



ACADÉMIE
D'AIX-MARSEILLE

*Liberté
Égalité
Fraternité*



Dossier Finale sur le Projet PMV.



SESSION : 2023

Sommaire

Le Scénario principal du Projet	1-2
Analyse du Système PMV en 2022	3
/Diagramme de Block.....	4
Partie BOURGET Baptiste (IR 1)	5
Reverse Engineering.....	8
/Clonage	
Contenu du Système PMV	8-9
L'ensemble des modifications effectuées	9
/La Boîte de Dialogue : Fenêtre d'Authentification	9-10
/La Fenêtre Principale du programme	10
/Bouton ARRÊT	11-13
/Date de la Session	13-14
/Exportation du Fichier CSV	14-15
/Les problèmes rencontrés au niveau du Programme	15-16
Classe Cl2c	17-18
Gestion de la Base de Données	18
/Les problèmes rencontrés au niveau de la Base De Données ...	19
Contenu de la Base de Données « PMVBdd »	20-22
/Table Authentification	22
/Table Coureurs	
/Table Courses	23
/Table Sessions	
/Vérification des sessions actives	
/Diagramme d'Activité	24
/Résumé	
Conclusion	25
Partie Gallego Simon(EC 1)	26
Planification du projet	
Analyse des différents capteurs	27-29

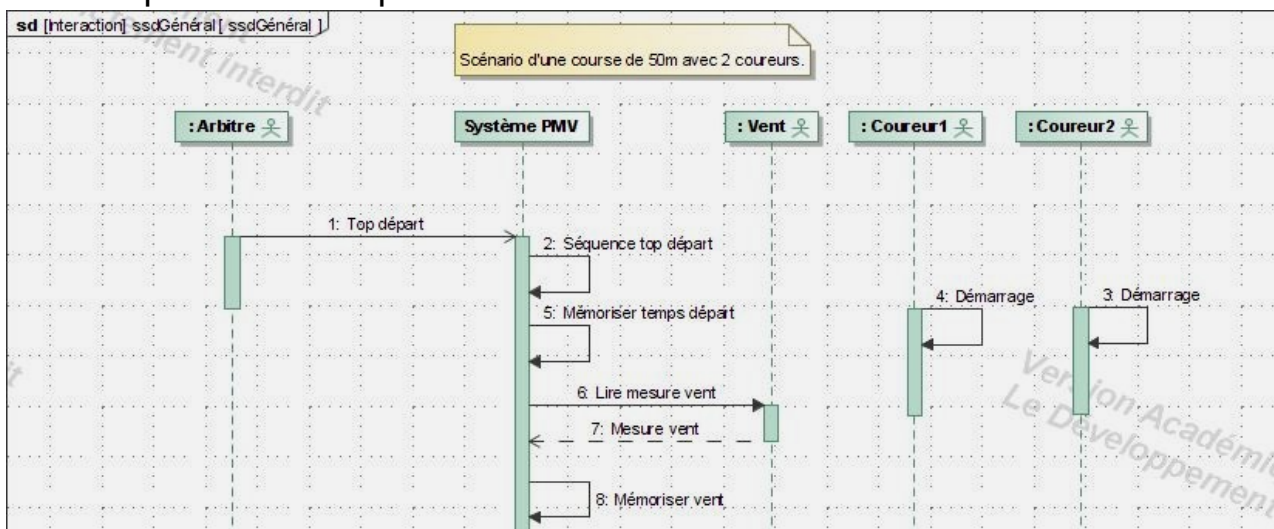
LiaisonXUB9APANL2-Batterie	29-30
Mise en œuvre du capteur	30-32
Régulateur à découpage	32
TSR 3-1250	33
Mise en œuvre du régulateur à découpage	
Détection de la tension de la batterie	33-34
Convertisseur analogique-numérique	34
Conception de la nouvelle carte raspberry	
/Schéma version 2022	35
/Schéma version 2023	36
/Comparaison des deux schémas	37-38
Liste des composants	39
Partie Roda Adan(EC 2)	40
/Capteurs couloir 1	49
/Pour le capteur couloir 2	50-52
/Le buzzer	52
/L'anémomètre et la girouette	53
/La Batterie	
Partie Physique	54
/Le test de la girouette	
Les Empreintes de composants	55-56
Conclusion	56

Le Scénario principal du Projet :

Le but principal du Projet étant de préparer le Système PMV, à la demande du coordinateur des enseignants de sport du Lycée Alphonse Benoît. En effet, il s'agit d'une course de 50 m avec 2 coureurs qui courent à chaque courses.

On s'est donné une limite de 20 courses contenant maximum jusqu'à 40 élèves. Les professeurs ont besoin d'un système de chronométrage précis pour les entraînements de courses.

Comme vous pouvez le voir sur le Diagramme de Séquence, il y a plusieurs acteurs qui ont un rôle particulier.

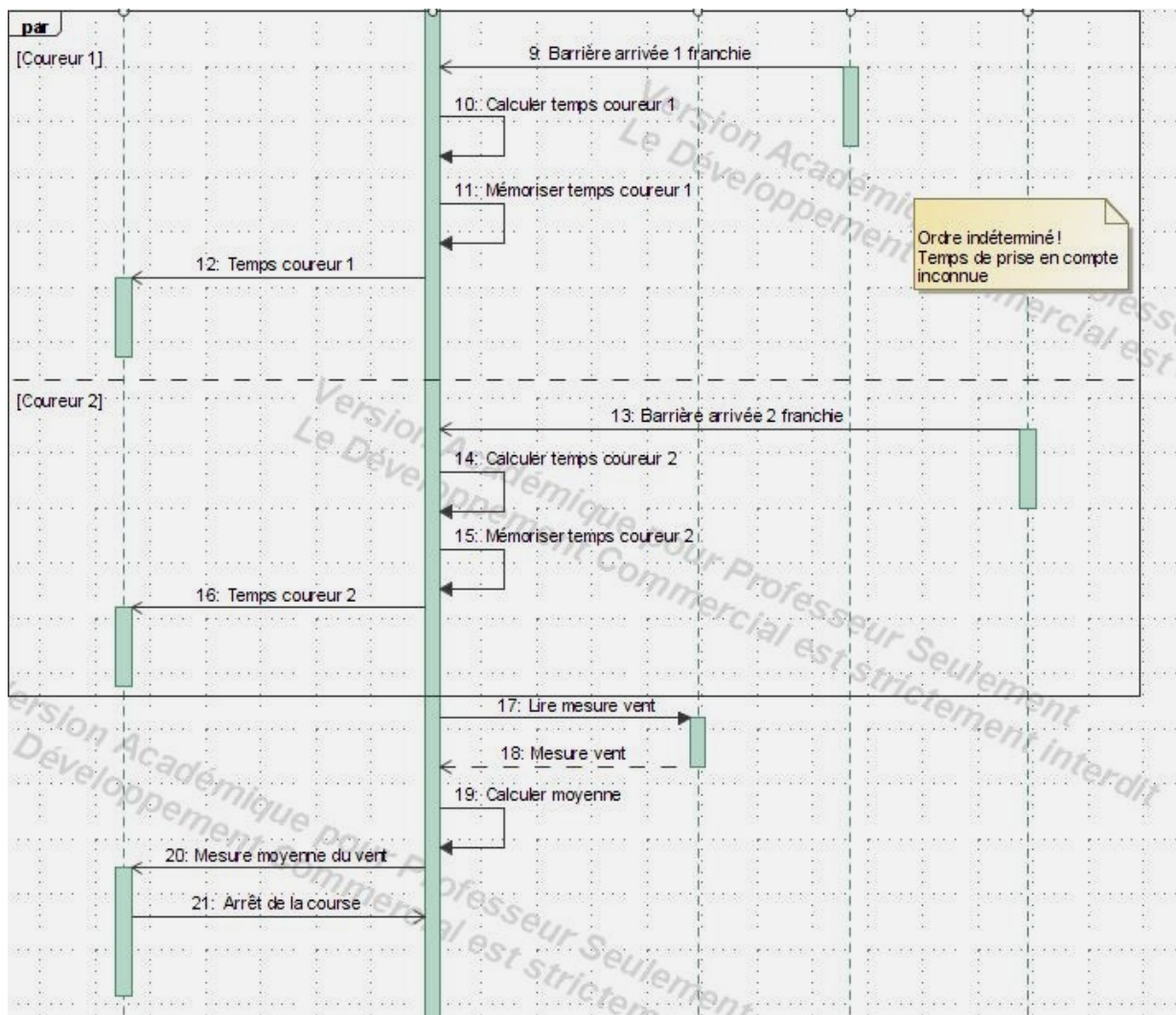


-L'Arbitre (le Professeur) va indiquer les différentes étapes aux coureurs (PRÉPARATION, A VOS MARQUES, PRÊT, PARTEZ).

-Mais avant le processus, il y a l'Anémomètre qui fait partie du système PMV, qui indique la vitesse du Vent. Afin de savoir, si la vitesse du vent dépasse les 15 km/h. Sinon, une alarme va retentir pour annuler la séance. La vitesse du Vent sera mémoriser par la suite.

En effet, le vent peut perturber la performance des coureurs.

-Si le vent est acceptable, alors on lance le top départ pour les coureurs.

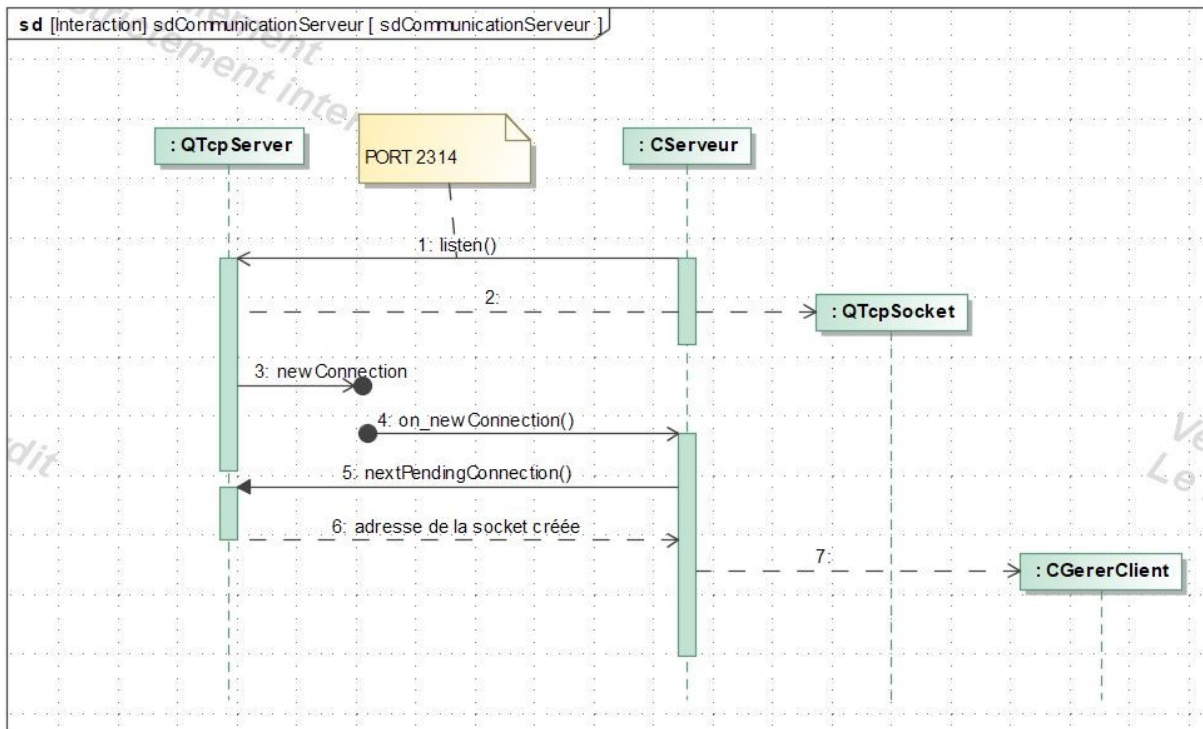


-Une fois qu'ils ont franchi la barrière, leurs temps d'arrivées seront calculer, mémoriser et afficher sur l'IHM; plus précisément, sur l'écran d'Affichage pour l'arbitre.

-Et en même temps, l'anémomètre va mesurer la vitesse du vent, la girouette va se charger d'indiquer la direction et la pression du vent. Pour ensuite, calculer la moyenne de cette vitesse et la course se termine.

Analyse du Système PMV en 2022 :

/Communication Serveur.



Le Diagramme de Séquence représente une analyse sur la Communication Serveur afin de se connecter à Internet par l'intermédiaire de la Tablette. C'est ce qui avait été prévu au tout début du projet.

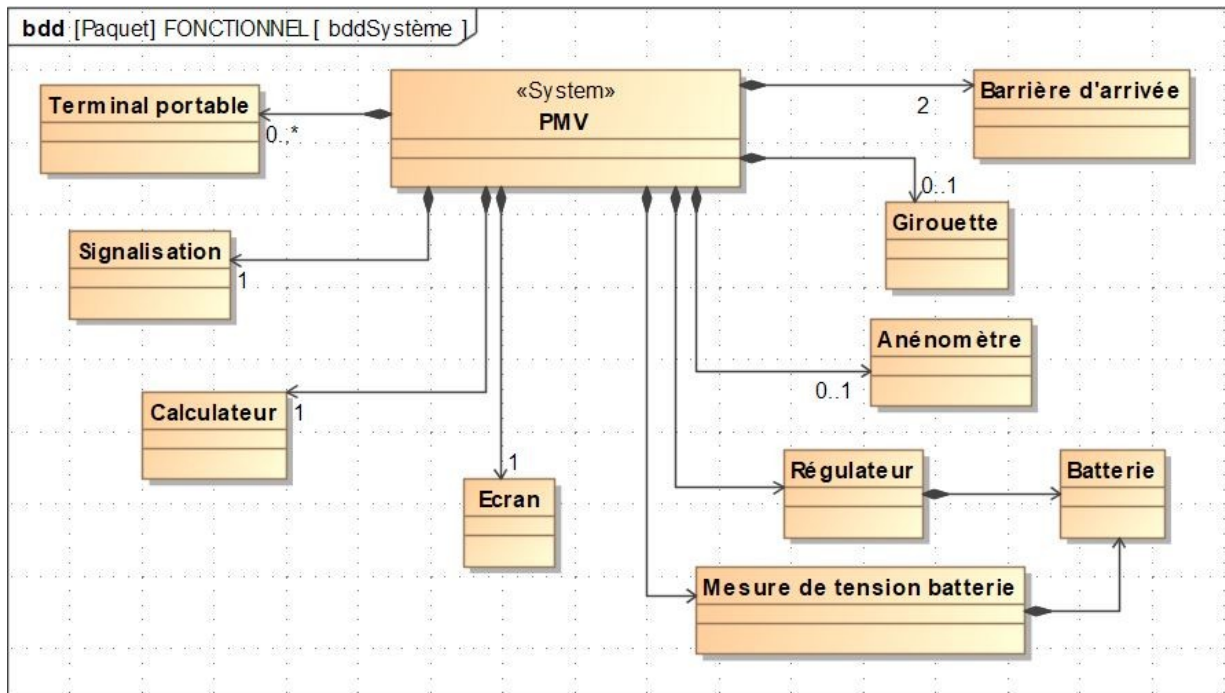
En effet, la Classe **C Serveur** va écouter avec la Méthode «**listen()**» sur le PORT 2314; qui est le port du vrai Serveur, afin de vérifier si le Serveur est actif sur le Réseau. La Classe **QtcpServer** va alors instancier une requête Socket de la Classe **QtcpSocket**, pour nous confirmer que le Serveur à bien reçu notre demande pour la Communication.

Puis, la Classe **QtcpServer** va envoyer un signal pour nous indiquer que la connexion a bien eu lieu et un slot pour recevoir la confirmation du Serveur.

On lui envoie une méthode «**nextPendingConnection()**» pour attendre la création de l'Adresse de la Socket.

Enfin, Le Serveur nous envoie l'Adresse de la Socket créée. On instancie cette adresse vers la Classe **CGererClient** pour se connecter à internet. Mais comme l'IHM de la tablette avait des problèmes, comme c'était compliquer, la communication Client Serveur n'aura pas d'importance pour cette année.

/Diagramme de Block



Sur le Diagramme de Block, la Base De Données dans le Système PMV devait fonctionner de cette façon ci-dessous, pour enregistrer et repérer l'ensemble des composants.

Par exemple, le système PMV est composé d'un seul et unique Calculateur.

Ou, le Système est composé de zéro ou d'un Anémomètre.

Ou alors, le Système PMV est composé de 2 Barrières d'arrivée.

Le Losange Noir est une composition pour indiquer qu'une Classe est composée de cette classe pointée par la flèche.

Partie BOURGET Baptiste(IR 1)

Tout d'abord, j'ai eu l'occasion de pouvoir être choisi pour ce projet sur le système PMV (Prise de Mesure de Vitesse).

Le Projet a été entamé depuis l'année dernière en 2022. Les Étudiants Erwan, Nathan et Alexandre avaient développer le Projet, mais malgré leurs efforts, à cause des conditions du Covid-19, ils n'avaient pas pu finir le Projet. Par conséquent, cette année, je suis en charge d'effectuer des modifications au niveau du Système PMV, en fonction du cahier des charges.

Planification Prévisionnelle :

		PMV	205 h	Mar 03/01/2	Mer 24/05/2		
		Activités communes de départ	10 h	Mar 03/01/23	Mer 04/01/23		
		Lecture du dossier	2 h	Mar 03/01/23	Mar 03/01/23		BOURGET; GALEGO;RODA
		Réunion de Projet 1	45 min	Mar 03/01/23	Mar 03/01/23	3	BOURGET; GALEGO;RODA
		Planification du Plan	7,25 h	Mar 03/01/23	Mer 04/01/23	4	BOURGET
		Développement de la RPI	195 h	Lun 09/01/23	Mer 24/05/23	2	
		Reverse Engineering	7 h	Lun 09/01/23	Mar 10/01/23	5	BOURGET
		Essai de programme	6 h	Mer 11/01/23	Mer 11/01/23	7	BOURGET
		Test de la consommation du Hat	4 h	Lun 16/01/23	Mer 18/01/23	8	BOURGET; GALEGO; RODA
		Changer bouton STOP	1 h	Mer 25/01/23	Mer 25/01/23	9	BOURGET
		Authentification de départ	12 h	Mer 25/01/23	Mar 31/01/23	10	BOURGET
		Surveillance du fonctionnement	8 h	Mer 01/02/23	Lun 06/02/23	11	BOURGET
		Fichier CSV de départ	12 h	Lun 06/02/23	Lun 27/02/23	12	BOURGET
		Rapport de revue 1	3 h	Lun 27/02/23	Mar 28/02/23	13	BOURGET
		REVUE 1	30 min	Mar 07/02/2	Mar 07/02/2		
16		Mettre en place un cadre pour mettre la date	26 h	Lun 13/03/23	Mer 22/03/23		BOURGET
17		Gestion de la fin de la session et de la clef	4 h	Lun 27/03/23	Mar 28/03/23	16	BOURGET
18		Disqualification du coureur	7 h	Mar 28/03/23	Mer 29/03/23	17	BOURGET
19		Gestion de la Base De Données	13 h	Lun 03/04/23	Mer 05/04/23	18	BOURGET
20		Diagramme d'Activité	7 h	Mar 11/04/23	Mer 12/04/23	19	BOURGET
21		Rapport de revue 2	3 h	Mar 02/05/23	Mar 02/05/23		BOURGET
22		REVUE 2	30 min	Mar 02/05/2	Mar 02/05/2		
23		Tension batterie restante	12 h	Mar 02/05/23	Mer 10/05/23	21	BOURGET
24		Fiabilisation du programme	14 h	Mer 10/05/23	Mer 17/05/23	23	BOURGET
25		Rapport de revue finale	6 h	Mer 24/05/23	Mer 24/05/23	24	BOURGET
26		REVUE FINALE	60 min	Lun 12/06/2	Lun 12/06/2		

Planification Réelle :

	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs	Noms ressources
1		PMV	297 h	Mar 03/01/23	Lun 12/06/23		
2		Activités communes de départ	297 h	Mar 03/01/23	Lun 12/06/23		
3		Lecture du dossier	2 h	Mar 03/01/23	Mar 03/01/23		BOURGET;GALEG
4		Réunion de Projet 1	45 min	Mar 03/01/23	Mar 03/01/23	3	BOURGET;GALEG
5		Planification du Plan Prévisionnel	7,25 h	Mar 03/01/23	Mer 04/01/23	4	BOURGET
6		Développement de la RPI	287 h	Lun 09/01/23	Lun 12/06/23		
7		Reverse Engineering	39 h	Lun 09/01/23	Mer 25/01/23		BOURGET
8		Essai du vrai Programme	2 h	Lun 30/01/23	Lun 30/01/23	7	BOURGET
9		Commencer La Revue n°1 du Projet	2 h	Mar 31/01/23	Mar 31/01/23	8	BOURGET
10		Amélioration du Programme	241 h	Mer 01/02/23	Lun 12/06/23		
11		Diagramme de Séquence	19 h	Mer 01/02/23	Mer 08/02/23		BOURGET
12		Modification du bouton STOP en ARRET	13 h	Lun 27/02/23	Mer 01/03/23	11	BOURGET
13		Test de l'authentification de Départ	7 h	Lun 06/03/23	Mar 07/03/23	12	BOURGET
14		REVUE 1	30 min	Mer 08/03/23	Mer 08/03/23		BOURGET
15		Test de l'importation du fichier CSV	20 min	Mer 08/03/23	Mer 08/03/23	13	BOURGET
16		Améliorer l'importation du Fichier CSV	16 h	Lun 13/03/23	Lun 20/03/23	15	BOURGET
17		Diagramme de Séquence (Importation du fichier CSV)	2 h	Mar 21/03/23	Mar 21/03/23	16	BOURGET
18		Faire le Diagramme de Séquence(Exportati du fichier CSV)	4 h	Mer 22/03/23	Mer 22/03/23	17	BOURGET
19		Effectuer des modifications au niveau de l'Exportation du Fichier CSV	12,67 h	Mer 22/03/23	Mer 29/03/23	18	BOURGET
20		Gestion de Base de Données	13 h	Lun 03/04/23	Mer 05/04/23		BOURGET
21		Finalisation de l'onglet Importation	7,33 h	Lun 10/04/23	Mer 12/04/23	20	BOURGET
22		Mise en place d'un QLineEdit pour la date de la session	6 h	Mer 12/04/23	Lun 17/04/23	21	BOURGET
23		Problèmes logiciels	4 h	Mar 02/05/23	Mar 02/05/23	22	BOURGET
24		Disqualification du coureur	6 h	Mer 03/05/23	Mer 03/05/23	23	BOURGET
25		Modification des tables et des données	6 h	Lun 08/05/23	Mar 09/05/23	24	BOURGET
26		REVUE 2	30 min	Mar 09/05/23	Mar 09/05/23		BOURGET
27		Diagramme d'Activité	6 h	Mer 10/05/23	Jeu 11/05/23		BOURGET
28		Réunion de Projet n°2	20 min	Lun 22/05/23	Lun 22/05/23	27	BOURGET; GALEGO;RODA
29		Dossier de Projet FINALE	12,67 h	Mar 23/05/23	Lun 29/05/23	28	BOURGET
30		Fiabilisation du Programme	20 h	Mar 30/05/23	Mer 07/06/23	29	BOURGET
31		REVUE FINALE	60 min	Lun 12/06/23	Lun 12/06/23		BOURGET

Reverse Engineering :

La nouveauté cette année, étant de faire ce qu'on appelle un 'Reverse Engineering'; qui est une analyse personnelle d'un système pour en déduire son fonctionnement interne.

Plus précisément, faire des diagrammes des classes pour avoir un aperçu du fonctionnement du programme.

/Clonage

Avant d'effectuer le Reverse Engineering, par l'intermédiaire de la Konsole sur Rasp OS, il fallait que j'exécute la commande **git clone**.

J'avais besoin de Git Hub pour sauvegarder en Cloud le Dossier du Projet, afin d'éviter de le perdre définitivement.

Cette commande permet d'enregistrer et de recevoir en même temps le Programme sous le FrameWork Qt, en fonction du dossier de destination.

Contenu du Système PMV :



Ceci est un schéma des Broches de la Raspberry Pi.

Ce qu'il y a en rouge ce sont les connecteurs qui sont branchés aux broches respectives, et en bleu leurs descriptifs.

La GPIO n°1 fait passer une Tension de 3,3 V, on l'utilise pour faire descendre en logique bas les autres connecteurs, afin de simuler l'arrivée d'un coureur.

Les GPIO's n°11 et 13 sont des broches qui sont des entrées et sorties numériques.

En effet, ils sont capables de fournir et de recevoir des signaux numériques entre 1 et 0 sous la forme de tensions 0 Volt et 3,3 Volts.

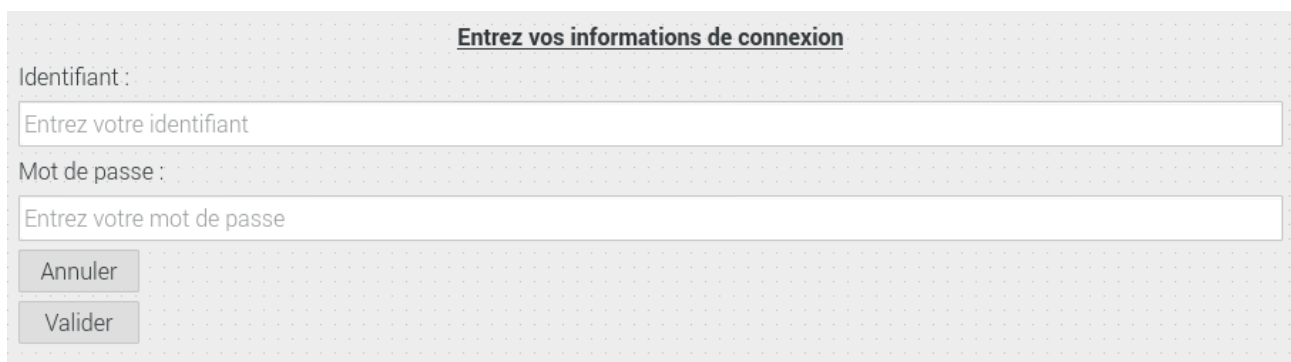
Sauf que, le signal d'entrée est bas ou haut, en le comparant à un seuil de tension. Par défaut, le seuil vaut environ 1,8 volt, mais il n'est pas garanti. Puisque le seuil peut se situer n'importe où entre la tension maximum donnant un 0 et la tension minimum donnant un 1.

Pour le signal de sortie, il y a plusieurs portes qui sont placés en parallèle et en fonction du nombre de portes en service, le courant de sortie peut varier de 2 mA et 16 mA par pas de 2 mA.

Ce courant est disponible aussi bien que quand la porte fournit du courant(niveau haut) que quand elle en absorbe(niveau bas).

L'ensemble des modifications effectuées : /La Boîte de Dialogue : Fenêtre d'Authentification

IHM Authentification 2022 :



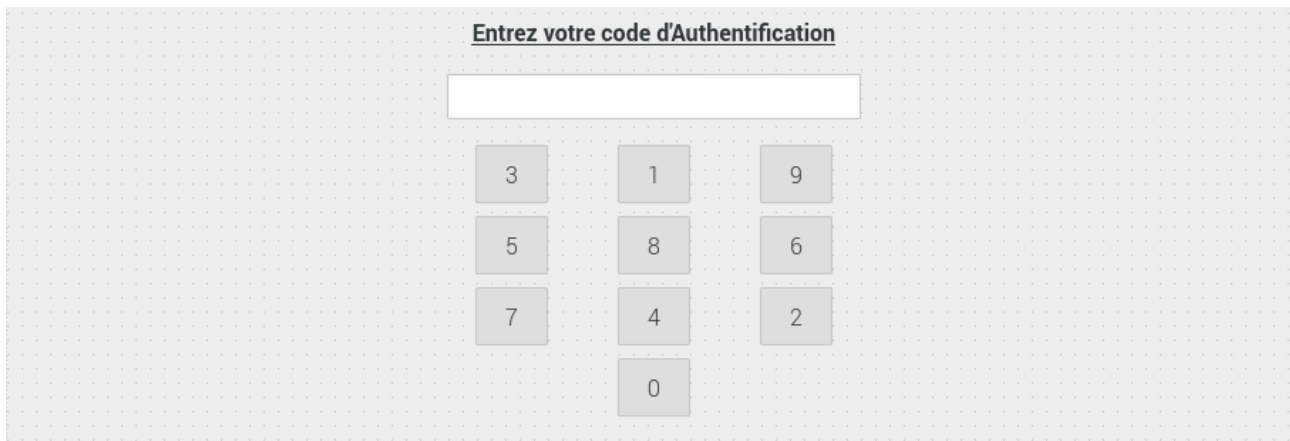
The screenshot shows a login dialog box with a light gray background and a dotted pattern. At the top, the title "Entrez vos informations de connexion" is centered. Below the title, there are two input fields: "Identifiant :" followed by a text box containing "Entrez votre identifiant", and "Mot de passe :" followed by a text box containing "Entrez votre mot de passe". At the bottom left, there are two buttons: "Annuler" and "Valider".

L'année dernière, le cahier des charges indiquait qu'il fallait mettre en place une authentification avec un **IDENTIFIANT** et un **MOT DE PASSE**.

Sauf que, les Étudiants de l'année dernière devaient le faire sur une tablette Tactile, pour effectuer cette Authentification et notamment la Communication Serveur que j'avais citée précédemment.



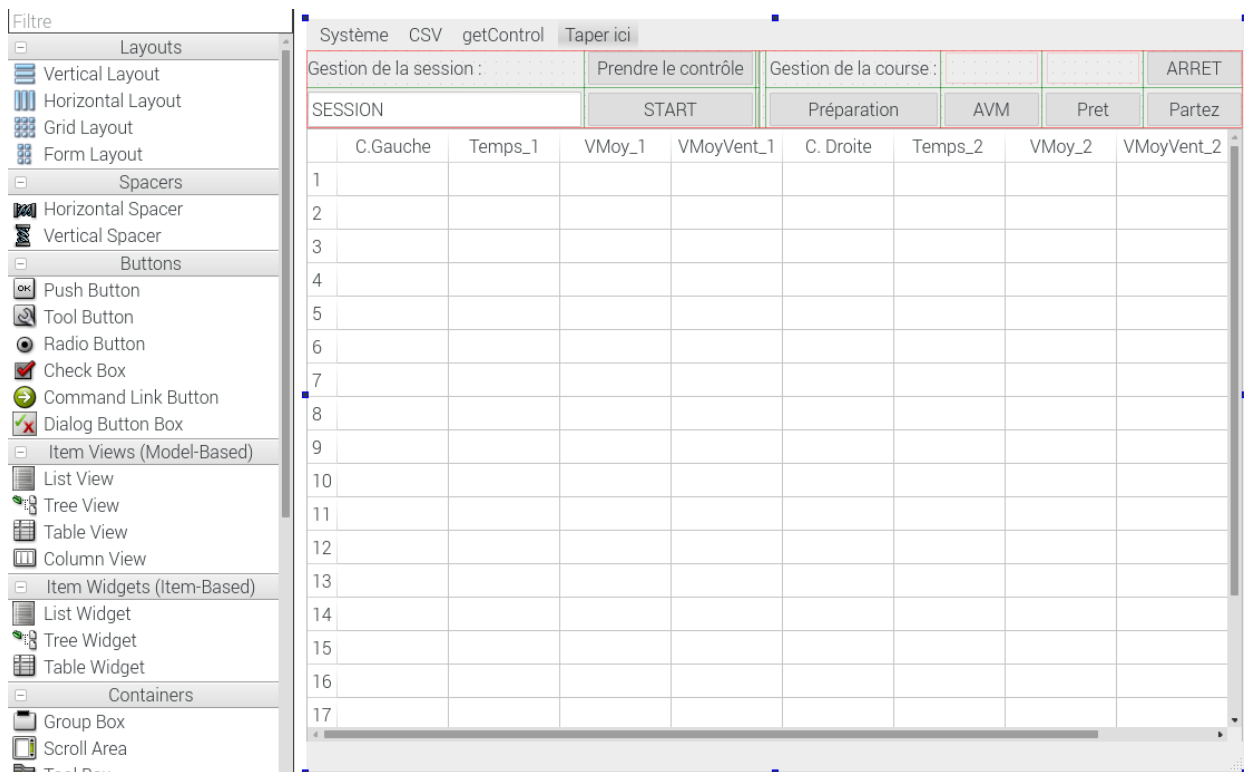
IHM Authentification 2023 :



Cette année, je suis chargé de mettre en place un code d'Authentification pour que l'un des professeurs puissent s'authentifier. C'est un code PIN à 4 chiffres que je dois mettre en place; mais, le professeur à la possibilité de changer de code s'il le souhaite.

Cet IHM est surnommé une boîte de dialogue, car elle apparaît toujours avant la fenêtre principale pour s'authentifier. Elle hérite de la Classe **QDialog** créer par les développeurs de Qt Creator.

/La Fenêtre Principale du programme



Ceci est la fenêtre principale nommée « **CIhm** », dans laquelle où l'on reçoit les résultats des coureurs.

/Bouton ARRÊT

Dans le cahier des charges, il fallait que je modifie le nom du Bouton STOP de la Session en FIN, mais ne surtout pas confondre avec le Bouton STOP pour arrêter la Course.

- Changer le nom du bouton STOP de la session en FIN. Il existe déjà un bouton STOP pour la course.

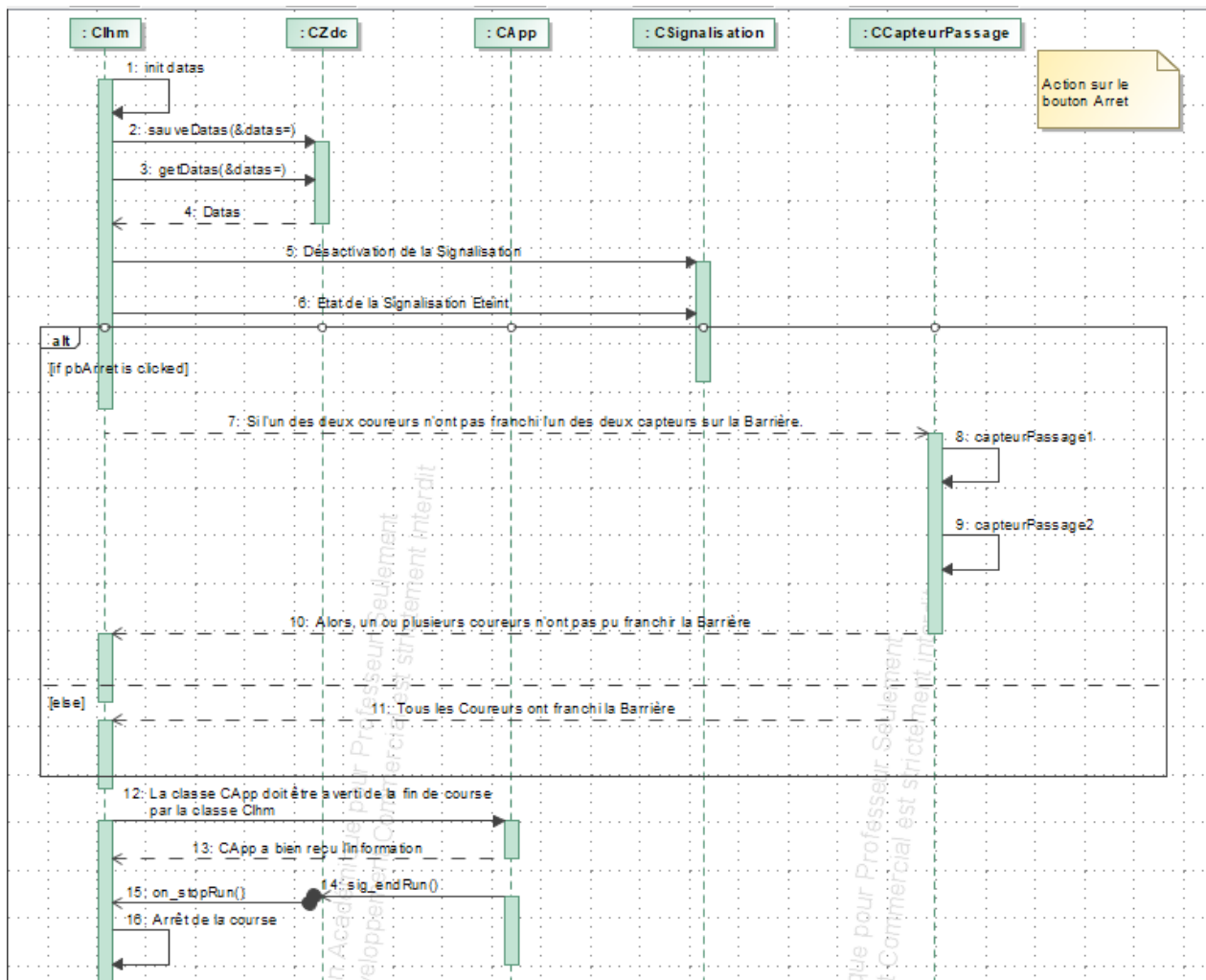


Le Bouton « **START** » permet de démarrer une nouvelle Session et le mot « **FIN** » apparaît pour donner la possibilité au professeur d'annuler la Session.

Le Bouton « **ARRÊT** » permet d'arrêter la course si l'un des coureurs n'a pas franchi la Barrière ou qu'il y est un retardataire.



```
186 void CIhm::on_pbArret_clicked()
187 {
188     T_DATAS datas;
189     _zdc->getDatas(datas);
190     datas.activeSignalisation = false;
191     datas.modeDeFonctionnement = ETEINT;
192     _zdc->sauveDatas(datas);
193
194     if(ui->pbArret->text()=="ARRET")
195     {
196
197         if(_capteurPassage1 && _capteurPassage2)
198         {
199             QMessageBox msgBox;
200             msgBox.setText("Si un ou plusieurs coureurs n'ont pas pu franchir la barrière.\n");
201             msgBox.setInformativeText("Alors, ils sont disqualifié(e)s.\n");
202             msgBox.setDefaultButton(QMessageBox::Ok);
203             msgBox.exec();
204         }
205     }
206     else
207     {
208         ui->pbArret->setDisabled(true);
209         ui->pbPreparation->setEnabled(true);
210
211         }// if => _capteurPassage1 / _capteurPassage2
212
213     }// if
214
215     on_stopRun();// Slot pour la fin de chaque course
216
217     // emit sig_finCourse()
218
219     emit sig_finCourse();
220
221     // avertir _app de l'arret de course
222 }
```



Pour expliquer en détails le Diagramme de Séquence et le code juste au-dessus, ils indiquent l'événement dès qu'on appuie sur le Bouton Arrêt. Mais, le Diagramme de Séquence est ici pour expliquer en détails le fonctionnement du code.

Au début, la Classe Principale « **Clhm** » va enregistrer les données du Bouton ARRÊT dans la Zone de Donnée Commune, pour qu'elle puisse indiquer les données nécessaires à la fenêtre Principale(IHM).

La Signalisation restera désactiver par défaut, puisque dès que les coureurs se mettent à courir, le flux Lumineux reste allumée pendant 5 secondes avant de s'éteindre complètement.

Par la suite, il y a une condition qui indique que, si un des coureurs n'a pas franchi la Barrière au niveau des capteurs de passages.

Alors, un message d'avertissement va apparaître qu'une seule fois, qui va indiquer au professeur que l'un des coureurs n'a pas pu franchir la Barrière.

Dans le cas contraire, les deux coureurs ont tous franchis la Barrière et leurs résultats ont bien été stocké dans le Système PMV et la Base De Données automatiquement.

Mais, l'Application dont la Classe « **CApp** » doit être averti de la fin de la course par la Classe « **Clhm** ». Si l'Application a été averti, alors elle envoie un signal **sig_endRun** pour confirmer qu'elle a bien été averti. Et, le slot **on_stopRun** de la Classe **Clhm** recevra, par la suite, la confirmation de la Classe « **CApp** ».

/Date de la Session

L'onglet que vous voyais actuellement est un QLineEdit, une zone d'édition de texte sur une seule ligne.

Cette zone d'édition de texte où il y a marqué « **SESSION** », va permettre d'afficher en temps réel la date et l'heure courante en **HEURES:MINUTES:SECONDES**.



Le code ci-dessous représente la Date de la Session qui sera affiché dans le QLineEdit :

```
12      /* Lancement de l'interface principale */
13      ui->setupUi(this);
14      QDateTime dt;
15      dt = QDateTime::currentDateTime();
16      ui->leNomSession->setText(dt.toString());
17
```

Ligne 13 : Cette ligne va exécuter le démarrage de la fenêtre principale de la Classe **Clhm**.

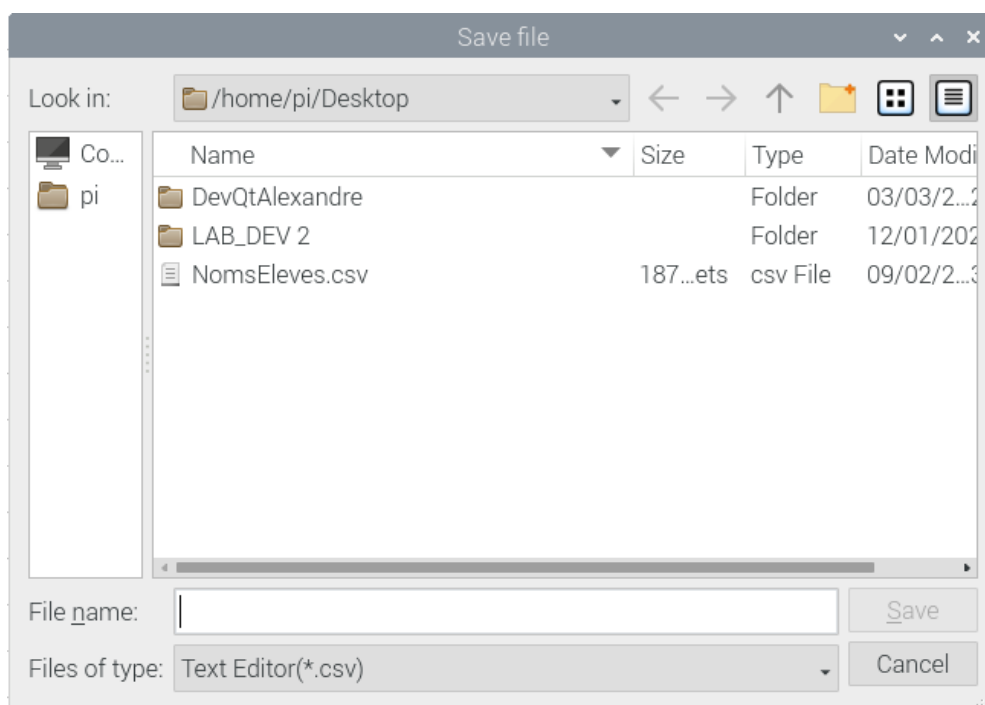
Ligne 14 : On initialise une variable de type **QDateTime** nommé «dt» pour la date.

Ligne 15 : On applique au niveau de cette variable une méthode de QDateTime => **currentDateTime()**; qui va afficher la date et l'heure exprimée en local.

Ligne 16 : A partir de la variable **ui** (User Interface), on va pointer vers un objet de type **QLineEdit**, pour ensuite, l'afficher en format «Text». Mais, la variable «**dt**» qui contient la date et l'heure courante devra être converti en chaîne de caractères à l'aide de la Méthode **toString()**. Ces valeurs seront ensuite affichés dans le QLineEdit nommé «**leNomSession**».

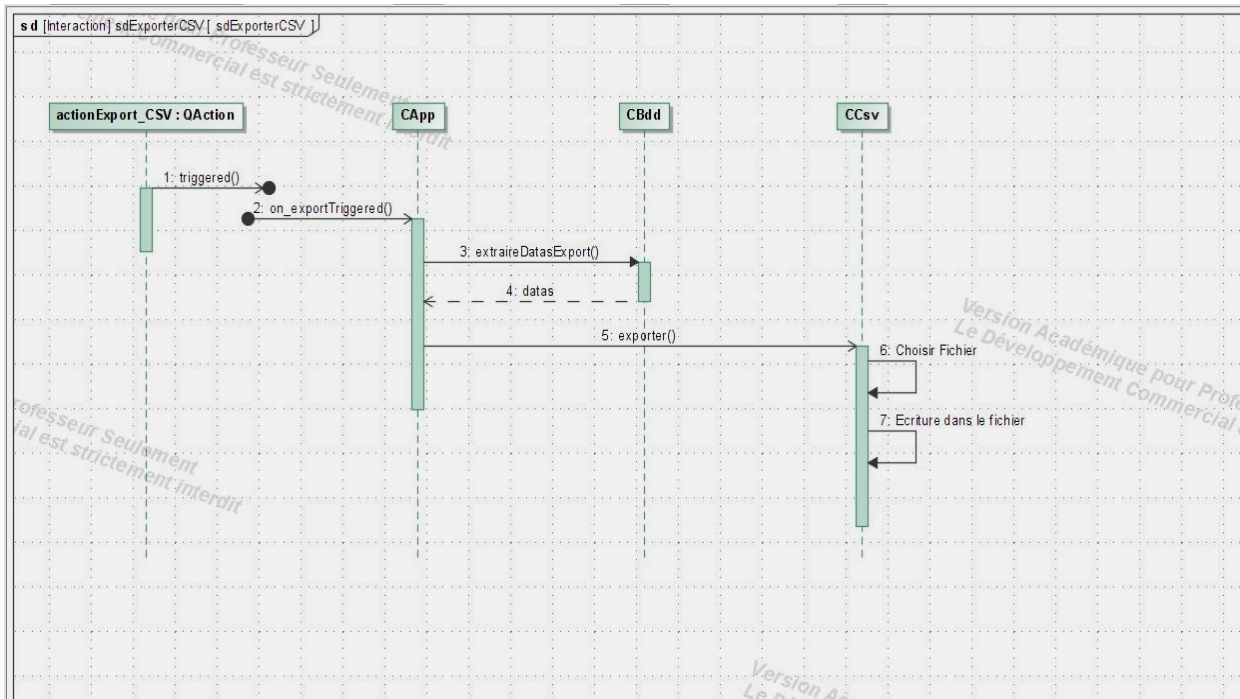
La Date de la Session sera importante pour le Professeur, puisqu'elle permet d'indiquer quand avait eu lieu la session précédente et si elle a bien été fermé correctement.

/Exportation du Fichier CSV



Malheureusement, pour l'exportation du Fichier CSV, je n'arrive toujours pas à comprendre pourquoi la Popup refuse d'exporter et enregistrer le fichier CSV.

Certes, sur cette image, la Popup nommé « **Save file** » permet de sauvegarder un fichier de type «**.csv**» pour ensuite exporter les résultats. Sauf qu'après avoir nommé le fichier et cliquer sur le Bouton «**Save**», il enregistre le fichier, mais refuse d'exporter les données.

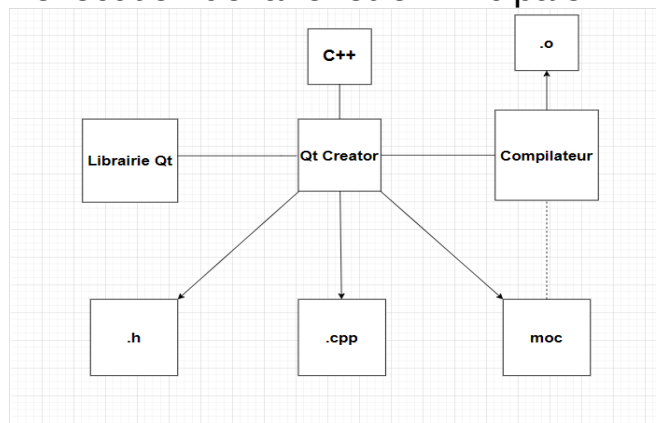


Sur le Diagramme de Séquence, à partir de l'action «**exporter**» faisant partie de la Classe **QAction**. Il y a la possibilité d'écrire dans le fichier, après avoir exporter les données.

Malgré une grosse recherche dans la documentation de Qt Creator et des différents forums. C'est la seule tâche du Projet auquel, je n'arrive pas à déceler ce problème d'envoi des données des coureurs.

/Les problèmes rencontrés au niveau du Programme

J'ai sélectionné le Problème le plus intéressant qui m'était arrivé lorsque je tentais de tester l'exécution de la fenêtre Principale.



Qt Creator en C++ contenant toute sorte de librairies développer par les développeurs de Qt Creator. L'IDE contient des fichiers d'entêtes de type **.h** pour déclarer une Classe, des fichiers qui implémentent les Classes qui ont été déclarés de type **.cpp** et les fichiers temporaires qui seront supprimer par le compilateur notamment les fichiers de type **moc** (Meta Object Compiler) et **.o**.

ccapteurpassage.o	412,7 Kio	22/05/2023 09:46
cgpio.o	419,7 Kio	22/05/2023 09:46
moc_ci2c.o	426,8 Kio	22/05/2023 09:48
cprotocole.o	448,1 Kio	22/05/2023 09:45
moc_ccsv.o	449,2 Kio	22/05/2023 09:48
moc_capp.o	482,8 Kio	22/05/2023 09:47
ccsv.o	483,6 Kio	22/05/2023 09:46
cbdd.o	502,7 Kio	22/05/2023 09:45
capp.o	557,5 Kio	22/05/2023 09:45
cserveur.o	566,4 Kio	22/05/2023 09:45
moc_cbdd.o	591,8 Kio	22/05/2023 09:48
moc_czdc.o	592,1 Kio	22/05/2023 09:47
moc_cgererclient.o	604,1 Kio	22/05/2023 09:48
moc_csignalisation.o	668,9 Kio	22/05/2023 09:47
moc_cprotocole.o	671,1 Kio	22/05/2023 09:47
moc_clogindialog.o	743,3 Kio	22/05/2023 09:47
main.o	789,1 Kio	22/05/2023 09:44
moc_cserveur.o	867,3 Kio	22/05/2023 09:47
clogindialog.o	870,6 Kio	22/05/2023 11:46
moc_cihm.o	881,9 Kio	22/05/2023 09:46
cgererclient.o	887,6 Kio	22/05/2023 09:46
cihm.o	1,1 Mio	22/05/2023 09:44
pmv2023	6,6 Mio	22/05/2023 11:46

Avec l'aide du professeur Mr.Antoine, on compilé un à un tous les fichiers, on s'était rendu compte que le compilateur n'arrivait pas à trouver les fichiers **.o**.

Effectivement, ils étaient dans un autre répertoire. Par conséquent, on a dû refaire un dossier qui est officiellement celui qui contient le Projet PMV 2023. Depuis, l'IHM de la Fenêtre Principale refonctionnait à nouveau.

La seule cause possible de ce problème étant sûrement un déplacement de fichiers.

Classe CI2c:

La Classe CI2c a été faite entièrement par Mr.Antoine, pour la communication I2C qui transite dans l'Anémomètre.

```
1 #include "ci2c.h"
2
3 CI2c::CI2c(QObject *parent, char noBus) :
4     QObject(parent)
5 {
6     m_parent = parent;
7     m_noBus = noBus;
8     m_nblink=0;
9 } // constructeur
10
11 CI2c * CI2c::m_singleton = NULL;
12
13 int CI2c::lire(unsigned char addr, unsigned char *buffer, int lg)
14 {
15     if(ioctl(m_fileI2c, I2C_SLAVE, addr)!=0) { // Règle le driver I2C sur l'adresse.
16         QString mess="CI2c::lire Erreur ioctl acces au bus I2C";
17         qDebug() << mess;
18         emit sigErreur(mess);
19         return -1;
20     } // if ioctl
21     bzero(buffer, lg);
22     QMutexLocker lock(&this->m_mutexI2c); // verrouillage du mutex. Il est libéré en sortie de méthode
23
24     int nb=read(m_fileI2c, buffer, lg);
25     // qDebug() << "CI2c::lire: " << buffer[0] << " " << buffer[1] << buffer[2] << " " << buffer[3] << buffer[4] << " " << buffer[5];
26     return nb;
27 } // lire
28
29 int CI2c::ecrire(unsigned char addr, unsigned char *buffer, int lg)
30 {
31     if(ioctl(m_fileI2c, I2C_SLAVE, addr)!=0) { // Règle le driver I2C sur l'adresse.
32         QString mess="CI2c::ecrire Erreur ioctl acces au bus I2C";
33         qDebug() << mess;
34         emit sigErreur(mess);
35         return -1;
36     } // if ioctl
37
38     QMutexLocker lock(&this->m_mutexI2c); // verrouillage du mutex. Il est libéré en sortie de méthode
39
40     int nb=write(m_fileI2c, buffer, lg);
41     // qDebug() << "CI2c::ecrire: nb=" << nb << " : " << buffer[0] << " " << buffer[1] << buffer[2];
42     return nb;
43 } // écrire
```

```
35     return -1;
36 } // if ioctl
37
38 QMutexLocker lock(&this->m_mutexI2c); // verrouillage du mutex. Il est libéré en sortie de méthode
39
40 int nb=write(m_fileI2c, buffer, lg);
41 // qDebug() << "CI2c::ecrire: nb=" << nb << " : " << buffer[0] << " " << buffer[1] << buffer[2];
42 return nb;
43 } // écrire
44
45 int CI2c::init()
46 {
47     char filename[20];
48     sprintf(filename, "/dev/i2c-%c",m_noBus);
49     if((m_fileI2c=open(filename, O_RDWR))!=-1) { // ouvre le fichier virtuel d'accès à l'I2C
50         QString mess="CI2c::init Erreur ouverture acces au bus I2C";
51         qDebug() << mess;
52         emit sigErreur(mess);
53         return -1;
54     } // if open
55     return m_fileI2c;
56 } // init
57
58 int CI2c::getNblink()
59 {
60     return m_nblink;
61 } // getNblink
62
63 CI2c *CI2c::getInstance(QObject *parent, char no)
64 {
65     if (m_singleton == NULL)
66     {
67         qDebug("L'objet CI2c est créé");
68         m_singleton = new CI2c(parent, no);
69         m_singleton->init();
70         m_singleton->m_nblink=1;
71     }
72     else
73     {
74         m_singleton->m_nblink++;
75         qDebug("singleton est déjà existant");
76     }
77     return m_singleton;
78 } // getInstance
```



```

80 void CI2c::freeInstance()
81 {
82     if (m_singleton != NULL)
83     {
84         m_singleton->m_nbLink--;
85         if (m_singleton->m_nbLink==0) {
86             close(m_singleton->m_fileI2c);
87             delete m_singleton;
88             m_singleton = NULL;
89         } // if mNbLink
90     } // if null
91 } // freeInstance
92

```

Je me suis permis de vous montrer cette classe qui est assez intéressante, parce que, ça m'a permis de comprendre comment on pouvait faire transiter des trames dans l'Anémomètre.

En résumé, cette Classe va vérifier les informations qui transitent dans l'Anémomètre, notamment avec le Singleton qui va pointer vers des objets uniques, afin d'identifier l'adresse du Maître et celui de l'Esclave.

Gestion de la Base de Données :

Je vais à présent, vous présenter la Base De Données.

Alexandre était chargé d'administrer les différentes tables du «**PMVBdd**».



Avec la Tablette, ils devaient fabriquer un logiciel nommé « PMV ».

Ce logiciel contenait la Fenêtre Principale et la Boîte de Dialogue pour s'authentifier.

Ils devaient utiliser le Format **JSON** afin d'effectuer l'échange de données léger, notamment pour la communication Client Serveur et l'accès à la BDD.

Cette année, j'étais en charge de faire le plus de modifications possibles dans la BDD, et qu'elle puisse stocker les données essentiels.

/Les problèmes rencontrés au niveau de la Base De Données

Avant d'énumérer en détails la Base de Données, je devais administrer le «**PMVBdd**» sur le système d'exploitation de la Raspberry Pi; avec le logiciel **dbBrowserForSQLite**.



Ce logiciel avait un dysfonctionnement très sévère, c'est-à-dire qu'on avait aucunes possibilités et ni actions possibles pour la modifier. J'avais demandé des conseils à Mr.DEFRANCE sur le choix des logiciels en rapport avec la BDD.

Il m'avait conseillé :



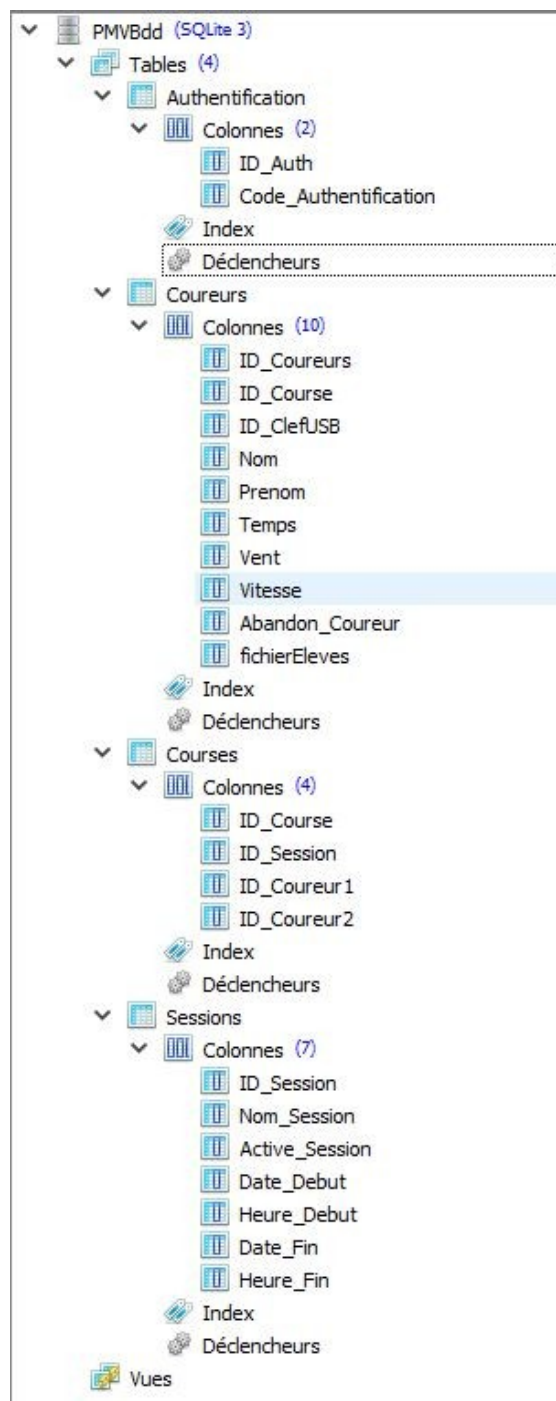
- **WinSCP** : C'est un logiciel qui permet la copie sécurisée de fichiers entre un ordinateur local et un ordinateur distant. Ce logiciel utilise le SSH, OpenSource et le Protocole SCP qui est également supporté.



- **SQLiteStudio** : Un outil qui permet d'administrer facilement une base de données SQLite.

Ces logiciels sont uniquement sur Windows. Donc, si je faisais des modifications, je glisserais le fichier soit de la Rasp OS vers Windows pour avoir les anciennes sauvegardes en cas de plantage, ou soit de Windows vers Rasp OS.

Contenu de la Base de Données « PMVBdd »: Architecture Globale.



La BDD contient 4 tables contenant des données spécifiques.





```

1 CREATE TABLE Coureurs (ID_Coureurs INTEGER PRIMARY KEY, ID_Course INTEGER, Nom TEXT, Prenom TEXT, Temps DECIMAL(4,2), Vitesse DECIMAL(3,2), FOREIGN KEY(ID_Course) REFERENCES Course(ID_Course));
2 CREATE TABLE Courses (ID_Course INTEGER PRIMARY KEY);
3 CREATE TABLE IF NOT EXISTS "Elevés" (
4   "ID_Elevés" INTEGER,
5   "Nom" TEXT,
6   "Prenom" TEXT,
7   PRIMARY KEY("ID_Elevés")
8 );
9 CREATE TABLE IF NOT EXISTS "Authentication" (
10  "ID_Auth" INTEGER,
11  "Login" TEXT UNIQUE,
12  "Password" TEXT,
13  PRIMARY KEY("ID_Auth")
14 );
15 CREATE TABLE IF NOT EXISTS "Sessions" (
16  "ID_Session" INTEGER,
17  "Nom_Session" TEXT,
18  "Active_Session" INTEGER
19 );
20

```

La partie SQL montre l'ensemble du contenu du **PMVBdd** avec les commandes SQL.
 J'ai aussi la possibilité de faire les modifications avec les commandes SQL.









/Table Authentification

	Nom	Type de données	Primary Key	Foreign Key	Unique	Contrôle	Not NULL	Collate	Generated	Valeur par défaut
1	ID_Auth	INTEGER								NULL
2	Code_Authentification	TEXT (4)								0

	ID_Auth	Code_Authentification
1	1	0000










Cette table va vérifier la limite des valeurs du code, quand le professeur va s'authentifier.

/Table Coureurs

	Nom	Type de données	Primary Key	Foreign Key	Unique	Contrôle	Not NULL	Collate	Generated	Valeur par défaut
1	ID_Coureurs	INTEGER								NULL
2	ID_Course	INTEGER								NULL
3	ID_ClefUSB	INTEGER								NULL
4	Nom	TEXT								NULL
5	Prenom	TEXT								NULL
6	Temps	DECIMAL (4, 2)								NULL
7	Vent	INTEGER								NULL
8	Vitesse	DECIMAL (3, 2)								NULL
9	Abandon_Coureur	REAL								NULL
10	fichierEleves	TEXT								NULL




Cette table va contenir les données des 40 élèves et de leurs données après la fin complète de la séance.

/Table Courses

	Nom	Type de données	Primary Key	Foreign Key	Unique	Contrôle	Not NULL	Collate	Generated	Valeur par défaut
1	ID_Course	INTEGER								NULL
2	ID_Session	INTEGER								NULL
3	ID_Coureur1	INTEGER								NULL
4	ID_Coureur2	INTEGER								NULL

Cette table a le rôle de vérifier le nombre de courses; et notamment, les couloirs où les coureurs sont en place.

/Table Sessions

	Nom	Type de données	Primary Key	Foreign Key	Unique	Contrôle	Not NULL	Collate	Generated	Valeur par défaut
1	ID_Session	INTEGER								NULL
2	Nom_Session	TEXT								NULL
3	Active_Session	INTEGER								NULL
4	Date_Debut	DATE								NULL
5	Heure_Debut	TIME								NULL
6	Date_Fin	DATE								NULL
7	Heure_Fin	TIME								NULL

	ID_Session	Nom_Sessi	Active_Ses	Date_Debu	Heure_Deb	Date_Fin	Heure_Fin
1	1	Bidon1	0	NULL	NULL	NULL	NULL
2	2	Bidon2	1	NULL	NULL	NULL	NULL
3	3	Bidon3	0	NULL	NULL	NULL	NULL

La table va afficher la date de début et de fin de la session pour chaque courses.

Les 3 Sessions qui sont en dessous sont ici pour simuler les sessions actives.

/Vérification des sessions actives

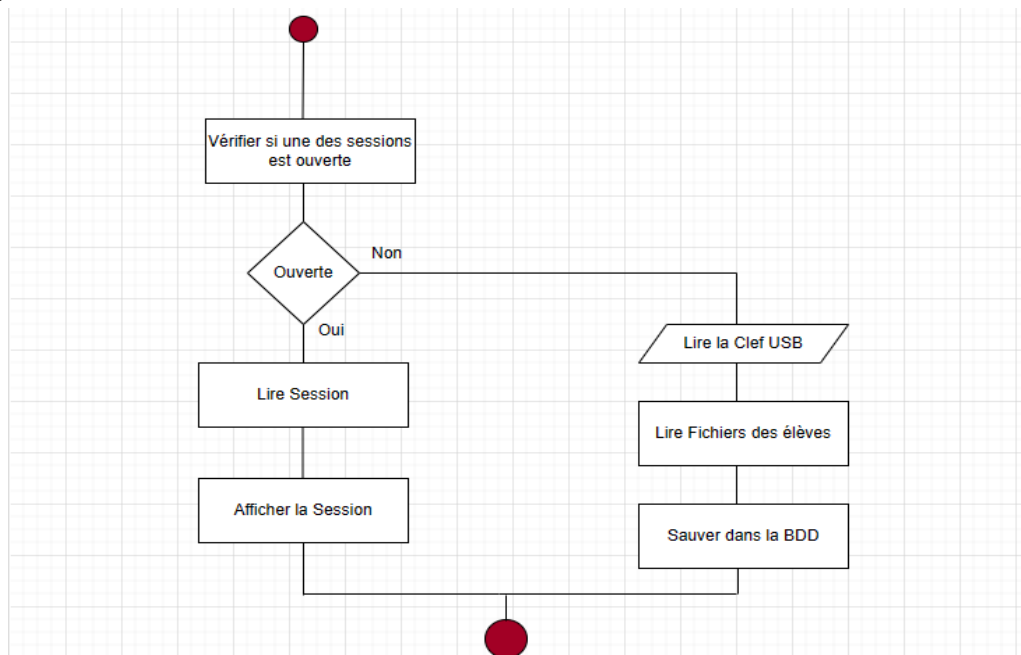
Comme on venait de voir à l'instant, on simule les sessions actives car une seule et unique session peut être active dans la Base de Données.

Le Professeur fera plusieurs sessions lors de l'utilisation du système. Comme par exemple, cela peut arriver que le professeur oublie de quitter la session. Du coup, admettons qu'il ait effectué une Session le 12 Février 2024 et qu'il avait oublier de quitter la session.

La Session restera active dans la Base De Données, jusqu'à ce qu'il redémarre le Système.

Et, un message provenant du programme, lui préviendra de bien quitter l'ancienne Session avant d'entamer sur une nouvelle.

/Diagramme d'Activité



Pour effectuer cette Vérification, il faut que l'application doit vérifier si une des sessions est ouverte.

Si une session est ouverte, alors la Base De Données va lire les Données de la Table «**Sessions**» et l'afficher dans la Fenêtre Principale.

Dans le cas contraire, l'Application va aller lire ou plutôt vérifier la Clé USB, pour chercher le Fichier CSV contenant la liste des élèves avant de les sauvegarder dans la Base De Données.

/Résumé

Pour résumé, l'intérêt d'avoir une BDD dans le Système PMV est d'exploiter une grande recherche de données contenues dans les différentes tables; afin que, les informations puissent correspondre au fonctionnement du Programme.

Conclusion :



Conclusion, la Partie Informatique et programmation de mon côté ont pu porter ces fruits, malgré mes troubles autistiques. Le plus difficile pour moi, étant de planifier et d'analyser le système pour identifier l'endroit où une modification doit être effectuée.

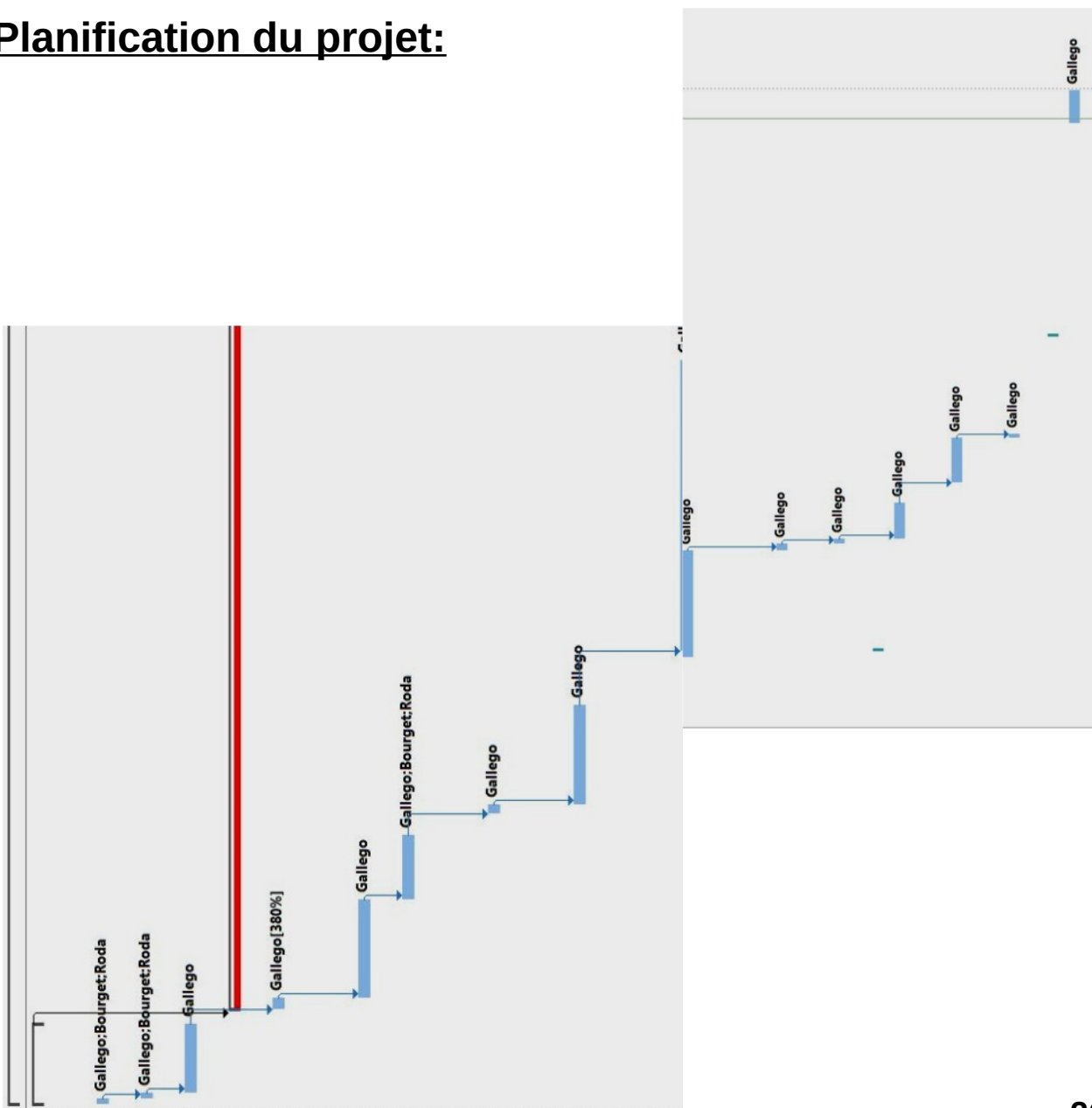
Pour l'exportation du Fichier CSV, je vais essayer de continuer à trouver une solution et j'en parlerais à la REVUE Finale.

Partie Gallego Simon(EC 1)

Pour ma partie, je dois apporter certaines modifications ainsi que des améliorations au sein du projet.

Plus précisément, tester de nouveaux capteurs rétro-réfléchissant, remplacement d'un régulateur, remplacement d'un convertisseur analogique-numérique, ajout d'une structure permettant d'avoir une mesure de la tension de batterie et effectuer une saisie de schéma et de routage pour la conception de la nouvelle carte RPI.

Planification du projet:



Analyse des différents capteurs:

L'année dernière, deux types de capteurs ont été mis en œuvre lors des essais.

-**XUB9APANL2**

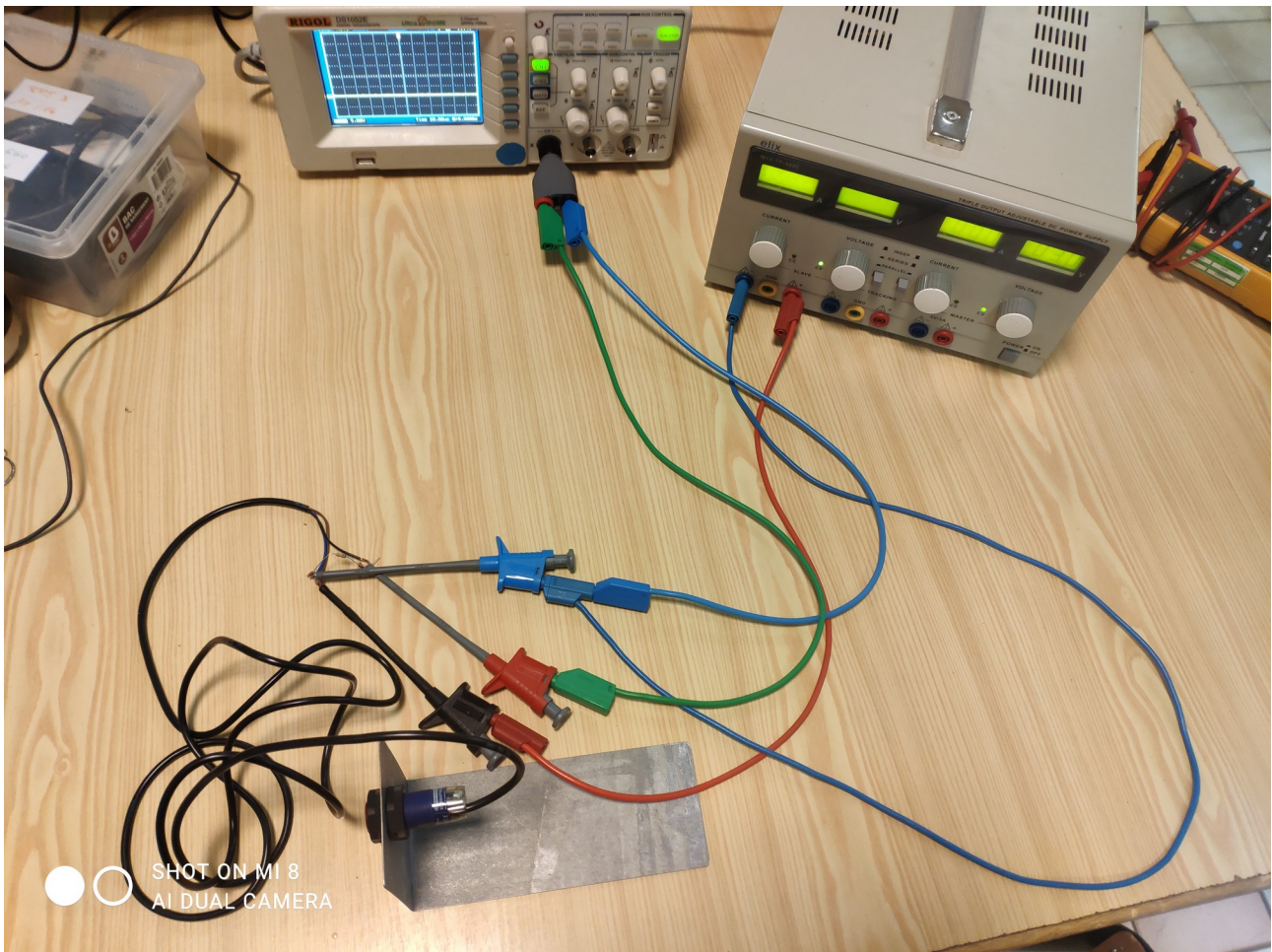
-**PA18CAD10**

Le capteur rétro-réfléchissant **PA18CAD10** est pratique lors de l'installation, car la réflexion se fait directement sur les coureurs. Néanmoins celui-ci n'a pas été gardé car la couleur noir absorbe le faisceau, ils étaient alors obligés d'imposer aux élèves une couleur au niveau des vêtements.

Le capteur rétro-réfléchissant **XUB9APANL2** a besoin d'un réflecteur, pour que celui-ci puisse fonctionner. Tout cela nécessite un positionnement parfait du faisceau avec le réflecteur, ce qui n'est pas à son avantage.

Le capteur **XUB9APANL2** va être sélectionné pour la suite de ce projet, car Monsieur Hortolland a commandé un réflecteur plus grand que le précédent (\approx 4 fois plus grand). Nous allons donc effectuer quelques test pour pouvoir valider ce choix.

J'ai donc procédé au branchement du capteur, ainsi que celui de l'oscilloscope pour pouvoir étudier son fonctionnement. Le capteur doit être alimenté entre 12-24V.



Après avoir tout mis en fonctionnement, je me suis rapidement aperçu que la taille du réflecteur avait un réel impact sur son positionnement, car le petit réflecteur demande beaucoup plus de précision, alors que le nouveau est plus pratique et plus rapide lors du calibrage.

Une fois la praticité validée, j'ai pu analyser le fonctionnement du capteur **XUB9APANL2**. Celui-ci est à l'état haut quand il reçoit son signal réfléchi et passe à l'état bas une fois que son faisceau est coupé par un obstacle.

Nous avons donc fixés le réflecteur et le capteur sur des équerres en fer, tout cela est fixé sur des trépieds réglables au niveau de la hauteur. On le positionne sur un sol plat, le calibrage est toujours simple à effectuer.



Liaison XUB9APANL2-Batterie :

Pour pouvoir alimenter notre capteur à l'extérieur, nous allons nous appropriés d'une batterie de 12V en tension continue. Pour réaliser cette liaison entre les deux, nous avons soudés une connectique pour relier les 2 pôles.



Mise en œuvre du capteur:

Suite à tous ces essais effectués en classe, nous allons mettre en œuvre toute cette structure sur la piste d'athlétisme qui se situe juste à coté du lycée.



Le calibrage sur la piste est aussi simple et rapide qu'en classe.

Les pieds réglables sont très utiles lors de ce projet, car à l'une des extrémité de la largeur de la piste, se trouve un petit rebord d'une hauteur de 2-3cm. Ce qui aurait pu aller à l'encontre lors de ce positionnement, si ce réglage de la hauteur n'avait pas été présent.

Pour la batterie, nous avons pu la positionner sur l'équerre où le capteur est fixé. Ce positionnement est temporaire, nous avons fait comme cela pour réaliser ces test.



Régulateur à découpage:

Sur notre carte actuelle, un régulateur à découpage est disposé. Néanmoins celui-ci est mal dimensionné, il délivre une tension trop basse et pour la consommation assez élevée du montage qu'il alimente.

Nous avons donc cherchés avec Monsieur Hortolland, un régulateur qui pourrait éventuellement correspondre à nos attentes.

Après nos recherches, le régulateur TSR 3-1250 a été sélectionné pour ce changement.

TSR 3-1250 :



Mise en œuvre du régulateur à découpage :

Après la réception de celui-ci, nous avons pu le mettre en œuvre.

Il est parfaitement dimensionné, car il nous délivre une tension adéquate pour pouvoir alimenter notre carte Raspberry; tout en éliminant le message d'erreur qu'on pouvait apercevoir lors du démarrage.

Détection de la tension de la batterie :

Comme sur tout appareil électronique (téléphone, tablette ou pc), un message d'alerte nous informe une fois que le pourcentage de l'appareil est trop bas.

Pour cela, nous devons mettre en place une détection de la tension.

Nous allons relever la tension à la sortie de la batterie une fois que le message d'erreur apparaît.

Nous pourrions ensuite ajuster cette tension avec un pont diviseur de tension pour pouvoir accéder au convertisseur analogique-numérique.

Convertisseur analogique-numérique :

Suite à l'ajout de la girouette et du pont diviseur de tension, nous avons du procédé au remplacement du convertisseur analogique-numérique.

Le convertisseur de l'année précédente (**ADC101C021**) n'était plus conforme à nos attentes pour cette année, car celui ne dispose pas assez d'entrées pour pouvoir accueillir les ajouts cités juste au-dessus. Nous avons du chercher un convertisseur qui possède plus d'entrées.

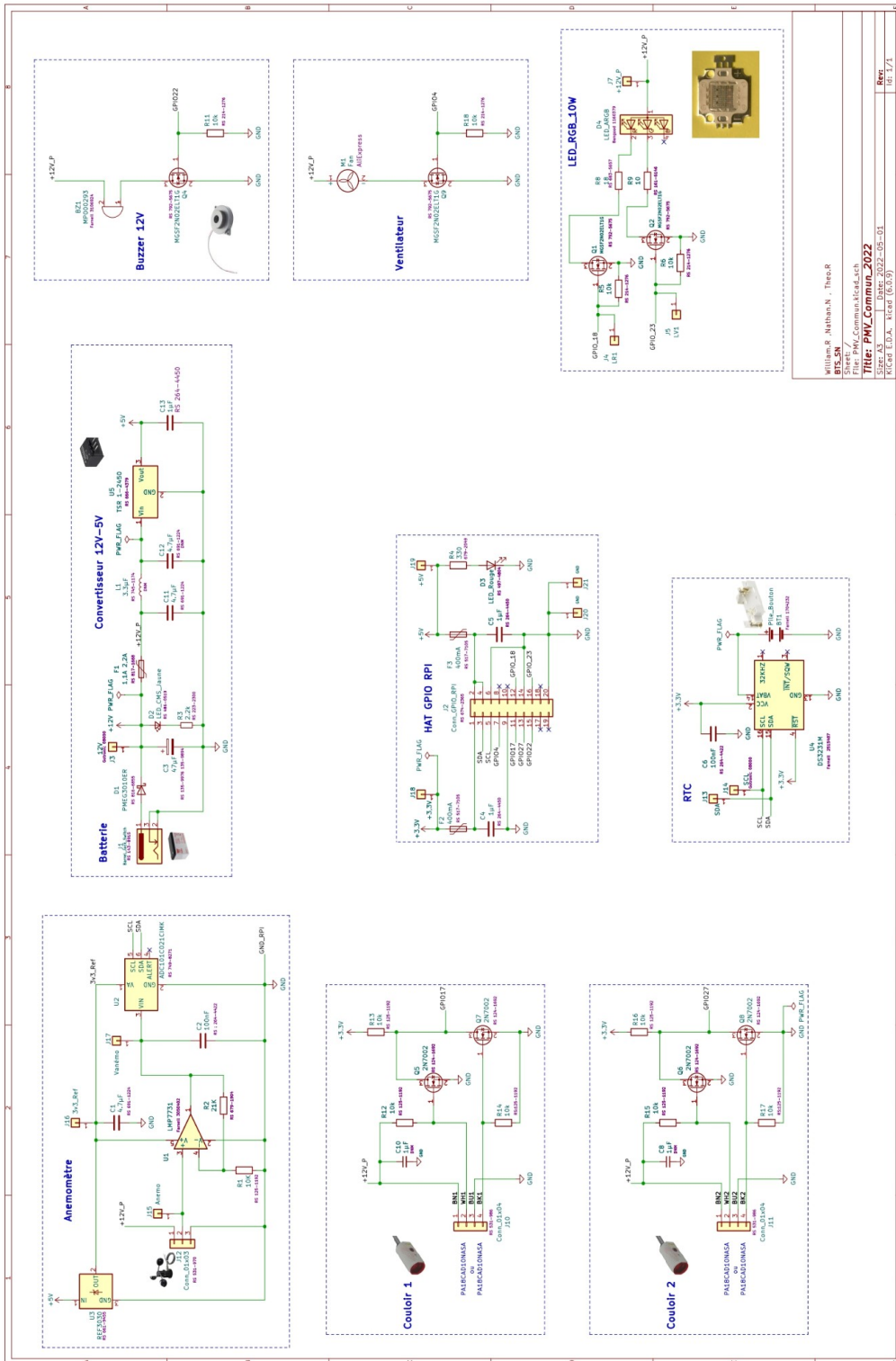
Le convertisseur **ADC MAX11607** correspond exactement à nos attentes. Celui-ci dispose de 4 entrées analogique-numérique, nous allons donc le garder pour la suite de ce projet.

Conception de la nouvelle carte Raspberry :

Après avoir validé toutes les modifications réalisés sur la carte Raspberry, Nous allons pouvoir débuté la conception de cette nouvelle carte.

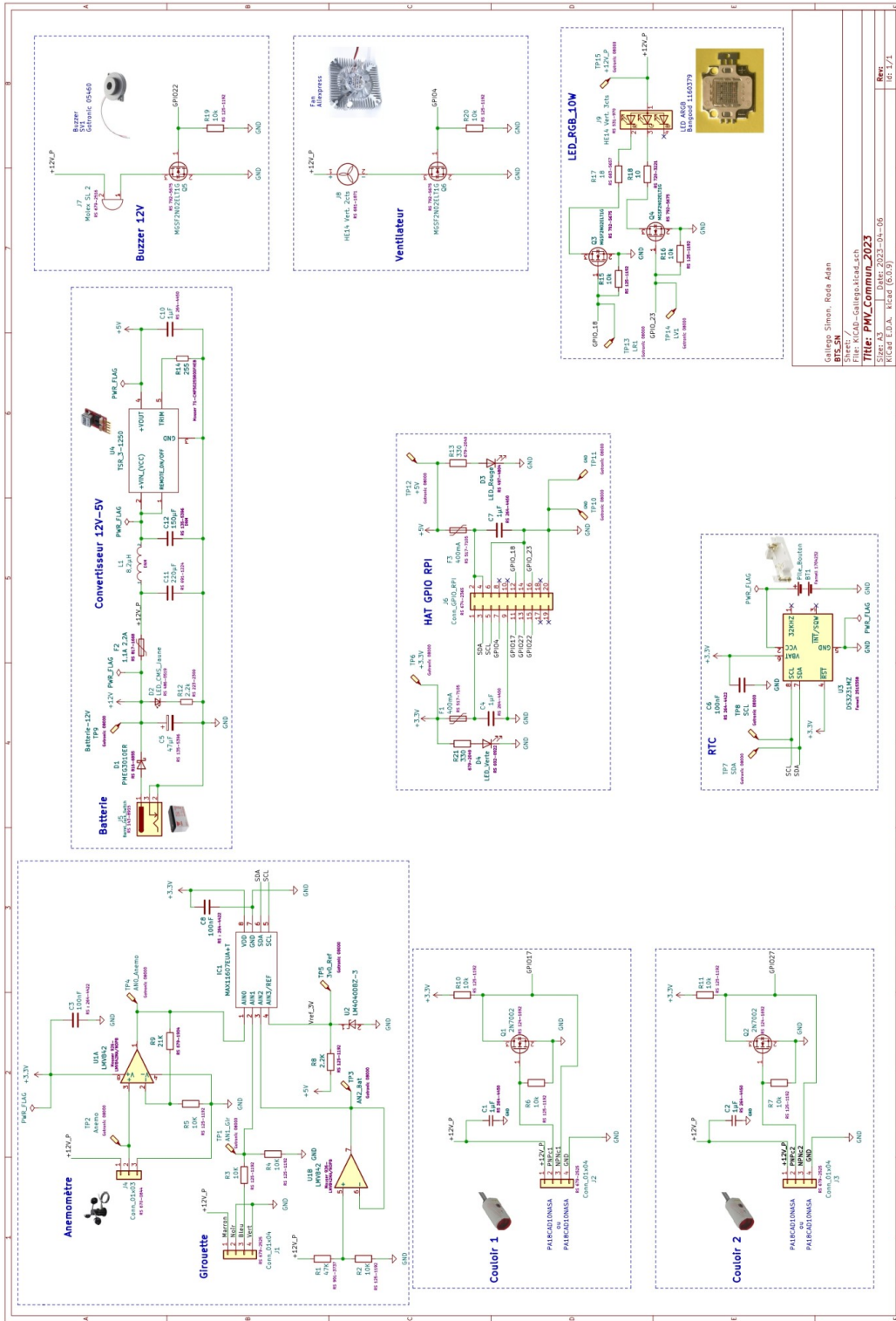
Dans un premier temps, j'ai dû reprendre les schémas des différents circuits réalisé par les élèves de l'année dernière, pour pouvoir réalisés les modifications.

/Schéma version 2022 :



William S. Nabhan, N. Thero R
Sheet /
Project /
Version /
Size /
Scale /
Client /
Doc. /
Rev. /

/Schéma version 2023 :



/Comparaison des deux schémas :

Il y a quatre changements majeurs sur cette carte après cette modification :

-L'ajout d'un connecteur à 4 broches pour pouvoir alimenter la girouette.

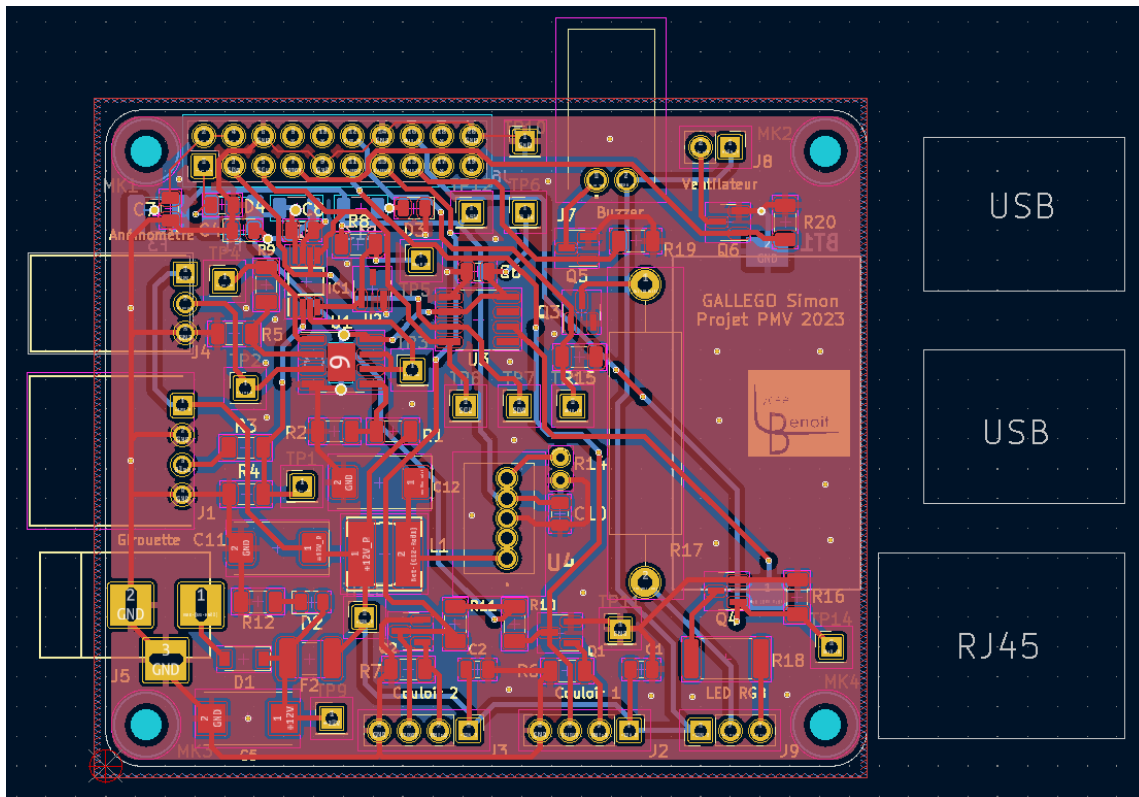
-L'ajout d'un pont diviseur de tension suivi d'un amplificateur suiveur pour la détection de batterie.

-Le remplacement du régulateur de tension pour l'alimentation du Hat Raspberry.

-Le remplacement du convertisseur analogique-numérique pour pouvoir accueillir la girouette ainsi que la détection de batterie.

Une fois les schémas terminés, J'ai pu commencer le routage de la carte Raspberry.

Après avoir importés tout les composants, les avoir positionnés ainsi que les reliés, nous avons pu finaliser la conception de cette carte.



Nous avons mis les connecteurs tout au long de la périphérie de la carte, et nous avons fais au mieux pour pouvoir accéder, plus facilement, aux nombreux points de test qu'on a pu ajouter à la fin de la disposition des composants.

Certains composants ont besoin d'un minimum de réflexion pour leur placement sur la carte. Plus particulièrement, pour les condensateurs, ce type de composant doit être le plus rapproché possible de la partie qui l'alimente, pour pouvoir stabiliser au maximum la tension.

Liste des composants :

	Valeurs :	Codes Commandes :	Quantités :
BT1	Pile_Bouton	Farnell 1704232	1
> C1, C2, C4, C7, C10	1µF	RS 264-4450	5
> C3, C6, C8	100nF	RS : 264-4422	3
C5	47µF	RS 135-5396	1
C11	220µF	RS 691-1224	1
C12	150µF	RS 135-5396	1
D1	PMEG3010ER	RS 816-6855	1
D2	LED_CMS_Jaune	RS 486-0519	1
D3	LED_Rouge	RS 497-4804	1
D4	LED_Verte	RS 692-0922	1
F2	1,1A 2,2A	RS 817-1668	1
> F1, F3	400mA	RS 517-7105	2
IC1	MAX11607EUA+T	Mouser 700- MAX11607EUA+T	1
J1	Conn_01x04	RS 679-2525	1
> J2, J3	Conn_01x04	RS 679-2525	2
J4	Conn_01x03	RS 670-0844	1
J5	Barrel_Jack_Switch	RS 143-8915	1
J6	Conn_GPIO_RPI	RS 674-2365	1
J7	Molex SL 2	RS 679-2516	1
J8	HE14 Vert. 2cts	RS 681-1871	1
J9	HE14 Vert. 3cts	RS 531-970	1
L1	8,2µH	RS 228-7838	1
> Q1, Q2	2N7002	RS 124-1692	2
> Q3-Q6	MGSF2N02ELT1G	RS 792-5675	2
R1	47K	RS 901-3737	1
> R2-R5	10K	RS 125-1192	4
> R6, R7, R10, R11, R15,	10k	RS 125-1192	8
R8	2.2K	RS 125-1192	1
R9	21K	RS 679-1904	1
R12	2.2k	RS 223-2300	1
R14	255	Mouser 71-CMF50255R00FHFB	1
R17	18	RS 683-5657	1
R18	10	RS 720-3221	1
> R13, R21	330	679-2049	2
TP1	AN1_Gir	Gotronic 08000	1
TP2	Anemo	Gotronic 08000	1
TP3	AN2_Bat	Gotronic 08000	1
TP4	AN0_Anemo	Gotronic 08000	1
TP5	3v0_Ref	Gotronic 08000	1
TP6	+3.3V	Gotronic 08000	1
TP7	SDA	Gotronic 08000	1
TP8	SCL	Gotronic 08000	1
TP9	Batterie-12V	Gotronic 08000	1
> TP10, TP11	GND	Gotronic 08000	2
TP12	+5V	Gotronic 08000	1
TP13	LR1	Gotronic 08000	1
TP14	LV1	Gotronic 08000	1
TP15	+12V_P	Gotronic 08000	1
U1	LMV842	Mouser 926-	1
U2	LM4040DBZ-3	LMV842MA/NOPB	1
U3	DS3231MZ	Farnell 2909767	1
U4	TSR_3-1250	RS 781-3288	1

Partie Roda Adan(EC 2)

Dans ce projet PMV (Prise De Mesure Vitesse), on m'a attribué le rôle de EC 2 où je dois analyser la structure proposé sur la version 2022, concevoir un hat rpi permettant de récupérer les données envoyer par l'anémomètre et la girouette (qui est nouveau sur le projet). Pour finir je dois concevoir une interface pour les capteurs qui permettront de mesurer le temps.

Voici une la comparaison finale entre le diagramme de GANTT prévisionnelle et le diagramme actuelle :

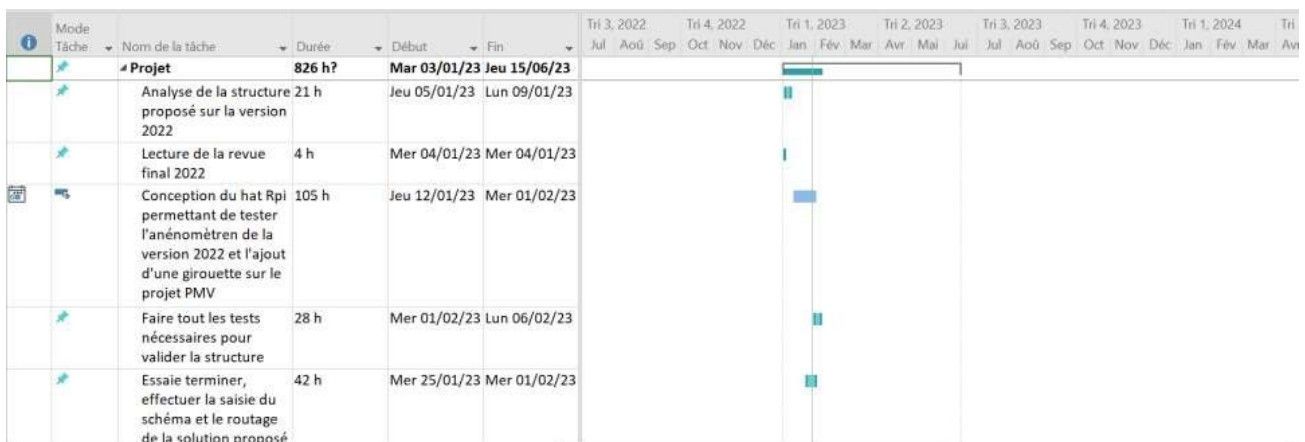
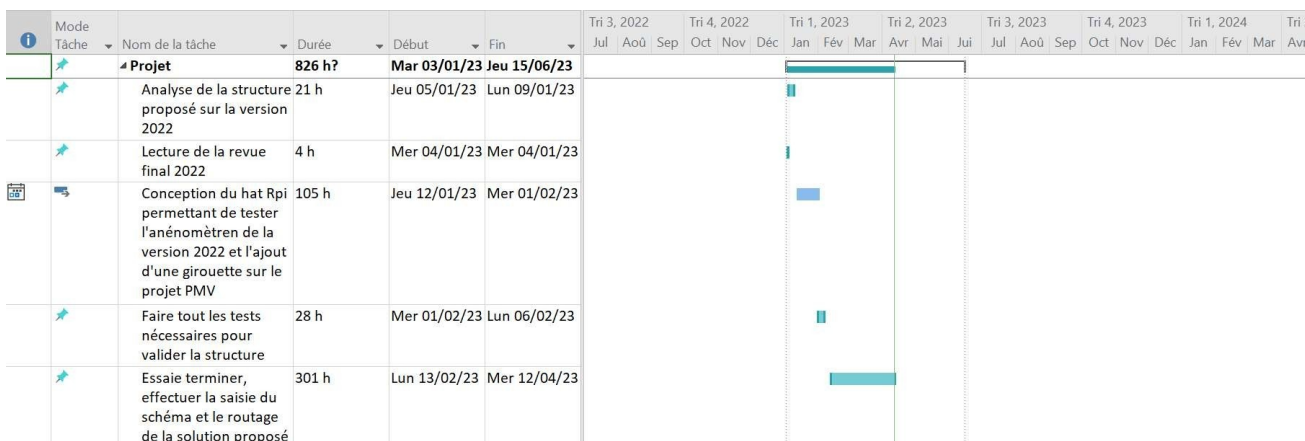
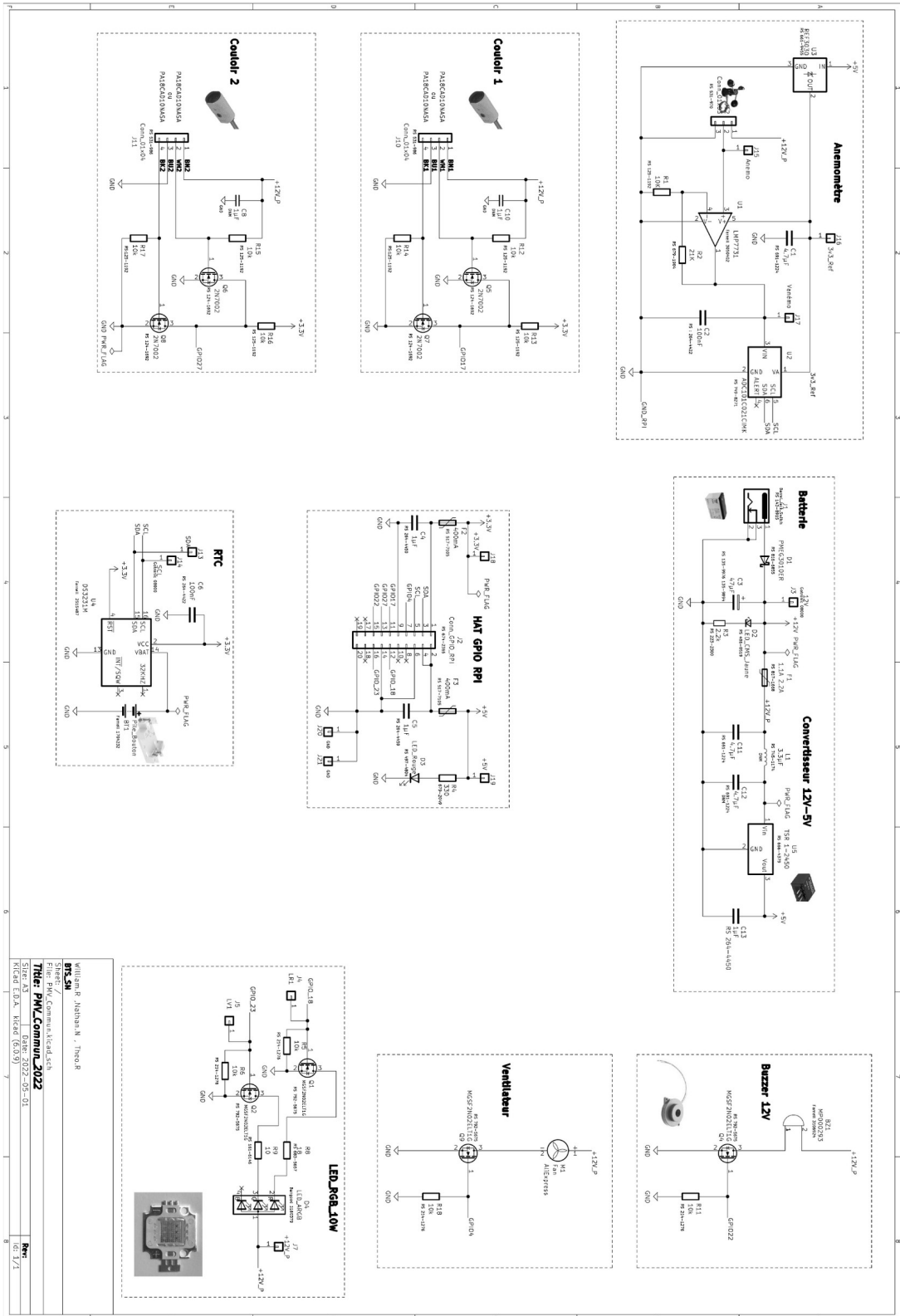


Diagramme Actuel :



Avant de commencer mon schéma KiCad et le Routage, j'ai d'abord dû tester tous les composants du projet PMV de l'année 2022 qui sont :



WILLIAM R. JOSHUA M. THEA R
INTSA
 File: PMV_Communicat.kicad_sch
Titre: PMV_Communicat_2022
 Sizer: AS_2022-05-01
 Riche: EDA_Kicad (5.0.0)
 Rev: 1/1

Composant	Visuel	Documentation																																		
<p>La Girouette</p>	 <p>360 degrees 70cm cable 0-5V output carbon material</p>	<table border="1"> <thead> <tr> <th>Module</th> <th>Wind speed sensor</th> <th>Module</th> <th>Wind direction sensor</th> </tr> </thead> <tbody> <tr> <td>power supply</td> <td>10 ~ 30V DC</td> <td>power supply</td> <td>10 ~ 30V DC</td> </tr> <tr> <td>Measuring range</td> <td>0 ~ 60m / s</td> <td>Measuring range</td> <td>8 indicate the direction</td> </tr> <tr> <td>Accuracy</td> <td>± (0.2+0.03V) m/s</td> <td>Working Temp</td> <td>-20°C~+60°C, 0%RH~80%RH</td> </tr> <tr> <td>Resolution</td> <td>0.1 m / s</td> <td rowspan="4">Parameters</td> <td rowspan="4">485 (modbus) protocol Baud Rate: 2400, 4800 (default), 9600 Data bit length: 8 Parity: none Stop bit length: 1 The default ModBus Address: 1 Support Function code: 03</td> </tr> <tr> <td>power consumption</td> <td>0.4W</td> </tr> <tr> <td>Response Time</td> <td>≤ 0.5s</td> </tr> <tr> <td>Start wind speed</td> <td>≤ 0.2m / s</td> </tr> <tr> <td>Working Temp</td> <td>-20°C~+60°C , 0%RH~80%RH</td> <td>response speed</td> <td>≤ 0.5s</td> </tr> <tr> <td>Output signal</td> <td>RS485/Analog/Pulse</td> <td>Output signal</td> <td>RS485/Analog</td> </tr> </tbody> </table>	Module	Wind speed sensor	Module	Wind direction sensor	power supply	10 ~ 30V DC	power supply	10 ~ 30V DC	Measuring range	0 ~ 60m / s	Measuring range	8 indicate the direction	Accuracy	± (0.2+0.03V) m/s	Working Temp	-20°C~+60°C, 0%RH~80%RH	Resolution	0.1 m / s	Parameters	485 (modbus) protocol Baud Rate: 2400, 4800 (default), 9600 Data bit length: 8 Parity: none Stop bit length: 1 The default ModBus Address: 1 Support Function code: 03	power consumption	0.4W	Response Time	≤ 0.5s	Start wind speed	≤ 0.2m / s	Working Temp	-20°C~+60°C , 0%RH~80%RH	response speed	≤ 0.5s	Output signal	RS485/Analog/Pulse	Output signal	RS485/Analog
Module	Wind speed sensor	Module	Wind direction sensor																																	
power supply	10 ~ 30V DC	power supply	10 ~ 30V DC																																	
Measuring range	0 ~ 60m / s	Measuring range	8 indicate the direction																																	
Accuracy	± (0.2+0.03V) m/s	Working Temp	-20°C~+60°C, 0%RH~80%RH																																	
Resolution	0.1 m / s	Parameters	485 (modbus) protocol Baud Rate: 2400, 4800 (default), 9600 Data bit length: 8 Parity: none Stop bit length: 1 The default ModBus Address: 1 Support Function code: 03																																	
power consumption	0.4W																																			
Response Time	≤ 0.5s																																			
Start wind speed	≤ 0.2m / s																																			
Working Temp	-20°C~+60°C , 0%RH~80%RH	response speed	≤ 0.5s																																	
Output signal	RS485/Analog/Pulse	Output signal	RS485/Analog																																	
<p>L'anémomètre</p>		<p>Paramètres techniques -Précision : ±1m/s -Force du vent au démarrage : 0,2-0,4 m/s ▲ Type de sortie en tension Plage de mesure : 0~32. 4 m/s Tension d'alimentation : 7V~24 V <u>DC</u> Signal de sortie : 0,4~2V ou 0~5 V, 1~5 V Valeur de la vitesse du vent = (tension de sortie - 0,4)/1,6*32,4 ▲ Type de sortie courant Plage de mesure : 0~32,4 m/s Tension d'alimentation : 12V~24V <u>DC</u> Signal de sortie : 4~20 mA Capacité de charge : ≤200Ω Valeur de la vitesse du vent = (courant de sortie - 4)/16*32,4 ▲ Type de sortie à impulsions Plage de mesure : 0~60 m/s Signal de sortie : impulsion (chaque impulsion correspond à 0,88 m/s) Tension d'alimentation : 5V~24V <u>DC</u> ▲ Type 485 Gamme : 0~32.4 m/s Tension d'alimentation : 7V~24V <u>DC</u> Protocole de communication : <u>Modbus-RTU</u></p>																																		
<p>Le buzzer</p>		<ul style="list-style-type: none"> • type de signal: ton continu / signal pulsé • tension d'opération: 3-18Vcc • tension nom.: 12Vcc • consommation: 30mA • fréquence d'osc.: 2.9kHz • niveau sonore: 100 dB • type de connexion: fils <ul style="list-style-type: none"> ◦ câble noir: masse ◦ câble rouge: ton continu ◦ câble jaune: signal pulse • couleur boîtier: blanc • poids: 22g 																																		

Les capteurs :

PA18CAD10NASA :



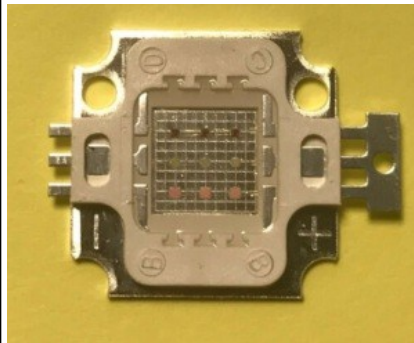
Rated operating distance (S _a)	Up to 1 m, Up to 0.8 m reference target Kodak test card R27, white, 90% reflective, 200 x 200 mm	OFF-state current (I _o)	≤ 100 µA
Axial type (A)		Voltage drop (U _d)	≤ 2.0 VDC @ 100 mA
Radial type (R)		Protection	Short-circuit, reverse polarity and transients
Blind zone	20 mm	Light source	InGaAlP, LED, 625 nm
Sensitivity control	Adjustable by potentiometer 270°	Light type	Red, modulated
Adjustable distance		Sensing angle	± 2°
Axial types	50-1000 mm	Ambient light	30.000 lux Incandescent lamp
Radial types	50-800 mm	Light spot Diameter	Ø 52 mm @ 0.5 m
Temperature drift	≤ 0.2%/°C	Operating frequency	500 Hz
Hysteresis (H) (differential travel)	≤ 20%	Response time	OFF-ON (t _{on}) ON-OFF (t _{off})
Rated operational volt. (U _e)	10 to 30 VDC (ripple included)	ON-OFF (t _{off})	≤ 1.0 ms
Ripple (U _{rip})	≤ 10%	ON-OFF (t _{off})	≤ 1.0 ms
Output current	≤ 100 mA ≤ 100 mA (max. load capacity 100 nF)	Power ON delay (t _o)	≤ 300 ms
Continuous (I _c)		Output function	Type Switching function
Short-time (I _s)		Type	NPN or PNP
No load supply current (I _o)	≤ 20 mA @ 24 VDC	Switching function	NO and NC
Minimum operational current (I _o)	0.5 mA	Indication	LED, yellow LED, green
Environment		Output ON	Signal stability and power ON
Installation category	III (IEC 60664/60664A; 60947-1)	Housing material	IEC protection class III
Pollution degree	3 (IEC 60664/60664A; 60947-1)	Body	ABS, grey
Degree of protection	IP 67, IP 69K*	Front material	PMMA, red
Ambient temperature		Locknuts	PBTB, black
Operating	-25° to +60°C (-13° to +140°F)	Mounting bracket	PPA, black
Storage	-40° to +70°C (-40° to +158°F)	Connection	
Vibration	10 to 55 Hz, 0.5 mm/7.5 g (IEC 60068-2-6)	Cable	PVC, grey, 2 m 4 x 0.25 mm ² , Ø = 4.5 mm
Shock	30 g / 11 ms, 3 pos, 3 neg per axis (IEC 60068-2-6, 60068-2-32)	Plug	M12, 4-pin (CONNM14NF-series)
Rated insulation voltage	500 VAC (rms)	Weight	With cable: 40 g With plug: 10 g

PA18CAD10NAM1SA :



Rated operating distance (S _a)	Up to 1 m, Up to 0.8 m reference target Kodak test card R27, white, 90% reflective, 200 x 200 mm	OFF-state current (I _o)	≤ 100 µA
Axial type (A)		Voltage drop (U _d)	≤ 2.0 VDC @ 100 mA
Radial type (R)		Protection	Short-circuit, reverse polarity and transients
Blind zone	20 mm	Light source	InGaAlP, LED, 625 nm
Sensitivity control	Adjustable by potentiometer 270°	Light type	Red, modulated
Adjustable distance		Sensing angle	± 2°
Axial types	50-1000 mm	Ambient light	30.000 lux Incandescent lamp
Radial types	50-800 mm	Light spot Diameter	Ø 52 mm @ 0.5 m
Temperature drift	≤ 0.2%/°C	Operating frequency	500 Hz
Hysteresis (H) (differential travel)	≤ 20%	Response time	OFF-ON (t _{on}) ON-OFF (t _{off})
Rated operational volt. (U _e)	10 to 30 VDC (ripple included)	ON-OFF (t _{off})	≤ 1.0 ms
Ripple (U _{rip})	≤ 10%	ON-OFF (t _{off})	≤ 1.0 ms
Output current	≤ 100 mA ≤ 100 mA (max. load capacity 100 nF)	Power ON delay (t _o)	≤ 300 ms
Continuous (I _c)		Output function	Type Switching function
Short-time (I _s)		Type	NPN or PNP
No load supply current (I _o)	≤ 20 mA @ 24 VDC	Switching function	NO and NC
Minimum operational current (I _o)	0.5 mA	Indication	LED, yellow LED, green
Environment		Output ON	Signal stability and power ON
Installation category	III (IEC 60664/60664A; 60947-1)	Housing material	IEC protection class III
Pollution degree	3 (IEC 60664/60664A; 60947-1)	Body	ABS, grey
Degree of protection	IP 67, IP 69K*	Front material	PMMA, red
Ambient temperature		Locknuts	PBTB, black
Operating	-25° to +60°C (-13° to +140°F)	Mounting bracket	PPA, black
Storage	-40° to +70°C (-40° to +158°F)	Connection	
Vibration	10 to 55 Hz, 0.5 mm/7.5 g (IEC 60068-2-6)	Cable	PVC, grey, 2 m 4 x 0.25 mm ² , Ø = 4.5 mm
Shock	30 g / 11 ms, 3 pos, 3 neg per axis (IEC 60068-2-6, 60068-2-32)	Plug	M12, 4-pin (CONNM14NF-series)
Rated insulation voltage	500 VAC (rms)	Weight	With cable: 40 g With plug: 10 g

La LED



- bleu 460NM, rouge 620NM, vert 530NM
- Intensité lumineuse : rouge : 100lm, vert : 270lm, bleu : 100lm
- Tension rouge : 6-6.4V, vert : 9-12V, bleu 9-12V
- Courant : 300mA
- 10Watt de puissance totale avec contrôle de toutes les couleurs

Le Ventilateur



- Tension nominale: cc 12V
- Note actuelle: 0.07A
- Consommation électrique: 0.75 W
- Révolution: 5200 ± 5% tr/min
- Bruit: 23DBA
- Roulement: roulement de douille
- Durée de vie prévue: 40000 heures
- Connexion: le rouge est positif +, le noir est la cathode-, le bleu est le signal RD
- Interface: PH2.0 3pin

Pour vérifier le bon fonctionnement de chaque composants que j'ai récupérés, j'ai dû écrire plusieurs codes :

En tout premier j'ai commencé par coder le pilotage du ventilateur et de la led

Voici le code pour le ventilateur :

```
1  #include <stdio.h>
2  #include <bcm2835.h>
3
4  #define ventilateur RPI_BPLUS_GPIO_J8_07 // Numéro de la broche sur le GPIO
5
6  //using namespace std;
7
8  // g++ Lecture_BP_GPIO04ventilateur.cpp -l bcm2835 -o Lecture_BP_GPIO04ventil
9
10 void setup()
11 {
12     if (!bcm2835_init())
13     {
14         printf("bcm2835_init failed. Are you running as root??\n");
15     }
16 }
17
18 int main(void)
19 {
20     setup();
21     bcm2835_gpio_fsel(ventilateur, BCM2835_GPIO_FSEL_OUTP); // Broche Ventilateur
22     printf("Mise en service ventilateur du GPIO04 \n");
23     while(1)
24     {
25         bcm2835_gpio_write(ventilateur, LOW); //ventilateur à l'arrêt
26         bcm2835_delay(2000);
27         bcm2835_gpio_write(ventilateur, HIGH); //ventilateur en marche
28         bcm2835_delay(2000);
29     }
30     bcm2835_close();
31     return 0;
32 }
```

Le code ci dessus permet de mettre en service le ventilateur, puis de l'éteindre avec un délai de 2 secondes entre chaque arrêt et démarrage.

Puis j'ai dû vérifier le bon fonctionnement de la LED pour voir si la couleur demandé était bien allumé. Pour se faire, j'ai dû effectuer en premier deux programmes différents, un pour afficher la couleur Rouge et un autre programme pour afficher la couleur Verte :

Voici le programme pour la LED Rouge :

```
1 #include <stdio.h>
2 #include <bcm2835.h>
3
4 #define LEDRouge RPI_BPLUS_GPIO_J8_12 // Numéro de la broche sur le GPIO12 LED Rouge
5
6 //using namespace std;
7
8 // g++ PMV_Commande_LED_Verte.cpp -l bcm2835 -o PMV_Commande_LED_Rouge // Ligne de compilation
9
10 void setup()
11 {
12     if (!bcm2835_init())
13     {
14         printf("bcm2835_init failed. Are you running as root??\n");
15     }
16 }
17
18 int main(void)
19 {
20     setup();
21     bcm2835_gpio_fsel(LEDRouge, BCM2835_GPIO_FSEL_OUTP); // Broche Ventilateur (GPIO04) en sortie
22     printf("Mise en service ventilateur du GPIO04 \n");
23     while(1)
24     {
25         bcm2835_gpio_write(LEDRouge, LOW); //ventilateur à l'arrêt
26         bcm2835_delay(2000);
27         bcm2835_gpio_write(LEDRouge, HIGH); //ventilateur en marche
28         bcm2835_delay(2000);
29     }
30     bcm2835_close();
31     return 0;
32 }
33
34
```

Programme pour la LED Verte :

```
1 #include <stdio.h>
2 #include <bcm2835.h>
3
4 #define LEDVerte RPI_BPLUS_GPIO_J8_16 // Numéro de la broche sur le GPIO12 LED Verte
5
6 //using namespace std;
7
8 // g++ PMV_Commande_LED_Verte.cpp -l bcm2835 -o PMV_Commande_LED_Verte // Ligne de compilation
9
10 void setup()
11 {
12     if (!bcm2835_init())
13     {
14         printf("bcm2835_init failed. Are you running as root??\n");
15     }
16 }
17
18 int main(void)
19 {
20     setup();
21     bcm2835_gpio_fsel(LEDVerte, BCM2835_GPIO_FSEL_OUTP); // Broche Ventilateur (GPIO04) en sortie
22     printf("Mise en service ventilateur du GPIO04 \n");
23     while(1)
24     {
25         bcm2835_gpio_write(LEDVerte, LOW); //ventilateur à l'arrêt
26         bcm2835_delay(2000);
27         bcm2835_gpio_write(LEDVerte, HIGH); //ventilateur en marche
28         bcm2835_delay(2000);
29     }
30     bcm2835_close();
31     return 0;
32 }
33
34
```

J'ai ensuite essayer les deux programmes , le test se concluant par le bon fonctionnement des deux programmes.

Voici le programme pour la LED_VERT_ROUGE :

```
1  #include <stdio.h>
2  #include <bcm2835.h>
3
4  #define LEDRouge RPI_BPLUS_GPIO_J8_12 // Numéro de la broche sur le GPIO12 LED Rouge
5  #define LEDVerte RPI_BPLUS_GPIO_J8_16 // Numéro de la broche sur le GPIO16 LED Vert
6
7  //using namespace std;
8
9  // g++ PMV_LEDrougeVert.cpp -l bcm2835 -o PMV_LEDrougeVert // Ligne de compilation
10
11 void setup()
12 {
13     if (!bcm2835_init())
14     {
15         printf("bcm2835_init failed. Are you running as root??\n");
16     }
17 }
18
19 int main(void)
20 {
21     setup();
22     bcm2835_gpio_fsel(LEDRouge, BCM2835_GPIO_FSEL_OUTP); // LED Rouge (GPIO18) en sortie
23     bcm2835_gpio_fsel(LEDVerte, BCM2835_GPIO_FSEL_OUTP); // LED Verte (GPIO23) en sortie
24     printf("Mise en service des LED Rouge et Verte \n");
25     while(1)
26     {
27         bcm2835_gpio_write(LEDRouge, LOW); //ventilateur à l'arrêt
28         bcm2835_gpio_write(LEDVerte, LOW); //ventilateur à l'arrêt
29         bcm2835_delay(2000);
30         bcm2835_gpio_write(LEDRouge, HIGH); //ventilateur en marche
31         bcm2835_delay(2000);
32         bcm2835_gpio_write(LEDRouge, LOW); //ventilateur à l'arrêt
33         bcm2835_delay(2000);
34         bcm2835_gpio_write(LEDVerte, HIGH); //ventilateur en marche
35         bcm2835_delay(2000);
36         bcm2835_gpio_write(LEDVerte, LOW); //ventilateur à l'arrêt
37         bcm2835_delay(2000);
38         bcm2835_gpio_write(LEDVerte, HIGH); //ventilateur en marche
39         bcm2835_gpio_write(LEDRouge, HIGH); //ventilateur en marche
40         bcm2835_delay(2000);
41         bcm2835_gpio_write(LEDVerte, LOW); //ventilateur à l'arrêt
42         bcm2835_gpio_write(LEDRouge, LOW); //ventilateur à l'arrêt
43         bcm2835_delay(2000);
44     }
45     bcm2835_close(); ;
46     return 0;
47 }
48
49
50
```

Une fois que le programme a été complété, je l'ai essayé avec le composant ainsi une fois le programme lancé cela me montrera un bon fonctionnement avec la LED qui s'allume Rouge puis en VERT avec une délai de 2 seconde.

Pour finir le test de tout les composants, il ne me restait plus qu'à programmer le pilotage du buzzer pour vérifier son bon fonctionnement :

Voici le programme du buzzer :

```
1  #include <stdio.h>
2  #include <bcm2835.h>
3
4  #define Buzzer RPI_BPLUS_GPIO_J8_15 // Numéro de la broche sur le GPIO22 Buzzer
5
6  //using namespace std;
7
8  // g++ PMV_Buzzer.cpp -l bcm2835 -o PMV_Buzzer // Ligne de compilation
9
10 void setup()
11 {
12     if (!bcm2835_init())
13     {
14         printf("bcm2835_init failed. Are you running as root??\n");
15     }
16 }
17
18 int main(void)
19 {
20     setup();
21     bcm2835_gpio_fsel(Buzzer, BCM2835_GPIO_FSEL_OUTP); // Broche Buzzer (GPIO15) en sortie
22     printf("Mise en service ventilateur du GPIO \n");
23     while(1)
24     {
25         bcm2835_gpio_write(Buzzer, LOW); //ventilateur à l'arrêt
26         bcm2835_delay(2000);
27         bcm2835_gpio_write(Buzzer, HIGH); //ventilateur en marche
28         bcm2835_delay(2000);
29     }
30     bcm2835_close();
31     return 0;
32 }
33
34
```

Une fois le programme finalise, je l'ai testé, le buzzer fonctionner et bipier correctement toutes les 2 secondes.

Après avoir fini la programmation du buzzer, je me suis occupé de l'Anémomètre. Tout d'abord, je l'ai branché correctement, puis je l'ai testé avec le programme qui m'a été fourni par Mr. Hortolland :

Voici le programme pour l'Anémomètre

```
1 // Mesure de tension avec l'ADC101 et une tension de référence de 3,3V (REF3231).
2 // Puis calcul et affichage de la vitesse du vent en km/h et en m/sec
3 // La calcul se fait à partir de 5 échantillon successifs.
4 // Les calculs se font en float, puis avant l'affiche passage en int
5 // Fichier Vitesse_vent_03.cpp
6 #include <iostream>
7 #include <bcm2835.h>
8 using namespace std;
9
10 #define clk_div BCM2835_I2C_CLOCK_DIVIDER_2500
11
12 // g++ Vitesse_vent_03.cpp -l bcm2835 -o Vitesse_vent_03
13
14 void init_I2c_bcm2835()
15 {
16     if (!bcm2835_init())
17     {
18         printf("bcm2835_init failed. Are you running as root??\n");
19     }
20     if (!bcm2835_i2c_begin())
21     {
22         printf("bcm2835_i2c_begin failed. Are you running as root??\n");
23     }
24     bcm2835_i2c_setClockDivider(clk_div); // Division de l'horloge Rpi pour obtenir une vitesse de 100KHz
25 }
26
27 int main(void)
28 {
29     double mesure;
30     unsigned int slave_address=0x54; // Adresse du DS1621
31     char i2cOut[3]; // Tableau des données I2C à sortir
32     char i2cIn[2]; // Tableau des données I2C entrées
33     int resultat_ADC =0;
34     float tension=0;
35     float vitesseKMH=0;
36     float vitesseMS=0;
37     int vitesse_kmh=0;
38     int vitesse_ms=0;
39
40
41     init_I2c_bcm2835();
42     bcm2835_i2c_setSlaveAddress(slave_address); // Affectation de l'adresse de l'esclave
43
44     i2cOut[0] = 0x00;
45     bcm2835_i2c_write(i2cOut, 1); // Prépositionne la valeur du pointeur 0x00 Pour être en mode "Conversion Result)
46     delay(1000); // 1 seconde avant première lecture
47
48     while(1)
49     {
50         resultat_ADC =0;
51         tension = 0;
52         for (int i=0; i<5; i++)
53         {
54             bcm2835_i2c_read(i2cIn,2);
55             resultat_ADC = ((i2cIn[1]>>2)& 0b00111111) + (64* (i2cIn[0] & 0b00011111));
56             tension += resultat_ADC*3.3/1023;
57             delay(10);
58         }
59         tension/=5.0;
60         vitesseKMH = (tension*23.58) -29.24;
61         if (vitesseKMH<0) vitesseKMH=0.0;
62         vitesseMS = (vitesseKMH*1000)/3600;
63         if (vitesseMS<0) vitesseMS=0.0;
64         vitesse_kmh = (int) vitesseKMH;
65         vitesse_ms = (int) vitesseMS;
66         cout<<"Vitesse du vent = "<<vitesse_kmh<<" km/h = "<<vitesse_ms<<" m/sec "<<endl;
67         delay(1000); // 1 seconde entre chaque conversion
68     }
69
70     bcm2835_close();
71     return 0;
72 }
73
```


Ensuite, j'ai du créer un programme qui lorsqu'une certaine vitesse est atteinte de mettre en service le buzzer, pour effectuer cette tâche, j'ai rajouté cette partie du code dans le programme de l'anémomètre

```
    if (vitesse_kmh>11)
    {
        bcm2835_gpio_write(Buzzer, LOW);
        //bcm2835_delay(2000);
        bcm2835_gpio_write(Buzzer, HIGH); //ventilateur en marche
        bcm2835_delay(2000);
        bcm2835_gpio_write(Buzzer, LOW);
    }
    // 1 seconde entre chaque conversion
}

bcm2835_close();
return 0;
}
```

La raison pour laquelle la ligne « ***bcm2835_gpio_write(Buzzer, Low);*** » est qu'elle permet d'éteindre le buzzer, afin d'éviter qu'il reste allumer, car oui sans cette ligne le buzzer ne s'arrêterait pas.

J'ai effectuer quelques tests pour voir si cela fonctionnait et il s'est avéré très concluant ainsi qu'un bon fonctionnement du buzzer lorsqu'une certaine vitesse est atteinte.

Une fois que j'ai vérifier ces composants, il ne me restait plus que les anciens capteurs à vérifier, j'ai donc utilisé le programme de l'année 2022.

Voici le programme des deux capteurs :
/Capteurs couloir 1

```
1  #include <iostream>
2  #include <bcm2835.h>
3  using namespace std;
4
5  #define Couloir_2 RPI_BPLUS_GPIO_J8_11 // Numéro de la broche sur le GPIO
6
7  //using namespace std;
8
9  // g++ Lecture_couloir_1.cpp -l bcm2835 -o Lecture_couloir_1 // Ligne de compilation
10
11 void setup()
12 {
13     if (!bcm2835_init())
14     {
15         printf("bcm2835_init failed. Are you running as root??\n");
16     }
17 }
18
19 int main(void)
20 {
21     setup();
22     bcm2835_gpio_fsel(Couloir_2, BCM2835_GPIO_FSEL_INPT); // Broche couloir 2 en sortie
23     cout<<"Lecture couloir 1 "<<endl;
24     while(1)
25     {
26         if (bcm2835_gpio_lev(Couloir_2)) cout <<"Pas de détection"<< endl;
27         else cout<<"Détection de passage"<<endl; //
28             bcm2835_delay(500);
29     }
30     bcm2835_close();
31     return 0;
32 }
33
```



/Pour le capteur couloir 2

```
1 #include <iostream>
2 #include <bcm2835.h>
3 using namespace std;
4
5 #define Couloir_2 RPI_BPLUS_GPIO_J8_13 // Numéro de la broche sur le GPIO
6
7 //using namespace std;
8
9 // g++ Lecture_couloir_2.cpp -l bcm2835 -o Lecture_couloir_2 // Ligne de compilation
10
11 void setup()
12 {
13     if (!bcm2835_init())
14     {
15         printf("bcm2835_init failed. Are you running as root??\n");
16     }
17 }
18
19 int main(void)
20 {
21     setup();
22     bcm2835_gpio_fsel(Couloir_2, BCM2835_GPIO_FSEL_INPT); // Broche couloir 2 en sortie
23     cout<<"Lecture couloir 2 "<<endl;
24     while(1)
25     {
26         if (bcm2835_gpio_lev(Couloir_2)) cout <<"Pas de détection"<< endl;
27         else cout<<"Détection de passage"<<endl; //
28         bcm2835_delay(500);
29     }
30     bcm2835_close();
31     return 0;
32 }
33
```



Tous les composants vérifiés et testés avec Simon (EC1), nous devons essayer le capteur en condition réel.

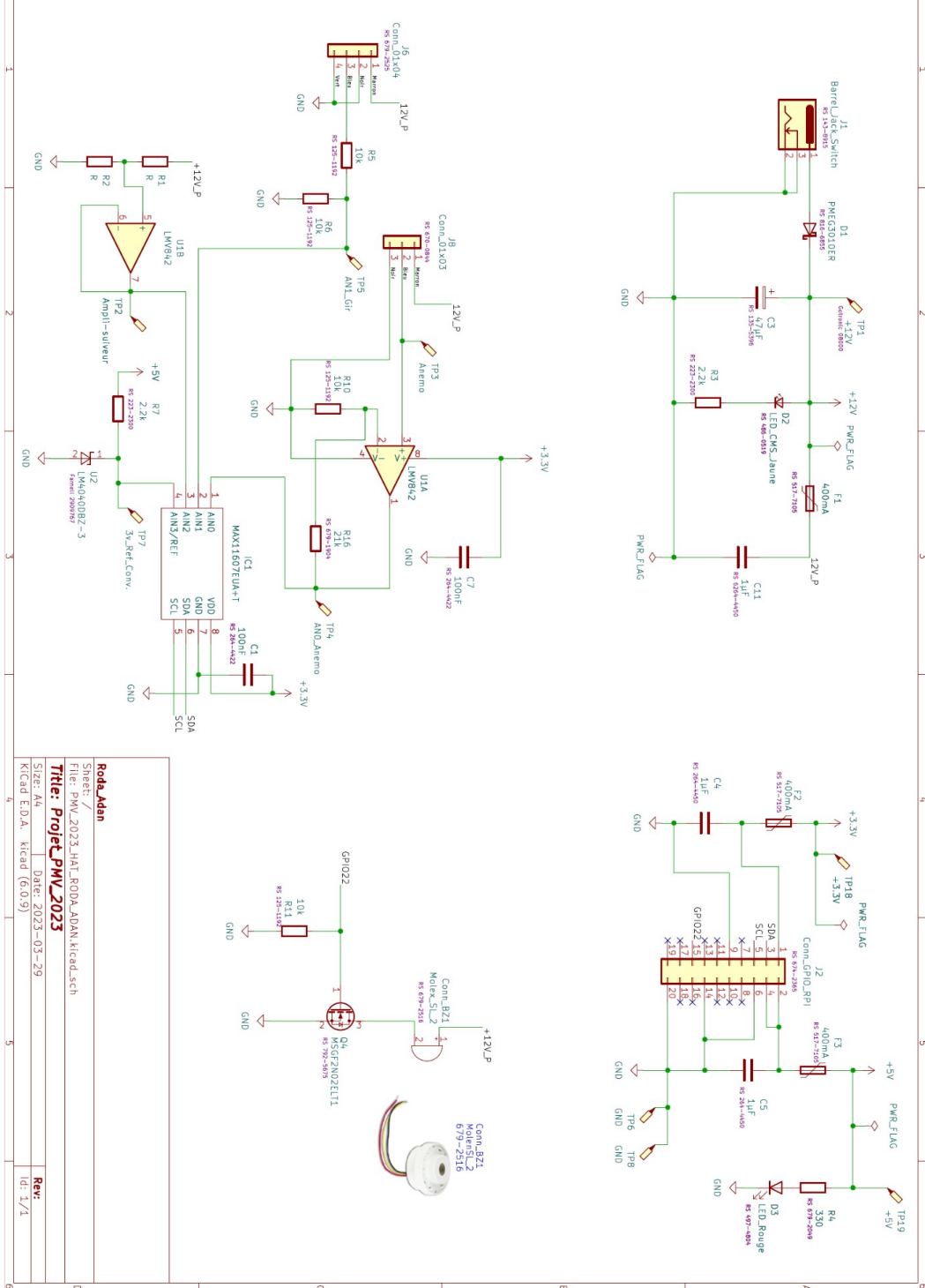
Voici le capteurs utilisé et testé en condition réel :



Complémentaires

Matière du boîtier	PBT
Matière de la lentille	PMMA
Portée maximale	3 m réflect polarisé
Type de sortie	Statique
Sortie additionnelle	Sans
Matière de l'isolant du fil	PvR
État LED	1 LED (jaune) pour état sortie
[Us] tension d'alimentation	12...24 V DC avec protection contre l'inversion de polarité
Limites de la tension d'alimentation	10...36 V CC
Pouvoir de commutation en mA	≤ 100 mA (protection contre les surcharges et court-circuits)
Fréquence de commutation	≤ 500 Hz
Tension de déchet	1,5 V (régime fermé)
Consommation électrique	35 mA (sans charge)
Retard à la disponibilité	< 15 ms
Retard réponse	< 1 ms
Retard récupération	< 1 ms
Réglage	Sans réglage sensibilité
Diamètre	18 mm
Longueur	62 mm
Poids	0.04 kg

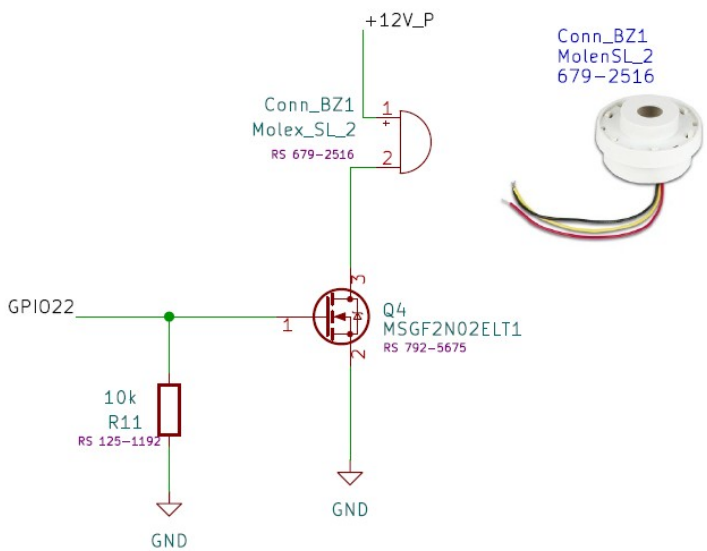
Une fois que j'ai fini d'essayer tout les composants j'ai donc commencé à faire le projet sur kicad, voici le schéma final de la carte :



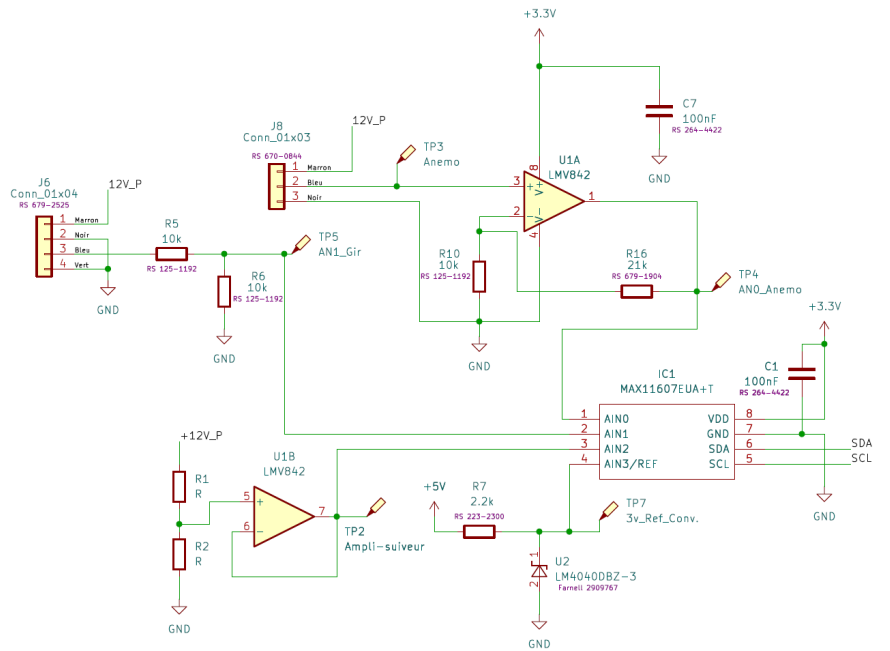
Roda Adan
 Sheet: /
 File: PMV_2023_HAT_RODA_ADAN_kicad.sch
Title: Projet PMV 2023
 Size: A4
 Date: 2023-03-29
 Rev: /
 Kicad E.D.A. kicad (6.0.5)

Le schéma KICAD ci-dessus est divisé en quatre parties :

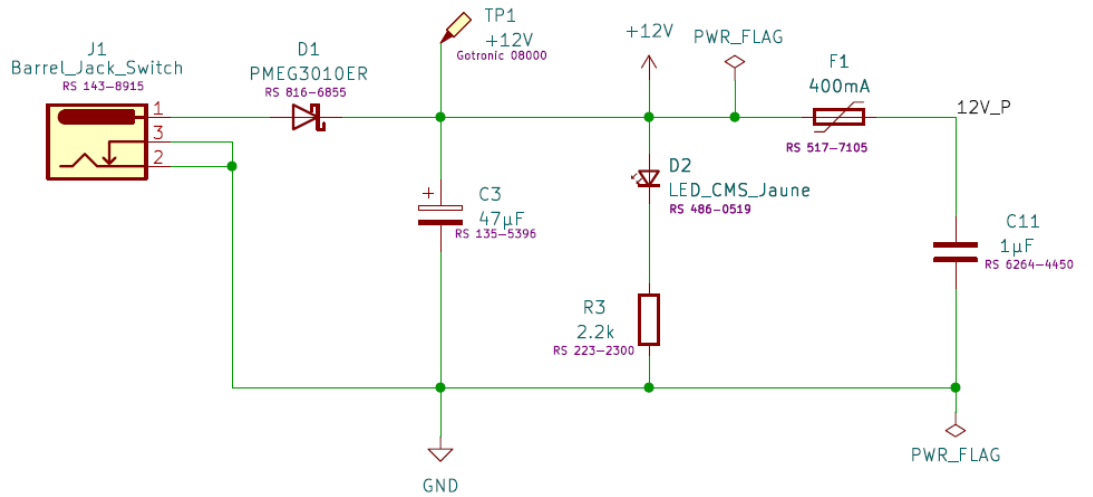
/Le buzzer :



/L'anémomètre et la girouette :



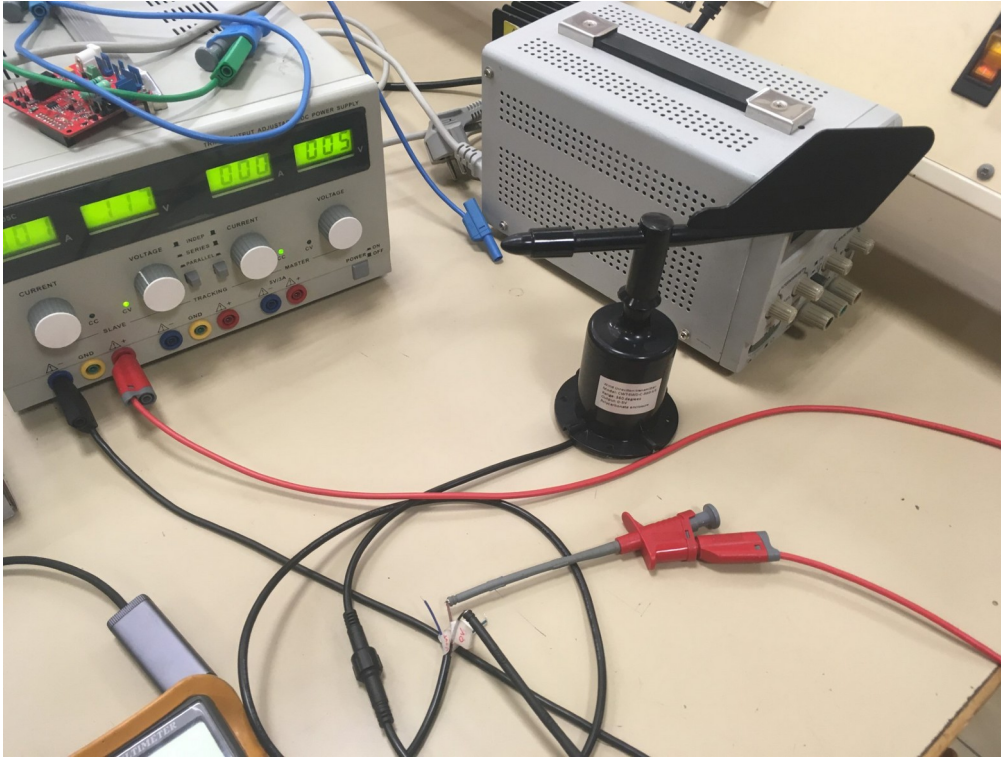
/La Batterie :



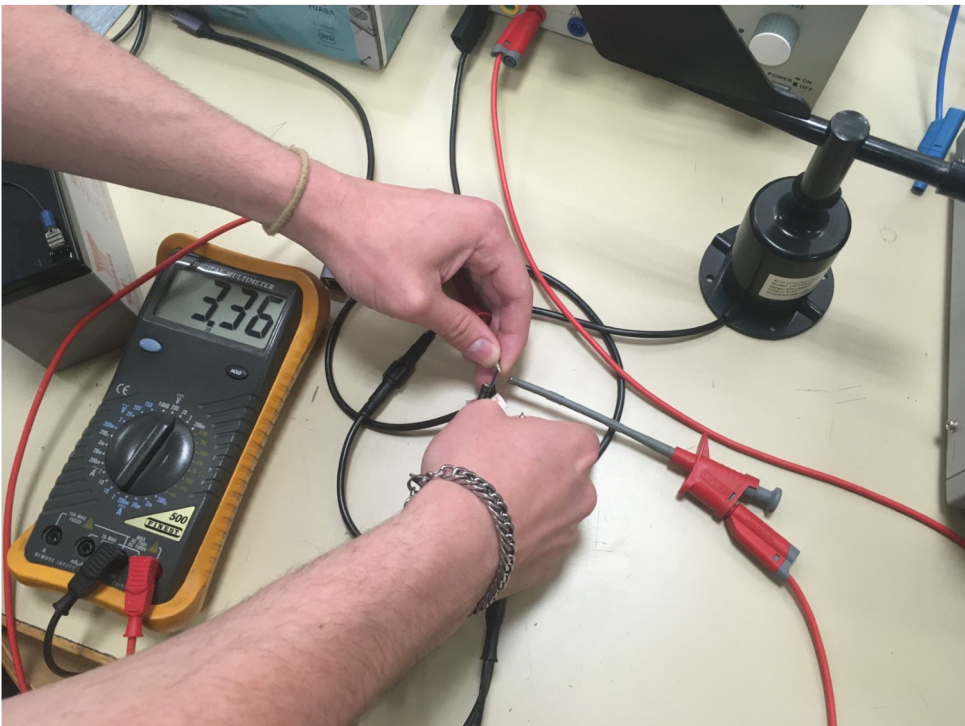
Partie Physique :

Pour voir si la girouette était en état de marche je l'ai essayer sans le brancher à la carte raspberry :

Voici le montage



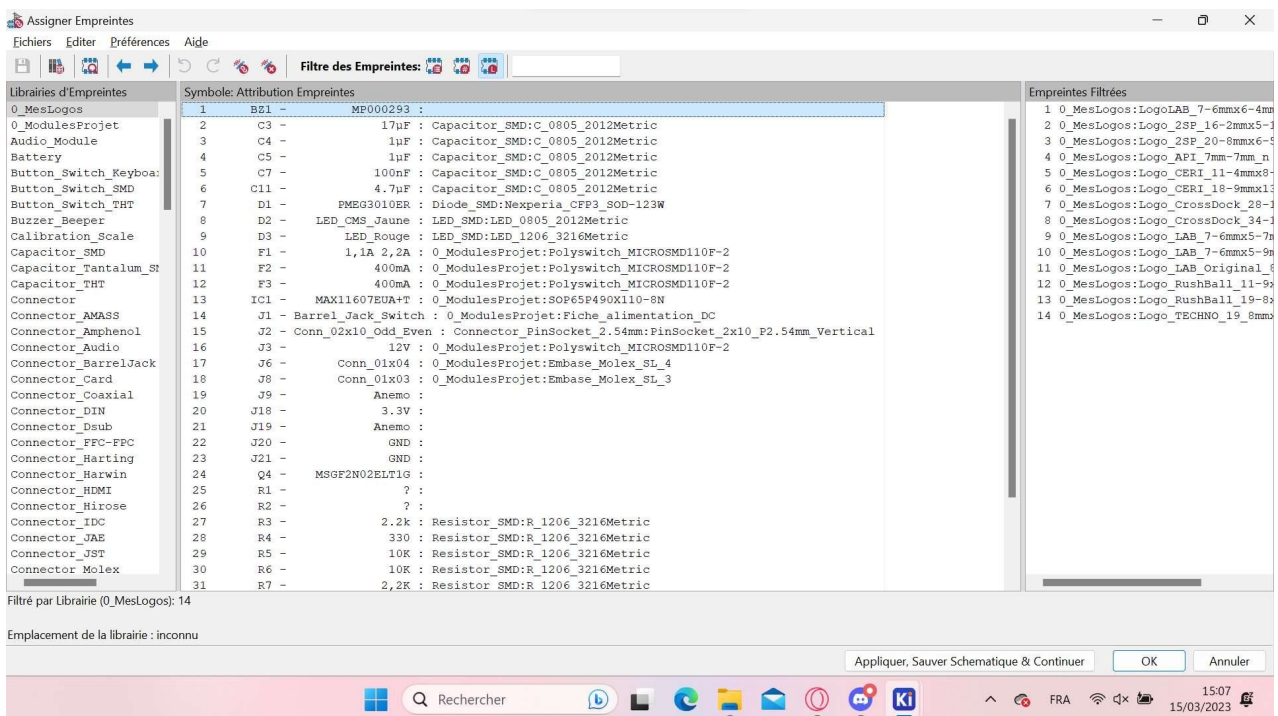
/Le test de la girouette



On peut voir que la girouette est en état de marche, elle délivre bien une tension différente à chaque fois qu'elle point une direction

Les Empreintes de composants :

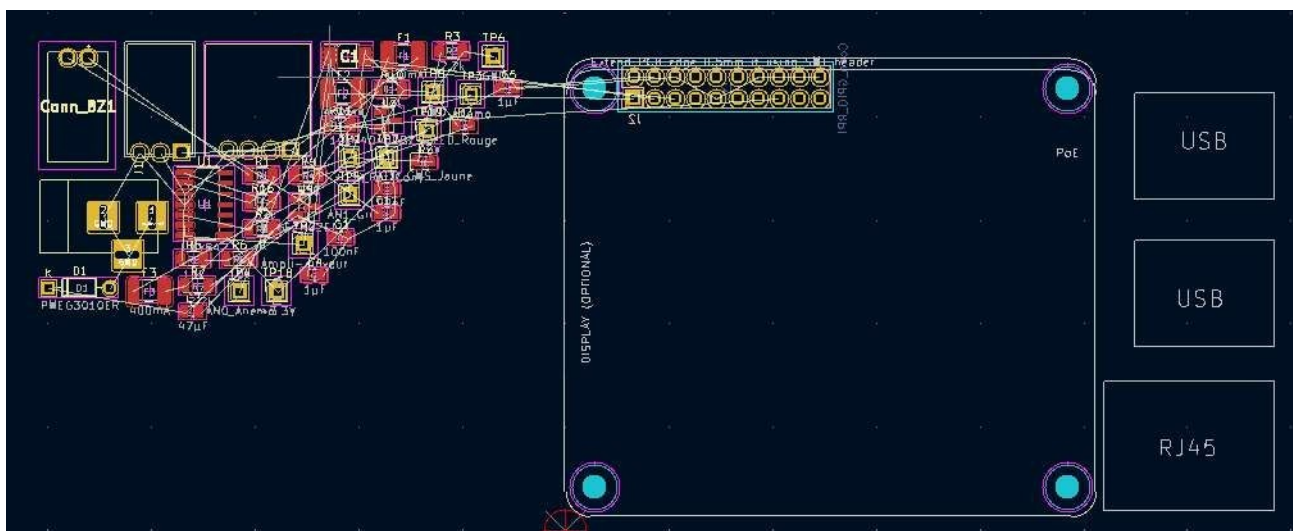
Une fois que le schéma Kicad fut terminé et envoyé pour une vérification final, j'ai ensuite dû appliquer les empreintes de composants, qui m'aideront sur le routage qui sera effectué.



Les empreintes des composants électroniques : Une fois que les empreintes ont été appliqués et que le professeur a pu vérifier les empreintes, j'ai donc enfin commencé le routage de ma carte.

Quand j'ai ouvert le fichier routage pour la première fois, j'ai donc vérifié chacun des composants pour savoir s'ils étaient bien présents.

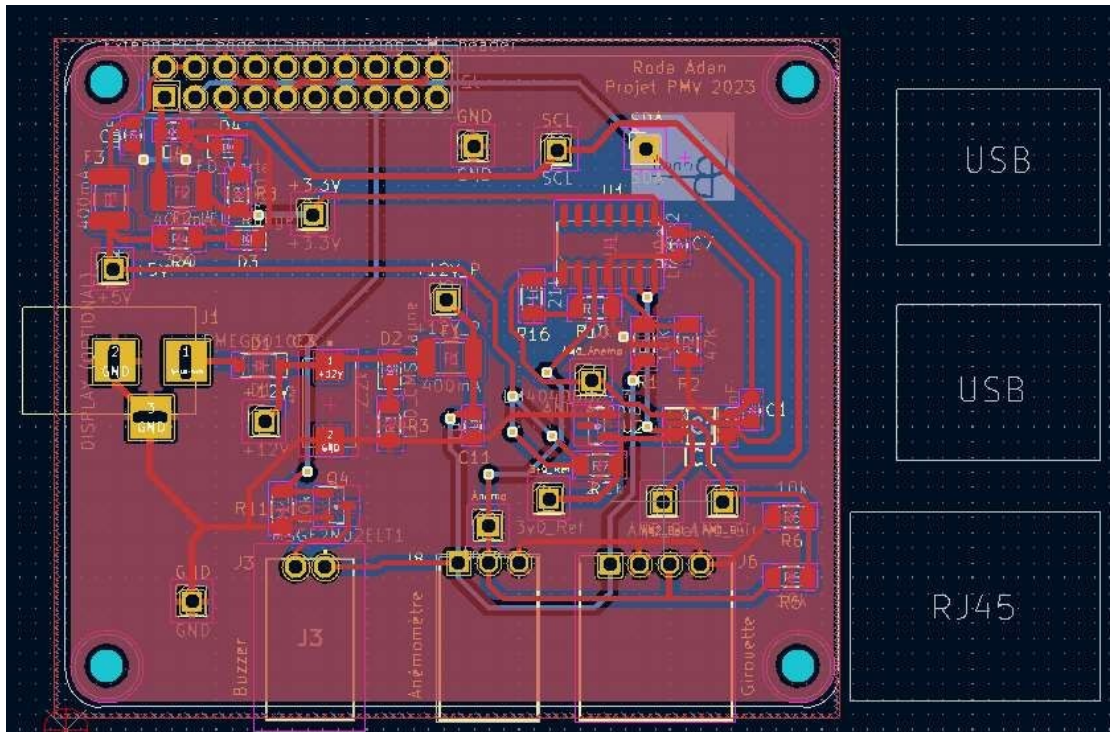
Première ouverture du fichier routage :



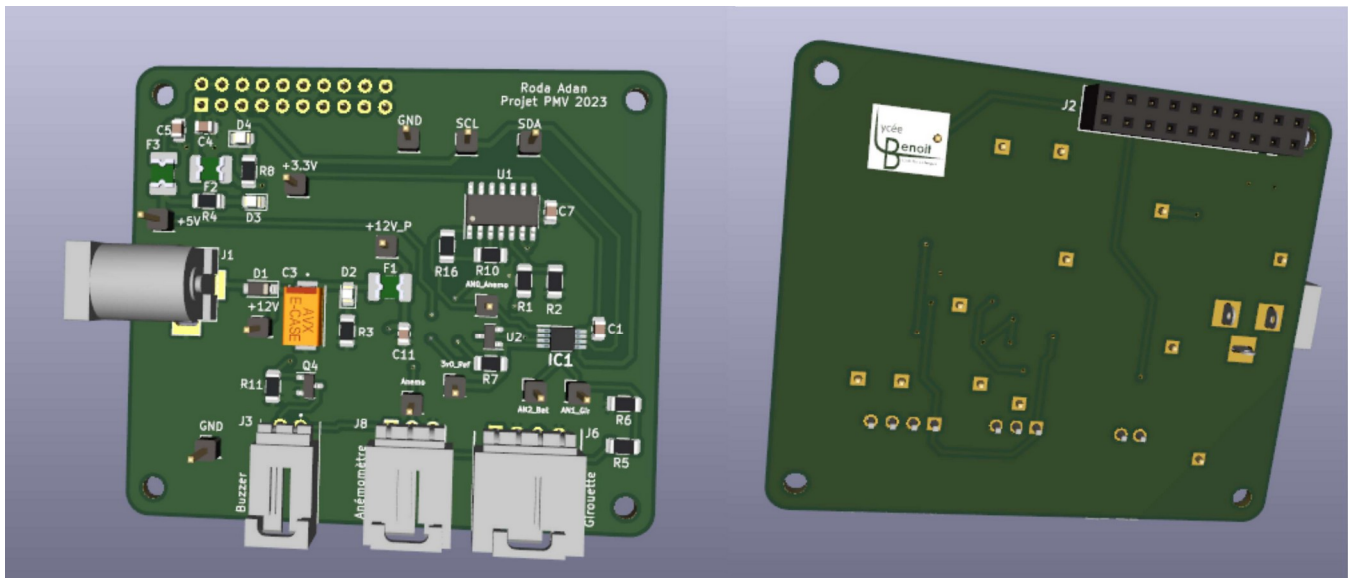
Puis après quelque retouches pour bien placer les composants et éviter des erreurs et vérification du professeur.

55.

Voici la version finale du routage :



Voici un visuelle 3D :



Conclusion :

Pour conclure, je dirais que j'ai bien avancé sur le projet et il va falloir attendre l'arrivée des cartes pour pouvoir

effectuer des tests. Il ne me reste que l'interface des capteurs à faire et les cartes à souder quand elles arriveront.

56.