Rapport Projet Ingénieur :
Plateforme de TP à distance

Encadrants : Alexandre Boé, Xavier Redon et Thomas Vantroys

Louis Wadbled
SE 5 - SC

Sommaire

Contexte

La crise de la Covid-19 et les différents confinements ou cours à distance ont montré les limites des outils disponibles pour de l'enseignement à distance ou pour les enseignements en dehors des temps dédiés. Par exemple, il est extrêmement difficile de réviser un TP ou de le terminer en dehors des heures d'enseignement car l'accès aux salles est compliqué, notamment les soirs ou les week-end.

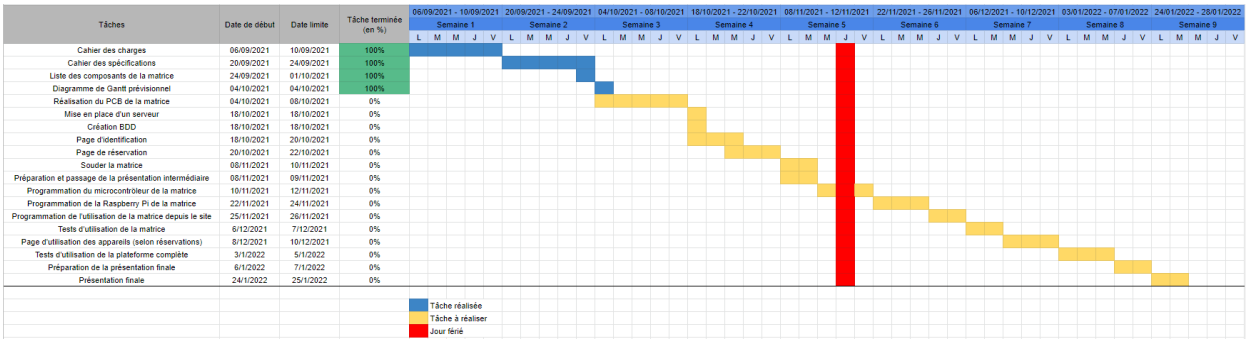
Objectif du projet

L'objectif du projet est de réaliser un POC de la plateforme pour montrer qu'il est possible de réaliser des TP d'électronique à distance. Le POC doit contenir les éléments suivants :

- Une plateforme Web complète avec : une connexion sécurisée, un planning de réservation pour les utilisateurs afin d'utiliser les appareils sur des créneaux définis, un système de visualisation et de commande des appareils.
- Un système physique composé d'une matrice de commutation qui permet de relier des appareils de mesures à des circuits de TP. Toute cette partie constitue ce que j'appellerai un banc d'observation. La matrice doit pouvoir être pilotée depuis l'interface Web pour permettre aux utilisateurs de choisir quel circuit ils veulent observer.
- Lors de l'utilisation d'un banc d'observation sur l'interface par un utilisateur, ce dernier doit être le seul à pouvoir utiliser le banc pendant son créneau.

Organisation du travail

Étant seul sur le projet, l'organisation est assez simple. En début de projet, j'ai réalisé un diagramme de Gantt en résumant les tâches à réaliser. Je place évidemment les tâches à réaliser en priorité avant les autres tâches.



Tâches	Date de début	Date limite	Tâche terminée (en %)
Cahier des charges	06/09/2021	10/09/2021	100%
Cahier des spécifications	20/09/2021	24/09/2021	100%
Liste des composants de la matrice	24/09/2021	01/10/2021	100%
Diagramme de Gantt prévisionnel	04/10/2021	04/10/2021	100%
Réalisation du PCB de la matrice	04/10/2021	08/10/2021	0%
Mise en place d'un serveur	18/10/2021	18/10/2021	0%
Création BDD	18/10/2021	18/10/2021	0%
Page d'identification	18/10/2021	20/10/2021	0%
Page de réservation	20/10/2021	22/10/2021	0%
Souder la matrice	08/11/2021	10/11/2021	0%
Préparation et passage de la présentation intermédiaire	08/11/2021	09/11/2021	0%
Programmation du microcontrôleur de la matrice	10/11/2021	12/11/2021	0%
Programmation de la Raspberry Pi de la matrice	22/11/2021	24/11/2021	0%
Programmation de l'utilisation de la matrice depuis le site	25/11/2021	26/11/2021	0%
Tests d'utilisation de la matrice	6/12/2021	7/12/2021	0%
Page d'utilisation des appareils (selon réservations)	8/12/2021	10/12/2021	0%
Tests d'utilisation de la plateforme complète	3/1/2022	5/1/2022	0%
Préparation de la présentation finale	6/1/2022	7/1/2022	0%
Présentation finale	24/1/2022	25/1/2022	0%

Figure 1 : Diagramme de Gantt prévisionnel.

A la fin du projet, le diagramme de Gantt est rempli de la manière suivante :

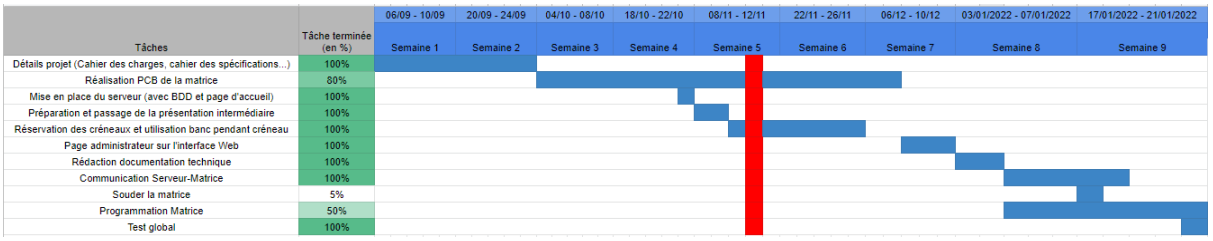


Figure 2 : Diagramme de Gantt (simplifié) à la fin du projet.

Pendant la réalisation du projet, certaines tâches ont été beaucoup plus longues que prévu, notamment la réalisation du PCB, ce qui a décalé certaines tâches liées à la matrice. Je savais que j'étais en retard sur la conception du PCB, alors je travaillais en parallèle sur l'interface Web pour essayer de prendre le moins de retard possible sur cette partie. Comme je me sentais plus à l'aise dans mon travail sur l'interface Web, j'ai réussi à éviter du retard sur cette partie également et donc bien avancer sur le projet.

Développements techniques

Les développements techniques sont divisés en trois parties. Tout d'abord, je vais présenter mon travail sur la réalisation de la matrice, puis la réalisation de l'interface Web et enfin la communication entre la matrice et le serveur.

Le système souhaité est visible sur la figure 3.

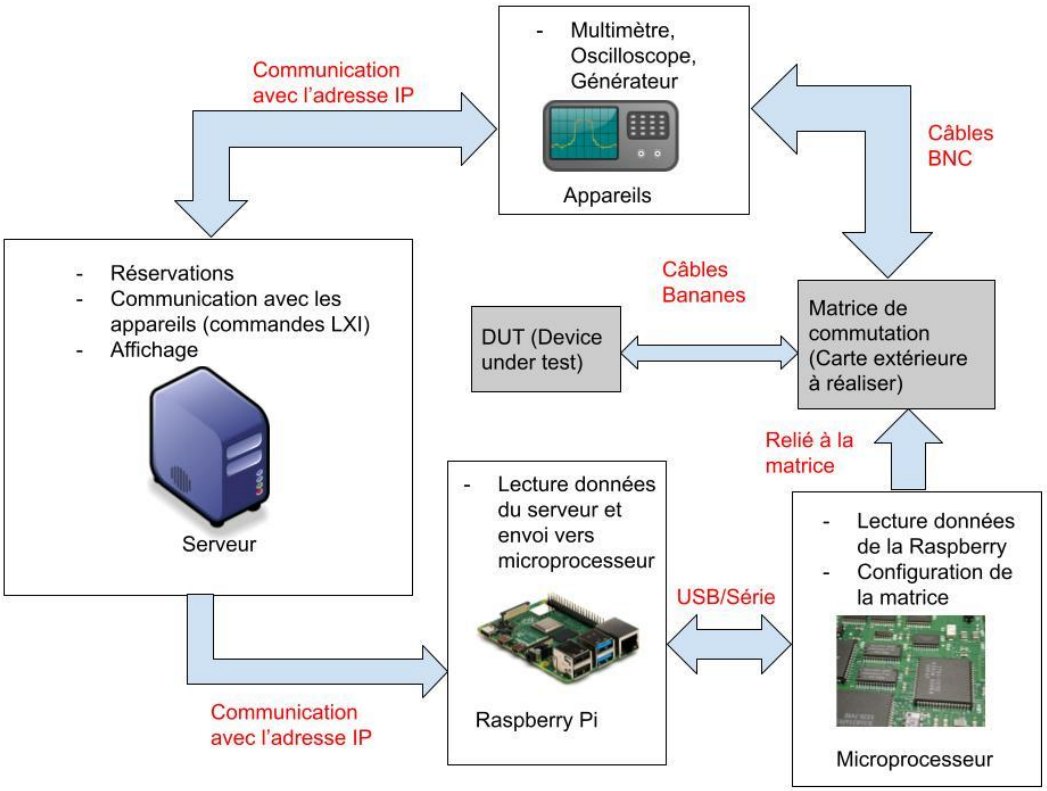


Figure 3 : Système prévu à mettre en place pour le fonctionnement du projet.

1) Réalisation de la matrice

Afin de choisir les appareils de mesures et les circuits de TP à relier sur les bancs d'essais, une matrice de commutation est nécessaire. Le principe est simple, d'un côté on veut pouvoir connecter des entrées (appareils de mesures) avec des connecteurs BNC, et de l'autre des sorties (circuits de TP) avec des fiches Banane. Pour pouvoir relier ces entrées et sorties, il faut utiliser des relais. Ces relais permettent un contact entre les entrées et sorties selon la tension dans la bobine des relais. J'utilise en plus une diode de roue libre afin de protéger un minimum le circuit. Pour commander les relais, il est alors nécessaire d'utiliser des transistors MOSFET, qui agissent comme un interrupteur entre le drain et la source selon la tension dans la grille. Il faut alors pouvoir commander la tension de la commande du transistor. Pour cela, on va utiliser une carte Nucleo, que l'on pourra programmer pour commander les transistors selon les choix des utilisateurs. Cela amène l'utilisation de connecteurs (Header) pour lier les pistes de la matrice avec la Nucleo. Et de l'autre côté pour lier la Nucleo avec le serveur, j'utiliserai une Raspberry Pi entre le serveur et la Nucleo. Afin de réaliser le PCB sur une face, j'utilise en plus de tous les composants des résistances 0 Ohm qui permettent de passer au-dessus de certaines pistes. Avec toutes ces informations, je peux imaginer la conception du PCB sans avoir besoin des références du matériel.

Maintenant que la liste des composants nécessaires est réalisée, il faut choisir les références pour pouvoir les commander. Tout d'abord concernant la carte Nucleo, j'utilise une Nucleo possédant 144 contacts afin d'être sûr de ne pas manquer de broches pour commander les transistors. Ensuite, le courant maximum du transistor doit être supérieur au courant de la bobine du relais. Les deux composants sont alors choisis selon ce critère. Pour la diode, on préfère une capacité de surcharge de courant élevée.

Une fois les références des composants choisis, je peux commencer à réaliser le PCB de la matrice. Pour commencer, je choisis le logiciel Altium Designer, puisque je me sentais à l'aise pour évoluer sur ce logiciel. Ensuite, il me faut utiliser les schémas et empreintes des différents composants. Pour cela, je trouve un site [1] très utile répertoriant les empreintes de plusieurs composants dont j'ai besoin. Un exemple de schéma d'une entrée est disponible en *Annexe 1*. Pour les empreintes manquantes, j'ai contacté le support du site qui a pu me les fournir en moins de deux jours. Ensuite avec tous les schémas et toutes les empreintes réunies, j'ai pu commencer à travailler sur le PCB de la matrice.

J'ai tout d'abord réalisé un PCB extrêmement grand (environ 38 cm par 48 cm). Le but n'était pas d'avoir une taille fixée sur le PCB, mais de pouvoir le graver à Polytech (donc une taille de maximum 22,9 cm par 29 cm). Pour cela, j'ai donc évidemment dû réduire la taille de la matrice, ce qui m'a pris énormément de temps par rapport à ce que j'avais prévu. Au final, elle mesure 22 cm par 26,5 cm.

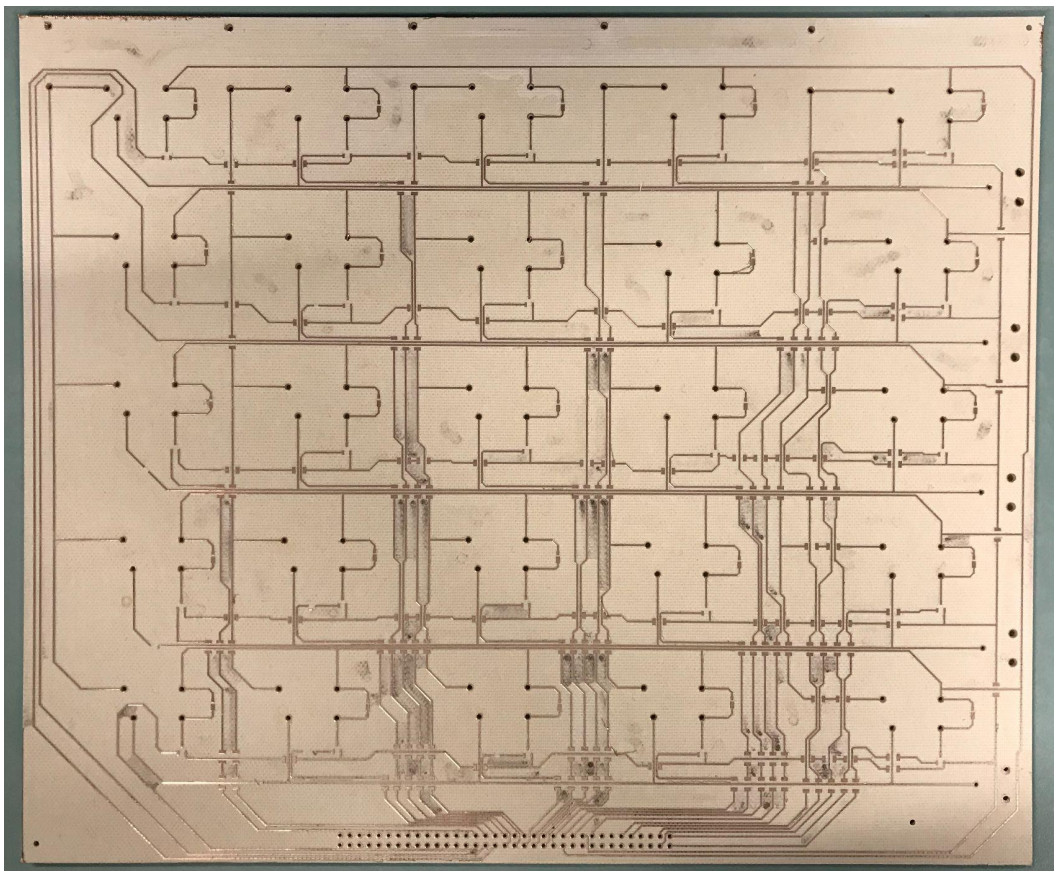


Figure 4 : PCB gravé de la matrice.

Après avoir récupéré la matrice gravée, je savais que je n'aurais pas le temps de souder tous les composants. J'ai alors décidé de souder uniquement de quoi démontrer que le contact entre une entrée et une sortie était possible. En effet, le schéma est toujours le même entre chaque entrée et sortie, donc selon moi, si le contact est possible entre Pour cela, je soude un connecteur BNC (l'entrée), un relais, une diode, un transistor, plusieurs résistances 0 Ohm et le connecteur vers la carte Nucleo. Je ne peux pas souder la fiche Banane (la sortie) car le composant est plus grand que l'empreinte trouvée sur internet. Pour tester uniquement le contact ce n'est pas grave puisque je peux simplement placer des pointes de touches d'un part et d'autre de l'interrupteur du relais.

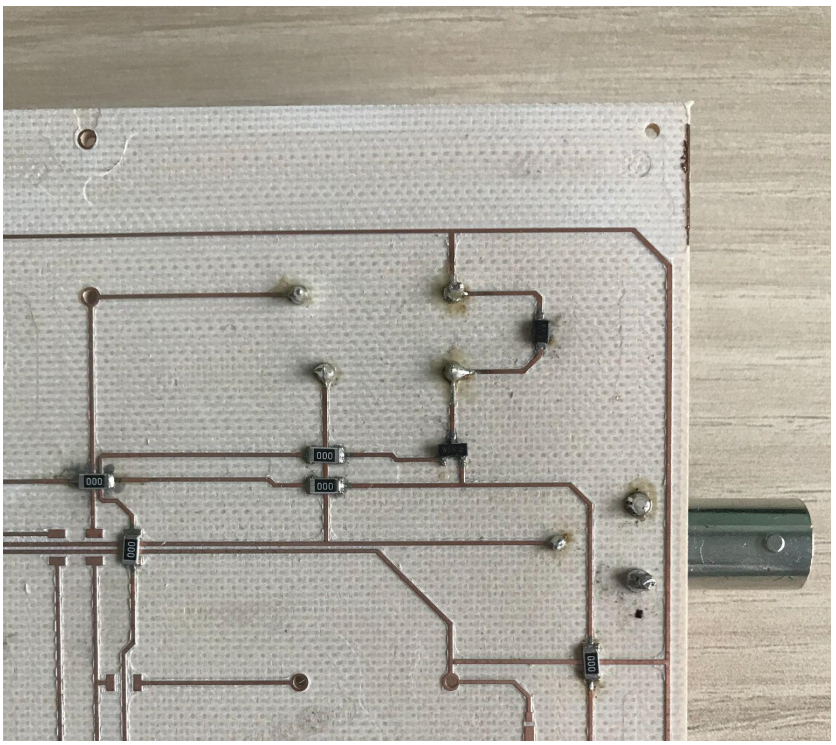


Figure 5 : Soudure d'un relais pour l'essai d'un contact entre une entrée et une sortie de la matrice.

En branchant la masse et la broche de 5V de la Nucleo sur la matrice, ainsi que la masse sur la commande du transistor, il n'y a pas de contact entre l'entrée et la sortie. C'est normal puisque l'interrupteur du relais est naturellement ouvert. En appliquant ensuite une tension sur le transistor, l'interrupteur se ferme. Le contact fonctionne alors correctement entre une entrée et une sortie, et j'imagine que c'est possible pour tous les autres contacts, mais par manque de temps je n'ai pas pu souder toute la matrice.

2) Réalisation de l'interface Web

La réalisation de l'interface Web nécessite tout d'abord l'utilisation d'un serveur, j'utiliserai alors un serveur Apache2, étant donné que c'est très simple à installer sous Linux. Les langages que j'utilise sont : PHP, JavaScript et HTML5 parce que je connais ces langages donc ce sera plus simple pour moi d'évoluer sur ceux-ci, et en plus une partie de l'interface Web a déjà été réalisée avec ces langages. J'aurai besoin de mettre en place une base de données pour réaliser le système de connexion et de réservation, alors j'utiliserai une base de données MySQL puisque j'ai déjà utilisé ce type de base de données dans mon cursus.

Concernant la base de données, je crée différentes tables pour contenir les informations des identifiants et des réservations. J'associe alors un identifiant avec un banc et un créneau pour former une réservation. Pour utiliser un banc sur l'interface, j'ai besoin d'avoir les informations des appareils (notamment leur type et leur IP pour afficher l'appareil correctement et communiquer avec lui). Des tables sont créées en conséquence pour contenir ces informations.

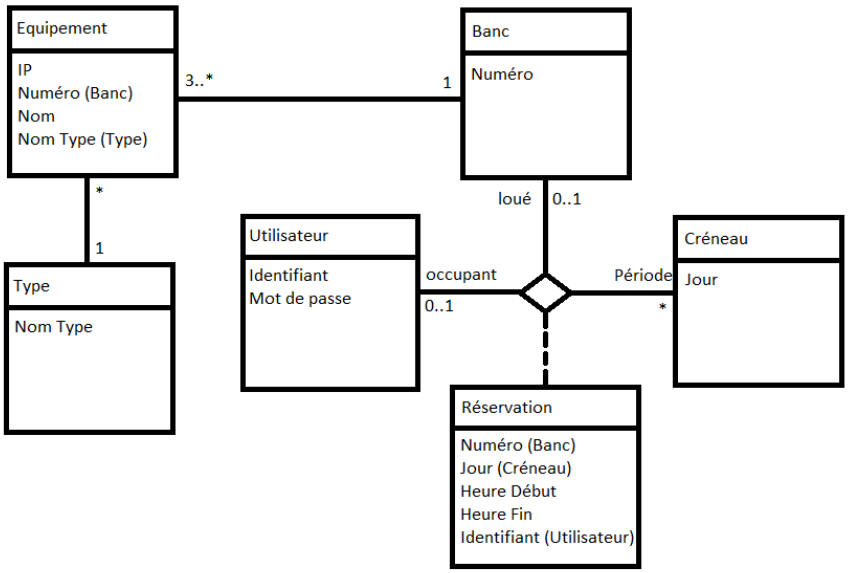


Figure 6 : Diagramme UML de la base de données.

Le diagramme UML peut être lu de la manière suivante : on associe à un banc un minimum de 3 équipements. Un équipement ne peut être associé qu'à un unique banc. Un équipement possède un type et un type peut être utilisé par une infinité d'équipement. Un utilisateur peut réserver un banc sur un créneau.

La plateforme Web est utilisée de la manière suivante : tout d'abord l'utilisateur se connecte avec son identifiant et son mot de passe. Une fois connecté, il a accès au menu de la plateforme où il lui est proposé de réserver un créneau pour un banc d'essai, ou alors consulter ses réservations.

Figure 7 : Menu après la connexion à la plateforme.

S'il veut réserver un créneau, il faut choisir différents critères comme la date, le banc et l'heure du créneau souhaité. Il n'est pas possible pour l'utilisateur de réserver un créneau déjà passé ou déjà réservé par quelqu'un d'autre. Lorsque l'utilisateur a choisi les critères qu'il souhaite et réserve le créneau, toutes les informations sont entrées dans la base de données dans la table réservation.

Figure 8 :Réservation d'un appareil.

S'il veut consulter ses réservations, on parcourt la table réservation de la base de données et on affiche les créneaux qui correspondent à l'utilisateur et qui ne sont pas déjà passés. On affiche les informations concernant le créneau pour l'utilisateur (le banc choisi, la date et l'horaire du créneau). L'utilisateur peut ensuite supprimer une réservation s'il le souhaite ou accéder au banc si la date et l'heure du créneau correspond avec la date et l'heure actuelle.

Figure 9 : Consultation des réservations.

Lorsque l'utilisateur accède au banc, il peut visualiser et commander les appareils qui sont associés au banc. Cette partie a déjà été réalisée pour deux types d'appareils (Oscilloscope et générateur de signaux). La façon de reçevoir les données des appareils sur cette page au départ n'est pas celle que j'utilise à la fin du projet. En effet la partie visualisation et commande des appareils n'utilisait pas de base de données mais recevait toutes les informations depuis une méthode POST en HTML (c'est-à-dire que les informations provenaient de la page précédente directement). Dans mon cas, je n'ai que le numéro du banc associé à la réservation, donc je parcours la base de données pour ensuite afficher les bons appareils et permettre à l'utilisateur d'utiliser l'ensemble du banc. Lors de la fin du créneau, l'utilisateur voit une alerte s'afficher sur l'écran pour lui communiquer que le créneau est terminé, et ensuite il est directement renvoyé vers le menu principal.

Pour communiquer avec les appareils, nous avons besoin de l'adresse IP de l'appareil, du paquetage 'lxi-tools' installé sur le serveur, ainsi que des commandes SCPI des appareils que l'on peut trouver dans leur documentation. Pour communiquer les commandes aux appareils, avec ces informations, on peut utiliser la fonction PHP 'shell_exec' pour réaliser les commandes et prendre des captures d'écran des appareils pour les afficher sur l'interface.

Pour avoir des bancs d'observation complets, j'ajoute deux nouveaux appareils : la Matrice et un Multimètre. Pour cela, un protocole d'ajout d'appareils existe déjà, donc en suivant le protocole, je peux ajouter ces types d'appareils. Pour le multimètre, il n'existe que très peu de documentation, alors la seule commande possible mise en place sur l'interface est le changement d'unité. Pour la matrice, l'interface n'est pas très intuitive pour l'utilisateur, mais est fonctionnelle. Pour les commandes, pas besoin du paquetage LXI, j'utilise du Python pour envoyer la commande à la Raspberry Pi, ce que j'expliquerai dans la partie 3).

Pour permettre de gérer plus facilement les appareils, les bancs et les réservations dans la base de données, je crée un compte administrateur sur l'interface. Plutôt que de devoir se connecter à la base de données directement où il faut connaître la syntaxe du langage SQL pour modifier des paramètres de la base de données, on peut simplement se connecter sur l'interface avec le compte administrateur. Il est possible de supprimer/ajouter des appareils et des bancs, et pour les réservations, on peut consulter les réservations passées, supprimer des créneaux ou les rendre disponible à la réservation.

Figure 10 : Menu de connexion en tant qu'administrateur.

Au total, tout ceci représente environ 1100 lignes de codes écrites pour l'interface Web.

3) Communication Matrice-Serveur

Pour commander la matrice, il faut pouvoir envoyer les données directement vers la carte Nucleo qui devra commander la matrice. Pour cela, on ajoute une Raspberry Pi entre la Nucleo et le serveur. Pour la communication entre ces deux éléments, je réalise ceci en Python avec des sockets en UDP. Le principe est que la Raspberry Pi va attendre de

recevoir des données sur un port défini de son adresse IP. Etant donné que dans la base de données on connaît l'adresse IP de la Raspberry, on peut alors envoyer des données en Python depuis le serveur. Les codes Python sont disponibles en annexe.

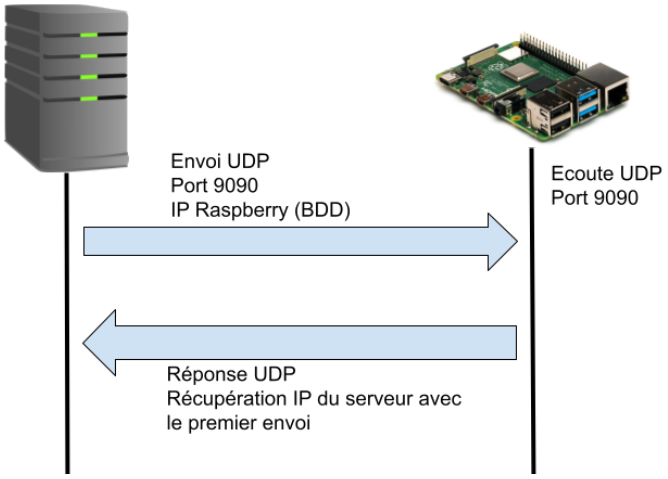


Figure 11 : Schéma du fonctionnement de la communication entre le serveur et la Raspberry Pi.

Le serveur envoie une chaîne de 25 caractères (chaque caractère correspondant à un relais) selon les données que l'utilisateur a choisi sur l'interface. Les caractères sont soit 0 soit 1 selon l'état voulu du relais (0 = ouvert et 1 = fermé).

Il faut ensuite envoyer ces données à la carte Nucleo. On peut le faire en Python depuis la Raspberry en utilisant une communication série USB.

La programmation de la Nucleo et la soudure de l'ensemble de la matrice manquent pour pouvoir faire fonctionner l'ensemble de la plateforme.

Prise de recul sur le travail

Si le projet était à refaire, je pense que je me serais concentré plus vite sur la réalisation du PCB de la matrice, tout d'abord parce que cela m'a pris énormément de temps comparé avec ce que j'avais prévu, et aussi parce que j'aurai pu réaliser une matrice beaucoup plus compact et donc plus facile à manipuler, même si celle réalisée semble fonctionner correctement. De plus, je pense qu'avec une semaine de plus, j'aurais pu terminer le projet, alors je pense que sans le retard accumulé sur la conception du PCB, l'ensemble des points du cahier des charges auraient pu être remplis. Hormis ceci, je pense que tous les autres points du cahier des charges ont été remplis.

Conclusion

Pour résumer, toute l'interface Web est fonctionnelle et tous les points du cahier des charges lui correspondant ont été réalisés (connexion, réservation, utilisation du banc et la partie administrateur). Concernant la matrice de commutation, le PCB a été réalisé et un test sur le contact entre une entrée et une sortie de la matrice a été validé. La communication

entre le serveur et la Raspberry Pi est réalisée. Pour terminer cette partie et valider l'entièreté du cahier des charges, il manque la programmation de la carte Nucleo et la soudure des composants restants.

Pour conclure, le projet m'a permis de développer mes compétences techniques concernant les différents langages que j'ai pu utiliser (HTML5, PHP, JavaScript) et concernant la conception de PCB. J'ai également pu travailler sur mon autonomie et mon organisation pour avancer au maximum sur le projet.

Bibliographie

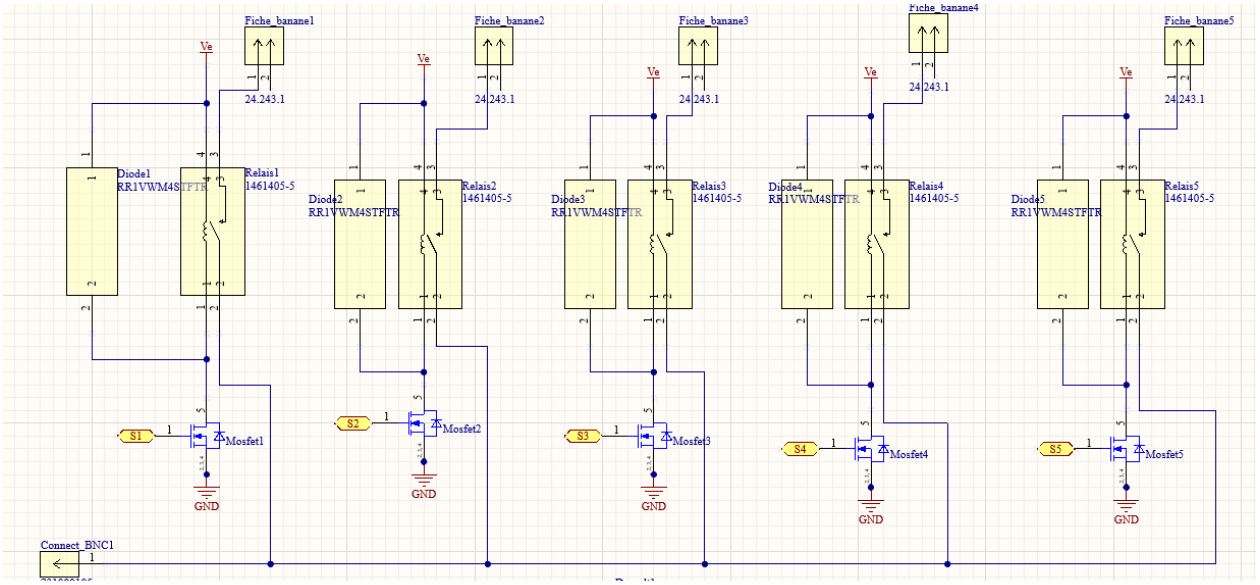
[1] https://www.ultralibrarian.com/

Page Wiki : https://projets-ima.plil.fr/mediawiki/index.php/IMA5_2021/2022_P28

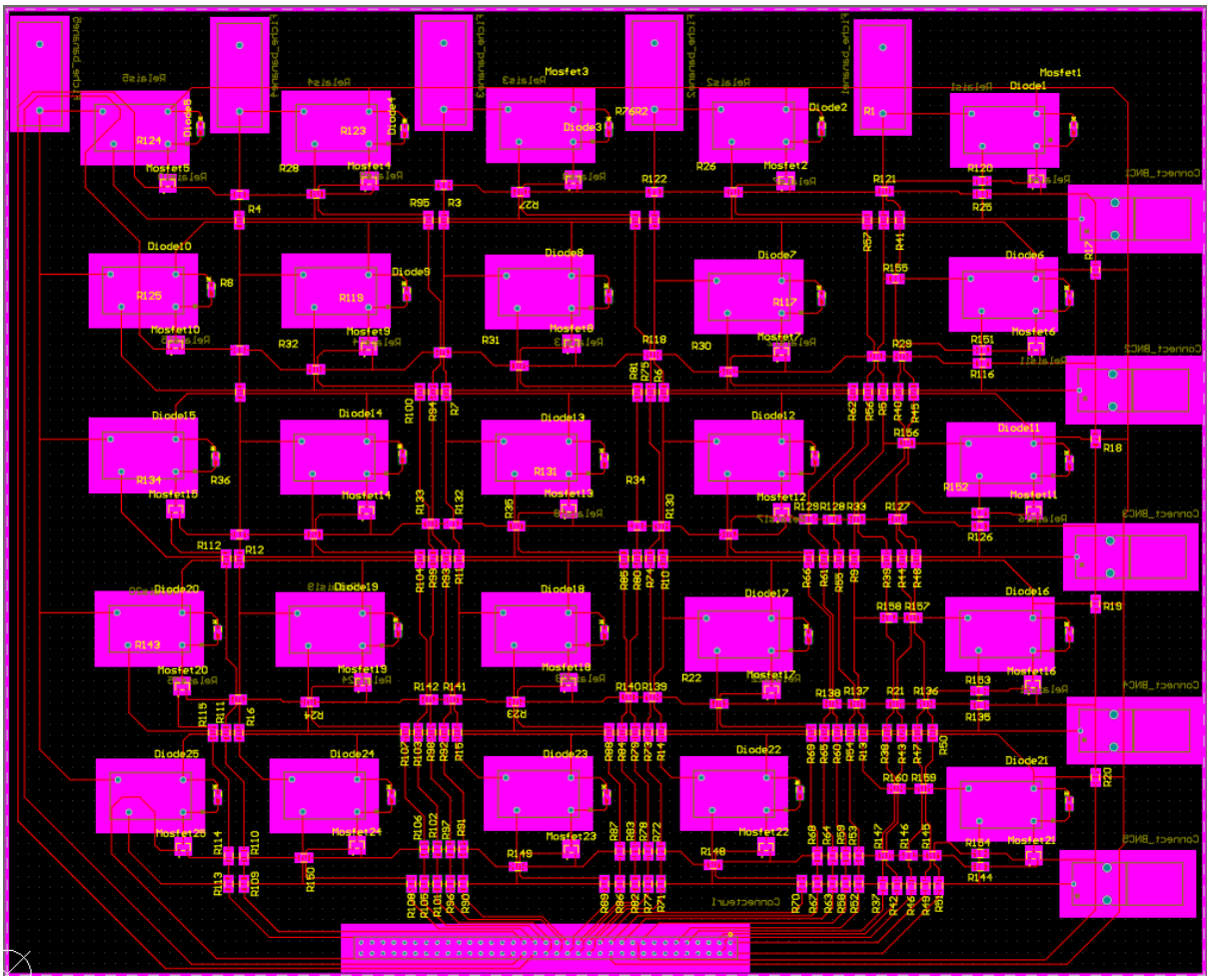
Dépôt GIT contenant les codes du projet :
https://archives.plil.fr/lwadbled/projet_ingenieur_lwadbled

Liste des figures

Annexes



Annexe 1 : Exemple du schéma entre une entrée et cinq sorties.



Annexe 2 : PCB de la matrice sous Altium Designer.

Annexe 3 : Affichage de la gestion des bancs par l'administrateur.

Annexe 4 : Affichage de la gestion des appareils par l'administrateur.

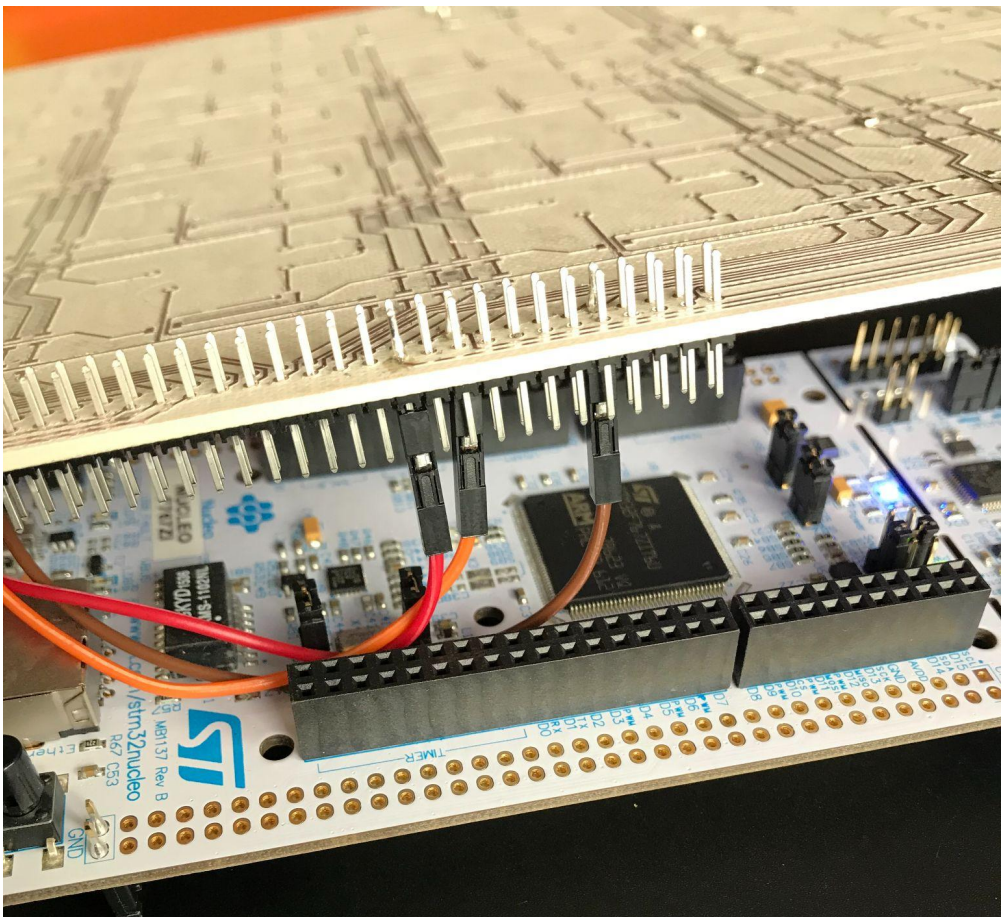
Annexe 5 : Affichage de la gestion des réservations par l'administrateur.

```
import socket
import serial

if __name__=='__main__':
        serverSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM);
        serverSocket.bind(("172.26.145.62",9090));
        ser = serial.Serial("/dev/ttyAMA0",9600);
        while(True):
                        dataFromClient, sourceAddress = serverSocket.recvfrom(1024);
                        print("Data from %s is %s"%(sourceAddress,dataFromClient.decode()));
                        serverSocket.sendto("Communication réalisée".encode(),sourceAddress);
                        ser.write(dataFromClient);
```

Annexe 6 : Code Python de la Raspberry.

```
import socket
import sys

if __name__ == '__main__':
        clientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM);
        data = sys.argv[2]; # sys.argv[2] = Données à envoyer à la Raspberry
        clientSocket.sendto(data.encode(),(sys.argv[1],9090)); # sys.argv[1] = IP de la Raspberry
        dataFromServer,sourceAddress = clientSocket.recvfrom(1024);
        print(dataFromServer.decode());
```

Annexe 7 : Code Python pour l'envoi des données depuis le serveur.



Annexe 8 : Branchement pour le test de contact entre une entrée et une sortie après la soudure d'un relais.