RAPPORT DE PROJET
Supervision d'un atelier

_____

HAVARD Nicolas
IMA

POLYTECH LILLE

Boulevard Paul Langevin – Cité Scientifique
59655 VILLENEUVE D'ASCQ CEDX

Tel : +33 (0)3 28 76 73 60
Fax : +33 (0)3 28 76 73 61

Tuteur école : M. CHEVALIER Florian

REMERCIEMENTS

Je tenais à remercier mon tuteur, M. CHEVALIER Florian, pour m'avoir encadré et conseillé pendant ce projet. Merci à M. FLAMEN Thierry pour son aide et ses renseignements lors de l'élaboration de la carte électronique. Et enfin, merci à M. DELBROUCQ Hugo pour m'avoir permis de souder ma carte avec son matériel.

Table des matières

INTRODUCTION

Lors des travaux pratiques d'électrotechnique, les élèves ingénieurs de Polytech suivant la formation Informatique, Microélectronique et Automatique ont pu observer un problème technique se propageant dans la salle : l'extinction des afficheurs permettant la lecture de la tension qu'ils règlent pour alimenter divers appareils tels que des machines industrielles.

Cette extinction est due à l'utilisation d'un circuit non isolé du circuit de puissance et donc sensible aux appels de courant, détruisant par conséquent le circuit.

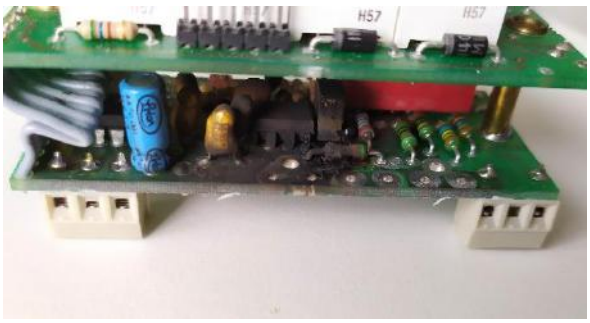


Photo d'un des anciens afficheurs ayant été détruit

Afin de résoudre ce problème, M. CHEVALIER a donc demandé à un de ces étudiants de fournir une solution durable dans le cadre de son PFE, afin d'assurer la possibilité de ses successeurs de participer à ces travaux pratiques. De plus, ce projet permettra de connecter ces afficheurs à l'ordinateur du superviseur de ces travaux pratiques afin de surveiller les tensions en sortie des générateurs depuis son bureau. La disposition du matériel est donc la suivante :

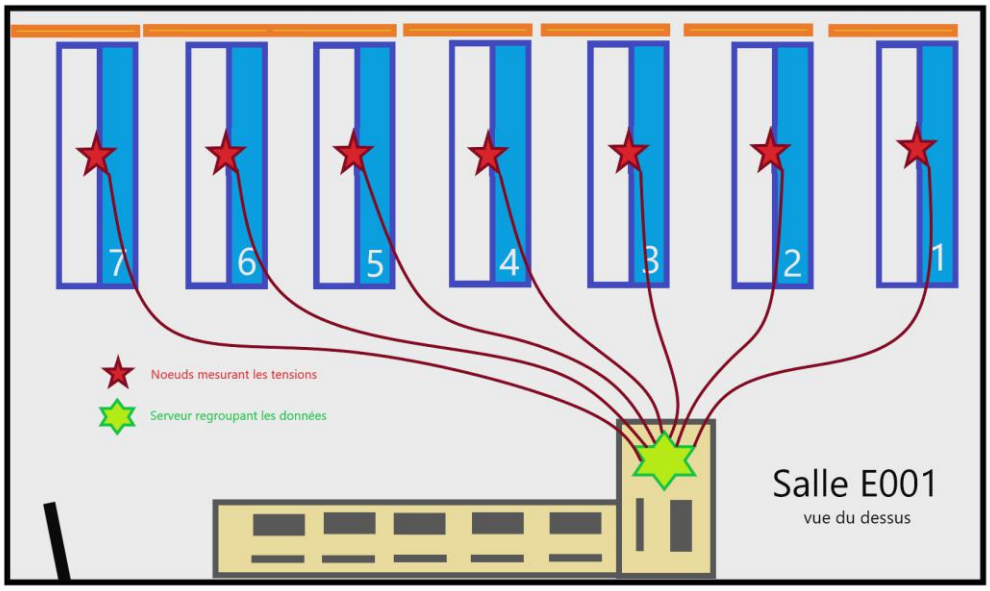


Schéma de la salle E001 et positionnement des cartes électroniques à concevoir, reliées au serveur par l'intermédiaire d'un switch

Les objectifs de ce projet s'articulent autour des trois points suivants :

- L'objectif principal de ce projet est de remplacer les actuels afficheurs des bancs de tests de la salle E001 par une solution isolée du circuit de puissance afin d'éviter la destruction du matériel.

- Le projet devra ensuite répondre au besoin d'une surveillance centralisée sur l'écran de l'encadrant permettant la visualisation des tensions aux bornes des sources de chaque banc de tests.

- Le dernier objectif est la rédaction d'un protocole de sécurité permettant d'informer les futurs opérateurs de la bonne utilisation de ce nouveau matériel.

SPECIFICATIONS

En début de projet, un certain nombre de points ont été abordé : ceux-ci précisent aussi bien les attentes du projet que les moyens mis à disposition pour sa réussite.

Le projet consistera à mettre au point un appareil présent dans chacun des bancs de travaux pratiques de la salle E001 afin de mesurer la tension présente aux bornes des trois sources de tensions variables. Ces sources étant constituées d'un générateur de tension variable triphasée pouvant atteindre 400 VAC efficace lié à une source de tension continue allant jusqu'à 250 VDC, un interrupteur permettant de passer de l'un à l'autre, ainsi qu'un dernier générateur de tension continue atteignant 250 VDC, indépendant des deux autres sources.

Contrairement au système utilisé jusqu'alors, le projet devra permettre la mise en place d'une solution assurant l'isolation galvanique entre le circuit de puissance et le circuit de mesures afin d'assurer la pérennité de celle-ci. La précision des mesures, elles, devront être de ±2 V environ.

La solution devra permettre l'affichage des tensions sur le banc de travaux pratiques via des afficheurs 7 segments placés aux emplacements dédiés, qui sont ceux utilisés aujourd'hui. Ces emplacements sont au nombre de trois : celui à gauche du tableau de bord permet d'afficher la tension du générateur de tension continue isolé des deux autres, qui se trouvent à droite du banc.

La carte de mesures sera alimentée par une alimentation 12 VDC, présente sur chacune des paillasses. Elle aura aussi à sa disposition un câble Ethernet relié au PC central de la salle E001 afin d'y envoyer les données mesurées pour les afficher sur un écran à disposition de l'encadrant du TP.

Enfin, la solution ne devra nécessiter aucune maintenance quelconque, qu'elle soit logicielle ou matérielle. Il est ainsi demandé de ne pas utiliser de batteries pour alimenter le module.

REALISATIONS

Afin d'expliquer la réalisation de ce projet, nous allons découper ce compte rendu en trois parties qui sont la création des cartes électroniques, la programmation des nœuds de capteurs et l'installation d'un serveur afin d'afficher les données centralisées.

Chaque partie détaillera les choix techniques réalisés en mettant en lumière les parties du cahier des charges motivant ces choix.

CREATION DES CARTES ELECTRONIQUES

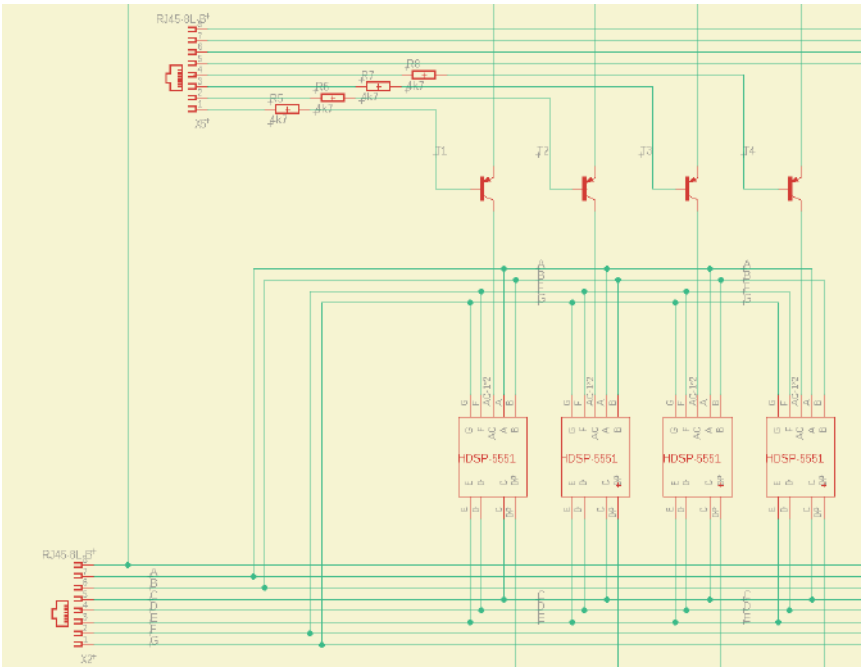
Ce projet ayant une partie conception électronique importante, il a rapidement fallu déterminer les composants qui allaient être utilisés dans le cadre de ce projet, mais aussi comment ces cartes allaient intégrer les bancs de travaux pratiques.

Afin de remplacer les afficheurs déjà présents sur les bancs, il a été nécessaire de créer trois cartes "*afficheurs*" qui se logeront aux emplacements des cartes actuelles. Disposant d'une intelligence sur chaque banc de TP, une carte "*principale*" récupérant les tensions et traitant ces valeurs a été décidée. Elle communiquera donc avec les cartes "*afficheurs*" et le réseau Ethernet de la salle. Elle sera donc placée à distance des cartes *"afficheurs*", au centre du banc de TP.

Cartes "afficheurs"

Comme stipulé dans le cahier des charges, ces cartes doivent remplacer les afficheurs actuellement présents sur les bancs de TP et doivent donc montrer des caractéristiques similaires. Ainsi, il a été choisi d'utiliser 4 afficheurs 7 segments de couleur rouge par carte. Celles-ci dépendant des informations transmises par la carte "*principale*", des câbles Ethernet permettront la transmission des informations avec cette dernière. Des connecteurs RJ45 ont été choisi afin de brancher plus facilement les cartes entre elles.

Chaque carte disposera donc de 4 afficheurs 7 segments à anode commune pilotés par des transistors PNP. La commande de ces transistors proviendra du driver de LEDs MM5451 permettant un multiplexage des afficheurs et évitant donc d'utiliser deux fois plus de connecteurs RJ45 et donc deux fois plus de câbles.



Schema du multiplexage des afficheurs 7 segments :
En bas se trouvent les signaux permettant d'allumer les segments tandis qu'en haut se trouve la commande choisissant quel afficheur va être alimenté

Afin de limiter le nombre de connecteurs RJ45 sur la carte "*principale*", les deux cartes "*afficheurs*" à droite du banc de TP se partageront les informations. En effet, les câbles Ethernet disposent de 8 fils, et le choix de l'afficheur 7 segments à piloter ne demande que 4 fils par carte (4 digits par carte "*afficheurs*", un fil par digit). Nous profitons donc des 4 fils restant pour piloter les 4 digits de la carte suivante. Les informations sur les segments à allumer sont donc transportées par le même câble que pour la carte précédente, et il suffit simplement de prolonger ce câble à la dernière carte comme suit :

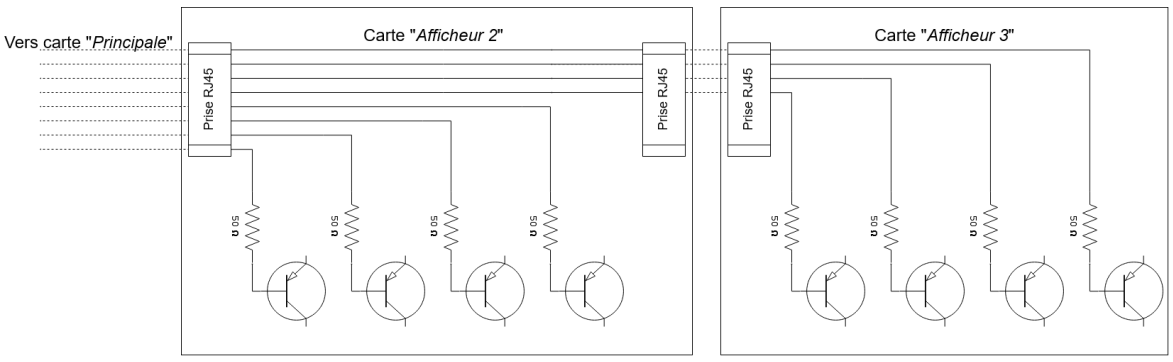


Schéma du partage du câble Ethernet entre l'afficheur central et celui de droite

Ces choix permettent donc d'économiser la place de deux connecteurs RJ45 sur la carte "*principale*" et du câble à conditions de modifier le code gérant ces afficheurs, le système se rapprochant alors d'un afficheur 4 digits à gauche du banc et d'un afficheur 8 digits à droite de celui-ci, tel que montré sur le schéma suivant :

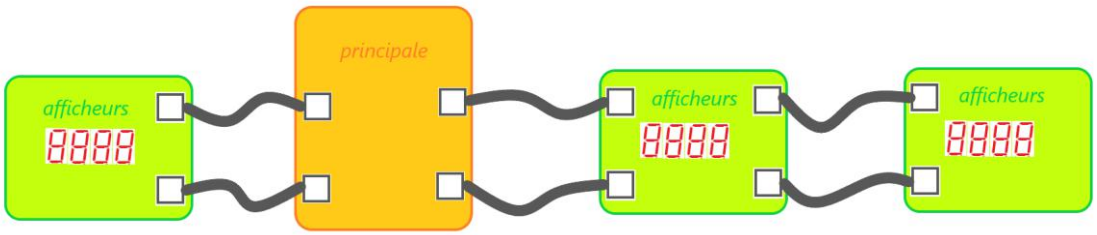


Schéma de la disposition des cartes sur le banc

Concernant les composants, chacune des trois cartes portera les composants suivants :

Description	Nombre	Référence	Prix / unité
Afficheur 7 segments à anode commune	4	HDSP-5551 de Broadcom	2€09
Transistor PNP	4	2DB1689-7 de DIODES INC.	0€223
Résistance 4k7	4	MCWR08X4701FTL de MULTICOMP Pro	0€0078
Prise RJ45	2	54601-908WPLF de AMPHENOL ICC	0€356
TOTAL			**9€9952**

Tableau des composants pour la carte "afficheurs" n°1

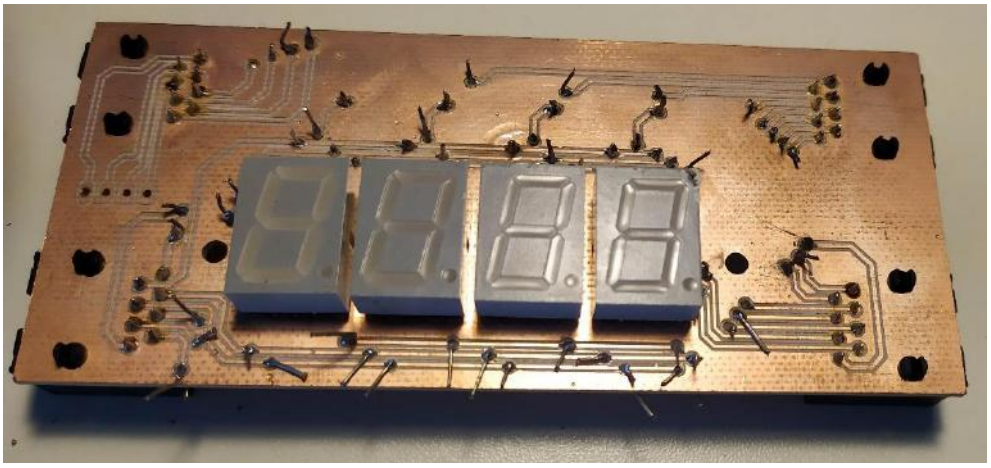
Description	Nombre	Référence	Prix / unité
Afficheur 7 segments à anode commune	4	HDSP-5551 de Broadcom	2€09
Transistor PNP	4	2DB1689-7 de DIODES INC.	0€223
Résistance 4k7	4	MCWR08X4701FTL de MULTICOMP Pro	0€0078
Prise RJ45	4	54601-908WPLF de AMPHENOL ICC	0€356
TOTAL			**10€7072**

Tableau des composants pour la carte "afficheurs" n°2

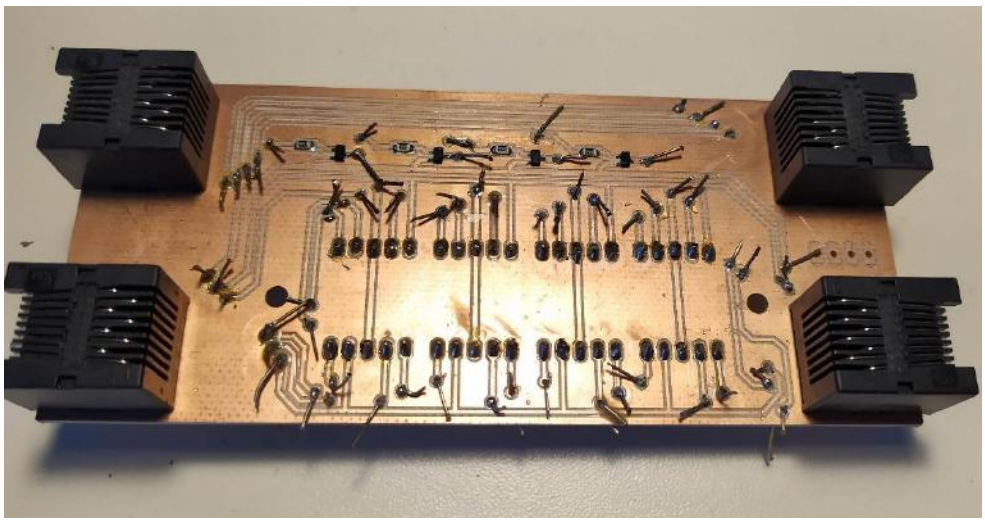
Description	Nombre	Référence	Prix / unité
Afficheur 7 segments à anode commune	4	HDSP-5551 de Broadcom	2€09
Transistor PNP	4	2DB1689-7 de DIODES INC.	0€223
Résistance 4k7	4	MCWR08X4701FTL de MULTICOMP Pro	0€0078
Prise RJ45	2	54601-908WPLF de AMPHENOL ICC	0€356
TOTAL			**9€9952**

Tableau des composants pour la carte "afficheurs" n°3

Une fois le schematique créé, nous avons pu imprimer les cartes et les souder avant de vérifier les éventuels courts-circuits. Un exemple du résultat est le suivant :



Bot du circuit « afficheur 2 »



Top du circuit « afficheur 2 »

Une fois la liste des composants décidée, nous avons pu passer à la création du schematic de chacune des cartes "*afficheurs*".

Puis nous sommes passés au routage de ces trois cartes et avons abouti au résultat suivant après une semaine de travail :

Après avoir expliqué la création des trois cartes "*afficheurs*", passons à la carte "*principale*".

Carte "principale"

Le but de cette carte est de récupérer les tensions aux bornes des générateurs, de traiter ces informations avant de les renvoyer sur le réseau Ethernet mais aussi de piloter les cartes "*afficheurs*" afin qu'elles affichent les tensions mesurées pour chaque source. Quatre blocs se distinguent donc sur cette carte :

- Récupération des tensions
- Pilotage des cartes "*afficheurs*"
- Envoi des données sur le réseau
- Traitement des informations faisant le lien entre les trois blocs précédemment cités

Récupération des tensions

L'objectif de ce projet étant de mesurer la tension présente aux bornes des différentes sources de tension des bancs de travaux pratiques, un des exigences principales a été de prévoir un circuit électronique permettant d'avoir une image fiable de ces valeurs.

Les tensions fournies par les sources pouvant atteindre 250 VDC ou 400 VAC, il est nécessaire de baisser cette tension à 5 VDC pour la lire à l'aide d'un microcontrôleur. Dans les cas où la tension à mesurer est continue, nous utilisons tout simplement un pont diviseur de tension pour obtenir une image de la tension en entrée comprise entre 0 et 5 VDC. Pour le générateur de tension alternative, nous utilisons un pont de diodes puis une capacité de filtrage afin de redresser le signal.

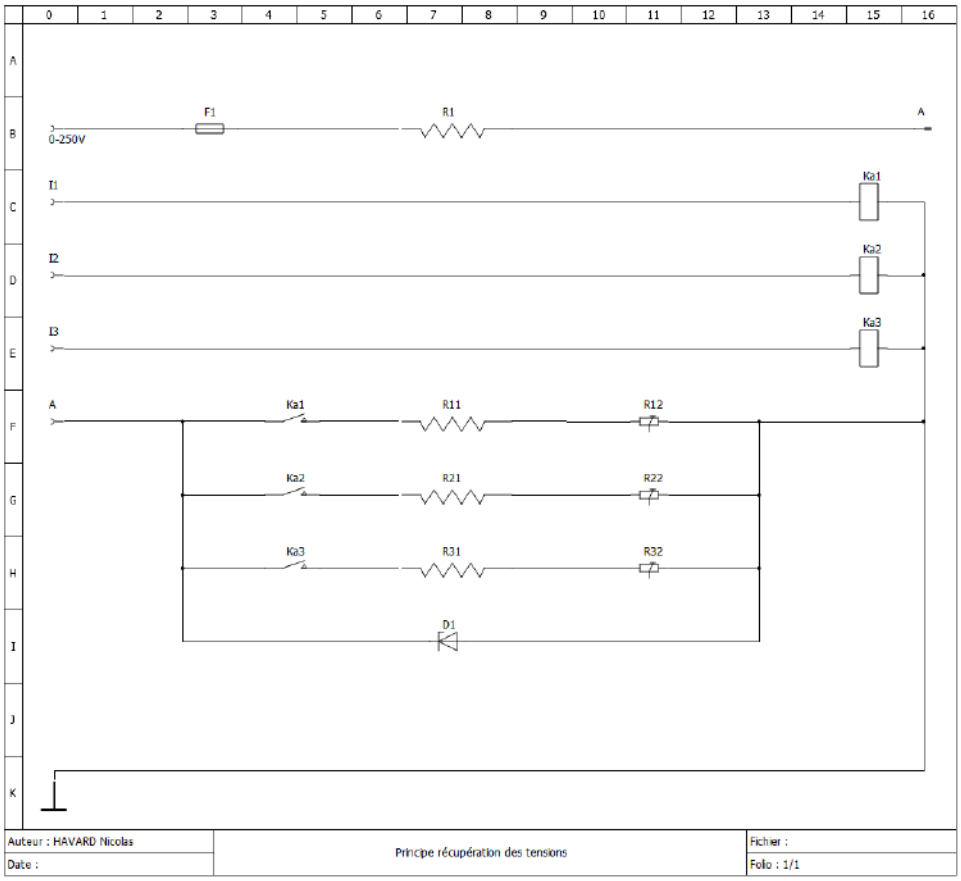
Afin d'assurer une meilleure précision lors de la mesure, il a été proposé d'utiliser plusieurs canaux de mesures en multipliant les ponts diviseurs de tension qui seront automatiquement activés par le microcontrôleur en fonction de la mesure actuelle de la tension. Afin de respecter le cahier des charges stipulant d'assurer l'isolation galvanique entre le circuit de puissance et le circuit de commande, nous avons choisi d'utiliser des relais pour piloter ces différents canaux de mesures étant donné que l'utilisation de transistors n'assure aucune isolation entre les deux circuits.

Le nombre de canaux par mesure a été déterminé à 3, soit une utilisation de 9 relais pilotés par le microcontrôleur pour garantir davantage de précision sur la mesure, et donc un même nombre de sorties numériques utilisées. Afin de limiter ce nombre, nous avons choisi d'utiliser des démultiplexeurs 2 vers 4 MC74ACT139 de On Semiconductor afin de gagner quelques sorties numériques sur le microcontrôleur. Ainsi, la gestion des relais pour les canaux de mesures n'a besoin que de 6 sorties au lieu des 9 qui auraient été nécessaires sans l'utilisation de ces démultiplexeurs.

Une fois que nous obtenons une image des mesures, il est nécessaire d'envoyer cette tension sur l'une des entrées analogiques de notre microcontrôleur. Afin d'assurer l'isolation galvanique entre le circuit de mesures et le microcontrôleur, nous utilisons un amplificateur d'isolation qui permet de transmettre un signal entre deux circuits tout en permettant l'isolation galvanique grâce à un optocoupleur. Dans notre cas, nous avons choisi d'utiliser l'ACPL-C87B

d'Avago qui demande de diminuer le signal en entrée entre 0 et 2 VDC. Ce module nécessite une tension d'alimentation comprise entre 4,5 et 5,5 VDC du côté du circuit de mesures et une tension d'alimentation entre 3 et 5,5 VDC de l'autre côté pour la sortie.

Si l'ACPL-C87B ne récupère le signal à transmettre que sur une seule entrée, il le transmet grâce à deux signaux dont la différence des potentiels est égale à la tension en entrée du module. L'ACPL-C87B nécessite donc l'utilisation d'un amplificateur opérationnel pour soustraire les deux signaux à sa sortie avant d'envoyer l'information sur une entrée du convertisseur analogique vers numérique du microcontrôleur, nous choisissons d'utiliser un OPA237 de Ti et profitons de ce montage différentiel pour choisir des résistances adaptées et amplifier la tension de 2 à 5 VDC maximum, soit un gain de 2,5 en sortie de l'amplificateur opérationnel.



Circuit du principe permettant de récupérer la tension

Sur le circuit ci-dessus, nous traitons le cas où la source de tension est une source de tension continue. Délivrant une tension pouvant atteindre les 250 VDC, celle-ci est abaissé à environ 2 V grâce aux différents ponts diviseurs de tension fermés avec les contacts des relais Ka1, Ka2 et Ka3 pilotés respectivement par les entrées I1, I2 et I3.

Concernant les ponts diviseurs de tension, il a donc fallu diminuer la tension maximale de chaque canal de mesures pour qu'elle ne dépasse pas 2 V. De plus, afin de diminuer la tension aux bornes de chaque résistance et ainsi limiter le claquage de ces dernières, nous multiplions les résistances de plus faibles valeurs que nous mettons en série : plutôt que de

n'avoir qu'une seule résistance R1 de 1,25 MΩ qui aurait une tension à ses bornes pouvant atteindre 250 VDC, nous utilisons 5 résistances de 249 kΩ qui auront alors une tension maximale de 50 VDC à leurs bornes.

TABLEAU DU CALCUL DES RESISTANCES

	Source de tension continue			Source de tension alternative		
Tension max (V)	250	150	50	500	150	50
Tension min (V)	150	50	0	150	50	10
ratio 2V/Input	0,008	0,01333	0,04	0,004	0,01333	0,04
R1 théorique (Ohm)			1 240 000			2 490 000
R1 choisie (Ohm)			1 245 000			2 490 000
R2 théorique (Ohm)	10 000	16824	51875	10000	33649	103750
R2 choisie (Ohm)	10 000	15 740	47 000	10000	33000	94000
Tension max réelle (V)	251	160	55	500	153	55

Les canaux de mesures, fixés par le client, ont servi de base à la détermination des valeurs des différentes résistances utilisées. En fixant R_2 à 10 kΩ afin d'obtenir une impédance minimale pour la plus faible résistance, nous obtenons alors une résistance théorique de 1 240 kΩ pour R_1 dans le cas continue et 2 490 kΩ pour le cas alternatif. Ces valeurs ont été déterminées en se basant sur la formule du pont diviseur de tension, en posant X la tension d'entrée et Y la tension aux bornes de R_2 :

$$\frac{X}{Y} = \frac{R_2}{R_1 + R_2}$$

En posant $Z = \frac{X}{Y}$, on aboutit alors à l'équation suivante :

$$Z = \frac{R_2}{R_1 + R_2}$$

$$Z \,.\, (R_1 + R_2) = R_2$$

$$Z \,.\, R_1 = R_2 \,.\, (1 - Z)$$

$$R_1 = R_2 \,.\, \frac{(1 - Z)}{Z}$$

Ayant posé R_2 = 10 kΩ, nous arrivons donc aux valeurs théoriques de 1 240 kΩ et 2 490 kΩ pour R_1, en fonction de si la nature de la source de tensions, comme nous l'avions calculé dans le tableau vu plus haut.

Une fois R1 déterminée, nous pouvons utiliser la formule écrite ci-dessus pour fixer R_2 de la même manière pour les différents canaux de mesures, ce qui nous donne les valeurs 17 kΩ et 52 kΩ pour la source continue et 34 kΩ et 104 kΩ pour la source alternative. Afin de prévoir une plage permettant le changement de ces canaux, nous réduisons donc la valeur des résistances pour avoir une tension maximale légèrement plus grande.

Les résistances sont finalement fixées dans un but d'uniformiser les résistances choisies afin de diminuer les coûts en prenant plusieurs dizaines de références : la résistance 15 740 Ω pour le canal 50-150 V de la source de tension continue, par exemple, est en fait constituée de deux résistances en série ayant une impédance de 7,87 kΩ chacune, ces résistances étant retrouvées ailleurs dans le projet.

Une fois les valeurs des résistances décidées, nous arrivons finalement à la liste de composants suivantes pour la partie récupération des tensions :

Description	Nombre	Référence	Prix (€) par unité
Amplificateur d'isolation	2	ACPL-C87B-000E d'Avago	6,35
Amplificateur opérationnel	1	OPA2374AIDCNR de TI	1,65
Démultiplexeur 2 vers 4	1	MC74ACT139DG de On Semiconductor	0,48
Relais reed	6	HE3621A0500 de Littlefuse	0,86
Prise RJ45	1	54601-908WPLF de Amphenol ICC	0,356
Diode Zener 2,2 V	2	UDZVTE-172.2B de ROHM	0,152
Diode Zener 36V	2	1N5938BRLG de On Semiconductor	0,328
Résistance 1 k	4	MCWR08X1001FTL de Multicomp Pro	0,0078
Résistance 2,55 k	4	ERJP06F2551V de Panasonic	0,0949
Résistance 7,87 k	20	MCWR08X7871FTL de Multicomp Pro	0,0041
Résistance 10 k	4	MCWR08X1002FTL de Multicomp Pro	0,0078
Résistance 47 k	2	MCWR08X4702FTL de Multicomp Pro	0,0078
Résistance 249 k	10	CRCW0805249KFKEA de Vishay	0,0834
Potentiomètre variable 2,5 k	6	3362P-1-252LF de BOURNS	0,289
Condensateur 100 n, 25 V	12	CL21B104KACNNNC de Samsung Electro-Mechanics	0,0511
Condensateur 100 n, 50 V	2	C0805C104M5RACTU de KEMET	0,0489
Bornier	2	PM5.08/2/90 de WEIDMULLER	0,505
TOTAL			**26,1346**

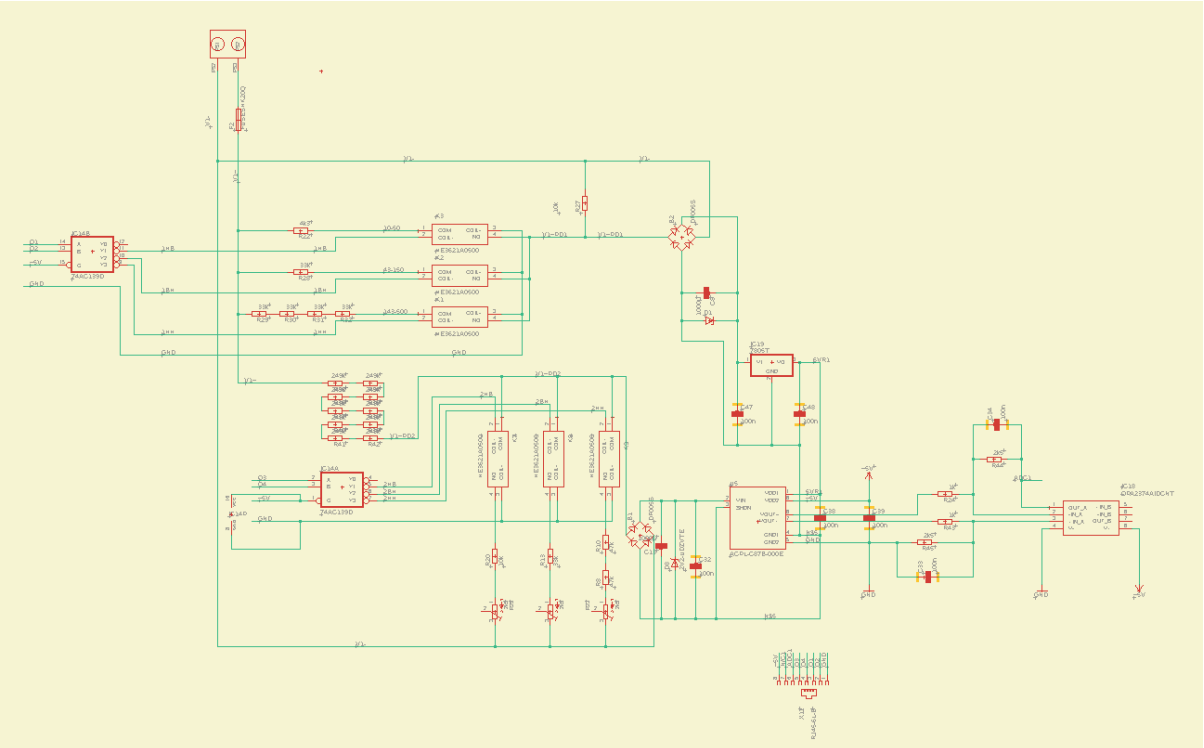
Tableau des composants pour le bloc 'récupération des valeurs' : Tensions continues DC1 et DC2

Description	Nombre	Référence	Prix (€) par unité
Amplificateur d'isolation	1	ACPL-C87B-000E d'Avago	6,35
Amplificateur opérationnel	1	OPA2374AIDCNR de TI	1,65
Démultiplexeur 2 vers 4	1	MC74ACT139DG de On Semiconductor	0,48
Relais reed	6	HE3621A0500 de Littlefuse	0,86
Prise RJ45	1	54601-908WPLF de Amphenol ICC	0,356
Diode Zener 2,2 V	1	UDZVTE-172.2B de ROHM	0,152
Diode Zener 36V	1	1N5938BRLG de On Semiconductor	0,328
Pont redresseur à diodes, 50 V	2	DF005S2 de On Semiconductor	0,39
Résistance 1 k	2	MCWR08X1001FTL de Multicomp Pro	0,0078
Résistance 2,55 k	2	ERJP06F2551V de Panasonic	0,0949
Résistance 4,3 k	1	MCWR08X4301FTL de Multicomp Pro	0,007
Résistance 10 k	2	MCWR08X1002FTL de Multicomp Pro	0,0078
Résistance 33 k	6	MCWR08X3302FTL de Multicomp Pro	0,0078

Résistance 47 k	2	MCWR08X4702FTL de Multicomp Pro	0,0078
Résistance 249 k	10	CRCW0805249KFKEA de Vishay	0,0834
Potentiomètre variable 2,5 k	3	3362P-1-252LF de BOURNS	0,289
Condensateur 100 n, 25 V	6	CL21B104KACNNNC de Samsung Electro-Mechanics	0,0511
Condensateur 100 n, 50 V	1	C0805C104M5RACTU de KEMET	0,0489
Condensateur 1 m, 50 V	2	MCGPR50V108M16X26 de Multicomp Pro	0,705
Bornier	1	PM5.08/2/90 de WEIDMULLER	0,505
TOTAL			**19,5629**

Tableau des composants pour le bloc 'récupération des valeurs' : Tension alternative AC

Une fois les composants listés, nous créons donc le schematics sur lequel nous allons nous appuyer pour réaliser le PCB. Celui affiché ci-dessous est celui associé à la récupération des tensions alternatives, il diffère de ceux mesurant la tension continue étant donné qu'il est nécessaire de redresser la tension en entrée de l'ACPL-C87B et du 78L05. De plus, il dispose de 3 ponts diviseurs de tensions avec leurs relais associés avant l'alimentation du régulateur de tension 5V :



Schematic du bloc de mesures des tensions pour la source alternative

Pilotage des cartes "*afficheurs*"

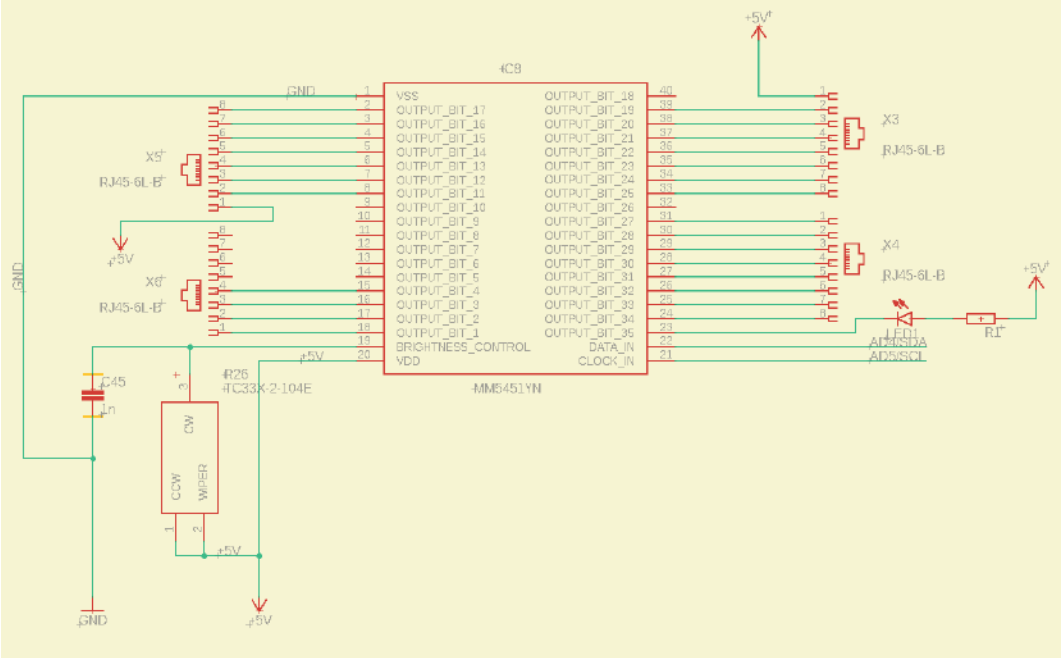
Afin d'afficher la tension mesurée sur le banc, nous utilisons un total de 12 afficheurs 7 segments. Les commander tous grâce aux sorties du microcontrôleur auraient demandé de dédier un trop gros nombre de ces sorties à cette fonction, même en utilisant un multiplexage des digits. C'est pourquoi nous avons recouru à l'utilisation d'un driver de LEDs pour piloter de nombreuses sorties avec un nombre d'entrées limitées.

Nous avions besoin de 7 sorties pour piloter les segments des 4 afficheurs à gauche du banc et 7 autres pour les 8 afficheurs à droite, soit déjà 14 sorties dédiées à la sélection des segments à allumer. Utilisant un multiplexage des 4 digits à gauche et des 8 digits à droite, 12 autres sorties ont été nécessaires à la sélection de ces digits. Le nombre total de sorties à piloter par le module est donc de 26.

Ne nécessitant que deux entrées numériques, notre choix s'est porté sur le module MM5451 de Microchip et ses 35 sorties. Ce module, alimenté par une tension comprise entre 4,75 et 11 VDC, dispose donc de 35 sorties fonctionnant comme des puits de courant. Ainsi, ces sorties se connectent aux LEDs sur leur cathode comme une masse dont le potentiel varie en fonction de la consigne : ces LEDs nécessitent donc d'être alimentées par une source de tension sur leur anode, et non l'inverse.

Une entrée du module permet de faire varier la luminosité de l'affichage et peut être utilisée pour automatiser le réglage de cette luminosité en fonction de celle de la pièce. Dans notre cas, nous nous contenterons d'utiliser un potentiomètre de très forte impédance pour régler les afficheurs une fois qu'ils seront en place étant donné que l'éclairage de la salle E001 est relativement constant.

Pour connecter les 26 sorties utilisées du MM5451 aux trois cartes "*afficheurs*", nous utilisons là encore des prises RJ45 comme nous l'avons fait de l'autre côté : 4 prises RJ45 sont donc utilisées afin d'envoyer les signaux par le câble Ethernet, deux prises pour chaque côté du banc.



Schematic du bloc gérant les afficheurs 7 segments, basé sur le MM5451

Le matériel nécessaire pour envoyer les informations aux cartes "afficheurs" est donc regroupé dans le tableau suivant :

Description	Nombre	Référence	Prix / unité
Driver de LEDs 35 sorties	1	MM5451 de Microchip	4€72
Prise RJ45	4	54601-908WPLF de AMPHENOL ICC	0€356
Potentiomètre 100 kΩ	1	TC33X-2-104E de Bourns	0€23
Capacité 1 nF	1	MC0805B102K500CT de Multicomp Pro	0€064
TOTAL			**6€438**

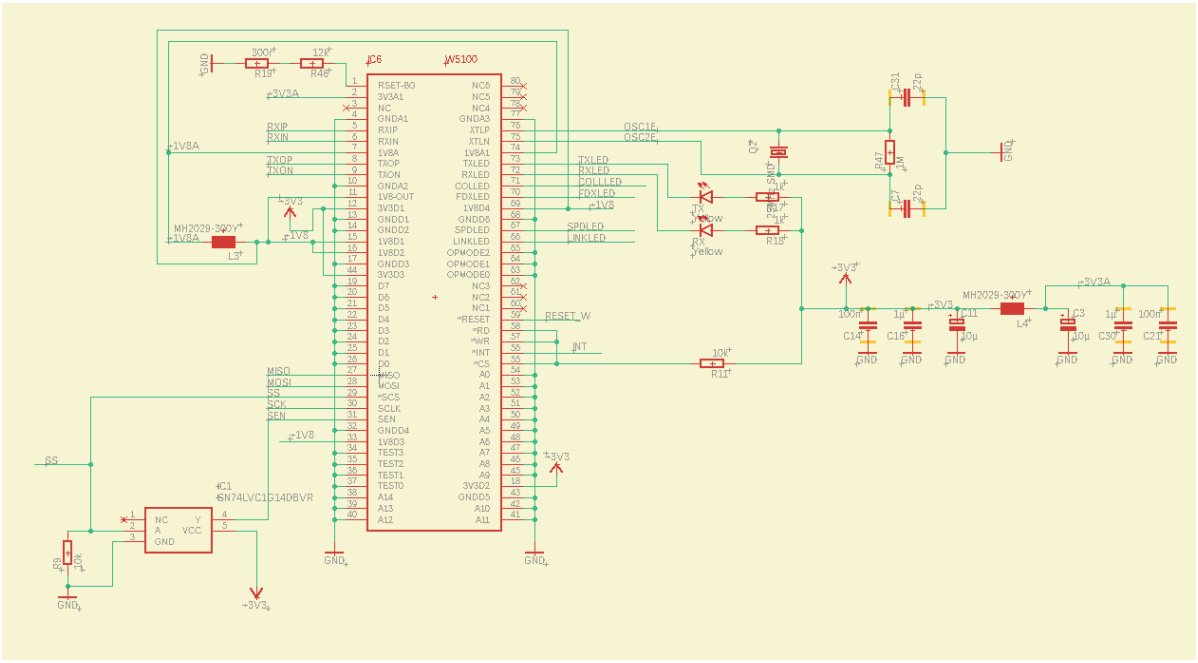
Tableau des composants pour le bloc 'Pilotage des cartes afficheurs'

Envoi des données sur le réseau

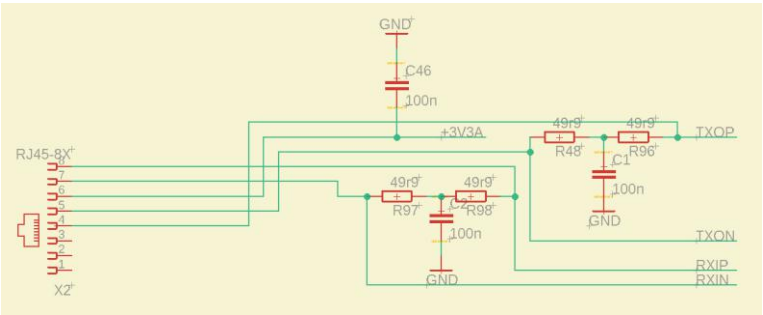
Enfin, le dernier bloc gère la transmission des données vers le réseau Ethernet de la salle afin de les stocker sur une base de données. Ce bloc sera principalement composé d'une prise RJ45 afin d'y connecter un câble Ethernet mais aussi un module permettant la communication avec les réseaux grâce au protocole TCP/IP. Ayant déjà utilisé des shields Ethernet conçus pour permettre à des cartes Arduino d'envoyer des données sur le réseau, j'ai souhaité me baser sur leur architecture et ai donc choisi les puces W5100 de Wiznet pour les intégrer à la carte "*principale*".

Ce choix de puce me permettait par la même occasion de pouvoir utiliser mon shield Ethernet pour réaliser un prototype et vérifier que la partie logicielle du projet fonctionne, éliminant donc les problèmes d'infrastructures et me permettant de gagner du temps sur le développement du programme.

Les puces W5X00 de Wiznet communiquent avec le microcontrôleur grâce au protocole SPI nécessitant 4 pins.



Schematic du W5100 et des composants qui lui sont liés



Schematic de la prise RJ45 permettant l'envoi des données sur le réseau

Finalement, nous parvenons à la liste de matériel suivante pour la fonction d'envoi des données sur le réseau de la salle :

Description	Nombre	Référence	Prix / unité
Puce Ethernet	1	W5100 de WIZnet	5€03
Cristal 25 MHz	1	LFXTAL033342 de IQD	0€332
Connecteur RJ45	1	54601-908WPLF de Amphenol ICC	0€356
Comparateur	1	SN74LVC1G14DBVR de TI	0€23
Perle de ferrite	2	MH2029-300Y de BOURNS	0€0684
LED jaune	2	597-5406-507F de Dialight	0€324
Résistance 49R9	4	MCWR08X49R9FTL de Multicomp Pro	0€0078
Résistance 300R	1	MCWR08X3000FTL de Multicomp Pro	0€0078
Résistance 1k	2	MCWR08X1001FTL de Multicomp Pro	0€0078
Résistance 10k	1	MCWR08X1002FTL de Multicomp Pro	0€0078
Résistance 12k	1	WCR0805-12KFI de TT Electronics	0€0344
Résistance 1M	1	MCWR08X1004FTL de Multicomp Pro	0€0078
Capacité 22p	2	885012007012 de Wurth Elektronik	0€08
Capacité 100n	5	CL21B104KACNNNC de Samsung Electro-mechanics	0€0511
Capacité 1µ	2	CL21B105KOFNNNE de Samsung Electro-mechanics	0€0626
Capacité 10µ, polarisée	2	MCGPR25V106M5X11 de Multicomp Pro	0€0452
TOTAL			**7€3695**

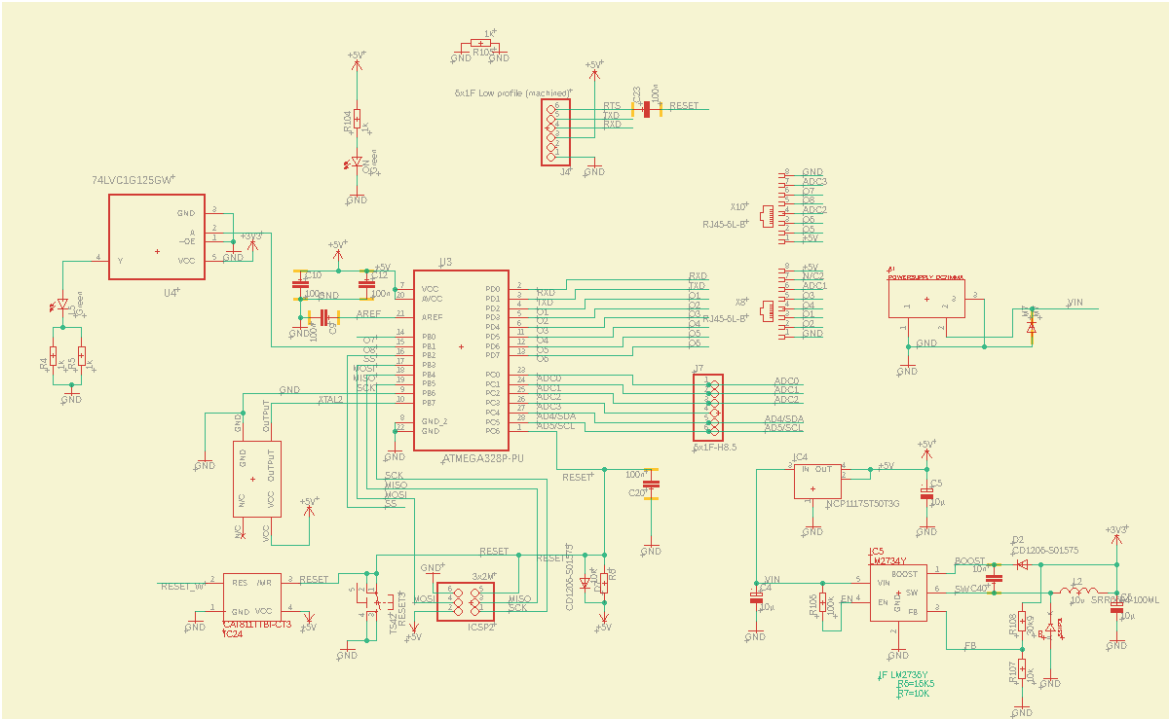
Tableau des composants pour le bloc 'Envoi des données sur le réseau'

Traitement des informations faisant le lien entre les trois blocs précédemment traités

Afin de regrouper les différents blocs et traiter les informations, il a enfin fallu choisir un microcontrôleur disposant donc de 3 entrées analogiques pour récupérer les mesures de tensions, de 6 sorties numériques pour la gestion des canaux de mesures, 2 sorties numériques pour la gestion de l'alimentation du module 78L05 dans le cas alternatif, 2 autres sorties numériques pour la gestion des afficheurs et enfin 4 autres sorties numériques pour la communication avec le module Ethernet, soit un total de 3 entrées analogiques et 14 sorties numériques.

Le microcontrôleur Atmega-328P de Microchip dispose de 14 I/O numériques et de 6 entrées analogiques et est donc parfaitement adapté à notre utilisation. Dans le but de conserver les pins liés à la communication série à des fins de déverminage, nous avons utilisé 2 des I/O analogiques comme sorties numériques afin de piloter le driver de LEDs MM5451. Ainsi, tous les pins de l'Arduino, hormis une entrée analogique, sont utilisés.

Ce microcontrôleur étant utilisé par Arduino comme base de leur carte UNO, il est bien connu dans l'univers de l'open source et de nombreux liens nous ont permis de créer notre carte autour de ce microcontrôleur afin de réguler les tensions 5V et 3V3 nécessaires au bon fonctionnement de notre carte par exemple.



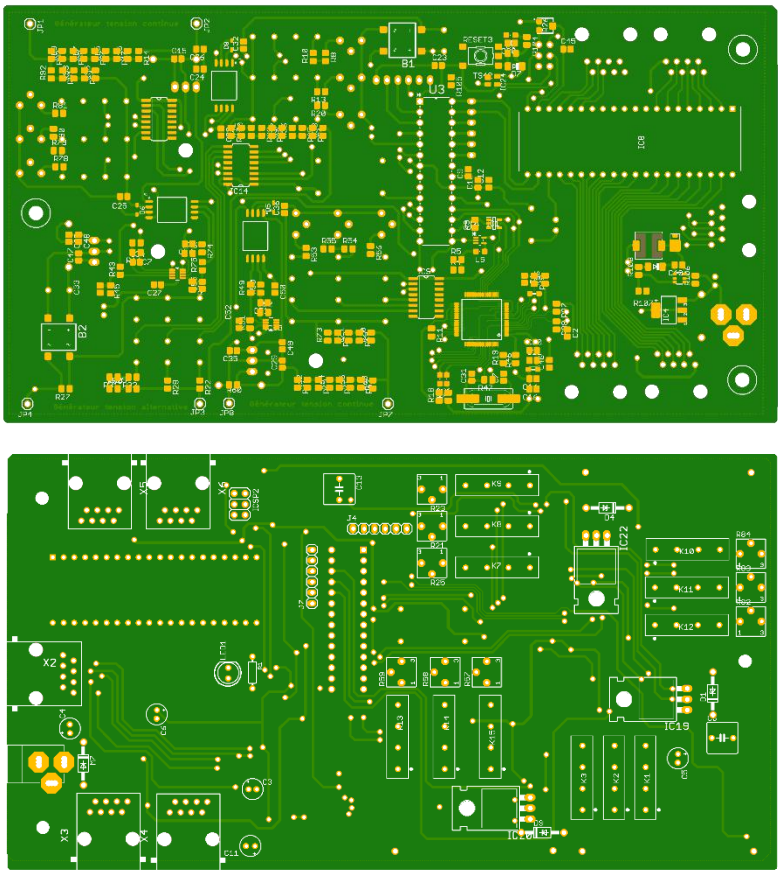
Schematic du bloc 'traitement des données', comprenant l'atmega 328P et des composants qui lui sont liés

La liste des composants utilisés dans ce bloc est reprise dans le tableau suivant :

Description	Nombre	Référence	Prix (€) par unité
Microcontrôleur	1	ATMEGA328P-PU de Microchip	1,7
Oscillateur 16 MHz	1	NCT050C	
Prise jack DC 12V	1	MJ-179PH de Multicomp	0,696
Diode de récupération standard	1	1N4007G de ON Semiconductor	0,197
Régulateur de tension 5V	1	NCP1117ST50T3G de ON Semiconductor	0,423
Régulateur de tension de commutation pour 3V3	1	LM2734YQMKE/NOPB de TI	2,93
Diode et redresseur Schottky	1	SS1P3L-M3 de Vishay Semiconductors	0,441
Superviseur d'alimentation MCU	1	CAT811TTBI-GT3 de ON Semiconductor	0,394
Commutateur tactile	1	1825910-7 de Alcoswitch – TE Connectivity	0,0682
Buffer / Driver de ligne	1	74LVC1G125GW de Nexperia	0,162
Prise RJ45	2	54601-908WPLF de Amphenol ICC	0,356
Diode signaux faibles	2	CD1206-S01575 de BOURNS	0,0665
LED verte	2	597-5326-507F de Dialight	0,365
Inductance SRR0604-100ML	1	SRR0604-100ML de Bourns	0,62
Résistance 1k	4	MCWR08X1001FTL de Multicomp Pro	0,0078
Résistance 10k	2	MCWR08X1002FTL de Multicomp Pro	0,0078
Résistance 30,9k	1	ERJP06F3092V de Panasonic	0,0833
Résistance 100k	1	MCWR08X1003FTL de Multicomp Pro	0,0077
Capacité 10n	1	885012207092 de Wurth Elektronik	0,08
Capacité 100n	5	CL21B104KACNNNC de Samsung Electro-mechanics	0,0511
Capacité 10µ, polarisée	3	MCGPR25V106M5X11 de Multicomp Pro	0,0452
TOTAL			**9,8151**

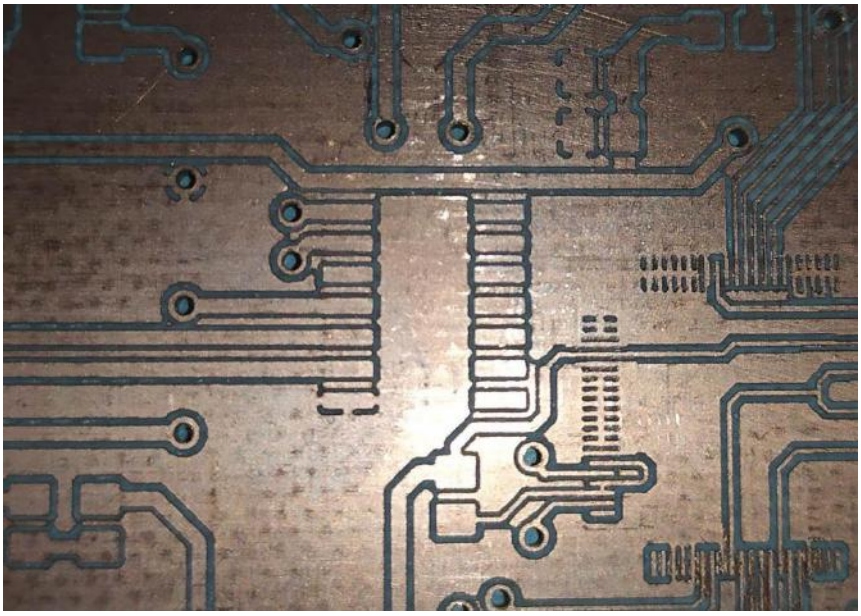
Création du PCB

Une fois la liste des composants décidée et le schematic les liant créé, il resta alors le routage de la carte. Le design de celle-ci fut d'abord pensé pour que les quatre blocs énoncés plus tôt soient liés au même PCB et éviter l'utilisation de câble entre ceux-ci. Après deux semaines de travail, le résultat fut le suivant :



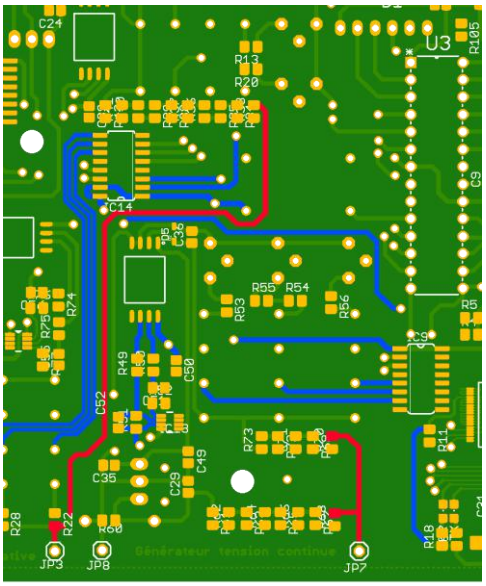
Top et bot de la première version de la carte "principale"

La première version de cette carte fut récupérée le 30 novembre, deux semaines après l'envoi à l'atelier. Cette carte souffre cependant de problèmes tels que des îlots orphelins, se logeant parfois entre les pins de certains composants et pouvant donc compliquer la soudure de ces derniers.



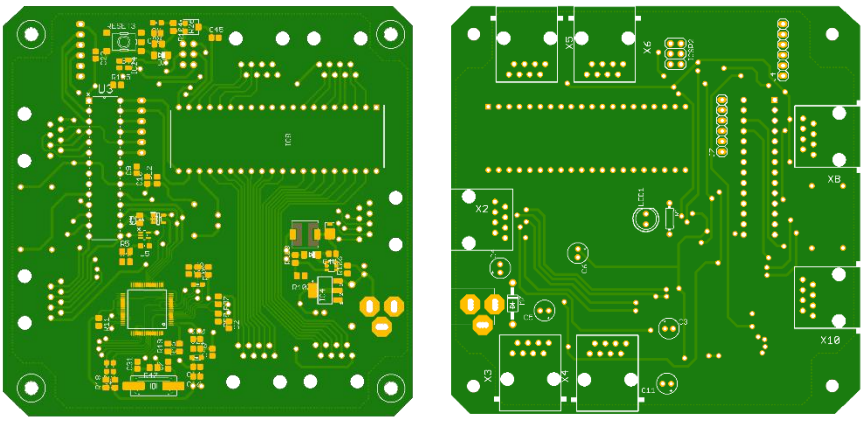
Sur cette photo se trouve un exemple des îlots se trouvant entre les pattes du composant à souder

Plus important, la phase de la source alternative traverse une grande partie du circuit et frôle les composants de la partie très basse tension à quelques millimètres près. La distance minimale entre deux pistes afin d'assurer l'isolation galvanique étant de 6 mm, la sécurité de cette carte est donc compromise et il a été nécessaire de travailler sur une seconde version de cette carte en utilisant davantage d'espace pour assurer l'isolation galvanique demandée par le client.

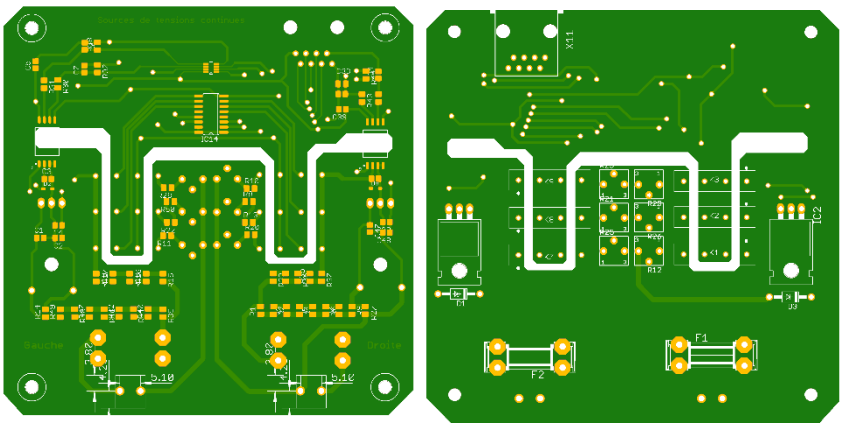


Sur ce morceau de carte, en rouge, nous pouvons apercevoir la phase de la source alternative en bas à gauche et la borne positive de la source continue en bas à droite. En bleu se trouve le circuit lié directement au microcontrôleur

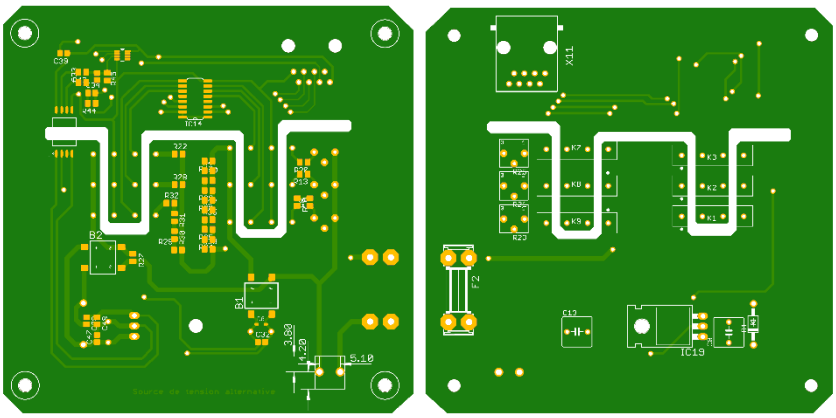
Cependant, la carte conçue mi-novembre dispose déjà une surface d'environ 10x20 cm², surface maximale pouvant être utilisée pour l'intégration de celle-ci dans un des bancs de la salle E001, et la répartition des composants sur cette surface ne peut pas beaucoup plus être optimisée. Il a donc été décidé de découper cette carte en trois sous cartes jointes entre elles par des câbles Ethernet comme nous le faisons pour les cartes "*afficheurs*".



Top et bot de la carte sur laquelle se trouve le microcontrôleur, le W5100 et le driver de LEDs



Top et bot de la carte permettant la mesure des tensions continues



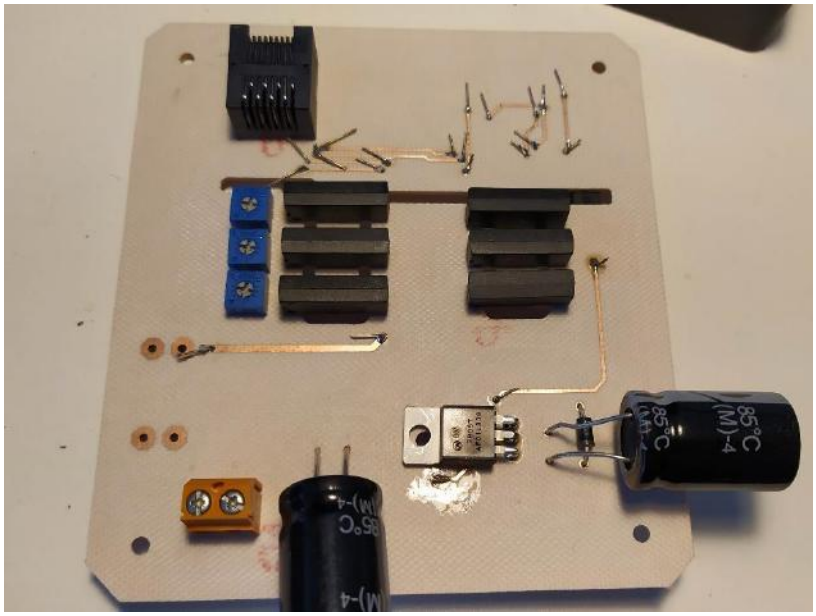
Top et bot de la carte permettant la mesure de la tension alternative

Ces trois cartes ont été dessinées à partir du même layout et sont donc toutes trois aux mêmes dimensions afin de les empiler facilement si besoin : de cette manière, nous comblons le manque de surface en exploitant la hauteur ou en séparant les cartes lors de l'intégration, et permettent à la solution de gagner en flexibilité.

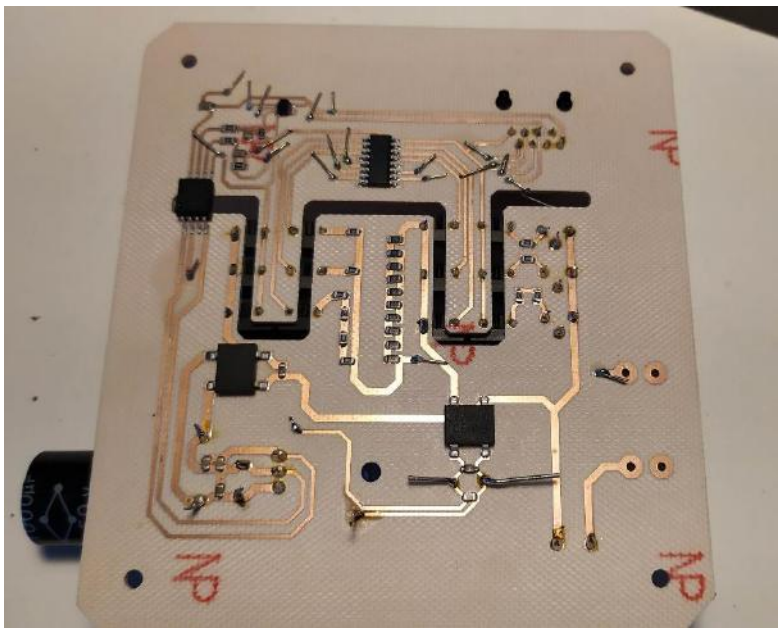
Envoyées le 4 décembre au service d'impression de Polytech en urgence afin de rattraper le retard apporté avec cette seconde impression, ces cartes n'ont pu être imprimées que le 16 décembre suite à un problème technique sur la machine permettant l'impression des cartes électroniques.

Bien que M. FLAMEN ait utilisé son outil le plus fin lors de l'élaboration de la carte, cela n'a pas été suffisant pour graver les pistes autour du W5100 qui sont trop fines pour être faites en interne à Polytech. L'utilisation de ce composant nécessite donc de sous-traiter la carte par une entreprise extérieure.

J'ai toutefois pu récupérer la carte permettant la mesure des tensions alternative et j'ai donc soudé les composants sur cette carte avant de la vérifier pour éviter les courts-circuits. Cependant, en ayant eu la carte la veille du rendu du projet, je n'ai pu la tester sur le banc de TP comme espéré.



Face top de la carte permettant le traitement de la tension alternative



Face bot de la carte permettant le traitement de la tension alternative

Mais avant de récupérer ces cartes, une fois qu'elles avaient été conçues et envoyées à l'impression, il restait alors la partie logicielle à concevoir afin de mener à bien ce projet en écrivant le programme permettant au microcontrôleur de récupérer les données et de les envoyer comme attendu, mais aussi la préparation du serveur pour héberger ces données.

INSTALLATION DU SERVEUR ET PROGRAMMATION DE L'APPLICATION

Afin de préparer l'arrivée des données en provenance de la carte, il a d'abord fallu installer le serveur. Une machine nous a ainsi été attribuée pour mener à bien nos essais et montrer les résultats de notre projet.

Contrairement au serveur de la salle tournant sous Lubuntu, la machine à notre disposition a été configurée pour tourner sous Ubuntu 18.04 LTS. L'adresse MAC de cette machine a été fournie au service informatique de Polytech afin de pouvoir l'utiliser sur le réseau de l'école, étape nécessaire pour l'installation des différents paquets dont nous avions besoin pour la configuration de notre serveur LAMP (Linux, Apache, MySQL, PHP). Le choix de cette solution a été fait sur la robustesse de ce système très utilisé de par le monde, et de sa simplicité à être mis en place sur un réseau local.

Une fois la connexion au réseau autorisée, nous avons pu installer sur la machine le serveur Apache2, MySQL 8.1 et PHP 7.2 ainsi que les différents paquets nécessaires à leur utilisation.

Il a alors fallu créer notre base de données pour héberger nos valeurs. Cette base a été appelée *E001* et ne contient qu'une seule table nommée *Tensions*, créée avec la configuration suivante :

Nom	Type	Description
ID	BIG INT UNSIGNED	*NOT NULL AUTO_INCREMENT PRIMARY KEY*
Banc	INT	*DEFAULT 0*
DC1	INT	*DEFAULT 0*
AC	INT	*DEFAULT 0*
DC2	INT	*DEFAULT 0*
Date	DATETIME	*DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP*

Tableau répertoriant les champs composants la table Tensions de la base E001

Le format *BIGINT UNSIGNED* utilisé pour le champ **ID** permet d'entrer 2^64-1 valeurs, soit de remonter plus de 10^19 informations en provenance des bancs de TP. Avec un rythme de 4 envois par seconde par module, nous atteignons donc plus de 10^14 heures de remontées d'informations avant de saturer la base de données.

Pour le champ **Date**, le type *Datetime* utilisé permet de stocker des dates et heures au format suivant 'YYYY-MM-DD hh:mm:ss' et a la particularité de pouvoir stocker des données du 1er janvier 1000 au 31 décembre 9999, échelle bien plus que suffisante pour notre application. Concernant la valeur par défaut, cette précision permet de mettre la date et l'heure actuelle lors de la sauvegarde de la donnée dans la table : le banc de TP n'envoyant les données qu'au moment où les mesures sont faites, nous ne risquons donc pas de fausser les valeurs en enregistrant à la date actuelle des données plus vieilles.

Afin d'utiliser la base de données, nous avons créé un nouvel utilisateur nommé '*4dm1n*' qui dispose de tous les privilèges et est accessible avec le mot de passe '*p4ssw0rd*'.

```
CREATE USER '4dm1n'@'localhost' IDENTIFIED BY 'p4ssw0rd';
GRANT ALL PRIVILEGES ON * . * TO '4dm1n'@'localhost';
FLUSH PRIVILEGES;
```

Cet utilisateur nous permet ensuite de nous connecter à la base de données dans nos scripts PHP en faisant appel à une fonction PHP contenant les lignes suivantes :

```php
<?php

    function Connection()
    {
        $server="localhost";
        $user="4dm1n";
        $pass="p4ssw0rd";
        $db="E001";

        $connection = mysqli_connect($server, $user, $pass, $db);

        if (mysqli_connect_errno())
        {
            echo "Failed to connect to MySQL : " . mysqli_connect_error() ;
            exit() ;
        }

        //old mysql : mysql_select_db($db) or die( 'MySQL ERROR: '. mysql_error() );
        return $connection;
    }


?>
```

Fonction PHP à inclure dans chaque page pour accéder à la base de données

La fonction écrite ci-dessus permet de se connecter à la base de données *E001* avec l'identifiant *4dm1n* afin de faire d'envoyer des requêtes à cette dernière. Par exemple, la page d'accueil de l'application récupère d'abord l'accès à la base de données grâce à cette fonction dans la variable *$link* avant d'effectuer une requête SQL pour remplir le tableau, créé dynamiquement, avec les données des 7 bancs de la salle.

```php
<?php

    include("connect.php");
    $link=Connection();
?>

<html>
    <head>
      <title>Valeurs mesurées</Title>
      <meta charset="UTF-8">
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
      <link rel="stylesheet" media="screen" href="https://fontlibrary.org/face/segment7" type="text/css"/>
    </head>


    <body>
       <h1>Valeurs mesurées</h1>

       <table id="voltmetres" border="1" cellspacing="1" cellpadding="1">
            <tr>
                <th> Banc </th>
                <th> Source 1 </th>
                <th> Source 2 </th>
                <th> Source 3 </th>
                <th> Date </th>
            </tr>

            <?php
                for ($i = 1; $i <= 7; $i++)
                {
                    $request = "SELECT DC1, AC, DC2, `Date` FROM `Tensions` WHERE Banc=".$i." ORDER BY date DESC LIMIT 1 ;" ;
                    $result=mysqli_query($link, $request);


                    if($result!==FALSE)
                    {
                        while($row = mysqli_fetch_array($result))
                        {
                            printf("<tr><td>  %s </td>
                                <td><span class=\"seg\">  %s </span></td>
                                <td><span class=\"seg\">  %s </span></td>
                                <td><span class=\"seg\">  %s </span></td>
                                <td>  %s  </td>
                                <td><form action=\"tab.php\" method=\"post\"><input type=\"hidden\" name=\"banc\" value=\"%s\">
                                        <input id=\"courbe\" type=\"submit\" value=\"\"></form></tr>",
                                $i, $row["DC1"], $row["AC"], $row["DC2"], $row["Date"], $i);
                        }
                        mysqli_free_result($result);
                    }

                }
            ?>


       </table>

       <p>
            <form action="reset.php" method="post">
                <button type="submit" class="reset" onclick="reset_tab();">Réinitialiser l'affichage
                    <span class="tooltiptext">Insère des 0 dans la base pour effacer les données trop vieilles à l'écran</span>
                </button>
            </form>
       </p>

       <script>
            setInterval('load_data()', 250) ;
            function load_data()
            {
                $('#voltmetres').load('load_voltmetres.php') ;
            }
       </script>

    </body>
</html>
```

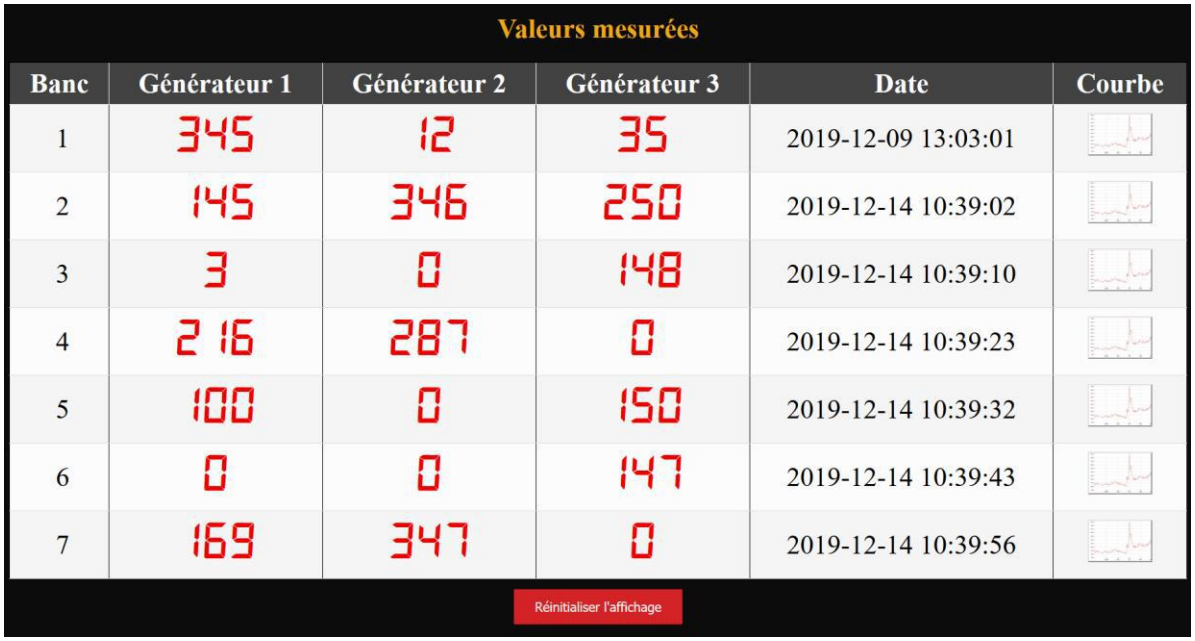
Code de la page d'accueil de l'application

Les lignes 57 à 63 concernent le bouton "*Réinitialiser l'affichage*" qui appelle le fichier "*reset.php*" après avoir cliqué dessus. Ce fichier contient un script PHP qui effectue une requête SQL à la base de données pour injecter des tensions nulles sur les 7 bancs : le but de cette manœuvre est de remettre l'affichage à 0 pour effacer de l'écran les valeurs rémanentes d'un précédent TP. Après avoir envoyé ces requêtes, le script rouvre la page d'accueil de l'application de manière à ce que cela soit transparent pour l'utilisateur.

Bouton permettant de réinitialiser l'affichage de la page d'accueil de l'application

Afin de réactualiser le tableau de données affiché sur la page en permanence et sans rafraichir la page entière, nous ajoutant une fonction Javascript visible des lignes 65 à 71. Cette fonction permet de charger le contenu du script PHP "*load_voltmetres.php*" dans le div '*voltmetres'* de manière périodique. Ici, nous choisissons de réactualiser le tableau toutes les 250 ms et donc de proposer à l'encadrant un aperçu assez proche de la réalité des valeurs visibles sur les bancs de TP.

Ainsi, la page d'accueil de l'application ressemble à celle-ci :



Page d'accueil de l'application affichant les tensions de chaque banc de TP

Afin de proposer un suivi des valeurs, nous avons ajouté la possibilité de visualiser l'historique des données. Il suffit pour cela de cliquer sur l'image présente dans la dernière colonne de la ligne dont on souhaite en apprendre davantage. Une nouvelle page s'ouvre alors et affiche un graphique montrant l'évolution des tensions aux bornes des trois sources dans la journée.

Exemple de graphique obtenu après avoir cliqué sur l'image d'une des lignes de la page d'accueil

Ce graphique est réalisé grâce à *Highcharts*, une API open-source programmée en Javascript permettant la création de graphiques dynamiques et fonctionnels qui est gratuite pour un usage non-commercial comme ce projet. La requête SQL sur laquelle se base cette courbe récupère toutes les valeurs des trois sources de tension enregistrées aujourd'hui et provenant du banc choisi sur la page d'accueil. Le graphique affiche alors l'évolution chronologique des valeurs.

Pour obtenir plus de précision sur la valeur d'un point, l'utilisateur peut passer le curseur sur ce dernier : une infobulle apparait alors au survol et précise la valeur mesurée. Il est possible d'isoler une courbe en masquant les autres en cliquant sur ces dernières dans la légende à droite du graphique. Pour les faire réapparaitre, il suffit alors de cliquer à nouveau sur celles-ci.

Il est aussi possible de zoomer sur le graphique en sélectionnant une plage de données : pour cela, il suffit de cliquer sur le graphique au point de départ de la sélection et de maintenir ce clic jusqu'à la valeur finale à observer. Cela permet de sélectionner une plage de temps et donc de restreindre l'échelle des abscisses. Pour revenir à la vue d'ensemble de la journée, il suffit de cliquer sur le bouton '*Reset zoom*' qui apparait après avoir zoomé sur le graphique.

En haut à droite du graphique se trouve un bouton hamburger qui permet, après avoir cliqué dessus, de voir le graphique en plein écran, de l'imprimer, de le télécharger en divers formats (PNG, JPEG, PDF ou SVG) ou encore d'en extraire les données dans un fichier CSV ou XLS.

Un bouton tout en bas permet de revenir à la page d'accueil de l'application et avoir une vue sur les valeurs des 7 bancs.

L'application comporte les différents fichiers suivants :

- *index.php* : page d'accueil de l'application, c'est elle qui permet de visualiser le tableau de valeurs des 7 bancs en temps réel. C'est depuis cette page que nous pouvons afficher les courbes en cliquant sur le graphique à la fin de la ligne.

- *load_voltmetres.php* : script PHP qui s'insère dans le tableau de la page *index.php* afin de le mettre à jour plusieurs fois par seconde.

- *reset.php* : script PHP qui est déclenché après un clic sur le bouton "*Réinitialiser l'affichage*" de *index.php*. Il permet de mettre des 0 pour toutes les valeurs des 7 bancs et efface donc les valeurs actuelles du tableau.

- *courbe.php* : page affichant l'évolution des données d'un des bancs de TP pour le jour en cours. S'ouvre après avoir cliqué sur l'image à la fin d'une des lignes de la page *index.php*.

- *add.php* : page sur laquelle sont envoyées les informations en provenance des bancs de TP afin d'être stockées dans la base de données du serveur. Cette page n'est pas destinée à être ouverte par l'utilisateur et n'affiche rien à l'écran. Si l'utilisateur ouvre cette page tout de même, il est redirigé vers la page *index.php*.

- *records.php* : page affichant toutes les données stockées dans la table par ordre chronologique décroissant. Ne sert qu'à des fins de tests.

- *form_post.html* : Page affichant un formulaire à remplir pour enregistrer des données manuellement dans la base de données. Ne sert qu'à des fins de tests.

- *action.php* : Script PHP permettant d'ajouter à la base de données les informations envoyées lors de la soumission du formulaire présent à la page *form_post.html*.

- *style.css* : Fichier CSS regroupant les caractéristiques esthétiques d'*index.php* et de *form_post.html.*

Une fois ces différents fichiers créés, nous avons pu tester l'application sur le serveur en envoyant des informations manuellement grâce à *form_post.html* et en visualisant ces données sur *index.php* et *courbe.php*. L'application fonctionnant avec des valeurs simulées, il resta donc à faire remonter les informations en provenance des bancs de TP de manière automatique en réalisant l'algorithme de la carte électronique présente sur le banc.

PROGRAMMATION DE LA CARTE "*PRINCIPALE*"

Afin d'écrire l'algorithme de la carte, nous avons décidé d'utiliser l'Arduino IDE et le langage de programmation des Arduino puisqu'une optimisation des ressources très poussée n'était pas nécessaire dans notre cas, et qu'il nous fallait gagner du temps sur le développement. Tout comme pour la conception de la carte électronique *"principale*", nous avons découpé le travail à réaliser en trois grandes parties qui sont :

- La récupération des données et leur traitement
- La manipulation des afficheurs 7 segments
- L'envoi des données au serveur

Afin de mettre au point l'algorithme des cartes présentes sur les bancs de TP, nous avons utilisé du matériel prêt à l'emploi pour gagner du temps en attendant les cartes électroniques. Ainsi, un Arduino Uno et son shield Ethernet nous ont permis de simuler le traitement des données et leur envoi sur le serveur de la salle. Une fois le driver de LEDs récupéré, nous avons pu le tester avec quelques LEDs sur une breadboard, puis les afficheurs 7 segments utilisés pour les cartes "*afficheurs*" afin de se rapprocher au plus près de l'aspect final du projet.

Pour ce qui est de la partie récupération des tensions, le prototype utilisé n'a pu se rapprocher de l'objectif final étant donné que la majorité des composants sont des CMS et nécessite donc la carte électronique créée plus tôt. Nous avons donc utilisé un potentiomètre afin de faire varier une tension de 0 à 5 VDC. Si le principe est le même que ce qui sera fait en fin de projet, cette approximation retarde la mise en place des coefficients finaux qui seront donc à régler lors de l'acquisition de la carte, sans quoi un décalage entre l'affichage et les valeurs réelles persistera.

Récupération des données

La récupération des tensions est, pour l'Atmega, une simple lecture d'une tension analogique à l'une de ses bornes. Cette lecture est possible par l'utilisation de la fonction analogRead() qui renvoie un entier compris entre 0 et 1023 proportionnellement à la valeur de la tension lue, à condition que celle-ci n'excède pas 5 VDC.

```
#define VDC1_pin A2                          // U1 -> tension continue à gauche du banc, lié à ADC2
int VDC1_value = 1025 ;
int VDC1_PdT = 0 ;                           // Pont diviseur de tension 1 -> plus grande échelle possible

const int VDC_R1 = 2490 ;                    // valeur de R2, en kohm
const int VDC_R2[] = {10, 33, 94} ;          // valeurs possibles de R1, en kohm
const int VDC_PdT_tens_hautes[] = {252, 161, 55} ;   // tensions (V) correspondant réciproquement aux valeurs à 1024 des différents ponts
const int VDC_PdT_lim_hautes[] = {1014, 955, 928} ;  // correspond réciproquement à des limites de 250V (1014 - 1.98/2), 150V (955 - 1.87/2) et 50V (928 - 1.82/2)
const int VDC_PdT_lim_basses[] = {569, 254, 185} ;   // correspond réciproquement à des limites de 140V (569 - 1.111/2), 40V (254 - 0.497/2) et 10V (185 - 0.362/2)

#define VDC1_mux_A 6                         // Voie A du démux 2 vers 4 lié à la gestion des ponts diviseurs de tension de DC1
#define VDC1_mux_B 7                         // Voie B ...
```

Données théoriques des seuils pour chaque canal de mesures

Nous créons donc des tableaux avec les limites théoriques sur 10 bits correspondant aux tensions de seuils des différentes échelles imposées par le choix des résistances qui vont nous permettre de donner la valeur de la tension après avoir lu une valeur grâce à la fonction map(), dont le principe est de donner une valeur multiplié par un facteur dépendant de l'échelle à laquelle nous souhaitons placer cette valeur. Par exemple, pour une valeur de 8 comprise entre 0 et 10, nous pouvons replacer ce 8 sur une échelle de 0 à 100, correspondant à un facteur 10 de la première échelle : la valeur sera alors 80.

En définissant nos bornes, nous pouvons donc traduire une valeur comprise entre 0 et 1024 pour déterminer la tension correspondant à celle-ci.

```
39  void setup()
40  {
41    /* Activer les ponts diviseurs de tension permettant la lecture des plus hautes tensions par securité
42         -> Pont diviseur de tension pour l'alimentation du 78L05 en cas alternatif : A et B à 5V
43         -> Pont diviseur de tension pour obtenir 2V en entrée de l'ACPL-C87B : A à 5V et B à 0V
44    */
45    Serial.begin(9600) ;
46    digitalWrite(PdT_mux_A, HIGH) ; digitalWrite(PdT_mux_B, HIGH) ;
47
48    VDC1_PdT = 0 ; digitalWrite(VDC1_mux_A, HIGH) ; digitalWrite(VDC1_mux_B, LOW) ;
49    VAC_PdT = 0 ; digitalWrite(VAC_mux_A, HIGH) ; digitalWrite(VAC_mux_B, LOW) ;
50    VDC2_PdT = 0 ; digitalWrite(VDC2_mux_A, HIGH) ; digitalWrite(VDC2_mux_B, LOW) ;
51  }
52
```

Initialisation des canaux de mesures

Afin de déterminer l'échelle de mesures à utiliser, nous plaçons donc notre système dans une configuration par défaut lors de son initialisation. Cette configuration est celle comportant l'échelle la plus grande et qui permet donc de lire l'intégralité des valeurs sans poser de problème au système : il s'agit de l'échelle 0-250 V pour les sources continues et de l'échelle 0-400 V pour les sources alternatives. Ces échelles sont obtenues en envoyant 5 VDC sur la voie A du démultiplexeur et en connectant la voie B à la masse.

Pour l'alimentation du régulateur de tension 5 VDC en entrée de l'ACPL-C87B, la logique est différente et il est nécessaire d'envoyer 5 VDC sur les deux entrées du démultiplexeur 2 vers 4 afin de sélectionner le canal correspondant à une entrée pouvant atteindre les 400 VAC.

```ino
void loop()
{
  // Récupération des valeurs :
  VDC1_value = analogRead(VDC1_pin) ;
  /*Serial.print("La valeur mesurée est : ") ;
  Serial.println(VDC1_value) ;
  Serial.print("Le changement d'échelle se fait entre ") ;
  Serial.print(VDC_PdT_lim_basses[VDC1_PdT]) ;
  Serial.print(" et ");
  Serial.println(VDC_PdT_lim_hautes[VDC1_PdT]);*/

  // Test des échelles :
  if ((VDC1_value < VDC_PdT_lim_basses[VDC1_PdT]) and (VDC1_PdT < 2))
  {
      VDC1_PdT = VDC1_PdT + 1 ;   // on passe au pont diviseur du dessus, permettant de réduire l'échelle et d'augmenter la précision
      set_PdT_DC1(VDC1_PdT) ;
      Serial.print("Echelle réduite ! Echelle niveau : ") ;
      Serial.println(VDC1_PdT) ;
  }
  else if ((VDC1_value > VDC_PdT_lim_hautes[VDC1_PdT]) and (VDC1_PdT > 0))
  {
      VDC1_PdT = VDC1_PdT - 1 ;   // on passe au pont diviseur du dessous, permettant de lire des valeurs plus hautes
      set_PdT_DC1(VDC1_PdT) ;
      Serial.print("Echelle augmentée ! Echelle niveau : ") ;
      Serial.println(VDC1_PdT) ;
  }

  // Traduction valeur :
  else
  {
    Serial.print("L'échelle n'a pas changé. Echelle niveau : ") ;
    Serial.println(VDC1_PdT);

    //float temp = VDC1_value/1024 ; ///VDC_R1*VDC_R2[VDC1_PdT] ;

    // On récupère la tension correspondant à la valeur sur 1024. La tension la plus haute correspond à la valeur limite la plus haute pour un canal donné
    int val_VDC1 = map(VDC1_value, 0, VDC_PdT_lim_hautes[VDC1_PdT], 0, VDC_PdT_tens_hautes[VDC1_PdT]) ;
    Serial.print("Valeur affichée : ") ;
    Serial.println(tension) ;

    // Envoi de la valeur sur l'afficheur 7 segment et sur le réseau
  }
  delay(2000) ;
}
```

Code principal lié à la lecture de la tension

Comme nous pouvons le voir sur ce code, il est donc nécessaire de vérifier sur quel canal nous nous trouvons à chaque mesure et s'il est nécessaire de changer de canal de mesures avant de traduire celle-ci en une tension à envoyer au réseau et à afficher sur le banc de TP. Afin de choisir facilement le pont diviseur de tensions à utiliser, nous écrivons une courte fonction pour chacune des trois sources telle que nous pouvons le voir ci-dessous pour la source de tension continue DC1 :

```ino
void set_PdT_DC1(int val_pont)
{
  if (val_pont == 0)                    // Calibre 0-252 V
  {
    digitalWrite(VDC1_mux_A, HIGH) ;
    digitalWrite(VDC1_mux_B, LOW) ;
  }
  else if (val_pont == 1)               // Calibre 0-161 V
  {
    digitalWrite(VDC1_mux_A, LOW) ;
    digitalWrite(VDC1_mux_B, HIGH) ;
  }
  else if (val_pont == 2)               // Calibre 0-55 V
  {
    digitalWrite(VDC1_mux_A, HIGH) ;
    digitalWrite(VDC1_mux_B, HIGH) ;
  }
}
```

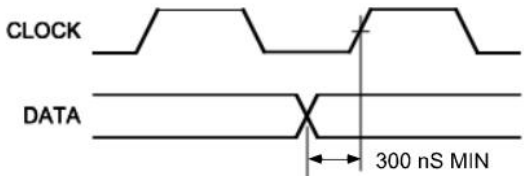
Fonction permettant de mettre à jour le pont diviseur de tension pour une des trois sources

Le principe de cette fonction est de vérifier le choix du pont diviseur de tension lorsque celui-ci est changé afin de mettre à 5 VDC ou à GND les deux signaux en entrée des

démultiplexeurs en sachant que, matériellement, (5, 0) correspond au canal de base qui est donc le plus grand, et que les couples (0, 5) et (5, 5) correspondent réciproquement au second et dernier canaux. Le couple (0, 0), lui, est utilisé comme sécurité afin de ne pas utiliser de pont diviseur de tension et donc de ne pas envoyer de valeurs non maitrisées à l'amplificateur d'isolation.

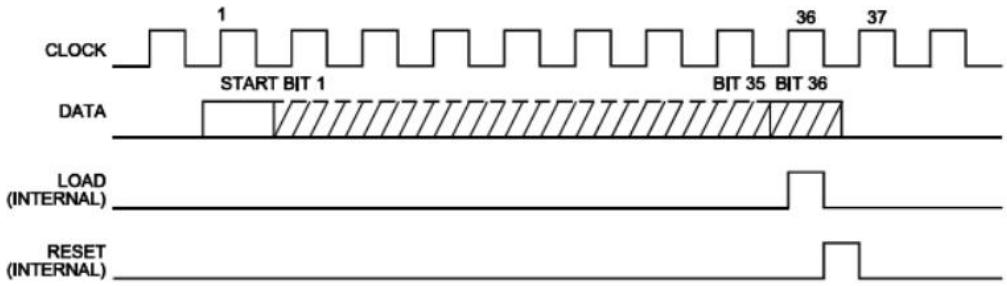
Affichage des données sur les cartes "*afficheurs*"

Pour afficher les valeurs des tensions mesurées sur les cartes "afficheurs", il suffit d'envoyer une trame adaptée au module MM5451. En effet, les deux entrées de cette puce sont un signal d'horloge '*clock*' et un signal de données '*data*' sur lequel est lue une trame de 36 bits : à chaque front montant de l'horloge, un buffer de 35 bits récupère l'état actuel du signal de données pour construire la trame avant de la traiter. Afin de garantir l'exactitude du message récupéré par le MM5451, 300 nanosecondes doivent séparer chaque front montant du signal *clock* d'un changement de valeur sur le signal *data*.



Condition temporelle à garantir lors de l'envoi de la trame

De plus, chaque trame doit être amorcée par un état haut lu sur le signal *data* afin de signaler au module l'arrivée d'une nouvelle trame : deux trames de données peuvent donc être séparées par une infinité de front montant de *clock*, à conditions que *data* soit à l'état bas. Ainsi, l'état des 35 sorties est représenté par l'état de *data* à partir du second front montant de *clock*.



Représentation d'une trame envoyée au module, caractérisée par 36 fronts montants de clock et d'un état haut de data en début de message

Dans notre cas, le signal clock n'est pas relié à une horloge mais est géré par le microcontrôleur qui permet de faire varier la fréquence d'envoi en fonction de notre utilisation. L'activation d'un segment, par exemple, peut donc se faire en plaçant la sortie du microcontrôleur liée à *data* à l'état haut au bon moment, puis en faisant suivre un état bas et un état haut sur la sortie liée à *clock*.

Afin d'allumer une LED sur deux avec ce module, il suffirait par exemple de placer la sortie liée à *data* sur l'état haut, d'enclencher un front montant de *clock*, puis de manipuler *data* en fonction de la parité du bit actuel en le faisant suivre d'un front montant de *clock*, comme nous le faisons dans cet extrait de code sous Arduino IDE :

```
digitalWrite(dataBit, HIGH) ;
pulseCLK() ;
for (int i = 0; i < 35; i++)
{
  if (i%2 == 0)
  {
    digitalWrite(dataBit, HIGH) ;
    pulseCLK() ;
    //Serial.print(i) ;
    //Serial.println(" : allumée !");
  }
  else
  {
    digitalWrite(dataBit, LOW) ;
    pulseCLK() ;
    //Serial.print(i) ;
    //Serial.println(" : éteinte !");

  }
}
```

Extrait de code permettant d'allumer une LED sur deux

Pour le pilotage des afficheurs 7 segments, nous créons tout d'abord une correspondance entre le chiffre à afficher et les différents segments à allumer en binaire :

```
// Tableau des codes binaires pour les afficheurs 7 segments
byte segCode[] =
{
  // GFEDCBA
  0b00111111,      // 0
  0b00000110,      // 1
  0b01011011,      // 2
  0b01001111,      // 3
  0b01100110,      // 4
  0b01101101,      // 5
  0b01111101,      // 6
  0b00000111,      // 7
  0b01111111,      // 8
  0b01100111       // 9
};
```

Tableau comprenant les codes binaires pour afficher les chiffres de 0 à 9 sur les afficheurs 7 segments

Une fois ce tableau de codes binaires écrits, il suffit de récupérer le chiffre à afficher, et d'utiliser le code binaire correspondant à ce chiffre. Dans notre cas, puisque les afficheurs sont connectés au MM5451 de manière à ce que la première sortie soit le segment A, la seconde le segment B, et ainsi de suite jusqu'à la septième sortie reliée au segment G, nous réalisons une comparaison sur la valeur du dernier bit avant de faire un décalage de bits par la droite : si la valeur de ce dernier bit est 1, alors *data* est mis à l'état haut et un front montant de *clock* est envoyé; sinon *data* est à l'état bas avant le front montant de *clock*. Cette fonction est appelée *ssrWriteLSB* :

```
1   void ssrWriteLSB(byte value)
2   {
3     for(int x =0; x < 8; x++)
4     {
5       byte temp = value & 0x01;
6       if (temp == 0x01)   digitalWrite(dataBit, 1); // data : état haut
7       else digitalWrite(dataBit, 0);                // data : état bas
8       pulseCLK();                                   // clock : front montant
9       value = value >> 0x01;                        // Décalage par la droite
10    }
11  }
```

Fonction ssrWriteLSB() permettant le décalage vers la droite du mot binaire et de l'envoi du bit de poids le plus faible vers le buffer du MM5451

Grâce à cette fonction, nous pouvons alors facilement injecter dans la trame les 7 bits indiquant les segments à allumer. Dans l'exemple ci-dessous, deux afficheurs 7-segments sont allumés par deux trames différentes : la première permet d'allumer les segments de l'afficheur de gauche, représentant les dizaines, tandis que la seconde allume ceux du dernier afficheur représentant les unités. L'intérêt de procéder de cette manière a été de simuler le pilotage des afficheurs à l'aide des transistors.

En effet, le multiplexage utilisé pour diminuer le nombre de fils nécessaires entre la carte "*principale*" et les cartes "*afficheurs*" demande en contrepartie de manipuler les afficheurs un par un. Cette technique demande donc d'utiliser 4 à 8 trames pour afficher l'information sur les afficheurs, diminuant donc la luminosité apparente.

```
int sensor_value = analogRead(sensor) ;                         // Récupération de la valeur à afficher (0-1024)

int u = sensor_value % 10 ;                                     // Unité de la valeur (exemple : pour 451, 1)
int d = (sensor_value % 100 - u)/10 ;                           // Dizaine de la valeur (exemple : pour 451, 5)
int c = (sensor_value % 1000 - d - u)/100 ;                     // Centaine de la valeur (exemple : pour 451, 4)
//int m = (sensor_value % 10000 - c - d - u)/1000 ;
//int dm = (sensor_value % 100000 - m - c - d - u)/10000 ;
//Serial.print(sensor_value) ; Serial.print(" : ") ; Serial.print(d) ; Serial.print(" | ") ; Serial.println(u) ;


/* Première trame pilotant le premier afficheur, celui des dizaines */
// start bit
digitalWrite(dataBit, 1);
pulseCLK();

ssrWriteLSB(segCode[d]);        // envoi des 8 bits indiquant à l'afficheur 7 segment la dizaine à afficher
zeroWrite(27);                  // On complète la trame avec des états bas (35-8=27)

//delay(1000);



/* Seconde trame pilotant le deuxième afficheur, celui des unités */
// start bit
digitalWrite(dataBit, 1);
pulseCLK();

zeroWrite(7);                   // on envoie 7 états bas afin de ne pas perturber l'afficheur des dizaines
ssrWriteLSB(segCode[u]);        // envoi des 8 bits indiquant à l'afficheur 7 segment l'unité à afficher
zeroWrite(20);                  // On complète la trame avec des états bas (35-7-8=20)

//delay(50);
```

Exemple de code permettant de récupérer la valeur d'un capteur et d'envoyer la dizaine et l'unité sur deux afficheurs 7 segments à l'aide de deux trames différentes

D'après ce test, la fréquence d'affichage des chiffres sur les afficheurs 7 segments est suffisante pour que l'œil humain puisse voir les chiffres alignés en même temps, sans décalage. De plus, la luminosité, bien que nettement plus faible avec le multiplexage, reste suffisante pour voir la valeur de loin avec une lumière ambiante importante. La manipulation des afficheurs 7 segments est donc satisfaisante et valide donc les critères attendus.

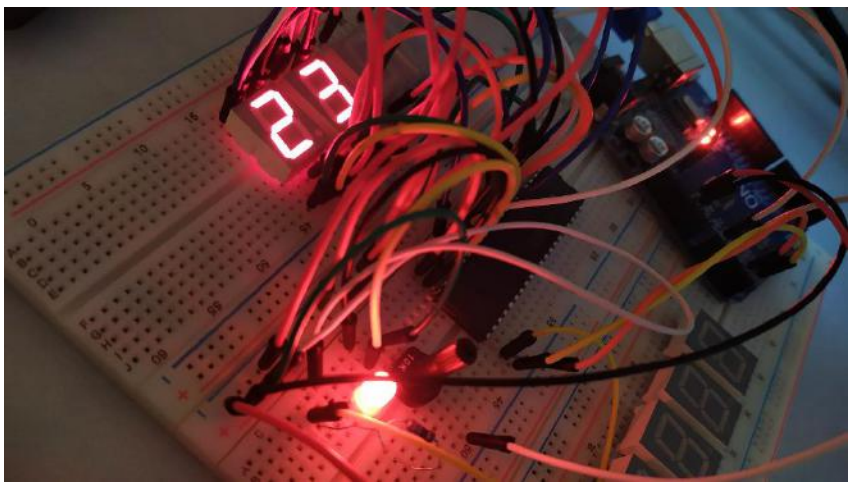


Photo de la manipulation de deux digits à l'aide de 8 trames simulant le cas final

Bien que les afficheurs permettront aux étudiants de visualiser les tensions aux bornes de leurs générateurs depuis leur banc de TP, il nous reste à aborder la remontée des données vers le serveur de la salle E001.

Envoi des données sur le serveur

Pour l'envoi des données sur le serveur, nous nous servons de la bibliothèque *ethernet* qui est open-source et qui dépend d'une autre appelée *spi*. Ces librairies sont par défaut inclues dans l'Arduino IDE et il suffit donc de les inclure en début de fichier pour les utiliser lors du développement. Ils permettent ensuite de faciliter la connexion au réseau pour la puce W5100 que nous avions choisi plus haut.

```
1   #include <Ethernet.h>
2   #include <SPI.h>
```

Utilisation des bibliothèques Ethernet et SPI dans notre programme

Il convient ensuite de paramétrer le client en lui affectant une adresse MAC et une adresse IP et d'indiquer l'adresse IP du serveur avec lequel nous allons communiquer avant d'initialiser la connexion :

```
1   byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x01 }; // RESERVED MAC ADDRESS
2   EthernetClient client;
3
4   IPAddress ip(192,168,1,3);
5   IPAddress server(192,168,1,1);
6   IPAddress myDns(192,168,1,1);
7   IPAddress gateway(192,168,1,1);
8
9
10  void setup()
11  {
12    Serial.begin(9600);
13    Ethernet.begin(mac, ip, myDns, gateway) ;
14    Serial.println(ip) ;
15    if (Ethernet.begin(mac, ip) == 0)
16    {
17      Serial.println("Failed to configure Ethernet using DHCP");
18    }
19  }
20
```

Paramétrage et initialisation de la connexion avec le réseau

Une fois la connexion avec le réseau établie, nous sommes parvenus à envoyer les informations au serveur avec le script suivant :

```
1  /* Données à envoyer */
2  banc = 7 ; dc1 = 123 ; ac = 312 ; dc2 = 230 ;
3
4  // Création de la chaine de caractère à envoyer par requête POST
5  data = "Banc=" + String(banc) + "&DC1=" + String(dc1) + "&AC=" + String(ac) + "&DC2=" + String(dc2) ;
6
7
8  if (client.connect(server,80)) // REPLACE WITH YOUR SERVER ADDRESS
9  {
10   //Serial.println("Client connecté");
11   client.println("POST /pfe/add.php/ HTTP/1.1");
12   client.println("Host: 192.168.1.1"); // SERVER ADDRESS HERE TOO
13   //Serial.println("Host: 169.254.116.49");
14   client.println("Content-Type: application/x-www-form-urlencoded");
15   client.print("Content-Length: ");
16   client.println(data.length());
17   client.println();
18   client.print(data);
19 }
20
21 if (client.connected()) {
22   client.stop();  // DISCONNECT FROM THE SERVER
23 }
24
25
```

Mise en forme de la requête à envoyer (ligne 5) et envoi de celle-ci sur le serveur à l'adresse 192.168.1.1 pour compléter les informations du formulaire add.php

Les données à envoyer au serveur, présentes sur l'extrait de code à la ligne 2, sont mises en forme de la manière suivante :

"*Nom du champ du formulaire*"="*Valeur liée à ce champ*"

Dans le cas où nous envoyons des données pour compléter plusieurs champs du formulaire de destination, ces informations doivent être séparées par le symbole '*&*' sans aucun espace. Un exemple se trouve à la ligne 5 du code précédent où nous souhaitons compléter les champs *Banc*, *DC1*, *AC* et *DC2* du formulaire de destination se trouvant à *192.168.1.1/pfe/add.php*. Ces champs sont donc complétés par les valeurs des variables de même nom, ayant été préalablement converties en chaine de caractères.

Arrive ensuite le test de connexion avec le serveur qui, s'il réussit, aboutit à l'envoi de la requête POST en précisant, dans l'ordre, où trouver le formulaire à remplir, à quelle adresse envoyer la requête, le format de la requête POST, sa longueur et enfin la requête créée plus tôt.

Le format permet de signaler que la requête POST est faite sous la forme d'une chaine de caractères respectant la construction que nous avons détaillé ci-dessus, ce format n'étant pas celui utilisé par les formulaires HTML après avoir cliqué sur un bouton soumettant le contenu de ce formulaire.

Afin de tester notre algorithme, nous avons utilisé le Shield Ethernet d'Arduino sur une carte Arduino Uno. Une fois connectée au serveur par l'intermédiaire d'un câble RJ45, et après avoir correctement paramétré les IP, nous obtenons sur l'application les données fixées dans le script : l'envoi des données vers le serveur fonctionne donc correctement.

RESULTATS

A l'issu de ce projet, nous n'avons donc pu récupérer les tensions sur les bancs de TP. Nous avons toutefois réussi à sortir les cartes *afficheurs* et à les faire fonctionner. Le code écrit pour le prototype reposant sur une carte Arduino permet de récupérer trois valeurs analogiques, et d'afficher les tensions correspondant à ces valeurs sur les afficheurs 7 segments présents sur les diverses cartes localisées sur le banc de TP.

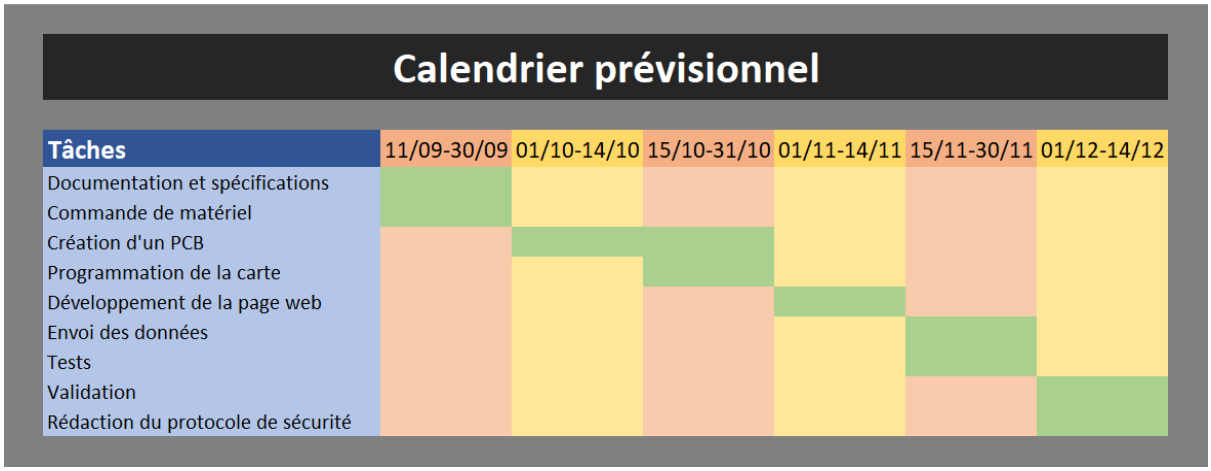
Enfin, ces tensions sont aussi envoyées à un serveur local qui affiche sur un écran les valeurs remontées par les différents bancs de TP dans un tableau qui s'actualise plusieurs fois chaque seconde afin de proposer à l'encadrant du TP des informations en temps réel. Il a aussi été ajouté la possibilité d'ouvrir une page montrant l'évolution des tensions au cours de la journée pour chaque banc de TP.

GESTION DE PROJET

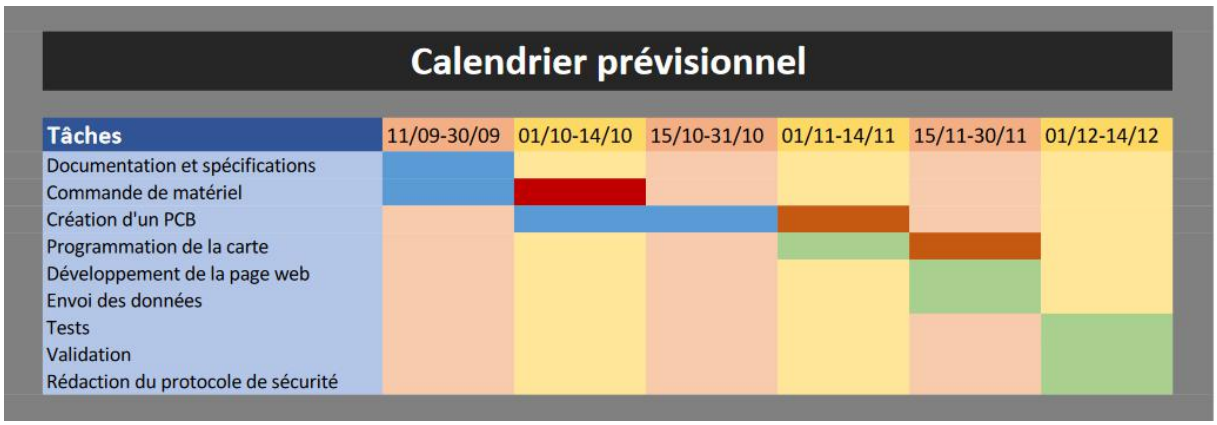
Retour sur la gestion du temps

Gantt initial et évolution chaque quinzaine

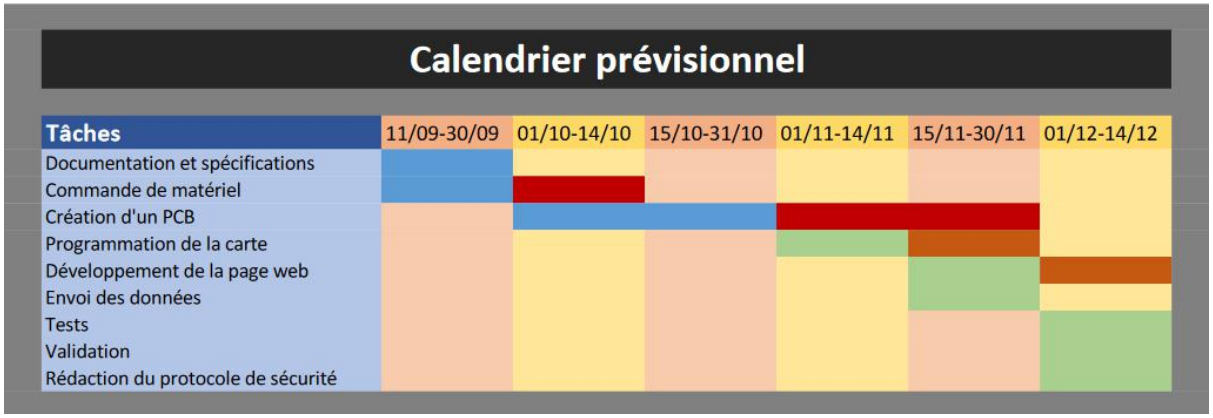
Au début du projet, nous avions élaboré un diagramme de Gantt afin d'estimer la durée de chaque phase du projet. Ce diagramme a été revu au fil du temps et en voici les principaux correspondant aux mises à jour mensuelles :



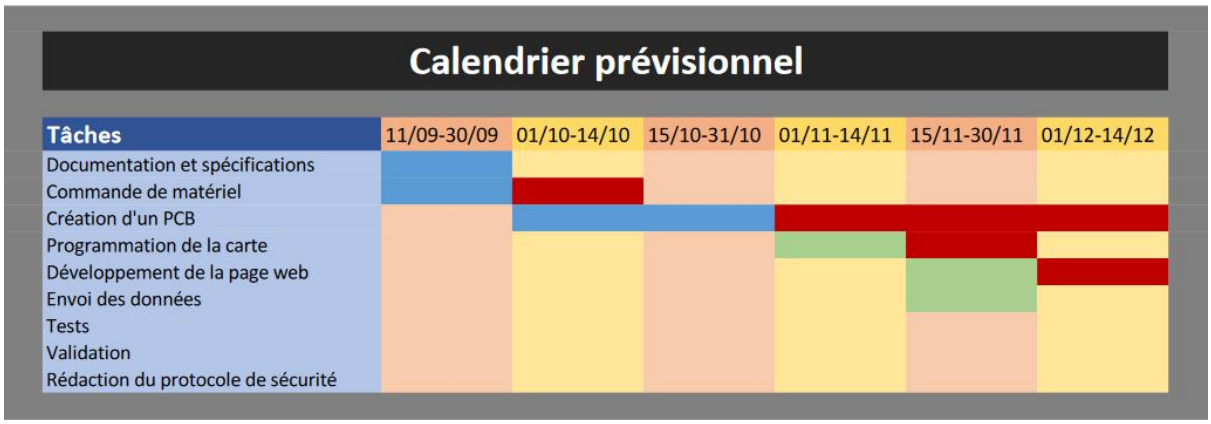
Gantt initial, créé le 26 septembre



Gantt révisé fin octobre



Gantt révisé mi-novembre



Gantt final

Comme nous pouvons le voir, les problèmes rencontrés avec la création de la carte "principale" ont étendu de manière importante la durée qui était destinée à la création des divers PCBs. Ce retard a donc engendré un léger décalage du développement de l'application web, mais a surtout pour conséquences l'échec du projet qui n'a pu être testé sur le banc, et qui n'a donc pu aboutir sur le résultat espéré en début de projet.

Problèmes rencontrés

L'écart entre le temps prévu et le temps utilisé pour accomplir ce projet peut s'expliquer par les raisons suivantes :

- Ce projet nous a demandé de nous former sur des composants électroniques que nous ne connaissions absolument pas. Par exemple, l'amplificateur d'isolation

recommandé par notre tuteur nous a presque pris une semaine pour comprendre le fonctionnement du module et le montage qui l'accompagnait.

- Lors de la commande du matériel, nous avons aussi dû revenir sur différents points et les faire valider à nouveau par notre tuteur : ces retours en arrière ont donc retardé la création des PCBs d'une semaine supplémentaire mais était nécessaire sans quoi le projet n'aurait pu répondre aux critères demandés

- Notre plus grosse erreur sur ce projet a été la suivante : bien que la création des PCBs ait pris un mois comme estimé, le retard accumulé par la commande du matériel nous a poussé à imprimer rapidement la carte.
  Cette carte ayant été dessinée avec une erreur compromettant l'isolation galvanique de celle-ci, il nous a fallu recréer une carte après l'impression de la première carte. Il aurait fallu mieux vérifier cette première carte avant de l'imprimer, même si cela demandait une semaine de retard supplémentaire. Une autre solution aurait été de commencé par la création de la carte "*principale*", plus compliquée et pouvant donc être plus problématique. De cette manière nous aurions pu créer les cartes "*afficheurs*" tandis que la première carte était vérifiée par des personnes plus compétentes.

- Ayant commencé la création des PCBs autour de mi-octobre, cela nous a donc laissé 2 mois pour créer les cartes et les intégrer aux bancs. Bien que l'impression des cartes prennent habituellement une semaine voire moins, ce qui a été le cas pour les cartes "*afficheurs*" qui ont été imprimées en moins de 3 jours, les quatre semaines qui ont été nécessaires à l'impression des deux exemplaires de la carte "*principale*" ont donc représenté un retard de deux semaines qui n'avait pas été pris en compte comme possibilité, même en anticipant les défauts du premier PCB. Ce retard de deux semaines peut être traduit comme 20% de la période n'ayant pu être utilisée suite à des problèmes avec l'externalisation des activités, ce qui est énorme pour un projet.

Coût du projet

Dans cette partie, nous allons revenir sur le coût financier qu'a été ce projet en reprenant les coûts des différentes parties du projet que nous avions détaillé plus tôt :

Description	Coût (€)
Afficheur n°1	9,9952
Afficheur n°2	10,7072
Afficheur n°3	9,9952

Récupération des tensions continues	26,1346
Récupération de la tension alternative	19,5629
Gestion du driver de LEDs	6,438
Envoi des données sur le réseau	7,3695
Traitement des données	9,8151
TOTAL	**99,9727**

Tableau répertoriant les dépenses pour chaque aspect du projet

Nous pouvons donc voir que le coût du projet a approché les 100€ : la partie permettant d'afficher les données a environ couté 30,70 € auquel nous pourrions rajouter presque 5 € pour le driver de LEDs MM5451, la capacité et le potentiomètre se trouvant sur la carte "*Principale*". La récupération des tensions aux bornes des trois sources de tension a couté 45,70 € tandis que l'envoi des données sur le réseau a couté 7,37 €. Enfin, le cœur de la carte "*principale*", comprenant le microcontrôleur ou encore les régulateurs de tensions, a coûté 9,82 € auquel on pourrait donc retrancher les 5€ attribué pour le MM5451.

Nous pouvons aussi relativiser ici le coût des afficheurs étant donné que nous avions tout d'abord choisi des afficheurs 7 segments 703-0169 de Multicomp Pro à 0,344 € par unité mais que nous avons finalement dû utiliser les HDSP-5551 de Broadcom à 3,13 € unité suite à une erreur de stock de Farnell entre le moment où la commande a été passée et le moment où elle a été envoyée. Concernant les afficheurs HDSP-5551, ils ont été fournis par M. FLAMEN qui venait de les récupérer d'un déstockage.

De même, l'oscillateur 16 MHz NCT050C a été récupéré du magasin de composants électroniques de Polytech afin de réduire les coûts du projet. Enfin les condensateurs 100 nF, dont une trentaine a été nécessaire pour ce projet, ont été acheté par centaine sur demande de M. FLAMEN qui a pu en profiter pour obtenir davantage de stock de ce composant : cela a permis de réduire de moitié le coût unitaire de ces capacités.

Toutefois, le prix final aurait pu être baissé de 2 ou 3 euros en choisissant une puce W5500 plutôt que la W5100 choisie ici. En effet, cette puce est semblable à celle choisie, mais semble même offrir après coup de meilleures possibilités pour un prix moindre.

Retour sur les fournisseurs

Durant ce projet, nous avons eu des expériences avec plusieurs fournisseurs lors des différentes étapes du projet. Pour la commande du matériel, nous avons en effet réalisé des commandes chez Farnell et Mouser en passant par l'école, puis une autre chez Farnell sans passer par l'école.

Dans le premier cas, les commandes ont été envoyées le 21 octobre et acceptées le 27 octobre. Les livraisons des composants, elles, ont été faites le 8 novembre pour la commande Farnell et le 12 novembre pour celle provenant de Mouser. Il a donc fallu 3 semaines et demies pour être livré de ces commandes qui ont été livrées incomplètes suite à une erreur de stock entre le moment où la commande a été passée et le moment où l'administration de Polytech l'a acceptée.

Ainsi, nous avons dû refaire une commande plus tard pour obtenir une résistance 30.9 kOhm et 3 diodes Zener 36 V nécessaires à la régulation de la tension en entrée du 78L05 alimentant les amplificateurs d'isolation. Cette commande a cependant permis de s'approvisionner en condensateurs 100 nF ayant une tension de claquage de 50 V devant être placée en parallèle de ces diodes Zener, mais aussi en borniers pour brancher les sources de tensions aux cartes électroniques. Cette commande, passée le 3 décembre, a été reçu seulement deux jours plus tard et permet donc de rapidement palier à des problèmes qui pourraient arriver en passant par l'école.

Pour la création de nos PCBs, nous avons eu l'occasion de faire sous-traiter la gravure par le service de Polytech Lille. La première de nos trois expériences s'est déroulée au mieux et nous avons reçu nos 3 PCBs le 18 novembre alors qu'ils n'avaient été envoyés que le 13. Cependant, après l'envoi du premier exemplaire de la carte "*principale*", nous avons dû attendre le 29 novembre avant d'avoir notre carte, nécessitant un réajustement. Envoyé le 4 décembre, cette seconde version n'a pu être récupérée que le 16 décembre. Les durées de gravure pour ces deux cartes ont dont été respectivement de 2 semaines et de 1 semaine et demie. Cela reste toutefois correct étant donné que les entreprises extérieures, si elles peuvent graver les cartes en moins de deux jours, nécessitent environ une semaine, voire deux, pour ensuite livrer les cartes.

AMELIORATIONS POSSIBLES DU PROJET

Une amélioration possible de ce projet serait l'utilisation d'une puce W5500 plutôt que la W5100 utilisée ici. En effet, cette puce est une version améliorée de la dernière et a l'avantage d'avoir un coût réduit de moitié. Elle n'a pas été utilisée ici suite à un manque de connaissances sur les différentes possibilités proposées par le marché.

CONCLUSION

Ce projet multidisciplinaire nous a permis de tester nos connaissances en fin de cursus et à mettre en avant nos compétences d'élève ingénieur sur une période longue et en autonomie.

Bien que certains points répondent aux attentes du client, ce projet n'a pu aboutir suite à un manque de connaissances sur l'isolation galvanique ayant pénalisé sévèrement la sortie d'une carte électronique fonctionnelle. Ce problème aurait toutefois pu être évité si les efforts avaient été concentrées sur la sortie de la carte la plus problématique plutôt que sur celles qui furent nettement plus simples.

Le projet n'a donc pas été réussi comme espéré. Toutefois les autres aspects du projet ont tous été abordés et réussis. Le projet aurait donc pu être un franc succès s'il avait concerné deux étudiants : un se serait occupé de concevoir des cartes aux normes pour récupérer les tensions aux bornes des générateur, avant de gérer leur intégration aux bancs de TP et d'écrire le protocole de sécurité, tandis que l'autre étudiant se serait occupé du reste, qui a été d'ailleurs été réussi dans ce projet. De cette manière, un prototype aurait clairement pu être envisagé et intégré sur un des bancs de TP.

REFERENCES

MYSQL :

- Datetime :
  https://dev.mysql.com/doc/refman/8.0/en/datetime.html
- Auto init datetime :
  https://dev.mysql.com/doc/refman/8.0/en/timestamp-initialization.html
- Datetime en millisecondes :
  https://dev.mysql.com/doc/refman/8.0/en/fractional-seconds.html
- Créer un utilisateur :
  https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql
- Limiter le nombre de valeur dans une requête SQL :
  https://stackoverflow.com/questions/20878089/php-and-mysql-select-a-single-value

HTML/CSS :

- W3School :
  https://www.w3schools.com/
- Police type 7 segments :
  https://fontlibrary.org/en/font/segment7

PHP :

- Formulaires et requêtes POST :
  https://www.php.net/manual/fr/tutorial.forms.php
- SQL Connect :
  https://www.php.net/manual/fr/function.mysqli-connect.php
- SQL Error :
  https://www.php.net/manual/fr/function.mysql-error.php
- Création dynamique d'un tableau
  https://openclassrooms.com/forum/sujet/boucle-php-pour-creer-un-tableau-html-13827
- Actualiser tableau grâce à JQuery :
  https://www.youtube.com/watch?v=HsvpKa-erpI

Highcharts :

- Site officiel :
  https://www.highcharts.com/

- Highcharts et PHP :
  https://openclassrooms.com/forum/sujet/highcharts-avec-php

Arduino :

- Librairie Ethernet :
  https://www.arduino.cc/en/Reference/Ethernet
- Arduino Ethernet Rev 3 :
  https://store.arduino.cc/arduino-ethernet-rev3-without-poe
- Envoi des données sur MySQL :
  https://www.instructables.com/id/PART-1-Send-Arduino-data-to-the-Web-PHP-MySQL-D3js/
- Envoi des données par requête POST :
  https://eskimon.fr/tuto-arduino-802-arduino-et-ethernet-client
  https://blog.protoneer.co.nz/arduino-http-post-requests/
- Récupération des données analogiques :
  https://eskimon.fr/tuto-arduino-401-les-entr%C3%A9es-analogiques-de-larduino

Amplificateur d'isolation :

- ACPL-C87B :
  https://www.mouser.com/datasheet/2/678/V02-3563EN_DS_ACPL-C87x_2016-09-05-909299.pdf

Driver de LEDs :

- Datasheet du MM5451 :
  http://www.farnell.com/datasheets/57504.pdf?_ga=2.83106198.1277133319.1574261698-2107771478.1573500755
- Exemple de programme :
  https://arduino.stackovernet.com/fr/q/4367
- Tutoriel :
  http://www.bristolwatch.com/ele2/arduino_MM5451.htm

Puce Ethernet :

- W5100 :
  https://www.mouser.fr/datasheet/2/443/W5100_Datasheet_v1.2.5-586411.pdf

W5500 :

- Comparaison W5100 et W5500 :
  https://forum.wiznet.io/t/topic/1396
- Shield Arduino à base de W5500 :
  https://store.arduino.cc/arduino-ethernet-shield-2

Electronique :

- Choix de la capacité de lissage :
  http://nononux.free.fr/index.php?page=elec-brico-outils#!elec-brico-outil-lissage-tension

ANNEXES

Le dossier joint à ce rapport contient les pièces annexes liées à ce projet telles que les schematics et les différentes versions des cartes électroniques dessinées, les fichiers ayant permis de créer l'application web ou encore les différents programmes .ino écrits durant ces trois derniers mois.

RESUME

Rapport du Projet de Fin d'Etudes de Nicolas HAVARD, étudiant en Informatique, Microelectronique et Automatique à Polytech Lille. Ce projet a débuté le 11 Septembre 2019 et s'est achevé le 18 Décembre 2019 et a consisté à étudier une solution de remplacement pour récupérer les tensions aux bornes des différentes sources de tensions disponibles sur les bancs de TP de la salle E001 et les afficher pour les étudiants.

Une seconde partie du projet a demandé la création d'une application web sur une machine dans la salle afin de récupérer l'intégralité des données des bancs et à les afficher sur une page à disposition de l'encadrant du TP pour surveiller les tensions manipulées par chaque étudiant.

Mots-clefs : PFE – Microélectronique – Application web – LAMP – Arduino – Voltmètre – E001