Fournier Alan – Boutte Alexis – Konkin Maksim
Dossier commun

Lycée LA FAYETTE Clermont-Ferrand

PROJET NICHOIR

AUTONOME CONNECTÉ

Sommaire

Présentation du système

Un représentant de la LPO Auvergne-Rhône-Alpes est venu nous voir pour exprimer le besoin d'un système permettant le suivi des oiseaux des villes. Notre client est une association comptant plus de 11 440 adhérents et 1500 bénévoles actifs qui effectuent annuellement 300 000 heures de bénévolat au sein de la structure. LPO signifie Ligue de Protection des Oiseaux. Bien que son nom suggère qu'il s'agit d'une association s'occupant uniquement de la protection des oiseaux, la LPO agit pour la biodiversité de manière générale. Par exemple, en Auvergne-Rhône-Alpes, la LPO peut très bien mener des actions de tout type, comme des chantiers nature, des sorties sur le terrain, des conférences, des ateliers, des tenues de stand,… Le tout dans un seul but précis, sensibiliser un public large et varié à la protection de la nature.

Cahier des charges

Le cahier des charges du système nous a été communiqué par le client lors de sa demande. Le périmètre du système s'étend aux villes, pour permettre aux bénévoles de l'association de contrôler plus simplement et efficacement la nidification des oiseaux dans le milieu urbain. La première contrainte concerne la forme du système. Il doit s'agir d'un nichoir qui relève la température et l'humidité intérieure et extérieure, qui prend des photos infrarouges et standards des oiseaux, et enfin qui pèse les oiseaux. Le coût final du produit doit être le plus faible possible, pour être profitable à l'association et être installé en grand nombre. Ensuite, pour permettre un domaine de supervision plus étendu, nous devons permettre une consultation des données sur deux supports, une application sur PC ou smartphone et un site web. Bien entendu, le système doit être sécurisé pour éviter aux données de l'association d'être recueillies par des tiers. Concernant les données, celles-ci doivent être stockées à distance, mais aussi localement en cas de panne de connexion. Enfin, la dernière contrainte est celle du temps. Le produit final doit être prêt en mai, ce qui signifie qu'à partir de la date d'expression du besoin, en début janvier, nous avons 5 mois pour produire un système fonctionnel et complet, répondant au cahier des charges explicité ci-dessus.

Description du système

Pour répondre aux besoins du client, nous avons pris comme base un nichoir en bois qui dispose d'un faux fond. Du point de vue technique, nous allons utiliser un mini-ordinateur comme cœur du système. Nous connecterons à cet appareil plusieurs capteurs. Pour commencer, afin de relever la température et l'humidité à l'intérieur et à l'extérieur du nichoir, nous utiliserons deux capteurs qui permettent la mesure simultanée de ces deux grandeurs. Ensuite, le faux fond du nichoir va nous permettre de mettre en place un système de balance relevant le poids des oiseaux. Pour finir, la capture des images à l'intérieur du nichoir sera faite pour que nous puissions distinguer la présence des oiseaux à l'aide d'une caméra infrarouge et d'une caméra standard. Au final, les données nécessaires au suivi de la nidification des oiseaux seront collectées par 5 capteurs différents. Le mini-ordinateur disposera d'un pare-feu pour interdire les connexions extérieures non souhaitées et ainsi sécuriser le système.

Le système enverra ensuite les données à une base de données hébergée sur un serveur distant, pour leur enregistrement. Le système sera sécurisé par un mot de passe fort, et l'accès sera possible uniquement sur le même réseau local. Les valeurs récupérées par les capteurs seront aussi enregistrées directement sur le mini-ordinateur en cas de panne de connexion. Toutes les acquisitions de données seront datées par la carte à leur envoi dans la base de données dans le but de produire des affichages des valeurs en fonction du temps sur les outils de consultation. Cela servira à suivre la progression des relevés de façon claire.

Objectifs du projet

Objectif principal

L'objectif du projet est de concevoir un service d'observation des oiseaux, afin de contrôler leur nidification. La supervision sera faite avec diffusion des informations en temps réel.

Il s'agira donc de disposer d'un nichoir qui comprend des outils de surveillance. Nous travaillerons sur l'acquisition et la mise en forme des observations pour les rendre facilement accessibles par les utilisateurs.

Objectifs techniques

Le premier objectif technique est d'écrire les programmes principaux sur la carte Raspberry qui permettront de récupérer et d'enregistrer les données issues des capteurs dans la base de données distante et locale avec les bons formats.

En amont, il est nécessaire de mettre en place un serveur de base de données distant qui servira à sauvegarder les informations issues des capteurs : température, humidité, poids, et les images des caméras de surveillance.

Nous devons concevoir l'application sur PC et le site web qui serviront tous les deux à récupérer les informations de la base de données distante et de les afficher de façon claire pour les utilisateurs.

Afin de pouvoir utiliser le site web sur le réseau local, nous devrons mettre en place un serveur web qui hébergera le site, qui servira à visualiser les données collectées par le nichoir au même titre que l'application sur PC.

Pour que les utilisateurs du système n'aient pas la contrainte d'intervenir de manière régulière dans le but de le recharger, l'ensemble du système faisant les acquisitions doit être autonome en énergie.

Enfin, les équipements du projet doivent être sécurisés en tout point. Cela signifie que les deux outils de consultation, la base de données distante ainsi que la partie physique se trouvant au cœur du nichoir sont concernés par le besoin de sécurité.

Choix techniques

Solutions possibles

Le tableau ci-dessous permet d'avoir une vision globale des besoins exprimés par le client et des solutions techniques qui peuvent être réalisées pour les satisfaire.

N° besoin	Besoin	Solutions possibles
1	Contrôler les capteurs	Raspberry Pi 4 Raspberry Pi Zero
2	Récupérer la température et l'humidité relative	DHT22 DHT11
3	Récupérer le poids	HX711 + cellule de charge 780g
4	Capturer des images infrarouges	AMG8833 RPI IR Camera
5	Capturer des images standards	Caméra ToF pour Raspberry Pi 4 Caméra HQ officielle – Monture M12 Raspicam v2.1
6	Gérer l'alimentation du système	Carte Witty Pi 4 Carte Witty Pi 4 Mini
7	Stocker l'énergie nécessaire au fonctionnement du système	Batterie Power Sonic PS1270
8	Contrôler la charge du système	Steca Solsum 6.6c Steca Solsum 8.8c Steca Solsum 10.10c
9	Enregistrer les données à distance	Serveur UwAmp MySQL Serveur MariaDB
10	Enregistrer les données localement	SQLite PostgreSQL
11	Héberger le site web	Serveur web Apache sur Raspberry Serveur web UwAmp distant

Solutions retenues

Le tableau ci-dessous présente les solutions qui ont été retenues pour répondre aux besoins et les raisons qui nous ont poussées à faire ces choix.

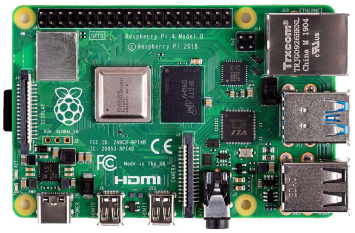
N° besoin	Solution retenue	Raison
1	Raspberry Pi 4	Ce mini-ordinateur dispose d'une plus grande puissance de calcul que la carte Raspberry Pi Zero.
2	DHT22	Le capteur de température du DHT22 peut relever des valeurs négatives, ce qui est utile durant l'hiver.
3	HX711 + cellule de charge 780g	Il s'agit du seul matériel dont nous disposons déjà. La cellule de poids aurait pu permettre de relever un poids plus élevé, mais 780g suffisent.
4	AMG8833	Il s'agit ici aussi du seul matériel dont nous disposons. Bien que sa résolution soit faible, son mode de fonctionnement est similaire aux capteurs avec lesquels nous avons l'habitude de travailler.
5	Raspicam v2.1	Cette caméra standard est la seule dont nous disposons. Cependant, son mode de fonctionnement est simple, notre choix se serait tout de même porté sur cette solution.
6	Carte Witty Pi 4	La carte Witty Pi 4 Mini est trop petite pour être maintenue entre le Raspberry Pi 4 et le shield, nous avons donc choisi la carte Witty Pi 4, dont le montage convient mieux.
7	Batterie Power Sonic PS1270	Il s'agit de la seule batterie dont nous disposons. Nous aurions pu choisir une batterie 5V, mais l'utilisation d'un contrôleur de charge nous permet d'utiliser une batterie 12V.
8	Steca Solsum 6.6c	Ce contrôleur de charge est le seul que nous avons à disposition. Nous n'avons pas cherché un autre modèle, car celui-ci remplit bien son rôle.
9	Serveur UwAmp MySQL	Nous avons l'habitude d'utiliser le serveur UwAmp. Son interface phpMyAdmin est pratique et claire, contrairement au serveur MariaDB qui fonctionne en lignes de commandes.
10	SQLite	La solution PostgreSQL aurait pu être envisagée au même titre que la base de données SQLite. Ces deux possibilités remplissent leur rôle de façon similaire.
11	Serveur web UwAmp distant	Le serveur UwAmp distant permet aussi d'héberger facilement un site web. Héberger le site sur la carte Raspberry n'est pas possible puisqu'elle s'éteint lorsqu'elle n'est pas utilisée.

Matériel utilisé

Nichoir en bois

Notre système physique prendra la forme d'un nichoir en bois dont la forme est similaire à l'image qui vous est présentée. Ce nichoir contiendra l'ensemble des capteurs ainsi que le mini-ordinateur qui s'occupera de l'acquisition et de l'enregistrement des données. À proximité du nichoir seront aussi placés les équipements nécessaires à l'autonomie du système tels que la batterie, la cellule photovoltaïque et le contrôleur de charge.

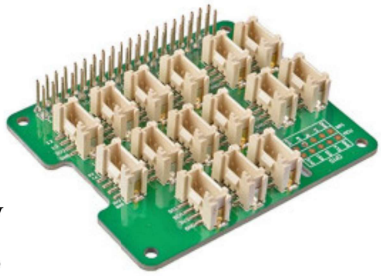
Mini-ordinateur Raspberry Pi 4



Le cœur du système physique est la carte mini-ordinateur Raspberry Pi 4. La Raspberry Pi 4 se différencie des microcontrôleurs tels que la carte Arduino Uno. En effet, elle est composée d'un microcontrôleur : processeur, mémoire vive et morte, interfaces d'entrées-sorties, ainsi que d'un système d'exploitation, lui valant l'appellation de « mini-ordinateur ».

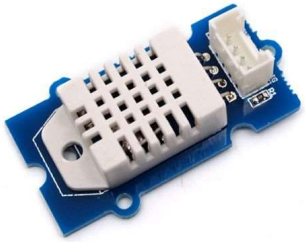
Au sein du projet, la Raspberry Pi 4 va nous servir à récupérer les données des différents capteurs pour les traiter puis les enregistrer dans la base de données distante. Par ailleurs, elle contiendra une base de données locale pour pouvoir tout de même enregistrer les données dans le cas où la connexion à la base de données distante serait interrompue.

Module Grove Base Hat



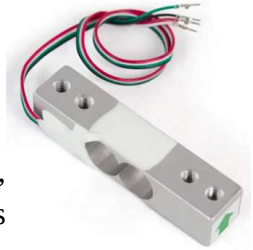
Les connexions entre tous les capteurs et la carte Raspberry Pi 4 sont simplifiées à l'aide du module Grove Base Hat, communément appelé « shield ». Ce module se branche directement sur le mini-ordinateur et met à disposition des ports physiques utilisant différents protocoles de communication en fonction des besoins de l'utilisateur.

Capteur de température et d’humidité DHT22



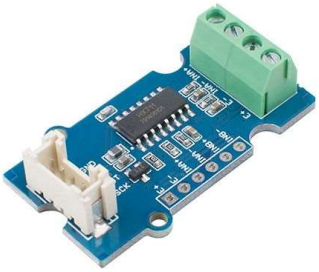
Le capteur DHT22 permet de faire l’acquisition de la température et de l’humidité relative de façon simultanée. Sa sensibilité est intéressante vis à vis des besoins du projet puisqu’elle est faible. Les mesures sont effectuées à 0,1°C près pour la température et 0,1 % près pour l’humidité relative. De même, sa plage de relevés est intéressante. Avec ce capteur, il est possible de mesurer la température de -40°C à 80°C, ce qui est idéal pour les relevés en hiver. L’humidité relative est quant à elle relevée de 0 à 100 %. Ce capteur communique avec la carte Raspberry Pi 4 à l’aide d’un port GPIO (General Purpose Input/Output) qui est un port série d’entrées-sorties.

Cellule de charge micro CZL611CD



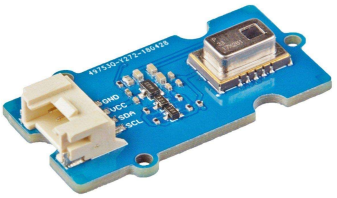
La cellule de charge utilisée peut mesurer une masse maximale de 780g, ce qui est suffisant en considérant que les oiseaux ne pèsent que quelques grammes. Elle est composée de 4 jauges de contraintes, qui sont des fils dont la résistance varie en fonction de la déformation qui leur est appliquée. Montées électriquement selon le schéma du pont de Wheatstone, il est ainsi possible de récupérer une tension de sortie en fonction de la charge appliquée.

Capteur de poids HX711



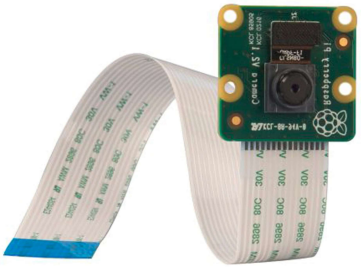
Le capteur HX711 récupère la tension en sortie de la cellule de charge, qui sert à la mesure du poids. La tension de sortie est ensuite traitée par le capteur pour être utilisée par la carte Raspberry Pi 4. Le HX711 communique avec le mini-ordinateur à l’aide d’un port série d’entrées-sorties GPIO.

Caméra infrarouge AMG8833



Cette caméra infrarouge permet de capturer des images à l'aide d'un capteur de résolution 8x8 pixels. Cette résolution n'étant pas suffisante pour interpréter les images capturées, il sera nécessaire de mettre en place une interpolation, qui consiste à générer des valeurs intermédiaires entre les valeurs réellement mesurées pour obtenir une meilleure résolution. La caméra infrarouge communique avec la carte Raspberry Pi 4 par l'intermédiaire du bus série I2C.

Caméra standard Raspicam v2.1



Conçue par les mêmes constructeurs que la carte Raspberry Pi 4, cette caméra standard permet de capturer des images en utilisant des lignes de commandes, contrairement au reste des capteurs qui doivent être programmés. Il est possible de régler plusieurs paramètres lors de la prise de l'image tels que la durée d'exposition et le grain par exemple. Cette caméra est reliée au port MIPI CSI situé sur la partie droite du mini-ordinateur. La caméra utilise une nappe et communique à l'aide de l'interface CSI.

Carte Witty Pi 4

La carte Witty Pi 4 gère l'alimentation de la carte Raspberry Pi 4. Elle permet notamment de définir les arrêts et démarrages programmés du système afin de consommer le moins d'énergie possible. Cette fonction est réalisée à l'aide de l'horloge RTC (Real Time Clock) intégrée à la carte qui conserve la date et l'heure en mémoire lorsque le système est éteint. La carte Witty Pi 4 se connecte directement sur la Raspberry Pi 4 de la même manière que le shield. Une fois connectée, l'alimentation doit être branchée sur la carte et non sur le mini-ordinateur.

Batterie Power Sonic PS1270

L’autonomie du système sera assurée par plusieurs éléments, dont la batterie Power Sonic PS1270. Elle délivre une tension de 12V et dispose d’une capacité de 7Ah, ce qui est amplement suffisant compte tenu de l’énergie consommée par le système en moyenne.

Cellule photovoltaïque Velleman SOL10P

Cette cellule photovoltaïque permettra de délivrer l’énergie nécessaire pour recharger la batterie en évitant les pertes. Son rendement est de 11 %, ce qui reste faible. Cependant, en hiver, dans le cas le plus défavorable, cela suffit à recharger suffisamment la batterie pour que le système ne s’éteigne pas.

Contrôleur de charge Steca Solsum 6.6c

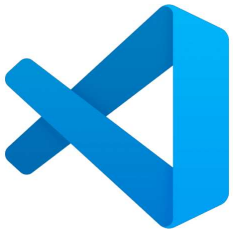
Afin de relier tous les éléments permettant de mettre en autonomie le système, nous aurons besoin du contrôleur de charge Steca Solsum 6.6c. Dessus, nous connecterons la batterie, la cellule photovoltaïque et la carte Witty Pi 4. Notamment, le contrôleur de charge permettra de délivrer les 5V nécessaires en sortie pour le système, avec une batterie de tension 12V.

Outils utilisés

Qt Creator

Afin de développer l'application de consultation sur PC, l'étudiant 2 va se servir de l'environnement de développement Qt Creator. Cet outil est multi-plateforme et dispose d'une multitude de bibliothèques pour mener à bien son projet. Le langage de programmation qui est utilisé est le C++.

Visual Studio Code



Développé par Microsoft, Visual Studio Code est un éditeur de code qui prend en charge un grand nombre de langages de programmation. Notamment, cet outil peut suggérer de compléter notre code, met en évidence la syntaxe, et prend en charge le débogage. Dans le cadre de notre projet, Visual Studio Code a été utilisé avec l'extension Live Server pour développer le site web de consultation en observant les modifications en temps réel.

Serveur UwAmp

Le serveur UwAmp est très simple à mettre en place et à utiliser. Il est disponible sous la forme d'une archive, qu'il suffit de décompresser. Les lettres « Amp » signifient respectivement « Apache », « MySQL » et « PHP ». Ce serveur est composé d'un serveur web Apache pour héberger les sites web, d'un serveur de bases de données MySQL et prend en charge le langage PHP. Cet outil permet donc à la fois d'héberger le site web de consultation ainsi que d'héberger la base de données distante, en proposant une interface claire et simple à utiliser.

GitHub

GitHub est une société qui offre un service d'hébergement de référentiel Git, qui est un système de contrôle de version open source spécifique créé en 2005. Le contrôle de version permet aux développeurs de suivre et gérer les modifications apportées au code d'un projet. Nous avons créé et utilisé un dépôt GitHub pour sauvegarder le code du projet et s'organiser en notant les tâches effectuées dans des fichiers texte.

Coût du projet

Le coût du projet correspond aux dépenses nécessaires à son fonctionnement. Les dépenses se limitent uniquement au matériel, car l'ensemble des outils et des programmes utilisés sont gratuits. Vous trouverez ci-dessous le prix de chaque élément composant du projet.

Composant	Prix
Nichoir en bois	30€
Raspberry Pi 4	60€
Grove Base Hat	10,6€
DHT22 x2	22,8€
HX711	5,9€
Cellule de charge 780g	7,9€
AMG8833	44,5€
Raspicam v2.1	29,9€
Witty Pi 4	37,95€
Power Sonic PS1270	14,9€
Velleman SOL10P	39,96€
Steca Solsum 6.6c	50,66€
Total	377,87€

Le coût total du matériel composant le projet est égal à 377,87€. Si nous considérons que l'ordinateur qui fait office de serveur est aussi à acheter, nous pouvons alors ajouter au total environ 200€, ce qui ferait un coût de 577,87€ pour l'ensemble. Les prix des composants ont été récupérés sur différents sites de fournisseurs. Il n'a pas été possible de récupérer tous les prix sur le même site.

Répartition des tâches

Pour suivre et organiser l'avancement du projet, nous avons mis en place un dépôt GitHub. Chaque élève dispose d'un fichier de suivi où il explique les tâches qu'il a effectuées, concluantes ou non, durant une session de projet. Ces informations nous permettront de fabriquer le diagramme de Gantt réel, pour le comparer au diagramme de Gant prévisionnel, présenté un peu plus tard dans le dossier commun. Le dépôt GitHub permet aussi de sauvegarder les fichiers et programmes du projet.

Étudiant 1 – Alan Fournier

Alan doit réaliser les programmes principaux sur la carte Raspberry qui permettent l'acquisition et la sauvegarde des données issues des capteurs. Il doit aussi implémenter le processus de prise d'images. Toutes les données récupérées doivent être enregistrées dans une base de données locale et distante. Il s'occupe aussi de mettre en place le service sur le système embarqué qui permet la mise en veille et le réveil programmés. Les tâches qu'il doit effectuer sont à la racine du projet puisque sans les données réelles des capteurs, les autres étudiants ne peuvent pas entièrement compléter leurs tâches.

Étudiant 2 – Alexis Boutte

Alexis s'occupe de la partie Interface Homme Machine (IHM) externe sous forme d'une application pour le contrôle distant du nichoir. Il doit mettre en place un serveur de base de données ainsi que créer la base de données distante qui sert au stockage des informations. Les données doivent pouvoir être affichées et consultées par les utilisateurs. L'application qu'il développe doit pouvoir interpréter les données dans le bon format pour les afficher de façon claire.

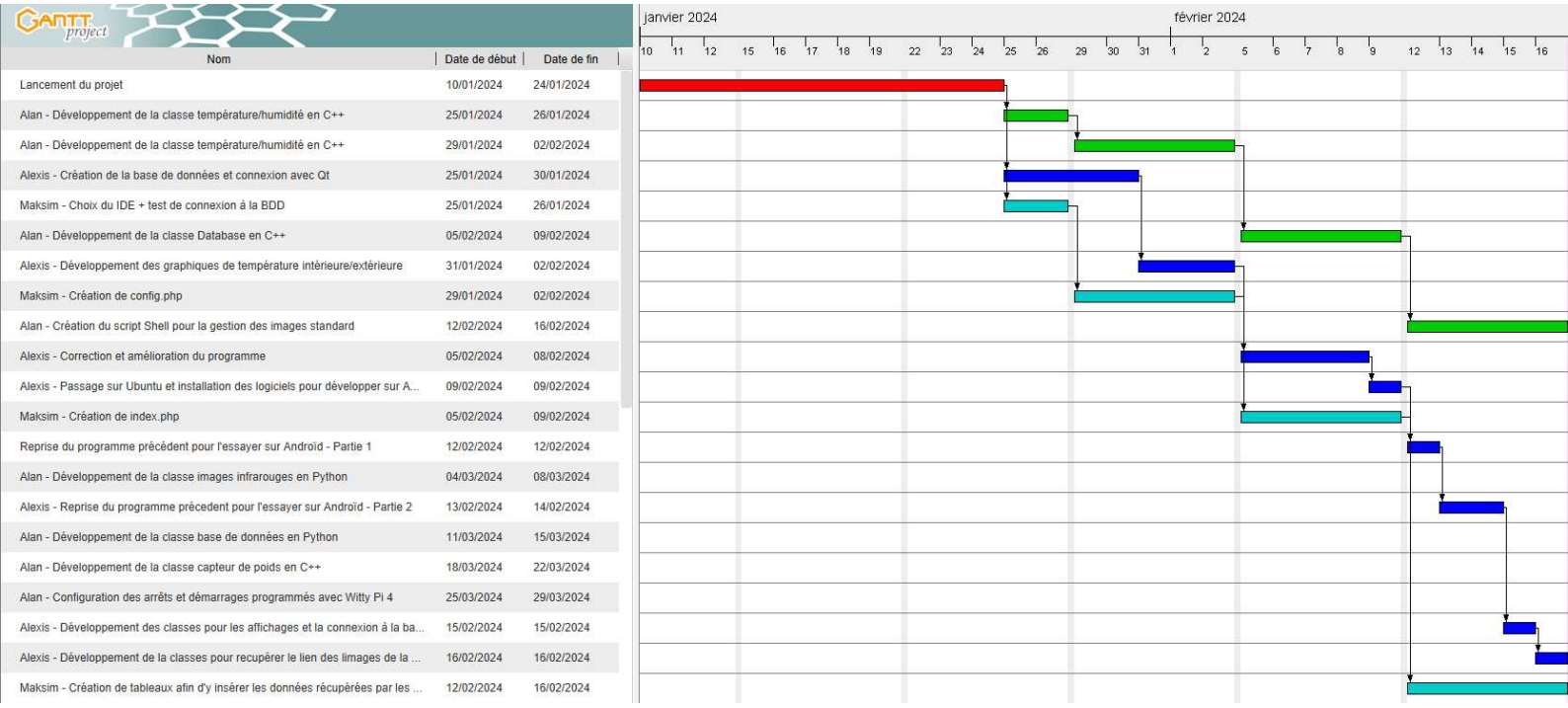
Étudiant 3 – Maksim Konkin

Maksim doit mettre en place un serveur web et développe un site Internet hébergé sur ce serveur permettant de visualiser les données collectées par le nichoir. Tout comme Alexis, l'outil de consultation de Maksim doit pouvoir interpréter les données de manière à ce qu'elles paraissent claires aux utilisateurs et contrôler le nichoir.

Diagramme de Gantt prévisionnel

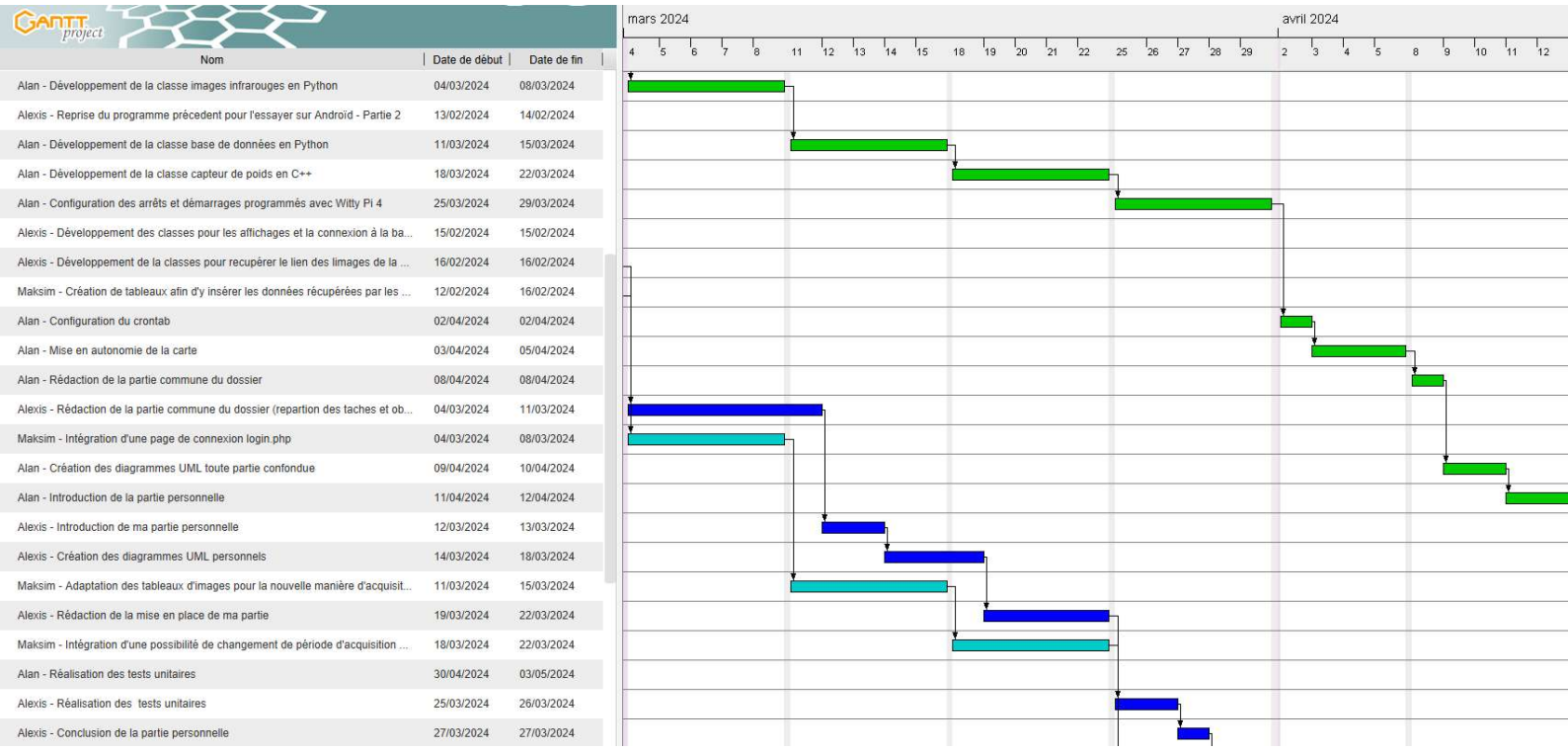
Janvier à Février

Les tâches en bleu foncé sont celles d'Alexis, en cyan celles de Maksim et celles en vert sont celles d'Alan.



Nom	Date de début	Date de fin
Lancement du projet	10/01/2024	24/01/2024
Alan - Développement de la classe température/humidité en C++	25/01/2024	26/01/2024
Alan - Développement de la classe température/humidité en C++	29/01/2024	02/02/2024
Alexis - Création de la base de données et connexion avec Qt	25/01/2024	30/01/2024
Maksim - Choix du IDE + test de connexion à la BDD	25/01/2024	26/01/2024
Alan - Développement de la classe Database en C++	05/02/2024	09/02/2024
Alexis - Développement des graphiques de température intérieure/extérieure	31/01/2024	02/02/2024
Maksim - Création de config.php	29/01/2024	02/02/2024
Alan - Création du script Shell pour la gestion des images standard	12/02/2024	16/02/2024
Alexis - Correction et amélioration du programme	05/02/2024	08/02/2024
Alexis - Passage sur Ubuntu et installation des logiciels pour développer sur A...	09/02/2024	09/02/2024
Maksim - Création de index.php	05/02/2024	09/02/2024
Reprise du programme précédent pour l'essayer sur Androïd - Partie 1	12/02/2024	12/02/2024
Alan - Développement de la classe images infrarouges en Python	04/03/2024	08/03/2024
Alexis - Reprise du programme précedent pour l'essayer sur Androïd - Partie 2	13/02/2024	14/02/2024
Alan - Développement de la classe base de données en Python	11/03/2024	15/03/2024
Alan - Développement de la classe capteur de poids en C++	18/03/2024	22/03/2024
Alan - Configuration des arrêts et démarrages programmés avec Witty Pi 4	25/03/2024	29/03/2024
Alexis - Développement des classes pour les affichages et la connexion à la ba...	15/02/2024	15/02/2024
Alexis - Développement de la classes pour recupérer le lien des limages de la ...	16/02/2024	16/02/2024
Maksim - Création de tableaux afin d'y insérer les données récupérées par les ...	12/02/2024	16/02/2024

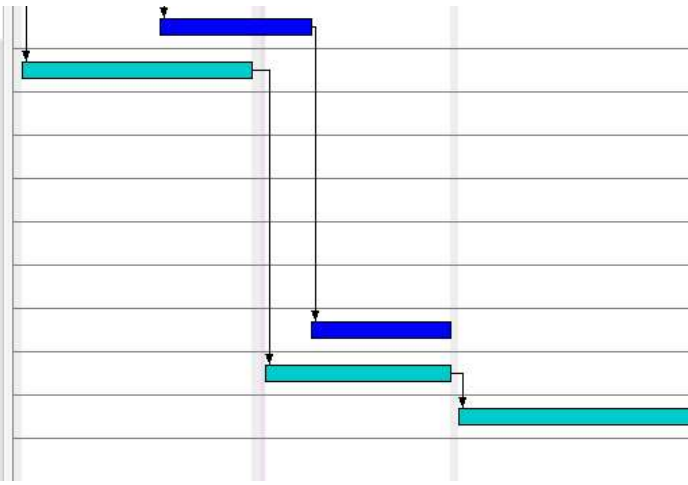
Mars à Avril 1e partie



Nom	Date de début	Date de fin
Alan - Développement de la classe images infrarouges en Python	04/03/2024	08/03/2024
Alexis - Reprise du programme précedent pour l'essayer sur Androïd - Partie 2	13/02/2024	14/02/2024
Alan - Développement de la classe base de données en Python	11/03/2024	15/03/2024
Alan - Développement de la classe capteur de poids en C++	18/03/2024	22/03/2024
Alan - Configuration des arrêts et démarrages programmés avec Witty Pi 4	25/03/2024	29/03/2024
Alexis - Développement des classes pour les affichages et la connexion à la ba...	15/02/2024	15/02/2024
Alexis - Développement de la classes pour recupérer le lien des limages de la ...	16/02/2024	16/02/2024
Maksim - Création de tableaux afin d'y insérer les données récupérées par les ...	12/02/2024	16/02/2024
Alan - Configuration du crontab	02/04/2024	02/04/2024
Alan - Mise en autonomie de la carte	03/04/2024	05/04/2024
Alan - Rédaction de la partie commune du dossier	08/04/2024	08/04/2024
Alexis - Rédaction de la partie commune du dossier (repartion des taches et ob...	04/03/2024	11/03/2024
Maksim - Intégration d'une page de connexion login.php	04/03/2024	08/03/2024
Alan - Création des diagrammes UML toute partie confondue	09/04/2024	10/04/2024
Alan - Introduction de la partie personnelle	11/04/2024	12/04/2024
Alexis - Introduction de ma partie personnelle	12/03/2024	13/03/2024
Alexis - Création des diagrammes UML personnels	14/03/2024	18/03/2024
Maksim - Adaptation des tableaux d'images pour la nouvelle manière d'acquisit...	11/03/2024	15/03/2024
Alexis - Rédaction de la mise en place de ma partie	19/03/2024	22/03/2024
Maksim - Intégration d'une possibilité de changement de période d'acquisition ...	18/03/2024	22/03/2024
Alan - Réalisation des tests unitaires	30/04/2024	03/05/2024
Alexis - Réalisation des tests unitaires	25/03/2024	26/03/2024
Alexis - Conclusion de la partie personnelle	27/03/2024	27/03/2024

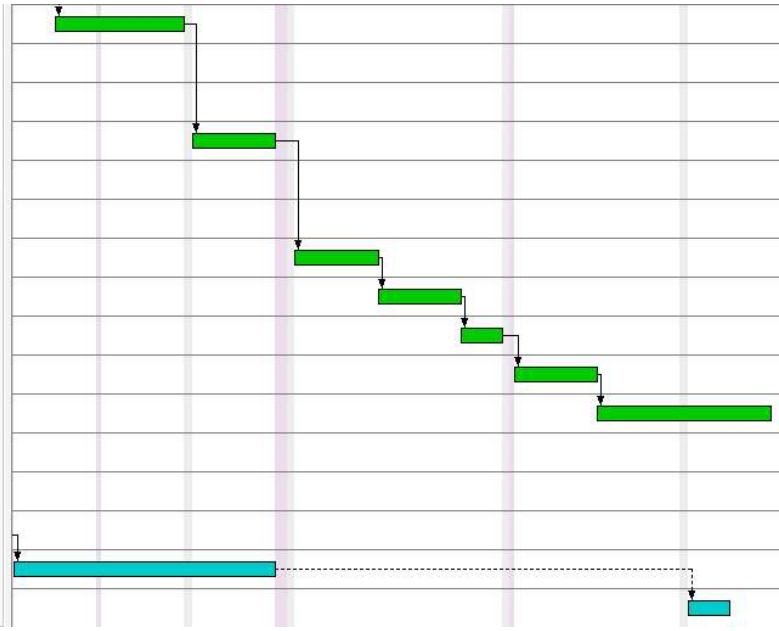
Mars à Avril 2e partie

Alexis - Relecture et correction du dossier personnel	28/03/2024	02/04/2024
Maksim - Création d'une page de style bdd.css	25/03/2024	29/03/2024
Alan - Rédaction de la mise en place matérielle et logicielle du système	13/05/2024	14/05/2024
Alan - Rédaction de la conclusion de la partie personnelle	15/05/2024	16/05/2024
Alan - Mise en commun du dossier	17/05/2024	17/05/2024
Alan - Relecture et correction du dossier complet	21/05/2024	22/05/2024
Alan - Création du diaporama de présentation et entraînement	23/05/2024	28/05/2024
Alexis - Création du diaporama et entraînement	03/04/2024	05/04/2024
Maksim - Intégration d'une possibilité de suppression de données au choix à p...	02/04/2024	05/04/2024
Maksim - Travail sur la partie personelle	08/04/2024	12/04/2024
Maksim - Création de diaporama et entraînement à l'oral	29/04/2024	07/05/2024



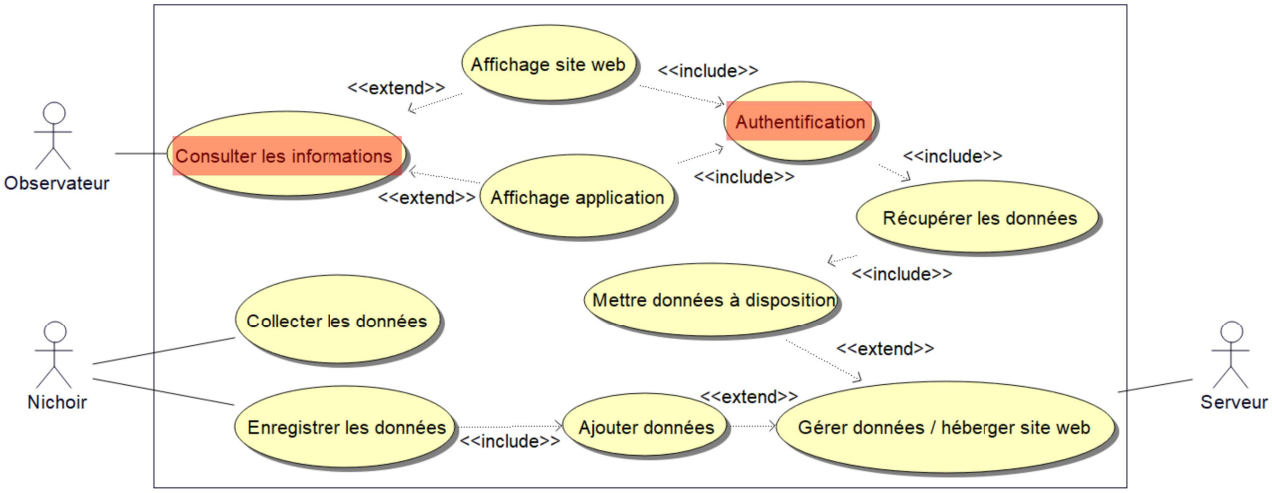
Mai

Alan - Réalisation des tests unitaires	30/04/2024	03/05/2024
Alexis - Réalisation des tests unitaires	25/03/2024	26/03/2024
Alexis - Conclusion de la partie personnelle	27/03/2024	27/03/2024
Alan - Rédaction de la mise en place matérielle et logicielle du système	06/05/2024	07/05/2024
Alexis - Relecture et correction du dossier personnel	28/03/2024	02/04/2024
Maksim - Création d'une page de style bdd.css	25/03/2024	29/03/2024
Alan - Rédaction de la mise en place matérielle et logicielle du système	13/05/2024	14/05/2024
Alan - Rédaction de la conclusion de la partie personnelle	15/05/2024	16/05/2024
Alan - Mise en commun du dossier	17/05/2024	17/05/2024
Alan - Relecture et correction du dossier complet	21/05/2024	22/05/2024
Alan - Création du diaporama de présentation et entraînement	23/05/2024	28/05/2024
Alexis - Création du diaporama et entraînement	03/04/2024	05/04/2024
Maksim - Intégration d'une possibilité de suppression de données au choix à p...	02/04/2024	05/04/2024
Maksim - Travail sur la partie personelle	08/04/2024	12/04/2024
Maksim - Création de diaporama et entraînement à l'oral	29/04/2024	07/05/2024
Maksim - Screenshot du site internet afin de les intégrer au dossier personnel	27/05/2024	27/05/2024



UML

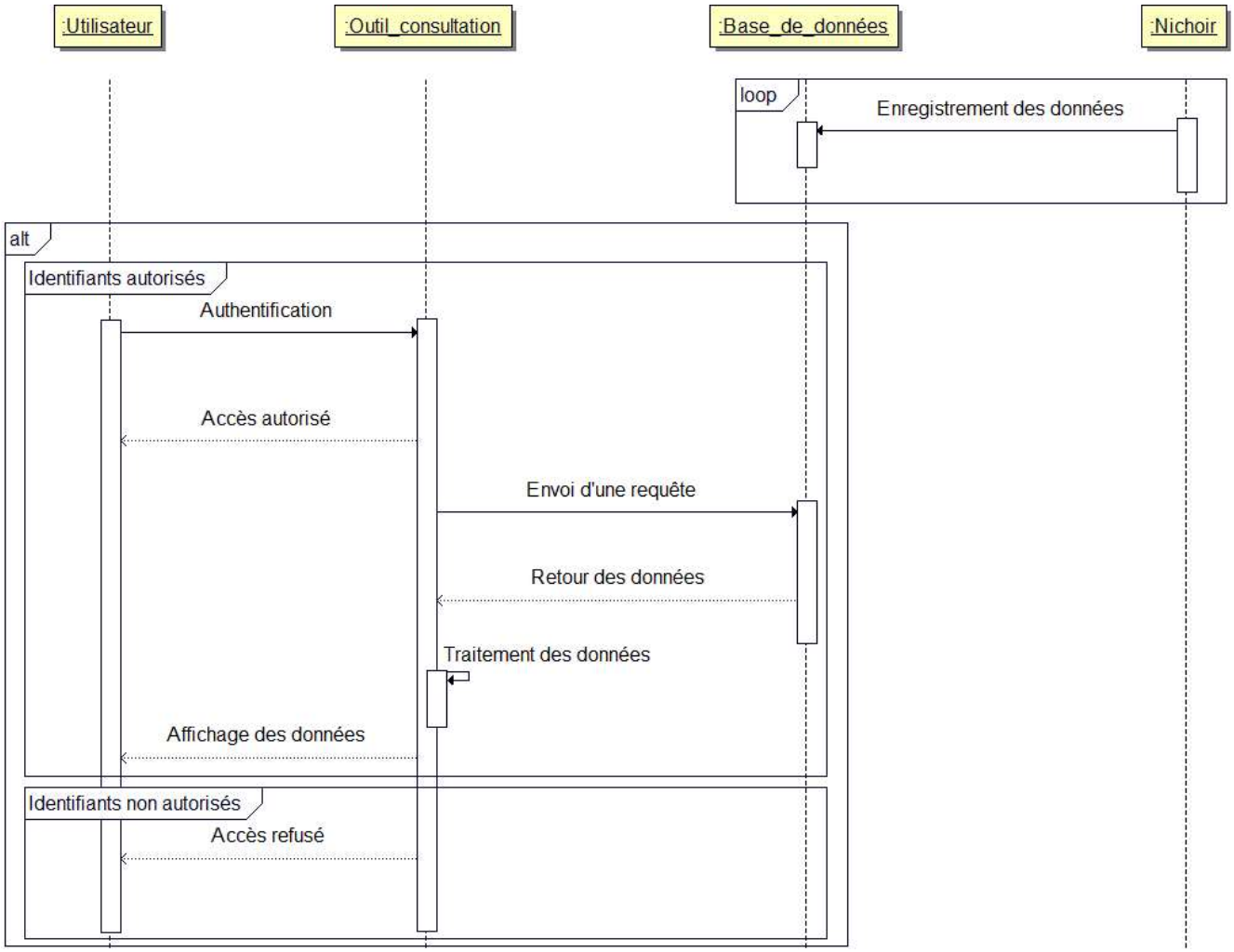
Diagramme des cas d’utilisation



Les zones vertes représentent les tâches effectuées par Alan, l’étudiant 1. Les zones bleues représentent les tâches du projet effectuées par Alexis, l’étudiant 2. Les tâches effectuées par Maksim, l’étudiant 3, sont représentées par les zones oranges. Enfin, les zones rouges représentent les tâches effectuées en commun par Alexis et Maksim, les étudiants 2 et 3.

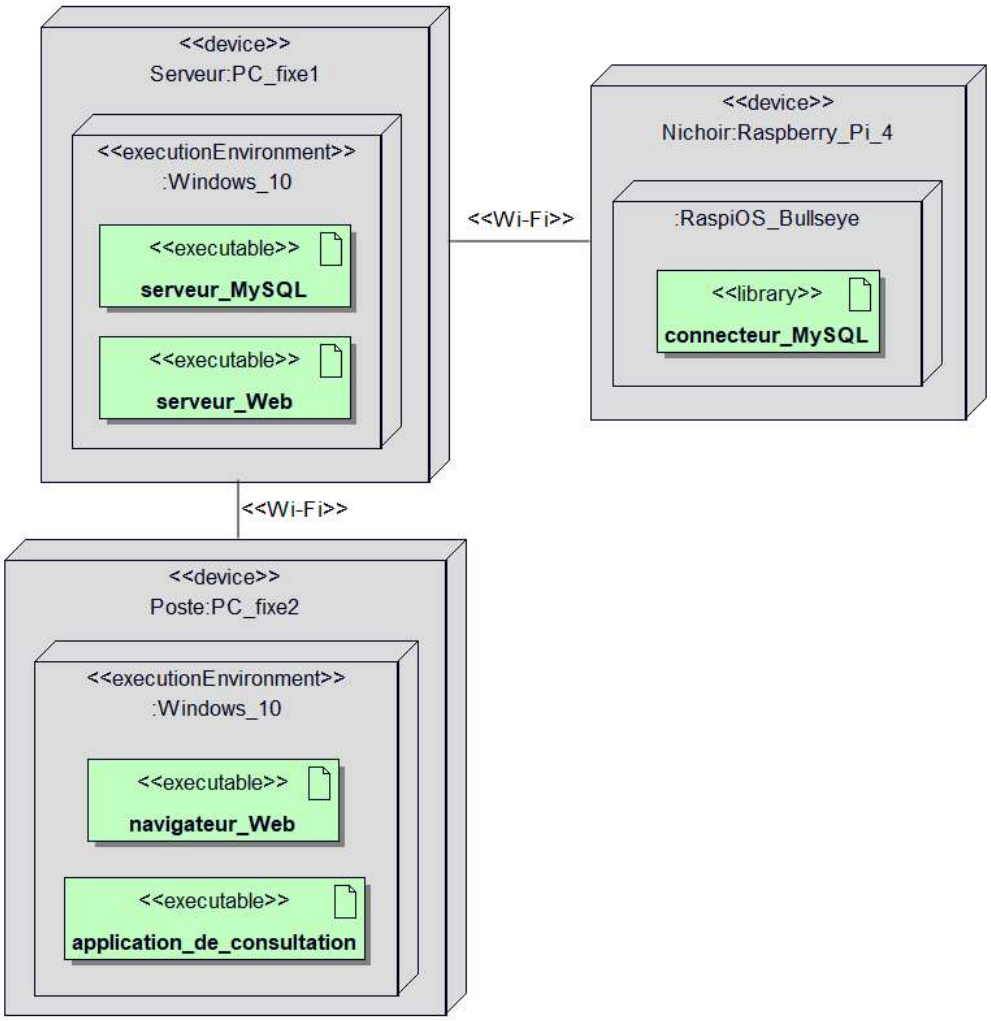
Lorsque l’observateur souhaite consulter les données, elles doivent au préalable avoir été collectées par le nichoir, puis sauvegardées sur le serveur de base de données. Les données sont ensuite affichées sur l’application ou sur le site web en fonction de la méthode souhaitée par l’observateur.

Diagramme de séquence



Si l'utilisateur dispose d'identifiants autorisés, il peut accéder aux données, l'outil de consultation envoie alors une requête à la base de données. Une fois récupérées, les données sont traitées puis affichées par l'outil de consultation. À l'inverse, si ses identifiants de l'utilisateur ne sont pas autorisés, l'utilisateur ne peut pas accéder aux données. L'outil de consultation lui indique que l'accès lui est refusé.

Diagramme de déploiement



La carte Raspberry Pi 4 communique en Wi-Fi avec la base de données distante pour enregistrer les données des capteurs qu'elle récupère. Sur un PC fixe, nous pourrons exécuter une application de consultation qui se connectera à la base de données en Wi-Fi pour accéder aux données et les afficher, ou alors utiliser un navigateur web pour consulter les données sur un site Internet.

Partie personnelle projet

Nom du projet : Nichoir connecté

Nom des étudiants : FOURNIER Alan / BOUTTE Alexis /KONKIN Maksim

Section : BTS Systèmes numériques option A informatique réseaux

Nom du lycée : Lycée Lafayette Clermont-Ferrand

Table des matières

1) Remerciements

Tout d’abord je remercie M.BARTOMEUF, M.BUCKLE, M.DEGOUTE mes profs d’enseignement de spécialité ainsi que Mme Chopin ma profs de physique appliquée de leur suivi quotidien est de leurs aides

2) Rappel des responsabilités dans le projet et dans le groupe

Rappel de mes tâches :

-Création de la base de données. Elles servira à stoker les données, envoyées par les capteurs du nichoirs

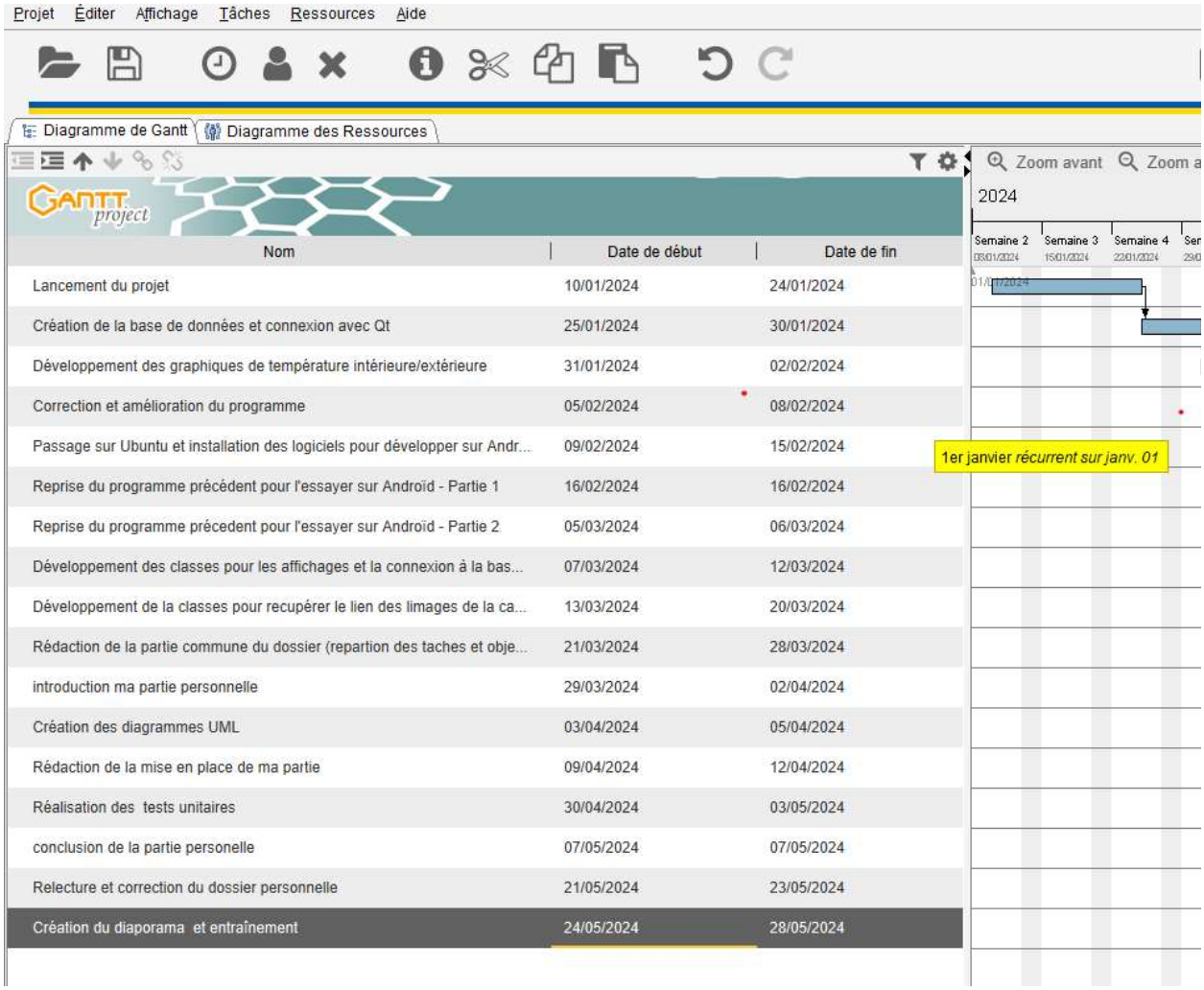
- Création d'une table de données pour chaque capteur (température int/ext, humidité int/ext et poids).

- Création de deux tables de données qui serviront à stocker les liens des images des caméras de surveillance intérieurs et extérieurs.

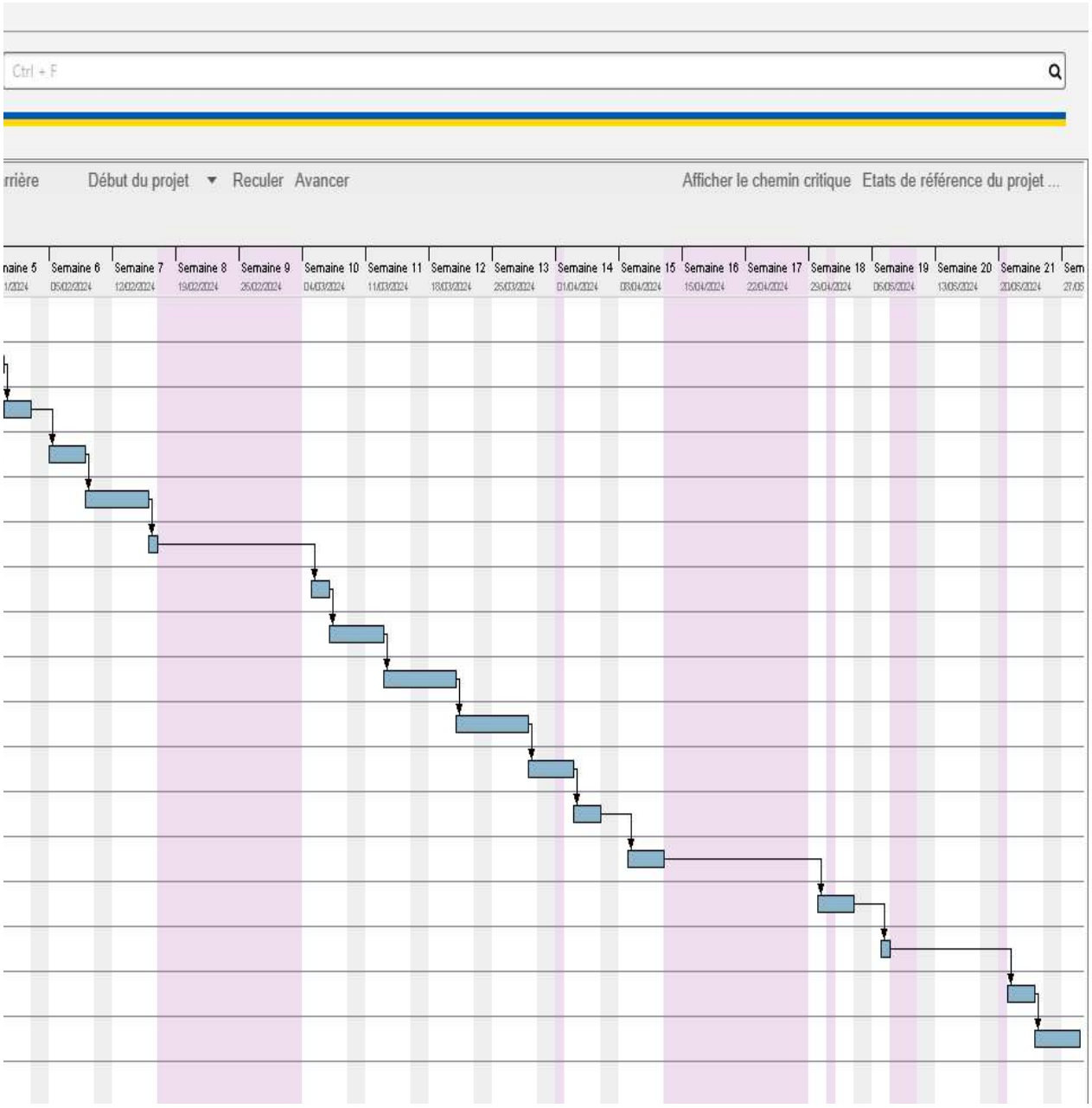
- Coder la classe (MainWindow) pour récupérer les données et la dernière des deux camera du nichoir (standard est infrarouges) de la base et les afficher

2 . 1)Gantt prévisionnel :

Gantt est un logiciel qui nous permettra d'organiser notre projet en proposant un système simple de planification de tâches. Chaque tâche aura été préalablement rentrée dans le logiciel avec une durée prévue pour la réaliser, nous permettant alors d'estimer notre temps de travail et par extension, d'optimiser notre avancée sur le projet.

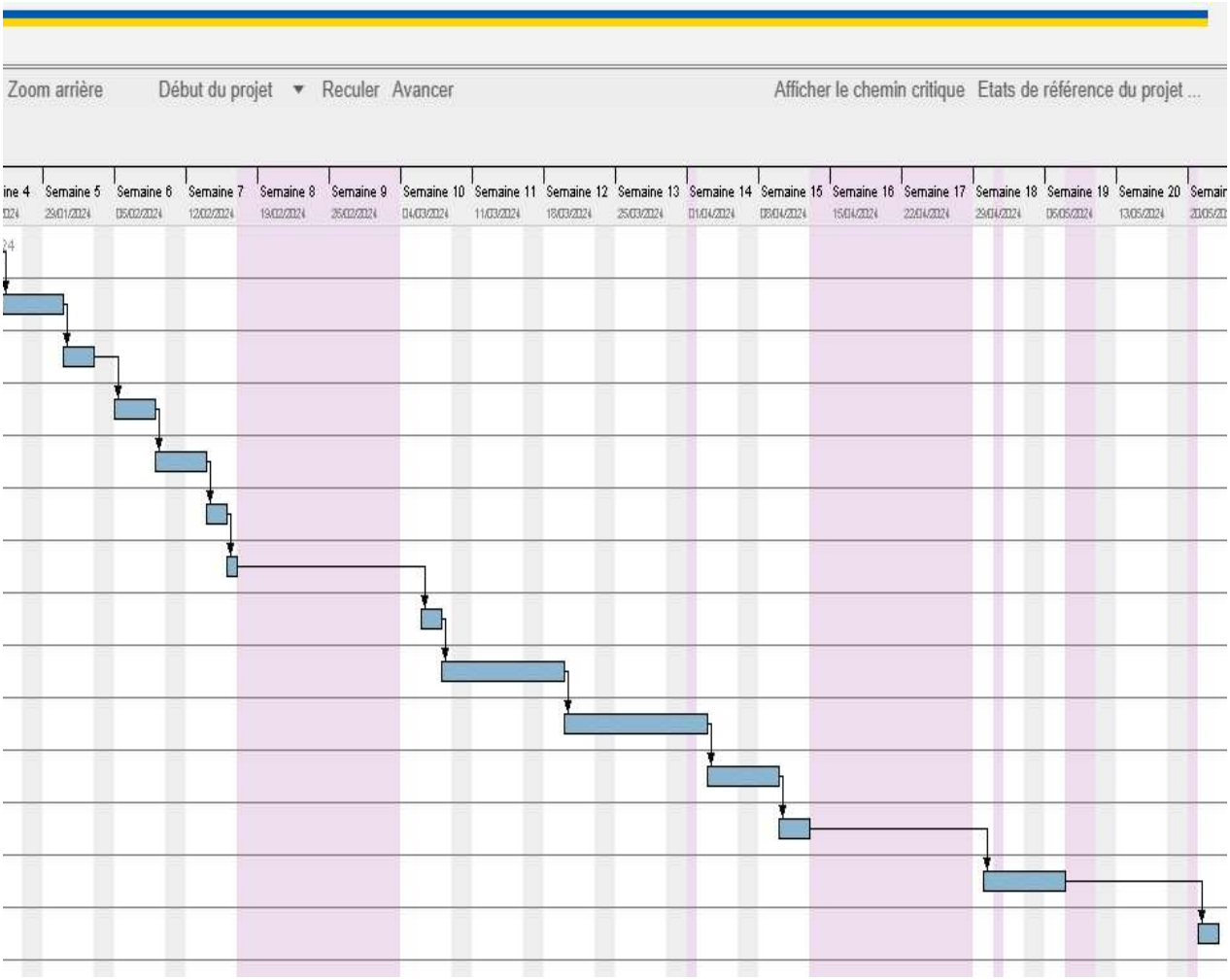


Nom	Date de début	Date de fin
Lancement du projet	10/01/2024	24/01/2024
Création de la base de données et connexion avec Qt	25/01/2024	30/01/2024
Développement des graphiques de température intérieure/extérieure	31/01/2024	02/02/2024
Correction et amélioration du programme	05/02/2024	08/02/2024
Passage sur Ubuntu et installation des logiciels pour développer sur Andr...	09/02/2024	15/02/2024
Reprise du programme précédent pour l'essayer sur Android - Partie 1	16/02/2024	16/02/2024
Reprise du programme précedent pour l'essayer sur Android - Partie 2	05/03/2024	06/03/2024
Développement des classes pour les affichages et la connexion à la bas...	07/03/2024	12/03/2024
Développement de la classes pour recupérer le lien des limages de la ca...	13/03/2024	20/03/2024
Rédaction de la partie commune du dossier (repartion des taches et obje...	21/03/2024	28/03/2024
introduction ma partie personnelle	29/03/2024	02/04/2024
Création des diagrammes UML	03/04/2024	05/04/2024
Rédaction de la mise en place de ma partie	09/04/2024	12/04/2024
Réalisation des tests unitaires	30/04/2024	03/05/2024
conclusion de la partie personelle	07/05/2024	07/05/2024
Relecture et correction du dossier personnelle	21/05/2024	23/05/2024
Création du diaporama et entraînement	24/05/2024	28/05/2024


Ctrl + F
rrière
Début du projet
Reculer
Avancer
Afficher le chemin critique
Etats de référence du projet ...
Semaine 5
Semaine 6
Semaine 7
Semaine 8
Semaine 9
Semaine 10
Semaine 11
Semaine 12
Semaine 13
Semaine 14
Semaine 15
Semaine 16
Semaine 17
Semaine 18
Semaine 19
Semaine 20
Semaine 21

Gantt réel

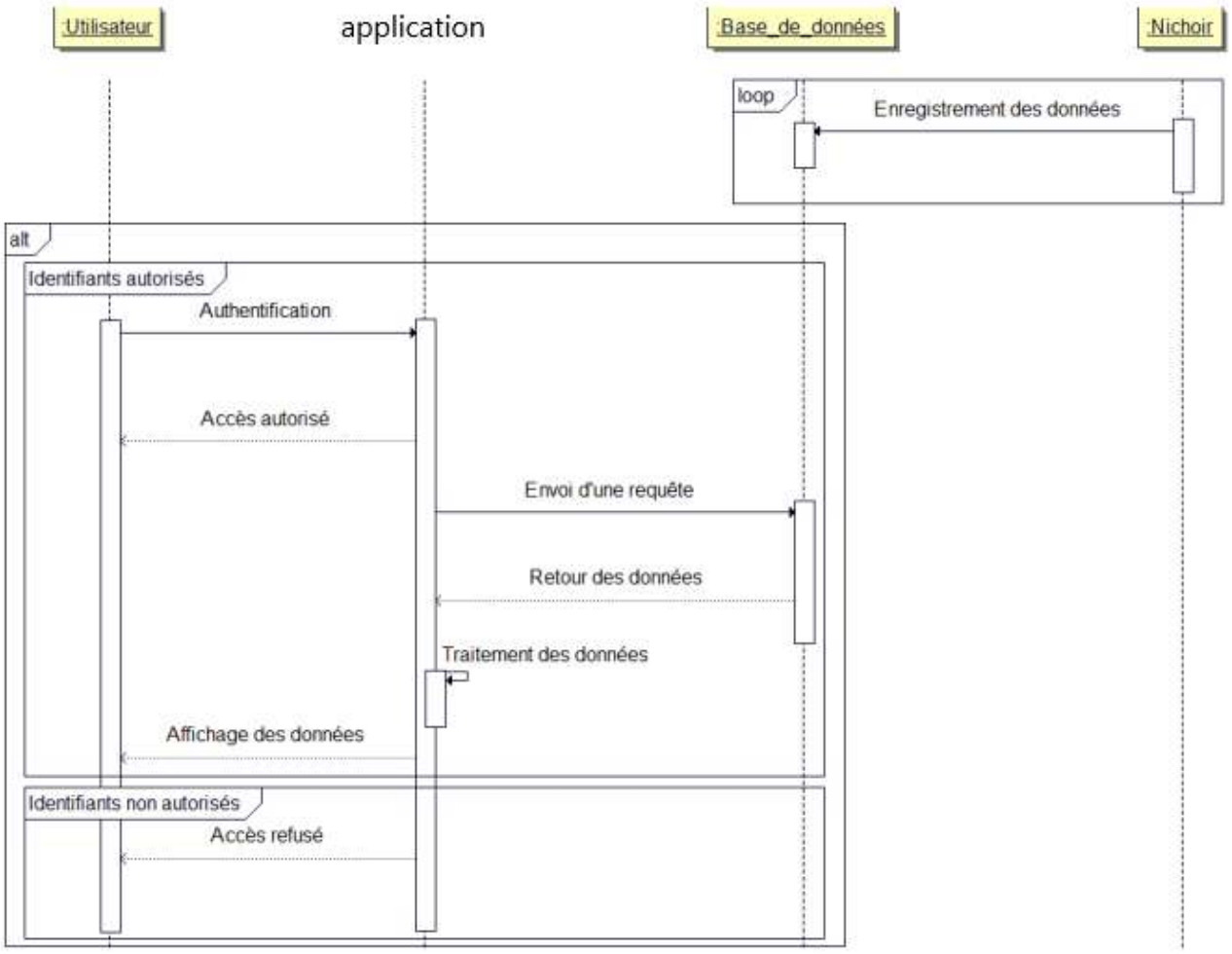
Nom	Date de début	Date de fin
Lancement du projet	10/01/2024	24/01/2024
Création de la base donnnées et connection avec QT	25/01/2024	30/01/2024
Développement du graphique de température intérieure/extérieure	31/01/2024	02/02/2024
Correction et amélioration du programme	05/02/2024	08/02/2024
Passage sur Ubuntu et installation des logiciels pour développer sur Android	09/02/2024	13/02/2024
Reprise du programme précédent pour l'essayer sur Android	14/02/2024	15/02/2024
Reprise du programme précédent sur Windows	16/02/2024	16/02/2024
Dévelloppment des graphiques d'humidité intérieur/extérieur et de poids	06/03/2024	07/03/2024
Développement de la classes pour recupérer le lien des limages de la came...	08/03/2024	19/03/2024
Correction et amélioration du programme	20/03/2024	02/04/2024
Modication du programme pour recupérer les images de la caméra et les aff...	03/04/2024	09/04/2024
Modification du programme pour afficher toutes les récupée sur le même gr...	10/04/2024	12/04/2024
Modification du programme pour afficher toutes les récupée sur le même gr...	30/04/2024	07/05/2024
Modification du programme pour afficher toutes les récupée sur le même gr...	21/05/2024	22/05/2024



Comme vous pouvez le voir ci-dessus, le Gantt prévisionnel est différent du Gantt réel. lors de mon projet j’ai rencontré quelque problèmes techniques.

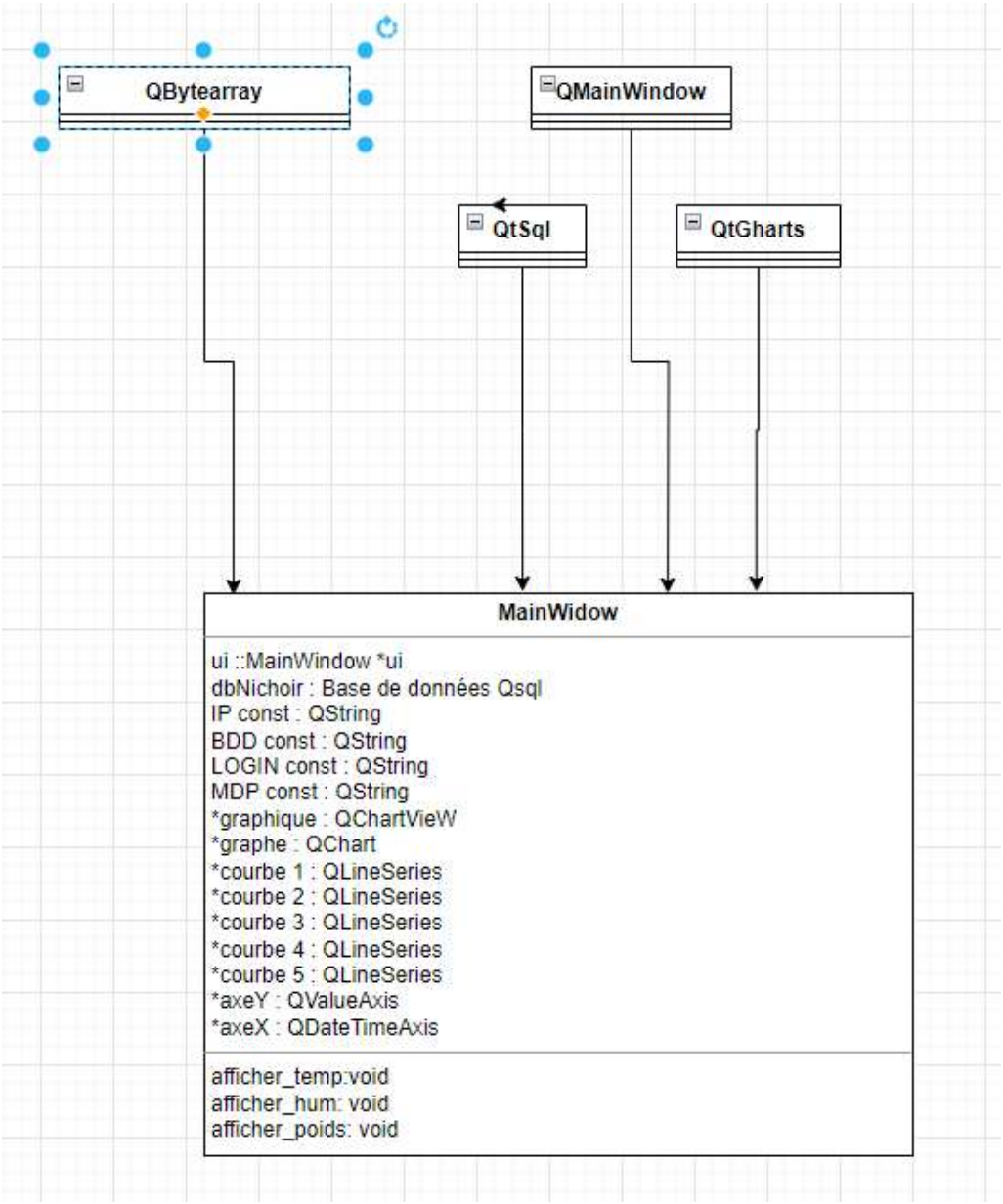
3) UML

3.1) Diagramme de séquence personnelle



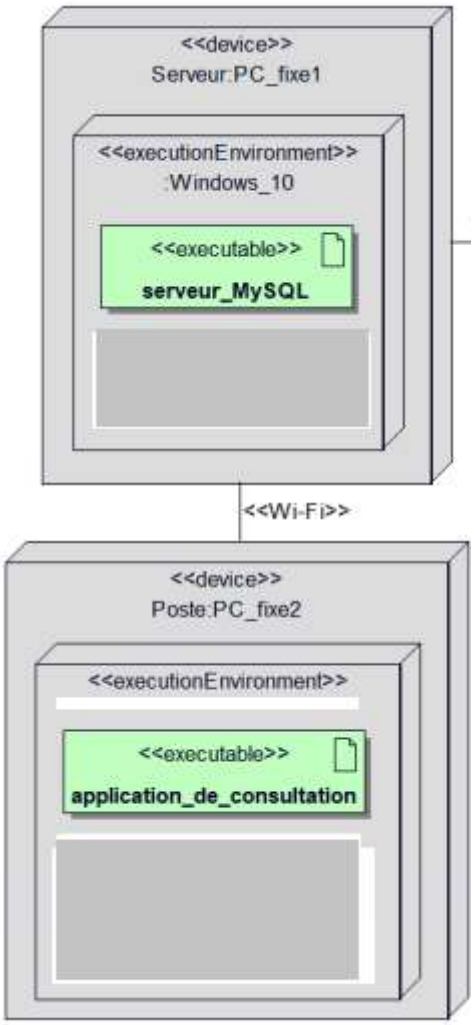
Si l'utilisateur dispose d'identifiants autorisés, il peut accéder aux données, l'aplication envoie alors une requête à la base de données. Une fois récupérées, les données sont traitées puis affichées par l'aplication. À l'inverse, si ses identifiants de l'utilisateur ne sont pas autorisés, l'utilisateur ne peut pas accéder aux données. L'application lui indique que l'accés à la base de donnée refusé.

3.2) Diagramme Classe avec ma partie



Pour crée mon application sur QT j'ai utilisé plusieurs librairies (QsqlDatabase , QmainWindow, QbyteArray, Qtsql, QtCharts), pour recupérer et afficher donnés issues des capteurs est les images des camera depuis la BDD .

3.3) Diagramme de déploiement (mise en avant de ma partie)



Sur un PC fixe, je pourrons exécuter une application de consultation qui se connectera à la base de données en Wi-Fi pour accéder aux données et les afficher.

4) Présentation de ma partie

4.1) Contextualisation

Choix du logiciel qt

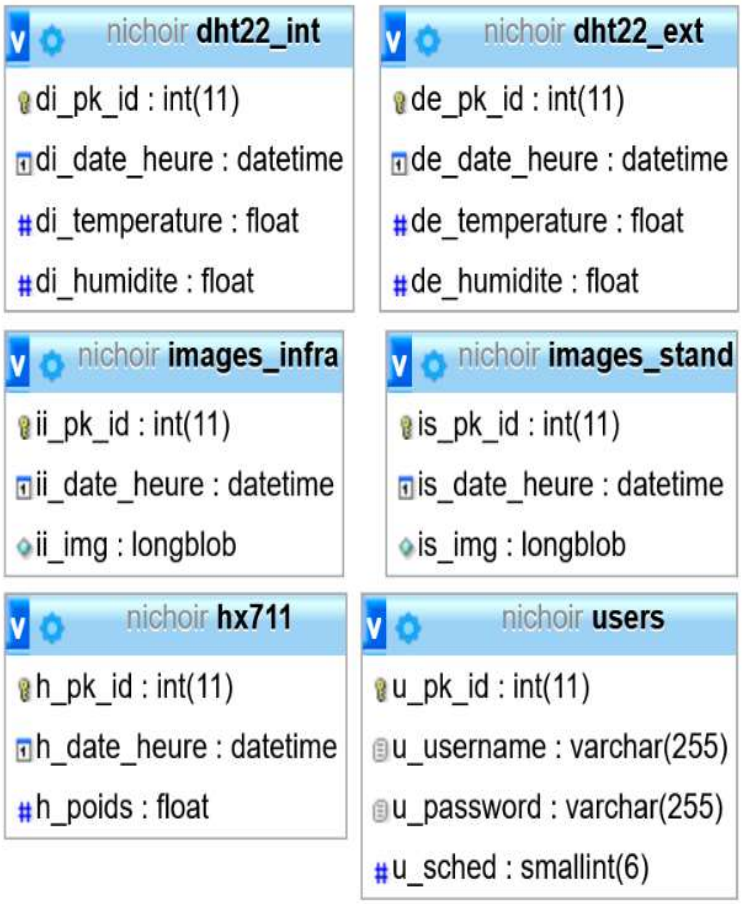
Pour développer mon application j'ai choisi le logiciel qt. Car avec ce logiciel j'ai récupéré les données de la BDD et les afficher en une seule interface. Aussi je peux consulter mon application un fois terminé sans compiler mon programme en la déployant.

Problème rencontré sur qt

Quand j'ai su qui fallait développer mon application sur smartphone. Je l'ai testé mais je n'arrivé pas à me connecter à la base de données donc j'ai décidé continuer le développement de de mon application sur pc. Ensuite la tâche qui consiste à créer la classe pour récupérer les images des caméras depuis le lien et les afficher à durer plus longtemps que prévu. Ensuite économiser l'énergie que consomme la carte rasberry pi mon collègue a décidé de stocker les images dans la base de données. Donc j'ai modifier mon application pour qu'elle récupère les images depuis la base de données .

4.2) Mise en place

Diagramme BDD :



J'ai crée une base de données sur phpmyadmin composées 6 tables (dht22_ext, dht22_int, hx711, images_ext, images_int et users)

- La tables dht22_ext sert à stocker les données envoyées par le capteur de température est humidité extérieur

- La tables dht22_int sert à stocker les donnés envoyées par le capteur de température est humidité intérieur.

- La tables hx711 sert à stocker les données envoyées par le capteur de poids.La tables images_ext sert à stocker les liens d'images envoyés par la caméra intérieure.

- La tables images_int sert à stocker les liens d'images envoyés par la caméra extérieure.

Présentation de la base de donnée :

Table	Action	Lignes	Type	Interclassement	Taille	Perte
dht22_ext	Afficher Structure Rechercher Insérer Vider Supprimer	107	InnoDB	latin1_swedish_ci	16 Kio	-
dht22_int	Afficher Structure Rechercher Insérer Vider Supprimer	121	InnoDB	latin1_swedish_ci	16 Kio	-
hx711	Afficher Structure Rechercher Insérer Vider Supprimer	129	InnoDB	latin1_swedish_ci	16 Kio	-
images_infra	Afficher Structure Rechercher Insérer Vider Supprimer	90	InnoDB	latin1_swedish_ci	3,4 Mio	-
images_stand	Afficher Structure Rechercher Insérer Vider Supprimer	86	InnoDB	latin1_swedish_ci	95,5 Mio	-
users	Afficher Structure Rechercher Insérer Vider Supprimer	2	InnoDB	latin1_swedish_ci	16 Kio	-
6 tables	Somme	535	InnoDB	latin1_swedish_ci	99 Mio	0 o

Comme nous avons vu précédemment cette base de données sert à stocker les données des capteurs, les liens, les images intérieur et extérieur des caméras de surveillance.

Elle est composée de 6 tables :

- 3 qui ont pour fonction de stocker les données de chaque capteur
- 2 qui permettent de stocker les liens des images des caméra int./ext.
- 1 pour stocker les noms des utilisateurs de la base de données

Mise en place de l'application QT

Dans la base de donées que j'ai crée , on peut retrouver la température, l'humidité, le poids, les images des camera (standard et infrarouges) et la date à laquelle ces données ont était prise afin de pouvoir les retrouver plus facilement.

J'ai crée une application sur QT qui servira à récupérer ces données et à les afficher

Pour commencer il faut se connecter à la base de données via le programme QT.

Pour ce faire, j'ai utiliser des driver SQL qui permettent de paramétrer le nom d'utilisateur et le mot de passe ; afin de connecter le nom d'hôte auquel se connecte la base de données où l'on souhaite stocker les données issues des capteurs.

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
     ui->setupUi(this);
     dbNichoir = QSqlDatabase::addDatabase("QMYSQL");
     dbNichoir.setHostName("172.21.28.52");
     dbNichoir.setDatabaseName("nichoir");
     dbNichoir.setUserName("root");
     dbNichoir.setPassword("tssn2.nichoir!");
```

Si le nom d'hôte, le nom d'utilisateur, et le mot passe sont corrects un message est affiché avec la méthode qDebug() indiquant « Ok – ouverture de la base de donnée ».

```
if (dbNichoir.open()) {
   qDebug() << "Ok - ouverture de la base donnée";
   QSqlQuery requete;
```

Si ce n'est pas le cas il affiche une message d'erreur s'affiche avec qDebug(), indiquant « Échec d'ouverture de la base de donnée »

```
else {
    qDebug() << "Echec d'ouverture de la base de donnée";
    qDebug() << dbNichoir.lastError();
}
```

Une fois connecter à la base de donnée il faut faire des requêtes pour recupérer les données stocker dans la base de données.

Si la requêtes s'exécute avec sucées, un message est afficher avec qDebug() indiquant « Ok – requête » .

Ensuite, une boucle while est utilisé pour parcourir les enregistrements renvoyés par la requête. Les valeurs de chaque champ de l'enregistrement de l'enregistrement sont affichées avec qDebug().

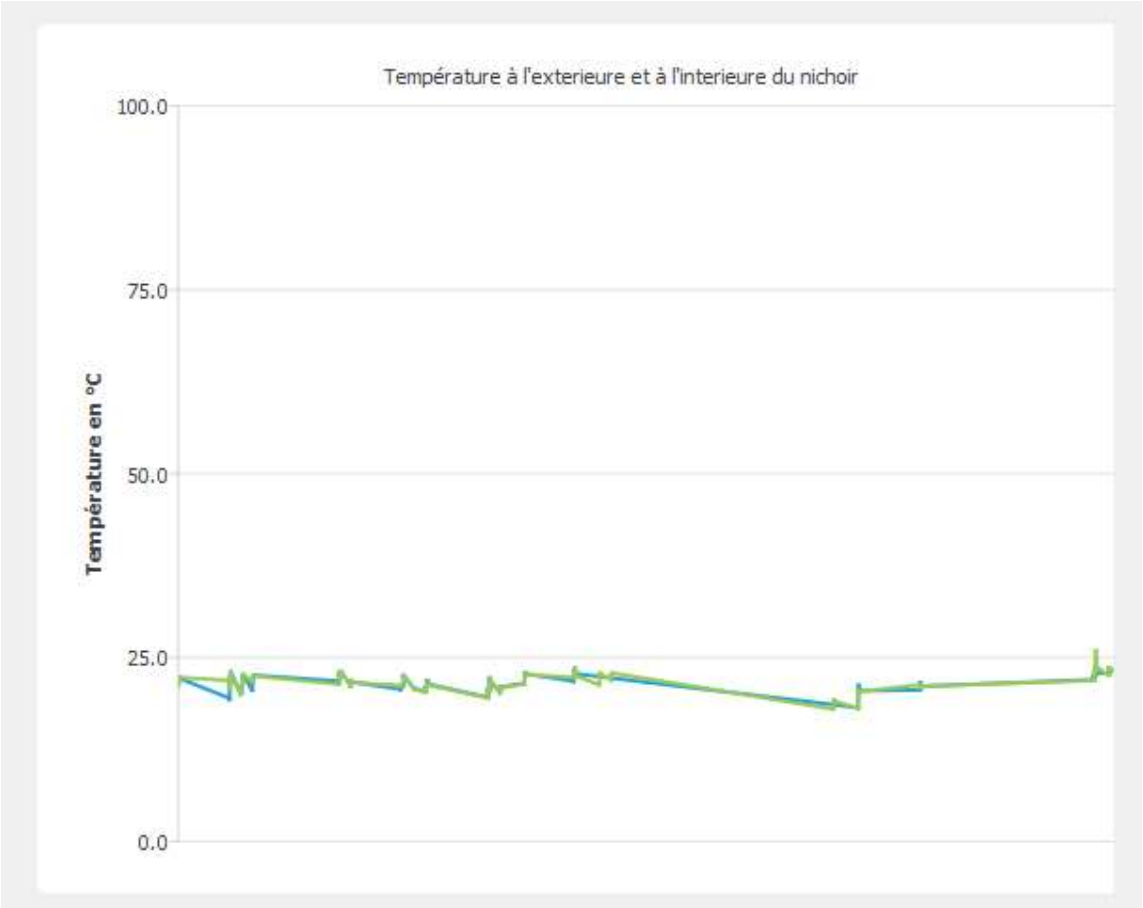
```
if(requete.exec("SELECT * FROM dht22_ext")) {
    qDebug() <<"Ok - requete";
```

```
while(requete.next()) {
    qDebug() << requete.value("de_temperature") <<requete.value("de_date_heure") <<requete.value("de_humidite");
```

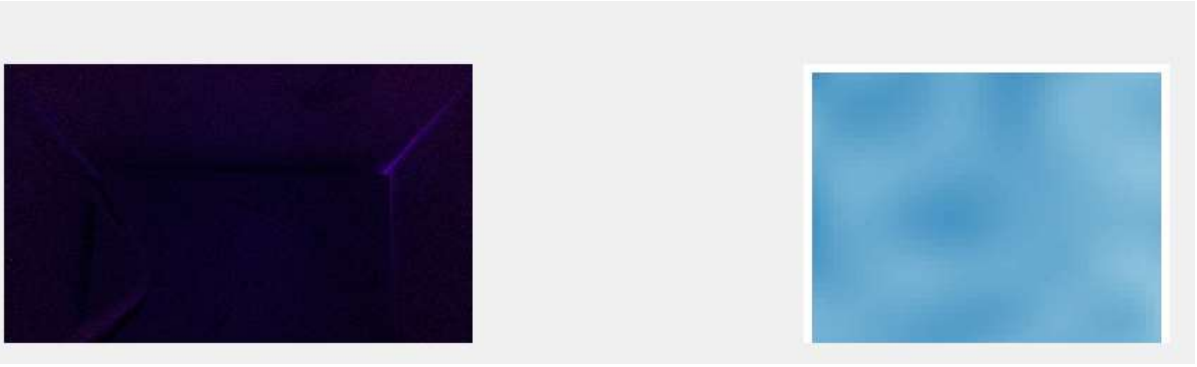
Si la requête échoue, un message est affiché avec qDebug(), indiquant « Échec à la requête ». La méthode lastError() de l'objet requête est utilisée pour afficher des informations détaillées sur l'erreur.

```
else {
    qDebug() << "Echec d'ouverture de la base de donnée";
    qDebug() << dbNichoir.lastError();
}
```

Une fois les données récupérées je les l'utilise pour crée des courbes pour ensuite les afficher sous forme de graphique.

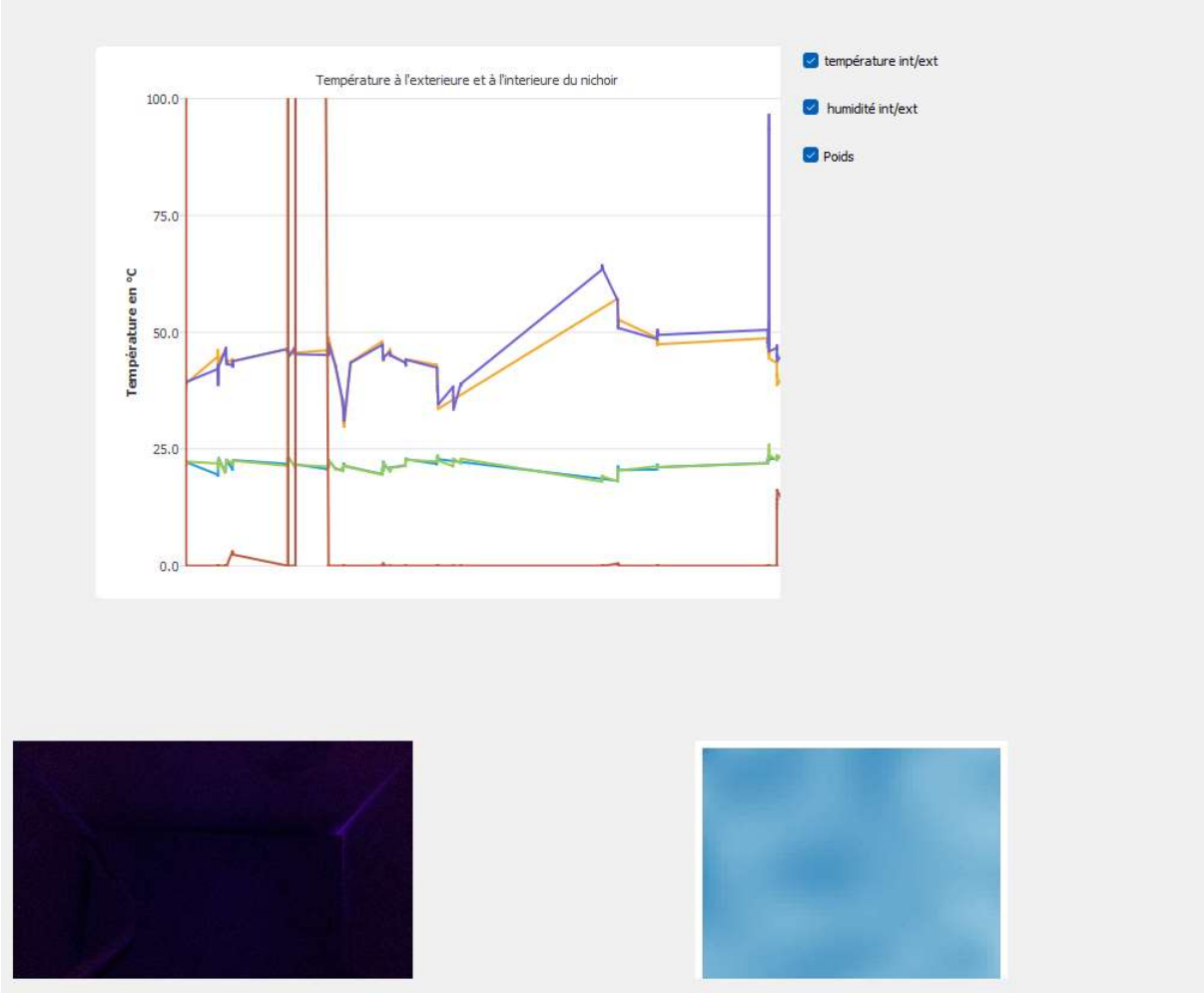


Pour afficher les images des cameras du nichoir c'est un peu différent : Je récupère la dernière image des deux camera puis ensuite je les affiche sur l'interface graphique que j'ai crée pour développer mon application.



4.3) Finalités

Quand l'application était à peu près terminée il y avait un dernier petit problème: pour passer d'un graphique à un autre je devais compiler mon programme à chaque fois donc pour régler ce problème j'ai affiché toutes les données que j'ai récupéré sur le même graphique.



5) Réalisation des tests unitaires

N°	ACTION	ATTENDU	RESULTAT
1	L'Identifiant et le mot de passe pour me connecté à la base de donner sont correcte	Ouverture de la Base de données sinon Échec d'ouverture à la base de données	Ok
2	La requête pour recupérer les données	Ok requête donner afficher sinon Échec de la requête	Ok
3	Affichage de donné de température	Les donné de la température sont affichées sous forme graphique	Ok
4	Affichage de donné d'humidité	Les données de l'humidité afficher sont sous forme graphique	Ok
5	Affichage de donné du poids	Les donné du poids sont afficher afficher sous forme graphique	Ok
6	Affichage des imges des cameras standard et infrarouges	Les dernières images des caméras sont affichées sur l'interface graphique	Ok

6) Conclusion

En conclusion, Ce projet de 5 mois ma permis mes en pratique les connaissances que j'ai acquis tous au long de mes 2 année de BTS ainsi que développer de nouvelle.

De plus, il m'a fait évoluer au niveau de ma timidité et a développé mes capacités d'échange et de communication. Le travail en collaboration avec mes collègues à été très bénéfique pour moi, j'ai grandement apprécié la cohésion et l'esprit d'équipe dont nous avons fait preuve pendant ce projet. Je souhaites par la suite mes résultat me le permettent de poursuive dans ce domaine en BUT Informatiques.