'eau à proximité, indispensable pour maintenir une température convenable à l'intérieur de la ruche et produire leur miel dans les meilleures conditions. Enfin quand les hausses s’alourdissent, il sera temps de la récolte. [1764]

En présage pour la saison froide, la survie de la colonie nécessite environ 18 kg de miel, si les réserves sont insuffisantes, l'apiculteur se doit de les compléter en fournissant du miel, du sirop ou encore du candi, en le donnant assez tôt pour que les abeilles puissent le mettre au chaud avant les grands froids. [17,64]

En hiver, la règle d’or est de "ne pas déranger", le travail de l'apiculteur devient ainsi limité : surveiller de loin et protéger des intempéries. Toute intrusion dans la ruche risquerait de perturber l'effort collectif de la colonie pour augmenter la température. [17,64]

1.4 Conclusion :

- ➢ L'univers des abeilles est fascinant par son organisation et par le mode de vie qu'adoptent ces dernières pour assurer leur survie en toutes circonstances.
- ➢ Suite à l'étude des différents facteurs qui peuvent affecter la ruche, il est nécessaire dans un premier temps de bien choisir l'emplacement de cette dernière, par rapport aux agents : température, humidité, lumière et taux de CO2, mais aussi de contrôler ces paramètres de sorte à s'assurer qu'ils sont bel et bien conformes pour un bon environnement des abeilles sans nuire à ces dernières.
- ➢ Quant aux variations de poids dans la ruche, il est primordial d'assurer un suivi de ces dernières de manière continuelle car ce paramètre en dit long sur l'état du rucher et peut alerter l'apiculteur en cas de différents problèmes qui peuvent surgir.
- ➢ Le rôle de l'apiculteur est indispensable à la survie de la colonie d'abeilles, cependant la gestion de plusieurs ruchers peut s'avérer contraignante, à laquelle s'ajoute des cas d'alertes ou de problèmes qui peuvent malheureusement échapper à l'apiculteur ou auxquels il ne peut remédier à temps, ce qui cause une perte à la fois pour les abeilles et pour ce dernier.
- ➢ C'est dans ce cadre que rentre le système de la ruche connectée permettant d'alléger la charge de l'apiculteur en assurant un suivi de la ruche d'abeilles et en l'alertant en cas de problème survenu pour l'une des causes précédemment citées.

Chapitre 2 : Etude des systèmes de ruche connectée

2.1 Etat de l'art :

L'apiculture étant une fonction indispensable à la protection et l'équilibre des écosystèmes, sa préservation et son développement sont aujourd'hui une priorité non négligeable. Ceci se confirme par le nombre croissant d'entreprises et de projets émergents et spécialisés dans ce domaine.

Pour révolutionner ce dernier, plusieurs solutions sont disponibles, dont l'apport de technologie au sein de la ruche. Ainsi plusieurs systèmes de ruche connectée ont vu le jour et sont vendus tout autour du monde.

Parmis ces systèmes nous citons :

2.1.1 Modèle VILKO :

La société ALYA propose un modèle de ruche connectée avec le produit VILKO, qui contient l'électronique nécessaire pour la mesure du poids de la ruche et d'autres paramètres météorologiques. Les données récupérées seront par la suite envoyées par GSM. [63]

Figure 2.1 système de pondération de la société ALYA.

2.1.2 Modèle BeeOnline :

La société BeeOnline propose plusieurs types de stations de mesure connectées, elles se différencient selon le nombre de capteurs intégrés (poids, température, pluviométrie, humidité, ...etc). Les données collectées seront envoyées par GSM et des alertes sont réceptionnées par SMS en cas de vol ou de chute.[62]

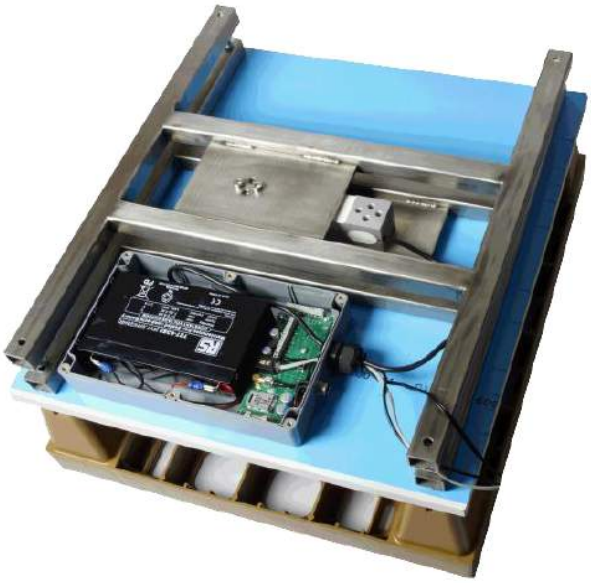

Figure 2.2 Station de mesure connectée de la société BeeOnline.

2.1.3 Modèle Label Abeille :

Le système Label Abeille se place sous n'importe quelle ruche afin d'en récolter les informations, grâce aux capteurs environnementaux qu'il contient : Orientation, luminosité, humidité, température, poids, géolocalisation et pression atmosphérique. [14]

Figure 2.3 Ruche connectée Label Abeille.

2.1.4 Modèle CAPAZ GSM200 :

Le système GSM200 effectue automatiquement, par mesure électronique, des relevés de nombreuses données : le poids de la ruche, la température, la pluviométrie, l'humidité ou en plus la température du couvain. Les données recueillies sont transmises quotidiennement par téléphone portable en SMS ou par E-Mail ou sur serveur CAPAZ Direct. [61]
La télémétrie est par technologie SMS, même un signal faible et suffisant pour émettre un SMS. [61]

Ce modèle est composé de deux parties, la station elle-même et le portable qui peut être détaché de la structure.

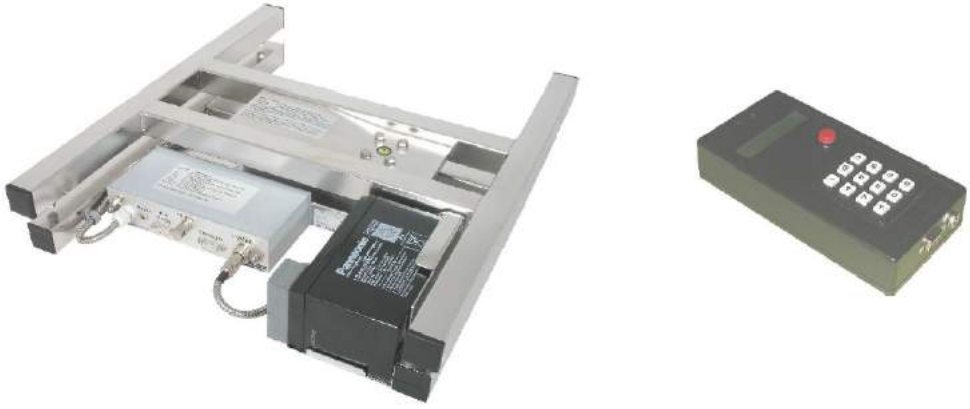

Figure 2.4 Ruche connectée CAPAZ GSM200.

2.1.5 Modèle BeeGuard :

La ruche connectée consistera en une enceinte en bois équipée de capteurs pour mesurer des paramètres internes et externes à la ruche. A l'intérieur de la ruche des capteurs de poids, température et humidité du système commercialisé par BeeGuard renseigneront sur la condition générale de la colonie. De nouveaux capteurs acoustiques et d'activité (compteur d'abeilles à l'entrée de la ruche) compléteront ces informations sur l'état des colonies. Des capteurs de poussière et de gaz (propane, hydrogène, charbon, dioxyde de carbone, benzène et alcool) indiqueront les taux de pollution de l'air dans la ruche. Enfin, une caméra thermique permettra de visualiser la taille de l'essaim et la distribution des abeilles dans la ruche. Un système GPS intégré dans les capteurs BeeGuard permettra également de connaître la position de la ruche et d'identifier tout changement en cas de déplacement (ex : transhumance). A l'extérieur de la ruche, les capteurs de pluviométrie, température, humidité du système BeeGuard augmentés par de nouveaux capteurs de vent et luminosité renseigneront sur les conditions microclimatiques. Des capteurs de poussière et de gaz indiqueront les taux de pollution de l'air. [60]

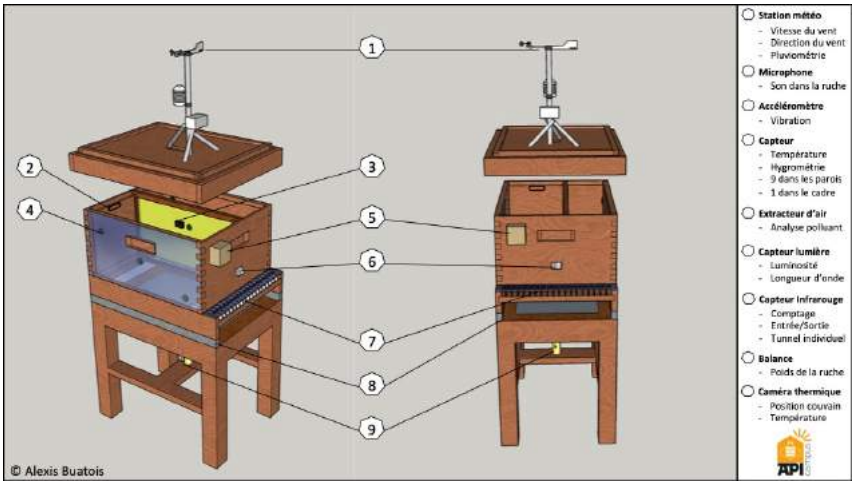

Figure 2.5 Ruche connectée BeeGuard.

2.2 Présentation du système :

Le concept d'une ruche connectée repose sur l'utilisation d'un équipement électronique composé de plusieurs capteurs et d'un calculateur pour remonter ces données, les enregistrer et les analyser ultérieurement. On peut ainsi assurer le suivi de la vie d'une ruche de manière indirecte (car il faut venir récupérer les données enregistrées) ou bien en temps réel si ces données sont transmises vers un serveur qui traitera l'information et prendra des décisions en conséquence.

Ce suivi consiste à vérifier continuellement l'état de la ruche y compris son poids (à l'aide de jauges de poids), la circulation des abeilles de et vers la ruche (entrées/sorties) ainsi que le comptage de leur nombre dans cette dernière. On peut grâce à cela détecter des dégradations possibles par des intrus.

A ce système s'ajoutent aussi des capteurs de mesure de température, d'humidité et de pression au niveau de la ruche, ainsi que de capteur de qualité d'air (taux de CO2) pour récolter une multitude de données et estimer différents paramètres concernant la ruche.

Ces données seront traitées par un calculateur type mbed NXP LPC1768, ensuite stockées dans un serveur implémenté sur une Raspberry PI 3 où elle. Ces données pourront soit être exploitées directement pour produire des alertes ou des solutions auprès de l'apiculteur (alarme intrus, ruche pleine de miel, disparition importante d'abeilles ...etc), ou bien, elles serviront à des chercheurs biologistes pour l'étude de la biodiversité.

Une autre solution est d'envoyer ces données sous forme d'un suivi quotidien à l'apiculteur via un réseau WIFI en utilisant un module de communication WIFI (ESP8288_01s) qui permettra l'échange avec un broker IOT via un protocole MQTT. Le client (l'apiculteur) n'aura plus qu'à se connecter à la plateforme réalisée pour consulter ses données, ceci permettra aussi de délivrer des notifications d'alerte dans le cas d'un problème détecté dans la ruche.

Le système doit être énergétiquement indépendant, ce qui exige une estimation de la consommation des composants et un établissement du bilan énergétique pour déterminer le dimensionnement adéquat de panneaux solaires pour la recharge de la batterie d'alimentation.

Figure 2.6 Illustration du système de la ruche connectée.

Notre système, comme précédemment expliqué, consiste en ce qui suit :

- ➢ Acquisition des données au niveau des capteurs et réception de ces dernières par le MBED LPC1768 selon différents protocoles.
- ➢ Envoi de ces données du MBED LPC1768 à l'ESP8266 01s selon les commandes reçues et signalisation des alarmes du système par une communication série.
- ➢ Transfert des données de l'ESP8266 01s vers le Broker Raspberry Pi par un protocole MQTT.
- ➢ Souscrire l'outil Node-Red en tant que client MQTT et stockage des données sur une base de données InfluxDB.
- ➢ Visualisation de ces données en temps réel à travers le dashboard de l'apiculteur conçu grâce au logiciel Grafana.

Cette chaîne peut être résumée par la figure suivante :

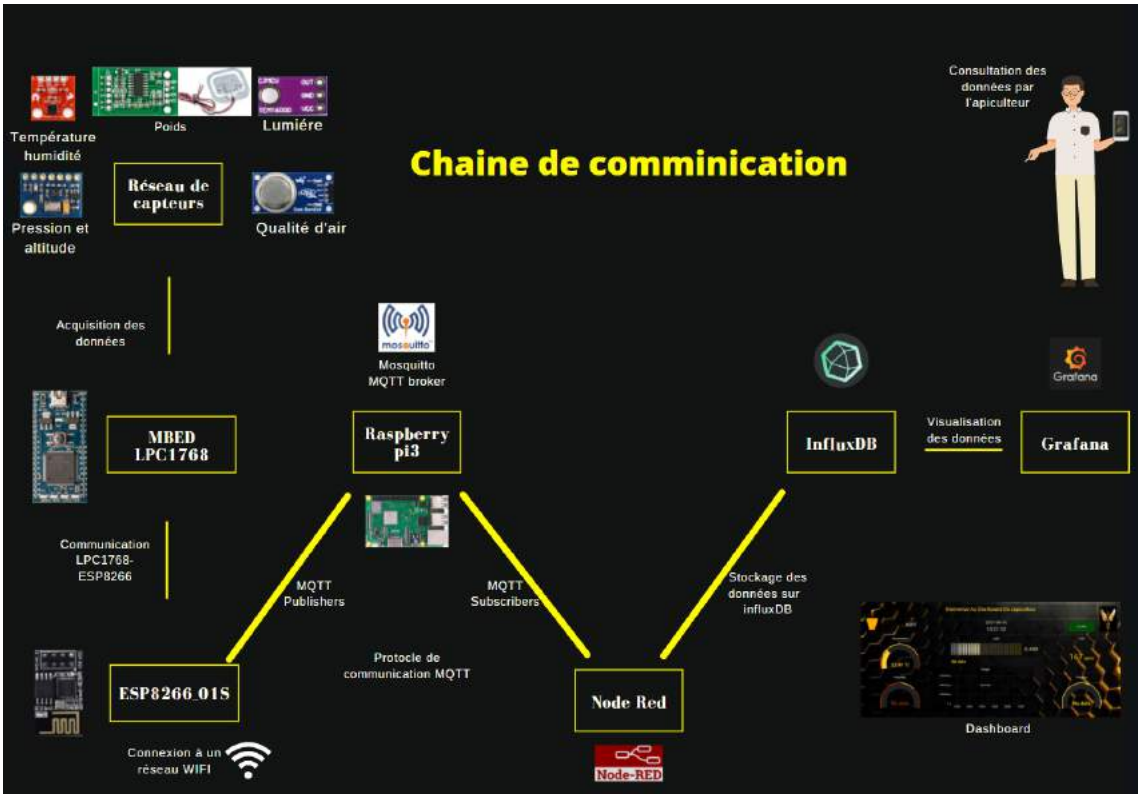

Figure 2.7 Chaîne récapitulative du système de la ruche connectée.

2.3 Cahier des charges :

2.3.1 Objectifs :

Le but de ce projet se résume en un cahier des charges suivant :

1. Concevoir un système permettant d'assurer un suivi quotidien de la ruche selon différents paramètres préalablement définis.
2. Assurer le traitement et le stockage approprié des données (dans un serveur) à des fins d'étude et/ou de recherche.

3. Assurer un système d'alerte permettant de notifier l'apiculteur en cas de problèmes au niveau de la ruche.
4. Assurer une autonomie totale en énergie du système réalisé afin de pouvoir faciliter son installation dans des lieux ne disposant pas d'alimentation électrique conventionnelle (exploitation de l'énergie solaire).

2.3.2 Tâches à réaliser :

Tâche 1 : **"Conception du réseau de capteurs"**

- ➢ Ceci implique dans un premier temps un choix approprié des capteurs à utiliser selon deux critères importants : "les contraintes relatives à la ruche" (conception, fonctionnement) et "les contraintes énergétiques" (optimisation de la consommation).
- ➢ Ces capteurs consistent en : un capteur de poids, un capteur de température et d'humidité, un capteur de pression, un capteur de lumière et un capteur de qualité d'air.
- ➢ Étude de la disposition de ces capteurs pour que la meilleure acquisition soit privilégiée et qu'ils ne soient pas invasifs pour les abeilles.
- ➢ Programmation du système pour l'acquisition et la collecte de ces différentes données.

Tâche 2 : **"Collecte et stockage des données"**

- ➢ Choix de la durée d'acquisition ainsi que du protocole de communication pour assurer la collecte des données.
- ➢ Conception d'un serveur pour le stockage et la gestion des données :
- ➢ Ce serveur sera conçu à l'aide d'une carte électronique grand public Raspberry pi 3.
- ➢ A la carte de programmation mbed LPC1768 sera ajouté un module wifi ESP8266 pour assurer le transfert des données issues des différents capteurs.

Tâche 3 : **"Traitement des données"**

- ➢ Le traitement vise à effectuer une mise en forme des données stockées pour pouvoir, dans un premier temps, **"mieux repérer les cas d'alertes"** à signaler à l'apiculteur, et **"structurer le suivi quotidien"** en fonction d'algorithmes pouvant êtres complexes, car multiparamétriques.

Tâche 4 : **"Transfert des données vers l'apiculteur"**

Les données collectées et traitées seront communiquées à l'apiculteur moyennant l'usage d'un protocole MQTT [40] comme suit :

- ➢ Les données seront transférées de la carte mbed NXP LPC1768 vers la carte ESP8266 01s à travers leurs ports séries avec un baud de 9600.
- ➢ La carte ESP8266 01s est implémentée en tant qu'un client Publisher du protocole MQTT.
- ➢ Un broker Mosquitto [42] est aussi implémenté sur la carte Raspberry pi 3 et recevra les données publiées par la carte ESP8266 01s.
- ➢ Le broker [41] enverra les données nécessaires aux différents clients Subscribers, dans notre cas l'apiculteur.
- ➢ L'apiculteur consultera les données stockées dans une base de données sur un dashboard réalisé avec l'outil Grafana [57]

Tâche 5 : **“Alimentation du système”.**

- ➢ Utilisation des panneaux solaires : assurer l’orientation, la charge et la consommation nécessaire à partir des panneaux solaires.
- ➢ Alternation de la consommation énergétique entre batterie et panneaux, des modules intelligents vendus dans le commerce sont capables de fournir une énergie régulée pour l’électronique. Il faudra toutefois s’assurer que la production et la consommation sont suffisamment équilibrées pour assurer la charge de la batterie et son maintien opérationnel en cas d'impossibilité de charger par les panneaux solaires. .

Tâche 6 : **“Test et suivi du système pour assurer son bon fonctionnement”.**

- ➢ Mise en fonctionnement du système quotidiennement à raison de 8h par jour pour assurer sa fiabilité et qu’aucune défaillance ne vient le perturber.
- ➢ Mise en place d’une série de tests pour examiner le bon fonctionnement des capteurs et la bonne réception des données au niveau de l’apiculteur.

Le diagramme de Gantt ci-dessous décrit la planification de ces tâches tout au long de la période du projet.

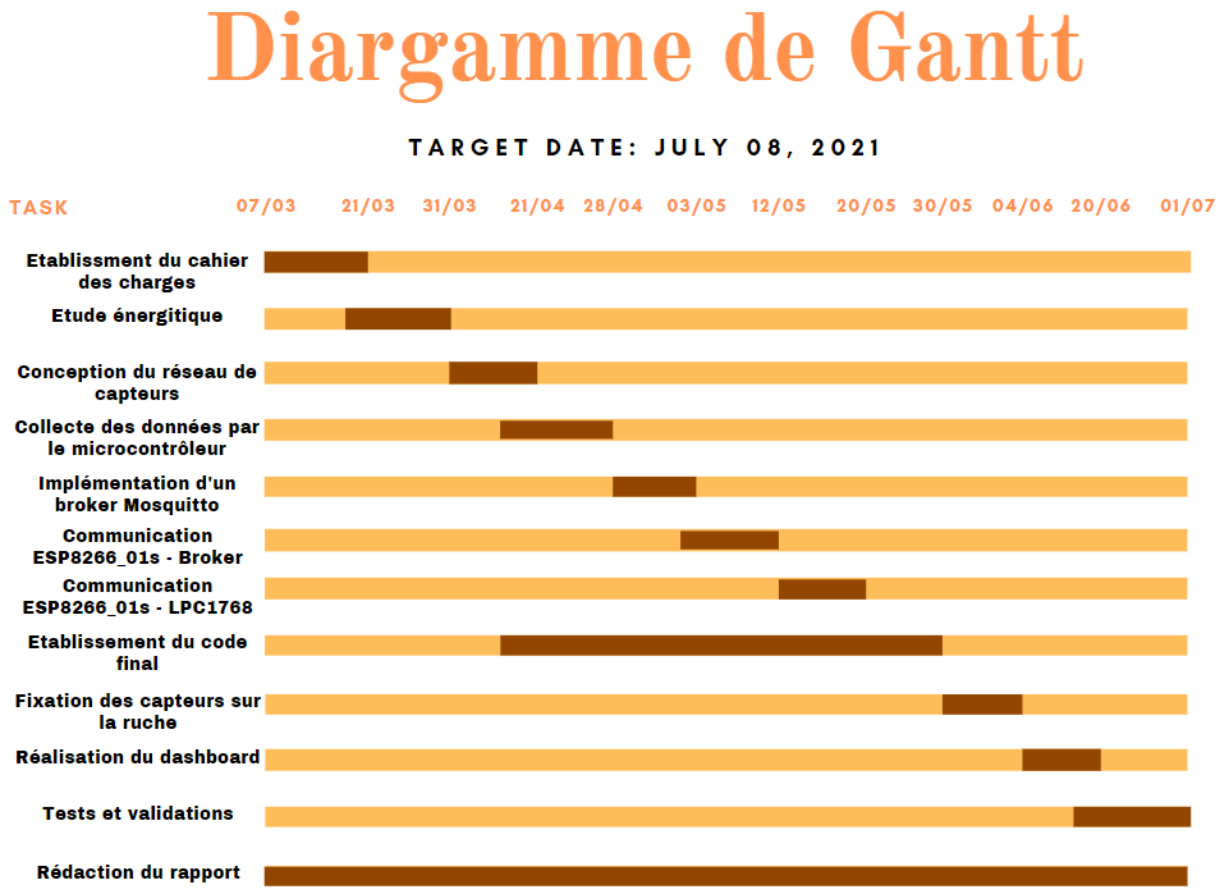

Figure 2.8 Diagramme de gantt.

2.3.3 Contraintes du projet :

- **Contrainte énergétique :** Nécessité d'une étude complète de la consommation énergétique du système selon un choix minutieux des composants pour en assurer une optimisation afin de maintenir la ruche connectée et en fonctionnement dans les périodes de conditions climatiques les plus défavorables.
- **Contrainte environnementale :** Il faut s'assurer que le système ne risque pas d'affecter l'environnement des abeilles ainsi que leur mode de vie (non intrusif).
- **Contrainte de disposition :** Il est important de faire une étude pour fixer les lieux de positionnement des capteurs au niveau de la ruche pour une fiabilité et validité des données récoltées.
- **Contrainte de réseau :** Il est primordial d'assurer la disponibilité du réseau au niveau des ruchers pour pouvoir communiquer avec l'apiculteur, soit de la disponibilité d'un réseau WIFI environnant la ruche.

2.4 Système de ruche connectée :

Ci-dessous une illustration qui permet de schématiser le système de ruche connectée prévue (cette illustration provient d'un projet développé et commercialisé par BeeGuard) [60]:

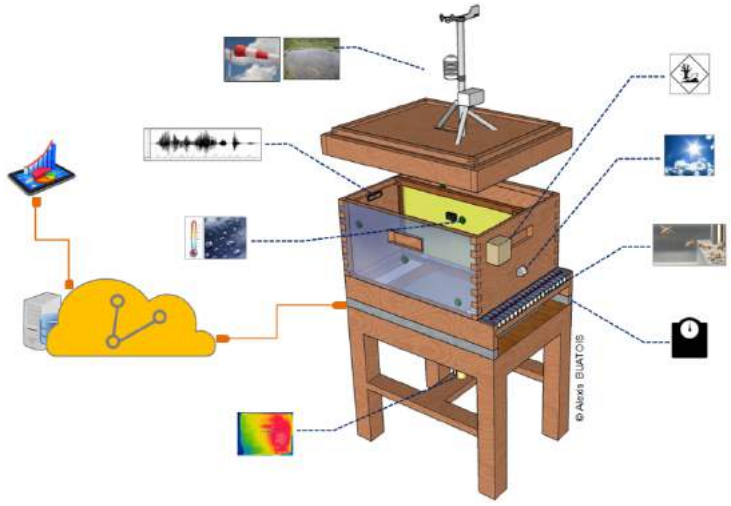

Figure 2.9 Illustration du système de la ruche connectée.

Chaque capteur du système a une fonction précise :

1. **La masse :** témoigne de la santé de la colonie et de l'état de la production :

➢ Détecter le début et la fin d'une miellée.
➢ Mesurer en hiver, le niveau de consommation des réserves.
➢ Signaler la possibilité d'un essaimage suite à une baisse brutale du poids de la ruche (entre 2 et 4 kilos en une heure).
➢ Détecter la destruction ou renversement d'une ruche (à cause du vent ou du vandalisme par exemple).

2. **L'humidité et la température :** signalent, par exemple, la nécessité d'abreuvage des abeilles, d'aérations de la ruche en cas de fortes chaleurs où toutes variations non régulières de ces paramètres auxquels il faudrait y remédier. L'impact de cela sur la ruche permettra à l'apiculteur d'anticiper la situation pour les années suivantes.

3. **La pression atmosphérique :** prévient d'un changement météorologique susceptible de provoquer un changement de comportement et un rassemblement de la colonie. Une baisse de la pression atmosphérique traduit une dépression donc un mauvais temps en perspective.

4. **La luminosité :** permet de faire le lien entre le rendement de la ruche et son influence sur la période de pollinisation des abeilles au cours de la journée.

5. **Le taux de CO2 :** permet d'inspecter la qualité d'air de la région avoisinant la ruche et d'avoir un compte rendu global du taux de CO2. L'augmentation de celui-ci pourrait traduire un malaise dans la ruche, intrusion d'un animal par exemple.

2.5 Choix du matériel :

Le choix du matériel s'est fait selon des critères à la fois en rapport avec les besoins du projet ainsi que la consommation énergétique à optimiser.

1. Le premier critère de choix des capteurs est de correspondre aux variations de paramètres à mesurer dans la ruche.
2. Tous les capteurs sélectionnés utilisent des protocoles de communication simples et peu consommateurs, entre autres le protocole I2C.
3. La carte de programmation et le module ESP8266 sont les plus énergivores. Pour cela, nous avons préconisé des méthodes d'optimisation en vue de réduire leur consommation.

Composant :	Critères de choix :	Référence choisie :	Caractéristiques :
Capteur de poids	Le capteur de poids nécessite une très bonne précision comme il permettra de suivre différents paramètres de la ruche.	Jauges de contraintes + module HX711	Interface série customisée. Précision : 24 bits.
Capteur température et humidité	La température au sein de la ruche doit être maintenue aux environs de 33 à 36°C, de ce fait le capteur doit être très précis et détecter toute variation. L'humidité aussi doit être maintenue entre 50 et 70%, il faut donc un capteur à la fois précis et avec une grande étendue.	HTU21-D	Précision en température : 0.3°C Précision en humidité 2%. Étendue de mesure humidité : 0-100%. Protocole : I2C.
Capteur de Lumière	L'activité au sein de la ruche ne se produit que durant la journée.	TEMT6000	Signal analogique.
Capteur de pression	Le rucher peut se trouver à des altitudes allant jusqu'à 2 Km.	MS5611-01BA03	Plage de mesure : 10 à 1200 hPa (jusqu'à une altitude de 10 Km) Protocole : I2C

Capteur de mesure de la qualité d'air	/	CCS811	Protocole : I2C Étendue de mesure de 400 ppm à 29206 ppm.
Carte microcontrôleur de gestion	Le critère principal dans ce choix est la consommation énergétique de la carte choisie, il est nécessaire de trouver un compromis entre cette dernière et la puissance de calcul.	LPC1768	Cette carte étant un premier choix, il reste néanmoins nécessaire de procéder à une optimisation pour minimiser l'énergie totale du système (un exemple serait de couper la communication ethernet qui permettrait de baisser la consommation d'environ 40 mA).
Panneau solaire	Le panneau solaire doit pouvoir fournir l'énergie nécessaire au système.	/	Dimensionnement choisi suite au bilan énergétique.
Batterie	La capacité de la batterie doit aussi correspondre aux besoins du système.	/	Batterie 12V.
Raspberry	Dans une première partie, nous avons opté pour la conception d'un serveur pour le stockage et la gestion des données récoltées par le réseau de capteurs.	Raspberry Pi 3 model B+	
Module wifi	Nécessaire pour communiquer les données de la carte mbed vers la Raspberry à travers le réseau wifi.	ESP8266 ESP-01S wifi	Simple et pratique à utiliser.

Table 2.1 Références du matériel choisi

2.5.1 Estimation budgétaire du système :

Composant :		Prix (DA):
Système de mesure de poids	Jauges de contrainte	2400
	Module HX711	1200
	Système mécanique	850
Capteur de température et humidité HTU21D		800
Capteur de pression MS5611		900
Capteur de lumière TEMT6000		900
Capteur de qualité d'air CCS811		1800
Microcontrôleur LPC1768		14500
Module WiFi ESP01s		900
Raspberry PI 3 model B+		14500
Carte SD		1600
Pin holders et Jumpers		1000
Régulateurs 3,3V et 5V		120
Ruche de test à 8 cadrans		3600
Prix total :		**45070**

Table 2.2 Estimation du budget nécessaire

- ➔ Les cartes microcontrôleur LPC1768 et Raspberry PI ainsi que le générateur ont été fournis par l'école.
- ➔ Le prix total du système s'élève aux environs de 45000 DA.

2.6 Bilan énergétique :

2.6.1 Optimisation de la carte mbed à l'aide de la libraire "PowerControl" :

Avant d'établir un bilan complet de notre système, nous avons pensé à effectuer certaines optimisations pour réduire la consommation de courant de ce dernier.
Cette optimisation repose sur le fait de désactiver l'interface ethernet de la carte LPC1768, pour cela nous avons eu recours à la librairie "EthernetPowerControl.h" qui permet de couper cette

dernière via une simple fonction en changeant l'état de certains registres, cette fonction est "PHY_PowerDown()".

1) En mode normal, la carte MBED LPC1768 implémentée avec le code destiné à notre application utilise un courant d'environ 170 mA sous une alimentation de 5V, comme montré dans la figure :

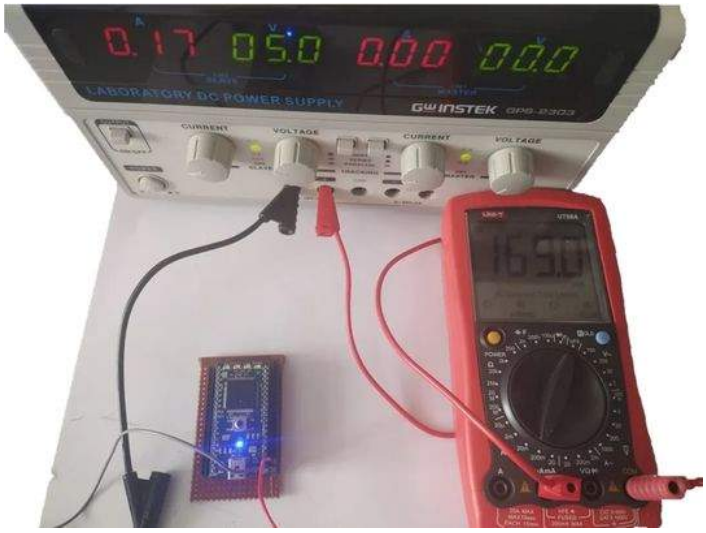

Figure 2.10 Mesure du courant consommé par la carte MBED LPC1768 en mode normal.

PS : Pour effectuer cette opération de mesure, il a suffit d'alimenter la carte MBED LPC1768 sous une tension de 5V et de la mettre en série avec un ampèremètre pour la mesure du courant.

2) En utilisant la librairie "PHY_PowerDown" citée pour optimiser cette consommation et en désactivant l'interface ethernet, le courant consommé par la carte MBED LPC1768 passe aux environs de 135 mA toujours sous une tension de 5V.

Figure 2.11 Mesure du courant consommé par la carte MBED LPC1768 après optimisation.

A partir des résultats obtenus, il est possible donc d'optimiser une consommation en courant de 35 mA sur la consommation totale de notre système, ce qui affecte énormément la partie énergétique et permet de gagner en longévité.

2.6.2 Bilan du système :

- Pour la conception du système autonome et énergétiquement indépendant, il est primordial d'établir un bilan énergétique optimisé dans sa globalité.
- Pour ce faire, nous avons estimé la consommation énergétique de chaque composant :

- La consommation des capteurs fait entrer en compte leur consommation en mode actif et standby ainsi que celle de transmission, pour cela il a été nécessaire de déduire les courants de ces deux modes, à partir de données des datasheets, du nombre de bits des données mais aussi de la fréquence de transmission utilisée.
- Pour la carte et le module wifi, nous avons pu trouver des valeurs globales de la consommation qu'ils peuvent atteindre.

Bilan énergétique du système :
Les données collectées des datasheets nous ont permis d'établir ce qui suit :

1) Courants d'acquisition moyens de certains capteurs :

Composant :	HTU21D	MS5611	HX711
Courant d'acquisition :	0,9 µA	0,9 µA	1,5 mA

Table 2.3 Courants d'acquisition moyens

2) Courants de Transmission de certains capteurs :

	Courant en mode actif Ia	Courant en mode standby Istb	Période T(s)	Nombre de bits de la donnée	Fréquence de transmission (Hz)	Temps de transmission Ttrans (s)	Courant de transmission
HTU21D	0,5 mA	0,14 µA	0,016	12	400000	0,00003	1,1 µA
MS5611	1,26 mA	0,14 µA	1	24	400000	0,00006	0,22 µA
HX711	1,4 mA	0,3 mA	0,4	24	400000	0,00006	0,51 µA

Table 2.4 Courants de transmissions

Tel que :

$$Temps\ de\ transmission\ =\ Nombre\ de\ bits\ de\ la\ donnée/Fréquence\ de\ transmission\ (Hz)$$

$$Courant\ de\ transmission\ =\ [(Ia * Ttrans)\ + Istb * (T - Ttrans)]/T$$

3) La consommation totale de chaque composant :

- La consommation totale des capteurs est donnée par une somme des deux consommations précédemment établies.
- La carte MBED LPC1768 consomme suite à son optimisation 0,14 A en tout.
- L'ESP8266 01s consomme une valeur moyenne de 80 mA soit 0,08 A.

	HTU21D	MS5611	HX711	TEMT6000	CCS811	LPC1768	ESP8266 01s
Courant total :	2 µA	1,12 µA	1,51 mA	50 µA	0,025 mA	0,14 A	0,08 A

Table 2.5 Consommation totale de chaque composant

➔ **La consommation totale en courant du système est de 0,25 A**

➔ **La capacité de la batterie nécessaire au système est de l'ordre de 6 AH par jour.**

Composant	Réference	Courant (µA) d'acquisition	Période (s)	Courants du datasheet En Activité (mA)	En Standby (µA)	Bits data	Fréquence Com KHz	T (ms) transmission	Courant (µA) de transmission	Energie AH	Energie par jour (AH)
Temp+Hum	HTU21D	0,9	0,016	0,5	0,14	12	400	0,03	1,08	0,000002	0,00005
Pression	MS5611	0,9	1	1,26	0,14	24	400	0,06	0,22	0,000001	0,000027
Poids +	Jauges										
Module	HX711	1500	0,4	1,4	0,3	24	400	0,06	0,51	0,0015	0,036
Lumière	TEMT6000	/	/		/	/	/	/	/	0,00005	0,0012
module CO2	CCS811	/	/	/	/	/	/	/	/	0,0255	0,612
Carte mbed	LCP1768	/	/	/	/	/	/	/	/	0,14	3,36
Module wifi	ESP-01S	/	/	/	/	/	/	/	/	0,08	1,92
										Total AH	Total Jour
										0,25	5,93

Figure 2.12 Capture d'écran sur le fichier excel de calcul du bilan énergétique.

Nous avons effectué une mesure suite à l'installation de notre système, en disposant d'un ampèremètre en série, ce qui a donné une consommation de 0,25A conforme avec notre bilan comme montré ci-dessous :

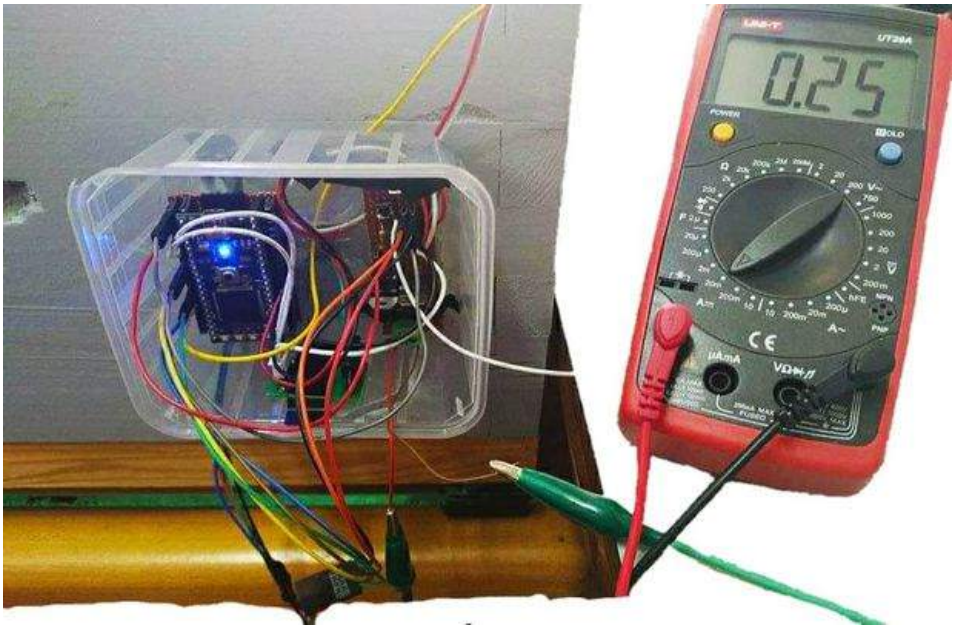

Figure 2.13 Mesure de la consommation.

Il faut noter que le travail réalisé est un POC (Proof Of Concept) qui a nécessité de s'équiper avec des microcontrôleurs disponibles. Il est clair que lors de la mise en place du prototype sur une vraie ruche, il faudra refaire l'étude sur le choix du microcontrôleur pour sélectionner un processeur moins gourmand et donc moins rapide en puissance de calcul. En effet, la fréquence de fonctionnement du processeur (dans notre cas 100Mhz) impacte directement sur la consommation vu que cette dernière est directement proportionnelle à la fréquence et au carré de la tension d'alimentation.
Donc le passage à une technologie 3,3V et une fréquence de 50MHz par exemple va nous permettre d'économiser pour passer de 135mA à 29,4mA (soit plus de 4,5 fois plus performant en longévité pour une même batterie).

2.6.3 Conclusions :

- ➢ La consommation de notre système atteint un maximum de 6AH par jour.
- ➢ La consommation des capteurs est pratiquement négligeable comparée au total qui est consommé par les microcontrôleurs. Les capteurs consomment moins de 1% du courant total du système.
- ➢ Autres que les capteurs, les éléments les plus gourmands en énergie sont la carte microcontrôleur ainsi que le module wifi.
- ➢ Suite au bilan effectué, nous pouvons estimer les dimensions du panneau solaire nécessaire ainsi que la capacité de la batterie.

2.6.4 Batterie et panneau solaire :

1. Batterie :

Il faut savoir qu'il n'est possible de consommer que 70% d'une batterie au maximum pour que cette dernière puisse durer, de ce fait ces 70% doivent correspondre à la consommation totale de notre système qui est de 6 AH.

➔ Par une règle de trois, la batterie qu'il nous faut est une batterie de 12 V, 8.6 AH

2. Panneau solaire :

Méthode 1 : **"Estimation des dimensions du panneau solaire"**

La puissance journalière de notre système est de 8,6 AH * 12V = 104 WH.
Pour 104 Wh/jour et un rayonnement solaire global de 8 kWh/(m²jour) en décembre (le mois où le rayonnement solaire atteint le plus bas taux), on calcule la surface du panneau comme suit :

$$S = \frac{1}{\eta} \frac{E_{demande}}{E_{disponible}}$$

Avec η le rendement de la technologie choisie (rendement minimal de 6% pour une technologie "silicium amorphe")

➔ On obtient ainsi un dimensionnement de 0.22 m².

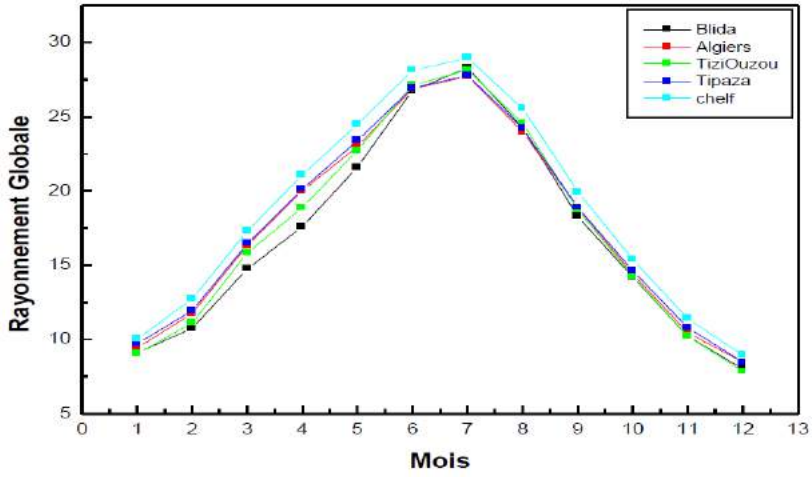

Figure 2.14 Rayonnement solaire global des régions du nord de l'Algérie.

PS : Le rendement d'un panneau est fonction de ses composants, comptez environ :

- ➢ 6% pour le silicium amorphe.
- ➢ 15% pour le silicium polycristallin.
- ➢ 16 à 24% pour le silicium monocristallin.

Méthode 2 : **“Estimation de la puissance nécessaire du panneau”**

Notre système a les caractéristiques suivantes :
0,25 A : Sa consommation totale de courant.
8,6 AH : La capacité de la batterie nécessaire de sorte que 6 AH (ce qu’il faut au système) représente 70% du total.
4 H : Le taux d’ensoleillement le plus critique, disponible au mois de décembre en Algérie.
Ce qui donne un besoin de : 0,25 + 8,6/4 = 2,4 A.

➔ Le panneau doit fournir un courant total de 2,4 A, à 3,3V sa puissance doit être au minimum 8 W.

PS :

- L’installation du panneau nécessite de prendre en compte son orientation qui sera importante pour optimiser la conversion d’énergie solaire en électricité.
- Cette partie n’a malheureusement pas été effectuée en pratique en raison de l'indisponibilité des panneaux solaires, la partie alimentation a donc simplement été réalisée en utilisant une batterie rechargeable.

Chapitre 3 : Conception du système de la ruche connectée

3.1 Réseau de capteurs :

Le réseau de capteurs doit répondre au cahier des charges fixé et assurer le suivi des différents paramètres essentiels à l'étude de l'état de la ruche entre autres : le poids, la température, l'humidité, la pression, la lumière et le taux de CO2. Ce réseau sera commandé par un microcontrôleur MBED LPC1768 qui sera connecté aux divers capteurs choisis comme le montre le schéma suivant :

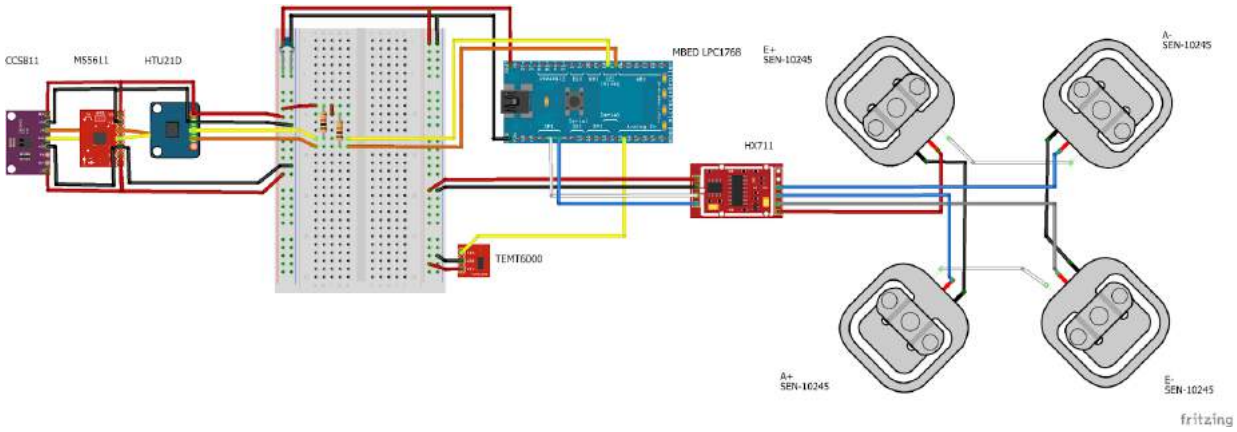

Figure 3.1 Circuit de branchement du réseau de capteurs.

Le choix des capteurs adéquats à notre système de ruche connectée a fait l'objet d'une étude rigoureuse, alliant des contraintes énergétiques et des contraintes liés à la ruche, sur ce qui suit une description détaillée de chaque capteur utilisé.

3.2 Capteur de Poids "Jauges de contraintes et acquisitions" :

3.2.1 Présentation du capteur :

Ce capteur est utilisé sous forme d'un système de balance composé de deux éléments importants :

- ➔ Un pont de wheatstone constitué de 4 cellules de charge (load cells) ayant une étendue de mesure qui peut atteindre jusqu'à 50 Kg par cellule, soit 4 x 50 = 200 Kg sur la totalité du système.
- ➔ Un module HX711 pour amplifier le signal à la sortie des jauges qui est très faible et effectuer une conversion analogique numérique avant l'acquisition de la donnée. Les données sont numérisées et transmises par un protocole série synchrone.

Figure 3.2 Capteur de masse composé des 4 jauges de contraintes de 50 Kg chacune et du module HX711.

3.2.2 Principe de fonctionnement et utilisation des cellules de charge :

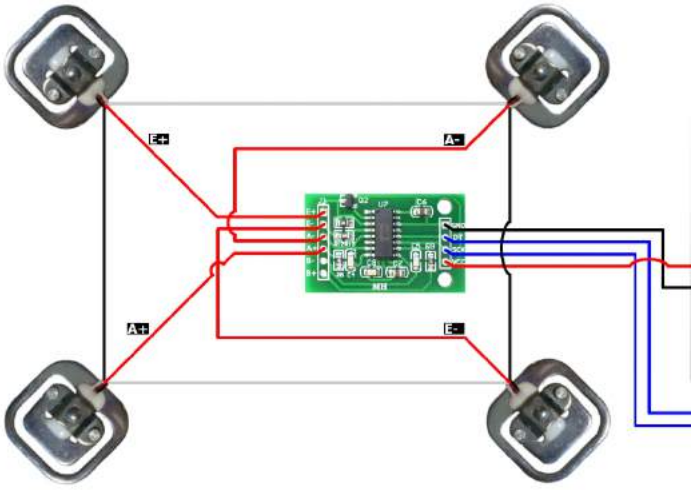

Figure 3.3 Cellules de charge sous forme d'un pont de wheatstone.

- Comme le montre la figure ci-dessus, les cellules de charge qui ne sont autres que des résistances variables, seront mises sous une configuration de pont de wheatstone alimentée par une tension d'excitation entre les bornes E+ et E-. [22,23]
- Le signal de sortie qui se trouve entre les bornes A+ et A- est dû à un déséquilibre dans le pont causé par une variation dans les valeurs des résistances. [22,23]
- Ces valeurs diffèrent selon l'intensité de la compression ou de la détente subie qui dépend essentiellement du poids ajouté sur la balance. [22,23]
- Tout ceci entraîne en sortie une très faible tension qui représente la donnée de poids. En cas d'équilibre du pont, cette sortie est nulle. [22,23]

- Configuration d'équilibre du pont : [22,23]

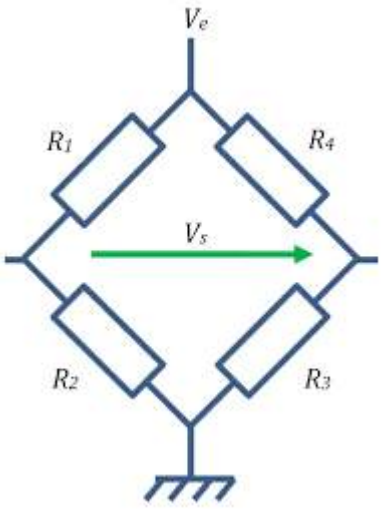

Figure 3.4 Représentation du pont de wheatstone.

Par analogie avec la figure précédente du pont de wheatstone et selon la configuration de nos cellules de charge, les résistances R1, R2, R3 et R4 de notre système se valent à l'équilibre et ont une valeur de 2 KOhm chacune. [22,23]

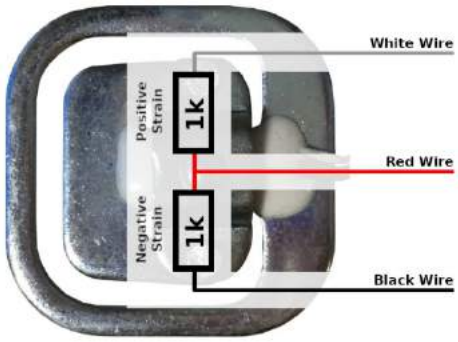

Figure 3.5 Configuration des résistances au niveau des cellules de charge.

3.2.3 Précision de la mesure :

- La précision de chaque jauge représente ±0,05% de sa pleine échelle (50 Kg), ce qui atteint ±0,025 Kg soit 25 grammes par jauge.[22]
- Le système composé de 4 jauges aura une précision de 100 grammes sur ses mesures. Cela représente un peu plus que le poids de 1200 abeilles. Pour une ruche de 40000 Abeilles, nous avons alors une sensibilité d'estimation du nombre d'abeilles à 3% près.
- Cette précision correspond bien à la précision nécessaire pour l'évaluation des variations de poids dans notre système de ruche connectée.

3.2.4 Module HX711 :

- Le module HX711 est composé d'un amplificateur et d'un convertisseur analogique numérique. Il permet l'amplification d'un signal envoyé par une cellule d'effort ainsi que la conversion analogique en numérique de ce dernier. [24]
- Il est spécifiquement conçu pour les applications de balance. Les cellules de charge qui mesurent généralement le poids fournissent des sorties de tension en millivolts. Ces sorties sont difficiles à gérer directement par les contrôleurs, nous utilisons donc le HX711 qui prend ces signaux de tension et fournit des valeurs numériques standards qui peuvent être utilisées par un microcontrôleur. La puce a un préamplificateur intégré spécifiquement pour gérer ces basses tensions. [24]
- Caractéristiques :

➔ Deux canaux d'entrée différentiels.

➔ Amplificateur à gain programmable actif à faible bruit avec gain sélectionnable de 32, 64 et 128.

➔ Convertisseur Analogique/Numérique à haute précision 24 bits.

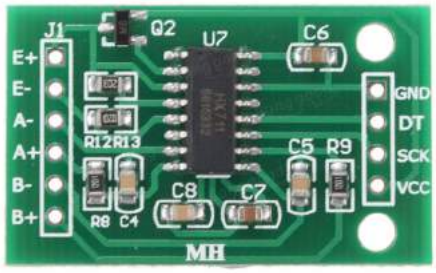

Figure 3.6 Module HX711.

- Les sorties du module DT (ou DOUT) et SCK représentent respectivement le signal de donnée numérique (Digital OUTput) et le signal Serial ClocK.
- L'alimentation de ce module se fait par du 2,6 ~ 5,5 V.
- Le module HX711 nécessite une librairie disponible sur MBED pour sa programmation.

3.2.5 Conditions nécessaires pour l'utilisation du capteur :

➢ Une première condition pour assurer une stabilité des mesures est d'utiliser un système semblable aux balances usuelles, pour cela nous avons préconisé une plaque de verre sur mesure pour la ruche à 8 mm d'épaisseur ainsi que de petits supports 3D retirés d'une ancienne balance (les supports devant être sur mesure, nous avons choisi de récupérer ceux d'une ancienne balance, nous facilitant ainsi la production du système par le principe d'assemblage de composants sur étagère du commerce.) pour une bonne fixation des cellules de charge au niveau de cette dernière. Cette étape est primordiale et permet d'assurer un maximum de stabilité et de garantir de bonnes mesures.

Ci joint une photo des supports utilisés :

Figure 3.7 Supports 3D pour la fixation des cellules de charge sur la plaque de balance.

➢ Une deuxième condition tout aussi importante est d'opérer une bonne calibration du système pour effectuer par la suite des mesures fiables. Cette calibration consiste en 2 étapes :

1. La mise à zéro : (set offset)

Pour cela nous allons d'abord déterminer l'offset ou l'erreur de zéro de notre balance, en effectuant une simple lecture de poids à vide, puis nous procédons à une tare qui permet d'éliminer ce biais sur toute mesure effectuée.

2. Etalonnage : (scaling)

Ceci consiste à comparer les valeurs mesurées par notre capteur à des étalons de poids connus, de ce fait nous pouvons faire correspondre nos mesures aux valeurs exactes pour assurer une exactitude de la mesure.

3.2.6 Système de mesure de poids complet :

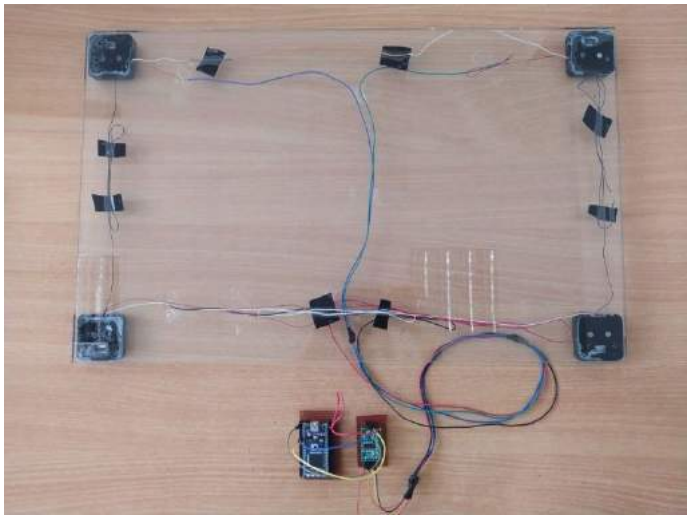

Figure 3.8 système de mesure de poids final.

3.3 Capteur de température et d'humidité "HTU21D" :

L'HTU21-D est une composition intégrée d'un capteur d'humidité et d'un capteur de température.

3.3.1 Principe de fonctionnement du capteur d'humidité :

Le capteur d'humidité repose sur le principe de deux électrodes mises en parallèles qui forment un condensateur de capacité de $C = \varepsilon_0.\varepsilon_r.S/d$. où S est la surface des électrodes, d la distances entre ces dernières, ε_0 constante électrique (permittivité du vide) et ε_r la permittivité relative du milieu. [25]

La permittivité peut varier en fonction de la fréquence et de l'humidité, dans le cas de l'HTU21-D, un isolant hygroscopique, un matériau (polymère) est inséré entre les deux plaques du condensateur dont la permittivité varie en fonction de la quantité d'humidité absorbée ou désorbée induisant un changement de la capacité qui peut être mesuré. [25]

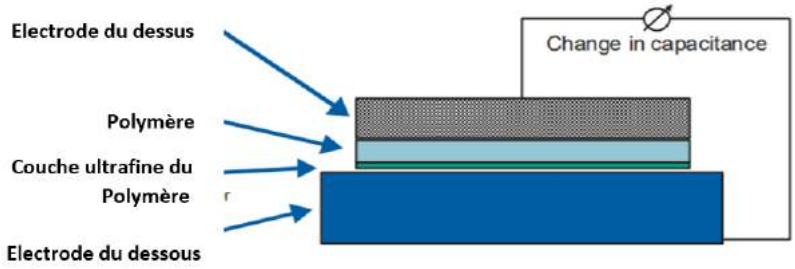

Figure 3.9 assemblage d'un élément de détection d'humidité capacitive.

Les éléments de détection sont assemblés dans un circuit en pont et le signal est enregistré par un amplificateur d'instrumentation. Après la conversion A/N, le signal est individuellement linéarisé, étalonné et compensé en température. Il est alors présent dans un format standard en sortie. [25]

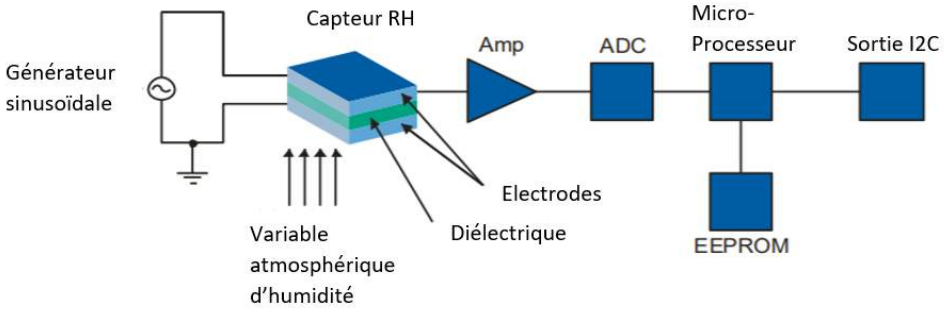

Figure 3.10 schéma électrique du capteur d'humidité HTU21D.

3.3.2 Protocole de communication :

Le protocole I2C est une forme évoluée de la communication série destinée aux accessoires domotiques développée à partir des années 1980 par Philipps. Il nécessite que deux lignes de communication pour connecter les dispositifs [28]:

- la ligne SDA (Serial Data line) : est une ligne de données bidirectionnelle.
- la ligne SCL (Serial Clock line) : ligne de l'horloge de synchronisation bidirectionnelle.

Il est indispensable pour l'établissement d'une communication I2C d'avoir tous les dispositifs connectés à la même masse électrique GND ainsi de connecter des résistances de pull-up entre les lignes SDA / SCL au Vcc. [28]
Il est à noter que chaque dispositif communiquant avec un protocole I2C jouit d'une adresse unique qui prend la forme d'une chaîne hexadécimale. [28]

Ainsi, le capteur HTU21D utilise **un protocole I2C** pour communiquer avec la carte mbed, celle-ci utilise l'adresse 0x40 pour adresser le capteur, et les adresses 0xE3 0xE5 (respectivement hold master et no hold master) pour enclencher la mesure de température et d'humidité. [33]

- La mesure de température est obtenue en utilisant la formule suivante sur le signal de température reçu par l'I2C:

$$Temp = -46.85 + 175.72 * \frac{S_{Temp}}{2^{16}}$$

- La mesure d'humidité est obtenue en utilisant la formule suivante sur le signal d'humidité reçu par l'I2C :

$$RH = -6 + 125 * \frac{S_{RH}}{2^{16}}$$

Plus de détails concernant ce composant sont disponibles en annexe C.

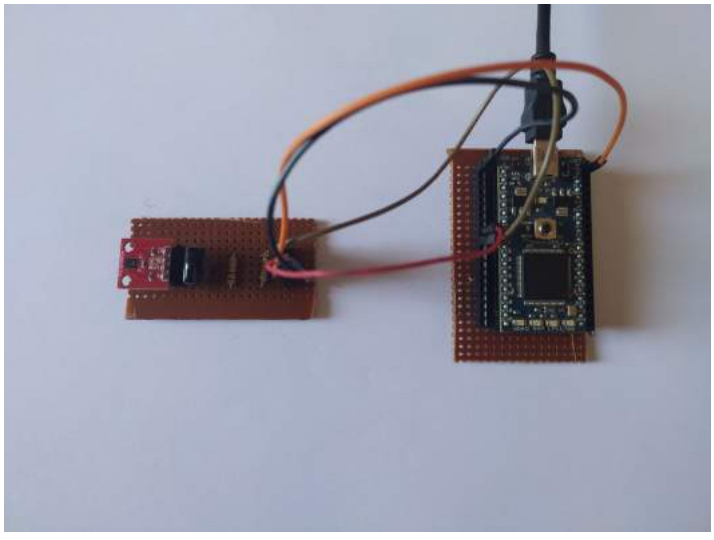

Figure 3.11 Circuit du capteur HTU 21-D.

3.4 Capteur de pression “MS5611” :

Le MS5611 consiste en un capteur piézorésistif avec une interface I2C, sa principale fonction est de convertir la tension de sortie analogique non compensée du capteur en une valeur numérique de 24-bits, il comprend également en option une sortie de 24-bits pour la mesure de température. [27]

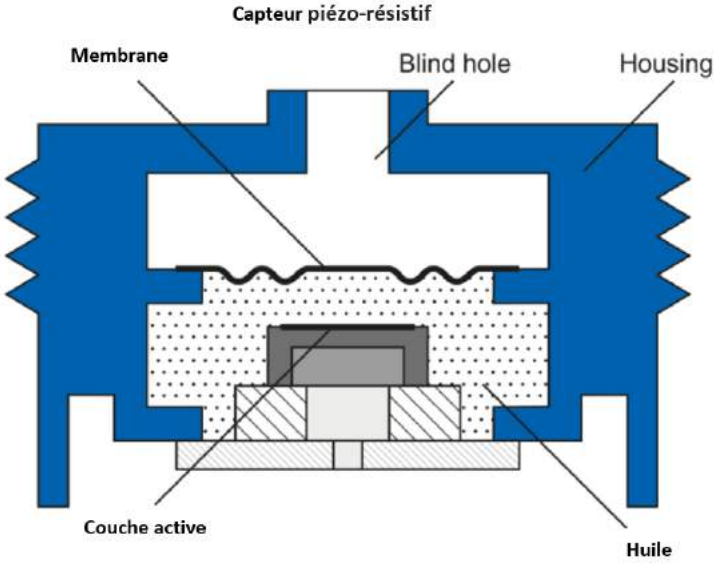

Figure 3.12 technologie du capteur de pression piézorésistif.

3.4.1 Principe :

Dans le cas des matériaux semi-conducteurs, la variation de la résistance spécifique, et donc du signal, découle de la mobilité variable des électrons dans la structure cristalline. Cette mobilité est influencée par la contrainte mécanique. La séparation de la puce de silicium sensible et du milieu du process est assurée par une membrane en acier inoxydable (encapsulage). Pour la transmission interne de la pression, on fait appel à un liquide spécial, de l’huile de paraffine ou de silicone selon l’application. [26]

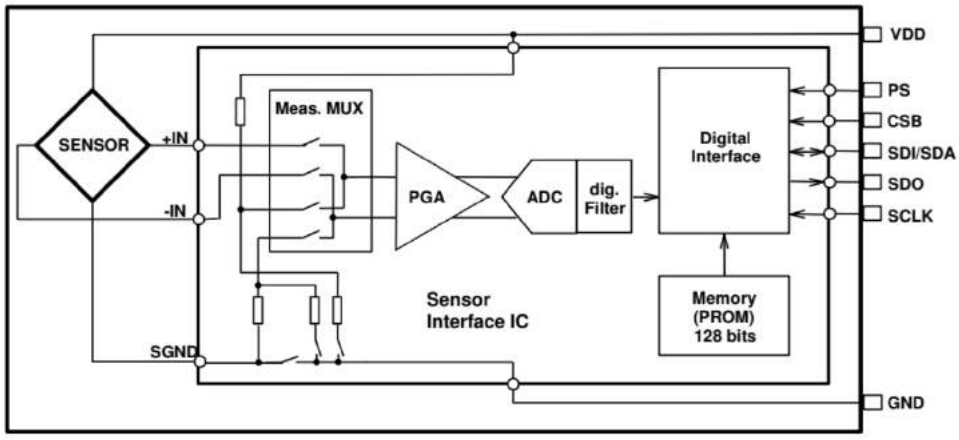

Figure 3.13 schéma électrique du capteur d'humidité MS5611

Tout comme le capteur HTU21D, La communication avec ce composant s'effectue avec le protocole I2C. Plus de détails à propos du composant sont disponibles en annexe D.

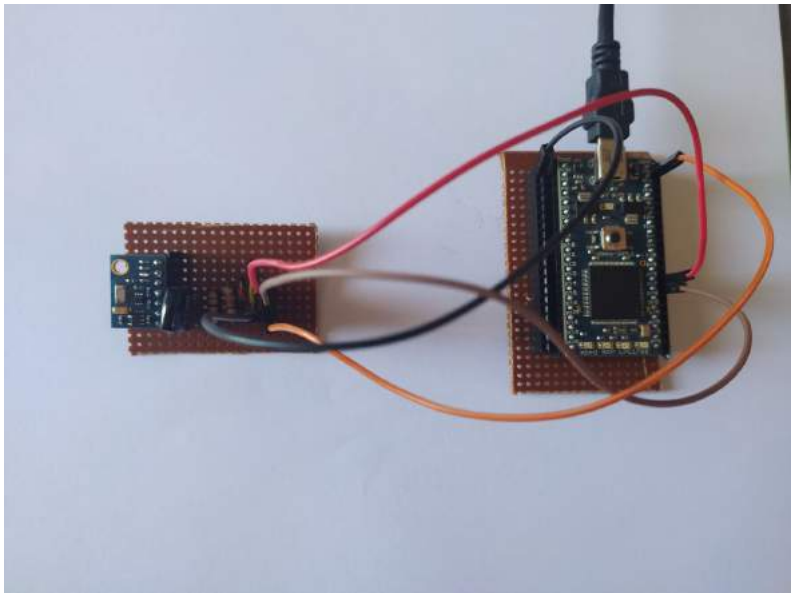

Figure 3.14 Circuit du capteur MS5611.

3.5 Capteur de lumière "TEMT6000" :

Le TEMT6000 est un phototransistor en silicium type NPN intégré dans un circuit imprimé.[34]

3.5.1 Principe de fonctionnement :

Dans le transistor, le courant circulant dans la base est amplifié et le courant du collecteur circule. Dans le phototransistor, la lumière entrante agit comme le courant de base et le courant du collecteur circule en fonction de la quantité de lumière entrante. [30]

Le phototransistor est composé de semi-conducteurs. Lorsque la lumière frappe le matériau, les électrons / trous libres du semi-conducteur provoquent le courant qui circule dans la région de base qui n'est utilisée que pour polariser le transistor. [30]

La lumière entre dans cette région et génère les paires électron-trou. La génération de ces paires se produit principalement dans la polarisation inverse. Le mouvement des électrons sous l'influence du champ électrique provoque le courant dans la région de base. Ce dernier injecte les électrons dans l'émetteur. [30]

Le capteur est relié à la carte mbed via une entrée analogique qui ne nécessite aucune librairie.

3.5.2 Informations complémentaires :

- Alimentation : 3 à 5 Vcc
- Bande passante : 360 à 970 nm.

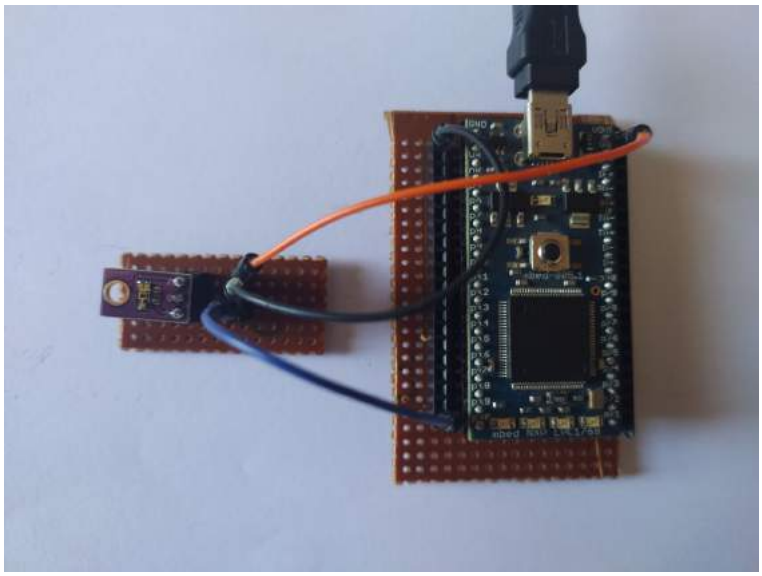

Figure 3.15 Circuit du capteur TEMT6000.

3.6 Capteur de qualité d'air "CCS811" :

3.6.1 Présentation :

Un capteur de qualité d'air permet de mesurer en continu le niveau de CO2 ambiant ainsi que celui de différents autres gaz et d'estimer par la suite le taux de pollution dans l'air environnant. Ce paramètre est très important dans notre cas car il permet d'étudier l'effet de ce critère sur l'environnement et le mode de vie des abeilles.

Le CCS811 permet de détecter une large gamme de Composants Volatiles Organiques Totaux (TVOC) y compris des niveaux équivalents de dioxyde de carbone (eCO2) et d'oxyde métallique (MOX). Les COV sont classés comme polluants et/ou irritants sensoriels et peuvent provenir de diverses sources. [31]

Ainsi, grâce au CCS811, il est possible entre autres de se procurer le taux de CO2 ambiant nécessaire à mesurer dans le cadre de notre projet.

3.6.2 Principe de fonctionnement :

Les performances du CCS811 dépendent des niveaux de sa résistance et de sa sensibilité, ce capteur est composé d'une résistance appelée "sensing resistance Rs", qui est sensible à la variation des différentes concentrations de gaz dans l'air, sa valeur correspond au taux de présence de CO2 (eCO2) donné en ppm (partie par million) et à celui du TVOC (eTVOC) donné en ppb (partie par billion). [31]

PS : 1 ppm correspond à un rapport de 10^{-6} , 1 ppb correspond à 10^{-9} .

Le taux de CO2 normal dans l'air ambiant est de 350 à 400 ppm, soit 0,35 à 0,4 ml/l.
Les différentes étendues de mesures du capteur sont comme suit :

- Le taux de CO2 mesuré va de 400 ppm à 29206 ppm. [36]
- La valeur de TVOC va de 0 à 32768 ppb. [36]

Pour son utilisation, le capteur nécessite une librairie disponible sur MBED.

3.6.3 Protocole de communication :

Ce capteur utilise le même protocole I2C de communication que pour les capteurs HTU21D et le MS5611. plus de détails du protocole sont donnés dans la documentation de référence.

On peut remarquer que la tension d'alimentation est de 1.8 à 3.6 V et que la consommation est de 46 mW pour une alimentation en 1.8V en mode actif.

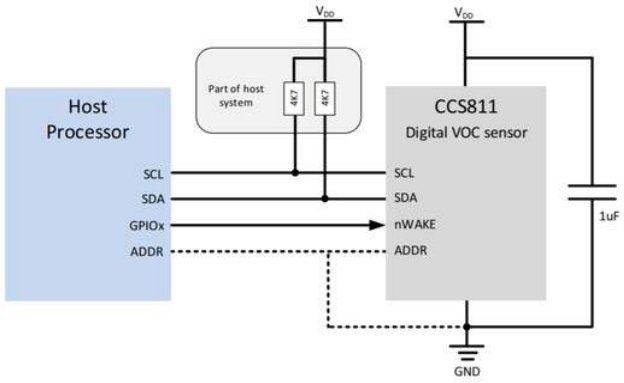

Figure 3.16 Circuit de branchement du CCS811

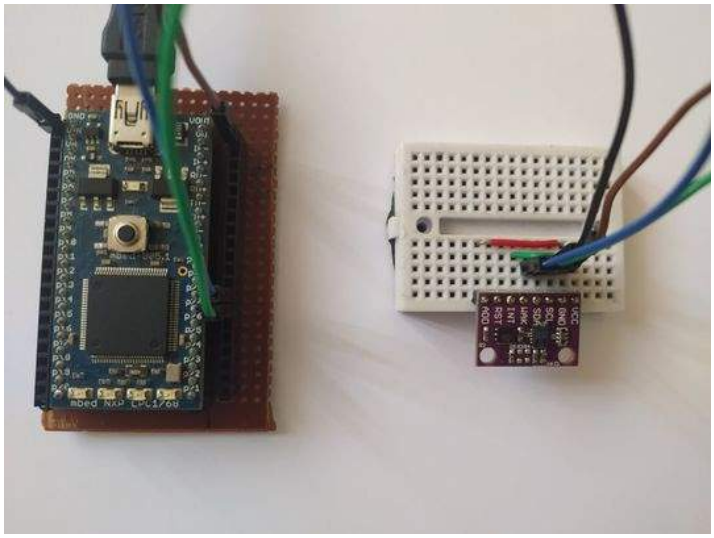

Figure 3.17 Circuit du capteur CCS811.

3.7 Le microcontrôleur “MBED NXP LPC1768” :

Le microcontrôleur mbed NXP LPC1768 fait partie de la série de cartes de développement de microcontrôleurs ARM conçues pour le prototypage rapide de toutes sortes d'applications et de systèmes, en particulier pour le milieu de l’embarqué. [37]
Il est présenté sous la forme d'un petit facteur de forme DIP pour le prototypage avec des circuits imprimés traversants, un stripboard et une maquette, et comprend un programmeur USB FLASH intégré. [37]

Figure 3.18 Microcontrôleur MBED NXP LPC1768.

Les détails techniques de ce composant sont donnés sur le site [37] et sont résumés en annexe E.

Pour la programmation de la carte, nous avons utilisé l’IDE en ligne : https://os.mbed.com/ , éditeur et compilateur de code en ligne gratuit, qui est une plate-forme et un système d'exploitation (MBED OS) pour les appareils connectés à Internet basés sur des microcontrôleurs ARM Cortex-M 32 bits.

Le langage de programmation utilisé sur la plateforme est : **C/C++.**

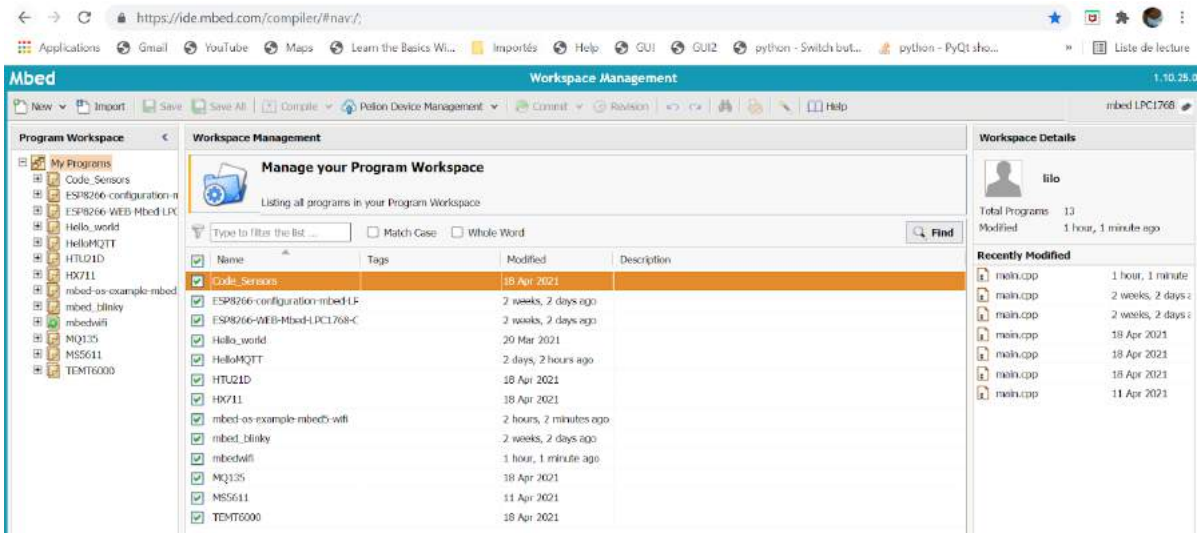

Figure 3.19 Plateforme os.mbed.

3.8 Collecte de données grâce au réseau de capteurs :

Pour l'utilisation des différents capteurs et la collecte des données respectives à chacun, il est nécessaire de disposer de librairies correspondantes à chacun d'entre eux contenant les différentes fonctions permettant la configuration et l'acquisition des données. Pour cela nous avons utilisé des librairies disponibles sur la plateforme MBED :

```
#include "mbed.h"
#include "HTU21D.h"
#include "MS5611I2C.h"
#include "HX711.h"
#include "CCS811.h"

HTU21D htu21d(p28,p27); //SDA,SCL
MS5611I2C ms5611(p28, p27, false);//SDA,SCL
CCS811 ccs811(p28, p27); //SDA,SCL
HX711 hx711(p6,p7,128);// p6 DT p7 SCK, gain
AnalogIn temt6000(p15);
```

Figure 3.20 Appel des librairies et attribution des pins.

PS : Les capteurs HTU21D, MS5611 et CCS811 sont branchés sur un même bus I2C.

Il est clair que l'utilisation d'un Bus I2C permet de réduire considérablement la complexité du câblage. d'ailleurs cette architecture de Bus multiplexé se généralise maintenant dans tous les systèmes avec des adaptations des protocoles en fonction de la créticité et de l'embarquabilité. On retrouvera par exemple le Bus CAN comme bus de multiplexage dans le monde de l'automobile.

Pour tester le bon fonctionnement de nos capteurs, nous avons utilisé des tests unitaires sur les fonctions de librairies et avons utilisé la communication Série entre le MBED LPC1768 et notre ordinateur pour visualiser les données acquises. Nous avons utilisé CoolTerm, un logiciel qui permet la lecture de données échangées via un port Serial du PC.

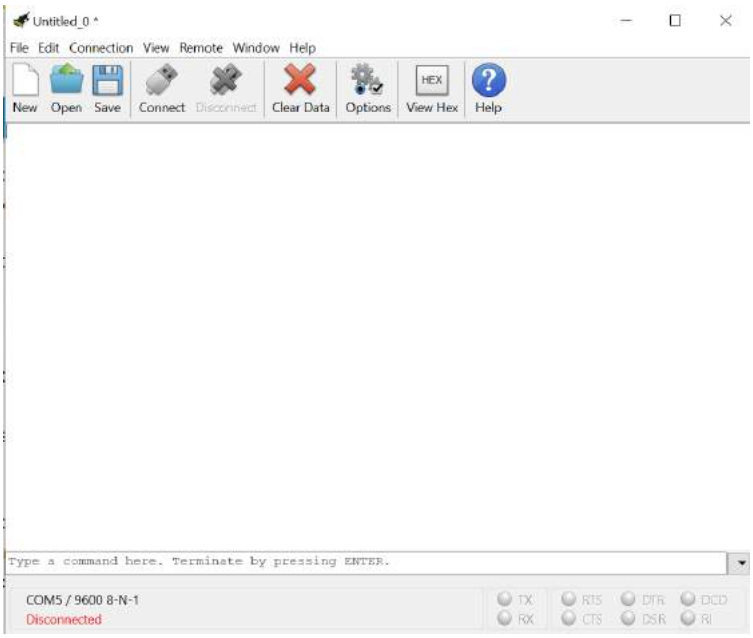

Figure 3.21 Interface de CoolTerm.

Un simple code d'essai permettant de faire une lecture sur chaque capteur et de récupérer ces données nous donne les résultats suivants :

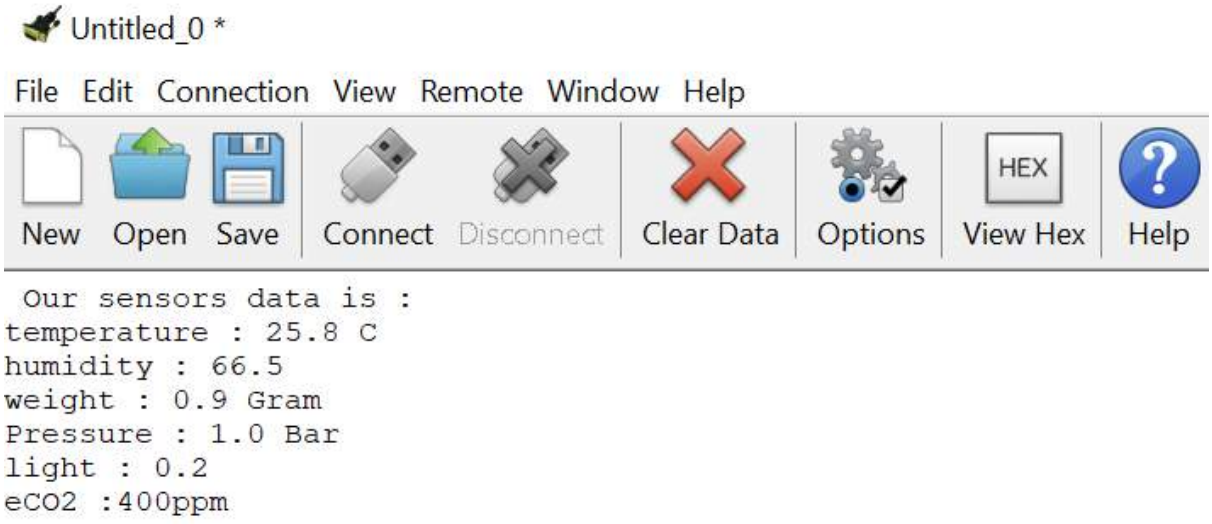

Figure 3.22 Données des différents capteurs du réseau.

3.8.1 Emplacement du réseau de capteurs au niveau de la ruche pour la collecte des données :

Il est important de bien placer les capteurs au niveau de la ruche pour ne pas fausser les mesures et assurer une fiabilité et justesse de ces dernières, pour cela nous avons fait une étude qui stipule ce qui suit :

- Le capteur de poids est naturellement disposé en dessous de la ruche, cette dernière reposera sur la balance reliée au module HX711 :

Figure 3.23 Balance de poids.

- Les capteurs de température/humidité, pression et taux de CO2 doivent être insérés à l'intérieur du corps de la ruche pour assurer les bonnes conditions de mesure de ces derniers :

Figure 3.24 Emplacement des capteurs de température/humidité, pression et taux de CO2 à l'intérieur de la ruche.

- Il faut installer le capteur de lumière sous la ruche pour obtenir l'ensoleillement diffus.
- L'ensoleillement direct n'est pas forcément pertinent car il saturera systématiquement le capteur.
- L'ensoleillement diffus renseigne mieux sur le niveau d'éclairage de la journée et évite les mesures erronées dues au passage des nuages :

Figure 3.25 Emplacement du capteur de lumière.

Chapitre 4 : Protocoles de communication

4.1 Communication au sein du système :

La communication au sein du système consiste en deux parties distinctes :

- Communication série entre le MBED LPC1768 et l'ESP8266 01s :

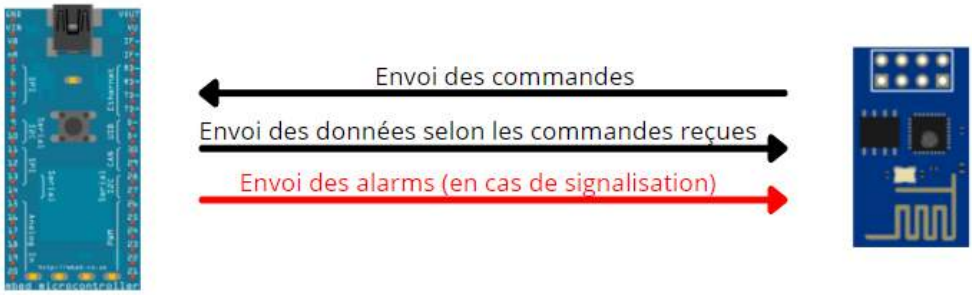

Figure 4.1 Communication entre le LPC1768 et l'ESP8266 01s.

- Communication via le protocole MQTT entre l'ESP8266 01s et l'outil Node-Red :

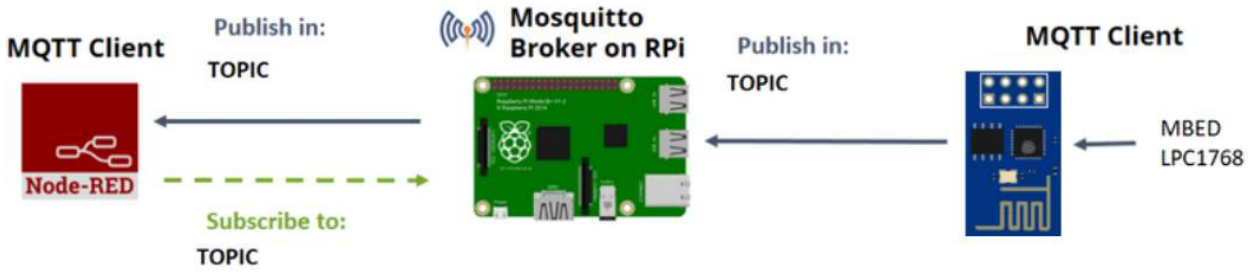

Figure 4.2 Communication entre le LPC1768 et l'outil Node-Red.

4.2 Module WIFI ESP8266 01s :

Le module WiFi ESP8266 est un SoC (System on Chip) autonome avec une pile de protocoles TCP/IP intégrée permettant l'adhésion d'un accès au réseau WiFi à n'importe quel autre microcontrôleur. [39]

L'ESP8266 est capable soit d'y programmer une application, soit d'asséner les fonctions de mise en réseau WiFi d'un autre processeur d'application. [38]

Chaque module ESP8266 est préprogrammé avec un firmware de jeu de commandes AT, permettant une connexion directe aux autres microcontrôleurs pouvant ainsi bénéficier d'autant de capacités WiFi provenant des offres de Shield WiFi, Toutefois ce firmware n'est plus utile lors de la programmation de son propre code sur l'ESP8266.[38]

Ce module dispose d'une capacité de traitement et de stockage embarquée suffisamment puissante qui lui permet d'être intégré aux capteurs et à d'autres dispositifs spécifiques aux applications via ses GPIO avec un développement initial minimal et un chargement rapide pendant l'exécution, les ports GPIO peuvent notamment être utilisée pour une communication série.[38]

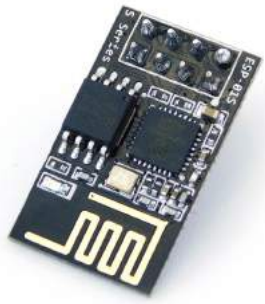

Figure 4.3 ESP8266 01s.

Les modules intégrant ce circuit sont très utilisés pour contrôler des périphériques par Internet. Toute la souplesse et la puissance de ce module résident dans la possibilité d'y développer et flasher un code spécifique à l'application visée, rendant ainsi le module entièrement autonome. [39]

L'ESP8266 est apparue en 2014 et le module a gagné rapidement en popularité. Il existe maintenant de nombreuses cartes à base de processeur ESP8266. Leurs caractéristiques diffèrent notamment en termes de nombres de ports d'entrées-sorties ou de taille de la mémoire flash. [39]
De petite taille, l'ESP8266 01s dispose de 8 broches. Il diffère des autres modèles sur les critères suivants :

- Il ne dispose pas de port USB pour le raccordement à un ordinateur.
- Ne présente que 2 ports GPIO (4 en comptant les broches Tx et Rx comme GPIO01 et GPIO03).
- Alimentation en 3.3v.

4.2.1 Rôle des broches :

- Tx/Rx : Liaison série mais peuvent également servir de GPIO (GPIO01 et GPIO03 respectivement).
- CH_PD : chip power-down doit être raccordé au 3,3V. Si cette broche est raccordée au GND, l'Esp entre dans un mode "basse consommation" dans lequel toutes les fonctions sont figées et le programme ne s'exécute plus. [39]
- GPIO0 : Utilisable comme broche I/O mais également permet d'entrer dans le mode « Flash mode » si elle est connectée à la masse lors de la mise sous tension (puis replacée sur VCC quelques secondes après). Dans ce mode, l'ESP8266 ne démarrera pas son programme interne mais entrera dans un mode appelé « UART download mode » où il écrira tout ce qu'il reçoit sur la liaison série, dans sa mémoire. C'est de cette manière qu'il faut procéder pour injecter un programme. Cette broche dispose d'une résistance de rappel (pull up) qui fait que par défaut elle est en état haut. [39]
- GPIO2 : Utilisable comme broche I/O. Cette broche dispose d'une résistance de rappel (pull up) qui fait que par défaut elle est en état haut. [39]

- RST doit être raccordé au 3.3v. Une pulse à GND sur cette broche génère un reset du composant. Le reset permet à l'Esp de redémarrer en exécutant son setup et lui permet également de sortir d'une mise en sommeil.

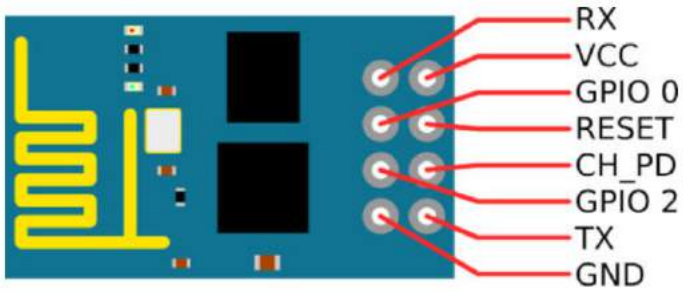

Figure 4.4 Description des pins de l'ESP8266 01s.

4.2.2 Etat des broches au démarrage :

Lorsque l'Esp démarre suite à une mise sous tension, l'état dans lequel se trouve la broche GPIO0 va déterminer le mode de fonctionnement du microcontrôleur.
Les broches RST et CH_PD doivent également être dans un état fixé [39]:

- La broche GPIO 0 doit être maintenue à la masse (GND) pendant le téléversement du code. Elle doit être déconnectée lorsque l'ESP passe en mode d'exécution normal ;
- La broche CH_PD doit toujours être maintenue à l'état haut ; la broche RESET est tirée à l'état haut avec une résistance de pull-up 10 kΩ, et est reliée à la masse GND sur appui du bouton RESET lorsqu'il faut redémarrer l'ESP. Il est important de presser le bouton RESET à chaque téléchargement du code, et à chaque reconnexion de la broche GPIO 0.

GPIO0	Mode de démarrage	Description
HIGH	Normale	Le microcontrôleur boot de manière normale sur son programme
LOW	UART	Le microcontrôleur boot sur un mode permettant l'injection de programme par liaison série (via les ports Rx/Tx)

Table 4.1 Modes de démarrage de l'ESP 01s.

4.2.3 Programmation :

Pour l'injection des programmes de l'ordinateur vers l'ESP8266 01s nous avons utilisée l'environnement de développement Arduino (il est possible d'utiliser tout autre environnement de développement capable de compiler du code pour ESP8266), et à cet effet deux conditions doivent être respectées :

- **Pouvoir connecter l'ESP8266 01s à l'ordinateur :** la seule possibilité est de le faire via un port USB. Or l'ESP8266 01s ne dispose pas de ce dernier. Il a donc fallu trouver un équipement qui sera capable de se connecter d'un côté en USB à l'ordinateur, et de l'autre à l'ESP.

  A cet escient, il est possible d'utiliser un programmateur FTDI 3.3V, Ce genre de modules embarquent une puce de conversion USB-série FTDI232RL permettant d'envoyer et recevoir des signaux UART via le port USB d'un ordinateur et de flasher un programme dans l'ESP.

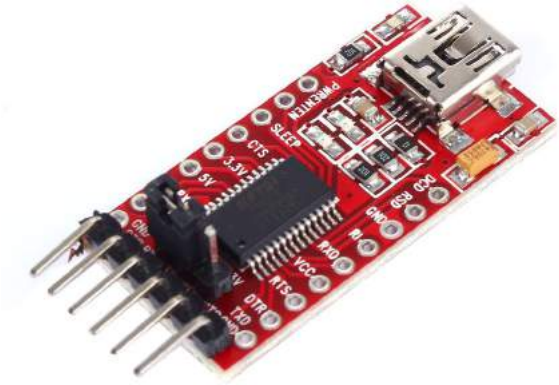

Figure 4.5 Programmateur FTDI.

- **Activer le mode "flash" :** Ce mode est nécessaire pour injecter un programme dans la mémoire de l'ESP. Le mode flash s'active quand l'ESP démarre avec sa broche GPIO0 raccordée au pôle négatif (GND).

4.2.4 Montage :

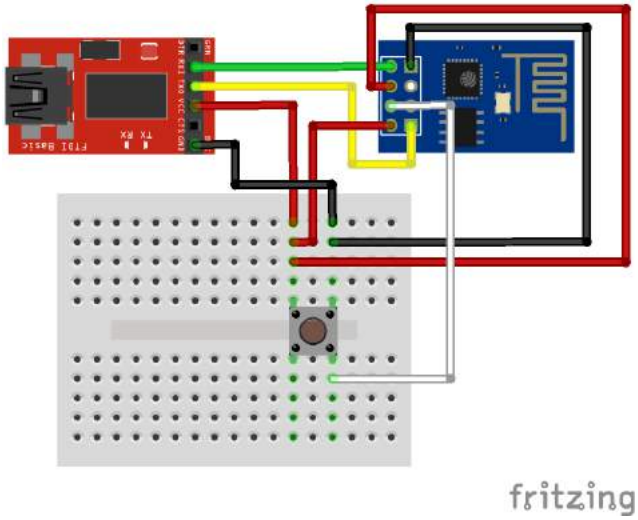

Figure 4.6 Branchement ESP8266 01s et programmateur FTDI.

4.3 Protocole MQTT :

4.3.1 Introduction à MQTT :

MQTT est un protocole de transport de messagerie entre un serveur et un client suivant un principe de publication/souscription (Publish/Subscribe). Très peu gourmand en espace mémoire et open source, il est conçu de manière à être facile et simple d'implémentation. Ces caractéristiques le rendent idéal pour une utilisation dans plusieurs situations notamment dans le contexte des communications machine à machine (M2M) et de l'internet des objets (IoT) où le code doit être optimisé et/ou la bande passante du réseau est primordiale. [40]

MQTT signifie MQ Telemetry Transport. "MQ" fait référence à la série MQ, un produit IBM. Plusieurs sources traduisent MQTT comme un Massage Queu protocole, ce qui est incorrect. [40]

4.3.2 Le modèle Publish/Subscribe :

Le modèle pub/sub (Publish/Subscribe) est une alternative de l'architecture traditionnelle client-serveur. Dans cette dernière, un client communique directement avec un destinataire. Dans le modèle pub/sub, les clients destinateurs (Publishers) et le ou les clients destinataires (Subscribers) sont découplés. Les Publishers et les Subscribers ne communiquent jamais directement entre eux, ils n'ont même pas conscience de l'existence de l'un et l'autre. La connexion entre eux est gérée par une troisième partie qui n'est autre que le broker. Ce dernier a comme attribution de filtrer tous les messages publiés et de les distribuer aux Subscribers ciblés, le filtrage permet entre autres de contrôler quel client souscripteur reçoit quel message.
Le découplage des clients selon la nature de la transaction (émission/réception) qu'ils effectuent comporte trois dimensions : spatial, temporel et de synchronisation [40]:

- Découplage spatial : les Publishers et Subscribers n'ont pas besoin de se connaître (ex: pas d'échange d'adresse IP).
- Découplage temporel : les Publishers et Subscribers n'ont pas besoin d'être actifs à la même période.
- Découplage de synchronisation : Les opérations sur les deux composants n'ont pas besoin d'être interrompues pendant la publication ou la réception.

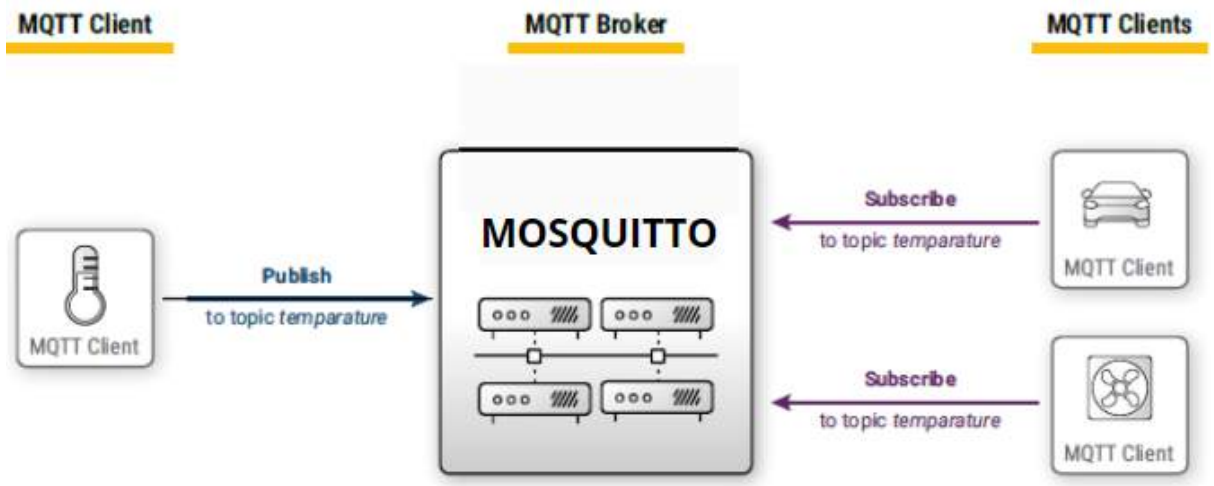

Figure 4.7 Exemple du modèle pub/sub.

Pub/Sub évolue mieux que l'approche client-serveur traditionnelle. En effet, les opérations sur le broker peuvent être hautement parallélisées et les messages peuvent être traités de manière événementielle. [40]

Selon l'application visée, le protocole MQTT incarne tous les aspects du modèle pub/sub mentionnés [40]:

- MQTT découple les clients spatialement, les Publishers et Subscribers ont besoin de connaître uniquement l'adresse IP et le port du broker.
- MQTT effectue le découplage temporel. Bien qu'il soit généralement utilisé pour des applications en temps réel, il peut, si souhaité, stocker les messages pour les clients qui ne sont pas connectés.
- MQTT fonctionne de manière asynchrone. Étant donné que la plupart des bibliothèques utilisées par les clients fonctionnent de manière asynchrone et sont basées sur des rappels ou un modèle similaire, les tâches ne sont pas bloquées lors de l'attente d'un message ou de la publication d'un message. Dans certains cas d'utilisation, la synchronisation est souhaitable et possible. Pour attendre un certain message, certaines bibliothèques ont des API synchronisées. Mais le flux est généralement asynchrone.
- MQTT est particulièrement facile à utiliser du côté du client, il est l'essence du modèle pub/sub quant à l'utilisation des bibliothèques du client rendant le protocole léger en taille pour des appareils petits.

Plus de détails sur le protocole MQTT sont disponibles en annexe F.

4.4 Broker Mosquitto sur Raspberry pi 3 :

Comme expliqué précédemment, le protocole MQTT permet de transférer des données entre différents clients connectés, pour ce faire, cette communication nécessite un "Broker".

Un Broker est un programme qui prend en charge la réception des informations publiées afin de les transmettre aux clients abonnés. Le broker a un rôle de relais ou d'intermédiaire. Il existe plusieurs types de brokers dont on cite : ActiveMQ, JoramMQ, RabbitMQ, ou encore, Mosquitto pour lequel nous avons opté. [41,42]

Eclipse Mosquitto est un open source message broker qui implémente différentes versions du protocole MQTT. Mosquitto est léger et convient à une utilisation sur tous les appareils, des ordinateurs monocarte à faible consommation aux serveurs complets, c'est pour cette raison que le choix s'est porté sur ce dernier car son implémentation est simple et correspond parfaitement à notre Raspberry pi qui fera office de serveur. [43]

Figure 4.8 Broker MQTT Mosquitto.

4.4.1 Implémentation de Mosquitto sur Raspberry :

- L'installation se fait par une simple commande : sudo apt-get install mosquitto
- Les clients MQTT nécessitent également une installation : sudo apt-get install mosquitto-clients
- Lancer Mosquitto : **mosquitto**

```
lili1310@Lilia:~$ mosquitto
1624194324: mosquitto version 1.4.15 (build date Tue, 18 Jun 2019 11:42:22 -0300) starting
1624194324: Using default config.
1624194324: Opening ipv4 listen socket on port 1883.
1624194324: Opening ipv6 listen socket on port 1883.
```

Figure 4.9 Lancement de Mosquitto sur terminal Linux.

- Le serveur écoute sur différents ports dont le 1883 qui représente le port par défaut standard pour le protocole MQTT.

4.4.2 Sécurisation du broker Mosquitto :

Il est important d'assurer une sécurisation des communications entre le broker MQTT et les différents clients, périphériques ou logiciels qui vont souscrire et/ou publier aux topics. A cet effet, il est nécessaire de modifier un fichier de configuration du broker Mosquitto pour y introduire le nom d'utilisateur ainsi que le mot de passe souhaités.

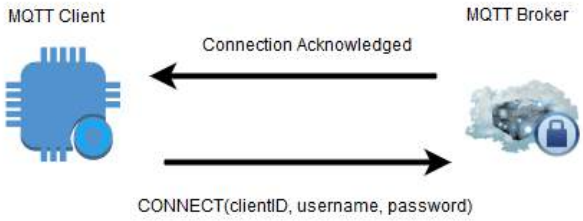

Figure 4.10 Effet de la sécurisation du broker dans la communication MQTT.

Cette sécurisation fait en sorte que lors des communications entre broker et clients MQTT, il est indispensable de fournir, en plus de l'identifiant client, le username et mot de passe du broker Mosquitto pour autoriser l'accès.

4.4.3 Attribution d'une adresse IP statique au broker :

Fixer une adresse IP statique a pour but d'attribuer au serveur Raspberry une adresse fixe qui ne change pas, cette opération est primordiale dans le cadre du projet car l'ESP8266 01s, étant un client MQTT, nécessite de connaître l'adresse du broker pour assurer la communication entre eux, une adresse identique peut être définie sur le code de l'ESP sans devoir la changer selon l'IP dynamique attribué à chaque fois au broker.
Pour cela, nous avons récupéré les différentes adresses IP caractérisant le réseau, entre autres l'IP du gateway et du DNS, nous avons fixé une adresse appartenant à la plage d'attribution du réseau pour le serveur Raspberry, et nous avons opéré la configuration nécessaire pour effectuer cette opération.

4.4.4 Essai de Publish/Subscribe entre deux terminaux connectés à Mosquitto :

Grâce aux fonctions Subscribe et Publish de Mosquitto, il est possible de faire un essai pour visualiser le transfert des données entre deux objets connectés au broker, pour cela, il est important de rajouter l'adresse IP du broker (définie par le localhost ici qui est statique), le username et le mot de passe prédéfinis dans la partie sécurisation :

```
lili1310@Lilia:~$ mosquitto_pub -h localhost -t TOPIC_test -m "Publishing a test topic through mosquitto" -u mqtt_broker
 -P mqtt_pw
lili1310@Lilia:~$
```

Figure 4.11 Essai d'un Publish sur Mosquitto.

```
lili1310@Lilia:~$ mosquitto_sub -h localhost -t TOPIC_test -u mqtt_broker -P mqtt_pw
Publishing a test topic through mosquitto
```

Figure 4.12 Essai d'un Subsribe sur Mosquitto.

4.5 Communication entre ESP8266 01s et le Broker :

La communication entre l'ESP8266 01s et le broker consiste en l'implémentation d'un client MQTT au niveau de l'ESP, entre autres connecter ce module via un protocole MQTT au broker pour qu'il puisse souscrire à des topics ou en publier. Pour établir cette communication, deux librairies sont nécessaires : ESP8266WiFi et PubSubClient.

4.5.1 Librairie ESP8266WiFi :

- Il est d'abord nécessaire de connecter le module à un même réseau WiFi local que le broker implémenté sur la Raspberry via la librairie citée. Pour cela, il faut créer au niveau de l'ESP un client WiFi pour assurer les échanges avec le serveur :

```
#include <ESP8266WiFi.h>
WiFiClient espClient;
```

- La connexion est établie grâce à la fonction WiFi.begin, ceci requiert les différents paramètres du réseau, entre autres le nom et mot de passe. Une boucle de confirmation est ajoutée pour vérifier si le client ESP est bel et bien connecté, une reconnexion est lancée en cas contraire :

```
WiFi.begin("Ooredoo 4G_4744D7","20152015");
while ((!(WiFi.status() == WL_CONNECTED))){
  delay(300);
  Serial.print("\n trying again"); }
```

- Une fois la connexion validée, une adresse IP locale est attribuée au client pour servir lors de ses échanges avec le serveur, il est possible d'afficher cette dernière :

```
Serial.println(WiFi.localIP());
```

- La figure suivante permet de résumer cette partie :

Figure 4.13 Établissement d'un client WiFi au niveau de l'ESP 01s.

4.5.2 Librairie PubSubClient :

- Cette librairie disponible sur Arduino permet d'implémenter un client MQTT au niveau de l'ESP. Suite à la configuration du client WiFi, il est possible de déclarer le même client en client MQTT :

```
#include <PubSubClient.h>
PubSubClient client(espClient);
```

- La configuration de la connexion au broker MQTT nécessite d'avoir les différents paramètres de ce dernier, entre autres le nom et le mot de passe, et les autres détails qui définissent le serveur à travers la fonction setServer :

```
char* mqtt_user ="ruche";
char* mqtt_pw ="ruche";
client.setServer("192.168.0.160", 1883);
```

- Suite à quoi, la connexion sera établie à l'aide de la fonction connect :

```
boolean connect (clientID, [username, password], [willTopic, willQoS,
willRetain, willMessage], [cleanSession]).
```

- Les différents paramètres : willTopic, willQoS, willRetain, willMessage et cleanSession, sont optionnels est expliqués en annexe F. L'établissement de la communication est fait via une fonction qui permet aussi de vérifier si la connexion a été convenablement établie en retournant true ou false, on peut éventuellement la relancer en cas contraire, comme suit :

```
while (!client.connected()){
        String clientId = "ESP8266Client-";
        if (client.connect(clientId.c_str(), mqtt_user, mqtt_pw)){
        Serial.print(client.state());}}
```

- Une fois l'ESP implémenté en client MQTT, il est possible de faire appel aux fonctions publish et subscribe, par exemple :

```
client.publish("TEMPERATURE", msgmqtt);
```

- La fonction de rappel est une partie importante lors des échanges avec le broker pour assurer une bonne communication et synchronisation d'un bout à l'autre, cette fonction Callback est appelée lorsqu'un problème se produit au niveau de cette communication, ceci arrive dans trois cas :

➢ Connection lost : quand la connexion entre le broker et le client est perdue ou interrompue.

➢ Delivery complete : est appelée pour que le client MQTT puisse remettre un jeton permettant d'accéder au message publié.

➢ Message arrived : apparaît lorsqu'une publication arrive pour le client qui s'est souscris au topic correspondant.

```
client.setCallback(callback);
```

4.6 Communication série entre MBED LPC1768 et ESP8266 01s :

L'échange est basé sur un protocole de communication série selon un transfert de données asynchrone (UART), l'émetteur, LPC ou ESP, doit fournir un signal de synchronisation avant de pouvoir transférer des données, et ce avant chaque message à transmettre.

Le principe de synchronisation de l'échange entre le LPC et l'ESP se fait par des interruptions asynchrones, la communication a été structurée comme suit :

- Communication de l'ESP vers le LPC :
- Envoi d'une commande d'interruption i de l'ESP pour demander une certaine donnée d'un capteur i.
- Réception de la commande d'interruption par le LPC et exécution de la fonction d'acquisition correspondante avant l'envoi de la donnée.

- Communication du LPC vers l'ESP :
- Envoi d'une interruption d'alerte '1' en cas d'enclenchement d'un problème vers l'ESP et '0' une fois que l'alarme n'est plus active.

Pour ce faire, le branchement a été établi comme le montre la figure :

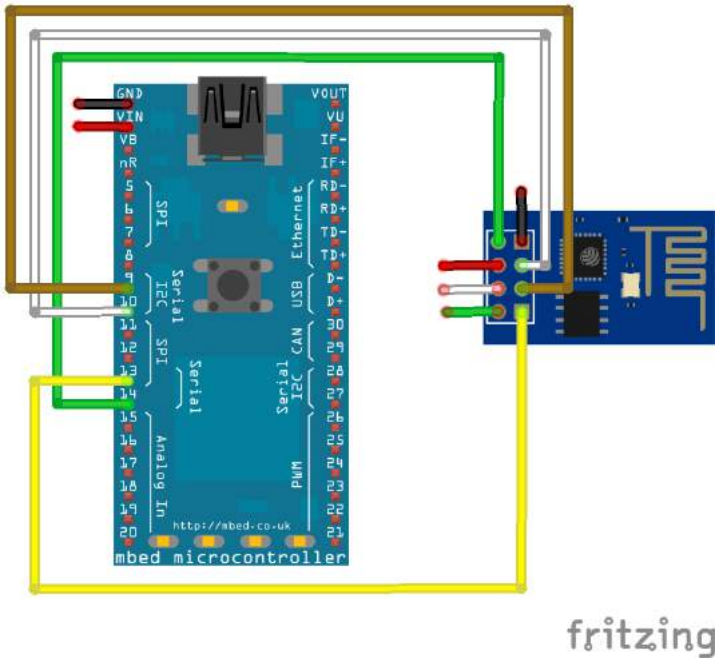

Figure 4.14 Schéma de branchement du LPC1768 et de l'ESP 01s.

- Le transfert des alarmes s'effectue via la liaison série UART (pin 9/10) du LPC vers le contrôleur ESP01_s (pin GPIO 0 et 2).
- L'échange de données a été établi à travers la liaison Rx/Tx de l'ESP et les pins p13/p14 du LPC.
- Les échanges de données et d'alarme ont été dissociés (sur deux liaisons différentes) pour assurer la validité de la réception sans avoir de chevauchement au niveau des informations émises.

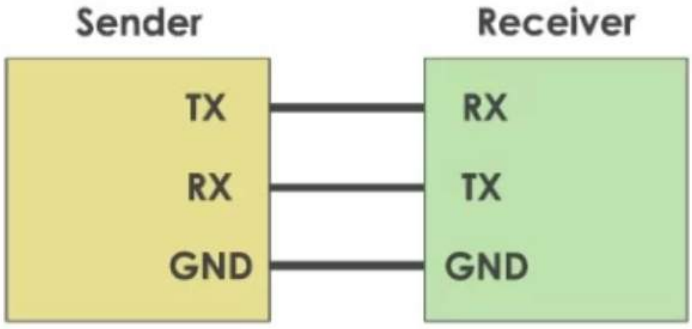

Figure 4.15 Liaison série UART

Les deux communications se basent sur un échange série choisi pour différents avantages :

- Nécessité d'une seule ligne pour communiquer et transférer des données.
- Utilisation d'une transmission en mode half-duplex disponible pour assurer un échange dans les deux sens avec alternance de l'émission et de la réception, comme dans le cas de l'ESP ou il est nécessaire d'envoyer des commandes d'interruption et de recevoir les données correspondantes suite à cela.
- La transmission est fiable et directe, c'est la plus pratique dans le cadre de la communication à établir.

4.6.1 Les organigrammes :

Les organigrammes des deux cartes ont été établi pour permettre une représentation graphique de l'enchaînement des opérations et des décisions effectuées par le LPC1768 et l'ESP8266, ci-joint ces deux représentations :

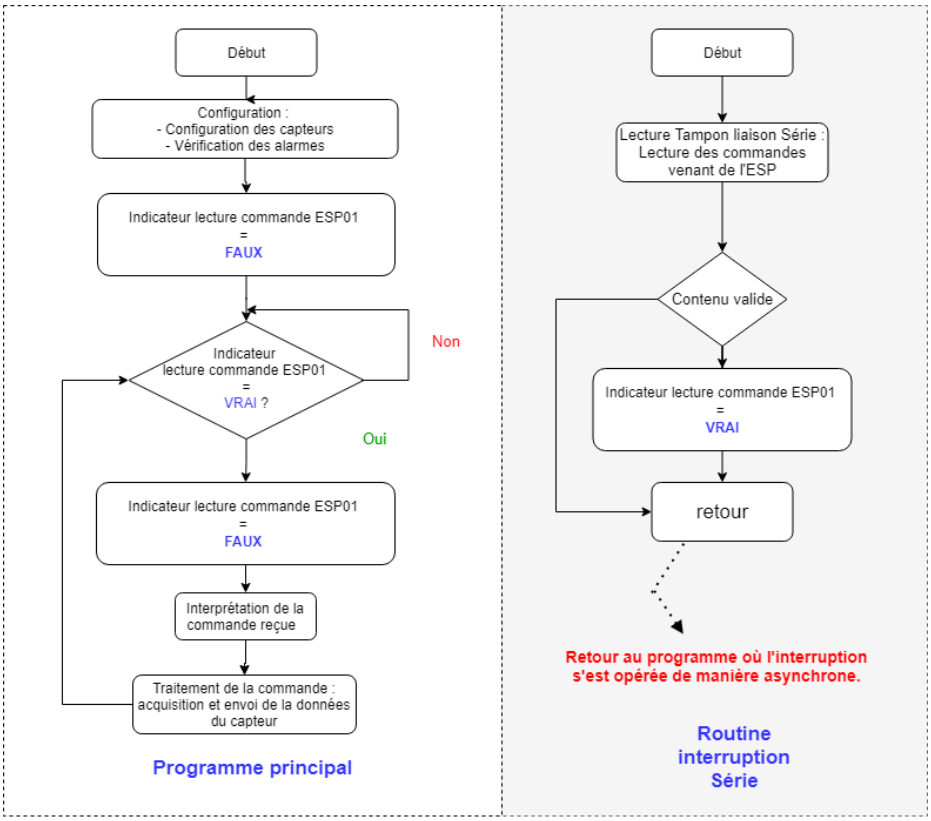

Figure 4.16 Organigramme du MBED LPC1768.

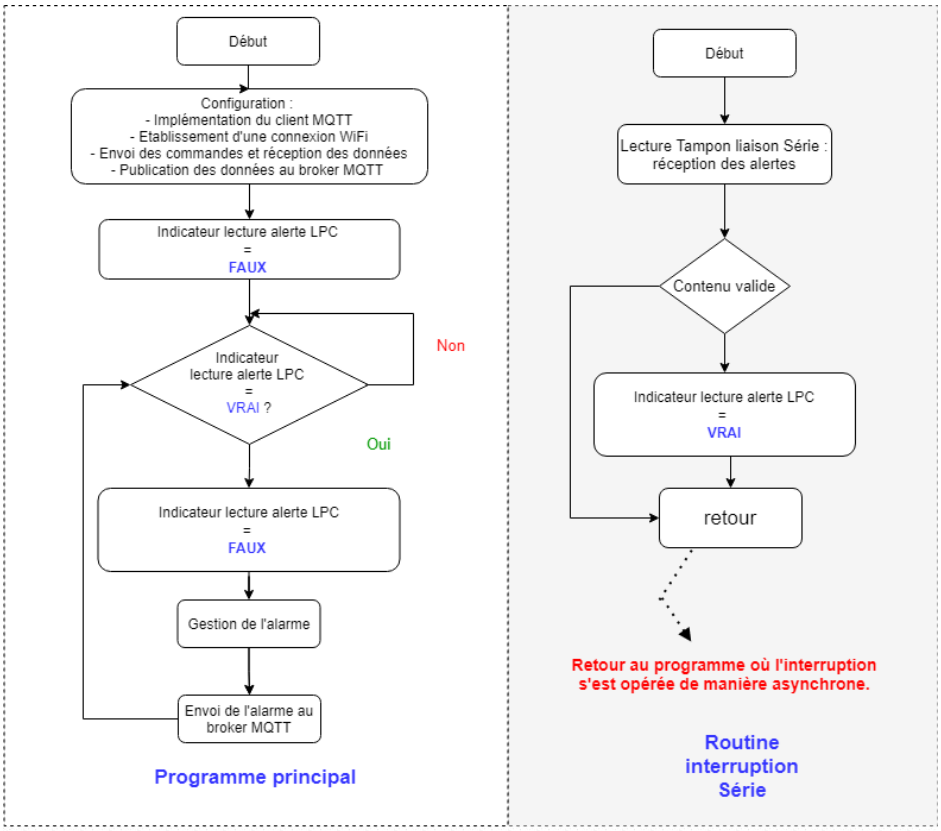

Figure 4.17 Organigramme de l'ESP8266 01s.

4.6.2 Synchronisation des échanges :

Malgré l'aspect asynchrone de la liaison série établie suite aux différentes interruptions échangées qui permettent d'enclencher la communication, il est primordial d'assurer une synchronisation entre l'envoi des commandes de l'ESP et la réception des données qui correspondent à cette demande.
Pour cela, les différents temps d'acquisition des capteurs ont été établis :

Capteur :	HX711	HTU21D (Température)	HTU21D (Humidité)	MS5611	CCS811	TEMT600
Temps d'acquisition (ms) :	3378	67	33	34	35	33

Table 4.2 Temps d'acquisition des différents capteurs.

Il est important que l'ESP valide un temps d'attente qui permet l'acquisition de la donnée avant d'envoyer une nouvelle commande. Pour réaliser cela, nous avons effectué des délais entre chaque deux envois qui correspondent aux différents temps d'acquisition mesurés.

Chapitre 5 : Base de données et Dashboard de l’apiculteur.

5.1 Outil Node-Red :

5.1.1 Description de l'outil Node-Red :

Node-Red est un outil de développement open source et gratuit conçu par IBM, il permet de gérer des flows d'événements, des séries de traitement à effectuer suite à la réception d'un message ou au déclenchement d'un événement. [49,50]

Node-Red fournit un éditeur de flux au sein du navigateur web, et est articulé autour de nœuds (nodes) qui représentent les fonctionnalités disponibles de cet outil. L'ensemble des nœuds ajoutés et liés formera un flow qui sera stocké à l'aide de fichier JSON, les fonctions sont enregistrées en JavaScript et les éléments des applications peuvent être enregistrés ou partagés pour être réutilisés. [49,50]

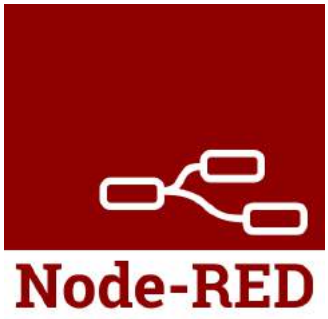

Figure 5.1 logo de Node-Red.

5.1.2 Interface Node-Red :

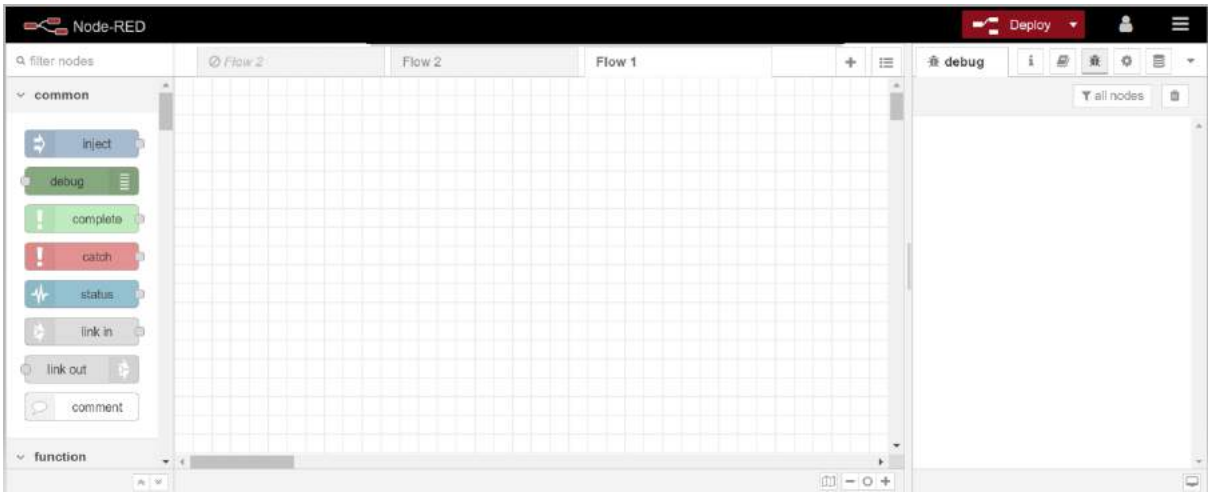

Figure 5.2 Interface Node-Red.

L'interface Node-Red est accessible en introduisant l'adresse IP de l'hôte sur lequel est installé le logiciel suivi du port 1880 (localhost:1880). L'interface est structurée en quatre parties [48]:

- Au centre : c'est l'espace de construction des flows, nous pouvons y développer autant que nous voulons de manière indépendante.

- À gauche : nous y trouvons la liste des nœuds disponibles, il suffit de sélectionner le nœud désiré et de le placer sur l'espace des flows.
- À droite : plusieurs onglets sont présents dont :

L'onglet i : permet d'avoir toutes les informations associées au nœud sélectionné.
L'onglet debug : représenté par une icône d'insecte, il permet de voir les messages de debug, pour cela il faudra placer un nœud debug.
L'onglet de configuration : il permet de configurer les nœuds.

- En haut : le bouton Deploy permet de déployer son flow et de le rendre actif, un bouton menu est également disponible contenant plusieurs options.

5.1.3 Implémentation de Node-Red comme client MQTT :

Node-Red est nativement en mesure de souscrire et de publier en MQTT, ce qui le rend un outil fort dans notre conception du système de la ruche connectée.
Parmi la liste des nœuds disponibles, nous trouvons le nœud relatif à la communication MQTT avec les options In et Out correspondant respectivement aux fonctions Subscribe et Publish. Dans notre cas nous avons implémenté l'outil Node-Red comme étant un client Subscriber, à cet effet nous avons eu recours à certaines configurations du nœud afin de le connecter à notre broker.
Pour cet envergure, nous avons procédés comme suit :

a- Nous avons sélectionné le nœud MQTT In et l'avons placé sur l'espace de notre flow.

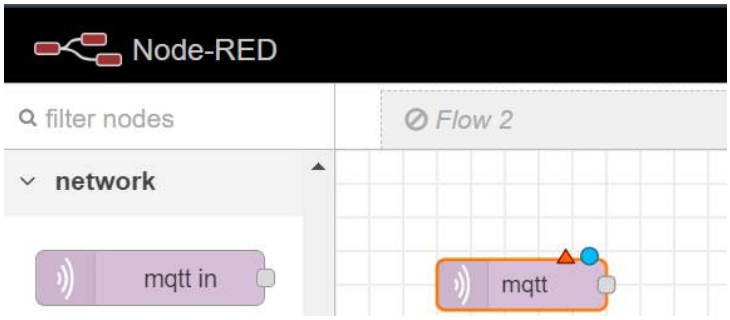

Figure 5.3 Nœud MQTT in

b- Par un double clique sur le nœud, une fenêtre de configuration s'offre à nous que nous avons complété comme suit :

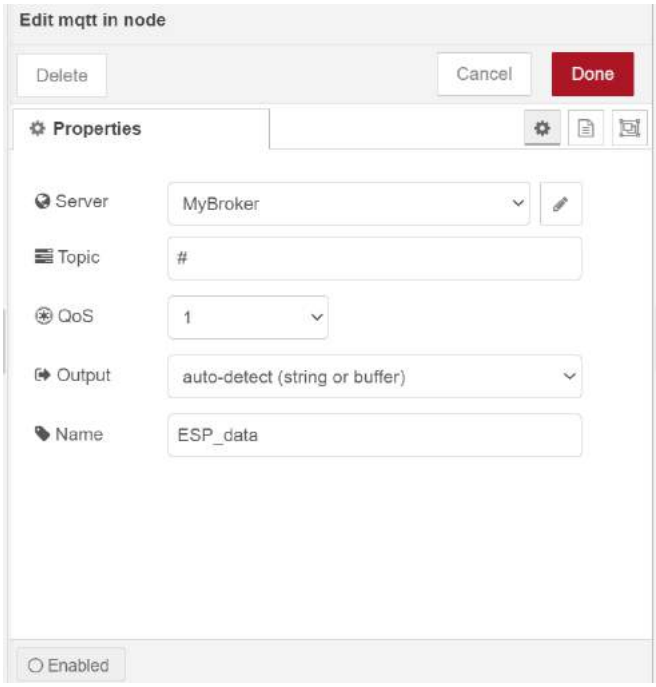

Figure 5.4 fenêtre de configuration du nœud MQTT.

Le signe "#" de la case Topic indique une souscription à tous les topics disponibles au niveau du broker, nous avons par ailleurs choisi un niveau de QoS réglé à 1 et avons nommé le nœud ESP_data en référence aux données publiées par le Publisher ESP8266 01s.

c- Grâce à l'onglet "Server" présent sur la fenêtre de configuration précédente, nous avons configuré les données relatives à notre broker et nécessaires pour l'établissement d'une connexion entre ce dernier et Node-Red :

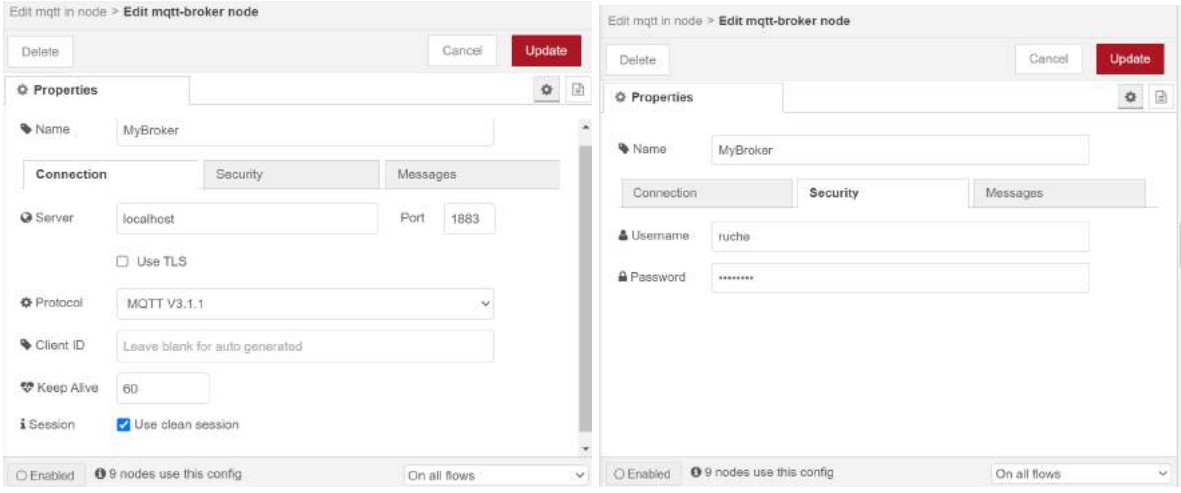

Figure 5.5 Configuration du broker.

Ainsi pour configurer le broker, il est important de spécifier son adresse IP (ici indiquée par localhost) ainsi que le port sur lequel s'établit la connexion, il est également important de

préciser la version du protocole MQTT qu'utilise le broker et aussi le Username et Password avec lesquels nous l'avons sécurisé.

d- Pour vérifier la réception des données; nous avons ajouté un nœud de debug, nous pouvons donc visualiser les données reçu grâce à l'onglet debug :

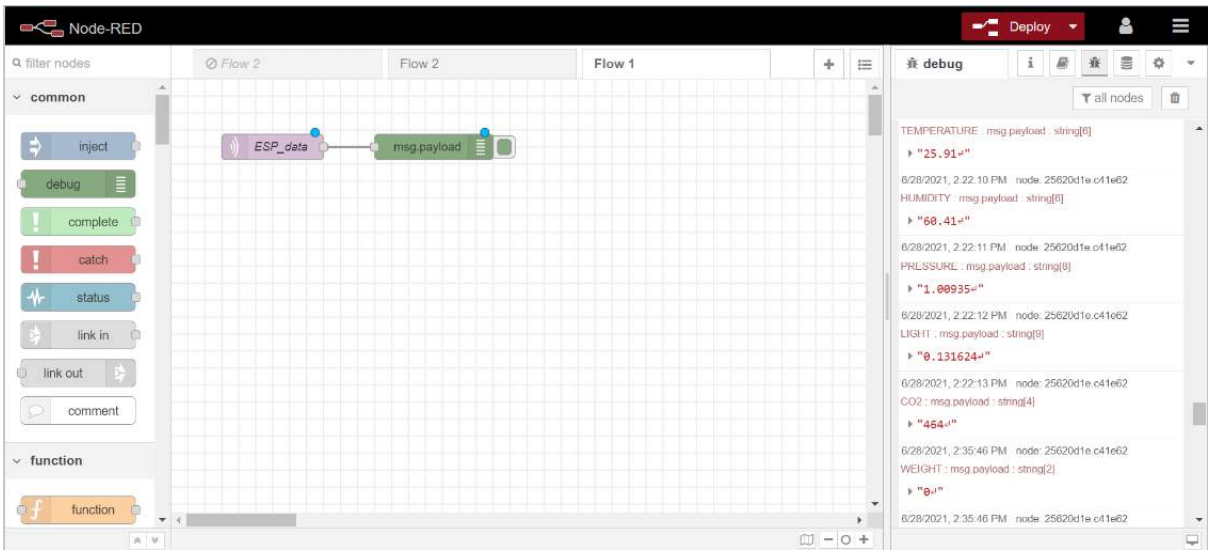

Figure 5.6 Visualisation des messages des topics souscrits via Node-Red.

e- L'outil Node-Red est maintenant implémenté comme client MQTT Subscriber, pour la suite de notre conception du système, nous transférerons les données reçues vers une base de données pour les y stockées. à cet escient, nous utiliserons le nœud InfluxDB out, qui permet d'insérer des données sur la base de données InfluxDB.

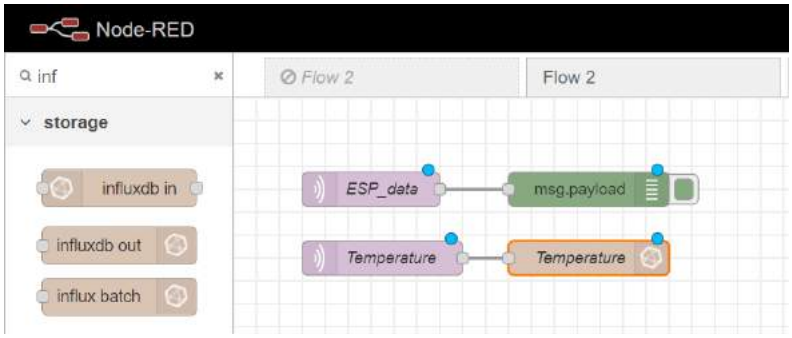

Figure 5.7 Ajout du nœud InfluxDB out pour la donnée de température.

Certains paramètres devront être configurés au sein du nœud, notamment le nom que nous voulons attribuer à la donnée au niveau de la base de données, le nom du Topic au niveau de Node-Red et la configuration du serveur selon les paramètres introduits lors de l'installation d'InfluxDB sur notre Raspberry Pi 3.

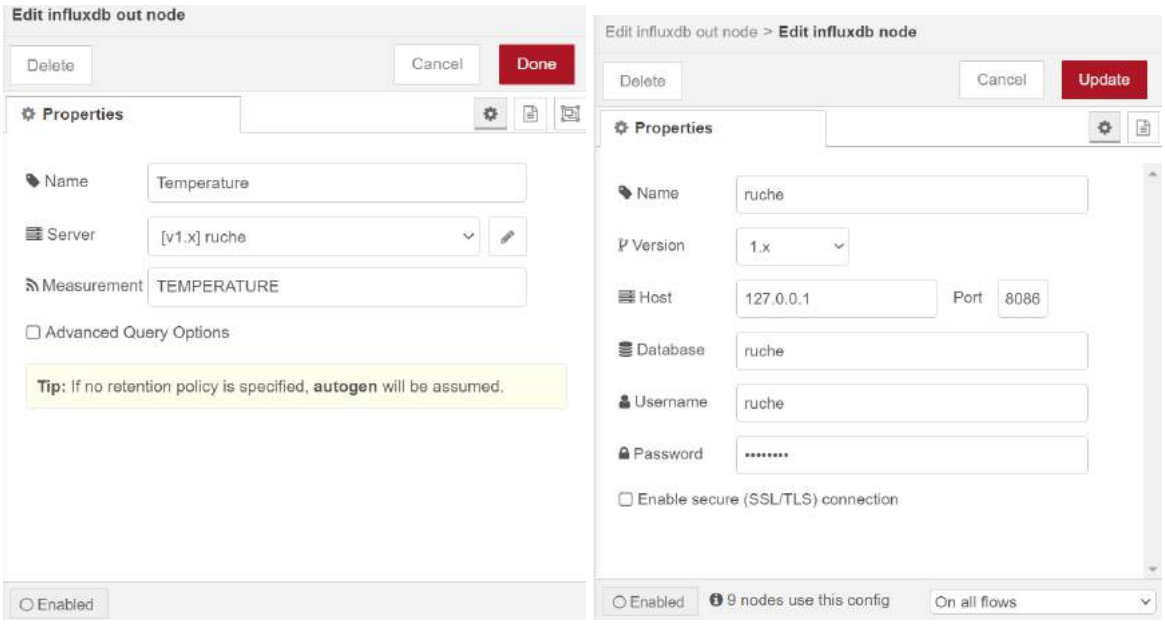

Figure 5.8 Configuration du nœud InfluxDB out pour la mesure de température.

Nous reproduisons la même procédure pour chaque topic souscrit pour transférer les données une à une à notre base de données InfluxDB.

5.2 Base de données InfluxDB :

5.2.1 Description de l'outil InfluxDB :

Le système de ruche connectée assure des mesures en temps réel importantes, les données collectées sont de ce fait en quantité énorme à de courts laps de temps, elles doivent faire l'objet d'un horodatage et être traitées correctement. Ces données chronologiques exigent donc des bases de données spéciales, c'est pourquoi il est nécessaire d'utiliser un outil comme InfluxDB.

InfluxDB est un système de gestion de base de données développé par la société InfluxData, Inc. InfluxDB est un logiciel open source et peut être utilisé gratuitement. [51,52]

Figure 5.9 Logo InfluxDB.

La nouvelle version InfluxDB 2.0 offre en plus un service cloud flexible et adaptable et comprend une interface utilisateur web pour saisir et visualiser les données. [51,52]

Le système de gestion de base de données InfluxDB utilise le langage de programmation Go de Google, encore appelé Golang. [52]
Le langage de requête adopté pour interroger les bases de données externes utilise dans la première version le langage InfluxQL (Influx Query Language), dans la version 2.0 un nouveau langage Flux est employé pour ce même but.[51]

5.2.2 Utilisation d'InfluxDB :

InfluxDB est conçu pour gérer les bases de données de type TSDB (time series database), qui contiennent des séries temporelles. Ces bases de données sont employées notamment pour stocker et analyser des données de capteurs ou autres horodatées sur une période donnée. [53]
Dans notre système, il est question d'un réseau de capteurs connectés qui fournit des mesures en flux continu. Ces données doivent être traitées rapidement une fois entrées dans la base de données, c'est pourquoi InfluxDB intègre un service de temps qui, grâce au Network Time Protocol (NTP), assure que l'heure est bien synchrone sur l'ensemble des objets connectés au réseau.
La conception de la base de données InfluxDB du système a été faite à travers les fonctionnalités disponibles sur l'outil Node-Red comme détaillé dans le paragraphe précédent, le flow réalisé est donné dans la figure suivante :

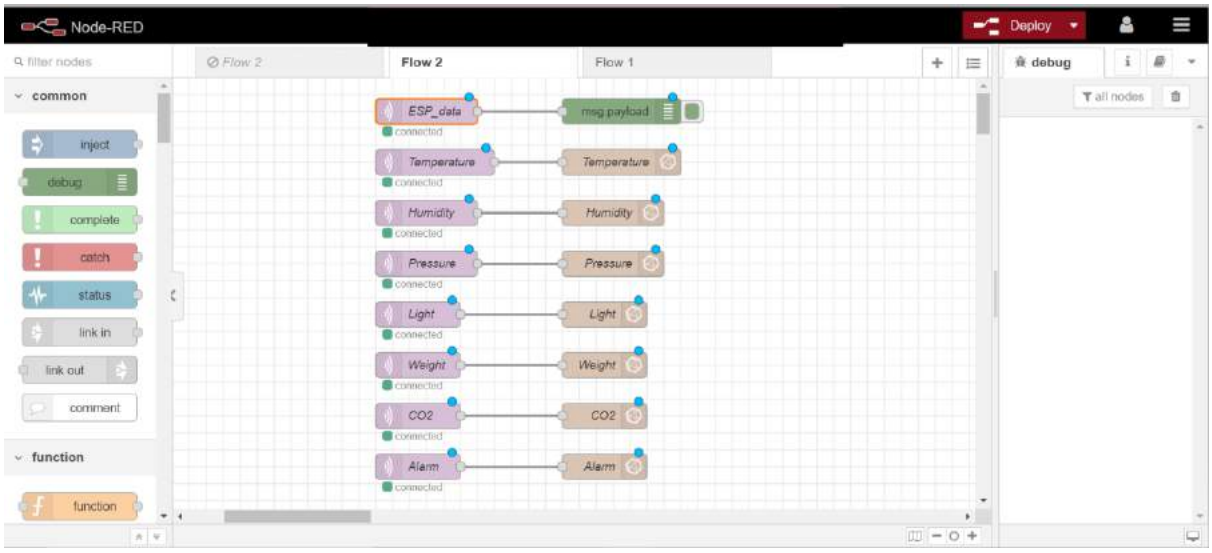

Figure 5.10 Flow réalisé sur Node-Red pour la configuration de la base de données InfluxDB.

La base de données créée pour le projet "ruche" peut être visualisée comme suit :

```
pi@raspberrypi:~ $ influx -username ruche -password ruche
Connected to http://localhost:8086 version 1.8.6
InfluxDB shell version: 1.8.6
> SHOW DATABASES
name: databases
name
----
_internal
ruche
>
```

Figure 5.11 Lancement InfluxDB et visualisation des bases de données disponibles.

Les différentes mesures de la base de données sont liées aux données apportées par les capteurs, entre autres : CO2, HUMIDITY, LIGHT, PRESSURE, TEMPERATURE et WEIGHT, ainsi qu'une mesure d'alarme pour stocker les états du système.

```
pi@raspberrypi:~ $ influx -username ruche -password ruche
Connected to http://localhost:8086 version 1.8.6
InfluxDB shell version: 1.8.6
> SHOW DATABASES
name: databases
name
----
_internal
ruche
> USE ruche
Using database ruche
> show measurements
name: measurements
name
----
ALARM
CO2
HUMIDITY
LIGHT
PRESSURE
TEMPERATURE
WEIGHT
>
```

Figure 5.12 Différentes mesures disponibles dans la base de données.

Une base de données InfluxDB est très compacte, elle ne doit pas comporter beaucoup de colonnes. Dans notre cas, on retrouve la source de nos données associées à un compteur, un exemple des données de température de notre capteur stockées dans la base de données est donné comme suit :

```
> select * from "TEMPERATURE"
name: TEMPERATURE
time                value
----                -----
1624818667847113138 25.98

1624818671400789936 25.95

1624818679859885217 25.96

1624818683414275556 25.91

1624818691873592504 25.99

1624818695427984770 25.9

1624818703885975208 25.98

1624818707452676224 25.91

1624818715899180463 25.98

1624818719455005386 25.93
```

Figure 5.13 Données de température mesurées stockées dans la base de données InfluxDB.

5.2.2 Avantages InfluxDB :

Par rapport aux bases de données relationnelles ordinaires, les bases de données TSDB comme InfluxDB offrent des avantages évidents en matière de rapidité pour l'enregistrement et le traitement des données horodatées. [51]
InfluxDB peut aussi maintenir des vitesses d'écriture élevées pendant une durée prolongée et rend les données facilement accessibles. [51]
Un autre point intéressant est la disponibilité de cette base de données sur l'outil Node-Red qui a facilité son utilisation.

5.3 Réalisation du dashboard avec Grafana :

L'outil Grafana est une multiplateformes open source sous licence GNU Affero General Public License Version 3. Créée par Torkel Ödegaard en janvier 2014, la plateforme permet la visualisation, la surveillance et l'analyse des données chronologiques. [55, 57]
Véritable éditeur de dashboards informatiques, Le logiciel est fourni avec un serveur web écrit en Go permettant d'y accéder avec une API HTTP. Grafana génère ses graphiques à partir de données récupérées des bases données de séries temporelles telle que InfluxDB ce qui le rend compatible à notre application. [56, 57]
Cet outil permet également le partage d'un tableau de bord actuel en créant un lien ou en créant un instantané statique de celui-ci et empêche les utilisateurs d'écraser accidentellement le dashboard par l'intégration d'un système de gestion des droits d'accès. [54]

Figure 5.14 Grafana logo.

Nous avons donc utilisé cet outil pour représenter et visualiser les données stockées sur notre base de données InfluxDB. Une fois connecter à la plateforme Grafana, nous avons configuré l'accès à notre base de données via les étapes suivantes :

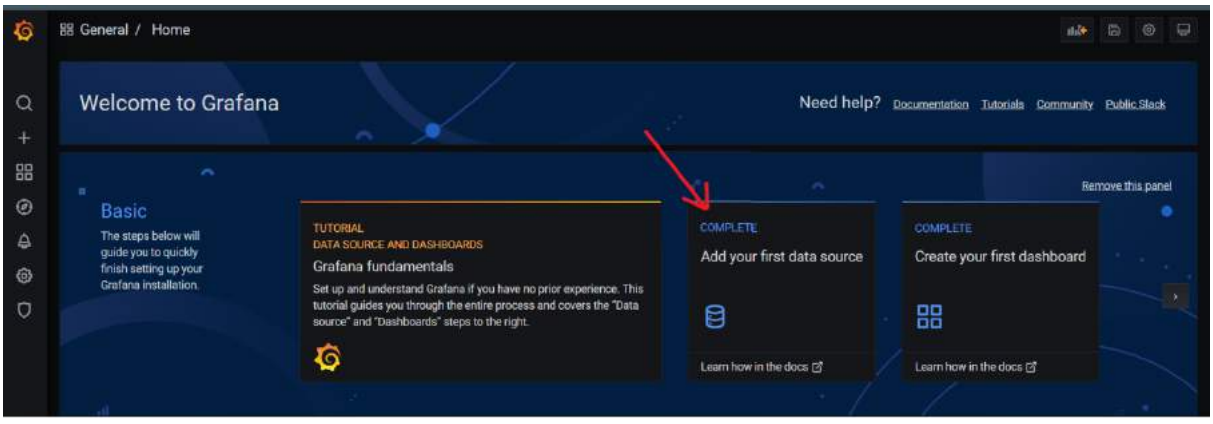

Figure 5.15 Ajout d'une base de données.

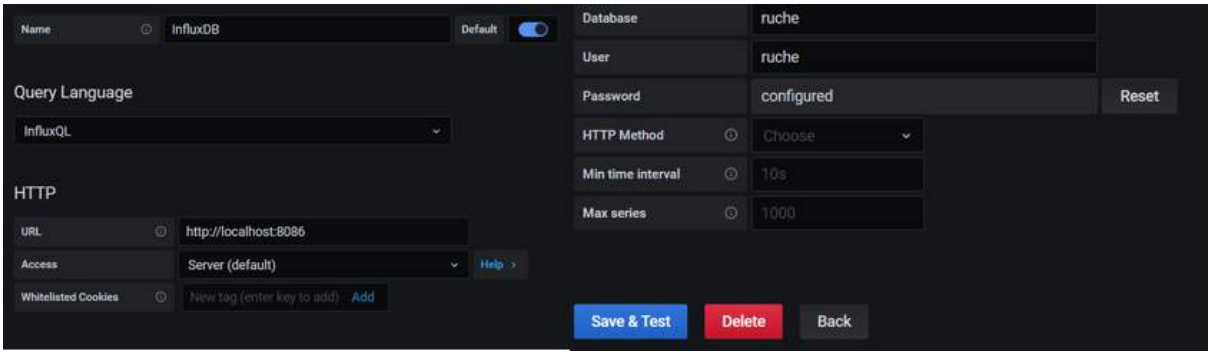

Figure 5.16 Paramétrage de la base de données.

Une fois cela fait, nous sommes passés à la création de notre dashboard en ajoutant des panels selon nos besoins. Le résultat final est montré ci-dessous :

Figure 5.17 Dashboard de l'apiculteur réalisé sous Grafana.

Chapitre 6 : Evaluation des performances du système et validation du prototype

6.1 Système final de la ruche connectée :

Pour finaliser notre système, une alimentation par une batterie 12V, rechargeable via des panneaux solaires est à prévoir.
Compte tenu du manque de matériel, nous avons directement utilisé un générateur 12V pour notre présentation.

Pour une meilleure présentation du travail, nous avons réalisé un PCB qui permet de rassembler les différents composants du système :

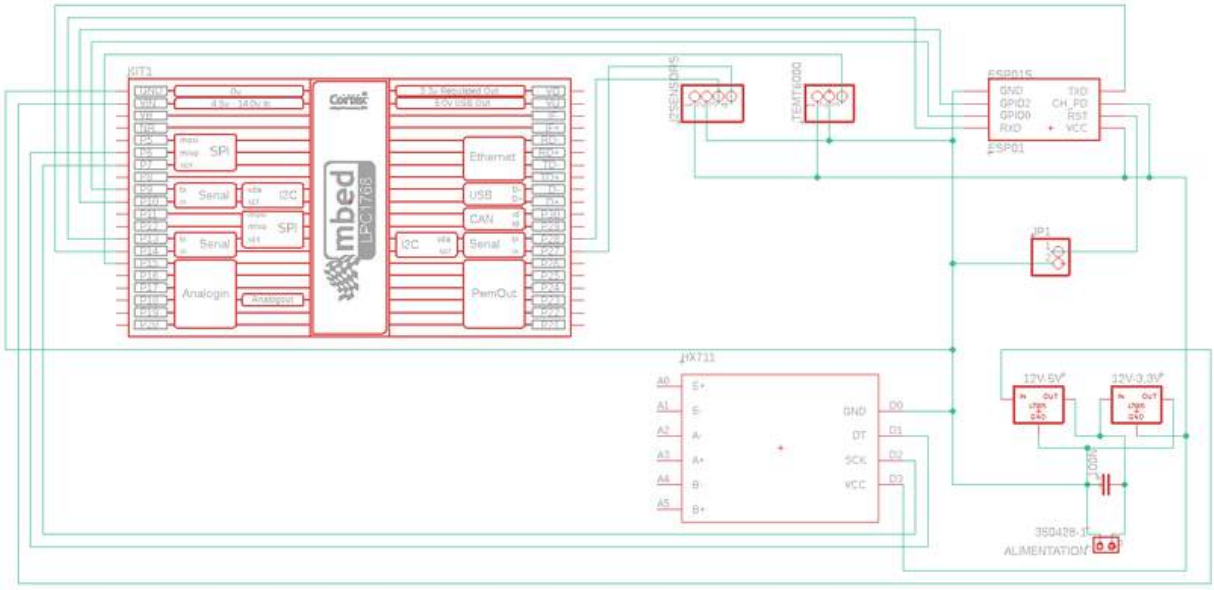

Figure 6.1 Circuit schématique pour la réalisation du PCB

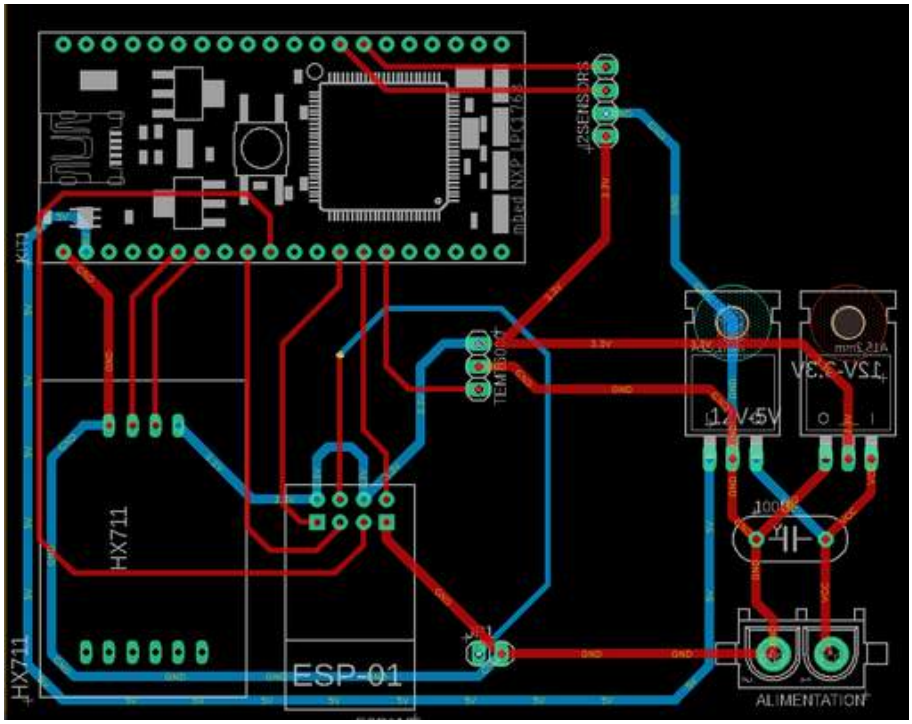

Figure 6.2 Circuit board pour la réalisation du PCB

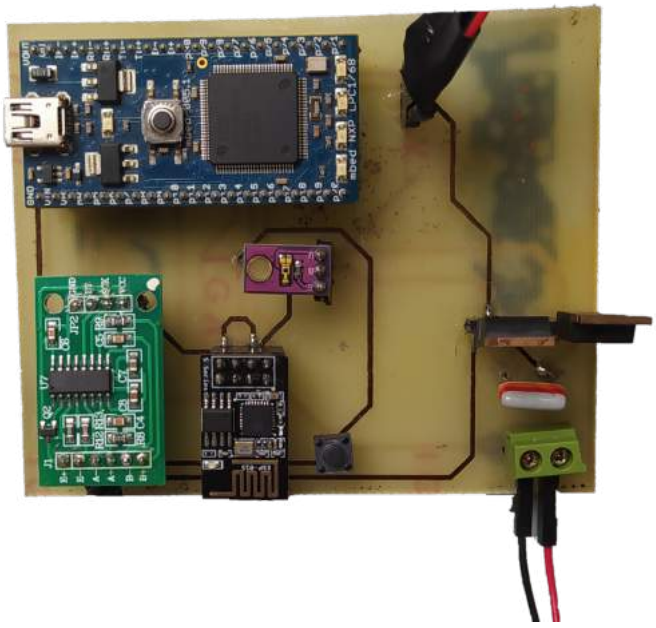

Figure 6.3 PCB final de la ruche connectée.

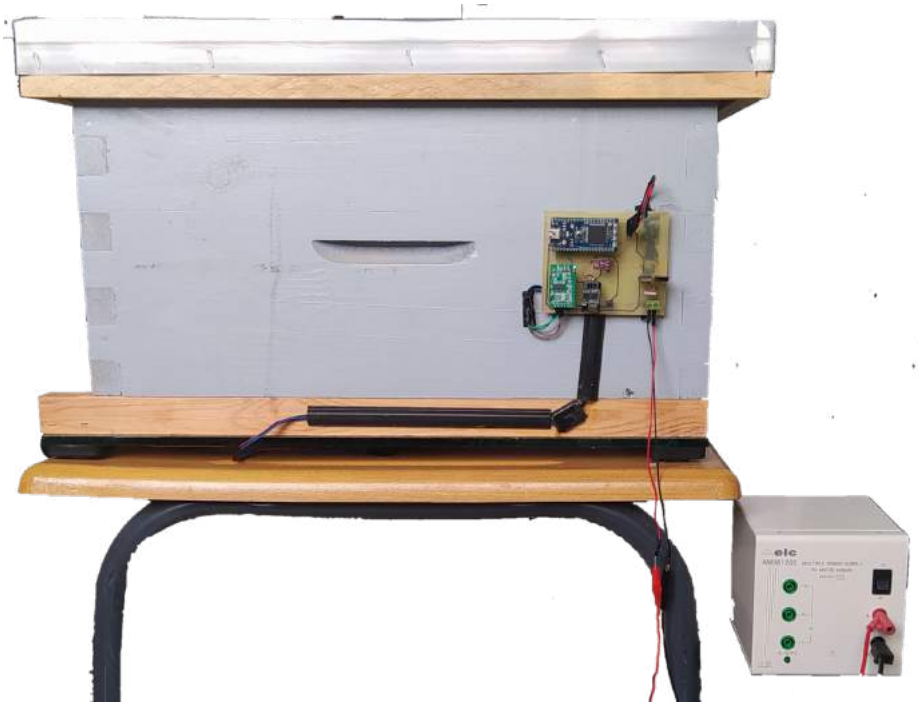

Figure 6.4 Système final de la ruche connectée.

6.2 Evaluation des performances du système :

Pour l'évaluation des performances de notre système, il est nécessaire de générer des variations des différents paramètres mesurés par notre réseau de capteurs et de s'assurer du bon transfert de ces données à travers les changements que présentent la visualisation du dashboard.

Pour ce faire, nous avons opéré une série de tests dont les résultats sont représentés par les figures suivantes :

Figure 6.5 Visualisation du dashboard avec système à vide.

Figure 6.6 Visualisation du dashboard suite à l'ajout d'un poids de 2 Kg.

Figure 6.7 Visualisation du dashboard suite à l'ajout d'un poids de 2 Kg et 200 g.

Figure 6.8 Visualisation du dashboard suite à l'ajout d'une source de chaleur.

Figure 6.9 Visualisation du dashboard suite à l'ajout d'une source d'humidité.

Figure 6.10 Visualisation du dashboard suite à l'ajout d'un produit polluant.

Figure 6.11 Visualisation du dashboard suite à l'ajout d'une source de lumière

Le système garantit donc des résultats adéquats à nos attentes. Les variations perçues correspondent parfaitement aux variations appliquées, leur précision reste néanmoins à confirmer par des étalons, ainsi notre système est bel et bien fonctionnel.

6.3 Fiabilité du système :

La fiabilité du système de la ruche connectée repose sur la stabilité de son fonctionnement assuré tout au long de la journée. Pour confirmer cela, nous avons mis en fonctionnement le système quotidiennement pour une durée moyenne de 8 heures en procédant à des vérifications de manière continuelle. Fort heureusement, aucun malheur n'est survenu, et le système a correctement répondu à nos attentes sans interruptions pendant plusieurs heures.

Conclusion générale et perspectives :

Le projet de ruche connectée se révèle être un atout majeur dans le domaine de l'apiculture future, ce secteur reposant jusqu'à présent sur des méthodes traditionnelles se voit révolutionné par l'apport de la technologie. Le système réalisé permet d'optimiser les efforts fournis par l'apiculteur et d'assurer un suivi minutieux de la colonie sans déplacement sur terrain. De plus, le système œuvre de manière indirecte à la protection et préservation de l'abeille indispensable à l'équilibre des écosystèmes.

Le déroulement de la réalisation du projet s'est fait sur plusieurs étapes. En premier lieu, la conception du réseau de capteurs et le traitement des données collectées par la carte de programmation LPC1768. Suite à quoi vint la conception de la chaîne de communication indispensable au transfert des données suivant un protocole MQTT à notre base de données. Et pour finir, la visualisation des paramètres associés à l'état de la ruche sur un tableau de bord configuré par nos soins et accessible à l'apiculteur.

Suite à la concrétisation de notre travail, plusieurs perspectives se sont présentées à nous dans le but de parfaire les résultats obtenus. Il serait intéressant de :

- ➢ Prévoir un autre réseau plus étendu pour la chaîne de communication comme le réseau LoRa.
- ➢ Ajout de capteurs supplémentaires pour plus de fonctionnalités comme un capteur infrarouge, une caméra de surveillance etc ... Et une meilleure exploitation des capteurs déjà disponibles par le développement des fonctions associées.
- ➢ La substitution des cartes de programmation utilisées par un module spécifique dédié à notre application, moins coûteux, réduit en taille et optimale.
- ➢ Étendre le système sur plusieurs ruches connectées gérées simultanément et indépendamment.
- ➢ Création d'une plateforme pour l'exploitation des données dans le domaine de la recherche.

Annexes

Annexe A : Code MBED LPC1768.

```
#include "mbed.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "HTU21D.h"
#include "MS5611I2C.h"
#include "HX711.h"
#include "CCS811.h"
#include "PowerControl/PowerControl.h"
   #include "PowerControl/EthernetPowerControl.h"

   // Pins declaration
   HTU21D htu21d(p28,p27); // SDA,SCL
   MS5611I2C ms5611(p28, p27, false);// SDA,SCL
   HX711 hx711(p6,p7,128);// Dt,SCK
   CCS811 ccs811(p28, p27);
   AnalogIn temt6000(p15);
   Serial esp_data(p13, p14); // tx, rx
   Serial esp_alarm(p9, p10);
   Serial pc(USBTX, USBRX); // tx, rx

   // Funcrions declaration
   void ESPcmd_interrupt();

   // Variables declaration
   int serialArrived = false, ESP_checked = false, valid =
false,temp,hum,number = 10;
   uint16_t eco2, tvoc;
   char ESP_cmd = 0, espcmd, data_string[100],end[]="\n";
   float weight,data, state, scale = 11.524;
   Timer times;

   //Interruption function
   void ESPcmd_interrupt(){ // Checks if cmd received from esp then
reads it

       char tmp=esp_data.getc();
       if ((tmp != '\n') && (tmp != '\r') &&((tmp == 'a') || (tmp
== 'b') || (tmp == 'c') ||(tmp == 'd') || (tmp == 'e') || (tmp ==
'f')))  {
```

```
37.             if(!ESP_checked) {
38.                 ESP_cmd = tmp;
39.                 ESP_checked = true;
40.             }
41.         }
42.     }
43.
44.     int main()
45.     {
46.         // add configuration
47.         PHY_PowerDown();
48.         ccs811.init();
49.         hx711.tare(number);
50.         hx711.setScale(scale);
51.
52.         // Launch interruption
53.         esp_data.attach(ESPcmd_interrupt, Serial::RxIrq); //
    Interruption routine set
54.         times.start();
55.
56.         while(1) {
57.         // Alarms part
58.             temp=float(htu21d.sample_ctemp())/100.0;
59.             hum=float(htu21d.sample_ctemp())/100.0;
60.             if (((temp<33) || (temp>36) || (hum<50) || (hum>70) ||
    (state > 5000)) && (times>10)){
61.                 esp_alarm.puts("A");
62.                 times.reset();
63.             }
64.
65.         // Treating ESP cmd
66.             if(ESP_checked){
67.                 pc.printf("treating the command \n");
68.                 ESP_checked = false;
69.                 switch(ESP_cmd){
70.                     case 'a': // temp reading
71.                         data = float(htu21d.sample_ctemp())/100.0;
72.                     break;
73.                     case 'b' : // humidity reading
74.                         data = float(htu21d.sample_humid())/100.0;
75.                     break;
76.                     case 'c': // pressure reading
77.                         data = (ms5611.getPressure())/100000;
```

```
                    break;
                    case 'd': // light reading
                        float x=temt6000;
                        data = x;
                    break;
                    case 'e': // CO2 reading
                        ccs811.readData(&eco2, &tvoc);
                        data = eco2;
                    break;
                    case 'f': // weight reading
                        data = hx711.getGram();
                        state = abs(data - weight);
                        weight = data;
                        if(data<0){data=0;}
                    break;
                 }
                sprintf(data_string, "%g", data);// convert data to string before writing it to ESP
                strcat(data_string,end);
                pc.printf("data : %s\n",data_string);
                esp_data.puts(data_string);
                wait_ms(500);
            }
        }
    }
```

Annexe B : Code ESP8266 01s.

```
// Libraries declaration
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Rx/Tx esp with adapt + Rx/Tx esp data with mbed (13,14) + 0/2 esp alarm with mbed (9,10)
SoftwareSerial serialalarm(0, 2);// Connect GPIO 0 & 2 with pins (9,10) of mbed for alarm

// Variables declaration
  WiFiClient espClient;
  PubSubClient client(espClient);
  const byte GPIO_pin = 0;
```

```
13.   char cmd[6] = {'a','b','c','d','e','f'}; // Commands to get data
      from mbed
14.   char msgmqtt[50];
15.   String data; // Received data from mbed
16.   volatile int detect_alarm = false;
17.   int cmp; // Counter to check if alarm is still up
18.   char* mqtt_user ="ruche";
19.   char* mqtt_pw ="ruche";
20.
21.   // Functions
22.   void LPC_interrupt_alarm(){ //Interruption function for alarm
23.     cmp = 0;
24.     detect_alarm = true;
25.
26.   }
27.
28.   void reconnectmqttserver() { // Connecting esp as an mqtt client
29.     while (!client.connected()) {
30.       Serial.print("Attempting MQTT connection...");
31.       String clientId = "ESP8266Client-";
32.        clientId += String(random(0xffff), HEX);
33.       if (client.connect(clientId.c_str(), mqtt_user, mqtt_pw)) {
34.         Serial.println("connected");
35.       } else {
36.         Serial.print("failed, rc=");
37.         Serial.print(client.state());
38.         Serial.println(" try again in 5 seconds");
39.         delay(5000);
40.       }
41.     }
42.   }
43.
44.   void callback(char* topic, byte* payload, unsigned int length) {
      // To callback the mqtt server in case of : lost connection,
      delivery complete, message arrived
45.     String MQTT_DATA = "";
46.     for (int i=0;i<length;i++) {
47.      MQTT_DATA += (char)payload[i];}
48.
49.   }
50.
51.   // Program
52.   void setup(){
```

```
53.     Serial.begin(9600);
54.     serialalarm.begin(9600);
55.     WiFi.disconnect();
56.     Serial.print("Start\n");
57.     // Connecting to wifi
58.     WiFi.begin("Ooredoo 4G_4744D7","20152015"); // Connecting esp
   to wifi network
59.     while ((!(WiFi.status() == WL_CONNECTED))){
60.       delay(300);
61.       Serial.print("\n trying again");
62.     }
63.     Serial.println("Connected");
64.     Serial.println("Your IP is");
65.     Serial.println((WiFi.localIP().toString()));
66.     client.setServer("192.168.0.160", 1883); //
67.     client.setCallback(callback);
68.     // Attaching interruption
69.     pinMode(GPIO_pin, INPUT_PULLUP); // Set interrupt pin as input
70.     attachInterrupt(digitalPinToInterrupt(GPIO_pin),
   LPC_interrupt_alarm, RISING); // Launch interruption from LPC
71.   }
72.
73.   void loop(){
74.   // Reconnecting client to MQTT broker
75.   if (!client.connected()) {
76.       reconnectmqttserver();
77.       }
78.
79.   // Sending commands and receiving data part
80.   if (!detect_alarm){
81.     for (int i=0;i<6;i++){ // for loop takes 9012 ms
82.       //delay(50);
83.       Serial.write(cmd[i]);
84.       if(i==5){
85.         delay(3500);
86.        }
87.       else{
88.         delay(200);
89.       }
90.         while (Serial.available()>0){
91.           data = Serial.readString();
92.           Serial.print(data);
93.           // Sending data to MQTT broker
```

```
94.           data.toCharArray(msgmqtt, data.length() + 1);
95.           switch (i){
96.             case 0 :
97.               client.publish("TEMPERATURE", msgmqtt);
98.               break;
99.             case 1 :
100.              client.publish("HUMIDITY", msgmqtt);
101.              break;
102.            case 2 :
103.              client.publish("PRESSURE", msgmqtt);
104.              break;
105.            case 3 :
106.              client.publish("LIGHT", msgmqtt);
107.              break;
108.            case 4 :
109.              client.publish("CO2", msgmqtt);
110.              break;
111.            case 5 :
112.              client.publish("WEIGHT", msgmqtt);
113.              break;
114.          }
115.        }
116.    }
117.    cmp++;
118.    if(cmp==3){
119.     Serial.print("Alarm down\n");
120.     client.publish("ALARM", "0");
121.    }
122. }
123.
124. // Treating alarm interruption part
125. else{
126.    Serial.print("Alarm received\n");
127.    client.publish("ALARM", "1");
128.    detect_alarm = false;
129. }
130. }
```

Annexe C : Détails sur le capteur de température et d'humidité l'HTU21D.

Spécification de l'interface :

La tension d'alimentation du capteur HTU21D doit être de 1.5V à 3.6V en DC, Cependant, le circuit d'application comprend une résistance de pull up de 10kΩ et un condensateur de découplage de 100 nF entre le pin VDD et GND, placé le plus près possible du capteur.

Le pin SCK est utilisé pour la synchronisation de la communication entre le microcontrôleur et le capteur HTU21D(F). Le pin SDA est utilisé pour le transfert de données bidirectionnelle de et vers le capteur, le pin SDA est valide sur le front montant du pin SCK et doit rester stable tant que le pin SCK est à l'état haut. Au front descendant du pin SCK, la valeur du pin SDA peut être modifiée. Pour une communication sécurisée, les données seront valable respectivement à t_{su} et t_{hu} avant le front montant et après le front descendant du pin SCK. Une résistance de pull-up externe (par exemple 10kΩ) sur le pin SCL et SDA est nécessaire pour tirer le signal vers le haut uniquement pour un collecteur ouvert ou microcontrôleurs à technologie à drain ouvert.

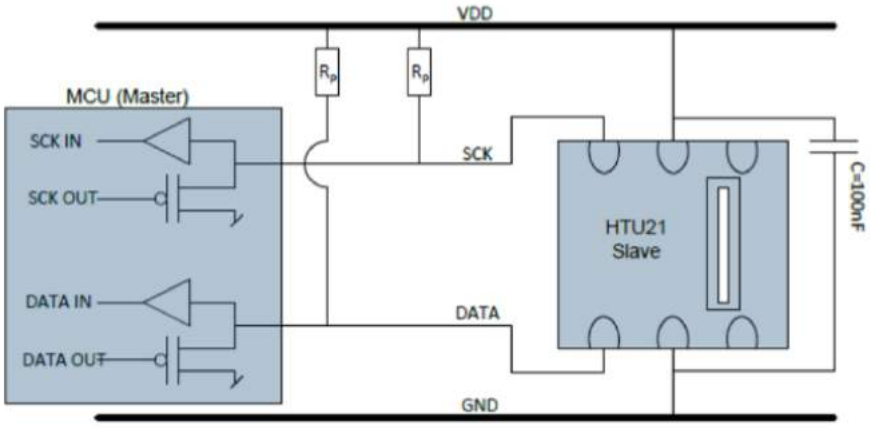

Figure C.1 Circuit d'application typique du HTU21D.

Informations complémentaires :

- Alimentation : idéalement 3V et un maximum de 3.6V.
- Résistance de pullup : 10kΩ.
- Précision de température : ± 0.3 °C.
- Plage de température : -40 à 125 °C.
- Précision d'humidité : ± 2%.
- Plage de température : 0 à 100%.

Annexe D : Détails sur le capteur de pression le MS5611.

Principe de communication avec la carte microcontrôleur :

Le microcontrôleur externe synchronise les données reçues par le composant via l'entrée SCLK (Serial CLocK) et SDA (Serial DAta). Le capteur répond sur la même broche SDA qui est bidirectionnelle pour l'interface de bus I2C. Ainsi, ce type d'interface n'utilise que 2 lignes de signal et ne nécessite pas de sélection de puce, ce qui peut être favorable pour réduire l'espace sur la carte.

La carte mbed communique avec le capteur MS5611 suivant l'adresse 0111011C où le C correspond à la valeur du pin CSB du capteur.

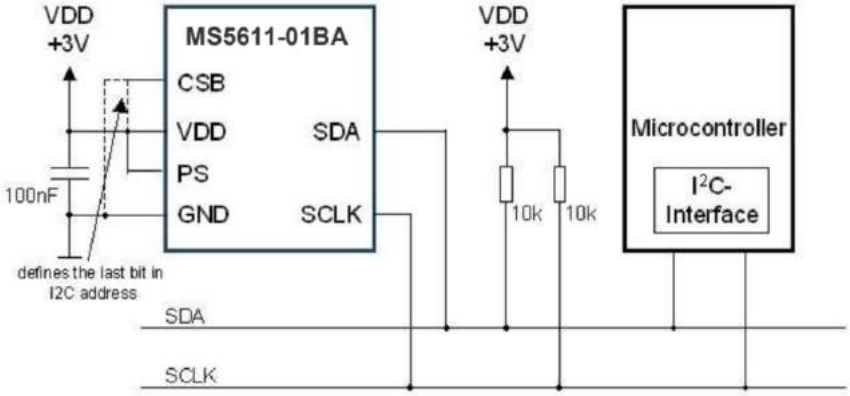

Figure D.1 Circuit d'application typique du MS5611 pour une communication I2C.

Informations complémentaires :

- Alimentation : idéalement 3V et un maximum de 3.6V.
- Résistance de pullup : 10kOhm.
- Précision de pression : ± 1.5mbar.
- Plage de pression : 10 à 1200 mbar.
- Précision de température : ± 0.8 °C.
- Plage de température : -40 à 85 °C.

Annexe E : Détails sur la carte microcontrôleur MBED NXP LPC1768.

Le microcontrôleur est basé sur le NXP LPC1768, avec un cœur ARM Cortex-M3 32 bits fonctionnant à 96 MHz. Il comprend 512 Ko FLASH, 32 Ko de RAM et de nombreuses interfaces, y compris Ethernet intégré, hôte et périphérique USB, CAN, SPI, I2C, ADC, DAC, PWM et d'autres interfaces d'E/S.

Ci-joint les pins description de la carte qui montre l'emplacement des différentes interfaces :

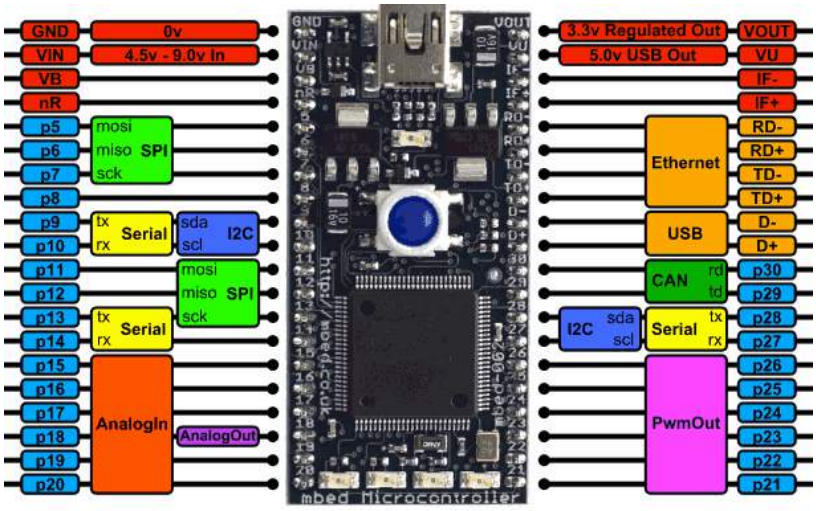

Figure E.1 Pins description MBED NXP LPC1768.

Caractéristiques de la carte :

- NXP LPC1768 MCU
- Haute performance ARM® Cortex™-M3 Core
- 96MHz, 32KB RAM, 512KB FLASH
- Ethernet, USB Host/Device, 2xSPI, 2xI2C, 3xUART, CAN, 6xPWM, 6xADC, GPIO
- Prototyping form-factor
- 40-pin 0.1" pitch DIP package, 54x26mm
- 5V USB or 4.5-9V supply
- Built-in USB drag 'n' drop FLASH programmer
- mbed.org Developer Website
- Lightweight Online Compiler
- High level C/C++ SDK

La carte de développement mbed NXP LPC1768 permet donc de concevoir rapidement des programmes sur son puissant processeur 32-bit ARM Cortex-M3. La programmation est grandement facilitée par la collection de librairies et de codes d'exemples disponibles ainsi que par les supports et ressources partagés dans la communauté MBED. C'est pourquoi le choix de microcontrôleur pour notre système s'est porté sur cette carte car elle est pratique et valide les besoins de notre projet.

Annexe F : Détails sur le protocole de communication MQTT.

Le protocole MQTT a été inventé en 1999 par Andy Stanford Clark (IBM) et Arlen Nipper (Arcom, maintenant Cirrus Link). Ils eurent besoin d'un protocole nécessitant une faible consommation électrique et dont la bande passante est minimale pour se connecter aux oléoducs par satellite. Les deux inventeurs se devaient de respecter un certain nombre d' exigences spécifiques au protocole :

- Une implémentation simple.
- La qualité du service de transmission des données.
- Léger et une bande passante optimale.
- Agnostique des données.

Ces objectifs font encore partie du corp du protocole MQTT. Néanmoins, l'intérêt principal est passé des systèmes embarqués à échelle propriétaire à une utilisation open source pour le domaine de l'internet des objets (IoT).

Le protocole MQTT applique un filtrage par sujet des messages. Chaque message est affilié à un topic (sujet) que le broker utilise pour déterminer à quel client souscripteur le message devra ou non être envoyé.
Pour relever le défis des systèmes pub/sub, MQTT a des niveaux de qualité de service (QoS : Quality of Service). Il est de ce fait possible de vérifier si un message a bien été transmis du client au broker et vice versa.
Il se peut qu'il n'y ait pas de client souscripteurs à un topic, le broker MQTT devra donc avoir un système de plugins, qui peut identifier ces éventualités. Dans ce cas, le broker peut soit prendre des mesures à définir ou simplement stocker les données dans une base de données pour des analyses historiques. Pour conserver la flexibilité de l'arborescence hiérarchique des topics, il est important de concevoir cette dernière très soigneusement et de laisser de la place pour de futurs cas d'utilisation.

Clients et connections MQTT :

Les clients sont, et les Publishers et les Subscribers. Le label Publisher et Subscriber indique si le client publie ou souscrit à des messages (les fonctions de publication et de souscription peuvent également être implémentées sur un même client MQTT).
Un client MQTT est n'importe quel appareil (d'un microcontrôleur à un serveur à part entière) qui exécute une bibliothèque MQTT et se connecte à un broker MQTT sur un réseau.
De manière générale, tout périphérique qui parle MQTT sur une pile TCP/IP peut être défini comme un client MQTT. L'implémentation au niveau client du protocole MQTT est très simple et rationalisée. La facilité de mise en œuvre est l'une des raisons pour lesquelles le protocole MQTT est parfaitement adapté aux petits appareils. Les bibliothèques pour clients MQTT sont

disponibles pour une grande variété de langages de programmation. Par exemple, Android, Arduino, C, C++, C#, Go, iOS, Java, JavaScript et .NET.
Le protocole MQTT est basé sur une transmission TCP/IP. Le client et le broker doivent tous deux avoir une pile TCP/IP.
La connexion MQTT est toujours entre un client et le broker, les clients ne se connectent jamais entre eux directement.
Pour initier une connexion, le client envoie un message de connexion au broker “CONNECT”. Le broker répond avec un message “CONNACK”. Une fois la connexion établie, le broker la garde ouverte jusqu'à ce que le client envoie une commande de déconnexion ou que la connexion s'interrompt.
Si le message de connexion est mal formé ou trop de temps s’écoule entre l’ouverture du réseau de communication et l’envoi de la commande de connexion, alors le broker ferme la connexion.

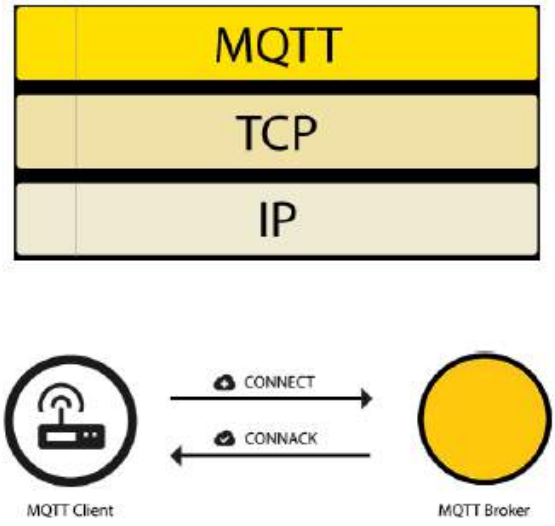

Figure F.1 initiation de la communication entre un client et le broker.

Un message de connexion correctement formé est constitué comme suit :

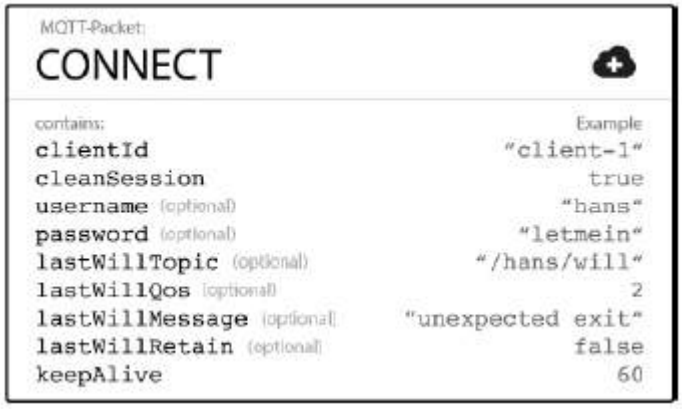

Figure F.2 contenu du message CONNECT.

- “ClientID” : l’identifiant du client identifie chaque client MQTT se connectant au broker MQTT. Cet identifiant est unique pour chaque client par broker.
- “Clean Session” : Ce signal d’état indique au broker si le client souhaite établir une session persistante ou non. Dans une session persistante (CleanSession = false), le broker enregistre toutes les souscriptions du client et tous les messages manqués par client qui a souscrit avec un QoS de niveau 1 ou 2. Dans une session non persistante (CleanSession = true), rien de cela n’est enregistré par le broker et ce dernier purge toutes les informations de toute session persistante précédente.
- “Username/Password” : MQTT peut envoyer un nom d’utilisateur et un mot de passe pour l’authentification du client.
- “Last Will and Testament (LWT)” : Last Will and Testament est utilisé pour informer les abonnés d'une rupture attendue de la connexion avec le Publisher.
- “KeepAlive” : le KeepAlive est une période définie en seconde que le client détermine et communique au broker une fois que la connexion entre ces derniers est établie. Cet intervalle spécifie la période la plus longue que peut endurer le broker et le client sans s’envoyer de messages.

Lorsque le broker reçoit un message de connexion CONNECT, il est indispensable qu’il y réponde avec un message de réponse CONNACK, celui-ci comprend deux données :

- SessionPresent : indique au client si le broker a une session persistante disponible à partir d’interactions précédentes avec le client. Ce signal d’état est mis à “false” si le client ne s’est pas connecté avec une session persistante, ou si le broker n’a aucune information enregistrée pour lui. Il est mis à “true” si le client s’est connecté avec une session persistante et que le broker a des informations disponibles pour le client.
- returnCode : indique au client si sa tentative de connexion est validée.

Return Code	Return Code Response
0	Connection accepted
1	Connection refused, unacceptable protocol version
2	Connection refused, identifier rejected
3	Connection refused, server unavailable
4	Connection refused, bad user name or password
5	Connection refused, not authorized

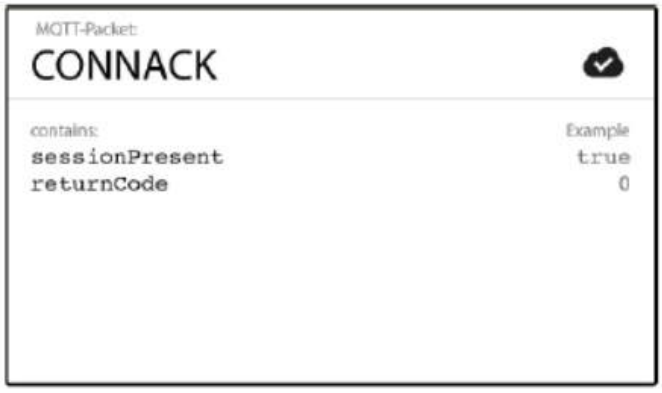

Figure F.3 contenu du message CONNACK et les valeurs du returnCode possibles.

Fonctions de publications et souscriptions (Publish / Subscribe) :

Publish :

Un client MQTT peut publier un message du moment qu’il établit une connexion au broker. Le protocole MQTT utilise un filtrage par sujet des messages, chaque message doit contenir un

topic dont le broker se sert pour transmettre le message aux clients intéressés. Typiquement, chaque message a un payload qui contient les données à transmettre en octet.
Un message publié via le protocole MQTT sera constitué des attributs suivants :

- “Topic Name” : une chaîne de caractères représentant le nom du Topic.
- “QoS” : Ce nombre indique le niveau de qualité de service du message. Il existe trois niveaux : 0, 1 et 2. Il détermine quel type de garantie a un message pour atteindre un destinataire (broker ou client).
- “Retain Flag” : Cet indicateur définit si le message est enregistré par le broker en tant que dernière bonne valeur connue pour un topic spécifié. Lorsqu'un nouveau client s'abonne à un topic, il reçoit le dernier message qui est retenu sur ce sujet.
- “Payload” : contenu actuel du message, le protocole MQTT permet l’envoi de divers types de données (images, textes, …).
- “ paquet Identifier” : L'identifiant de paquet identifie de manière unique un message lorsqu'il circule entre le client et le broker. L'identifiant de paquet n'est pertinent que pour les niveaux de QoS supérieurs à zéro.
- “DUP Flag” : L'indicateur indique que le message est une duplication et qu'il a été renvoyé car le destinataire prévu (client ou broker) n'a pas accusé réception du message d'origine. Ceci n'est pertinent que pour une QoS supérieure à 0.

Le Publisher n’est concerné que pas l’envoi du message au broker, il est de la responsabilité de ce dernier de parvenir le message aux Subscribers.

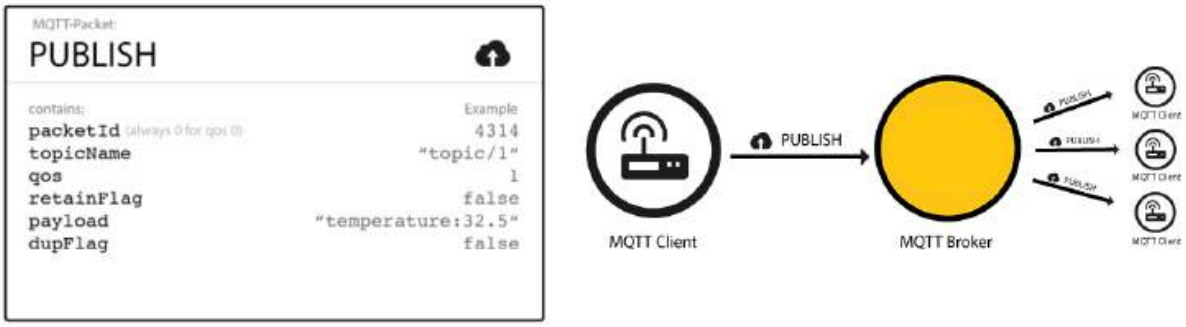

Figure F.4 Fonction Publish.

Subscribe :

La réception d’un message d’un topic spécifique requiert l’envoi d’un message de souscription (Subscribe) au broker MQTT. Ce message est très simple et ne contient qu’un identifiant de paquet et la liste des souscriptions (Noter que pour souscrire à tous les topics existants il suffit d’utiliser le “#” dans la liste des souscriptions).
Pour confirmer chaque souscription du client, le broker lui envoie un message d’accusé de réception appelé SUBACK. Ce message contient l'identifiant de paquet du message original de souscription et une liste de Return Code.
Le Return Code prend la valeur :

- 0 : souscription réussie, QoS maximal 0
- 1 : souscription réussie, QoS maximal 1
- 2 : souscription réussie, QoS maximal 2
- 128 : erreur.

Une fois qu'un client a réussi à envoyer le message SUBSCRIBE et à recevoir le message SUBACK, il obtient chaque message publié qui correspond à un topic de la souscription contenus dans le message SUBSCRIBE.

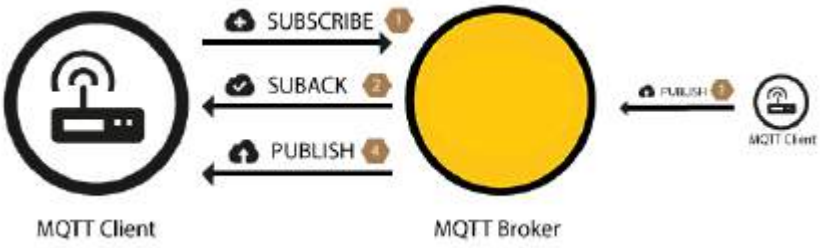

Figure F.5 Fonction Subscribe.

Niveau de Qualité de Service (QoS) :

Le niveau du QoS est un accord sur la garantie de livraison d'un message entre l'expéditeur et son destinataire. Il existe 3 niveaux de QoS dans MQTT :

- Au plus une fois (0) : c'est le niveau minimal, il ne garantit pas la réception du message par le destinataire.
- Au moins une fois (1) : ce niveau garantit la réception du message au moins une fois par le destinataire.
- Exactement une fois (2) : ce niveau garantit la réception du message seulement une fois par le destinataire.

Bibliographie

[1] Picbleu. Le rôle des abeilles : un maillon indispensable. [consulté le 21/04/2021]. Disponible sur :
https://www.picbleu.fr/page/role-des-abeilles-maillon-indispensable

[2] Philippe WEIBEL. Le rôle de l'abeille dans la biodiversité et dans l'agriculture. [consulté le 21/04/2021]. Disponible sur :
https://www.miel-lerucherdelours.fr/fr/content/107-un-pollinisateur-exceptionnel

[3] Patrick Straub, Futura. L'abeille, sentinelle écologique. [consulté le 21/04/2021]. Disponible sur :
https://www.futura-sciences.com/planete/dossiers/zoologie-abeille-sentinelle-ecologique-684/

[4] 2021 Weyn's Honingbedrijf. Importance de l'abeille. [consulté le 21/04/2021]. Disponible sur :
https://weynshoning.be/fr/abeilles-et-miel/importance-de-l-abeille/

[5] Patrick Straub, Futura. Importance de l'apiculture. [consulté le 21/04/2021]. Disponible sur :
https://www.futura-sciences.com/planete/dossiers/zoologie-abeille-sentinelle-ecologique-684/page/8/

[6] ICKO Apiculture. CONNAISSEZ-VOUS L'APICULTEUR? [consulté le 21/04/2021]. Disponible sur :
https://www.icko-apiculture.com/parcours-devenir-apiculteur#:~:text=Le%20r%C3%B4le%20de%20l%27apiculteur&text=il%20s%27occupe%20de%20la,la%20survie%20de%20la%20colonie

[7] Miel FACTORY. L'habitat des abeilles : la ruche. [consulté le 21/04/2021]. Disponible sur :
https://www.miel-factory.com/blogs/blog/lhabitat-des-abeilles-la-ruche

[8] Miels d'Anicet. L'organisation physique de la ruche, Miels d'Anicet, 16 mai 2018. [consulté le 25/04/2021]. Disponible sur :
https://mielsdanicet.com/fr-ca/technique-apicole-organisation-physique-de-la-ruche/

[9] Union nationale de l'apiculture française. La vie dans la ruche. [consulté le 25/04/2021]. Disponible sur :
https://www.abeillesentinelle.net/les-abeilles/la-vie-dans-la-ruche

[10] Le Musée de l'agriculture et de l'alimentation du Canada. La structure de la ruche. [consulté le 25/04/2021]. Disponible sur :
https://bees.techno-science.ca/francais/les-abeilles/la-ruche-et-la-colonie/structure.php#:~:text=G%C3%A9n%C3%A9ralement%2C%20une%20ruche%20manufactur%C3%A9e%20se,ouvri%C3%A8res%20s'occupent%20des%20larves

[11] apiculture.cornille@orange.fr. Découverte de la ruche. [consulté le 25/04/2021]. Disponible sur :
https://apiculture-familiale.pagesperso-orange.fr/nouvellepage2.htm

[12] Agnès Fayet. Les abeilles ont-elles trop chaud? [consulté le 25/04/2021]. Disponible sur : https://butine.info/les-abeilles-ont-elles-trop-chaud/

[13] Patrick Olivier. Un rucher au jardin. [consulté le 30/04/2021]. Disponible sur : https://unrucheraujardin.blogspot.com/2016/10/la-temperature-dans-la-ruche_21.html

[14] Label abeille. LUMINOSITÉ & APICULTURE. [consulté le 30/04/2021]. Disponible sur : https://www.label-abeille.org/fr/blog/294-luminosite-apiculture

[15] A. Deyme, G.I. Begue-Deyme. DÉTERMINATION DE LA DOSE NARCOTIQUE DE GAZ CARBONIQUE (CO2) EN FONCTION DE L'ÂGE CHEZ L'OUVRIÈRE D'ABEILLE (APIS MELLIFICA L.1). Apidologie, Springer Verlag, 1977, 8 (3), pp.217-228. ffhal-00890430f. [consulté le 30/04/2021]. Disponible sur : https://hal.archives-ouvertes.fr/hal-00890430/document

[16] Suivi de la masse des ruches : Intérêts pour l'apiculteur et pour le vétérinaire clinicien Clinique Vétérinaire des Mazets, 15400 RIOM-ES-MONTAGNES Membre de la commission apicole SNGTV, roille@orange.fr. [consulté le 02/05/2021]. Disponible sur : https://www.researchgate.net/publication/337821147_Suivi_de_la_masse_des_ruches_interets_pour_l'apiculteur_et_pour_le_veterinaire_clinicien

[17] Famille Mary, TRÉSORS DE LA RUCHE. Le travail apicole au fil de l'année. [consulté le 02/05/2021]. Disponible sur : https://www.famillemary.fr/blog/post/le-travail-apicole-au-fil-de-lannee

[18] Université Haute-Alsace. Projet ruche connectée. [consulté le 03/05/2021]. Disponible sur : http://www.projetsgeii.iutmulhouse.uha.fr/ruche-connectee/

[19] Aggregator. The connected Beehive. [consulté le 05/05/2021]. Disponible sur : http://touchardinforeseau.servehttp.com/Ruche/

[20] Label abeille. LA RUCHE CONNECTÉE. [consulté le 05/05/2021]. Disponible sur : https://www.label-abeille.org/fr/content/6-produit

[21] BeeGuard. Barres de mesure de poids et de météo pour vos ruches : WGuard. [consulté le 07/05/2021]. Disponible sur : https://www.siconsult.fr/index.php/s/wguard

[22] Wikipedia. Load cell. [consulté le 12/03/2021]. Disponible sur : https://en.wikipedia.org/wiki/Load_cell

[23] Indrek luuk. 50kg Load Cells with HX711 and Arduino. 4x, 2x, 1x Diagrams. [consulté le 12/03/2021]. Disponible sur : https://circuitjournal.com/50kg-load-cells-with-HX711

[24] COMPONENTS101. HX711–24 Bit Analog to Digital Converter (ADC). [consulté le 13/03/2021]. Disponible sur :

https://components101.com/ics/hx711-24-bit-analog-digital-converter-adc

[25] AMSYS GmbH & Co. KG. HTU21D - humidity and temperature sensor - assembly and applications. [consulté le 18/03/2021]. Disponible sur : https://www.amsys-sensor.com/downloads/notes/htu21d-humidity-and-temperature-sensor-assembly-and-applications-amsys-fan-004.pdf

[26] Baumer. Le fonctionnement et la technologie des capteurs de pression. [consulté le 14/03/2021]. Disponible sur : https://www.baumer.com/fr/fr/service-assistance/fonctionnement/le-fonctionnement-et-la-technologie-des-capteurs-de-pression/a/Know-how_Function_Pressure-sensors

[27] Digi-KEY electronics. Capteur de pression atmosphérique MS5611-01B. [consulté le 14/03/2021]. Disponible sur : https://www.digikey.fr/fr/product-highlight/t/te-connectivity-measurement-specialties/ms5611-01ba-barometric-pressure-sensor

[28] Domotique et objets connectés. débuter avec bus I2C. [consulté le 18/03/2021]. Disponible sur : https://projetsdiy.fr/arduino-debuter-bus-i2c-esp32-esp8266/#brochesi2c

[29] Malik BOUKHALFA, Yanis CHABANE. Utilisation d'un capteur de température et de lumière pour un émulateur photovoltaïque. Electronique industrielle. UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU, 2018. [consulté le 18/03/2021]. Disponible sur : https://dl.ummto.dz/bitstream/handle/ummto/6718/BoukhalfaMalik_ChabaneYanis.pdf?sequence=1&isAllowed=y

[30] Illustrationprize. Phototransistor. [consulté le 21/03/2021]. Disponible sur : https://illustrationprize.com/fr/304-phototransistor.html

[31] Netatmo. Capteur de qualité d'air. [consulté le 10/05/2021]. Disponible sur : https://www.netatmo.com/fr-be/guides/weather/air-care/pollution/air-quality-sensor

[32] AVIA SEMICONDUCTOR. 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales. [consulté le 25/02/2021 - 25/06/2021]. Disponible sur : https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

[33] measurement specialties. Digital Relative Humidity sensor with Temperature output. [consulté le 25/02/2021 - 25/06/2021]. Disponible sur : https://cdn-shop.adafruit.com/datasheets/1899_HTU21D.pdf

[34] TE connectivity. MS5611-01BA03. [consulté le 25/02/2021 - 25/06/2021]. Disponible sur : https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS5611-01BA03%7FB3%7Fpdf%7FEnglish%7FENG_DS_MS5611-01BA03_B3.pdf%7FCAT-BLPS0036

[35] Vishay Semiconductors. Ambient Light Sensor. [consulté le 25/02/2021 - 25/06/2021]. Disponible sur : https://www.vishay.com/docs/81579/temt6000.pdf

[36] ams Datasheet. CCS811 Ultra-Low Power Digital Gas Sensor for Monitoring Indoor Air Quality. [consulté le 25/02/2021 - 25/06/2021]. Disponible sur : https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf

[37] arm MBED. MBED LPC1768 [consulté le 25/02/2021 - 25/06/2021]. Disponible sur : https://os.mbed.com/platforms/mbed-LPC1768/

[38] developpez.com, f-leb, Claude Leloup, François Dorin. Module WiFi ESP8266. [consulté le 15/05/2021]. Disponible sur : https://f-leb.developpez.com/tutoriels/arduino/esp8266/debuter/

[39] Blogger. Les bases pour utiliser l'ESP01: réalisation d'un contrôle d'état des volets. [consulté le 15/05/2021]. Disponible sur : http://framboiseaupotager.blogspot.com/2018/05/tous-les-secrets-de-lesp01-realisation.html

[40] The HiveMQ Team. MQTT essentials ebook. Publié le 20 Octobre 2020. [consulté le 17/05/2021 - 10/06/2021]. Disponible sur : https://www.hivemq.com/blog/announcing-mqtt-ebook/

[41] DomoPi, Guillaume. MQTT, un protocole pour tous les rassembler - Partie 1 : le broker. [consulté le 17/05/2021]. Disponible sur : https://domopi.eu/mqtt-un-protocole-pour-tous-les-rassembler-partie-1-le-broker/

[42] JDN. MQTT : comment fonctionne ce protocole ? [consulté le 18/05/2021]. Disponible sur : https://www.journaldunet.fr/web-tech/dictionnaire-de-l-iot/1440686-mqtt-comment-fonctionne-ce-protocole/

[43] Eclipse Mosquitto. [consulté le 20/05/2021]. Disponible sur : https://mosquitto.org/

[44] ESP8266WiFi library. [consulté le 30/05/2021]. Disponible sur : https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

[45] Arduino Client for MQTT. [consulté le 30/05/2021]. Disponible sur : https://pubsubclient.knolleary.net/api

[46] Olga Weis. Protocoles de communication. Principaux types de transfert de données série. [consulté le 02/06/2021]. Disponible sur : https://www.serial-port-monitor.org/fr/articles/serial-communication/types-of-serial-protocols/#RS232

[47] Lycée Maurice JANETTI. La liaison série UART. [consulté le 02/06/2021]. Disponible sur : http://tpil.projet.free.fr/TP_Arduino/01_UART.html

[48] DeltaLab. Installation et utilisation de Node-RED. [consulté le 28/05/2021]. Disponible sur : https://deltalabprototype.fr/wp-content/uploads/2019/10/Serveur-NodeRED.pdf

[49] David Martin. Node-Red : l'IoT à portée de tous. [consulté le 28/05/2021]. Disponible sur : https://blog.ippon.fr/2017/03/28/node-red-liot-a-portee-de-tous/

[50] DomoPi Guillaume. Découverte de Node-RED, un puissant logiciel d'automatisation. [consulté le 28/05/2021]. Disponible sur : https://domopi.eu/decouverte-de-node-red-un-puissant-logiciel-d-automatisation/

[51] Digital Guide IONOSI. nfluxDB : présentation, avantages, premiers pas. [consulté le 03/06/2021]. Disponible sur :

https://www.ionos.fr/digitalguide/hebergement/aspects-techniques/quest-ce-quinfluxdb/

[52] Influxdata documentations. Manage your database using InfluxQL. [consulté le 04/06/2021]. Disponible sur :

https://docs.influxdata.com/influxdb/v1.8/query_language/manage-database/

[53] Influxdata documentations. Explore your schema using InfluxQL. [consulté le 04/06/2021]. Disponible sur :

https://docs.influxdata.com/influxdb/v1.8/query_language/explore-schema/

[54] Wikipedia. Grafana. [consulté le 12/06/2021]. Disponible sur :

https://fr.wikipedia.org/wiki/Grafana

[55] Jouranel Du Net. Grafana : la data visualisation du monitoring IT open source (gratuit). [consulté le 13/06/2021]. Disponible sur :

https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443882-grafana-la-data-visualisation-du-monitoring-it-open-source-gratuit/

[56] Saagle. Visualisez vos séries temporelles avec Grafana. [consulté le 15/06/2021]. Disponible sur : https://www.saagie.com/fr/blog/visualisez-vos-series-temporelles-avec-grafana/

[57] Laura cano. Qu’est-ce que Grafana et comment l’utiliser dans la supervision. [consulté le 18/06/2021]. Disponible sur : https://pandorafms.com/blog/fr/quest-ce-que-grafana/

[58] Patrick Olivier. Vie des abeilles. [consulté le 28/04/2021]. Disponible sur : https://www.futura-sciences.com/planete/dossiers/zoologie-abeille-sentinelle-ecologique-684/page/3/

[59] Charles Julien. Connaissance des abeilles et de leur mode de vie. [consulté le 26/04/2021]. Disponible sur : http://fleurdeciel.fr/savoir-faire/connaissance-des-abeilles-et-de-leur-mode-de-vie/

[60] ECONECT. Ruches et fleurs connectés Comportement, cognition et suivi des colonies d'abeilles domestiques. [consulté le 23/03/2021]. Disponible sur : https://econect.cnrs.fr/ruche-connectee/

[61] CAPAZ. CAPAZ. [consulté le 23/06/2021].Disponible sur : http://bienenwaage.de/pdf_biene/gsm_inff.pdf

[62] BeeOnline, Station de mesure apicole connectée. [consulté le 23/06/2021]. Disponible sur : https://www.bee-online.fr

[63] Alya. VILKO 02. [consulté le 23/06/2021].Disponible sur : http://www.alya.sk/vcelarska-vaha-vilko

[64] Karl Von Frisch. Vie et mœurs des abeilles. Publié en 1953. [consulté le 21/04/2021]. Disponible sur : https://www.unitheque.com/vie-moeurs-des-abeilles/bibliotheque-sciences/albin-michel/Livre/42493

Les librairies utilisées pour les codes :

[65] Bertrand Bouvier. Librarie HX711. Disponible sur : https://os.mbed.com/teams/Cr300-Litho/code/HX711//file/f82840dd806a/HX711.cpp/

[66] Alex Lipford. Librairie HTU21D. Disponible sur : https://os.mbed.com/users/alipford3/notebook/htu21d---temperature-and-humidity-sensor1/?fbclid=IwAR0C8QNt4vuRXwjhcaVkU9HbFj3J662Nx4paSJ4HuA-ucb-eiFhMhVw2RPs

[67] Hiroshi Yamaguchi. Librairie MS5611. Disponible sur : https://os.mbed.com/teams/Aerodyne/code/MS5611/

[68] Andrea Corrado. Librairie CCS811. Disponible sur : https://os.mbed.com/users/andcor02/code/CCS811-TEST/

[69] Michael Wei. Librairie PowerControl. Disponible sur : https://os.mbed.com/users/no2chem/code/PowerControl/