Académie Aix-Marseille
Lycée Alphonse Benoît
SESSION 2022

RAPPORT DE PROJET
REVUE 3

PRISE DE MESURE VITESSE

SOLER BOTHUAN JULLIAN
ROBILLARD NOËL RICHARD

SOMMAIRE

PARTIE COMMUNE

Contexte du projet

Parmi les activités développées en cours d'Éducation Physique et Sportive, il y a les épreuves de course sur une distance de 50m et de 4x50m (relais à 4 personnes).

Pour la course de 50m, 2 coureurs s'affrontent, chacun dans leur couloir autour de la piste, avec un couloir libre entre eux. La difficulté pour l'enseignant est d'effectuer une mesure de temps la plus juste possible pour les deux coureurs.

D'autre part, le vent peut être de la partie ce jour là et modifier les performances habituelles des coureurs. De plus, avoir des informations complémentaires comme sa vitesse moyenne par exemple permet d'éveiller l'esprit de compétition et ainsi permettre à des étudiants de progresser mais aussi de passer du temps agréable en travaillant.

Enfin, cela permet un suivi rigoureux sur le long terme à l'aide de données fiables et permet ainsi d'optimiser et d'accélérer le travail effectué par le professeur.

Les enseignant d'EPS du lycée Alphonse Benoit, représentés par M. Frédéric GUELLEC, nous soumettent alors la conception d'un système de prise de mesure de vitesse.

Présentation des rôles de chaque étudiants

Partie Informatique et Réseaux

Bothuan Erwan **IR 1**	*Liste des tâches assurées par l'étudiant* **Développement de l'application mobile permettant :** • L'authentification • La gestion de la session • La gestion de la course et des coureurs • La communication pour sauver en base de données les informations et récupérer les temps de course	**Installation :** EDI Android Studio **Mise en œuvre :** Web services, Base de données, Kotlin **Configuration :** WIFI, MySQL **Réalisation :** Application Android en Kotlin **Documentation :** Mise en forme du cahier de recette du système complet
Soler Julien **IR 2**	*Liste des tâches assurées par l'étudiant* **Développement partiel de l'application Raspberry permettant :** • La gestion de la session • La gestion de la course et des coureurs • La gestion de l'écran • La gestion des barrières d'arrivée	**Installation :** EDI Qt-Creator, Qt5 **Mise en œuvre :** Programmation C++ Qt **Configuration :** EDI, GPIO **Réalisation :** Logiciel partiel **Documentation :** Manuel d'installation
Jullian Alexandre **IR 3**	*Liste des tâches assurées par l'étudiant* **Développement partiel de l'application Raspberry permettant :** • La lecture de l'anémomètre • La gestion du feu de signalisation • L'accès à la BDD, web-services	**Installation :** EDI Qt-Creator, Qt5 **Mise en œuvre :** Web Services, Base de données **Configuration :** WIFI, MySQL, GPIO **Réalisation :** Classes logicielles **Documentation :** Manuel de pannes

Partie Électronique et Communication

Noël Nathan **EC 1**	*Liste des tâches assurées par l'étudiant* **Informer les coureurs du départ de la course :** • Participer à la conception d'une carte en choisissant, testant et validant une structure générant des signaux lumineux et sonores réalisant le starter, le tout devant s'intégrer sur un hat Raspberry Pi. • Effectuer tous les tests nécessaires pour valider les structures, et les modifier si nécessaire. • Une fois les essais terminés, la fusion des schémas des 3 étudiants EC du projet aboutira à un schéma unique. • Participer à la réflexion concernant la mise en boîtier. • Effectuer la saisie du schéma et le routage de la solution proposée complète (*routage individuel*). Produire les fichier Gerber afin que la fabrication du PCB soit sous-traitée. • Câbler la carte et effectuer les essais. • Documenter la mise en service de la carte finalisée.	**Installation :** Mise en service (initialisation/configuration) d'un Raspberry Pi, librairie BCM2835, Qt-Creator, autres si nécessaires **Mise en œuvre :** Tester/valider/modifier une structure pilotée par une carte Raspberry Pi pour générer un signal ayant une forte intensité lumineuse, et un signal sonore d'un volume suffisant. L'analyse devra être menée conjointement avec les étudiants EC2 et EC3 en charge des autres composants, l'ensemble des travaux devront faire partie de la carte unique finale (hat Raspberry). Effectuer les adaptations structurelles éventuellement nécessaires lors de l'intégration structurelle. **Réalisation :** • Suite aux essais, fusionner les 3 schémas en un seul et unique • Après validation de la solution, concevoir un circuit imprimé devant être fabriqué industriellement **Documentation :** • Schéma de câblage rapide (Fritzing) • Documents de fabrication de la carte (KiCAD). Ces documents devront avoir un niveau de qualité permettant une fabrication industrielle du circuit imprimé. • Schéma structurel avec contours IBD. • Liste complète des composants avec leurs sources d'approvisionnement et leur prix • Programme en C/C++ de communication sur le bus I2C, et de détection des impacts, accompagnés des commentaires et diagrammes nécessaires à sa compréhension. • Fiche de mise en service. • Fiche de dépannage.

Richard William **EC 2**	*Liste des tâches assurées par l'étudiant* **Mesurer les temps des coureurs :** • Participer à la conception d'une carte en choisissant, testant et validant une structure détecteur de passage de la ligne d'arrivée, et permettant de mesurer la durée de la course, le tout devant s'intégrer sur un hat Raspberry Pi. • Effectuer tous les tests nécessaires pour valider les structures, et les modifier si nécessaire. • Une fois les essais terminés, la fusion des schémas des 3 étudiants EC du projet. • Participer à la réflexion concernant la mise en boîtier. • Effectuer la saisie du schéma et le routage de la solution complète (*routage individuel*). Produire les fichiers Gerber afin que la fabrication du PCB soit sous-traitée • Câbler la carte et effectuer les essais • Documenter la mise en service de la carte finalisée	**Installation :** Mise en service (initialisation/configuration) d'un Raspberry Pi, librairie BCM2835, Qt-Creator, autres si nécessaires **Mise en œuvre :** Tester/valider/modifier une structure pilotée par une carte Raspberry Pi pour générer un signal ayant une forte intensité lumineuse, et un signal sonore d'un volume suffisant. L'analyse devra être menée conjointement avec les étudiants EC2 et EC3 en charge des autres composants, l'ensemble des travaux devront faire partie de la carte unique finale (hat Raspberry). Effectuer les adaptations structurelles éventuellement nécessaires lors de l'intégration structurelle. **Réalisation :** • Suite aux essais, fusionner les 3 schémas en un seul et unique • Après validation de la solution, concevoir un circuit imprimé devant être fabriqué industriellement **Documentation :** • Schéma de câblage rapide (Fritzing) • Documents de fabrication de la carte (KiCAD). Ces documents devront avoir un niveau de qualité permettant une fabrication industrielle du circuit imprimé. • Schéma structurel avec contours IBD. • Liste complète des composants avec leurs sources d'approvisionnement et leur prix • Programme en C/C++ de communication sur le bus I2C, et de détection des impacts, accompagnés des commentaires et diagrammes nécessaires à sa compréhension. • Fiche de mise en service. • Fiche de dépannage.

Robillard Théo **EC 3**	*Liste des tâches assurées par l'étudiant* **Mesurer la vitesse du vent et alimenter :** • Participer à la conception d'une carte en choisissant, testant et validant une structure mesurant la vitesse du vent, et permettant d'alimenter le calculateur et tout ce qui s'y raccorde. Le tout devant s'intégrer sur un hat Raspberry Pi. • Effectuer tous les tests nécessaires pour valider les structures, et les modifier si nécessaire. • L'alimentation se fera par une batterie 12V. Une atténuation de cette tension pour alimenter la carte Raspberry Pi sera à prévoir. • Une fois les essais terminés, la fusion des schémas des 3 étudiants EC du projet aboutira à un schéma unique. • Participer à la réflexion concernant la mise en boîtier. • Effectuer la saisie du schéma et le routage de la solution complète (*routage individuel*). Produire les fichiers Gerber afin que la fabrication du PCB soit sous-traitée. • Câbler la carte et effectuer les essais. • Documenter la mise en service de la carte finalisée.	**Installation :** Mise en service (initialisation/configuration) d'un Raspberry Pi, librairie BCM2835, Qt-Creator, autres si nécessaires **Mise en œuvre :** Tester/valider/modifier une structure pilotée par une carte Raspberry Pi pour générer un signal ayant une forte intensité lumineuse, et un signal sonore d'un volume suffisant. L'analyse devra être menée conjointement avec les étudiants EC2 et EC3 en charge des autres composants, l'ensemble des travaux devront faire partie de la carte unique finale (hat Raspberry). Effectuer les adaptations structurelles éventuellement nécessaires lors de l'intégration structurelle. **Réalisation :** • Suite aux essais, fusionner les 3 schémas en un seul et unique • Après validation de la solution, concevoir un circuit imprimé devant être fabriqué industriellement **Documentation :** • Schéma de câblage rapide (Fritzing) • Documents de fabrication de la carte (KiCAD). Ces documents devront avoir un niveau de qualité permettant une fabrication industrielle du circuit imprimé. • Schéma structurel avec contours IBD. • Liste complète des composants avec leurs sources d'approvisionnement et leur prix • Programme en C/C++ de communication sur le bus I2C, et de détection des impacts, accompagnés des commentaires et diagrammes nécessaires à sa compréhension. • Fiche de mise en service. • Fiche de dépannage.

Annexes

Diagramme des cas d'utilisations

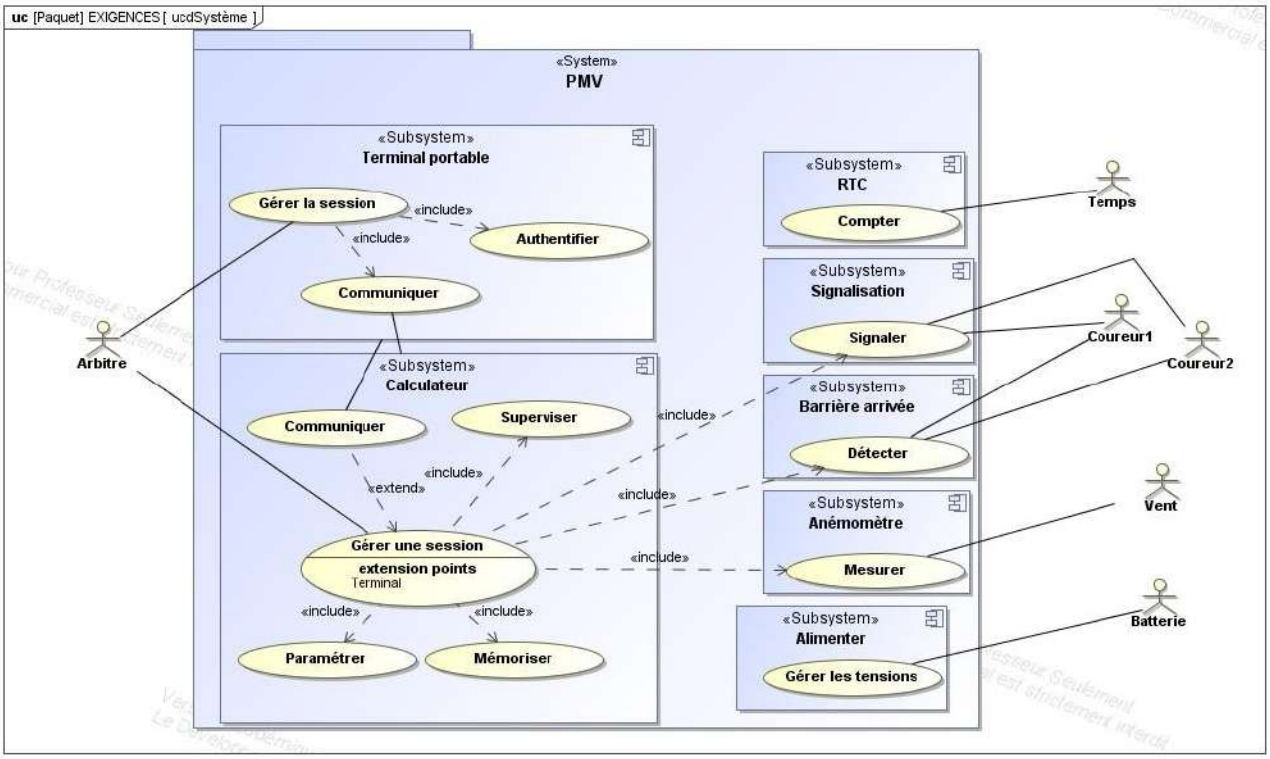
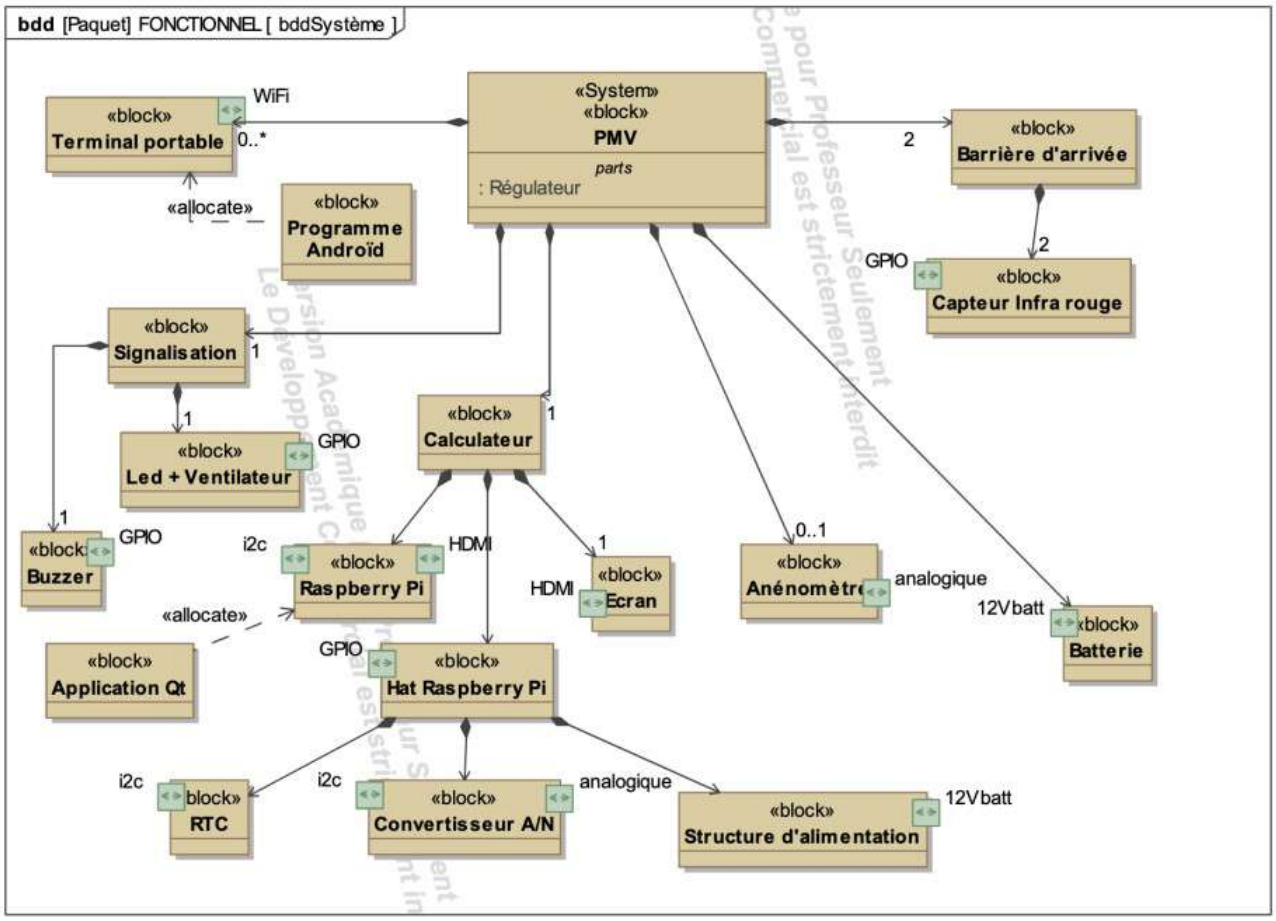


Diagramme de block



Rapport de réunion 06/01/2022

Anémomètre : analogique CAN I2C : I2C, gestion ⇒ Classe CI2C

2 GPIOs → Light + sound

GPIO barrière droite et GPIO barrière gauche (4GPIO en tout)

Buzzer (option atm)

Test LEDs RGB pour visibilité 50m

Remplacement potentiel de l'antenne + boost accès réseau à distance RPI

(Remplacer avec une antenne avec gain)

Structure métallique à déterminer

Signal alignement barrière (TUT) quand aligné

(Exemple : Fenêtre logicielle + LED VERT si OUI et OFF si NON)

Ajout de l'affichage de la vitesse

Bouton de "recalibrage" ?

Regarder ce qui existe déjà sur le PMV afin de savoir pourquoi au niveau pro, la réception des infos se fait aussi rapidement.

Rapport de réunion 03/02/2022

- Test de la distance du WIFI
- Perte de signal au bout de 40m (peut-être à cause de la sous-alimentation)
- D'autres essais seront effectués avec une nouvelle antenne

- Distance de visualisation du feu de signalisation

- Identification des GPIOs choisis
- GPIO 17 : Barrière de passage 1
- GPIO 27 : Barrière de passage 2
- GPIO 04 : Feu de signalisation
- GPIO 18 : Buzzer

- Test de la Raspberry sous batterie
  - Low Voltage affiché
  - Constat d'une perte significative le temps d'un instant

- Test de l'agencement de l'IHM sur la Raspberry

Rapport de réunion 17/03/2022

- Le capteur fonctionne avec l'application
- L'affichage se déroule comme prévu
- Problème du capteur au niveau de la couleur
  - Le capteur se met à 1 quand le faisceau réfléchi mais sur du noir, non

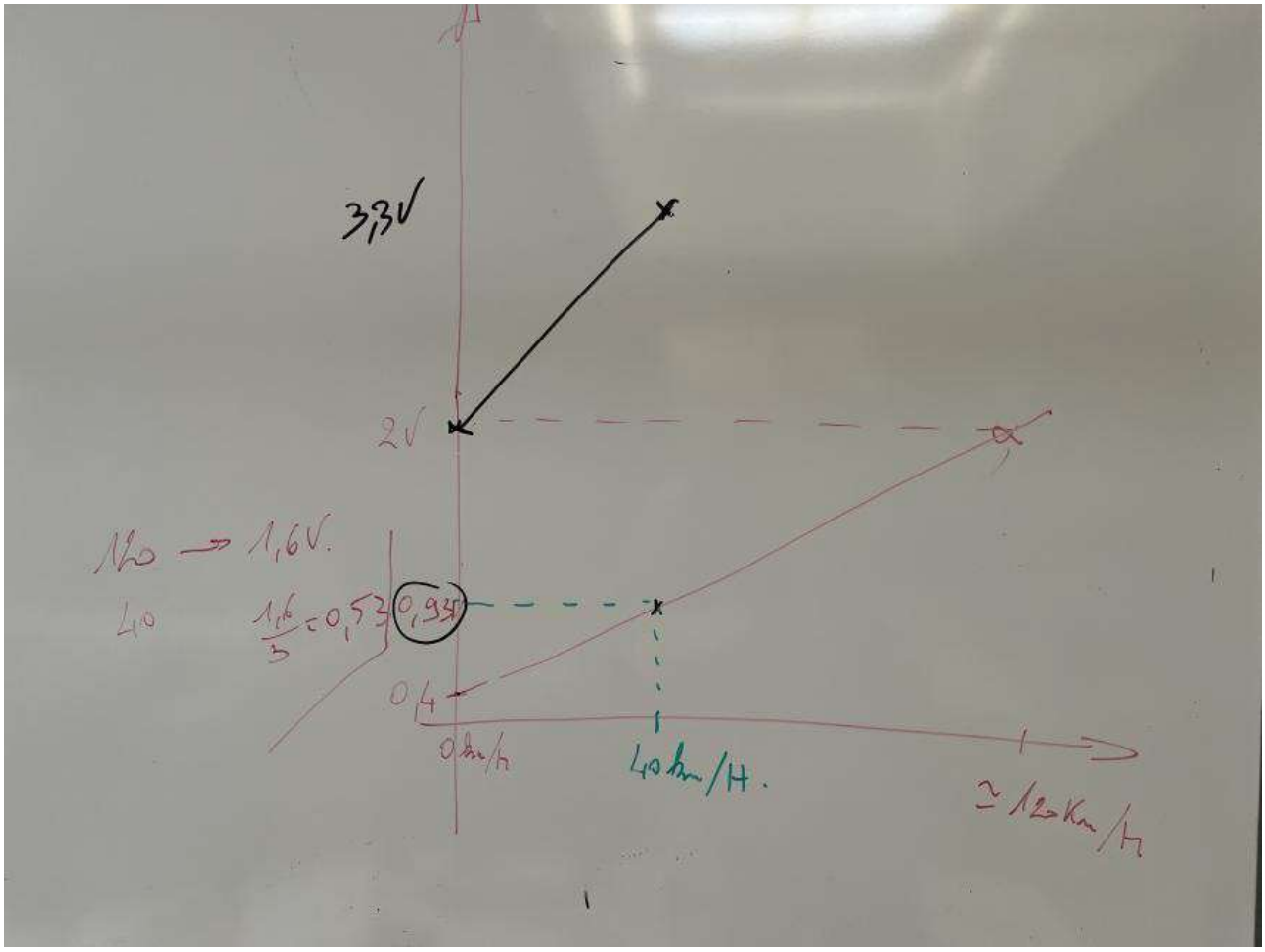
- Début de la réflexion pour régler le problème (Survêtement clair ? Bande claire à coller?)

Rapport de réunion 27/04/2022

Objet de la réunion : Comment gérer l'anémomètre

- Valeur en km/h proportionnelle à la tension
- De base : 0,4V ⇒ 2V
- Après modification : 2V ⇒ 3,3V
- Plage de mesure utilisée : 0 km/h ⇒ 40 km/h
- Précision : Environ 1/10^{e} de km/h de précision

Conclusion : Récupérer la valeur en km/h = faire un calcul à partir de la tension reçue sur le bus I2C



Rapport de réunion 28/04/2022

Objet de la réunion : Modification de la gestion de l'anémomètre

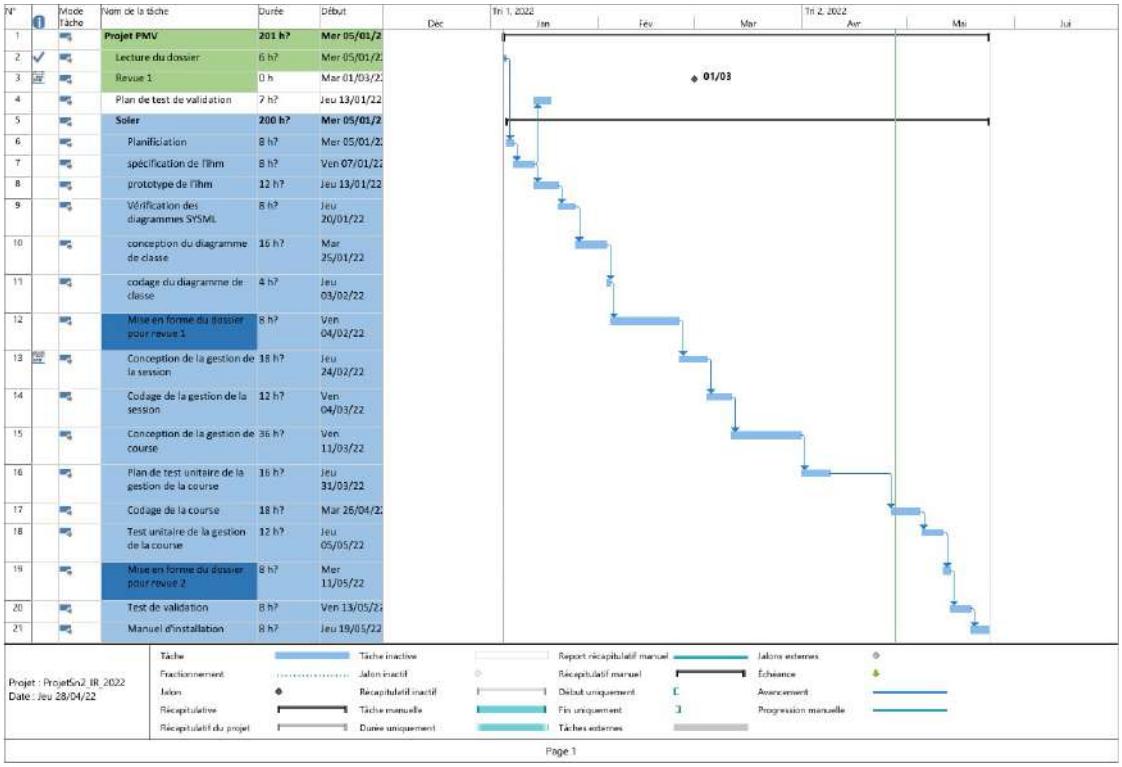
- De base : 0,4V ⇒ 2V
- Après modification : 1,06V ⇒ 3,3V
- Plage de mesure utilisée : 0km/h ⇒ 50km/h
- Précision modifiée : Environ 1/7^{e} de km/h de précision

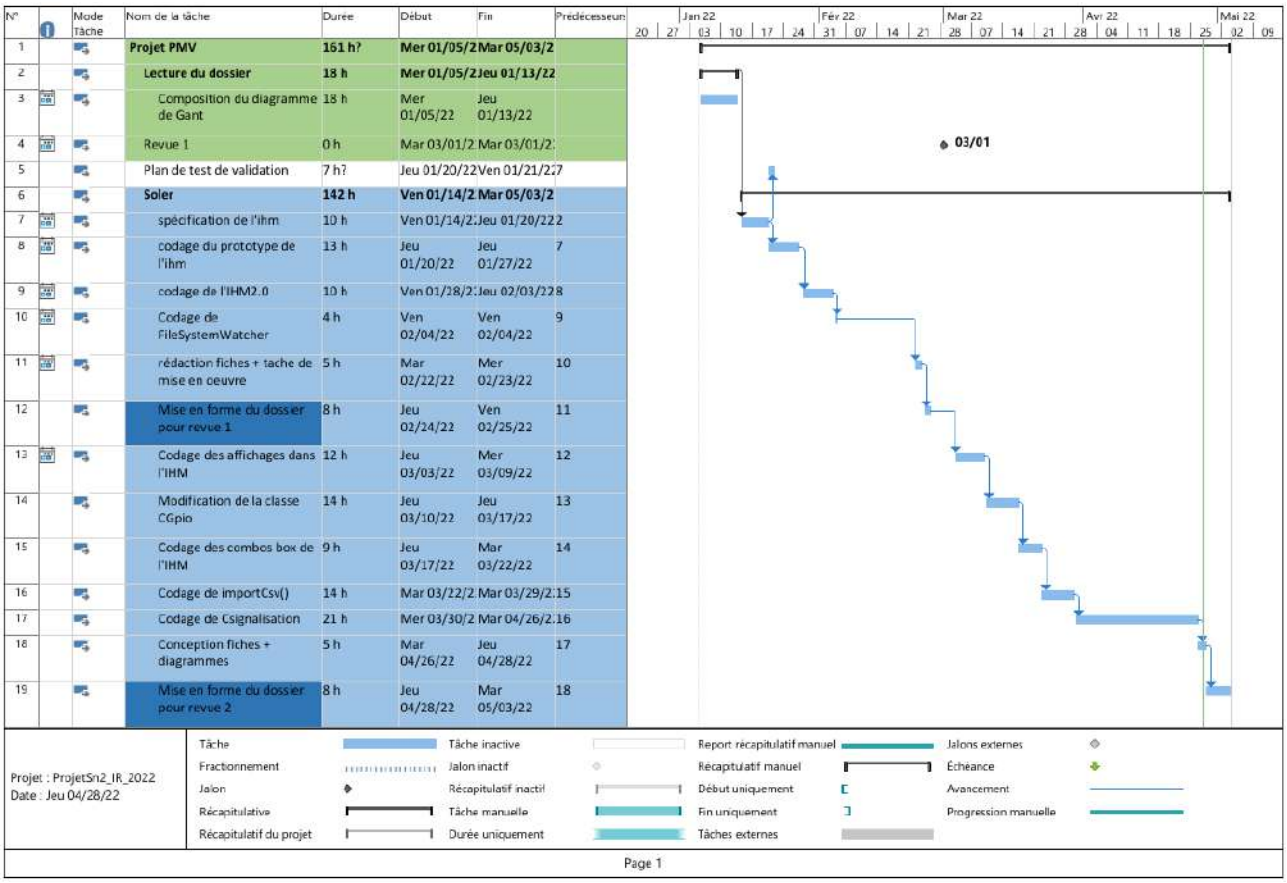
PARTIE INDIVIDUELLE SOLER JULIEN

J'ai pris comme rôle celui de l'IR2. Je dois m'occuper de la gestion de plusieurs parties de l'application : **gestion de la session ; gestion de la course et des coureurs ; gestion de l'écran ; gestion des barrières d'arrivées**. Mon rôle est alors d'assurer le bon fonctionnement de l'application sur la Raspberry et permettre au code de mes deux autres camarades de fonctionner correctement. J'ai choisi ce rôle car il était pour moi au centre de ce projet, c'était une sorte de responsabilité à tenir. De plus, je voulais développer moi même l'IHM, c'est à dire la partie graphique, mais aussi pouvoir développer mes compétences en C++ avec Qt-Creator, le langage et l'EDI utilisés. Cela me motive de savoir que le projet ne se réalisera pas si je ne travaille pas correctement et rapidement.

Partie Professionnelle

Gestion du projet





N°	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		**Projet PMV**	**161 h?**	**Mer 01/05/2**	**Mar 05/03/2**	
2		**Lecture du dossier**	**18 h**	**Mer 01/05/2**	**Jeu 01/13/22**	
3		Composition du diagramme de Gant	18 h	Mer 01/05/22	Jeu 01/13/22	
4		Revue 1	0 h	Mar 03/01/2	Mar 03/01/2	
5		Plan de test de validation	7 h?	Jeu 01/20/22	Ven 01/21/22	7
6		**Soler**	**142 h**	**Ven 01/14/2**	**Mar 05/03/2**	
7		spécification de l'ihm	10 h	Ven 01/14/2	Jeu 01/20/22	2
8		codage du prototype de l'ihm	13 h	Jeu 01/20/22	Jeu 01/27/22	7
9		codage de l'IHM2.0	10 h	Ven 01/28/2	Jeu 02/03/22	8
10		Codage de FileSystemWatcher	4 h	Ven 02/04/22	Ven 02/04/22	9
11		rédaction fiches + tache de mise en oeuvre	5 h	Mar 02/22/22	Mer 02/23/22	10
12		Mise en forme du dossier pour revue 1	8 h	Jeu 02/24/22	Ven 02/25/22	11
13		Codage des affichages dans l'IHM	12 h	Jeu 03/03/22	Mer 03/09/22	12
14		Modification de la classe CGpio	14 h	Jeu 03/10/22	Jeu 03/17/22	13
15		Codage des combos box de l'IHM	9 h	Jeu 03/17/22	Mar 03/22/22	14
16		Codage de importCsv()	14 h	Mar 03/22/2	Mar 03/29/2	15
17		Codage de Csignalisation	21 h	Mer 03/30/2	Mar 04/26/2	16
18		Conception fiches + diagrammes	5 h	Mar 04/26/22	Jeu 04/28/22	17
19		Mise en forme du dossier pour revue 2	8 h	Jeu 04/28/22	Mar 05/03/22	18

Planification prévisionnelle effectuée en début de projet (1) confrontée à celle des tâches réelles réalisées (2)

Gestion de la session

Le but final de la gestion de la session est :

- Si la dernière session n'a pas été enregistrée avant la fermeture de l'application, elle est automatiquement sauvegardée dans la base de donnée, et affichée lors du redémarrage de l'application.
- Si la dernière session avait été enregistrée, il faut alors choisir un nom pour cette session (une date par exemple) et démarrer la nouvelle session (cela permet d'activer tous les boutons de la gestion de la course et des coureurs).

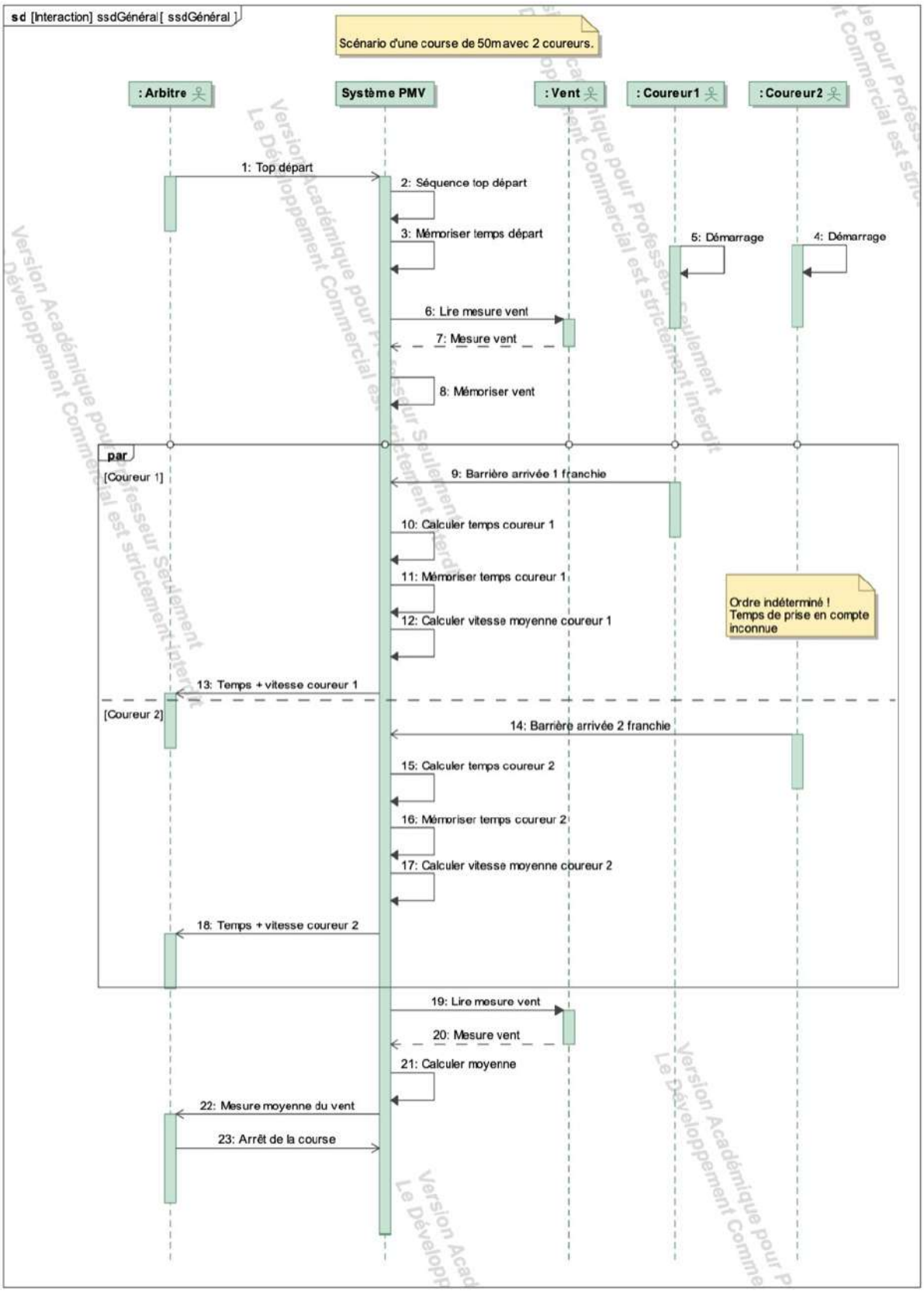
Pour ce qui est de la gestion de la course et des coureurs, elle se fait en 2 parties :

- La partie "Boutons" qui permet de piloter l'avancement du départ, c'est à dire :

  Préparation ; À vos marques ; Prêt et Partez (qui pilote aussi les feux de signalisation en fonction de l'avancement) mais aussi un bouton stop pour arrêter la course et l'annuler si un élève tombe durant la course par exemple. (Voir **Spécification techniques** pour comprendre l'état des boutons en fonction de la situation). Les boutons ici sont grisés car une session n'a toujours pas été démarrée.

- La partie "Tableau" qui regroupe toutes les données enregistrées et calculées durant la course (le temps de la course, la vitesse moyenne du coureur et du vent) ainsi que 2 colonnes pour le nom des deux coureurs qui s'affrontent en duel.

Scénario d'une course avec 2 coureurs



Gestion de l'écran

La gestion de l'écran comprend toute l'interface graphique de l'application et l'ergonomie de son utilisation. Ce fut toute une réflexion autour des besoins et des contraintes techniques que nous avions (taille de l'écran, visibilité des informations, etc).

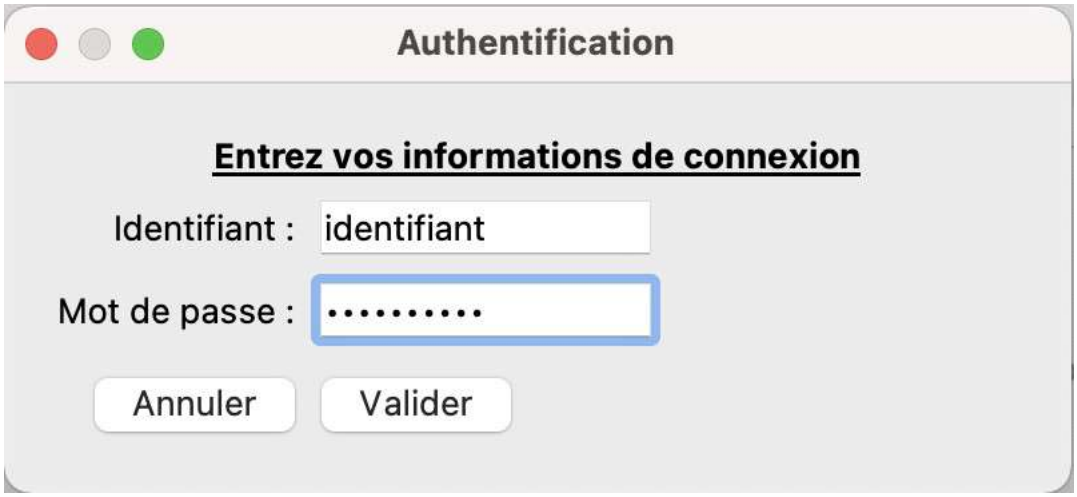
Les professeurs responsables du projet ont débattu et ont donc décidé de partir sur un écran tactile d'environ 7 pouces. Nous l'avons testé et validé selon ces points importants :

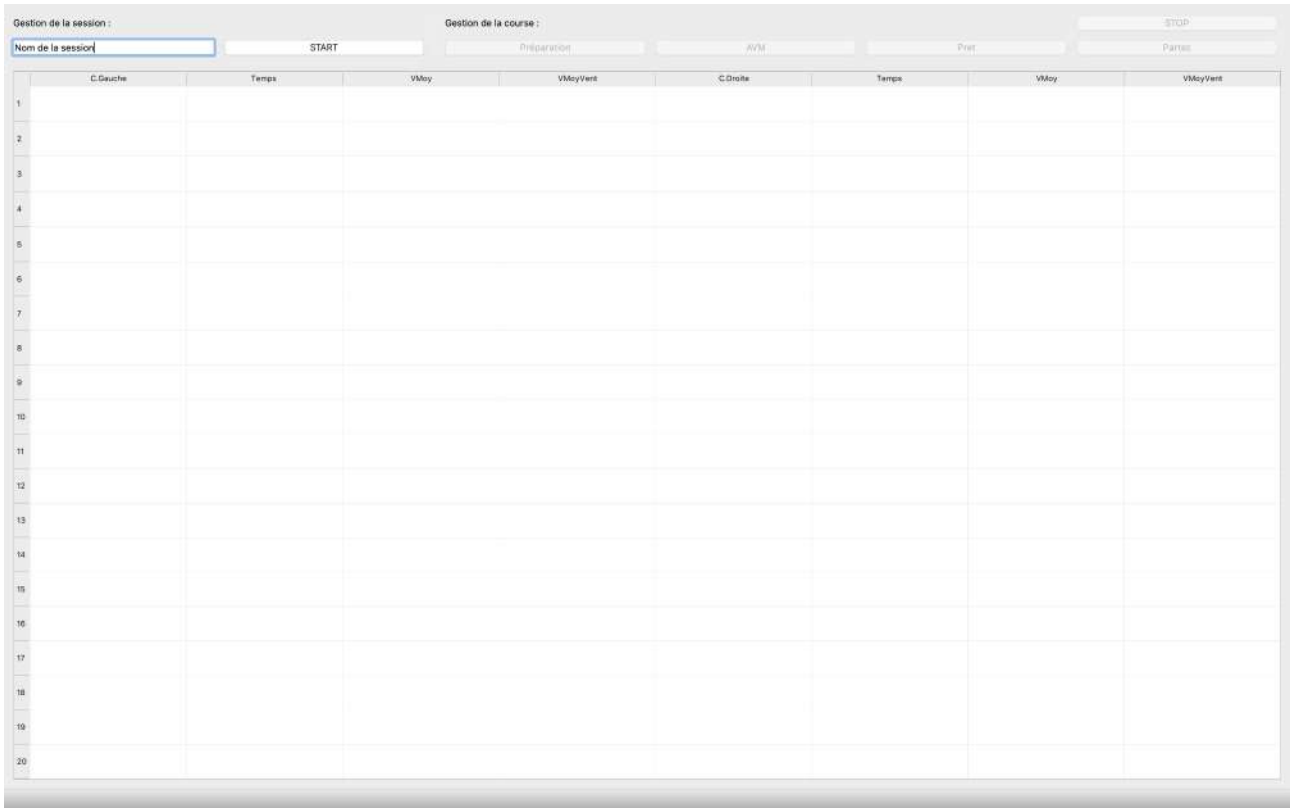
- 7 pouces sont suffisants pour une utilisation optimale
- Le tactile réagit bien et la précision du tactile est correcte
- La nappe pour connecter l'écran est assez longue pour pouvoir créer un support avec l'imprimante 3D pour une utilisation et une protection optimale du système

Nous avons ensuite effectué un brainstorming (voir photo page…) pour commencer à réfléchir sur tous les boutons et fonctionnalités que j'avais à développer pour aller au plus simple et efficace.

Ensuite, après quelques discussions et une fiche de lecture croisée, nous nous sommes mis d'accord sur une architecture de l'interface graphique que j'ai développée.

Nous noterons que l'application se lance directement en plein écran dès l'allumage de la Raspberry PI pour une meilleure efficacité et une meilleure rapidité d'exécution.





Vous retrouverez ci-dessus la pop-up d'authentification (1), l'IHM 2.0 complète (2), ainsi que la barre de menus (3) qui comprend les menus *Système* et *Outils* (voir **Spécification techniques : IHM 2.0**)

Gestion des barrières d'arrivées

La gestion des barrières d'arrivée consiste à pouvoir détecter le passage du coureur en quelques millisecondes maximum tout en sachant si c'est le coureur de droite ou de gauche (2 coureurs par course). C'est ici la pièce maîtresse du système car c'est elle qui garantit une bonne fiabilité de mesure du temps tout en respectant le cahier des charges.

Nous utilisons donc un capteur de passage connecté aux GPIOs 17 et 27. Le capteur est à un niveau logique stable "0". En effet, lorsqu'il y a un passage devant ce laser, celui-ci va réfléchi et capté par le capteur (qui possède un récepteur) qui sera ainsi à un niveau logique "1" durant un très court laps de temps.

Ce changement de niveau logique va s'effectuer dans un fichier s'appelant "value" dans le dossier de la GPIO en question (la 17 ou la 27). Ainsi, notre objet QfileSystemWatcher va remarquer le changement d'état du fichier "value" commandé par le système Linux et va ensuite émettre un signal se nommant ***fileChanged()*** connecté préalablement avec mon slot ***on_fileChanged()*** pour exécuter le code dans ce dernier.

Enfin, cette méthode va prendre une mesure du temps (la 2e de la course) pour que nous puissions ensuite calculer le ΔT avec la 1ère mesure (mesure prise lors du départ des coureurs) pour pouvoir donc calculer le temps de la course du coureur en question mais aussi sa vitesse moyenne car la course à une longueur fixe de 50m.

Nous noterons que la classe CcapteurPassage ne gère que le passage des coureurs et la 2e prise de temps. Le reste étant dirigé par Capp.cpp avec les méthodes ***calculateTime()*** et ***calculateSpeed()***.

```
#ifndef CCAPTEURPASSAGE_H
#define CCAPTEURPASSAGE_H

#include <QObject>
#include <QFileSystemWatcher>
#include <QDateTime>
#include "biblis/cgpio.h"


class CCapteurPassage : public QObject
{
    Q_OBJECT

public:
    explicit CCapteurPassage(QObject * parent = nullptr, int gpio = 17, int ordre = 1);
    ~CCapteurPassage();

private:
    QFileSystemWatcher _file;
    CGpio *_gpio;
    int _ordre;

signals:
    void sig_coureurArrived(int no, QDateTime dt2);

public slots:
    void on_filechanged();
};

#endif // CCAPTEURPASSAGE_H
```

CCapteurPassage.h

```cpp
#include "ccapteurpassage.h"

CCapteurPassage::CCapteurPassage(QObject *parent, int noGpio, int ordre) : QObject(parent)
{
    _ordre = ordre;
    _gpio = new CGpio(this, noGpio, IN);
    QString path("/sys/class/gpio/gpio"+QString::number(noGpio)+"/value");
    _file.addPath(path);
    connect(&_file, &QFileSystemWatcher::fileChanged, this, &CCapteurPassage::on_filechanged);

}

CCapteurPassage::~CCapteurPassage()
{
    delete _gpio;
}

void CCapteurPassage::on_filechanged()
{
    QDateTime dt2 = QDateTime::currentDateTime();
    emit sig_coureurArrived(_ordre, dt2);
}
```

CCapteurPassage.cpp

```cpp
#include "capp.h"

CApp::CApp()
{
    _capteurPassage1 = new CCapteurPassage(nullptr, 17, 1);
    _capteurPassage2 = new CCapteurPassage(nullptr, 27, 2);
    _ligne = -1;
}

CApp::~CApp()
{
    delete _capteurPassage1;
    delete _capteurPassage2;
}


void CApp::on_timerStart()
{
    _ligne++;

    _dt1 = QDateTime::currentDateTime();
    connect(_capteurPassage1, &CCapteurPassage::sig_coureurArrived, this, &CApp::on_timerStop);
    connect(_capteurPassage2, &CCapteurPassage::sig_coureurArrived, this, &CApp::on_timerStop);
}

void CApp::on_timerStop(int ordre, QDateTime dt2)
{
    CCapteurPassage *cp = static_cast<CCapteurPassage *>(sender());
    calculateTime(ordre, dt2);
    disconnect(cp, &CCapteurPassage::sig_coureurArrived, this, &CApp::on_timerStop);
}

void CApp::calculateSpeed(int ordre)
{
    double deltaTs = _deltaTs;
    double deltaTms = _deltaTms;

    if(_deltaTm > 0)
        deltaTs += 60; //Si la course dure plus d'1 min, on rajoute 60s au delta T qui représente que les secondes
    deltaTs = deltaTs + (deltaTms / 1000);

    double Calcul = 50.0 / deltaTs * 3.6;
    QString resultatVitesse = QString::number(Calcul, 'g', 2);
    emit sig_resVitesse(resultatVitesse, ordre, _ligne);

}

void CApp::calculateTime(int ordre, QDateTime dt2)
{
    _deltaTms = _dt1.msecsTo(dt2);
    _deltaTm = _deltaTms / 60000;
    _deltaTs = (_deltaTms % 60000) / 1000;
    _deltaTms = (_deltaTms % 60000) % 1000;
    calculateSpeed(ordre);

    QString resultatTemps = QString::number(_deltaTm) + " : " + QString::number(_deltaTs) +" : " + QString::number(_deltaTms);
    emit sig_resTemps(resultatTemps, ordre, _ligne);

}
```

CApp.cpp

Partie Physique : Colorimétrie

Contexte

J'ai choisi de parler de la colorimétrie pour la partie physique de mon projet. Ce choix est pour moi le plus judicieux car nous avons rencontré un problème majeur durant le test du passage d'un coureur devant le capteur après le développement de la classe CCapteurPassage.

En effet, lorsque nous avons testé le capteur, nous avons essayé avec une feuille blanche et la distance de détection était d'environ 1m50 / 2m. Je suis ensuite passé à 1m du capteur, puis à 50cm jusqu'à environ 10 / 15cm du capteur pour remarquer que le capteur avait enfin détecté mon passage. Nous avons discuté environ 20 minutes jusqu'à comprendre la nature du problème.

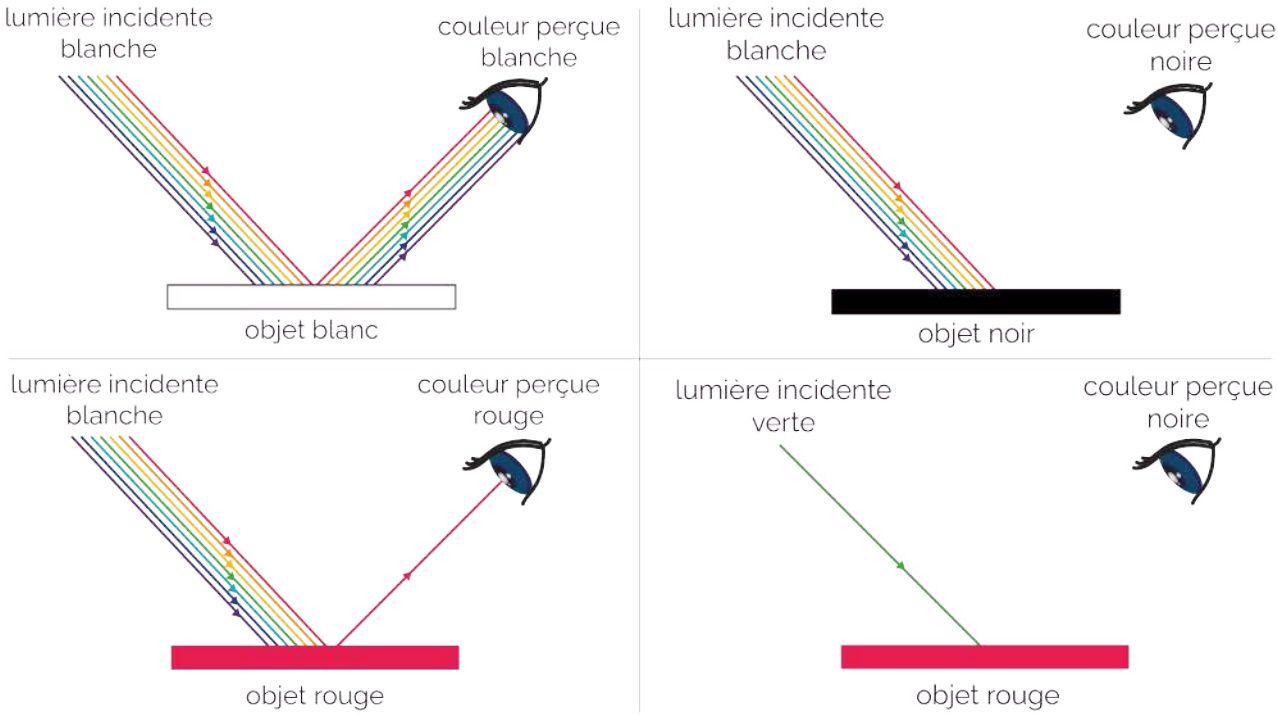
Il s'avère que ce jour là je portais un survêtement noir. C'est alors la couleur qui pose un problème sur la réflexion du faisceau laser émis par le capteur. Le capteur ne recevant pas le rayon qui ne peut être réfléchi, il n'y a pas de détection de passage. La puissance du laser et l'absence de noir pur explique pourquoi le capteur détecte mon passage à environ 10 / 15cm.

Absorption de la lumière par les objets

Les objets peuvent absorber une partie de la lumière en fonction de leur nature physique ou chimique. Une fois que l'objet reçoit de la lumière d'une source primaire, il ne renverra pas forcément toutes les radiations lumineuses de cette source.

En effet, la lumière blanche émise par le Soleil ou une lampe est composée de plusieurs couleurs. On peut d'ailleurs la décomposer pour obtenir son spectre.

On observe ci-dessous des objets de différentes couleurs éclairés par des lumières blanches ou vertes.



On remarque que :

- un objet blanc réémet toutes les radiations visibles
- un objet noir absorbe quasiment toutes les radiations visibles
- si un objet est éclairé en lumière blanche et nous apparaît comme rouge, c'est donc qu'il absorbe toutes les radiations sauf le rouge
- si la lumière éclairant l'objet n'est pas blanche, mais verte par exemple, alors un objet rouge de réémettra pas de lumière : l'objet rouge sera vu noir car la lumière verte ne contient pas de rouge

Conclusion

Pour conclure, nous avons trouvé la nature du problème grâce à nos connaissances de physique acquises lors de l'année. Nous pouvons ainsi trouver des solutions pour parer à ce problème. Certaines pistes penchent pour un chasuble quand d'autres plus ambitieuses préfèrent bricoler le capteur. C'est alors une des problématiques avec son lot de solutions que nous laissons aux futurs étudiants qui travailleront sur le projet.

Annexes

Spécifications techniques : Conception du prototype de l'IHM

QMessageBox de Login :

2 boutons :

- OK
- Cancel

LineEdit :

- Identifiant = "ID"

Spécification :

- Au bout de 3 essais : fermeture de l'application

QMessageBox de Mot de passe :

2 boutons :

- OK
- Cancel

LineEdit :

- Mot de passe = "MDP"

Spécification :

- Au bout de 3 essais: fermeture de l'application

Page de gestion (session et course) :

9 boutons :

- Start la session ⇒ Devient stop la session
- Préparation - des coureurs - (accès au bouton suivant)
- AVM – À Vos Marques (grise le précédent) (accès au bouton suivant)
- Prêt (grise les deux précédents) (accès au bouton suivant)
- Partez (grise les trois précédents) (accès au bouton suivant)
- Export CSV (disponible quand aucune session en cours) ⇒ QMessageBox(Yes / No)
- STOP Course (retour à la préparation comme pour une fin de course normale)
- Test des capteurs ⇒ QmessageBox(Cancel)
- Quitter l'application ⇒ QmessageBox(Yes / No)

LineEdit :

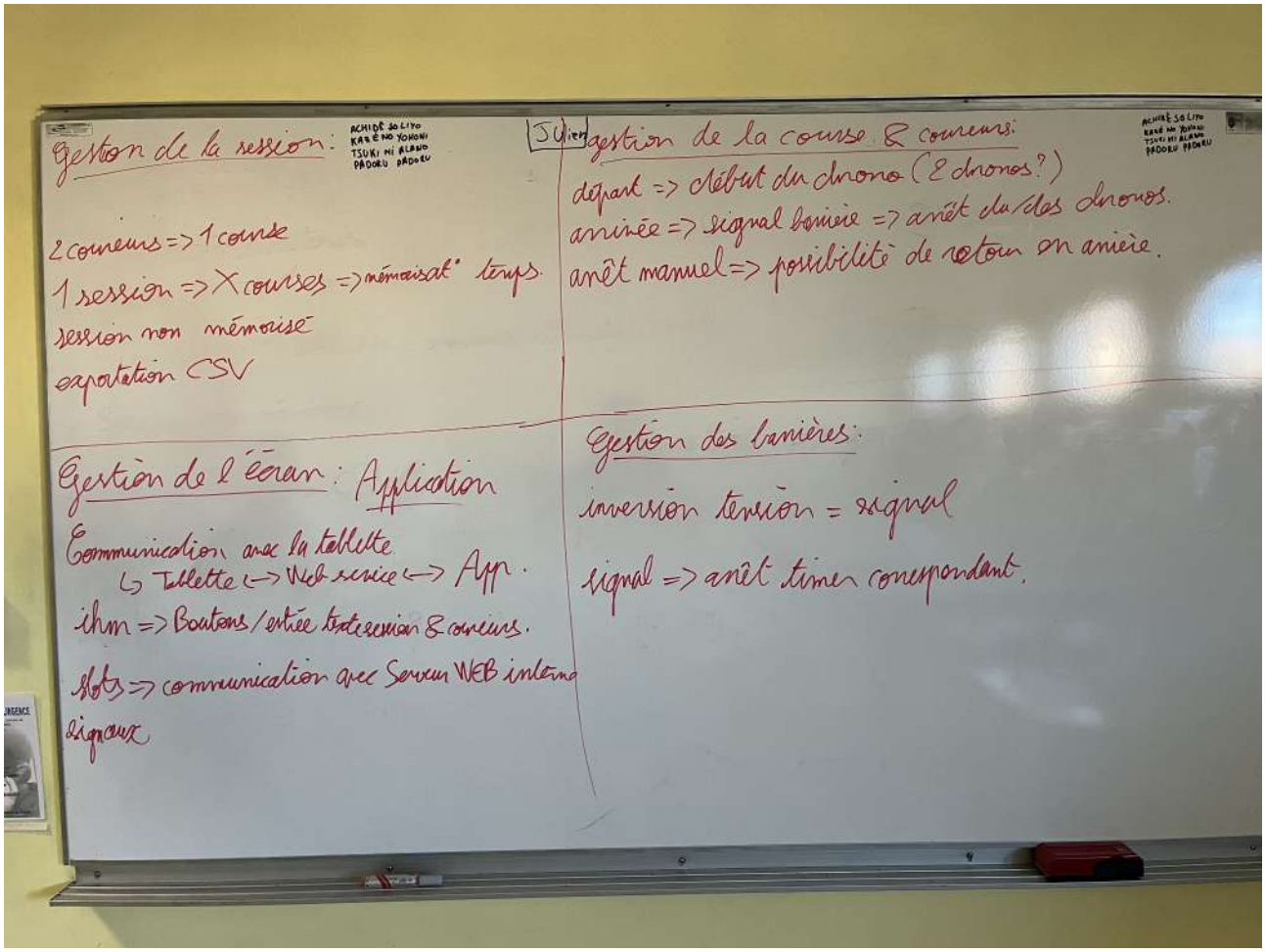
- Nom de la session (push button start = Nom de la session read-only)

TableWidget :

- 20(h) x 2(l) Vmoy ⇒ Vitesse moyenne du coureur sur 50m
- 20(h) x 2(l) VmoyVent ⇒ Vitesse moyenne des rafales de vent sur le temps de course
- 20(h) x 2(l) Temps ⇒ Temps pour parcourir les 50m
- 20(h) x 2(l) Coureur ⇒ Nom des coureurs de chaque couloir (2 par 2)

Labels :

- Gestion de la course
- Gestion de la session



Brainstorming avant la conception du prototype

1. Fiche de rapport de lecture croisée.		Réf. FRLC-SJ-RV1
Projet	***Auteur***	***Date***
PMV	Soler Julien	25/01/2022
Produit	***Durée de la réunion***	
IHM (interface graphique)	15 minutes	
Lecteur(s)	***Contrôleur des corrections***	
Bothuan Erwan / Jullian Alexandre	Mr Philippe Antoine	

Liste des erreurs détectées

- Boite de connexion en deux parties devrait être remplacée par boite dialogue avec dissimulation du mot de passe
- Connexion menu « Système » (Redémarrage de la pi, Arrêt, etc, etc) aux appels systèmes Linux

Cette fiche de rapport de lecture croisée a été rédigée dans le but d'optimiser le prototype de l'IHM et répondre au mieux aux besoins du projet

Fiche de suivi de modification.		**Réf.**	FSM-SJ-RV1
Auteur	Julien Soler	***Réf. FLC***	FRLC-SJ-RV1

Projet	***Demandeur***	***Date***
PMV	Erwan Bothuan / Alexandre Jullian	25/01/2022

Produit modifié	***Action réalisée***	***Date de fin***
- Ihm.h - Ihm.cpp - Ihm.ui *Ajout de :* - CLoginDialog.h - CLoginDialog.cpp - CLoginDialog.ui	Codage d'une QDialog pour permettre un système de login efficace. Fermeture de l'application si pression du bouton fermé ou valider. Fermeture de l'application après 3 échec. *Spécifications :* - 2 lineEdit (ID et MDP) - 2 label (ID et MDP) - 2 boutons (Annuler et Valider) *Codage d'un menu système pour :* - arrêter la Raspberry PI - Redémarrer la Raspberry PI	28/01/2022

Cette fiche de suivi de modification recense les modifications effectuées sur l'IHM après la fiche de rapport de lecture croisée qui énumère les problèmes observés.

Spécification techniques : IHM 2.0

Qdialog :

LineEdit :

- Identifiant
- Mot de passe

2 boutons :

- Annuler
- Valider

Spécifications :

- Fermeture de l'application au bout de 3 essais
- Mot de passe non visible

Page de gestion :

6 boutons :

- Start la session ⇒ Devient stop la session
- Préparation – des coureurs –
- AVM – À Vos Marques – (grise le précédent)
- Prêt (grise les 2 précédents)
- Partez (grise les 3 précédents)
- STOP – course – (retour à la préparation comme pour une fin de course normale)

LineEdit :

- Nom de la session (push du bouton start = Nom de la session en ReadOnly)

TableWidget :

- 20(h) x 2(l) Vmoy ⇒ Vitesse moyenne du coureur sur 50m
- 20(h) x 2(l) VmoyVent ⇒ Vitesse moyenne des rafales de vent sur le temps de course

- 20(h) x 2(l) Temps ⇒ Temps pour parcourir les 50m
- 20(h) x 2(l) Coureur ⇒ Nom des coureurs de chaque couloir (2 par 2)

2 Labels :

- Gestion de la session
- Gestion de la course

Spécification :

- Seul les deux colonnes Coureurs du TableWidget peuvent être modifiées et sélectionnées
- Besoin de "start" la session avant d'avoir accès à la gestion de la course

Barre de menus :

Outils :

- Export CSV
- Test de la fonctionnalité des capteurs

Système :

- Arrêter la Raspberry
- Redémarrer la Raspberry
- Quitter l'application PMV

Commentaire :

Cette fiche décrit toutes les spécifications techniques de l'IHM après les modifications effectuées

Tâche de Mise en Œuvre (TMO) : Simulation d'une course

Résultat attendu :

Affichage du temps et de la vitesse moyenne de chaque coureur après passage devant le capteur.

Démarche :

La course se déroule de la façon suivante :

- Commun aux deux coureurs :
  - Préparation
  - À Vos Marques
  - Prêt
  - Partez
  - Acquisition de la date de départ commune aux deux coureurs
- Pour chaque coureur :
  - Arrivée du coureur
  - Acquisition de la date d'arrivée
  - Calcul du ΔT : $T_{arrivée} - T_{départ}$
  - Traitement de la data (mise en forme)
  - Calcul de la vitesse moyenne sur les 50 mètres
  - Affichage du temps et de la vitesse moyenne dans l'affichage selon le coureur (1 ou 2)

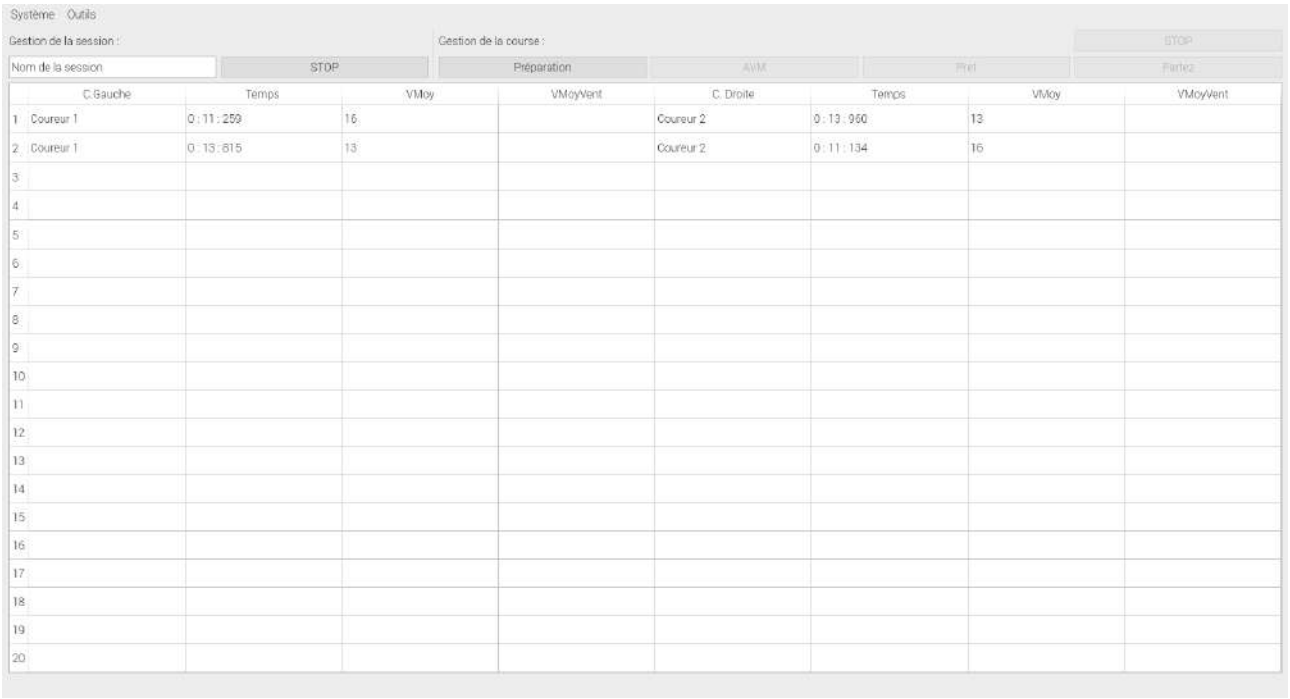
Nous allons ainsi simuler une course en condition réelle en reproduisant le comportement des capteurs (c'est à dire être tout le temps à 0 et passer à 1 lorsque le coureur passe devant le capteur) en reliant le 3,3V à la GPIO voulue pour simuler le passage du coureur devant le capteur au moment souhaité en forçant le passage à 1 de la valeur retournée par la GPIO.

Nous réaliserons 2 course de 2 coureurs qui arriveront à des moments différents.

- Course 1 :
  - Coureur 1 en 1er
  - Coureur 2 en 2e
- Course 2 :
  - Course 1 en 2e
  - Coureur 2 en 1er

Le but est alors de vérifier la détection du passage du coureur mais aussi le bon fonctionnement de l'affichage et des calculs effectués (temps correspondant au bon coureur ; passage à la ligne suivante pour la course suivante ; cohérence de la vitesse moyenne sur 50 mètres ; etc)

Résultats :



Nous retrouvons ainsi comme prévu l'affichage des temps en fonction du coureur ainsi que sa vitesse moyenne en km/h lors du passage devant le capteur.

Annexes :

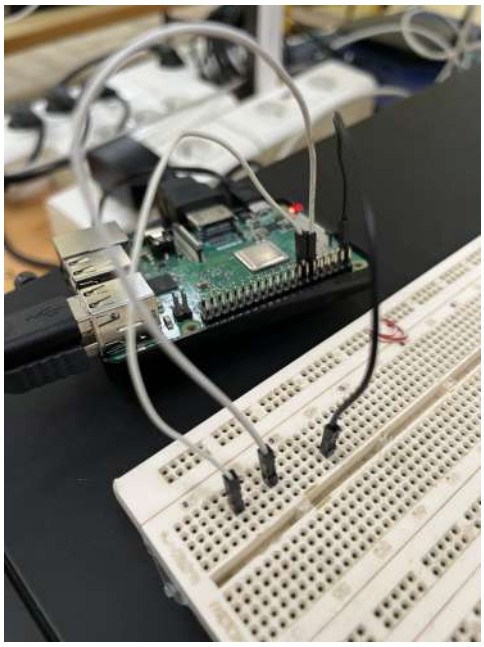
CApp.cpp : Détails des méthodes ***calculateSpeed()*** et ***calculateTime()***

```
void CApp::calculateSpeed(int ordre)
{
    double deltaTs = _deltaTs;
    double deltaTms = _deltaTms;

    if(_deltaTm > 0)
        deltaTs += 60; //Si la course dure plus d'1 min, on rajoute 60s au delta T qui représente que les secondes
    deltaTs = deltaTs + (deltaTms / 1000);

    double Calcul = 50.0 / deltaTs * 3.6;
    QString resultatVitesse = QString::number(Calcul, 'g', 2);
    emit sig_resVitesse(resultatVitesse, ordre, _ligne);

}

void CApp::calculateTime(QDateTime dt2, int ordre)
{
    _deltaTms = _dt1.msecsTo(dt2);
    _deltaTm = _deltaTms / 60000;
    _deltaTs = (_deltaTms % 60000) / 1000;
    _deltaTms = (_deltaTms % 60000) % 1000;
    calculateSpeed(ordre);

    QString resultatTemps = QString::number(_deltaTm) + " : " + QString::number(_deltaTs) +" : " + QString::number(_deltaTms);
    emit sig_resTemps(resultatTemps, ordre, _ligne);

}
```

Cihm.cpp : Détails des méthodes ***on_afficherResTemps()*** et ***on_afficherResVitesse()***

```
void CIhm::on_afficherResTemps(QString resultatTemps, int ordre, int ligne)
{
    if(ordre == 1)
        ui->tableWidget->setItem(ligne,1,new QTableWidgetItem(resultatTemps));
    else
        ui->tableWidget->setItem(ligne,5,new QTableWidgetItem(resultatTemps));
}

void CIhm::on_afficherResVitesse(QString resultatVitesse, int ordre, int ligne)
{
    if(ordre == 1)
        ui->tableWidget->setItem(ligne,2,new QTableWidgetItem(resultatVitesse));
    else
        ui->tableWidget->setItem(ligne,6,new QTableWidgetItem(resultatVitesse));
}
```

Montage de la simulation : Fils blanc = GPIO 17 et 27 (droite à gauche) ; Fil noir = 3,3V



Journal de bord

Jeudi 20 Janvier :

Codage des QmessageBox

Codage des boutons disable enable

Déellage du bug page login : quand on appuie sur cancel

Codage Page Login

Cellule du tableWidget en ReadOnly

Vendredi 21 Janvier :

Codage de la réapparition du login et mot de passe (3x maximum)

Mardi 25 Janvier :

Fiche de spécification IHM

Fiche de lecture croisée IHM

Codage des dernières QmessageBox pour les boutons

Rectification : ReadOnly sur le nom de la session seulement PENDANT la session

Mercredi 26 Janvier :

Début création et codage de la loginDialog

Jeudi 27 Janvier :

Finition de la loginDialog

Recherche sur les appels système Linux

Discussion avec l'équipe pour choisir le logo

Jeudi 03 Février :

Choix des GPIO

Test de l'IHM sur la Raspberry

Test de la Led du feu à 50m

Test de la distance du wifi (40m)

⇒ Voir Réunion 03/02

Rédaction des fiches

Vendredi 04 Février :

Mot de passe en étoile dans CLoginDialog

Modification de l'IHM ⇒ Simplification et épuration

Fin du codage de Redémarrer/Quitter/Éteindre

Codage de CCapteurPassage pour test le FileSystemWatcher puis test avec le vrai capteur

Rédaction de fiches

Mardi 22 Février :

Mise en page du rapport de projet

Réunion avec le professeur de sport (client) ⇒ présentation de l'avancée du développement

Idée émanante ⇒ créer une entreprise qui commercialise le projet (produit)

Mardi 01 Mars au Jeudi 03 Mars :

Codage de l'arrêt de la course quand le signal est envoyé par le capteur (au passage de l'élève)

Nombreux problèmes ⇒ QfileSystemWatcher ne fonctionne pas

⇒ Utilisation d'un QThread

Vendredi 04 Mars :

Avancement du codage pour l'affichage du temps de la course et de la vitesse moyenne après être passé devant le capteur géré par le Thread

Lundi 07 Mars au Mardi 15 Mars :

Modification de la classe CGpio

Possibilité d'utiliser l'objet QfileSystemWatcher de manière continue et fonctionnelle

Affichage du temps ligne par ligne et coureur par coureur avec la vitesse moyenne

Mercredi 16 Mars :

TMO CapteurPassage avec affichage IHM

Jeudi 17 Mars :

Test du capteur sur la GPIO (voir Réunion 17/03)

Codage des ComboBox pour choisir les noms des coureurs suivant la liste importée en CSV.

Début de la réflexion des méthodes de exportCSV et importCSV avec des QFileDialog

Vendredi 18 Mars :

Codage de importCSV mais erreur lors de l'implémentation des noms des élèves dans les ComboBox

Discussion sur le problème du capteur lors du passage devant avec une couleur trop sombre

Mardi 22 Mars :

Codage de importCSV ⇒ Fini

Mercredi 23 Mars :

Codage de exportCSV mais problème lors de la sélection du fichier ⇒ appli qui plante

Jeudi 24 Mars et Vendredi 25 Mars :

exportCSV ⇒ segmentation fault

erreur non réglée

Mardi 29 Mars :

exportCSV utilisera les données dans la BDD et non celles du tableWidget de l'IHM

Début de la réflexion des signaux et méthodes utilisés pour transférer les données à CBdd pour les implémenter ensuite dans la BDD

Début du code de CSignalisation

Jeudi 01 Avril :

Codage de CSignalisation terminé. Aide du professeur (signaux non traités quand ils sont reçu à cause du QThread)

Réaction du feu de signalisation comme nous voulons selon l'état des boutons de départ

Mardi 26 Avril :

Réflexion sur ce que je vais proposer lors de la revue 2

Organisation de la semaine pour faire tous les diagrammes pour la revue 2

Mercredi 27 Avril :

Conception diagramme de classe + simulation de course de 2 coureurs

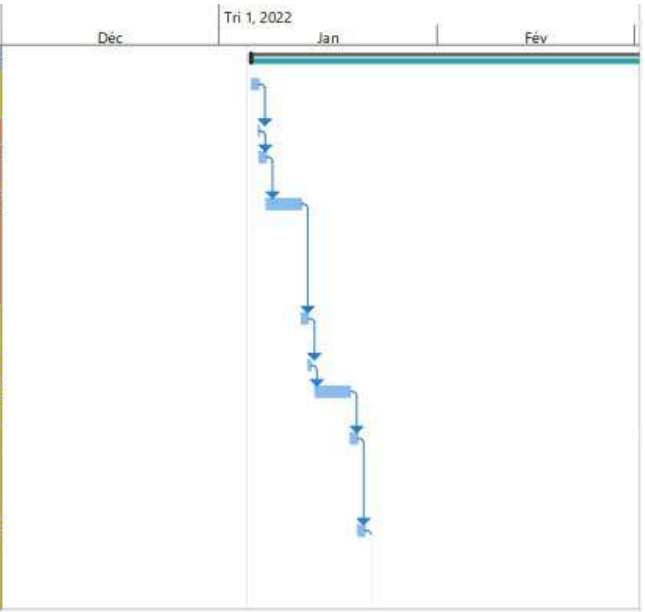
Début de la rédaction du dossier de projet

PARTIE INDIVIDUELLE NOEL NATHAN

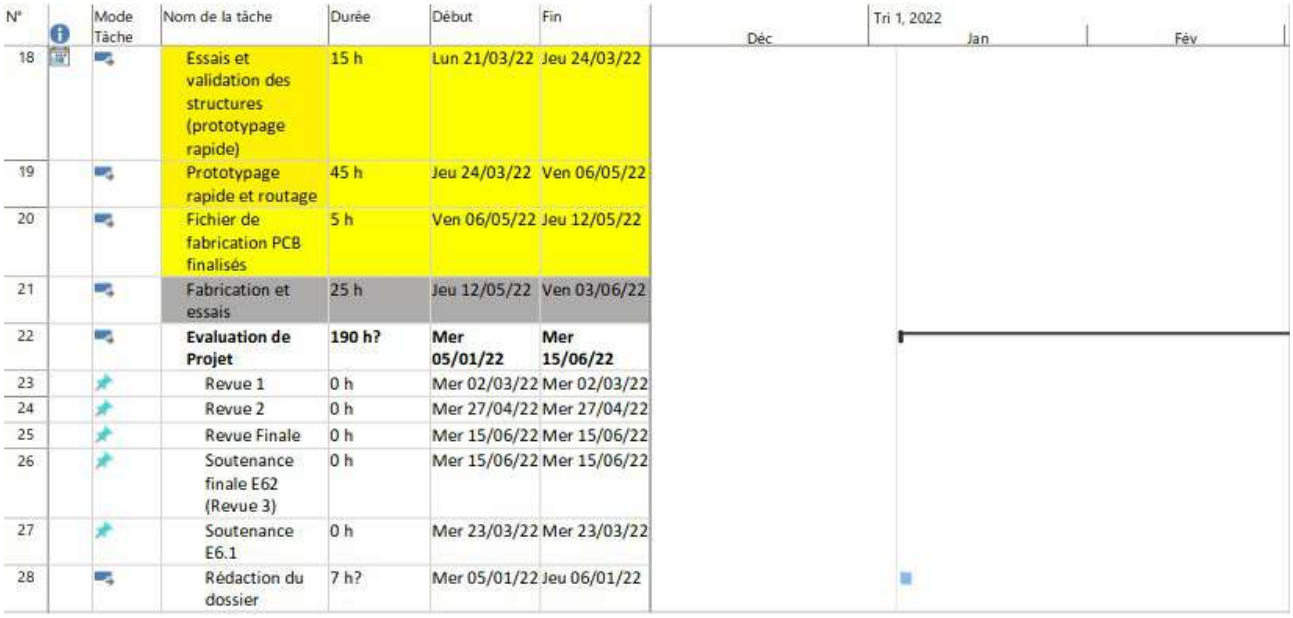
Pour assurer le bon fonctionnement du projet PMV, on m'a attribué le rôle de l'EC1. Ma partie consiste à : Informer les coureurs du départ de la course, le but étant de pouvoir prévenir les coureurs situés au bout du terrain, à 50m. Pour ce faire, on m'a confié le matériel suivant : Un buzzer 12V dont je ne connais pas la référence et un bouton poussoir "Big Dome Pushbutton – Red" de 12V. Ma tâche en tant que EC est la suivante : tester plusieurs LED de couleur rouge et savoir laquelle est la meilleure pour être vu à 50m en plein jour et tester plusieurs buzzer pour savoir lequel génère le plus de bruit et peut être entendu aussi à 50m.

Gestion du projet

N°	Mode Tâche	Nom de la tâche	Durée	Début	Fin
1		**Projet PMV**	**180 h?**	**Mer 05/01/22**	**Ven 03/06/22**
2		Lecture du dossier	4 h	Mer 05/01/22	Jeu 06/01/22
3		Réunion PROJET 1	1 h	Jeu 06/01/22	Jeu 06/01/22
4		Diagramme de Gantt	4 h	Jeu 06/01/22	Ven 07/01/22
5		Documentation sur principe du chronométrage dans les courses d'athlétisme	3 h	Ven 07/01/22	Mer 12/01/22
6		Documentation du buzzer	3 h	Mer 12/01/22	Jeu 13/01/22
7		Test du buzzer	2 h	Jeu 13/01/22	Jeu 13/01/22
8		Test d'un des hauts-parleurs	4 h	Ven 14/01/22	Mer 19/01/22
9		Proposer un autre composant pilotable en filaire	4 h	Mer 19/01/22	Jeu 20/01/22
10		Test d'un amplificateur BlueTooth pilotable par Rpi	4 h	Jeu 20/01/22	Ven 21/01/22



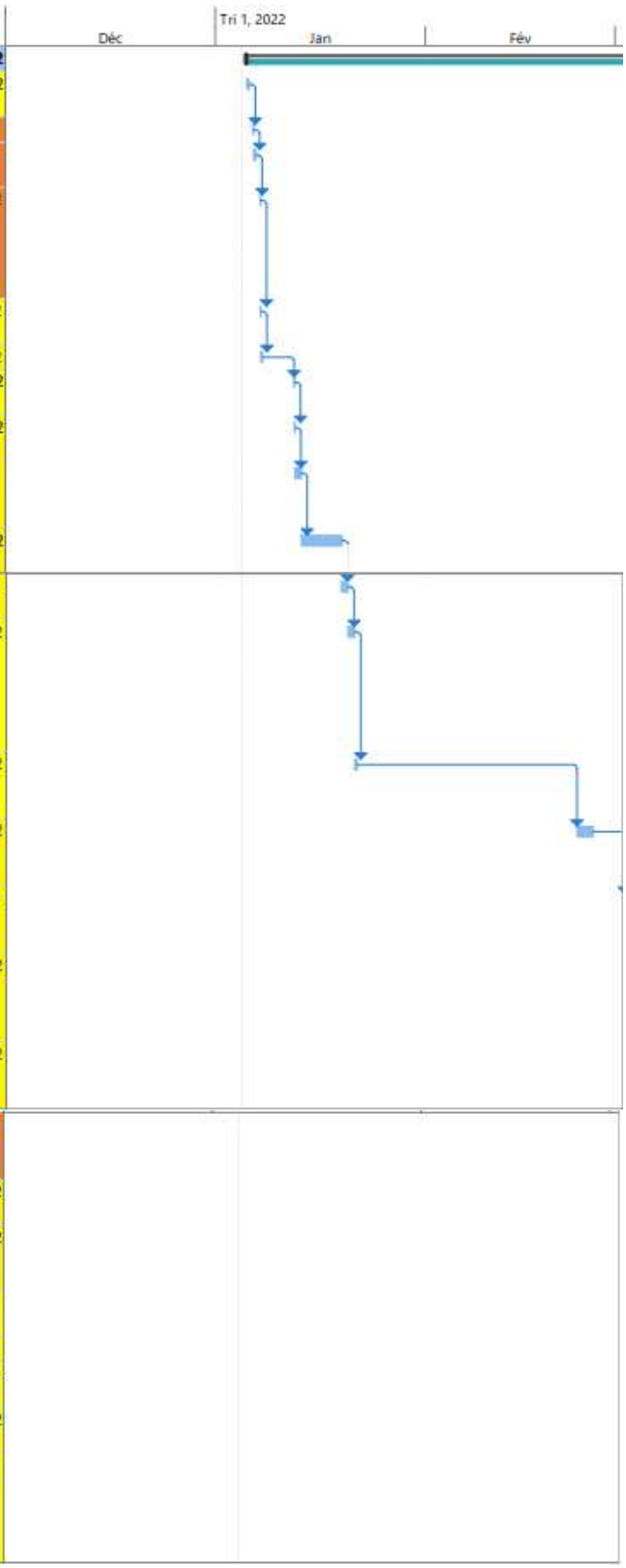
N°	Mode Tâche	Nom de la tâche	Durée	Début	Fin
11		Test du voyant du bouton poussoir	7 h	Ven 21/01/22	Jeu 27/01/22
12		Tester une autre LED haute luminosité si l'intensité n'est pas suffisante	10 h	Jeu 27/01/22	Jeu 03/02/22
13		Faire un schéma structurel	7 h	Jeu 03/02/22	Mer 23/02/22
14		essais et validation des structures	17 h	Jeu 24/02/22	Ven 04/03/22
15		Fusion des 3 Schémas Kicad	5 h	Mer 09/03/22	Jeu 10/03/22
16		Réflexion sur la mise en boitier des composants	8 h	Jeu 10/03/22	Mer 16/03/22
17		Dresser La liste des composants de la Carte Electronique	7 h	Jeu 17/03/22	Ven 18/03/22



N°	Mode Tâche	Nom de la tâche	Durée	Début	Fin
18		Essais et validation des structures (prototypage rapide)	15 h	Lun 21/03/22	Jeu 24/03/22
19		Prototypage rapide et routage	45 h	Jeu 24/03/22	Ven 06/05/22
20		Fichier de fabrication PCB finalisés	5 h	Ven 06/05/22	Jeu 12/05/22
21		Fabrication et essais	25 h	Jeu 12/05/22	Ven 03/06/22
22		**Evaluation de Projet**	**190 h?**	**Mer 05/01/22**	**Mer 15/06/22**
23		Revue 1	0 h	Mer 02/03/22	Mer 02/03/22
24		Revue 2	0 h	Mer 27/04/22	Mer 27/04/22
25		Revue Finale	0 h	Mer 15/06/22	Mer 15/06/22
26		Soutenance finale E62 (Revue 3)	0 h	Mer 15/06/22	Mer 15/06/22
27		Soutenance E6.1	0 h	Mer 23/03/22	Mer 23/03/22
28		Rédaction du dossier	7 h?	Mer 05/01/22	Jeu 06/01/22

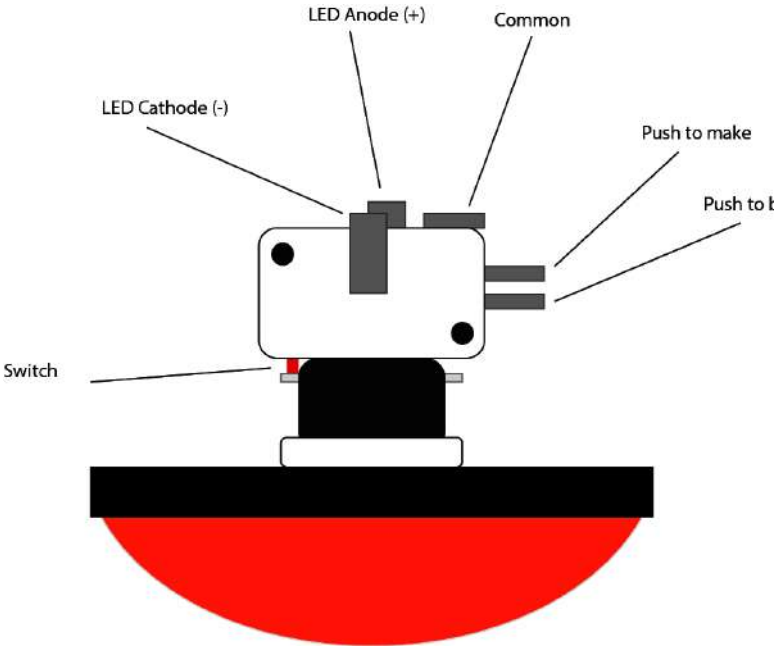
Planification prévisionnelle effectuée en début du projet confrontée à celle des tâches réelles réalisées.

N°	Mode Tâche	Nom de la tâche	Durée	Début	Fin
1		**Projet PMV**	**180 h?**	**Mer 05/01/22**	**Ven 03/06/22**
2		Lecture du dossier	3 h	Mer 05/01/22	Mer 05/01/22
3		Réunion PROJET 1	1 h	Jeu 06/01/22	Jeu 06/01/22
4		Diagramme de Gantt	3 h	Jeu 06/01/22	Jeu 06/01/22
5		Documentation sur principe du chronométrage dans les courses d'athlétisme	1 h	Ven 07/01/22	Ven 07/01/22
6		Documentation du buzzer	1 h	Ven 07/01/22	Ven 07/01/22
7		Test du buzzer	1 h	Ven 07/01/22	Ven 07/01/22
8		Test de la LED du bouton poussoir	1 h	Mer 12/01/22	Mer 12/01/22
9		Recherche d'un Buzzer 12V	1 h	Mer 12/01/22	Mer 12/01/22
10		Calcul d'une résistance pour une LED	3 h	Mer 12/01/22	Jeu 13/01/22
11		Test de l'enceinte JBL Flip Essential	7 h	Jeu 13/01/22	Mer 19/01/22
12		Test d'une LED sur Rpi	3 h	Mer 19/01/22	Jeu 20/01/22
13		Evaluer la différence de luminosité entre la LED du bouton poussoir et celle d'une autre LED	3 h	Jeu 20/01/22	Ven 21/01/22
14		Recherche d'un nouveau capteur pour William	2 h	Ven 21/01/22	Ven 21/01/22
15		Rédaction de la Revue de Projet N1	10 h	Mer 23/02/22	Ven 25/02/22
16		Test du Wi-Fi et de la LED à l'extérieur	4 h	Mer 02/03/22	Jeu 03/03/22
17		Test de plusieurs buzzer pour le départ des coureurs	4 h	Jeu 03/03/22	Ven 04/03/22
18		Réalisation Schéma Kicad PMV Personnel	37 h	Jeu 10/03/22	Ven 25/03/22
19		Réalisation Schéma Kicad PMV Commun	5 h	Mer 30/03/22	Jeu 31/03/22
20		Routage Carte Kicad	8 h	Jeu 31/03/22	Mer 06/04/22
21		Routage Kicad Commun (Suite&fin)	10 h	Jeu 07/04/22	Mer 27/04/22
22		Génération fichier Gerber	1 h	Jeu 28/04/22	Jeu 28/04/22
23		Nomemclature	11 h	Jeu 28/04/22	Jeu 05/05/22
24		Distribution des composants	4 h	Jeu 05/05/22	Ven 06/05/22
25		Soudure Carte Electronique	12 h	Ven 06/05/22	Mer 18/05/22
26		Proposer un nouveau schéma structurel de la synthèse d'ensemble	12 h	Mer 18/05/22	Jeu 02/06/22

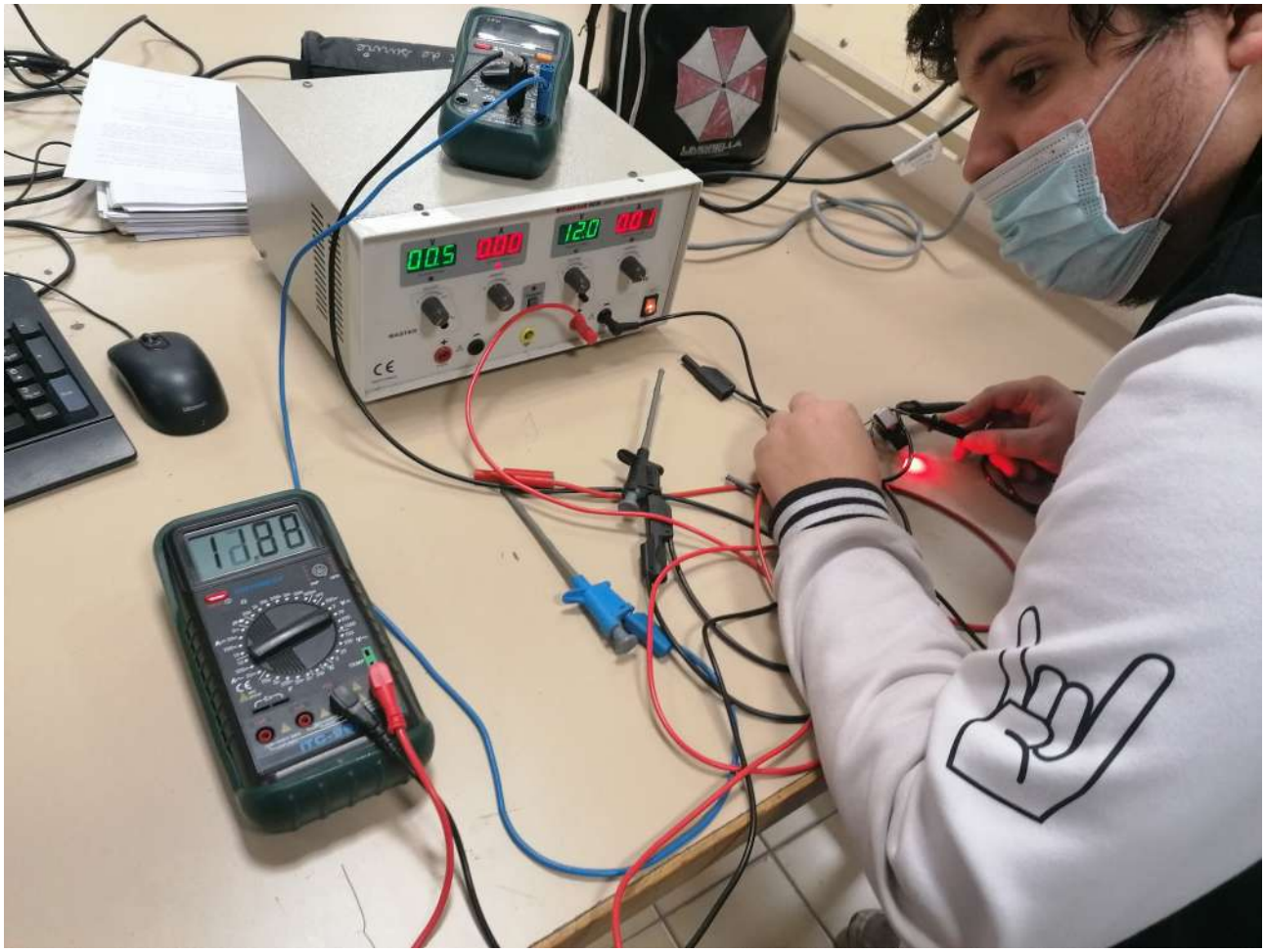


27	Essais et validation des structures (prototypage rapide)	12 h	Jeu 02/06/22	Jeu 09/06/22	
28	Fusion des 3 schémas	4 h	Jeu 09/06/22	Ven 10/06/22	
29	Reflexion sur la mise en boitier des composants	6 h	Mer 22/06/22	Jeu 23/06/22	
30	Prototypage rapide et routage	12 h	Jeu 23/06/22	Ven 01/07/22	
31	Fabrication et essais	15 h	Ven 01/07/22	Mer 13/07/22	
32	Fichier de fabrication PCB finalisés	10 h	Jeu 14/07/22	Mer 20/07/22	
33	**Evaluation de Projet**	**190 h?**	**Mer 05/01/22**	**Mer 15/06/22**	
34	Revue 1	0 h	Mer 02/03/22	Mer 02/03/22	
35	Revue 2	0 h	Mer 27/04/22	Mer 27/04/22	
36	Revue Finale	0 h	Mer 15/06/22	Mer 15/06/22	
37	Soutenance finale E62 (Revue 3)	0 h	Mer 15/06/22	Mer 15/06/22	
38	Soutenance E6.1	0 h	Mer 23/03/22	Mer 23/03/22	
39	Rédaction du dossier	7 h?	Mer 05/01/22	Jeu 06/01/22	

Tests



Pour commencer ma phase de tests, j'ai dû tester le matériel que l'on m'avait fourni, j'ai donc débuté par le bouton poussoir.



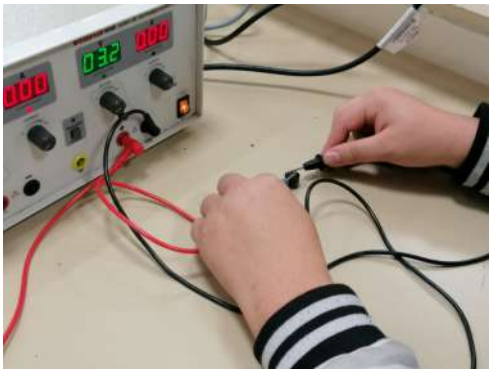
La première chose que j'ai dû vérifier, c'était de savoir où se trouve l'Anode (+) et la Cathode (-) de la LED pour la faire fonctionner. J'ai pu le savoir en faisant des recherches sur Internet, en recherchant le modèle du bouton.

Après avoir trouvé le + et le – de la LED, il ne me reste plus qu'à faire fonctionner la LED.

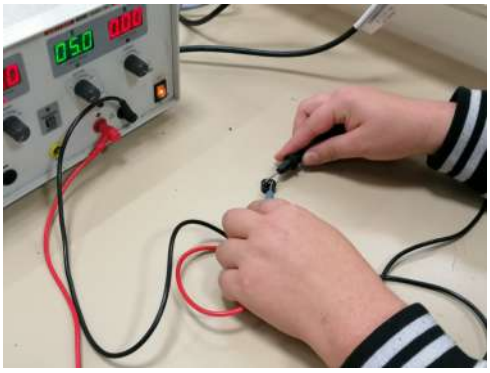
La LED fonctionne bien mais l'intensité de la lumière est très faible. Je ne pense pas que cette LED soit adéquate pour être vue en plein jour.

Après le test de la LED, j'ai pu tester le second composant que l'on m'a confié : le buzzer.

Bien sûr, comme pour la LED du bouton poussoir, j'ai dû chercher sur Internet où se trouve le + et le – du buzzer. J'ai d'abord voulu le tester sous 2 voltages, comme déjà indiqué précédemment la doc était déjà fournie et le signe + est gravé sur le composant.



Test en 3,3 Volts



Test en 5 Volts

Le buzzer testé n'émet pas assez de bruit pour pouvoir l'entendre à 50m. J'ai vu qu'il y avait un plastique jaune qui bloque le son je l'ai enlevé, j'ai remarqué que c'était légèrement mieux mais pas assez puissant pour émettre à 50m. Après, j'ai essayé de faire une corne avec une feuille pour teste, mais il n'y a pas vraiment de changement au niveau du son.

Mon avis est de trouver une alternative au buzzer, par exemple : une corne, un klaxon électrique ou un autre buzzer mais en 12V. J'ai donc dû chercher sur des sites Internet de fabriquant comme "RS" ou "Farnell" pour trouver ce que je cherche.

Après plusieurs recherches sur RS et Farnell, j'ai trouvé des Buzzer en 12V sur Farnell, on a fini par prendre celui-ci : le "Transducteur Piezzo MP000293".

Mais en regardant la doc du buzzer, le premier qu'on m'a fourni, j'ai remarqué qu'il pouvait déjà être alimenté sur 12V voire plus, j'ai fait un nouveau test et j'ai très vite entendu que le son n'était toujours pas suffisant pour pouvoir l'entendre correctement dehors.

En attendant que le buzzer que l'on a commandé sur Farnell arrive, j'ai voulu trouver une alternative à mon buzzer. J'ai vu dans le matériel que l'on pouvait avoir à disposition une enceinte Bluetooth "JBL Flip Essential" je me suis dit que ça pouvait être une bonne idée de tester et de voir si elle fait mieux que le buzzer. Le soucis, c'est que on ne l'a pas à

disposition, heureusement qu'un de nos collègues de classe a exactement le même modèle. Je lui ai donc demandé s'il pouvait nous la prêter le temps des tests.

Le but de ce test est de pouvoir la connecter sur une Raspberry puisque normalement, dans le cahier des charges, l'appareil qui doit émettre le son pour prévenir les coureurs doit être connecté sur un GPIO de la Raspberry, dans le cas de l'enceinte, elle sera juste connectée en Bluetooth pour le test et pour plus de facilité.

Avec l'aide du professeur et de mes collègues, j'ai tester cette enceinte connectée à ma Raspberry pour savoir à combien de mètres nous pouvions entendre le son et quand le son coupe.

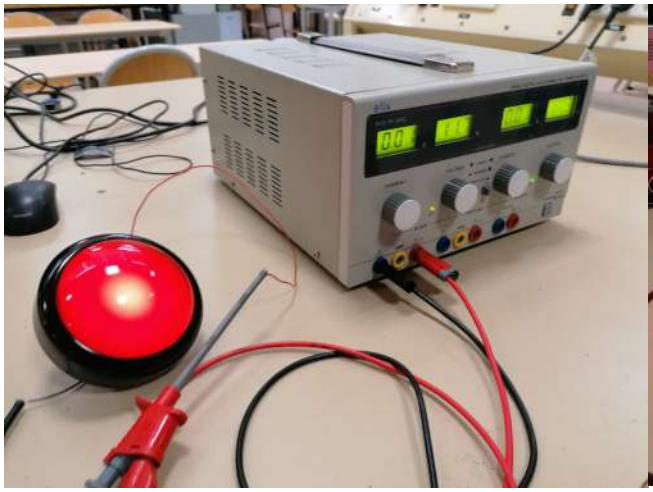
Pour faire le test, j'ai du connecter en Bluetooth l'enceinte à la Raspberry, j'ai ensuite dû trouver un son à envoyer dessus pour le test.

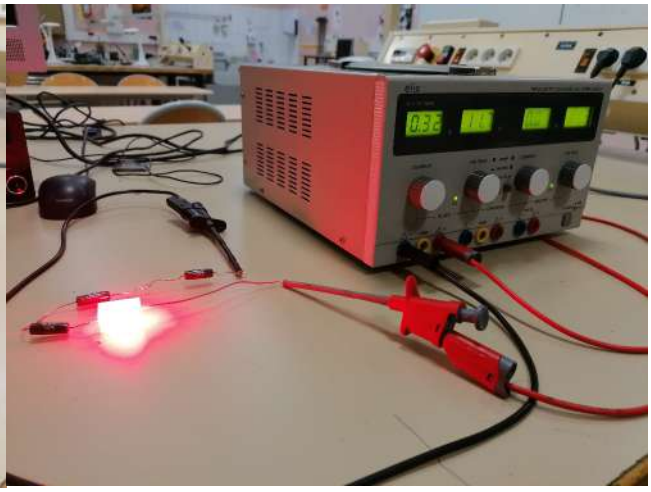
Ensuite, j'ai fait le test de portée dans le couloir d'à côté, mais on s'est rendu compte qu'à cause des murs de la salle de cours la portée était réduite, on a donc déplacé la Raspberry dans le couloir pour voir si on augmente la portée

On a vite remarqué que la portée a largement augmenté, je pense qu'on tenait largement les 50m.

J'ai même dû aller dehors pour voir si l'enceinte est toujours capable d'envoyer le son, et c'était le cas.

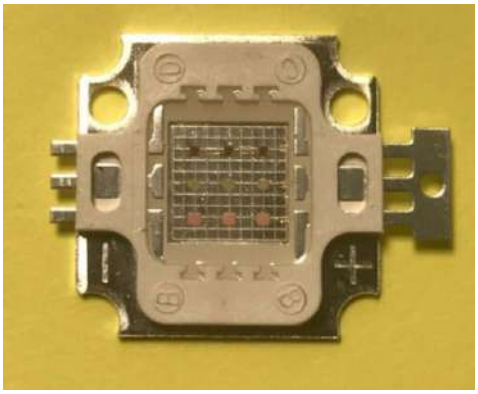
Mes tests sur l'enceinte Bluetooth sont très concluants mais je me suis rendu compte qu'il fallait trouver une autre LED car celle du bouton poussoir n'est pas assez forte. M. Hortolland m'a donné une nouvelle LED RGB de 3W pour voir si elle éclaire mieux que celle du bouton poussoir.





J'ai tout de suite remarqué que la nouvelle LED éclaire beaucoup mieux il n'y a rien à dire, j'ai même dû faire constater à M. Antoine la différence entre les deux LED.

1 semaine après la comparaison des deux LED, M. Hortolland m'a montré la future LED que nous allons mettre en œuvre plus tard, une "LED RGB 10W".



Avec M. Hortolland et M. Antoine, on a voulu faire des tests de lumière et de portée du WI-FI en champ libre dehors, mais avant, M. Hortolland m'a demandé de prendre des connecteurs et de rajouter des fils dessus pour pouvoir les connecter sur une batterie portable. Ensuite nous sommes allé dehors, nous avons tout mis en place pour les tests.

On l'a positionné au milieu du chemin avec la batterie à côté, avec mes collègues, nous nous sommes placés à environ 50m pour estimer si on distingue bien la LED

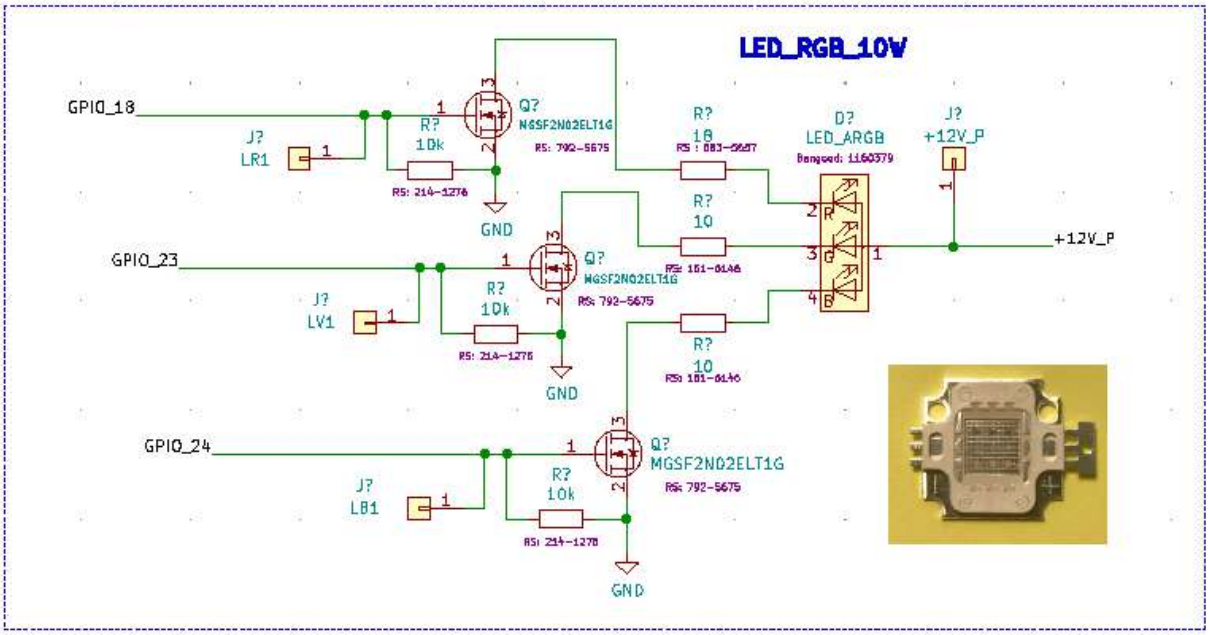
Je pense qu'avec cette LED il y a largement ce qu'il faut pour bien la distinguer sur 50m

Schéma et routage KiCad

La 1ère revue de projet étant passée, il fallait commencer le schéma structurel de notre future carte électronique sur KiCad, plus précisément, nous devions réaliser un "HAT" pour une Raspberry qu'on connectera via les 20 premières broches de la Raspberry.

Comme nous sommes 3 dans le projet et que nous avons tous une partie différente, nous avons fait chacun une partie du schéma structurel qui correspond à notre engagement.

Je m'occupe de la signalisation donc j'ai du faire un schéma qui regroupe les ports pour les GPIO de la Raspberry, faire une structure pour commander les LED et le buzzer.



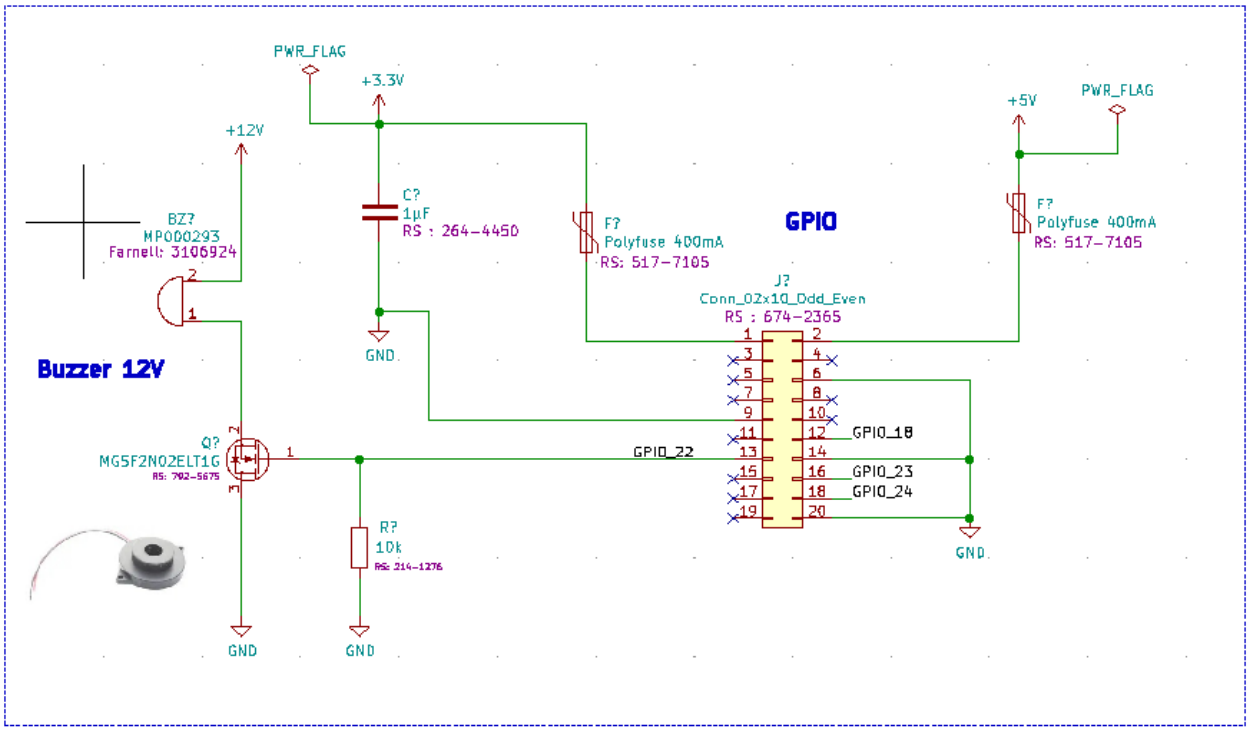
Pour la partie des LED, je me suis inspiré du schéma structurel du projet "RushBall" fait par les anciens 2nde année, j'ai informé mes collèges de mon choix de GPIO choisis pour piloter les LED. Nous avons choisis ces transistors pour les LED car ils supportent l'intensité qui les traverse, sur la doc du transistor, la Résistance Rdson (résistance entre drain et source à l'état passant "on") est de 85 mohm, pour savoir si le transistor supporte l'intensité qui le traverse, on fait une loi des mailles donc on fait la formule de la loi des mailles :

$$12 = V_d + 18 * I + R_{dson}$$

$$I = 12 - 6.5 - \frac{0.085}{18} = 300\,mA$$

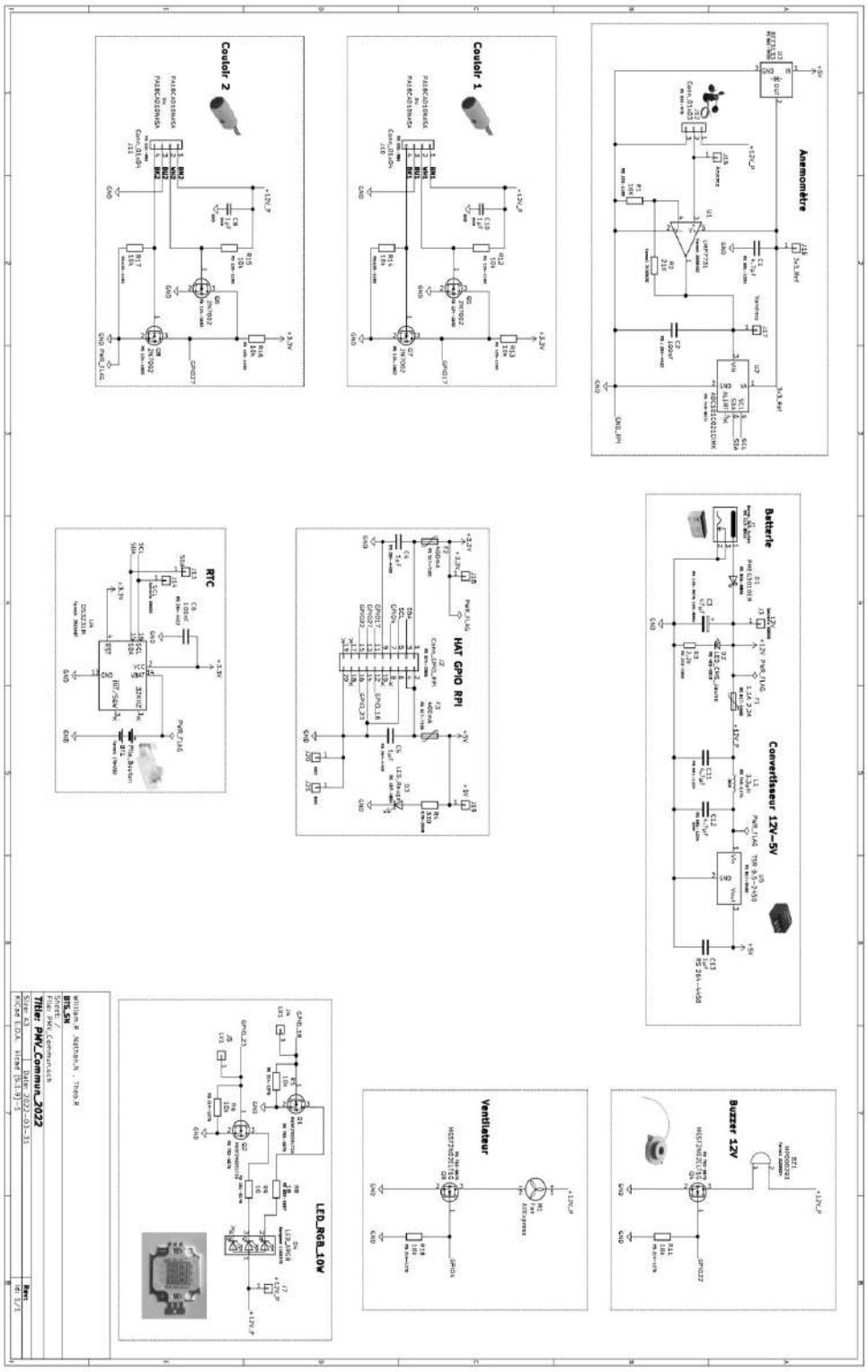
Si on prend la documentation de la LED on voit que son intensité est de 300mA c'est donc correct. Il faut aussi calculer la puissance dissipée par la résistance il faut donc faire

P= Rdson x Id (courant de drain)² = 85×10^{-3} x $2{,}8^2$ = 0,6664 W on voit donc qu'il n'y a pas besoin d'un dissipateur pour dissiper la chaleur du transistor.

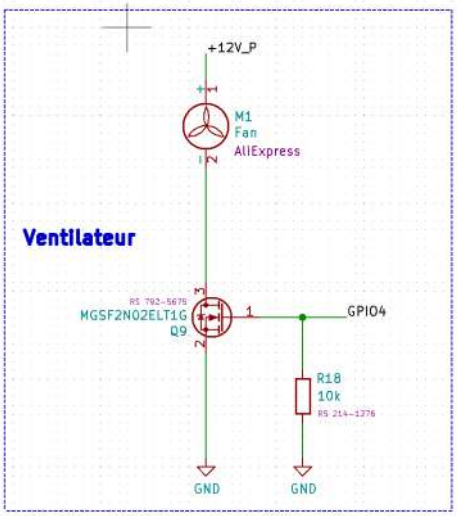


Pour la partie du buzzer et des broches du GPIO, je me suis inspiré d'un TP d'exemple pour m'aider à faire le schéma du GPIO. Pour le buzzer, j'ai eu un peu d'aide de la part de M. Hortolland en me disant que prendre le même transistor que celui des LED et de le connecter au GPIO que j'avais choisis d'utiliser.

Une fois que tout le monde avait réussi à faire sa partie du schéma, et avec l'approbation du professeur, nous devions fusionner nos schémas et ne former qu'un seul et grand schéma avec toutes les parties de notre carte.

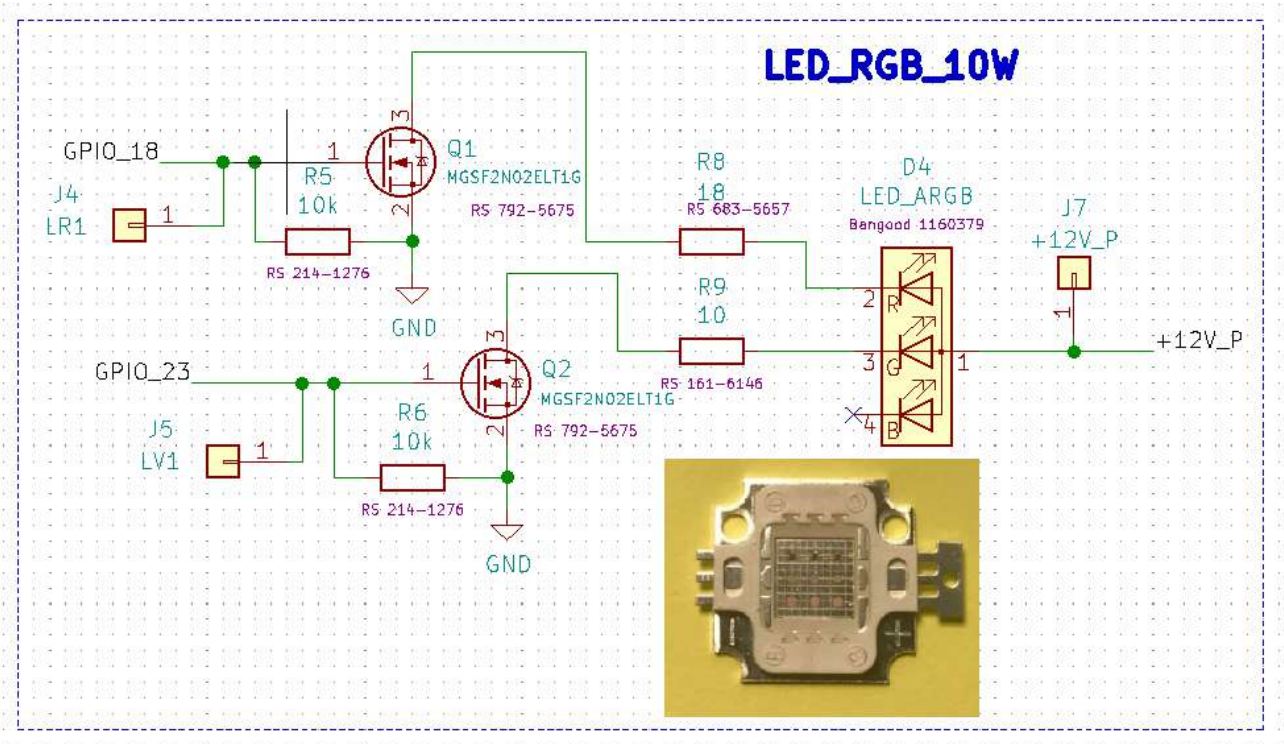

Anemomètre
Couloir 1
Couloir 2
Batterie
Convertisseur 12V-5V
HAT GPIO RPI
RTC
Buzzer 12V
Ventilateur
LED_RGB_10W
BTS_SN
Titre: PMV_Commun_2023

Bien sûr, toutes les parties n'étaient pas parfaites et il y avait toujours une possibilité de faire des améliorations, pour ma partie il n'y avait pas grand-chose à modifier mais on pouvait rajouter des choses ou supprimer des composants qui ne servaient pas.



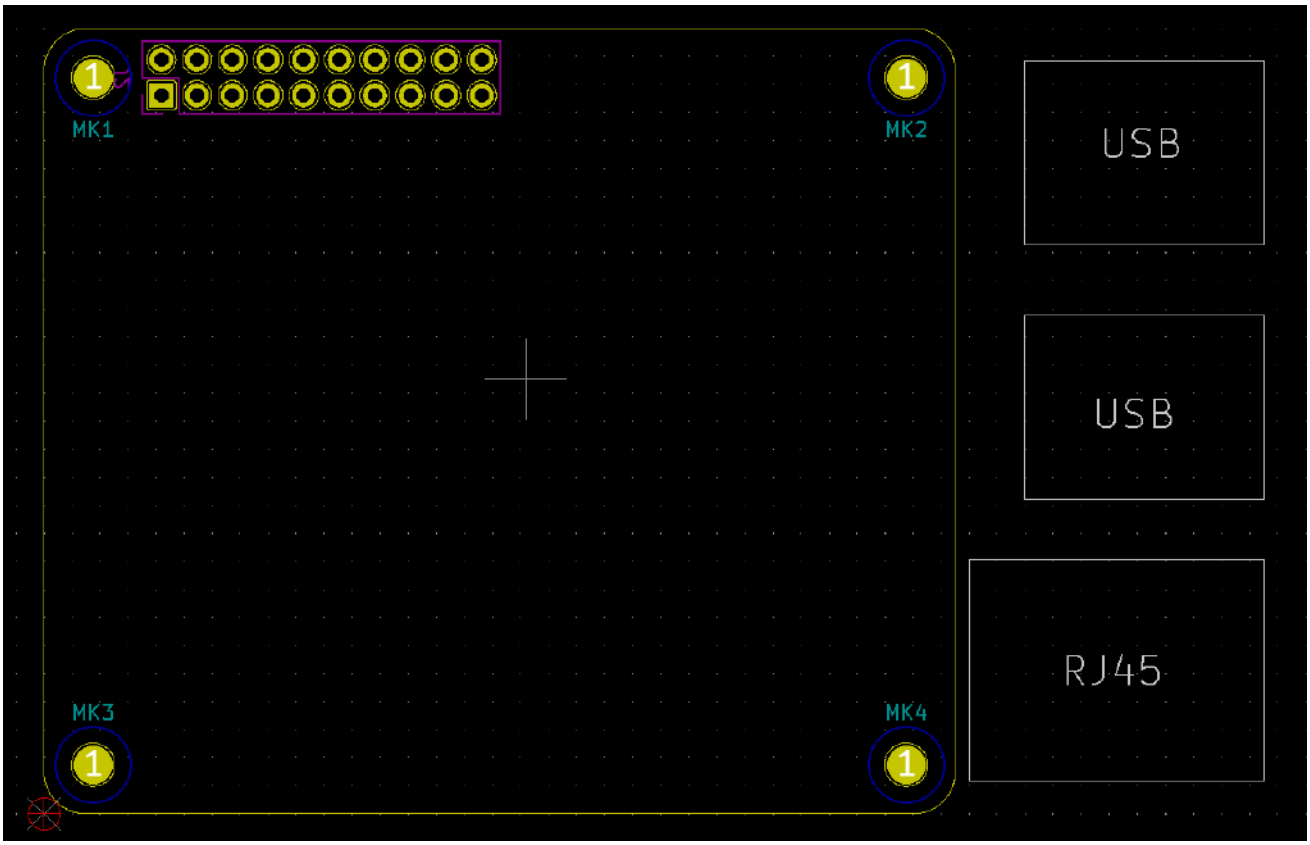
Sur ma partie, nous avons rajouté un ventilateur, commandé par le GPIO de la Raspberry, vu que principalement, la LED sera dehors sous le soleil et en plus la LED qui chauffe, ce n'est pas bon pour le composant, donc un ventilateur pour refroidir la LED.

Nous avons aussi sur ma partie, au niveau de la LED, retiré une couleur qui ne servira pas : le bleu. M. Hortolland nous a expliqué que ça ne servait à rien de laisser les 3 couleurs RVB car juste 2 seront utilisées, à la base, il voulait garder le rouge (couleur par défaut dans le cahier des charges) et bleu mais je me suis dit que ce n'était pas logique car les coureurs vont voir la couleur bleue ne vont pas comprendre alors que si on garde le vert, ils sauront que c'est le moment de partir, alors nous nous sommes mis d'accord sur le rouge et le vert.



Une fois les parties de tout le monde corrigées sur le schéma commun de la carte, il était temps de faire la partie "routage de la carte", comme nous sommes 3, nous devons faire la même carte mais faite un peu différemment des autres, Le contour du HAT étant

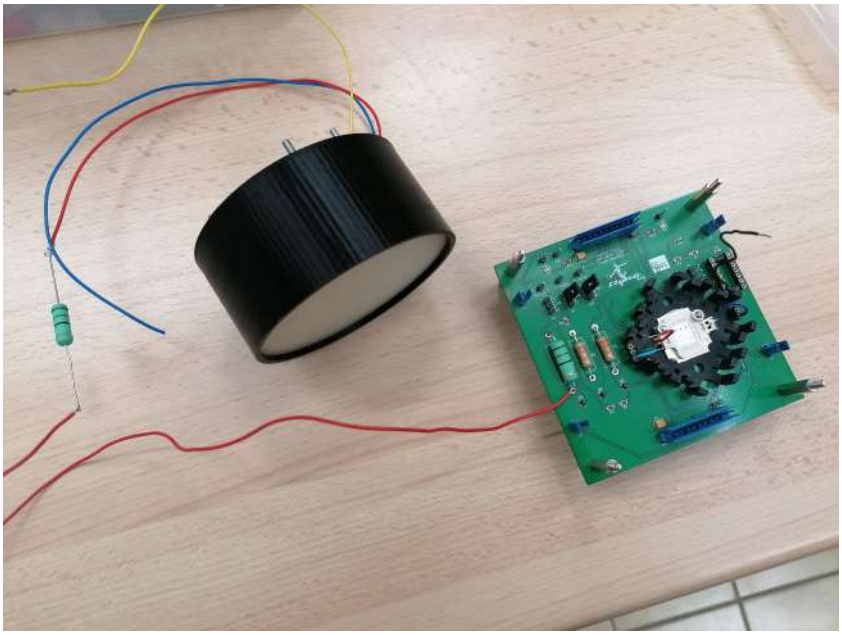
proposé sur KiCad, il a suffit de l'adapter pour notre application, chose faite par Mr hortolland.



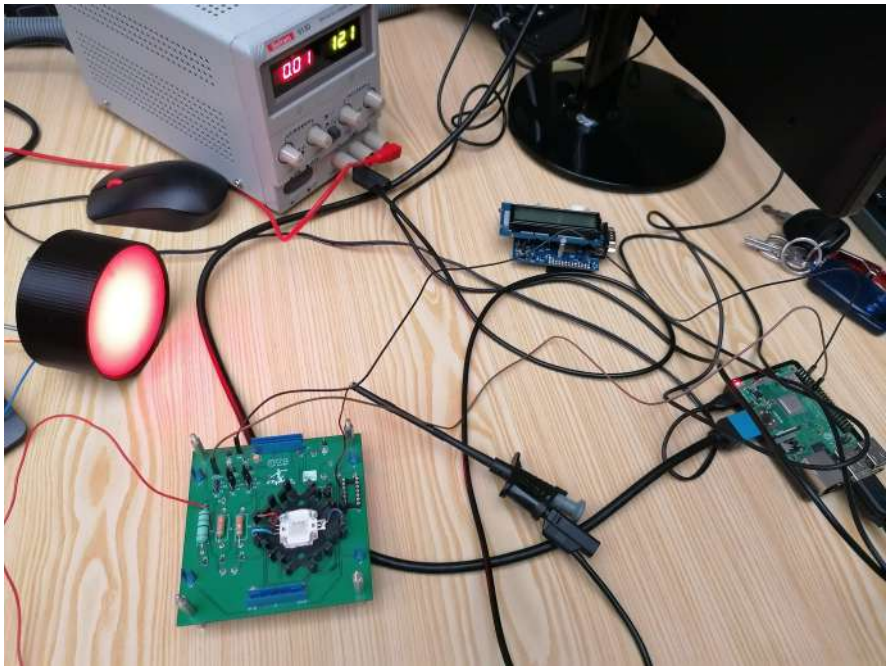
Le routage de la carte est un peu long et assez complexe, le but étant de placer tous les composants choisis dans le schéma structurel sur le modèle de la carte.

Le plus compliqué étant de placer tous les composants sur la carte et de câbler chaque composant en faisant attention de ne pas encercler les composants et donc ne plus pouvoir câbler.

Pendant que je faisais le routage, un élève du groupe IR du projet PMV a voulu faire des tests avec ma LED pour son application pour programmer le départ de la course. Avant de faire le test dans la salle d'IR, on a mis le fil du « - » de la LED sur la résistance reliée sur un transistor d'une carte servant pour faire marcher une autre LED



Une fois la tache effectuée, je suis allé faire le test dans la salle d'IR, on a connecté la LED sur le GPIO du Raspberry que j'ai choisis et connecter le GND de la LED sur une des broches de GND de la Raspberry

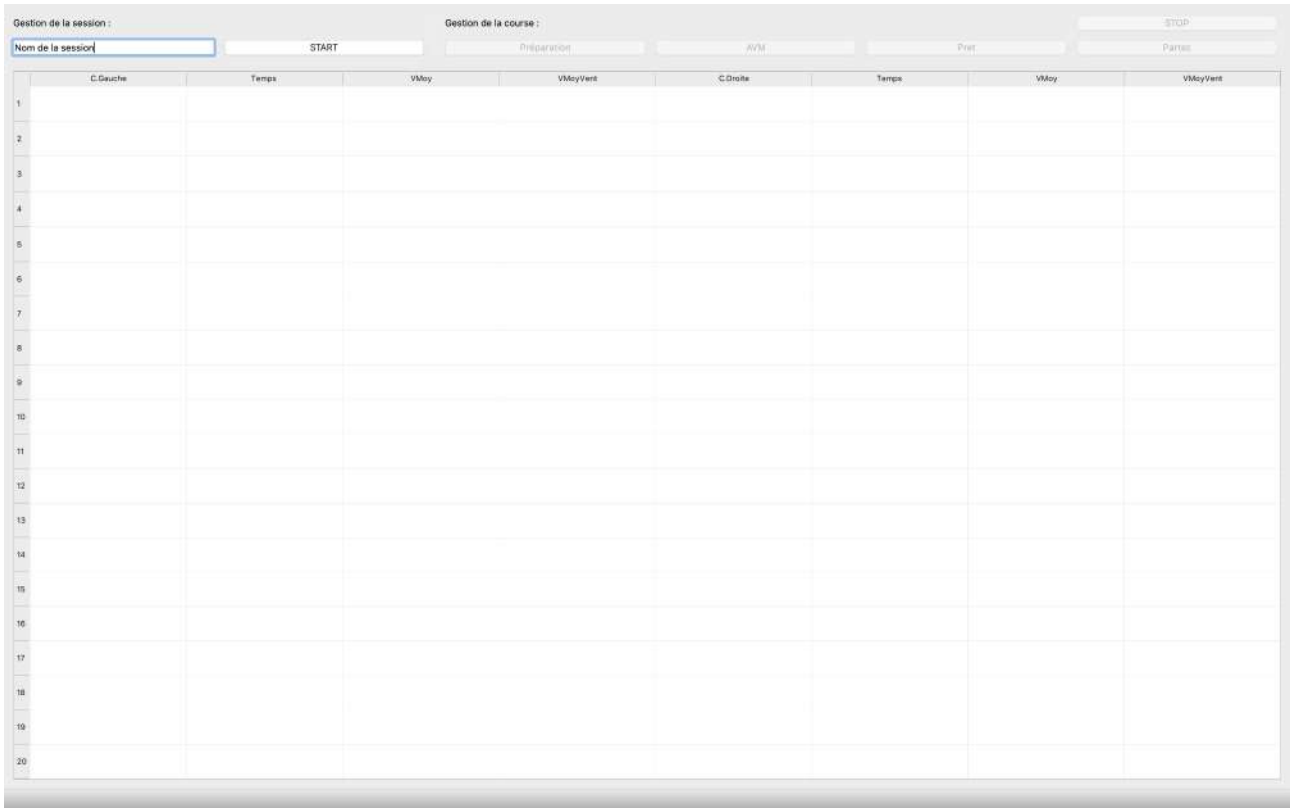


En faisant les tests, on s'est rendu compte que la LED s'allume correctement mais il y a un faux contact, on a vérifié avec un multimètre pour voir si le courant passe bien comme il faut ou si même en ne mettant pas la LED, le courant passe quand même, on a vu que le même en ne mettant pas la LED, le courant passe malgré tout alors que ce n'est pas

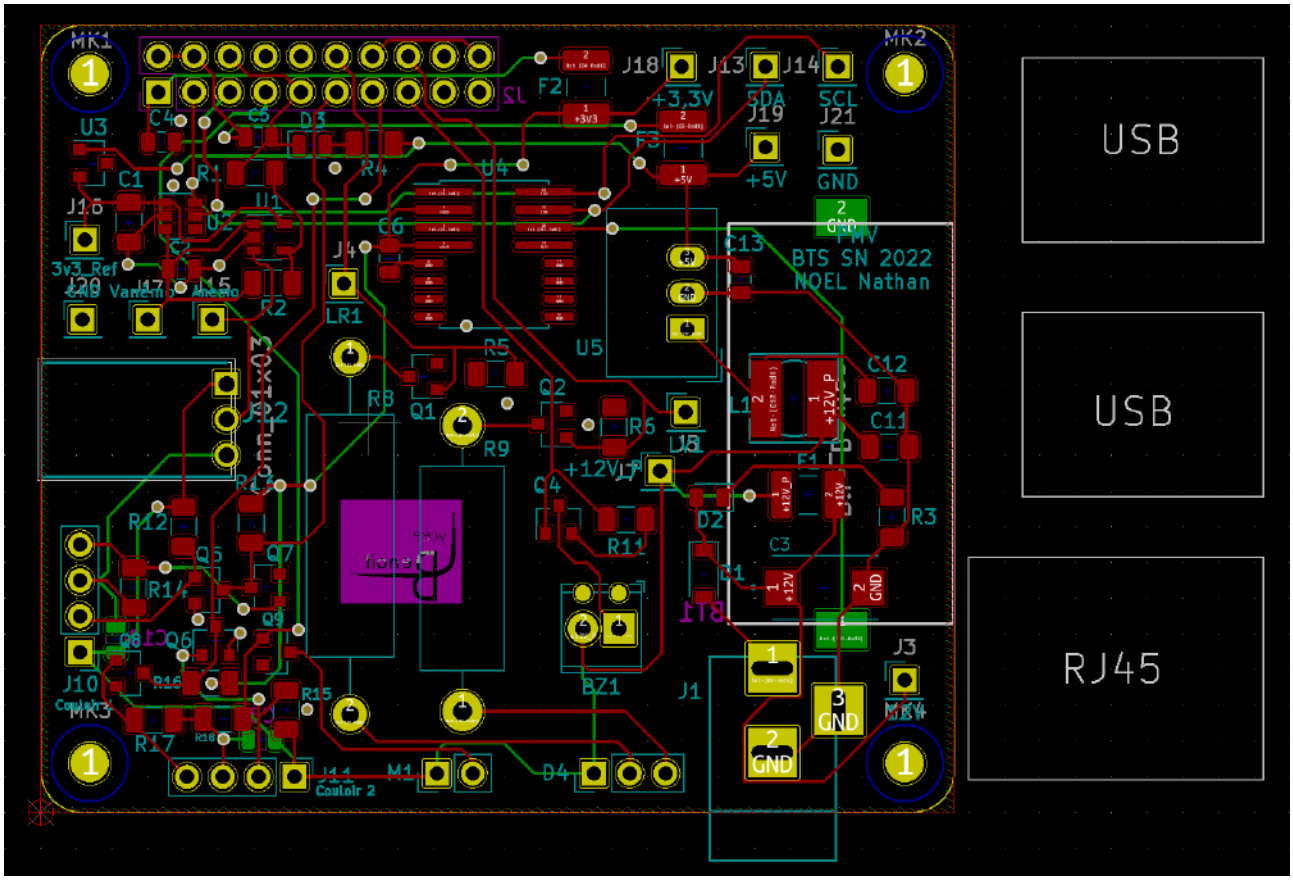
normal vu que la LED est connecter sur la résistance reliée au transistor MOS qui est censé bloquer le courant à une certaine tension.

Finalement, on a essayé de connecter la LED sur la résistance d'à côté pour essayer et là, le courant passe comme on le souhaite, on a diagnostiquer que le transistor précédant est défectueux.

Maintenant que ce problème est résolu, le test de l'application peut commencer, pour démarrer le test il faut appuyer sur « AVM (A vos marques) » cliquer ensuite sur « Prêt » et pour finir, cliquer sur « Partez » pour lancer le chrono, l'application peut stopper le chrono au bout de 5 secondes après avoir lancer la course.

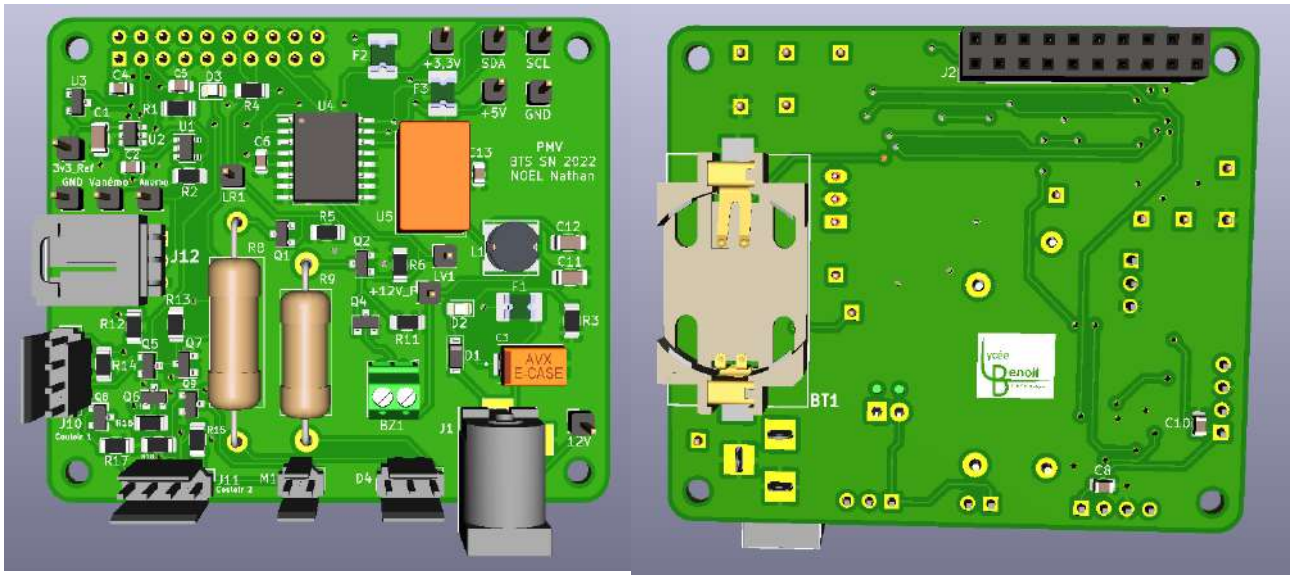


Après avoir fini le test de la LED sur la Raspberry, j'ai continué à faire le routage, après deux semaine de dur labeur la carte est enfin fini.

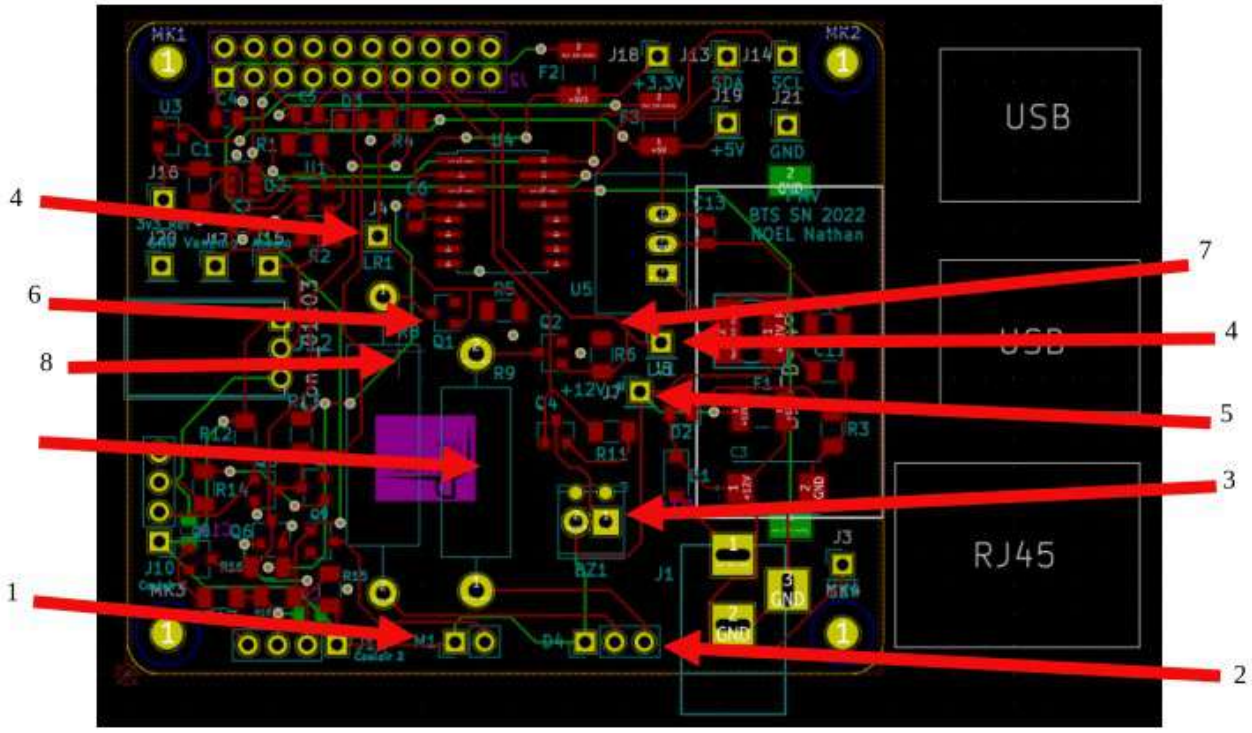


Pendant nos vacances je l'ai avancer de mon côté pour ne pas prendre de retard, les problème que j'ai eu pendant que je faisais le routage est que je me retrouvais souvent coincé à cause des fils rouge qui encercle les connecteurs des composants et qui du coup empêche de câbler ou de faire passer certain fils, un autre problème, c'était le placement des composants, en fonction de la ou j'ai décidé de les mettre, cela pouvait influer sur le passage des fils et donc me bloquer certains passages.

Voici le résultat de la carte en visualisation 3D (de face et de dos)



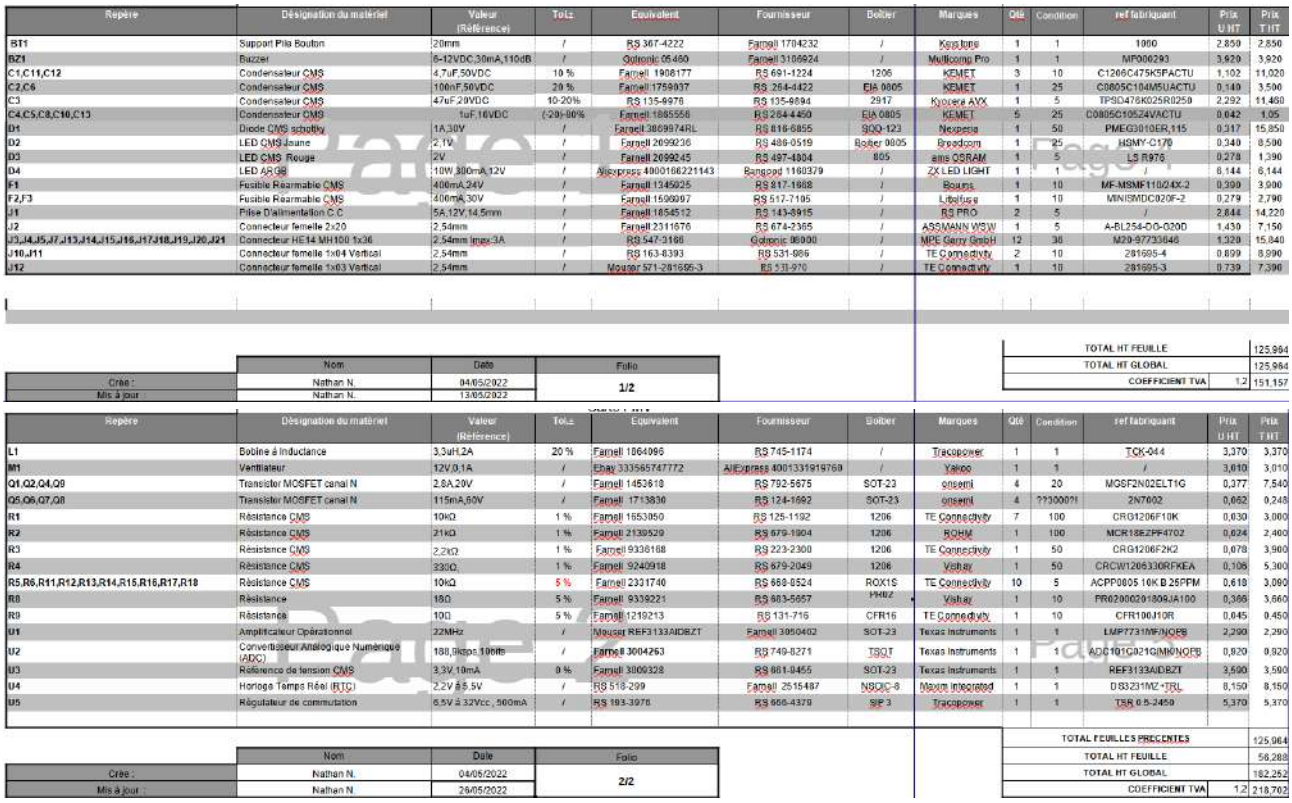
Analyse de ma partie sur la carte



1. **"M1"** : Connecteur 2 broches pour connecter le ventilateur de la LED
2. **"D4"** : Connecteur 3 broches pour connecter la LED RGB 10W
3. **"BZ1"** : Bornier 2 contact pour connecter le buzzer servant au signal sonore
4. **"J4,J5"** : Points de test des LED rouge et vert
5. **"J7"**: Point de test pour le 12V pour y piquer un multimètre
6. **"Q1, Q2, Q4, Q9"** : Transistor MOS canal N pour faire passer le courant à un niveau logique haut (1)
7. **"R5, R6, R11, R18"** : Résistances 10kΩ
8. **"R8"** : Résistance 18Ω
9. **"R9"** : Résistance 10Ω

Soudage de la carte

La deuxième revue de projet est passée, la carte est déjà faite mais on doit attendre que notre PCB arrive , apparemment la carte peut mettre du temps à arriver. En attendant que l'on reçoit le PCB, Mr Hortolland nous a demander de faire la liste de matériel (ou Nomenclature) qui nous liste tout les composants utilisé à la conception de la carte, on a pu récupérer un modèle d'une nomenclature sur le site du projet, avec les 2 collègues on a du récupérer chaque détail de chaque composant présent sur la carte (référence fabriquant, boîtier, valeur, tolérance, fournisseur, etc.)



Repère	Désignation du matériel	Valeur (Référence)	Tolé	Equivalent	Fournisseur	Boîtier	Marques	Qté	Condition	ref fabriquant	Prix U HT	Prix T HT
BT1	Support Pile Bouton	20mm	/	RS 367-4222	Farnell 1704232	/	Keystone	1	1	1060	2,850	2,850
BZ1	Buzzer	6-12VDC,30mA,110dB	/	Gotronic 05460	Farnell 3106924	/	Multicomp Pro	1	1	MP000293	3,920	3,920
C1,C11,C12	Condensateur CMS	4,7uF,50VDC	10 %	Farnell 1908177	RS 691-1224	1206	KEMET	3	10	C1206C475K5PACTU	1,102	11,020
C2,C6	Condensateur CMS	100nF,50VDC	20 %	Farnell 1759037	RS 264-4422	EIA 0805	KEMET	1	25	C0805C104M5UACTU	0,140	3,500
C3	Condensateur CMS	47uF,20VDC	10-20%	RS 135-6976	RS 135-6894	2917	Kyocera AVX	1	5	TPSD476K025R0250	2,292	11,460
C4,C5,C8,C10,C13	Condensateur CMS	1uF,16VDC	(-20)-80%	Farnell 1865556	RS 2844450	EIA 0805	KEMET	5	25	C0805C105Z4VACTU	0,042	1,05
D1	Diode CMS schottky	1A,30V	/	Farnell 3809974RL	RS 816-6855	SOD-123	Nexperia	1	50	PMEG3010ER,115	0,317	15,850
D2	LED CMS Jaune	2,1V	/	Farnell 2699236	RS 486-0519	Boîtier 0805	Broadcom	1	25	HSMY-C170	0,340	8,500
D3	LED CMS Rouge	2V	/	Farnell 2699245	RS 497-4004	805	ams OSRAM	1	5	LS R976	0,278	1,390
D4	LED ARGB	10W,300mA,12V	/	Aliexpress 4000186221143	Bangood 1160379	/	ZX LED LIGHT	1	1	/	6,144	6,144
F1	Fusible Rearmable CMS	400mA,24V	/	Farnell 1345025	RS 817-1668	/	Bourns	1	10	MF-MSMF110/24X-2	0,390	3,900
F2,F3	Fusible Rearmable CMS	400mA,30V	/	Farnell 1599097	RS 517-7105	/	Littelfuse	1	10	MINISMDC020F-2	0,279	2,790
J1	Prise D'alimentation C.C	5A,12V,14.5mm	/	Farnell 1854512	RS 143-8915	/	RS PRO	2	5	/	2,844	14,220
J2	Connecteur femelle 2x20	2,54mm	/	Farnell 2311676	RS 674-2365	/	ASSMANN WSW	1	5	A-BL254-DO-020D	1,430	7,150
J3,J4,J5,J7,J13,J14,J15,J16,J17,J18,J19,J20,J21	Connecteur HE14 MH100 1x36	2,54mm Imax 3A	/	RS 547-3166	Gotronic 06900	/	MPE Garry GmbH	12	36	M20-9773646	1,320	15,840
J10,J11	Connecteur femelle 1x04 Vertical	2,54mm	/	RS 163-8393	RS 531-986	/	TE Connectivity	2	10	281695-4	0,899	8,990
J12	Connecteur femelle 1x03 Vertical	2,54mm	/	Mouser 571-281695-3	RS 531-970	/	TE Connectivity	1	10	281695-3	0,739	7,390

	Nom	Date	Folio
Crée :	Nathan N.	04/05/2022	1/2
Mis à jour :	Nathan N.	13/05/2022	

TOTAL HT FEUILLE		125,964
TOTAL HT GLOBAL		125,964
COEFFICIENT TVA	1,2	151,157

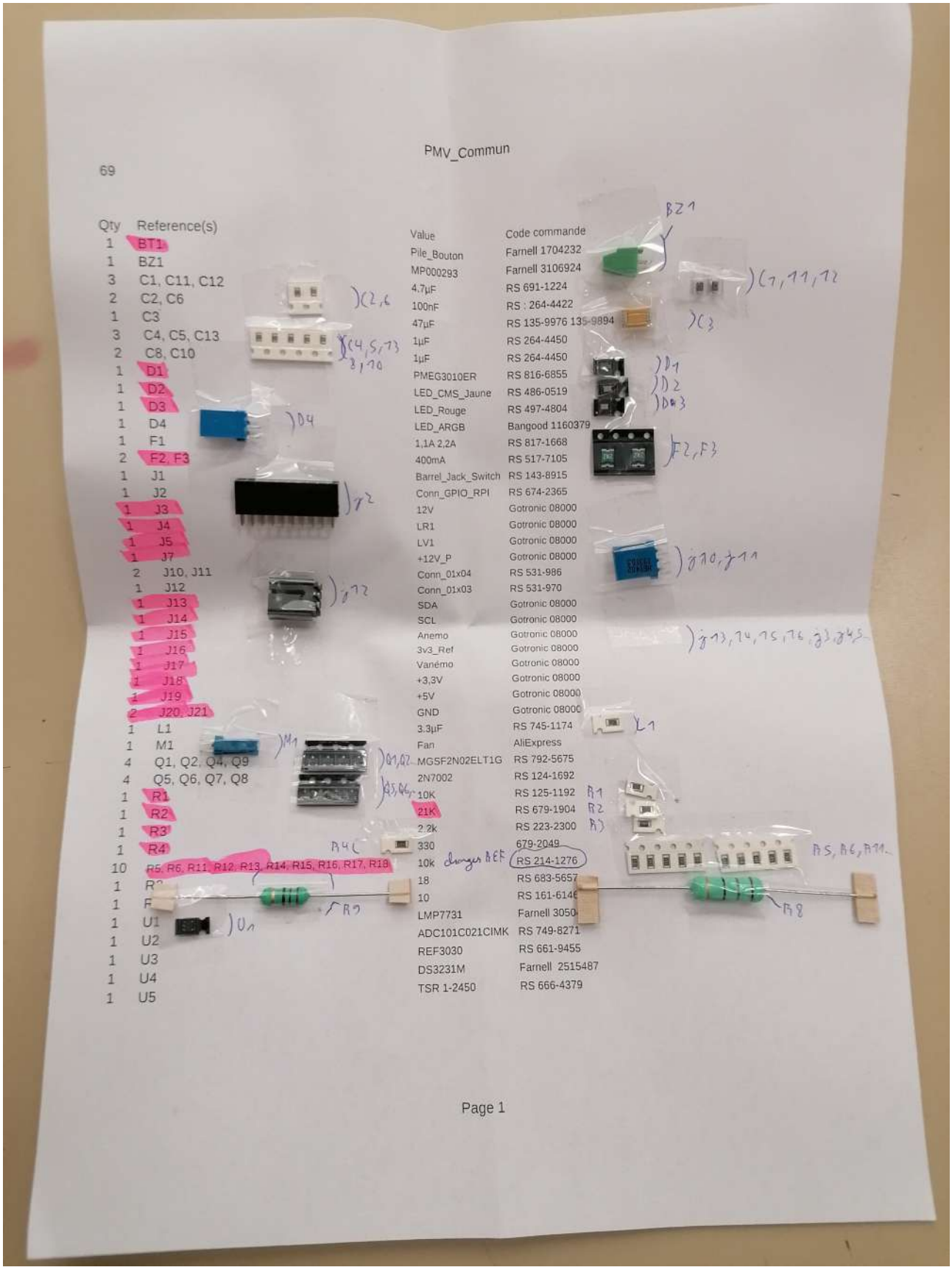
Repère	Désignation du matériel	Valeur (Référence)	Tolé	Equivalent	Fournisseur	Boîtier	Marques	Qté	Condition	ref fabriquant	Prix U HT	Prix T HT
L1	Bobine à inductance	3,3uH,2A	20 %	Farnell 1864096	RS 745-1174	/	Tracopower	1	1	TCK-044	3,370	3,370
M1	Ventilateur	12V,0,1A	/	Ebay 333565747772	AliExpress 4001331919769	/	Yaeco	1	1	/	3,010	3,010
Q1,Q2,Q4,Q9	Transistor MOSFET canal N	2,8A,20V	/	Farnell 1453618	RS 792-5675	SOT-23	onsemi	4	20	MGSF2N02ELT1G	0,377	7,540
Q5,Q6,Q7,Q8	Transistor MOSFET canal N	115mA,60V	/	Farnell 1713830	RS 124-1692	SOT-23	onsemi	4	??3000?!	2N7002	0,062	0,248
R1	Résistance CMS	10kΩ	1 %	Farnell 1653050	RS 125-1192	1206	TE Connectivity	7	100	CRG1206F10K	0,030	3,000
R2	Résistance CMS	21kΩ	1 %	Farnell 2130529	RS 679-1904	1206	ROHM	1	100	MCR18EZPF4702	0,024	2,400
R3	Résistance CMS	2,2kΩ	1 %	Farnell 9336168	RS 223-2300	1206	TE Connectivity	1	50	CRG1206F2K2	0,078	3,900
R4	Résistance CMS	330Ω	1 %	Farnell 9240918	RS 679-2049	1206	Vishay	1	50	CRCW1206330RFKEA	0,106	5,300
R5,R6,R11,R12,R13,R14,R15,R16,R17,R18	Résistance CMS	10kΩ	5 %	Farnell 2331740	RS 668-0524	ROX1S	TE Connectivity	10	5	ACPP0805 10K B 25PPM	0,618	3,090
R8	Résistance	15Ω	5 %	Farnell 9339221	RS 863-5657	PMDZ	Vishay	1	10	PR02000201509JA100	0,366	3,660
R9	Résistance	10Ω	5 %	Farnell 1219213	RS 131-716	CFR16	TE Connectivity	1	10	CFR100J10R	0,045	0,450
U1	Amplificateur Opérationnel	22MHz	/	Mouser REF3133AIDBZT	Farnell 3050402	SOT-23	Texas Instruments	1	1	LMP7731MF/NOPB	2,290	2,290
U2	Convertisseur Analogique Numérique (ADC)	18Bits,ksps,10bits	/	Farnell 3004263	RS 749-8271	TSOT	Texas Instruments	1	1	ADC101C021CIMK/NOPB	0,920	0,920
U3	Référence de tension CMS	3,3V,10mA	0 %	Farnell 3009328	RS 861-0455	SOT-23	Texas Instruments	1	1	REF3133AIDBZT	3,590	3,590
U4	Horloge Temps Réel (RTC)	2,2V à 5,5V	/	RS 518-299	Farnell 2515487	NSOIC-8	Maxim Integrated	1	1	DS3231MZ+TRL	8,150	8,150
U5	Régulateur de commutation	6,5V à 32Vcc ; 500mA	/	RS 193-3976	RS 666-4379	SIP 3	Tracopower	1	1	TSR 0.5-2450	5,370	5,370

	Nom	Date	Folio
Crée :	Nathan N.	04/05/2022	2/2
Mis à jour :	Nathan N.	26/05/2022	

TOTAL FEUILLES PRECENTES		125,964
TOTAL HT FEUILLE		56,288
TOTAL HT GLOBAL		182,252
COEFFICIENT TVA	1,2	218,702

Cette étape est un peu longue car il faut tout rechercher au cas ou un composant n'est pas disponible sur un site ou qu'il y a une pénurie de composant comme c'est le cas actuellement

Pendant que nous étions en train de remplir la nomenclature, Mr Hortolland nous a demander de venir pour qu'il puisse nous donner les composants nécessaire à la construction de notre carte, il nous a donner une liste de composant à partir de celle que l'on pouvait récupérer sur Kicad et nous a demander de combien de composants, nous avions besoin pour la carte



PMV_Commun

69

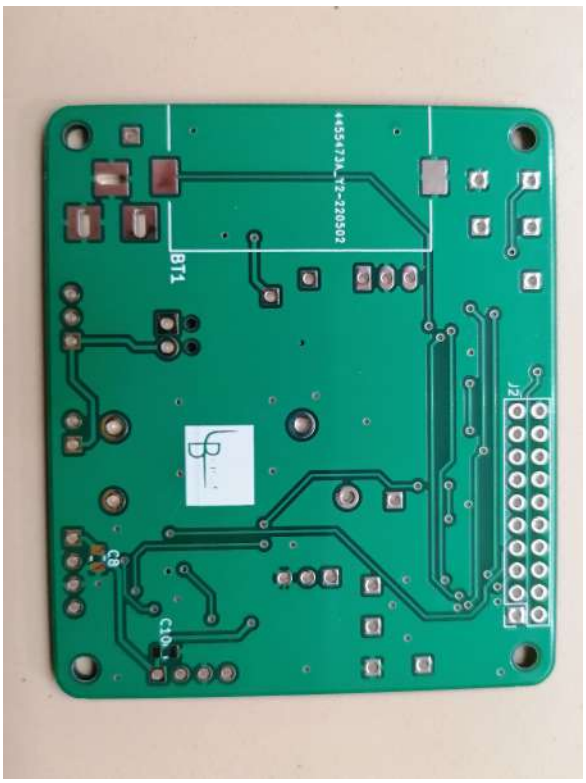
Qty	Reference(s)	Value	Code commande
1	BT1	Pile_Bouton	Farnell 1704232
1	BZ1	MP000293	Farnell 3106924
3	C1, C11, C12	4.7µF	RS 691-1224
2	C2, C6	100nF	RS : 264-4422
1	C3	47µF	RS 135-9976 135-9894
3	C4, C5, C13	1µF	RS 264-4450
2	C8, C10	1µF	RS 264-4450
1	D1	PMEG3010ER	RS 816-6855
1	D2	LED_CMS_Jaune	RS 486-0519
1	D3	LED_Rouge	RS 497-4804
1	D4	LED_ARGB	Bangood 1160379
1	F1	1,1A 2,2A	RS 817-1668
2	F2, F3	400mA	RS 517-7105
1	J1	Barrel_Jack_Switch	RS 143-8915
1	J2	Conn_GPIO_RPI	RS 674-2365
1	J3	12V	Gotronic 08000
1	J4	LR1	Gotronic 08000
1	J5	LV1	Gotronic 08000
1	J7	+12V_P	Gotronic 08000
2	J10, J11	Conn_01x04	RS 531-986
1	J12	Conn_01x03	RS 531-970
1	J13	SDA	Gotronic 08000
1	J14	SCL	Gotronic 08000
1	J15	Anemo	Gotronic 08000
1	J16	3v3_Ref	Gotronic 08000
1	J17	Vanémo	Gotronic 08000
1	J18	+3,3V	Gotronic 08000
1	J19	+5V	Gotronic 08000
2	J20, J21	GND	Gotronic 08000
1	L1	3.3µF	RS 745-1174
1	M1	Fan	AliExpress
4	Q1, Q2, Q4, Q9	MGSF2N02ELT1G	RS 792-5675
4	Q5, Q6, Q7, Q8	2N7002	RS 124-1692
1	R1	10K	RS 125-1192
1	R2	21K	RS 679-1904
1	R3	2.2k	RS 223-2300
1	R4	330	679-2049
10	R5, R6, R11, R12, R13, R14, R15, R16, R17, R18	10k	RS 214-1276
1	R?	18	RS 683-5657
1	R	10	RS 161-614
1	U1	LMP7731	Farnell 3050
1	U2	ADC101C021CIMK	RS 749-8271
1	U3	REF3030	RS 661-9455
1	U4	DS3231M	Farnell 2515487
1	U5	TSR 1-2450	RS 666-4379

Page 1

A cause de la pénurie de composant, Mr Hortolland n'a pas pu nous donner tout les composants mais nous a quand même donner les plus importants, en attendant, moi et mes collègues devions finir la nomenclature.

Au début du projet, on nous a prévenu que le budget serait de 300€, d'après la nomenclature finalisé l'estimation du coût des composants à acheter serait de 218€ donc le budget est bien respecter

Un jour plus tard, les cartes étaient enfin arrivé.



Comme nous avons 3 cartes mais qu'il fallait prendre en compte que les autres projet devaient souder eux aussi leur cartes, nous devions passer chacun notre tour pour souder notre carte.

Avant tout chose, comme nous avons des composants en CMS, on ne peux pas les souder sur la carte avec de l'étain car ce ne serait pas propre et on risque de crée un court-circuit, il faut donc appliquer de la pâte à braser sur chaque extrémité des composants

On va donc se servir d'un Stencil, qui est une petite feuille imprimé en 3d qui représente les contours des extrémités des composants a souder, avec ce stencil, on va enduire de pâte à braser les empreintes des composants

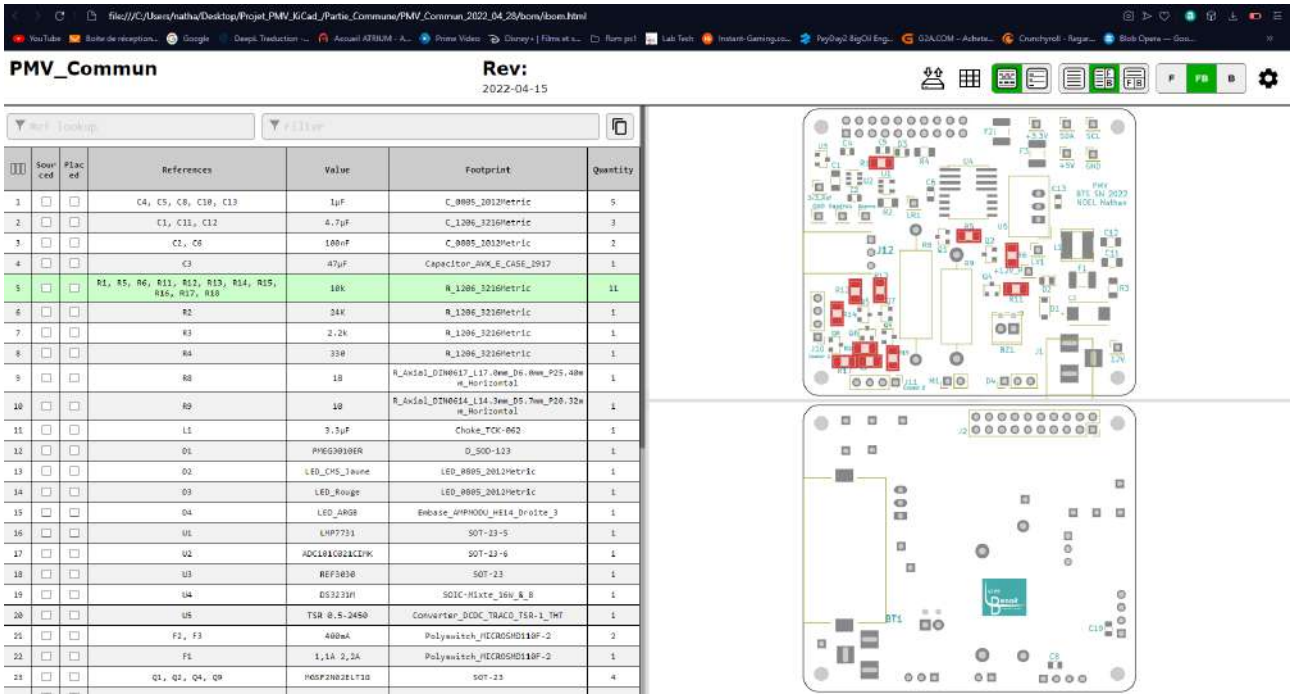
Pour bien stabiliser le Stencil, on s'est servie d'un moule fait en imprimante 3D pour caler la carte de façon à ne ce qu'elle ne bouge pas et on viens mettre le Stencil dessus qu'on bloque aussi avec des aimants pour que l'on puisse appliquer correctement la pâte à braser

Après que la pâte à braser est bien étaler sur tout les composants, il est temps de placer les composants en CMS, un par un, sur la carte. Pour m'aider a placer les composants, Mr Hortolland nous a montrer un plugin sur Kicad qui nous montre les emplacements exacte des composants à placer sur la carte, pour installer le plugin, il faut télécharger le plugin grâce à ce lien : https://github.com/openscopeproject/InteractiveHtmlBom

Ensuite, il faut suivre ce chemin de dossier specifique :

C:\Users\[USERNAME]\AppData\Roaming\kicad\scripting\plugins

Si dans le dossier « kicad » les dossiers « scripting » et « plugins » n'existe pas, il faut les créer à la main, dès que cela est fait, mettre le fichier du plugin dans le dossier « plugins » et ouvrir Kicad. Si l'installation est bien fait, dans le logiciel « Kicad » , sur la partie « Routage » dans l'onglet « outil » on peut voir dans la section « Plugins externes » une fonction appelée « Generate Interative HTML BOM » ce qui va permettre de génerer un plan interactif de la carte qui peut faciliter l'identification du placement des composants



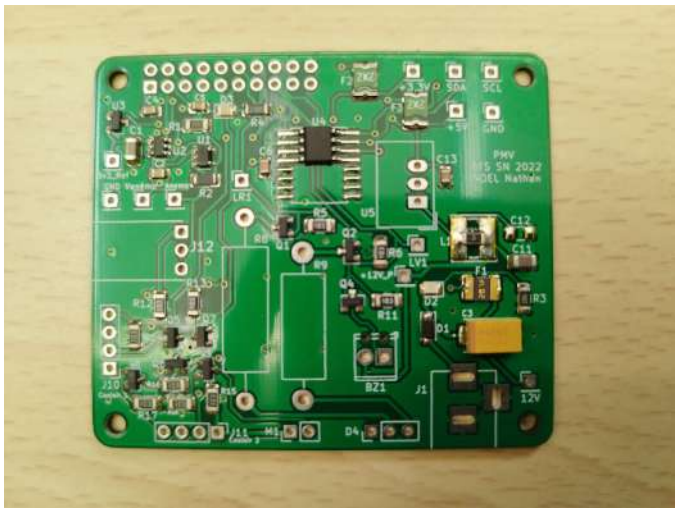
PMV_Commun **Rev:** 2022-04-15

	Sourced	Placed	References	Value	Footprint	Quantity
1	☐	☐	C4, C5, C8, C10, C13	1µF	C_0805_2012Metric	5
2	☐	☐	C1, C11, C12	4.7µF	C_1206_3216Metric	3
3	☐	☐	C2, C6	100nF	C_0805_2012Metric	2
4	☐	☐	C3	47µF	Capacitor_AVX_E_CASE_2917	1
5	☐	☐	R1, R5, R6, R11, R12, R13, R14, R15, R16, R17, R18	10k	R_1206_3216Metric	11
6	☐	☐	R2	24K	R_1206_3216Metric	1
7	☐	☐	R3	2.2k	R_1206_3216Metric	1
8	☐	☐	R4	330	R_1206_3216Metric	1
9	☐	☐	R8	18	R_Axial_DIN0617_L17.0mm_D6.0mm_P25.40mm_Horizontal	1
10	☐	☐	R9	10	R_Axial_DIN0614_L14.3mm_D5.7mm_P20.32mm_Horizontal	1
11	☐	☐	L1	3.3µF	Choke_TCK-062	1
12	☐	☐	D1	PMEG3010ER	D_SOD-123	1
13	☐	☐	D2	LED_CMS_Jaune	LED_0805_2012Metric	1
14	☐	☐	D3	LED_Rouge	LED_0805_2012Metric	1
15	☐	☐	D4	LED_ARGB	Embase_AMPHODU_HE14_Droite_3	1
16	☐	☐	U1	LMP7731	SOT-23-5	1
17	☐	☐	U2	ADC101C021CIMK	SOT-23-6	1
18	☐	☐	U3	REF3030	SOT-23	1
19	☐	☐	U4	DS3231M	SOIC-Mixte_16N_6_8	1
20	☐	☐	U5	TSR 0.5-2450	Converter_DCDC_TRACO_TSR-1_THT	1
21	☐	☐	F2, F3	400mA	Polyswitch_MICROSMD110F-2	2
22	☐	☐	F1	1,1A 2,2A	Polyswitch_MICROSMD110F-2	1
23	☐	☐	Q1, Q2, Q4, Q9	MGSF2N02ELT1G	SOT-23	4

Pour placer les composants je me suis servi d'une petite pince pour pouvoir attraper les CMS et les mettre correctement en faisant attention de bien respecter le sens de chaque composant, surtout pour les LED, car il fallait vérifier le sens de polarisation des LED, il fallait déterminer l'anode et la cathode de la LED. Et pour être sur que le LED est dans le bon sens je me suis servie d'un multimètre en position de test pour voir si la LED s'allume dans le bon sens ou non

Maintenant que tout les composants en CMS sont mis correctement et au bon endroit, il est temps de mettre la carte dans le four à fusion

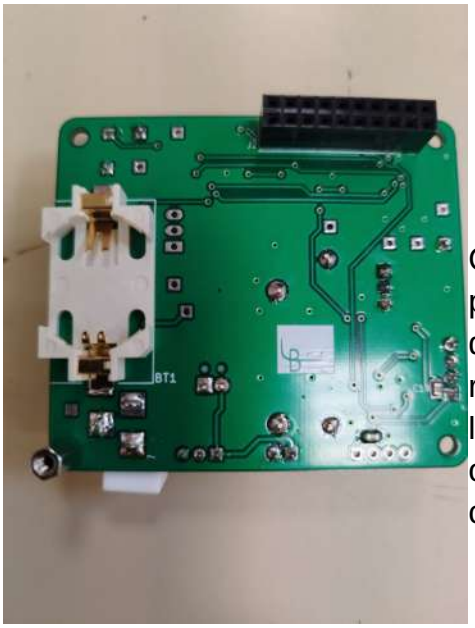
Le four à fusion peux chauffer de 95°C jusqu'à 255°C, il faut attendre 2min 50 pour que le four chauffe correctement la pâte à braser sur les composants. Une fois les 2min 50 passée, on arrête le four, on attend que la carte refroidisse et on la sort du four.Voici le résultat de la carte après le passage dans le four.



Maintenant que les composants en CMS sont soudés, il ne reste plus qu'a mettre les composants en traversant, pour les souder, c'est le plus simple, il sufit juste de prendre un fer chaud et un fil d'étain et de souder les composants traversant

Voici le résultat de la carte après avoir souder les composants traversants

Mais nous avons une contrainte, il y a deux composants en CMS que l'on ne peux pas souder avec le four a fusion car cette méthode n'est prévu que pour une face, il faut donc souder nos deux derniers composants en CMS à l'air chaud



On a eu aussi un problème sur une des résistances au moment de la distribution des composants, on nous as donner une résistance de 2K ohm qui n'est pas la bonne, sur notre liste de matériel, la résistance R1 à besoin de 21K ohm, on nous les distribuant Mr Hortolland a du se trompé, du coup on l'a vite changer.

Maintenant qu'on as fini de souder la carte, il ne reste plus qu'a nettoyer les imperfection de la carte due à la soudure avec de l'alcool isopropylique

Conclusion

A partir de maintenant , il va falloir finaliser les tests de la carte et voir si ma LED, mon buzzer et le ventilateur de la LED fonctionne sur la carte et voir aussi si les composants peuvent être piloté par la Raspberry. Pour le futur du projet, je pense qu'il faudrait faire une structure facile à installer et à transporter et de faire un genre de boîtier à positionner près de la structure des capteurs pour y mettre ma LED et mon buzzer à l'intérieur.

PARTIE INDIVIDUELLE RICHARD WILLIAM

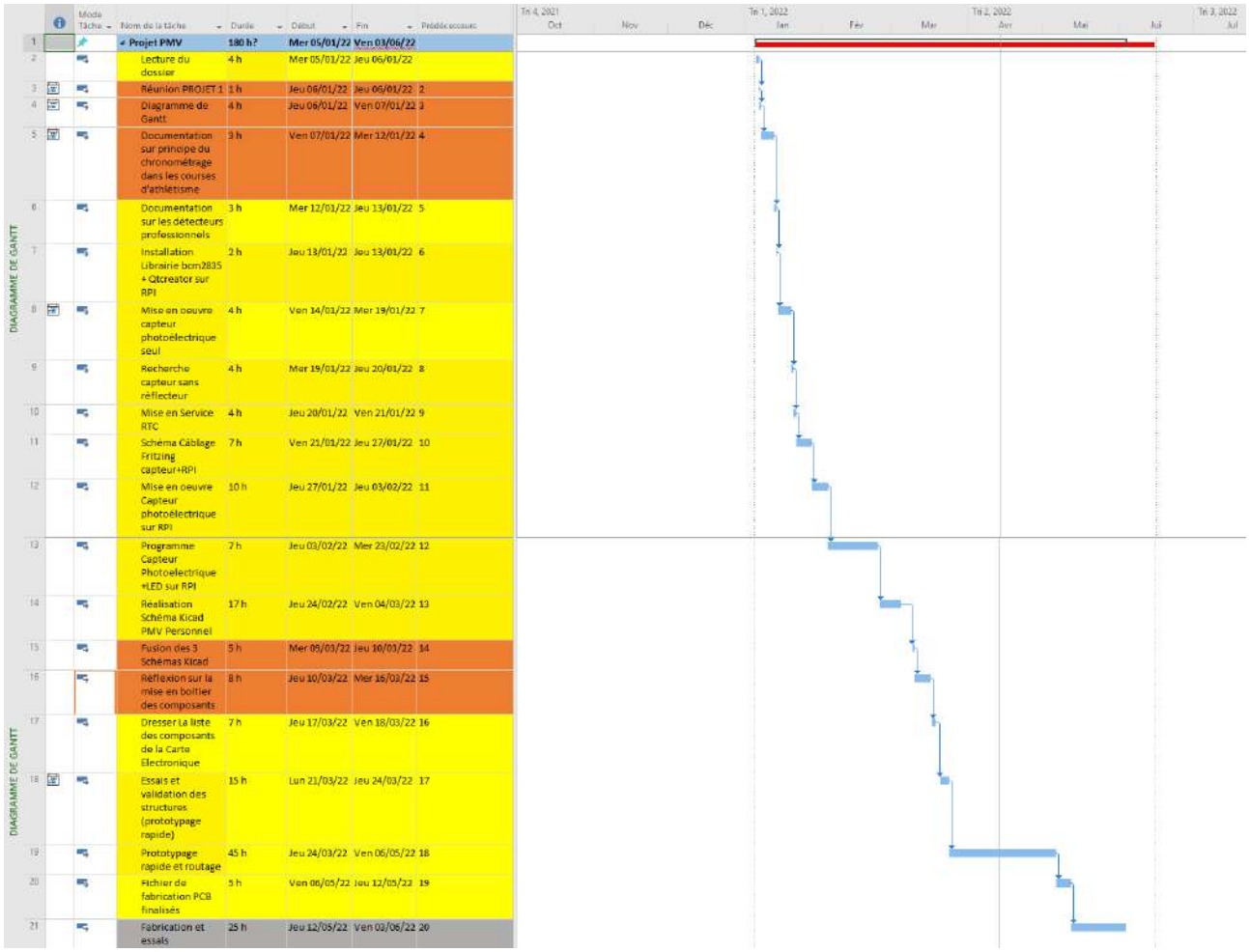
Compte rendu

J'ai pour rôle celui de l'EC2 qui consiste à choisir et à mettre en œuvre des capteurs de détection situés sur la ligne d'arrivée de la course afin de détecter le passage d'un coureur sur un couloir d'une distance de 1,2m ainsi que d'une Horloge Temps Réel (RTC) branchée sur un GPIO de la Raspberry Pi 3 permettant de compter le temps du coureur et de communiquer en I2C.

Le professeur m'a donc expliqué ma partie puis j'ai pris connaissance des documents présentant le projet PMV.

Diagramme de Gantt Prévisionnel

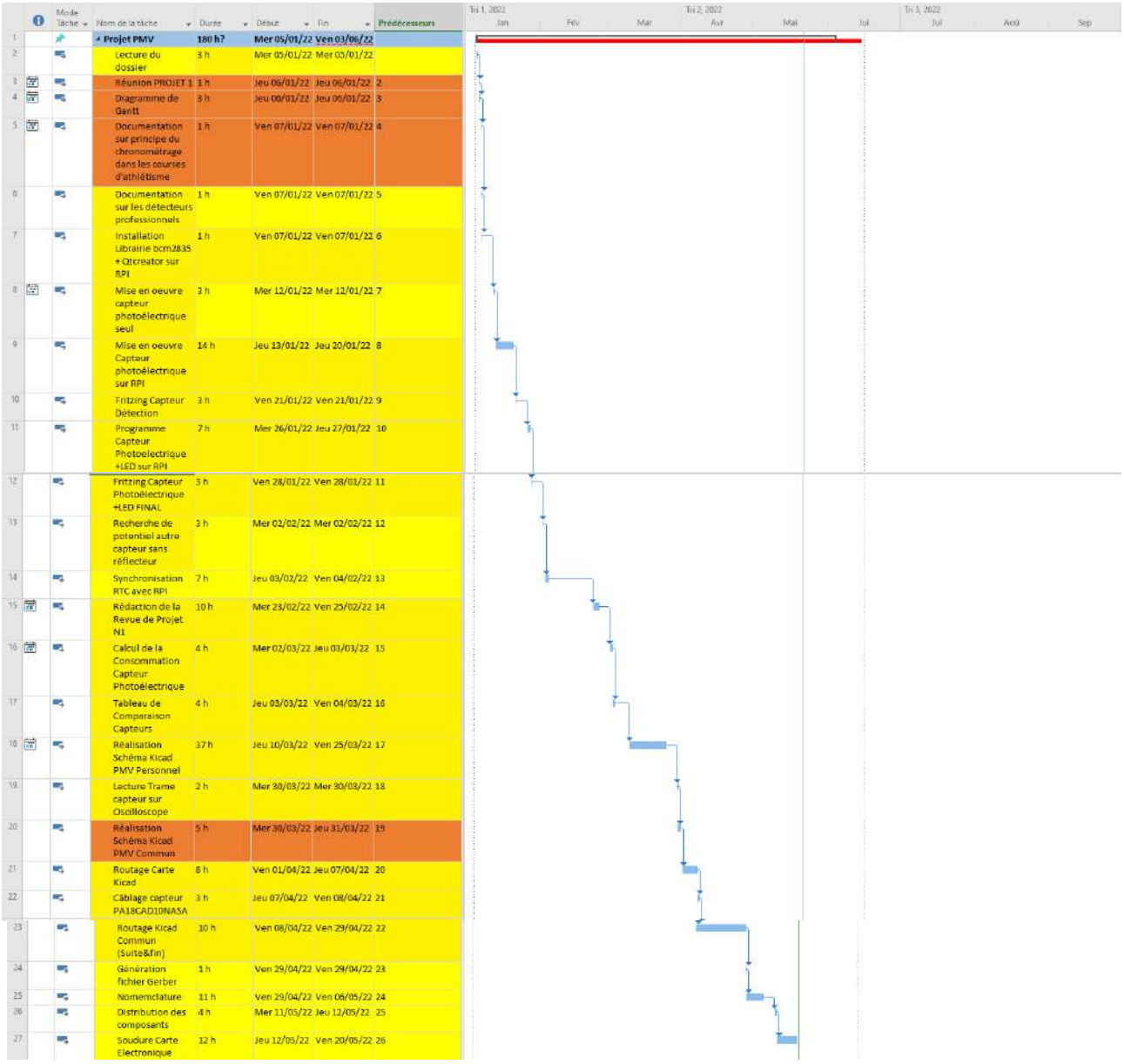
Voici le diagramme de Gantt montrant comment l'organisation était prévue et organisée au départ du projet



	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		Projet PMV	180 h?	Mer 05/01/22	Ven 03/06/22	
2		Lecture du dossier	4 h	Mer 05/01/22	Jeu 06/01/22	
3		Réunion PROJET 1	1 h	Jeu 06/01/22	Jeu 06/01/22	2
4		Diagramme de Gantt	4 h	Jeu 06/01/22	Ven 07/01/22	3
5		Documentation sur principe du chronométrage dans les courses d'athlétisme	3 h	Ven 07/01/22	Mer 12/01/22	4
6		Documentation sur les détecteurs professionnels	3 h	Mer 12/01/22	Jeu 13/01/22	5
7		Installation Librairie bcm2835 + Qtcreator sur RPI	2 h	Jeu 13/01/22	Jeu 13/01/22	6
8		Mise en oeuvre capteur photoélectrique seul	4 h	Ven 14/01/22	Mer 19/01/22	7
9		Recherche capteur sans réflecteur	4 h	Mer 19/01/22	Jeu 20/01/22	8
10		Mise en Service RTC	4 h	Jeu 20/01/22	Ven 21/01/22	9
11		Schéma Câblage Fritzing capteur+RPI	7 h	Ven 21/01/22	Jeu 27/01/22	10
12		Mise en oeuvre Capteur photoélectrique sur RPI	10 h	Jeu 27/01/22	Jeu 03/02/22	11
13		Programme Capteur Photoelectrique +LED sur RPI	7 h	Jeu 03/02/22	Mer 23/02/22	12
14		Réalisation Schéma Kicad PMV Personnel	17 h	Jeu 24/02/22	Ven 04/03/22	13
15		Fusion des 3 Schémas Kicad	5 h	Mer 09/03/22	Jeu 10/03/22	14
16		Réflexion sur la mise en boîtier des composants	8 h	Jeu 10/03/22	Mer 16/03/22	15
17		Dresser La liste des composants de la Carte Electronique	7 h	Jeu 17/03/22	Ven 18/03/22	16
18		Essais et validation des structures (prototypage rapide)	15 h	Lun 21/03/22	Jeu 24/03/22	17
19		Prototypage rapide et routage	45 h	Jeu 24/03/22	Ven 06/05/22	18
20		Fichier de fabrication PCB finalisés	5 h	Ven 06/05/22	Jeu 12/05/22	19
21		Fabrication et essais	25 h	Jeu 12/05/22	Ven 03/06/22	20

Diagramme de Gantt Actuel

Et celui-ci correspond au véritable déroulement des évènements au fil du projet:

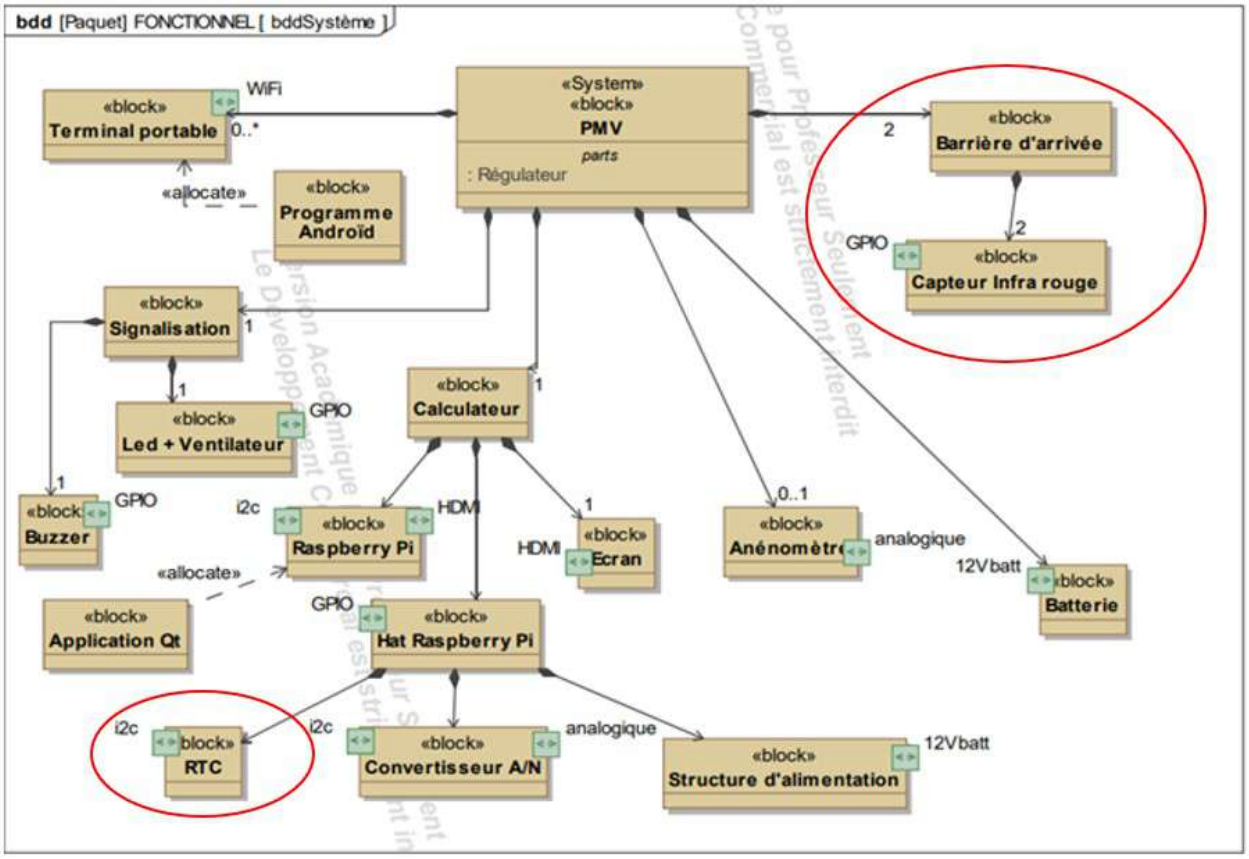


	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		Projet PMV	180 h?	Mer 05/01/22	Ven 03/06/22	
2		Lecture du dossier	3 h	Mer 05/01/22	Mer 05/01/22	
3		Réunion PROJET 1	1 h	Jeu 06/01/22	Jeu 06/01/22	2
4		Diagramme de Gantt	3 h	Jeu 06/01/22	Jeu 06/01/22	3
5		Documentation sur principe du chronométrage dans les courses d'athlétisme	1 h	Ven 07/01/22	Ven 07/01/22	4
6		Documentation sur les détecteurs professionnels	1 h	Ven 07/01/22	Ven 07/01/22	5
7		Installation Librairie bcm2835 + Qtcreator sur RPI	1 h	Ven 07/01/22	Ven 07/01/22	6
8		Mise en oeuvre capteur photoélectrique seul	3 h	Mer 12/01/22	Mer 12/01/22	7
9		Mise en oeuvre Capteur photoélectrique sur RPI	14 h	Jeu 13/01/22	Jeu 20/01/22	8
10		Fritzing Capteur Détection	3 h	Ven 21/01/22	Ven 21/01/22	9
11		Programme Capteur Photoelectrique +LED sur RPI	7 h	Mer 26/01/22	Jeu 27/01/22	10
12		Fritzing Capteur Photoélectrique +LED FINAL	3 h	Ven 28/01/22	Ven 28/01/22	11
13		Recherche de potentiel autre capteur sans réflecteur	3 h	Mer 02/02/22	Mer 02/02/22	12
14		Synchronisation RTC avec RPI	7 h	Jeu 03/02/22	Ven 04/02/22	13
15		Rédaction de la Revue de Projet N1	10 h	Mer 23/02/22	Ven 25/02/22	14
16		Calcul de la Consommation Capteur Photoélectrique	4 h	Mer 02/03/22	Jeu 03/03/22	15
17		Tableau de Comparaison Capteurs	4 h	Jeu 03/03/22	Ven 04/03/22	16
18		Réalisation Schéma Kicad PMV Personnel	37 h	Jeu 10/03/22	Ven 25/03/22	17
19		Lecture Trame capteur sur Oscilloscope	2 h	Mer 30/03/22	Mer 30/03/22	18
20		Réalisation Schéma Kicad PMV Commun	5 h	Mer 30/03/22	Jeu 31/03/22	19
21		Routage Carte Kicad	8 h	Ven 01/04/22	Jeu 07/04/22	20
22		Câblage capteur PA18CAD10NASA	3 h	Jeu 07/04/22	Ven 08/04/22	21
23		Routage Kicad Commun (Suite&fin)	10 h	Ven 08/04/22	Ven 29/04/22	22
24		Génération fichier Gerber	1 h	Ven 29/04/22	Ven 29/04/22	23
25		Nomenclature	11 h	Ven 29/04/22	Ven 06/05/22	24
26		Distribution des composants	4 h	Mer 11/05/22	Jeu 12/05/22	25
27		Soudure Carte Electronique	12 h	Jeu 12/05/22	Ven 20/05/22	26

Diagramme de Bloc internes

Pour comprendre le fonctionnement du système PMV, voici le diagramme de bloc interne montrant les différents composant et les éléments qui leurs sont liés :

Sur ce projet, je doit m'occuper de mettre en œuvre le capteur infra rouge pour détecter le passage des coureurs ainsi qu'une horloge temps réel(RTC) pour permettre la communication en I2C de composant tel que l'anémomètre et ma RTC mais également pour déclencher des évènements selon l'heure .



Capteur Photoélectrique XUB*9APANL2*

Après cela je suis allé sur le site de RS du capteur pour y relever différentes informations comme le prix, le code commande du produit, la tension d'alimentation, la distance d'émission du faisceau infrarouge…

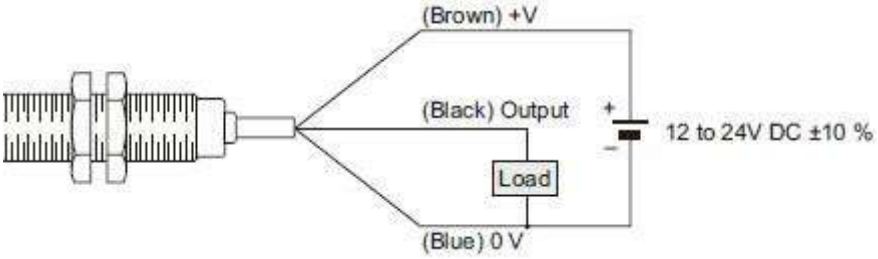
Documentation XUB9APANL2 + Schéma de raccordement

Main

Range of product	OsiSense XU
Series name	General purpose single mode
Electronic sensor type	Photo-electric sensor
Sensor name	XUB
Sensor design	Cylindrical M18
Detection system	Polarised reflex
Material	Plastic
Line of sight type	Axial
Type of output signal	Discrete
Supply circuit type	DC
Wiring technique	3-wire
Discrete output type	PNP
Discrete output function	1 NO
Electrical connection	Cable
Cable length	2 m
Product specific application	-
Emission	Red polarised reflex
[Sn] nominal sensing distance	2 m polarised reflex need reflector XUZC50

Complementary

Enclosure material	PBT
Lens material	PMMA
Maximum sensing distance	3 m polarised reflex
Output type	Solid state
Add on output	Without
Wire insulation material	PvR
Status LED	1 LED (yellow) for output state
[Us] rated supply voltage	12...24 V DC with reverse polarity protection
Supply voltage limits	10...36 V DC
Switching capacity in mA	<= 100 mA (overload and short-circuit protection)
Switching frequency	<= 500 Hz



Ensuite, j'ai été amené à mettre en œuvre ce capteur électrique avec une alimentation de 12 V, car l'ensemble des composant utilisé peuvent fonctionner à cette tension.

Raccordement Capteur

Ce capteur fonctionne avec un réflecteur de dimension 50x50mm qui sert a réfléchir l'onde, et qui doit être aligné avec le XUB pour pouvoir détecter le passage d'un coureur entre les deux:

Fiche produit
Caractéristiques

XUZC50
OsiSense XU - réflecteur universel - 50x50mm - pour détecteur

Green Premium

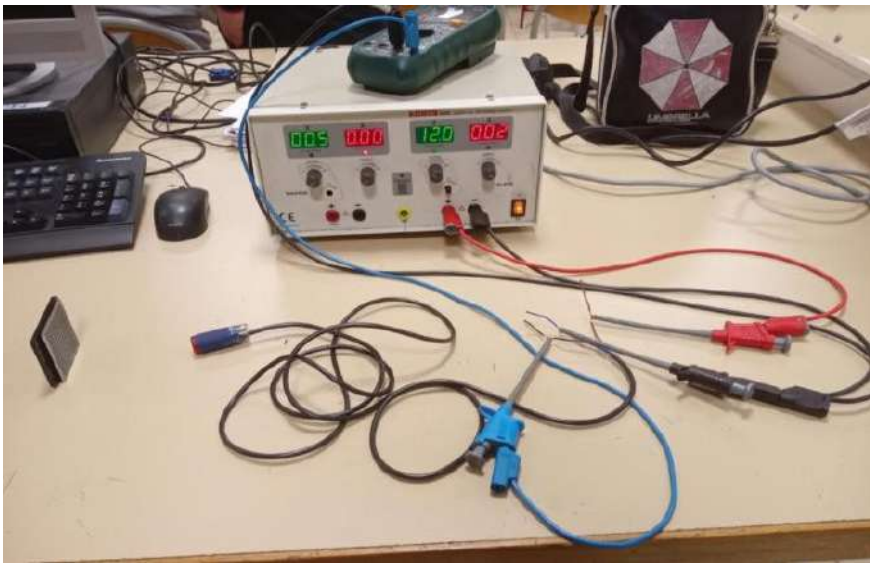
Statut commercial: Commercialisé

Principales

Gamme de produits	Telemecanique Photoelectric sensors XU
Catégorie d'accessoire / de pièce détachée	Accessoires de capteurs
Type d'accessoires / pièces séparées	Réflecteur
Destination d'accessoire / de pièce détachée	Détecteur photoélectrique à réflexion standard
Forme du réflecteur	Carré
Vente par quantité indivisible	1

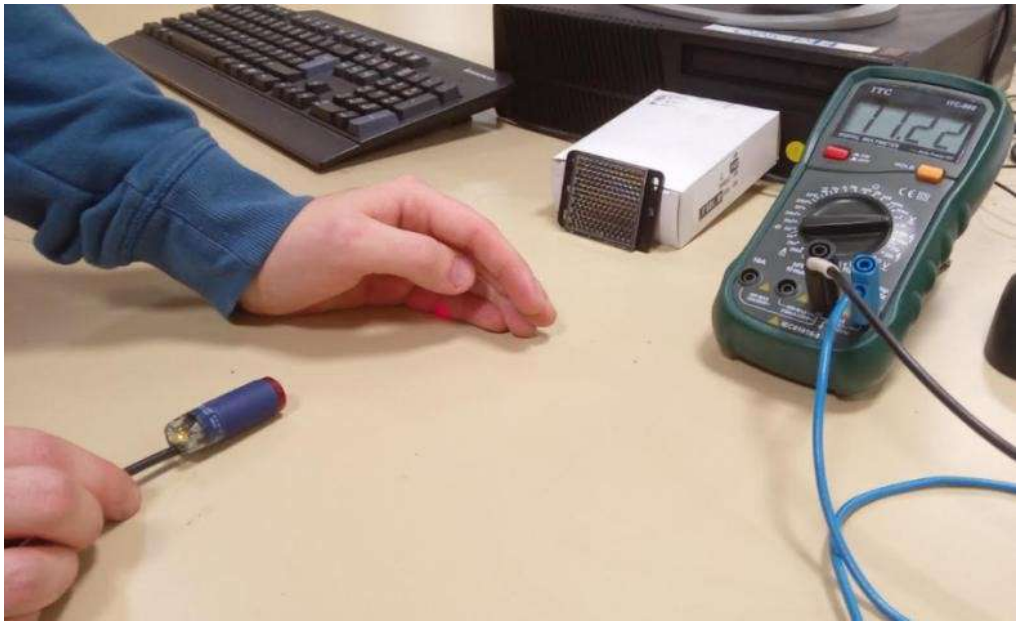
Complémentaires

Dimension de l'accessoire	50 x 50 mm
Poids du produit	0,02 Kg



Lorsque le capteur ne perçoit pas d'obstacle, le voyant lumineux est éteint et le capteur passe à l'état bas soit 0V comme ci-dessous:

Lorsqu’il en perçoit un, le voyant lumineux s’allume et le capteur passe à l’état haut soit 12V.



Une fois la phase de test terminé, j'ai donc débuté les recherches pour la mise en œuvre du capteur photoélectrique sur un Raspberry Pi en:

Effectuant les calculs d'un pont diviseur pour trouver la valeur des résistances en série E12 (étant les seules à notre disposition) à câbler sur la platine d'essais pour abaisser la tension d'entrée au niveau de celle de la Raspberry qui est alimenté en 3.3V:

Sachant que mon capteur photoélectrique est alimenté en 12 V :

E 3 (±20%) : 100 - 220 - 470

E 6 (±10%) : 100 - 150 - 220 - 330 - 470 - 680

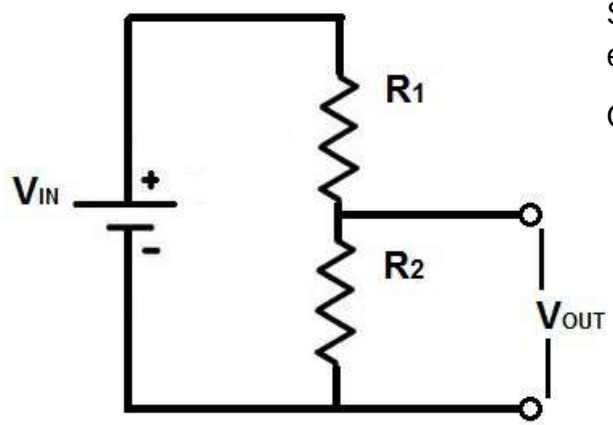
E12 (±10%) : 100 - 120 - 150 - 180 - 220 - 270 - 330
390 - 470 - 560 - 680 - 820

E24 (±5%) : 100 - 110 - 120 - 130 - 150 - 160 - 180
200 - 220 - 240 - 270 - 300 - 330 - 360 - 390
430 - 470 - 510 - 560 - 620 - 680 - 750 - 820 - 910

E48 : 100 - 105 - 110 - 115 - 121 - 127 - 133
140 - 147 - 154 - 162 - 169 - 178 - 187 - 196
205 - 215 - 226 - 237 - 249 - 261 - 274 - 287
301 - 316 - 332 - 348 - 365 - 383 - 402 - 422
442 - 464 - 487 - 511 - 536 - 562 - 590 - 619
649 - 681 - 715 - 750 - 787 - 825 - 866 - 909 - 953

E96 (±1%) : 100 - 102 - 105 - 107 - 110 - 113 - 115
118 - 121 - 124 - 127 - 130 - 133 - 137 - 140
143 - 147 - 150 - 154 - 158 - 162 - 165 - 169
174 - 178 - 182 - 187 - 191 - 196 - 200 - 205
210 - 215 - 221 - 226 - 232 - 237 - 243 - 249
255 - 261 - 267 - 274 - 280 - 287 - 294 - 301
309 - 316 - 324 - 332 - 340 - 348 - 357 - 365
374 - 383 - 392 - 402 - 412 - 422 - 432 - 442
453 - 464 - 475 - 487 - 499 - 511 - 523 - 536
549 - 562 - 576 - 590 - 604 - 619 - 634 - 649
665 - 681 - 698 - 715 - 732 - 750 - 768 - 787
806 - 825 - 845 - 866 - 887 - 909 - 931 - 953 - 976

Pont diviseur de tension



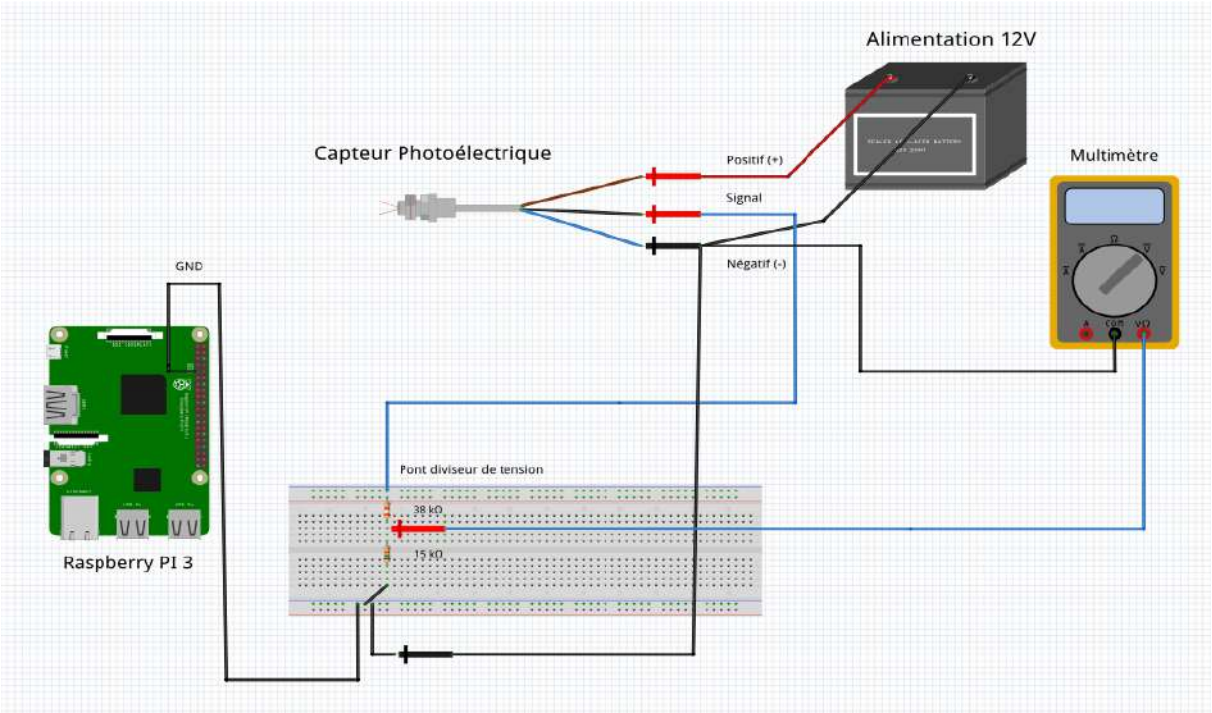
Sachant que mon capteur photoélectrique est alimenté en 12 V

On part de la loi d'ohm :

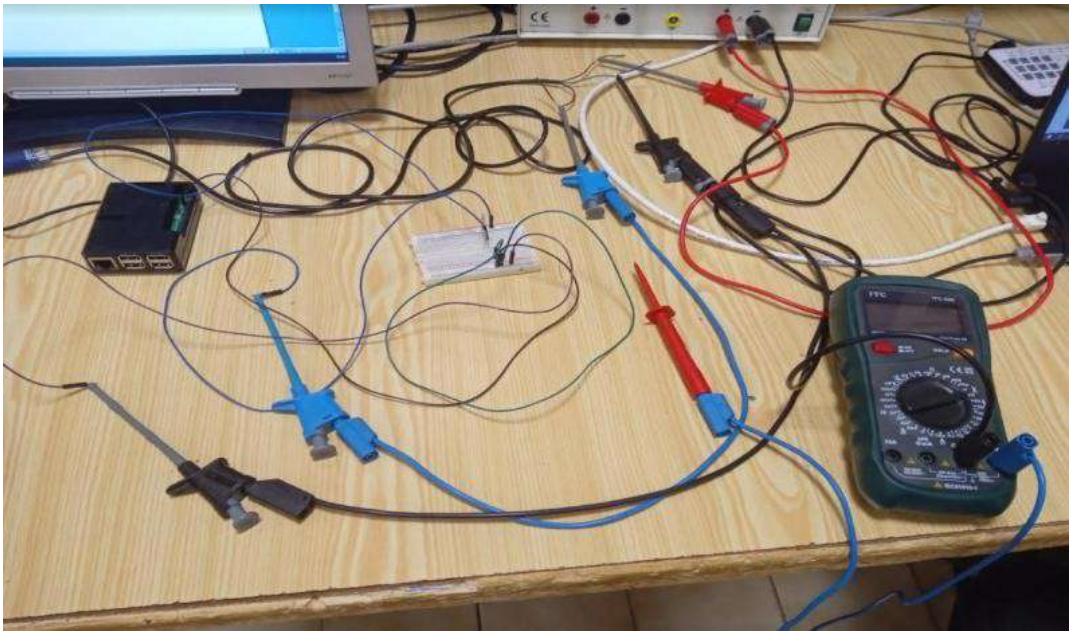
$$U = R * I$$
$$U = (R1 + R2) * I$$
$$I = \frac{U1}{R1 + R2}$$
$$V_{out} = V_{in} * \frac{R2}{R1 + R2}$$
$$V_{out} = 12 * \frac{15000}{39000 + 15000} = 3.3V$$

J’ai donc choisis une résistance de 39kΩ et une autre de 15kΩ car elles atténuent bien le niveau de tension ,Une fois cela fait, j’ai créé un schéma Fritzing de mon capteur sur Raspberry Pi à l’aide du schéma de raccordement ci-dessous , j’ai cherché une librairie incluant des grippes-fils , multimètre et une alimentation pour un résultat plus proche de la réalité:

Schéma Fritzing capteur Photoélectrique avec pont diviseur de tension

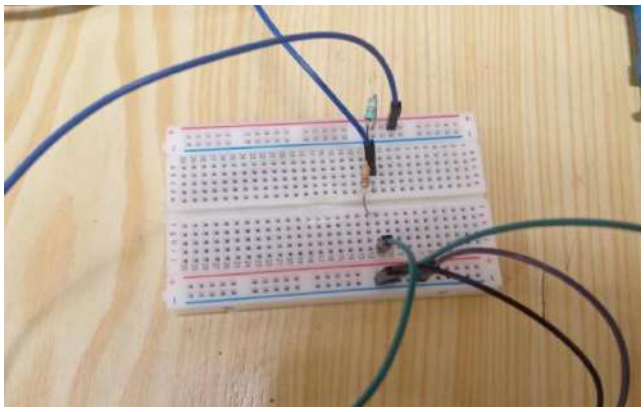


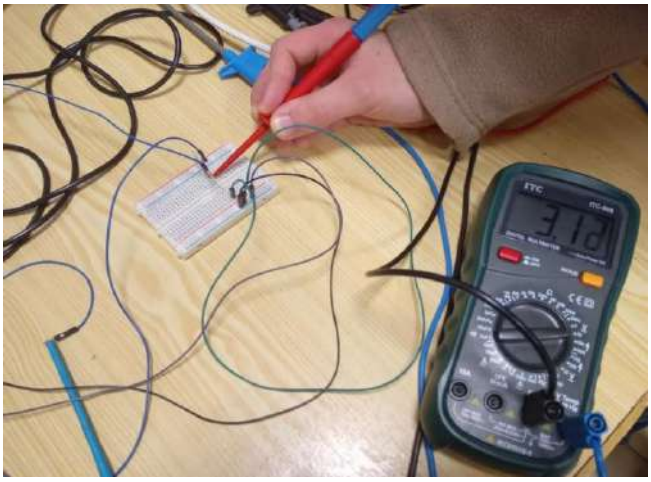
Raccordement final du capteur photoélectrique avec un pont diviseur de tension



Zoom sur la platine d'essais :

Le pont diviseur atténue donc bien la tension du générateur au niveau de la Raspberry Pi en 3.3V bien que la tolérance des résistances fait que la tension est légèrement en dessous.





Programme LED +État Haut/Bas

Pour permettre l'affichage de la détection, j'ai réalisé avec l'aide de mon professeurs un programme en langage C sur Rasbian (le système d'exploitation de Raspberry PI) ce basant sur la fusion des 2 ci-dessous permettant pour le premier, d'allumer la Led lors du passage d'un coureur et de l'éteindre et pour le second de définir l'état haut ou bas

J'ai également installé la librairie bcm2835 qui permet d'utiliser les GPIO (entrée sortie) de la carte Raspberry.

```
#include <stdio.h>
#include <bcm2835.h>

#define LED RPI_BPLUS_GPIO_J8_16 // Numéro de la broche 23 sur le GPIO

// g++ Clignotement_LED_GPIO23.cpp -l bcm2835 -o Clignotement_LED_GPIO23 // Ligne de compilation

void setup()
        {
            if (!bcm2835_init())
            {
              printf("bcm2835_init failed. Are you running as root??\n");
            }
        }

int main(void)
{
setup();
bcm2835_gpio_fsel(LED, BCM2835_GPIO_FSEL_OUTP); // Broche LED (GPIO23) en sortie
printf("Clignotement de la LED du GPIO23 (broche 16)\n");
while(1)
        {
        bcm2835_gpio_write(LED, LOW); //LED = 0
        bcm2835_delay(500);
        bcm2835_gpio_write(LED, HIGH); //LED = 1
        bcm2835_delay(500);
        }
bcm2835_close();
return 0;
}
```

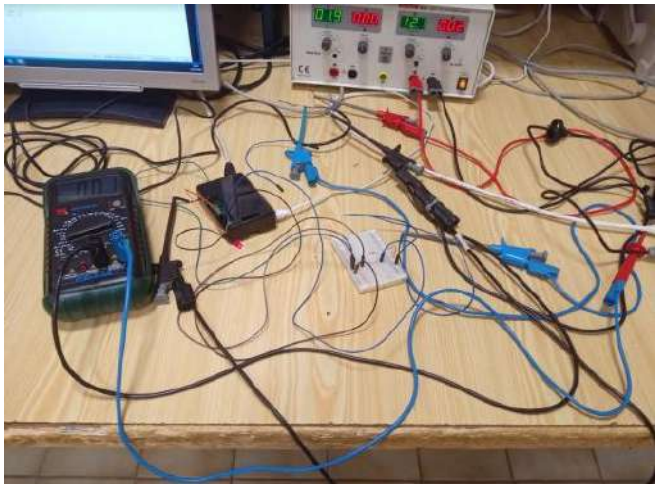
Programme de contrôle de la LED

```
#include <stdio.h>
#include <bcm2835.h>

#define NV RPI_BPLUS_GPIO_J8_15  // Numéro de la broche sur le GPIO

//using namespace std;

// g++ Lecture_Niveau_GPIO15.cpp -l bcm2835 -o Lecture_Niveau_GPIO15 // Ligne de compilation

void setup()
        {
            if (!bcm2835_init())
            {
              printf("bcm2835_init failed. Are you running as root??\n");
            }
        }

int main(void)
{
setup();
bcm2835_gpio_fsel(NV, BCM2835_GPIO_FSEL_INPT); // Broche Niveau (GPIO15) en entrée
printf("Lecture état Bouton Poussoir du GPIO15 \n");
while(1)
        {
        if (bcm2835_gpio_lev(NV)) printf("Niveau Haut \n");  //Niveau = 1
        else printf("Niveau Bas \n");  //Niveau = 0
        bcm2835_delay(500);
        }
bcm2835_close();
return 0;
}
```

Programme de contrôle de l'état haut/bas

Câblage Capteur sur RPI3 avec LED

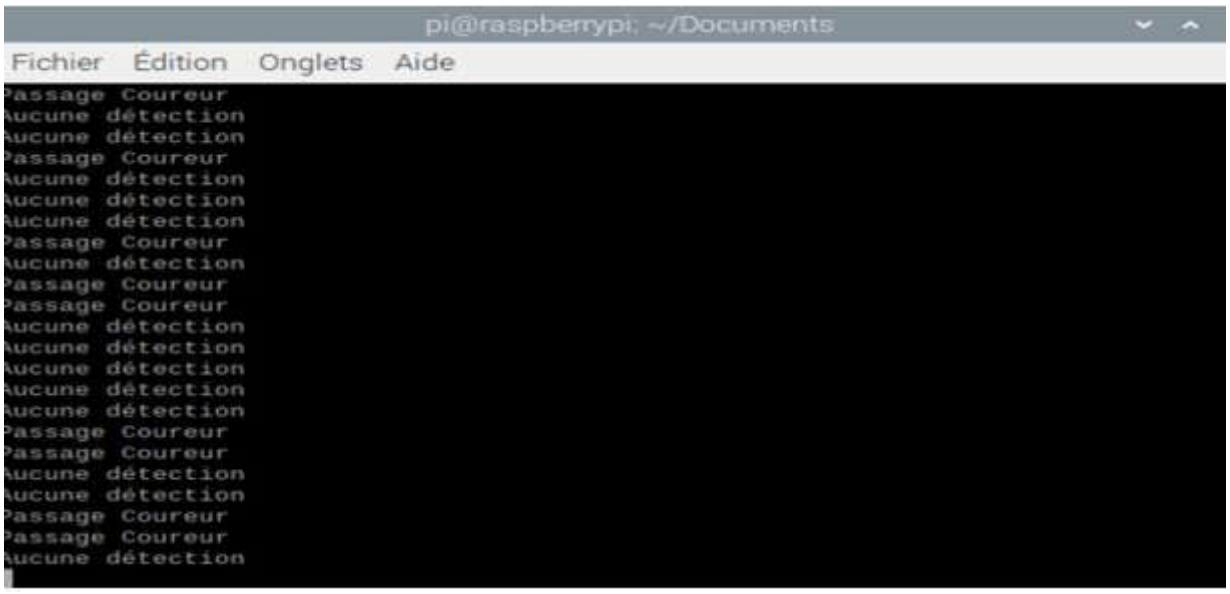
Début des tests sur le Raspberry pi pour faire en sorte que lorsqu'un coureur passe devant le capteur, il soit détecté et que la LED s'allume lorsqu'il passe et s'éteigne lorsqu'il ne détecte pas de coureur.



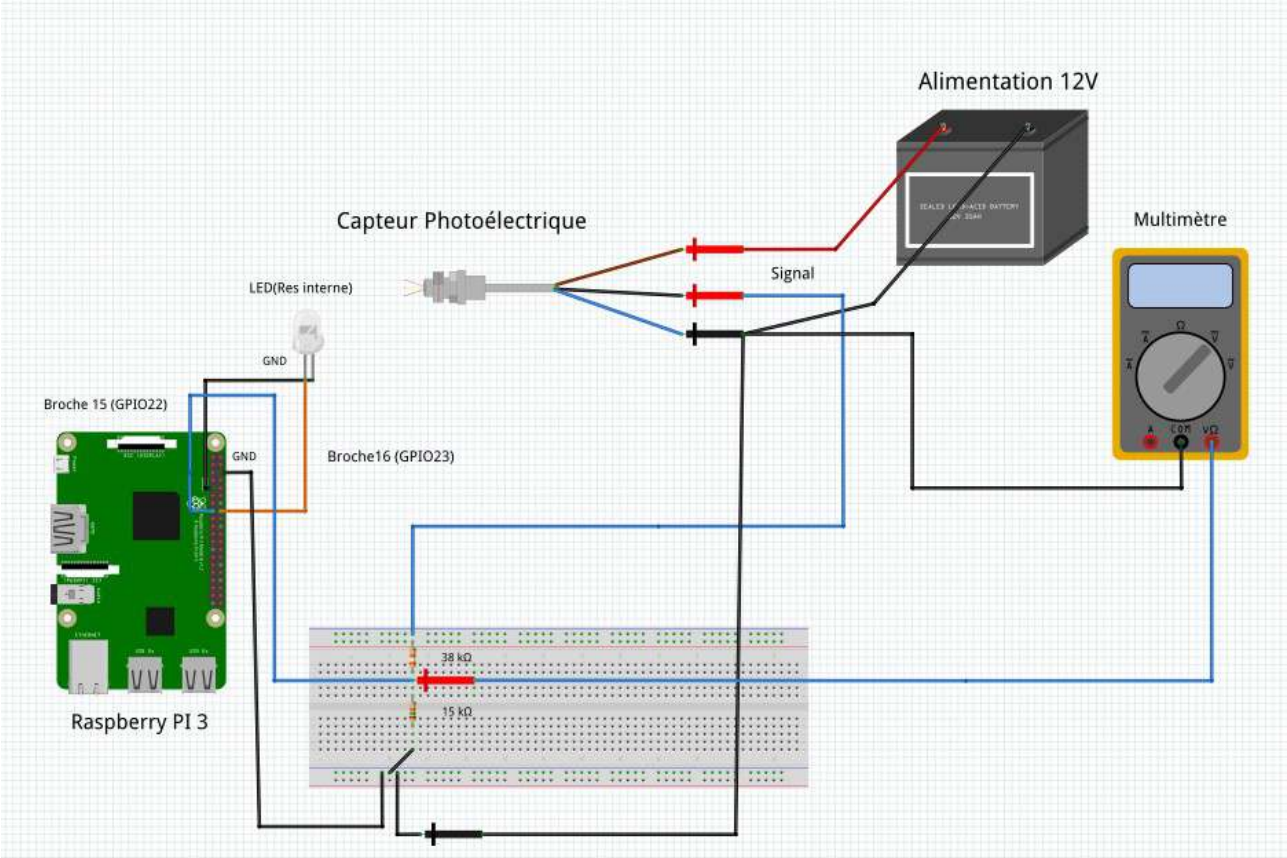
J'ai utilisé la fusion des programmes de la LED et du bouton poussoir pour arriver au programme suivant:

```
#include <stdio.h>
#include <bcm2835.h>

#define NV RPI_BPLUS_GPIO_J8_15  // Numéro de la broche de niveau HAUT ou BAS GPIO22 sur le GPIO

#define LED RPI_BPLUS_GPIO_J8_16 // Numéro de la broche GPIO23 sur le GPIO

//using namespace std;

//Ls -l // Trouver les documents
//Cd Documents //Accéder au dossier ou se trouve le programme
// g++ Detection_Passage_Avec_Led_GPIO15.cpp -l bcm2835 -o Detection_Passage_Avec_Led_GPIO15 // Ligne de compilation
//sudo ./Detection_Passage_Avec_Led_GPIO15 // Lancement du programme


void setup()
        {
            if (!bcm2835_init())
            {
              printf("bcm2835_init failed. Are you running as root??\n");
            }
        }

int main(void)
{
setup();
bcm2835_gpio_fsel(LED, BCM2835_GPIO_FSEL_OUTP); // Broche LED (GPIO23) en sortie
bcm2835_gpio_fsel(NV, BCM2835_GPIO_FSEL_INPT); // Broche NV (GPIO15) en entrée
printf("Lecture état Bouton Poussoir du GPIO15 \n");
while(1)
        {
        if (bcm2835_gpio_lev(NV))
        {
        printf("Passage Coureur \n");  //NV = 1
        bcm2835_gpio_write(LED, HIGH); //LED = 1

        }

        else
        {
        printf("Aucune détection \n");  //NV = 0
        bcm2835_gpio_write(LED, LOW); //LED = 0
        }
        bcm2835_delay(500);
        }
bcm2835_close();
return 0;
}
```

Au niveau des tests, voici ce que cela donne sur le terminal, un message s'actualisant toute les 500ms pour signifier s'il y a un passage de coureur ou non :

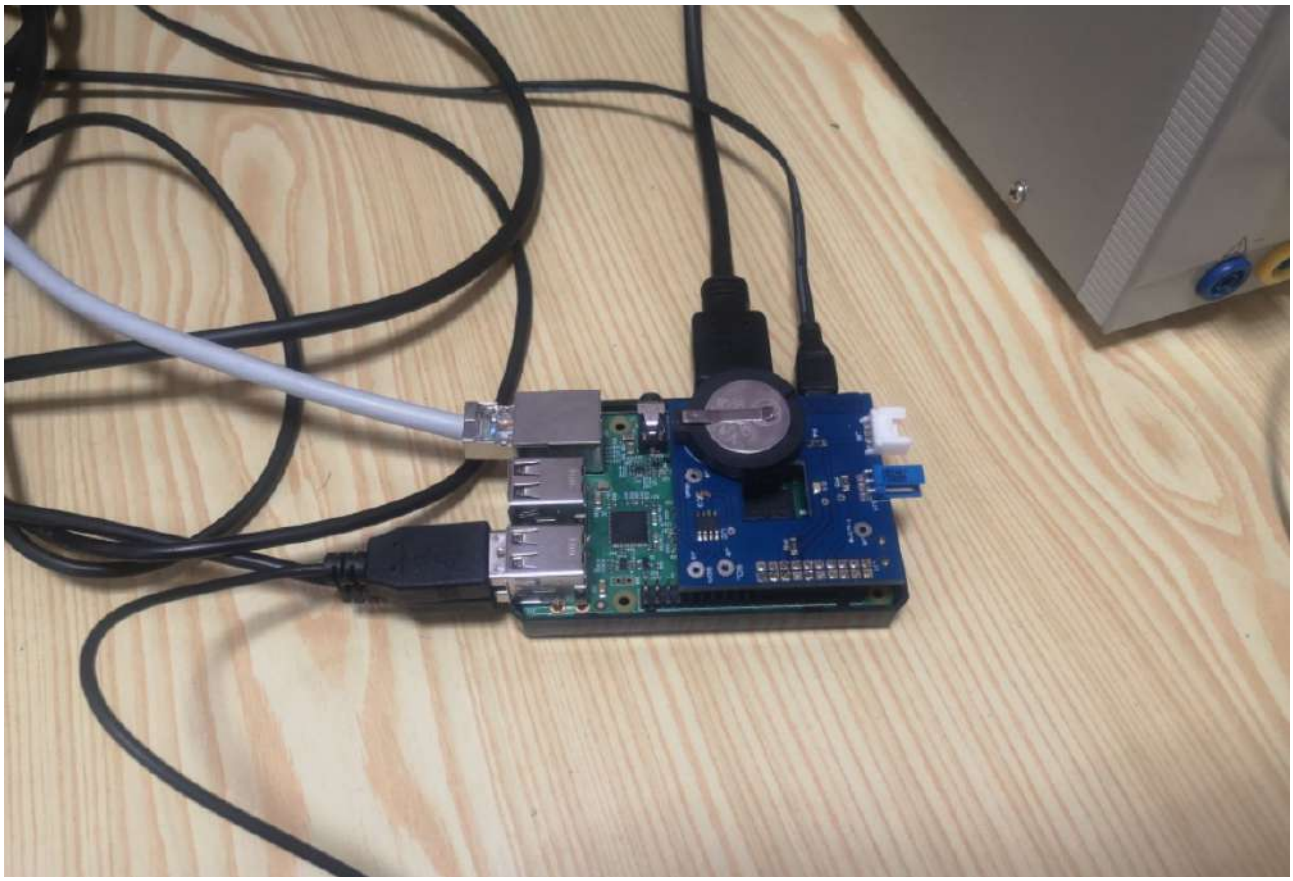


Fin de la réalisation du schéma final du Fritzing sur la platine d'essais(rajout de la LED et de broche d'état haut ou bas):

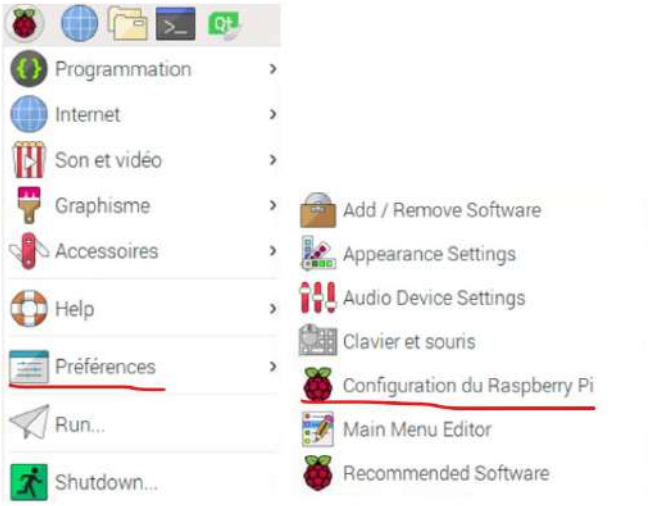


Mise à l'heure de l'Horloge en Temps Réel (RTC)

Juste après cela , j'ai effectué les branchements nécessaires entre une carte électronique disposant d'une RTC et la Raspberry Pi. Vu que je n'avais pas de carte électronique ayant une RTC a disposition à ce moment là , j'ai pris un HAT(extension de carte que l'on vient placer au dessus de la Raspberry Pi) d'un projet des années précédentes disposant de ce composant électronique pour effectuer la programmation dans ce test:



Avant de passer au programme , vu que l'horloge en temps réel fonctionne en I2C , je devais m'assurer que la liaison soit établie entre la carte RPI et la RTC pour permettre une communication:



A la suite de cela , j'ai mis a jour les paquets existant au travers de la commande

sudo apt-get update dans le terminal et avec la commande *sudo apt-get upgrade* j'installe les paquets qui ont des versions plus récentes dans la liste par rapport à celles installées sur le système:

Pour mettre a l'heure l'horloge , j'ai suivis un tutoriel en ligne dont voici les étapes que j'ai suivis:

sudo i2cdetect -y 1

cette instruction nous retourne les infos suivantes , ici le RPI à détecté un composant à l'adresse hexadécimal 0x68 qui d'après la doc constructeur , est l'adresse par défaut de la RTC:

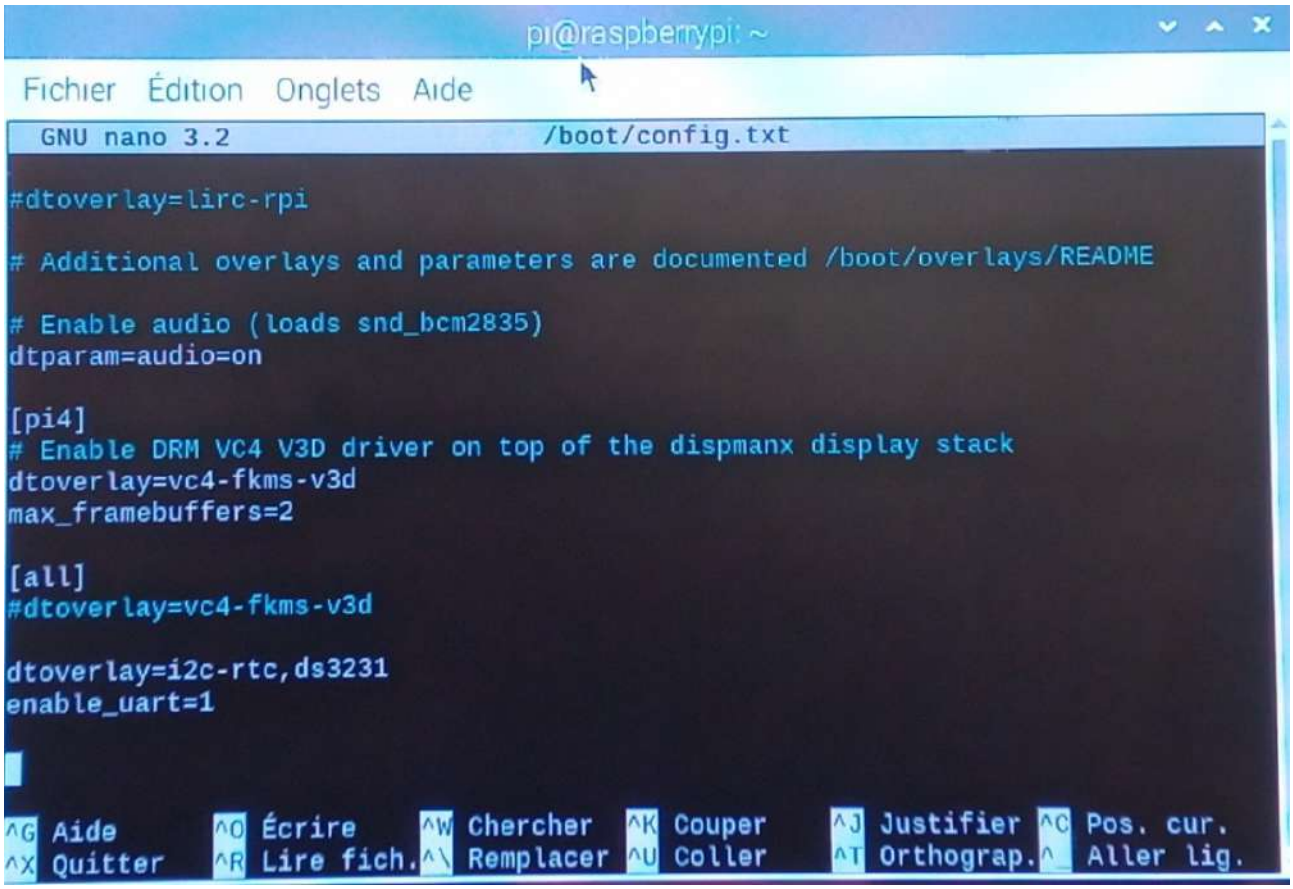
```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

deuxièmement , On va ajouter la gestion du module en éditant le fichier de configuration de boot.

sudo nano /boot/config.txt

Auquel on va ajouter un la ligne suivante à la fin du fichier :

dtoverlay=i2c-rtc,ds3231



ds3231 qui correspond a la référence de l'horloge

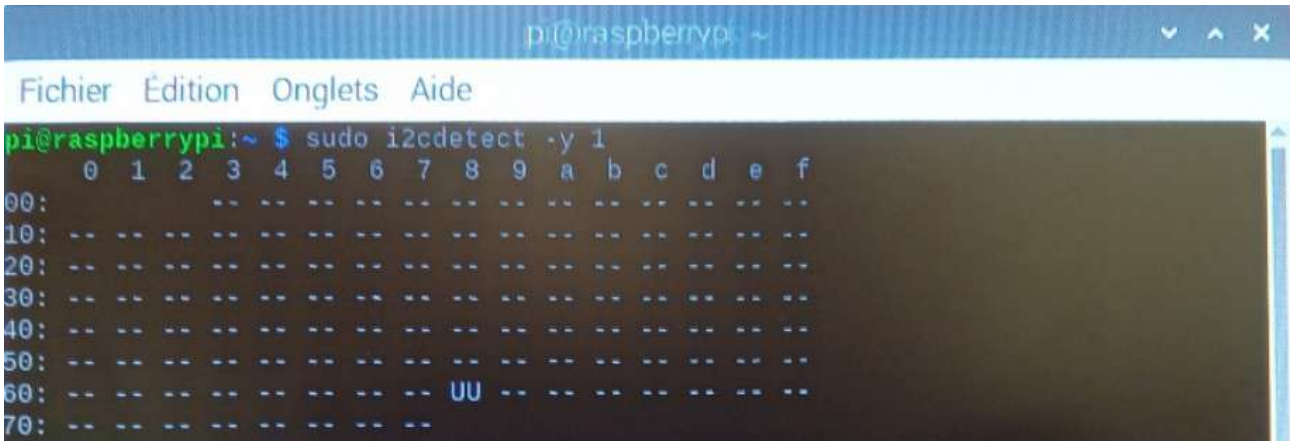
ensuite on effectue un reboot de la carte avec la commande:

sudo reboot

Une fois le système de nouveau actif, on peut exécuter de nouveau la commande:

sudo i2cdetect -y 1

On voit bien que le composant a changer d'adresse pour UU signifiant que la RTC est prête à être mise à l'heure .



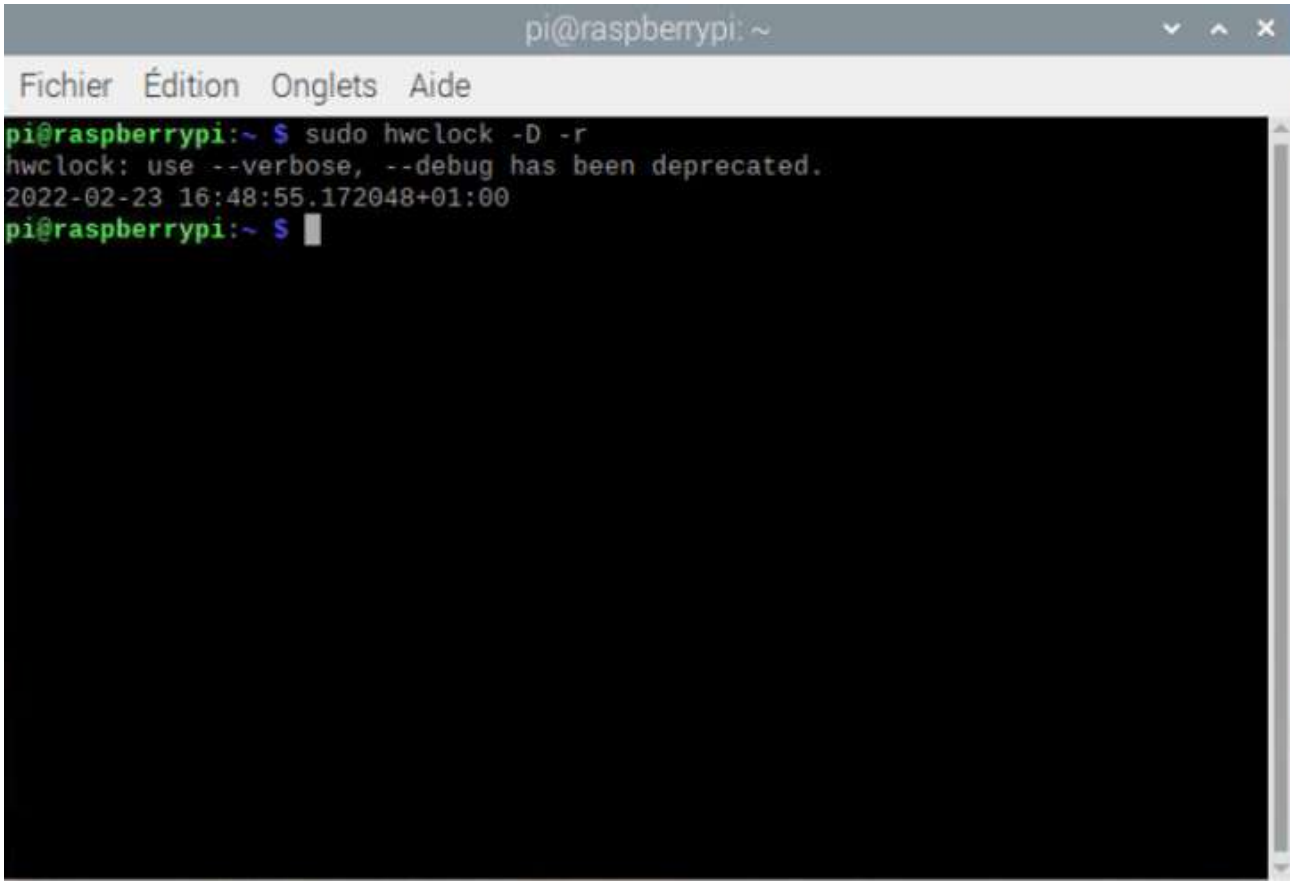
Avant dernière étape , on doit désormais désactiver la fausse horloge intégrée dans la RTC :

sudo apt-get y remove fake-hwclock

sudo update-rc.d f fakehwclock remove

Enfin , on peut désormais afficher l'heure actuelle avec la commande suivante:

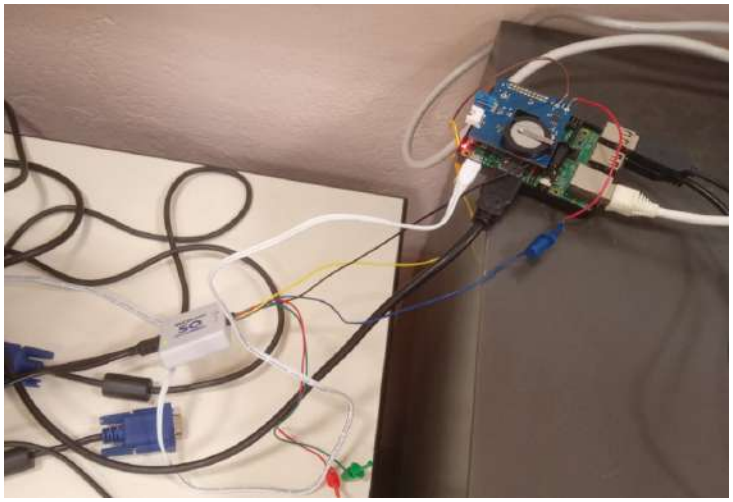
sudo hwclock -D -r



Lecture de la Trame I2C de l’Horloge Temps Réel

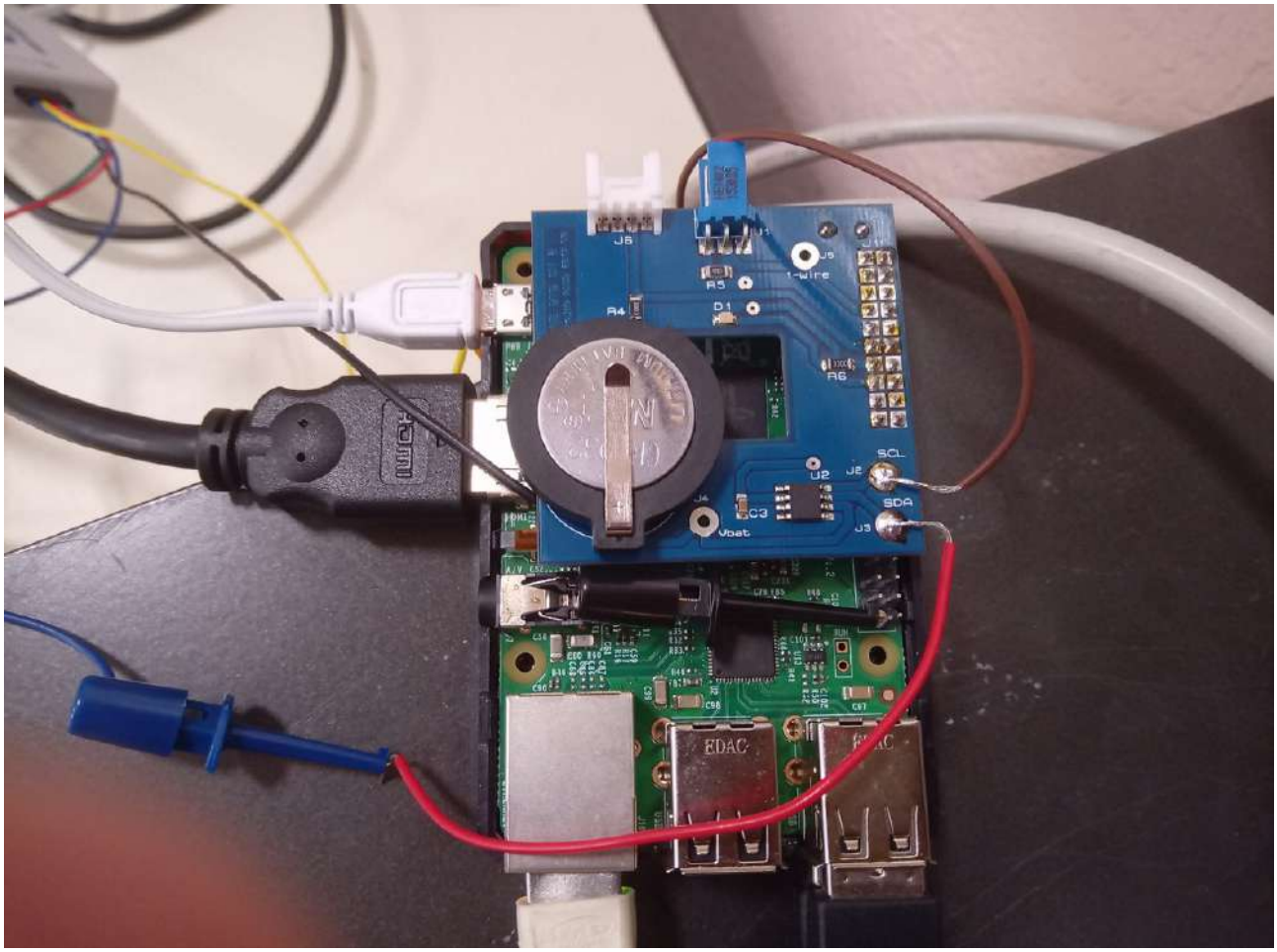
Après la mise a l’heure de l’horloge , il m’était demandé de lire la trame I2C et d’interpréter a quoi elle correspond en partant de l’heure obtenu avec la commande précédente , pour ce faire j’ai tout d’abord souder deux petit fils: une sur la broche SCL (clock/Horloge) et l’autre sur SDA (Data/Données) puis je l’ai connecté a un IKA Logic SQ25 , un analyseur logique permettant l’analyse de trame:

Câblage Ika Logic +RTC



En zoomant dessus la carte , voici comment est câbler l'ensemble:

SDA:Fil Bleu; SCL:Fil Jaune; GND:Fil Noir

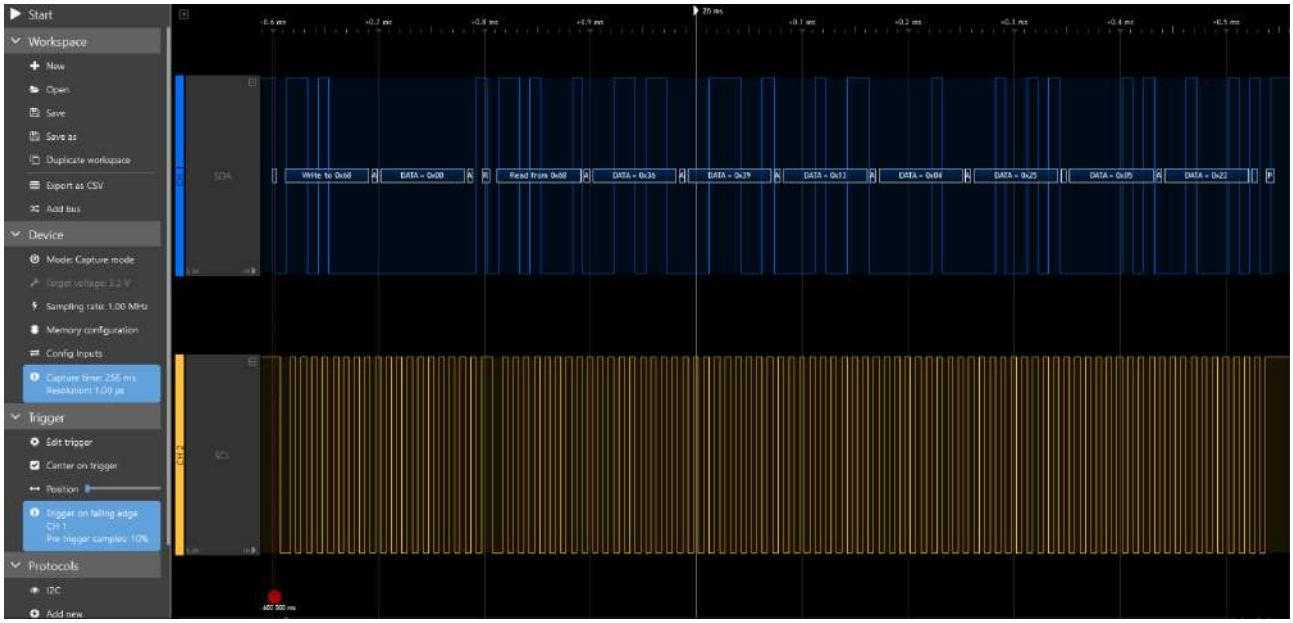


J’ai donc a nouveau sur ma Raspberry , effectuer la commande pour afficher l’heure:

```
pi@raspberrypi:~ $ sudo hwclock -D -r
hwclock: use --verbose, --debug has been deprecated.
2022-05-25 15:39:36.381444+02:00
```

Logiciel ScanaStudio

Ensuite a l’aide du logiciel ScanaStudio, je suis venu récupérer la trame de donnée correspondant a l’heure affiché dans la Raspberry 3.



Pour comprendre a quoi correspondais chacune des valeurs présente dans la trame , je suis aller voir dans la documentation constructeurs de ma RTC (DS3231) et j’ai trouvé le tableau suivant:

Table 2. Timekeeping Registers

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hours	10 Hours	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7

La colonne des adresses sert à signifier quel octet est lu sur la trame ,00h signifiant que le premier octet en hexadécimal correspondra au secondes , sur la trame cela correspond a 0x36 on convertis cela en binaire pour comparer avec le tableau:

8 4 2 1 8 4 2 1

0x36 → 0 0 1 1 0 1 1 0

Les bits 4 à 6 sont des multiples de 10 ,ce qui signifie que l'on doit multiplié la valeur de ces bits par ce dernier , ce qui donne

→ (1+2)x10 +6= 36sec on retombe bien sur le même temps que celui de la RTC ,

il suffit maintenant de répéter les opérations pour les minutes et les heures toujours en se fiant au tableau .

Puis j'ai commencé la recherche pour trouver un capteur photoélectrique permettant de renvoyer le signal du passage d'un coureur sans passer par un réflecteur télémécanique et pouvant mesurer un couloir de 1m22.

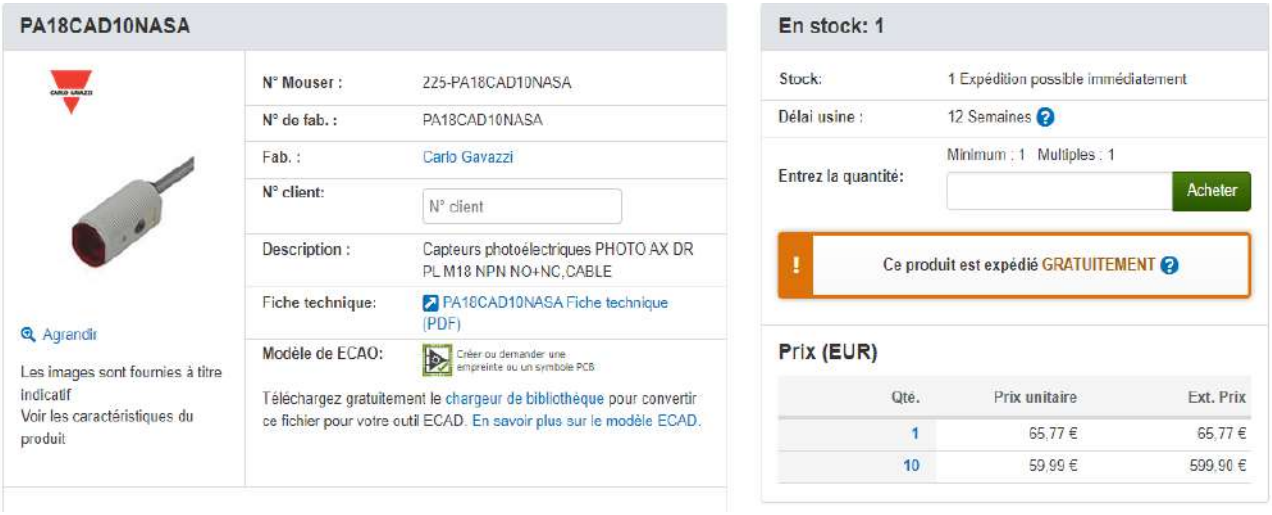
J'ai trouvé une technologie pouvant correspondre à ces critères :

Capteur à réflexion directe

Un détecteur photoélectrique à réflexion contient l'émetteur et le récepteur dans un même boîtier. Le détecteur émet un faisceau lumineux directement vers la cible distante agissant comme réflecteur, renvoyant une partie de la lumière émise vers le récepteur. Le récepteur détecte la quantité de lumière réfléchie par la cible, faisant commuter la sortie du détecteur lorsque l'intensité de lumière atteint une valeur seuil.

Voici le capteur que j'ai eu a testé correspondant à la technologie expliqué précédemment, le PA18CAD10NASA

Capteur PA18CAD10NASA



PA18CAD10NASA

N° Mouser : 225-PA18CAD10NASA
N° de fab. : PA18CAD10NASA
Fab. : Carlo Gavazzi
N° client: N° client
Description : Capteurs photoélectriques PHOTO AX DR PL M18 NPN NO+NC, CABLE
Fiche technique: PA18CAD10NASA Fiche technique (PDF)
Modèle de ECAO: Créer ou demander une empreinte ou un symbole PCB

Agrandir

Les images sont fournies à titre indicatif
Voir les caractéristiques du produit

Téléchargez gratuitement le chargeur de bibliothèque pour convertir ce fichier pour votre outil ECAD. En savoir plus sur le modèle ECAD.

En stock: 1

Stock: 1 Expédition possible immédiatement
Délai usine : 12 Semaines
Entrez la quantité: Minimum : 1 Multiples : 1
Acheter

Ce produit est expédié GRATUITEMENT

Prix (EUR)

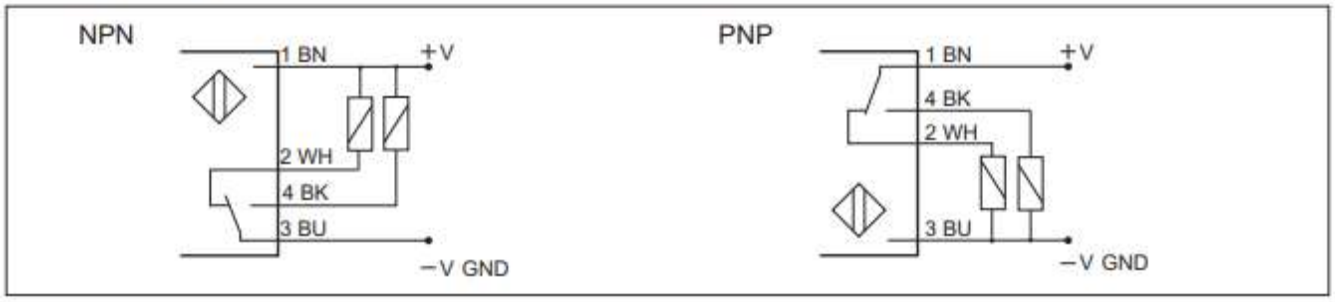
Qté.	Prix unitaire	Ext. Prix
1	65,77 €	65,77 €
10	59,99 €	599,90 €

Raccordement PA18CAD10

Il existe 2 versions de ce capteur,une avec une sortie de type PNP et l'autre de type NPN , le PNP lorsqu'il est activé met en contact le fil du signal et la tension ,la tension du fil de signal devient donc égal a celle du fil d'alimentation , cela a comme inconvénient de ne pas permettre de pouvoir choisir la tension de sortie .

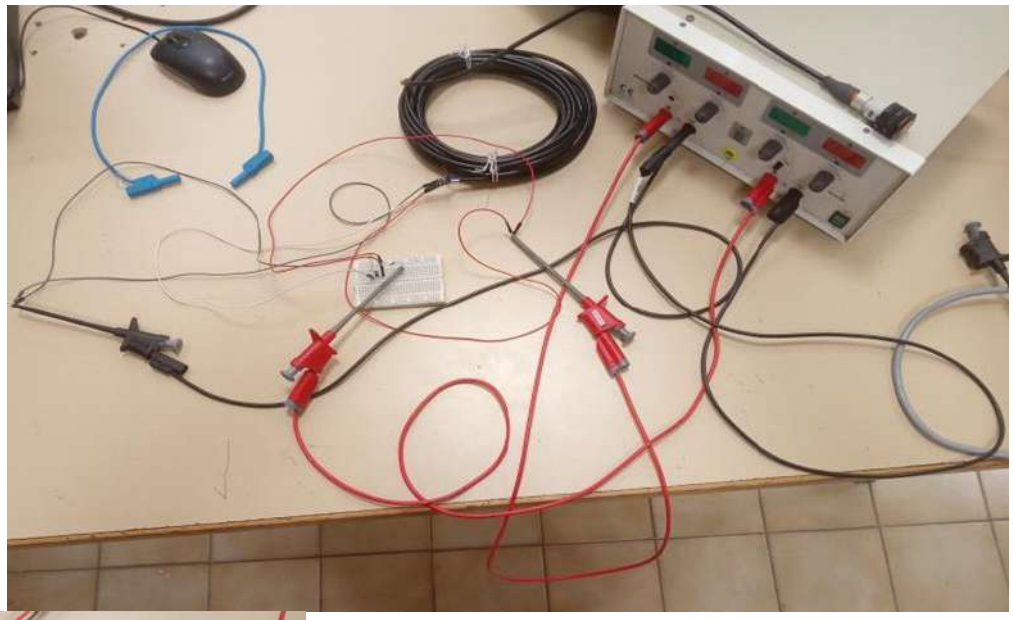
Le NPN lui , fonctionne à l'envers , Lorsque le capteur est actif, il met en contact le fil de sortie avec la masse. Ce qui veut dire qu'il n'injecte pas de courant dans le fil de sortie. Par conséquent, pour que le changement puisse être détecté, il faut que quelqu'un se charge de mettre de la tension sur le fil. C'est le rôle des résistances de Pull-up présente sur chacun des deux fils du signal.

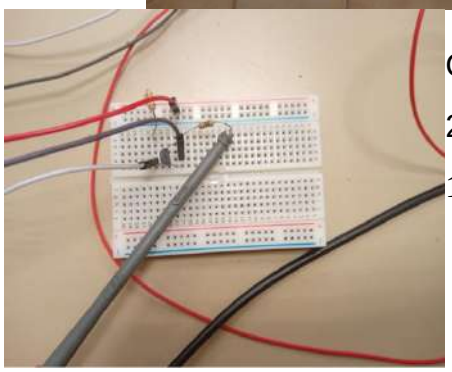
Wiring Diagrams



Câblage capteur PA18

Après cela, j'ai effectué le Câblage du capteur PA18CAD10NASA, avec émetteur/réflecteur intégré qui sera le modèle utilisé pour notre projet et qui sera par la suite raccordé sur les connecteurs de la carte électronique :





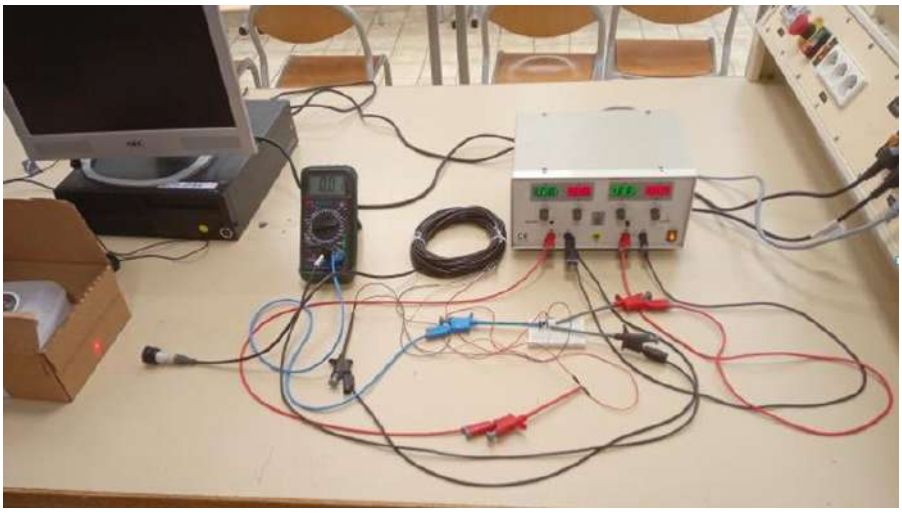
Comporte :

2 Résistances 10kΩ

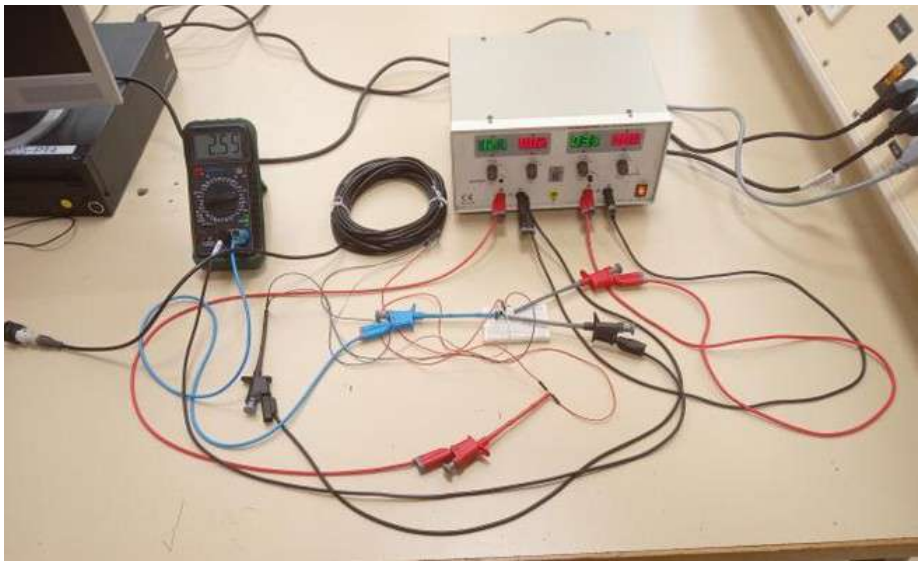
1 Transistor MOS Canal N

Zoom sur la platine d'essais

Fonctionnant avec un transistor MOSFET de canal N (Négatif), le capteur passe à l'état bas (soit à 0 V) et fonctionne en inverseur lorsqu'il détecte un objet/personne :

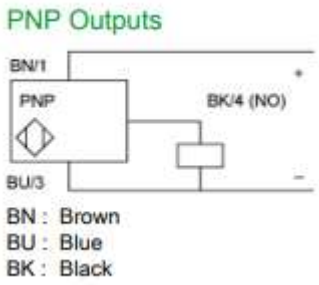
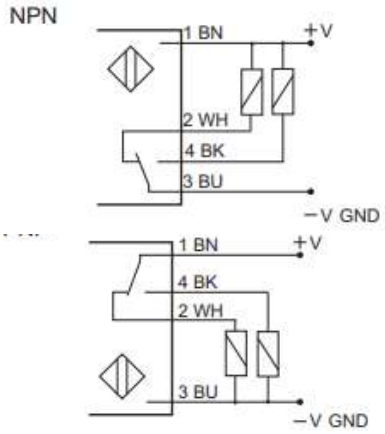


Tandis que si rien n'est détecté dans son périmètre, le capteur reste dans un état haut 3.3V, le fait que la tension soit de 2.5V sur le multimètre s'explique car le transistor n'est pas saturé correctement , je devrais donc réglé ce problème pour la revue 3.



Comparatif des capteurs

J'ai donc réalisé un tableau de comparaison de ces 2 capteurs pour savoir lequel était le plus pertinent à utiliser:

	Capteur photoélectrique XUB9APANL2	**Capteur Photoélectrique PA18CAD10**
		
Tension d'alimentation	12 à 24V	10 à 30V
Diagramme de câble		
Technologie	PNP	PNP ou NPN
Distance de détection	3m	1m
Réflecteur	Oui	Non
Délai de réponse	< 1ms	< 1ms
Prix	76,63€ TTC (One-Elec)	65,77€ TTC (Mouser)
Avantages / Inconvénients	✔ Mise en œuvre assez simple ✗ Nécessite un positionnement parfait du faisceau avec le réflecteur	✔ Réflexion directe sur les coureurs ✔ Deux modes de câblage possibles ✗ Absorbe la couleur noire

C’est pourquoi nous avons privilégié le capteur PA18CAD10, car le XUB est très contraignant a mettre en œuvre du au fait qu’il fonctionne avec un réflecteur devant être parfaitement alignés avec le capteur pour fonctionné contrairement au PA18 qui détectera la personne du moment qu’elle ce trouve dans le champs de détection mais aussi car il existe également deux versions de ce capteur , une en PNP et en NPN ce qui en cas de pénurie de connectique sur notre carte électrique, permet une plus grande flexibilité de mise en œuvre.

Schémas Partie Personnelle

Réalisation de la liste du HAT GPIO Raspberry Pi

Liste des GPIO du HAT Raspberry PI (Projet PMV)

Pour William RICHARD :

-GPIO17 : Capteur Couloir 1

-GPIO27 : Capteur Couloir 2

Pour Théo ROBILLARD :

-GPIO2 : SDA(Données)

-GPIO3 : SCL(Horloge)

Pour Nathan NOEL :

-GPIO18: Led ROUGE --GPIO22: Buzzer

-GPIO23: Led VERTE

Schema HAT RPI3 (2x10):

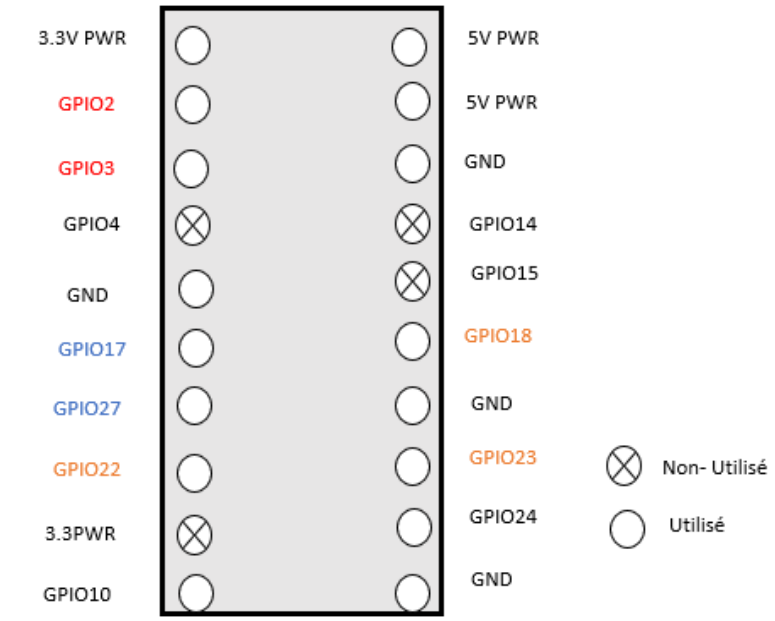
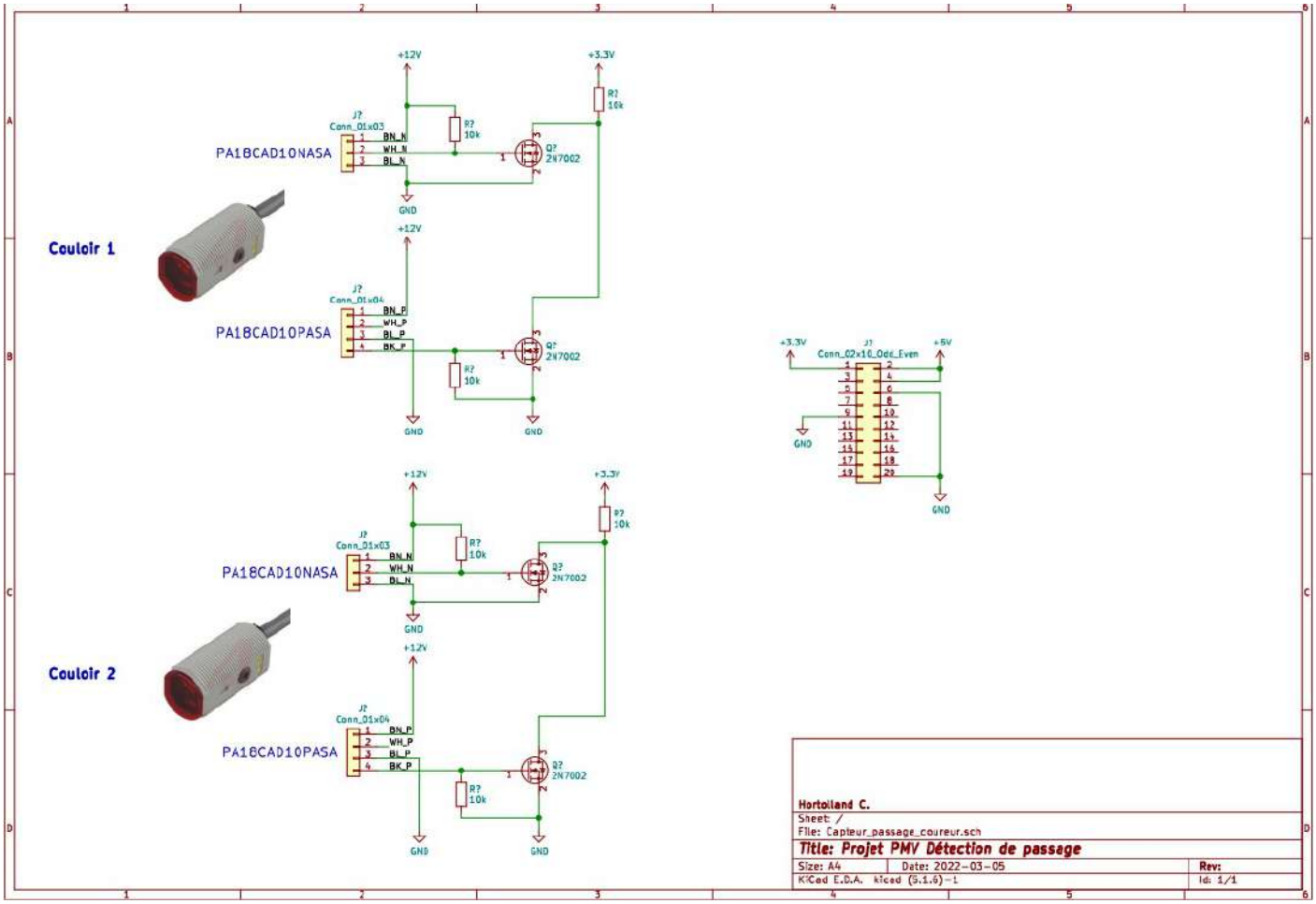


Schéma Kicad Individuel

Début de l'élaboration du schéma Kicad de ma partie personnelle sur la détection de passage à partir d'un modèle donné par mon professeur.



Ebauche de Schéma fournis par le Professeur

Après les modifications, voici le résultat final du schéma de ma partie personnelle comportant :

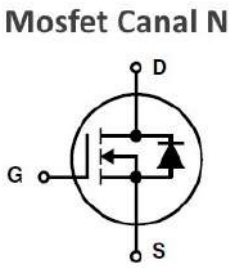
-Les 2 capteurs réfléchissant avec 2 connectiques différentes pour chacun des couloirs de la course

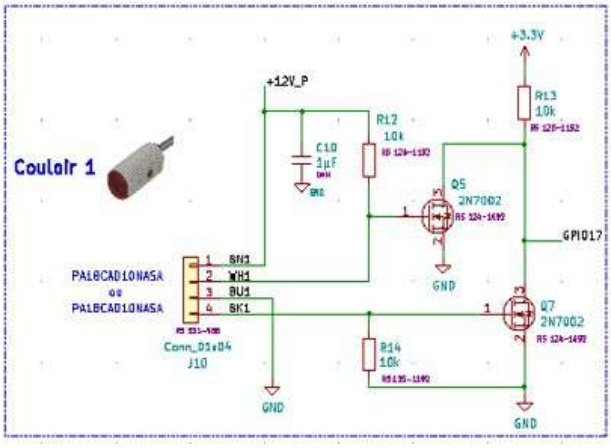
-Une horloge temps réel pour les communications I2C de l'anémomètre

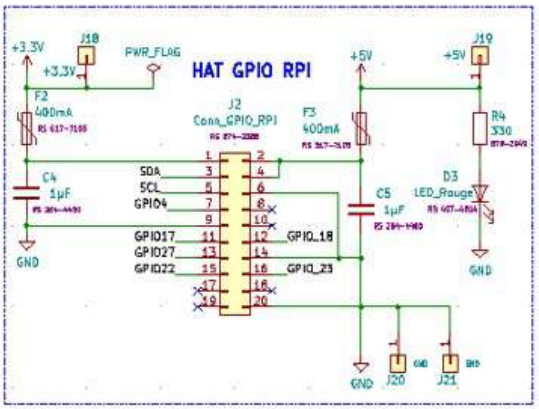
-Un HAT GPIO de Raspberry pi3 pour y connecter le tout.

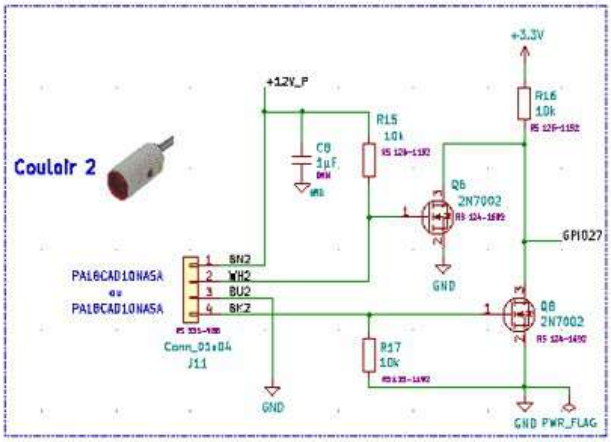
Schéma Kicad Individuel Final

Sur le schéma , on peut voir que mon capteur PA18 est branché a des transistors MOSFET de canal N , ces derniers fonctionnent avec une tension Vgs positive ce qui veut dire que plus la tension Vgs sera positive et plus le transistor MOSFET sera passant et le courant dans le drain sera de plus en plus fort devenant ainsi un interrupteur ouvert .









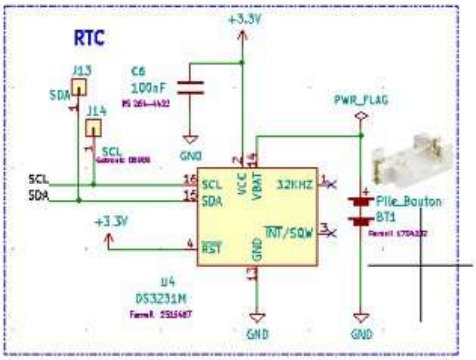


Schéma KiCad partie personnelle final

Ce qui permet de définir que la tension au borne de Vgs est nécessaire pour que le transistor deviennent passant est la tension de seuil ou en anglais Vth(Threshold) , cette valeur est fournis dans la documentation du transistor 2N7002:

infineon SN7002N

Thermal Characteristics

Parameter	Symbol	Values			Unit
		min.	typ.	max.	
Characteristics					
Thermal resistance, junction - ambient at minimal footprint	R_{thJA}	-	-	350	K/W

Electrical Characteristics, at T_j = 25 °C, unless otherwise specified

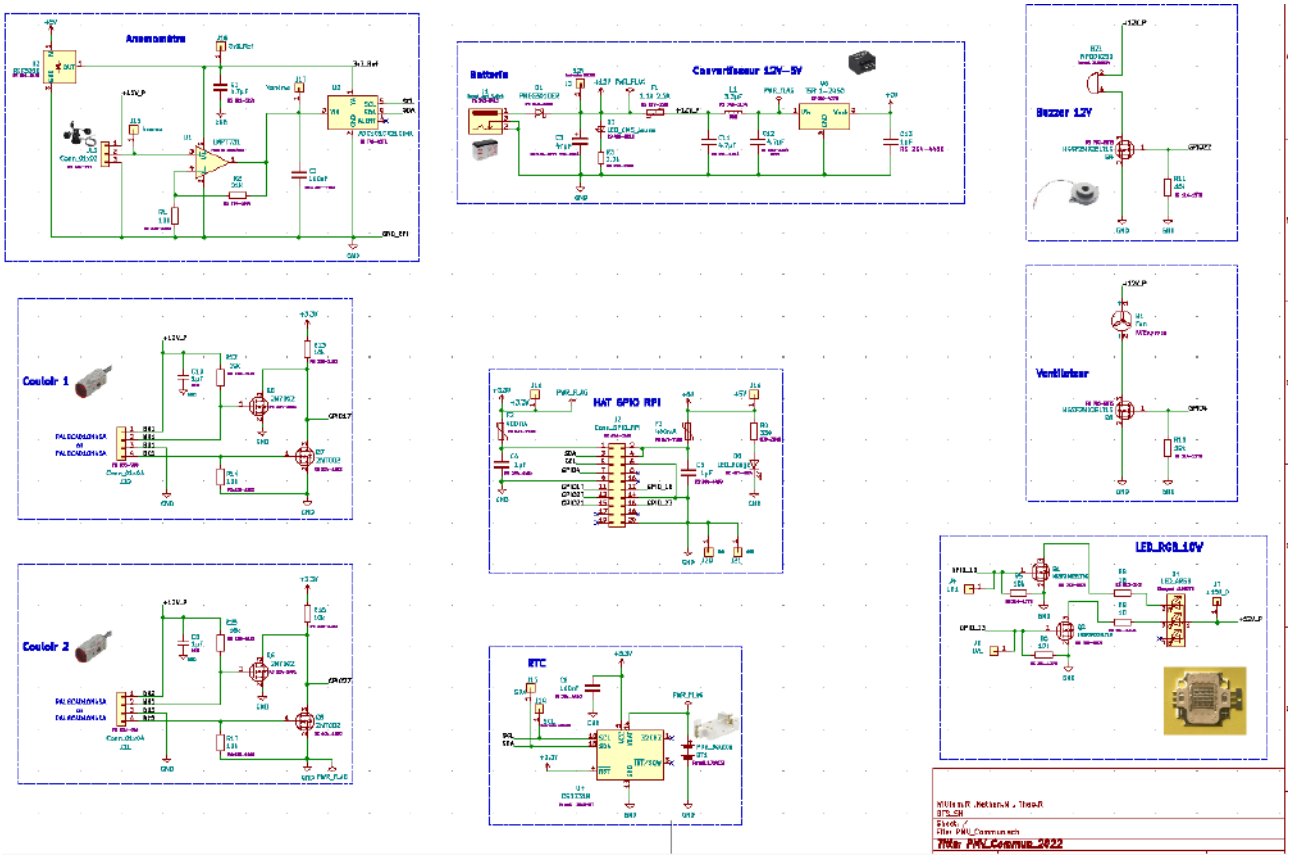
Parameter	Symbol	Values			Unit
		min.	typ.	max.	
Static Characteristics					
Drain-source breakdown voltage V_{GS}=0, I_D=250µA	$V_{(BR)DSS}$	60	-	-	V
Gate threshold voltage, V_{GS} = V_{DS} I_D=26µA	$V_{GS(th)}$	0.8	1.4	1.8	
Zero gate voltage drain current	I_{DSS}				µA
V_{DS}=60V, V_{GS}=0, T_j=25°C		-	-	0.1	
V_{DS}=60V, V_{GS}=0, T_j=150°C		-	-	5	
Gate-source leakage current V_{GS}=20V, V_{DS}=0	I_{GSS}	-	-	10	nA
Drain-source on-state resistance V_{GS}=4.5V, I_D=0.17A	$R_{DS(on)}$	-	3.9	7.5	Ω
Drain-source on-state resistance V_{GS}=10V, I_D=0.5A	$R_{DS(on)}$	-	2.5	5	

on prend la tension min Vth=0,8V car on doit toujours prévoir le pire cas pour que malgré cela il soit tout de même passant .

Schéma Kicad Commun

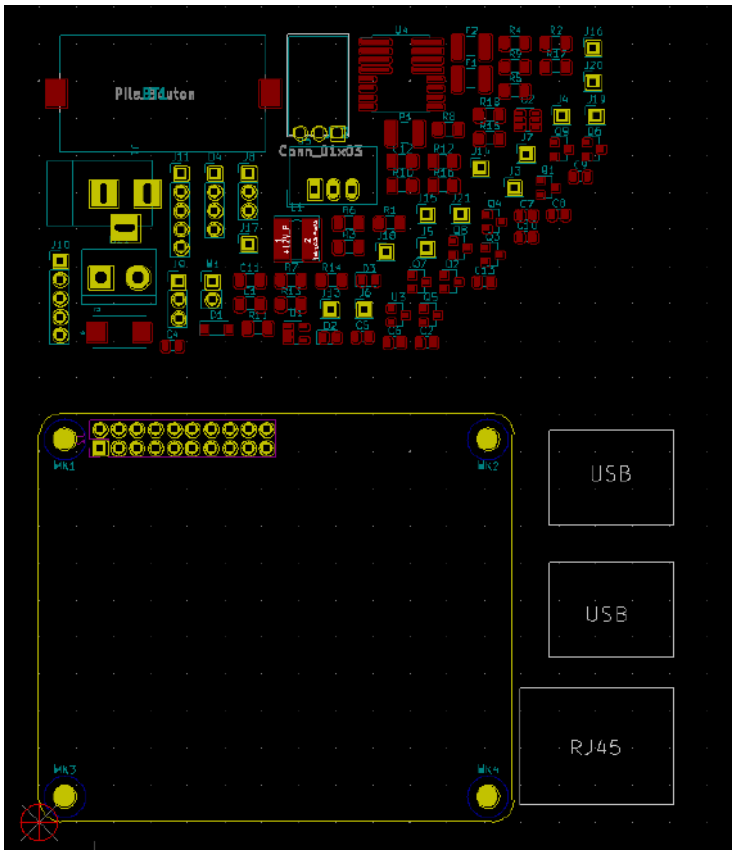
Peu après, lorsque nous avions tout les trois terminés les schémas, nous les avons mis en commun en adaptant les labels pour une meilleur compréhension des composants et mis à jour quelque codes commandes

Fusion des schémas individuelle des 3 élèves d'EC en vue du routage de la carte électronique:



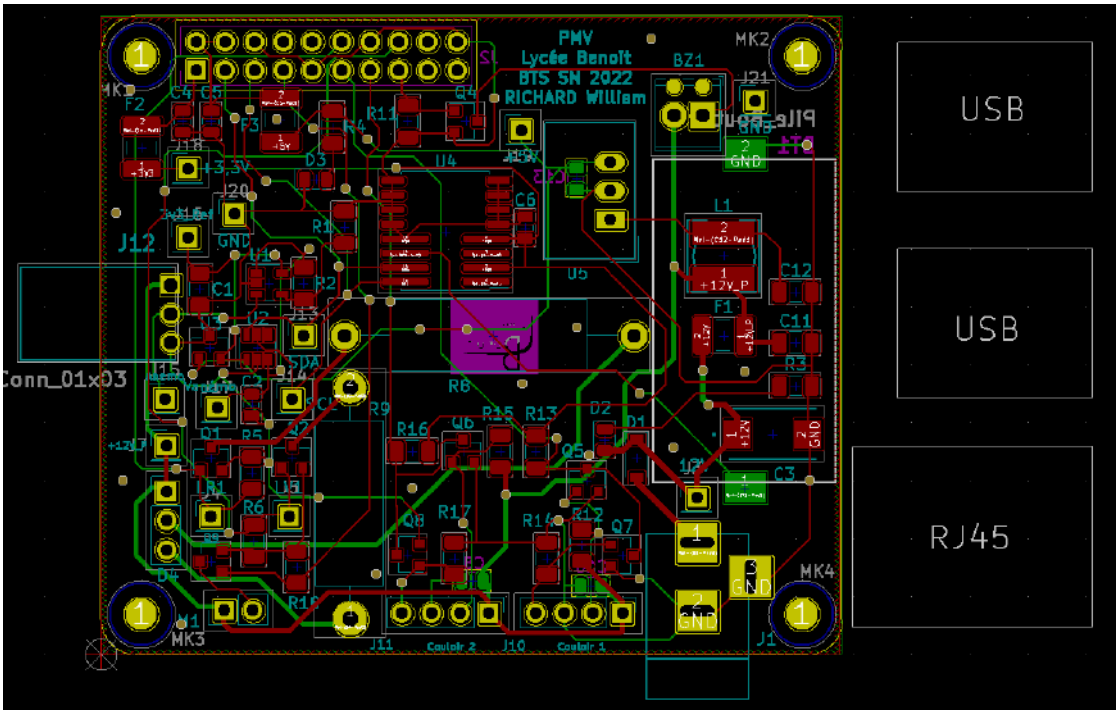
Routage Carte Kicad

Début du routage de la carte sur Kicad avec placement des composants par partie, définition des contraintes de places pour la pile bouton placés à l'arrière de la carte :

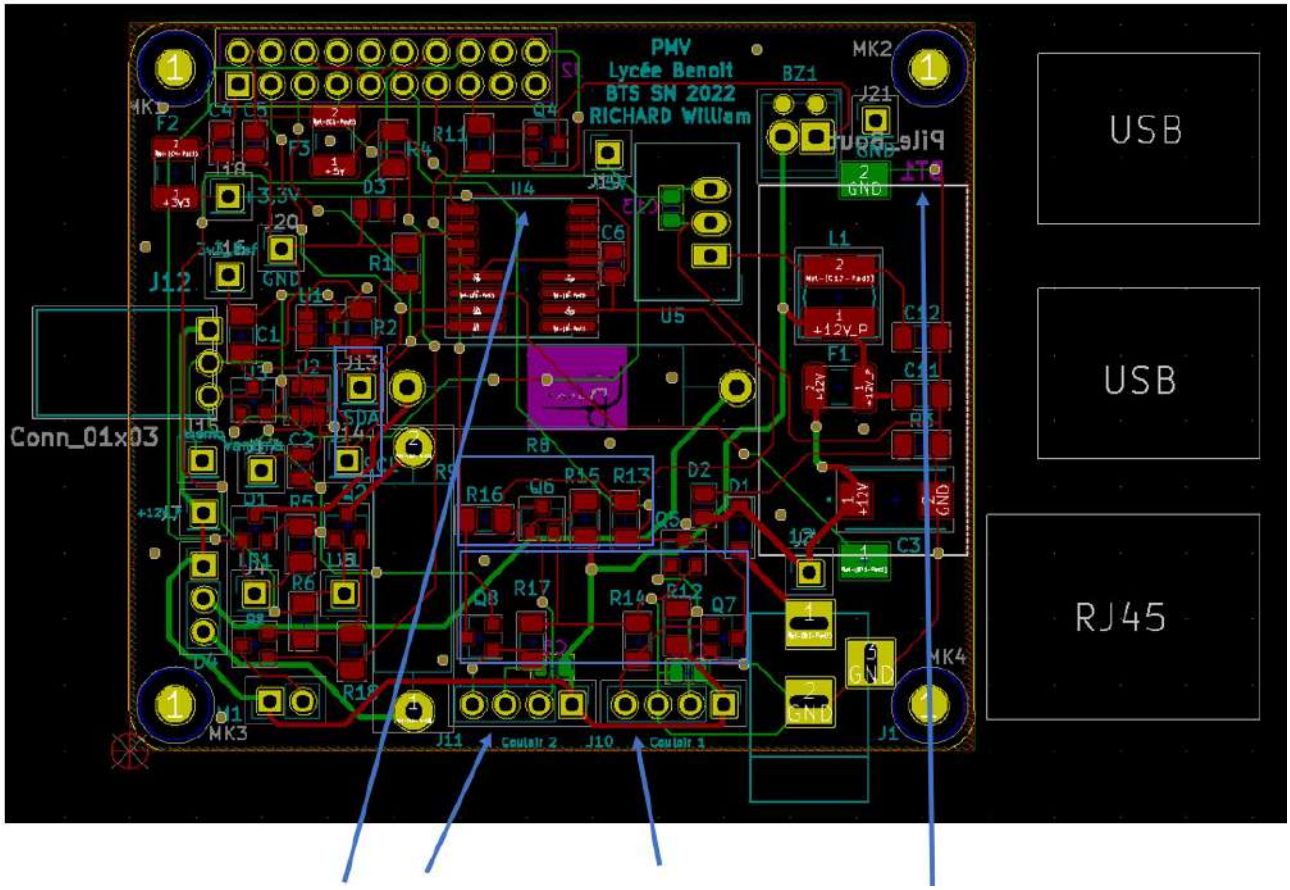


Carte de départ

Voilà le résultat après le placement des composants, des liaisons électriques et du plan de masse :



Analyse de ma partie de la carte



1) J10 & J11 : Connecteur 4 broches capteur infra rouge

2)U4: Horloge Temps Réel

3)BT1: Support Pile Bouton

4)Q5 à Q8: Transistor Mosfet Canal N

5) R12 à R17: Résistances 10k

6)C8 & C10: Condensateur 1uF

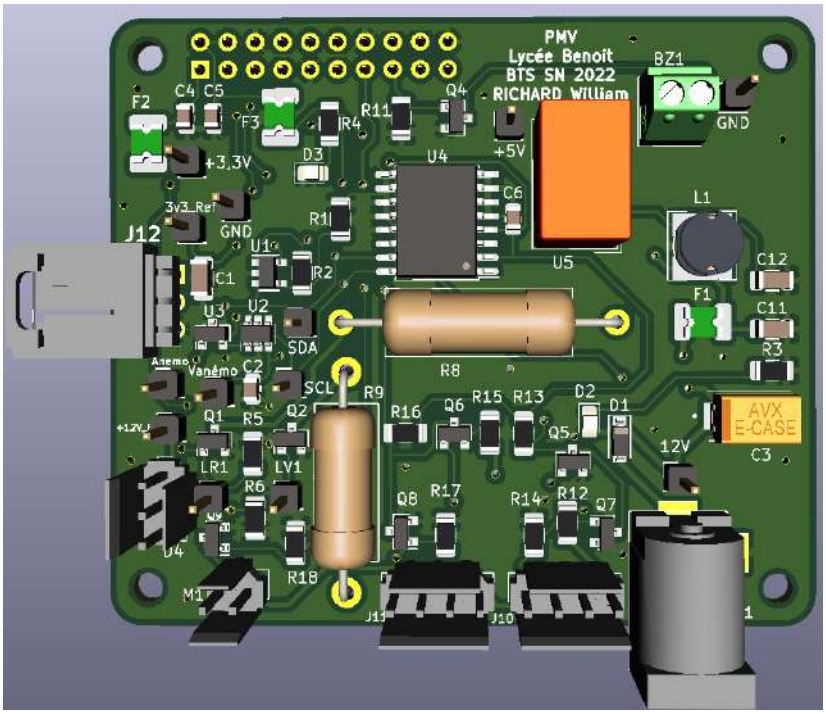
7)C6: Condensateur 100nF

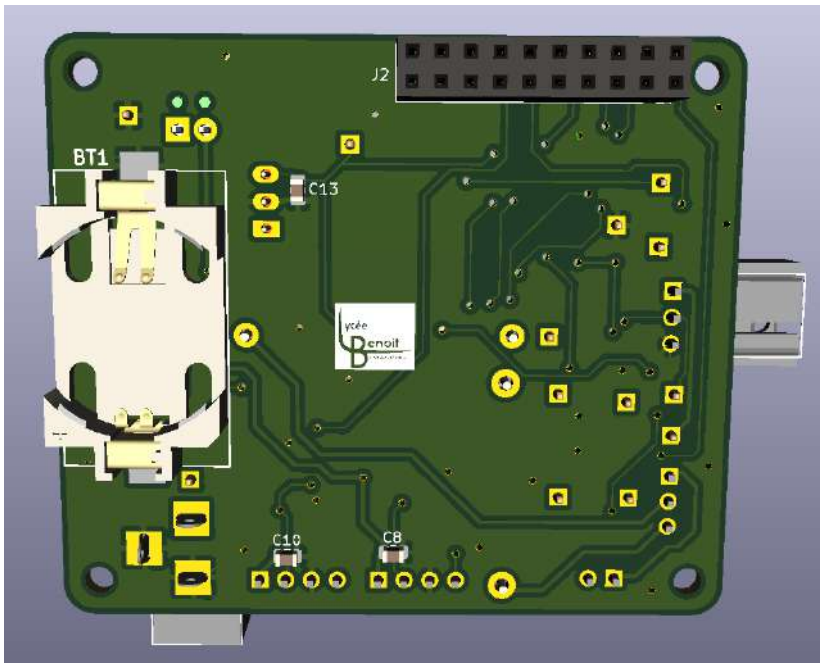
8)J13: Pointe de Test SDA (Data)

9)J14: Pointe de Test SCL (Clock)

Visualisation 3D Kicad

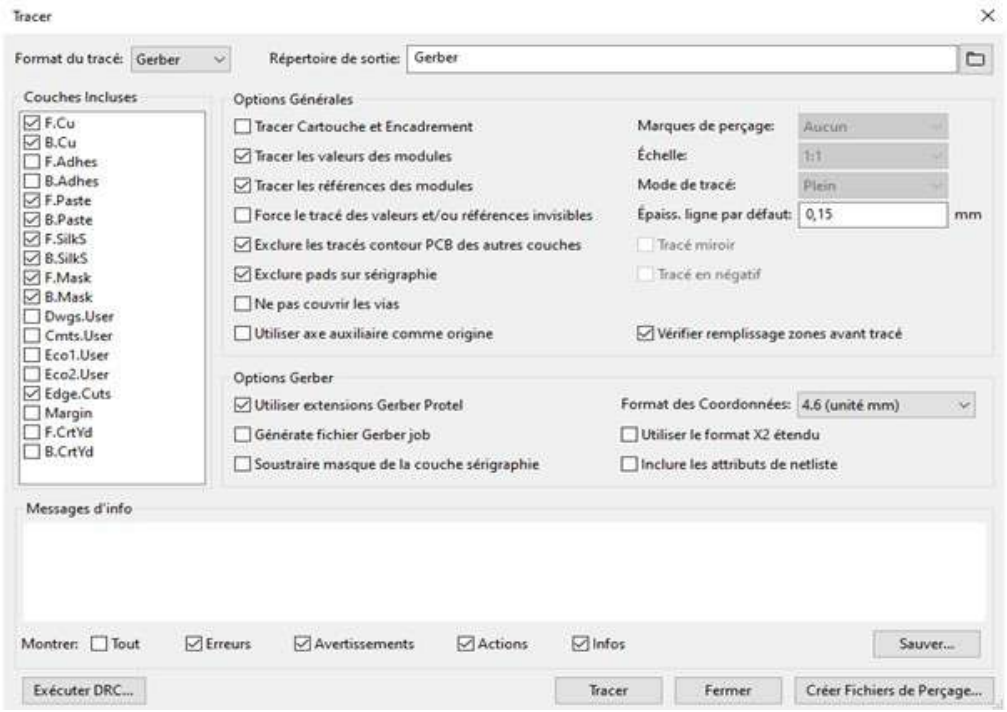
Voici ce que cela donne sur la visualisation 3D (de face et de dos):

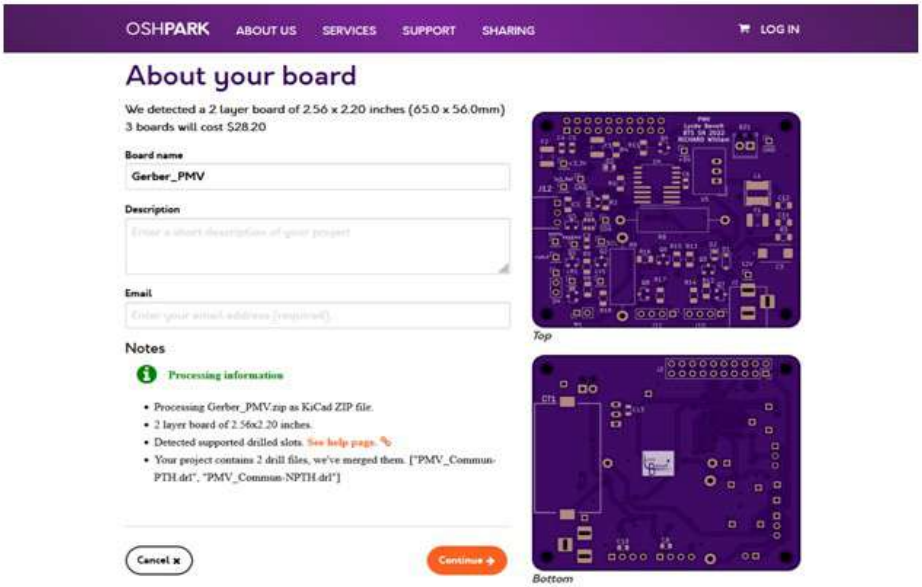




Génération fichier Gerber

J’ai ensuite généré les fichiers gerber et les ai testés sur OSHPark , un site de fabricant de PCB qui permet de déterminer si la carte est conforme ainsi qu’une estimation du prix de la carte

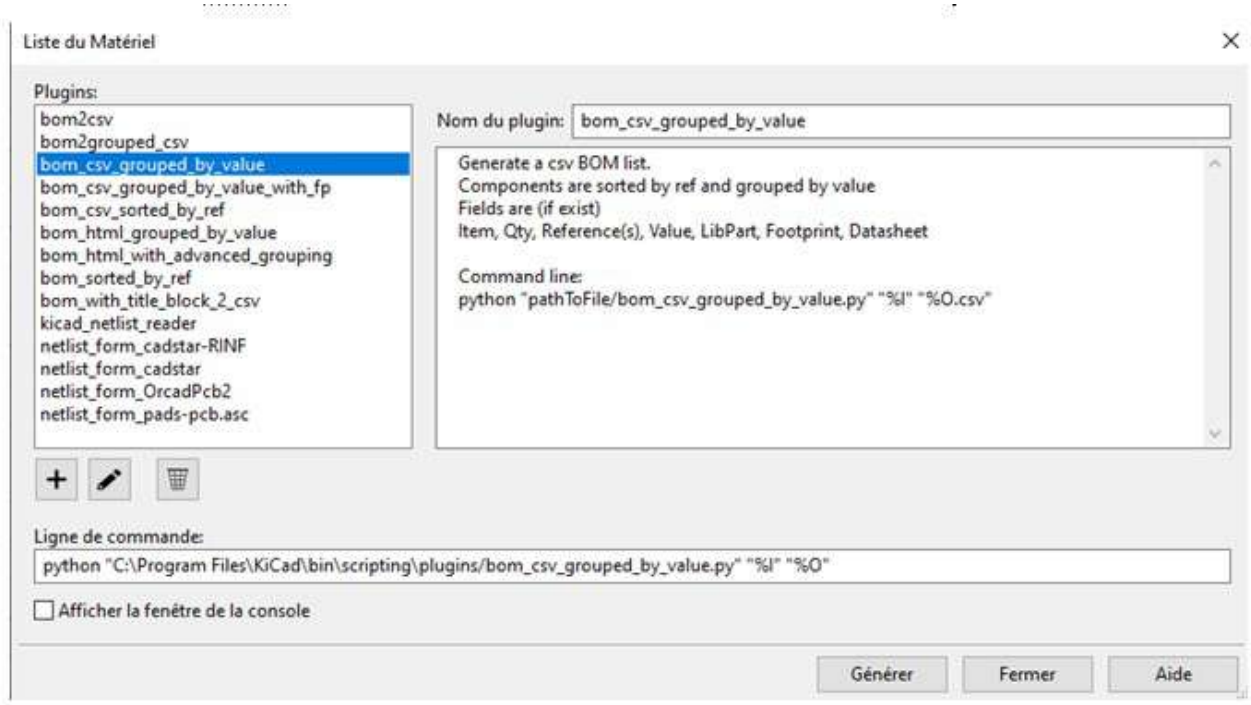

Tracer
Format du tracé: Gerber
Répertoire de sortie: Gerber
Couches Incluses
F.Cu
B.Cu
F.Adhes
B.Adhes
F.Paste
B.Paste
F.SilkS
B.SilkS
F.Mask
B.Mask
Dwgs.User
Cmts.User
Eco1.User
Eco2.User
Edge.Cuts
Margin
F.CrtYd
B.CrtYd
Options Générales
Tracer Cartouche et Encadrement
Tracer les valeurs des modules
Tracer les références des modules
Force le tracé des valeurs et/ou références invisibles
Exclure les tracés contour PCB des autres couches
Exclure pads sur sérigraphie
Ne pas couvrir les vias
Utiliser axe auxiliaire comme origine
Marques de perçage: Aucun
Échelle: 1:1
Mode de tracé: Plein
Épaiss. ligne par défaut: 0,15 mm
Tracé miroir
Tracé en négatif
Vérifier remplissage zones avant tracé
Options Gerber
Utiliser extensions Gerber Protel
Générate fichier Gerber job
Soustraire masque de la couche sérigraphie
Format des Coordonnées: 4.6 (unité mm)
Utiliser le format X2 étendu
Inclure les attributs de netliste
Messages d'info
Montrer: Tout Erreurs Avertissements Actions Infos
Sauver...
Exécuter DRC...
Tracer
Fermer
Créer Fichiers de Perçage...


OSHPARK
ABOUT US
SERVICES
SUPPORT
SHARING
LOG IN
About your board
We detected a 2 layer board of 2.56 x 2.20 inches (65.0 x 56.0mm)
3 boards will cost $28.20
Board name
Gerber_PMV
Description
Email
Notes
Processing information
Processing Gerber_PMV.zip as KiCad ZIP file.
2 layer board of 2.56x2.20 inches.
Detected supported drilled slots. See help page.
Your project contains 2 drill files, we've merged them. ["PMV_Commun-PTH.drl", "PMV_Commun-NPTH.drl"]
Cancel
Continue
Top
Bottom

Réalisation carte électronique

Liste de composant:

J'ai aussi commencé à générer la liste de matériel comportant les éléments suivant (Nom, Quantité, Référence, Code Commande et Prix)



Par la suite, j'ai réalisé la nomenclature pour y faire une estimation potentielle

du prix des composants mais aussi pour faire notre liste de matériel avec les composants qui nous ont étaient distribués, permettant de les repérer plus facilement :

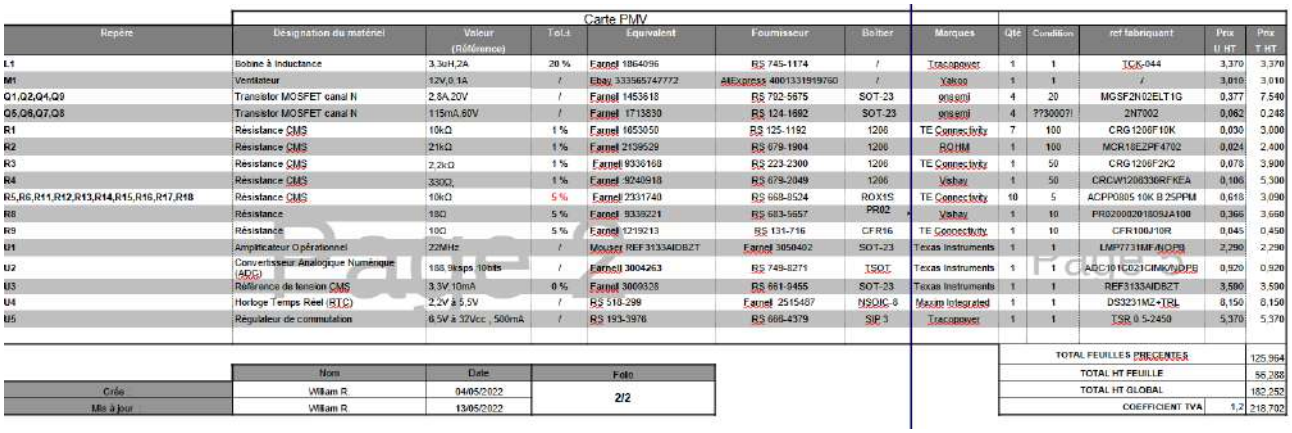
Nomenclature

Carte PMV

Repère	Désignation du matériel	Valeur (Référence)	Tol.±	Equivalent	Fournisseur	Boîtier	Marques	Qté	Condition	Ref fabriquant	Prix U HT	Prix T HT
BT1	Support Pile Bouton	20mm	/	RS 367-4222	Farnell 1704232	/	Keystone	1	1	1060	2,850	2,850
BZ1	Buzzer	6-12VDC 30mA,110dB	/	Gotronic 05460	Farnell 3108924	/	Multicomp Pro	1	1	MP000293	3,920	3,920
C1,C11,C12	Condensateur CMS	4.7uF 50VDC	10 %	Farnel 1908177	RS 691-1224	1206	KEMET	3	10	C1206C475K5PACTU	1.102	11,020
C2,C6	Condensateur CMS	100nF 50VDC	20 %	Farnel 1759037	RS 264-4422	EIA 0805	KEMET	1	25	C0805C104M5UACTU	0.140	3,500
C3	Condensateur CMS	47uF 20VDC	10-20%	RS 135-9976	RS 135-9094	2917	Kyocera AVX	1	5	TPSD476K025R0250	2.292	11,460
C4,C5,C8,C10,C13	Condensateur CMS	1uF 16VDC	(-20)-80%	Farnell 1865556	RS 264-4450	EIA 0805	KEMET	5	25	C0805C105Z4VACTU	0.042	1,05
D1	Diode CMS schottky	1A 30V	/	Farnell 3809974RL	RS 816-6855	SOD-123	Nexperia	1	50	PMEG3010ER,115	0,317	15,850
D2	LED CMS Jaune	2.1V,	/	Farnell 2099236	RS 486-0519	Boîtier 0805	Broadcom	1	25	HSMY-C170	0.340	8,500
D3	LED CMS Rouge	2V	/	Farnell 2099245	RS 497-4884	805	ams OSRAM	1	5	LS R976	0.278	1,390
D4	LED ARGB	10W,300mA,12V	/	Aliexpress:4000166221143	Banggood 1168379	/	ZX LED LIGHT	1	1	/	6,144	6,144
F1	Fusible Réarmable CMS	400mA,24V	/	Farnell 1345825	RS 817-1668	/	Bourns	1	10	MF-MSMF110/24X-2	0.390	3,900
F2,F3	Fusible Réarmable CMS	400mA,30V	/	Farnell 1596897	RS 517-7105	/	Littelfuse	1	10	MINISMDC020F-2	0.279	2,790
J1	Prise D'alimentation C.C	5A,12V,14,5mm	/	Farnell 1854512	RS 143-3915	/	RS PRO	2	5	/	2.844	14,220
J2	Connecteur femelle 2x20	2.54mm	/	Farnell 2311675	RS 674-2365	/	ASSMANN WSW	1	5	A-BL254-DG-G20D	1.430	7,150
J5,J4,J6,J7,J13,J14,J15,J16,J17,J18,J19,J20,J21	Connecteur HE14 MN180 1x36	2.54mm Imax:3A	/	RS 547-3166	Gotronic 08000	/	MPE Garry GmbH	12	36	M20-9773646	1.320	15,840
J10,J11	Connecteur femelle 1x04 Vertical	2.54mm	/	RS 163-8393	RS 531-986	/	TE Connectivity	2	10	281685-4	0.899	8,990
J12	Connecteur femelle 1x03 Vertical	2.54mm	/	Mouser 571-281685-3	RS 531-970	/	TE Connectivity	1	10	281685-3	0.739	7,390

	Nom	Date	Folio
Crée	Willam R.	04/05/2022	1/2
Mis à jour	Willam R.	13/05/2022	

TOTAL HT FEUILLE		125.964
TOTAL HT GLOBAL		125.964
COEFFICIENT TVA	1.2	151.157



Carte PMV

Repère	Désignation du matériel	Valeur (Référence)	Tol.±	Equivalent	Fournisseur	Boîtier	Marques	Qté	Condition	ref fabriquant	Prix U HT	Prix T HT
L1	Bobine à inductance	3,3uH,2A	20 %	Farnel 1864096	RS 745-1174	/	Tracopower	1	1	TCK-044	3,370	3,370
M1	Ventilateur	12V,0,1A	/	Ebay 333565747772	AliExpress 4001331919760	/	Yakoo	1	1	/	3,010	3,010
Q1,Q2,Q4,Q9	Transistor MOSFET canal N	2,8A,20V	/	Farnel 1453618	RS 792-5675	SOT-23	onsemi	4	20	MGSF2N02ELT1G	0,377	7,540
Q5,Q6,Q7,Q8	Transistor MOSFET canal N	115mA,60V	/	Farnel 1713830	RS 124-1692	SOT-23	onsemi	4	???0007!	2N7002	0,062	0,248
R1	Resistance CMS	10kΩ	1 %	Farnel 1653050	RS 125-1192	1206	TE Connectivity	7	100	CRG1206F10K	0,030	3,000
R2	Resistance CMS	21kΩ	1 %	Farnel 2139529	RS 679-1904	1206	ROHM	1	100	MCR18E2PF4702	0,024	2,400
R3	Resistance CMS	2,2kΩ	1 %	Farnel 9336168	RS 223-2300	1206	TE Connectivity	1	50	CRG1206F2K2	0,078	3,900
R4	Resistance CMS	330Ω	1 %	Farnel 9240918	RS 679-2049	1206	Vishay	1	50	CRCW1206330RFKEA	0,106	5,300
R5,R6,R11,R12,R13,R14,R15,R16,R17,R18	Resistance CMS	10kΩ	5 %	Farnel 2331740	RS 668-8524	ROX1S	TE Connectivity	10	5	ACPP0805 10K B 25PPM	0,618	3,090
R8	Résistance	18Ω	5 %	Farnel 9338221	RS 683-5657	PR02	Vishay	1	10	PR02000201809JA100	0,366	3,660
R9	Résistance	10Ω	5 %	Farnel 1219213	RS 131-716	CFR16	TE Connectivity	1	10	CFR100J10R	0,045	0,450
U1	Amplificateur Opérationnel	22MHz	/	Mouser REF3133AIDBZT	Farnel 3050402	SOT-23	Texas Instruments	1	1	LMP7731MF/NOPB	2,290	2,290
U2	Convertisseur Analogique Numérique (ADC)	188,9ksps,16bits	/	Farnell 3004263	RS 749-8271	TSOT	Texas Instruments	1	1	ADC161C021CIMK/NOPB	0,920	0,920
U3	Référence de tension CMS	3,3V,10mA	0 %	Farnel 3009328	RS 661-9455	SOT-23	Texas Instruments	1	1	REF3133AIDBZT	3,500	3,500
U4	Horloge Temps Réel (RTC)	2,2V à 5,5V	/	RS 510-299	Farnel 2515467	NSOIC-8	Maxim Integrated	1	1	DS3231MZ+TRL	8,150	8,150
U5	Régulateur de commutation	6,5V à 32Vcc ; 500mA	/	RS 193-3976	RS 668-4379	SIP 3	Tracopower	1	1	TSR 0.5-2450	5,370	5,370

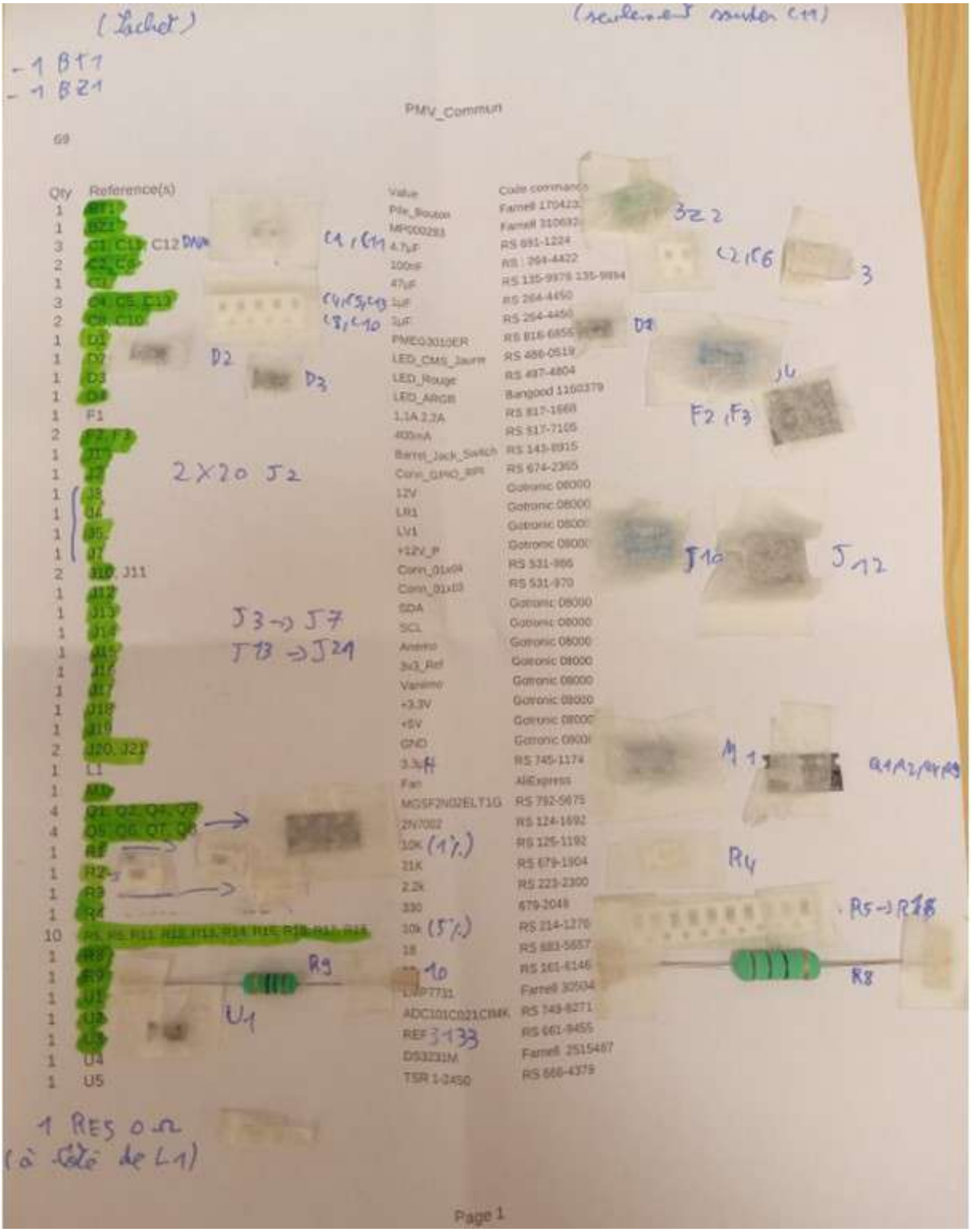
	Nom	Date	Folio
Crée	William R.	04/05/2022	2/2
Mis à jour	William R.	13/05/2022	

TOTAL FEUILLES PRECEDENTES		125,964
TOTAL HT FEUILLE		56,288
TOTAL HT GLOBAL		182,252
COEFFICIENT TVA	1,2	218,702

Distribution des composants

Voici la fiche avec laquelle nous avons fait la distribution de composant avec le professeur , ces derniers ont été collés pour qu'on ne les perdent pas et qu'on puisse y accéder directement lorsque l'on passerai au soudage des composants.

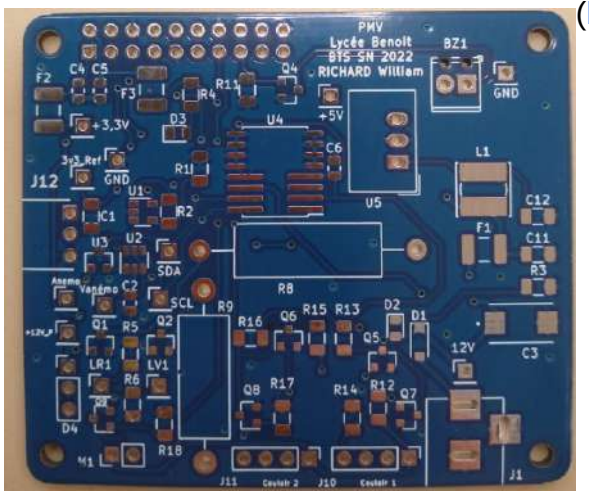
A noter que nous n'avons pas reçu tout les composants directement et que ceux n'étant par surligner sur la fiche ont été distribués par la suite

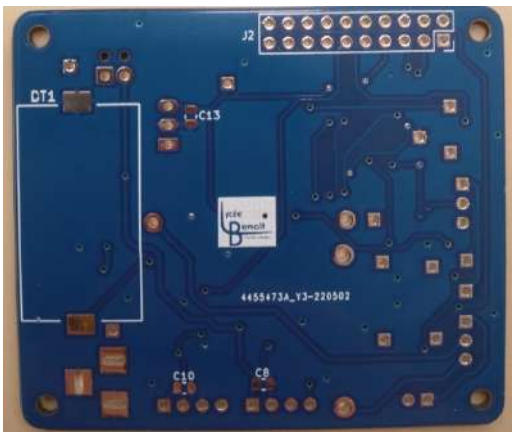

(Sachet)
- 1 BT1
- 1 BZ1
PMV_Commun
Qty Reference(s)
2X20 J2
J3 -> J7
J13 -> J21
J10
J12
R9
U1
R4
R5 -> R8
R8
1 RES 0 Ω
(à côté de L1)
Page 1

Réalisation carte électronique

Pour la réalisation de la carte électronique, nous avons choisis une couleur de carte différente, ayant uniquement pour but de différencier les cartes entre-elles qui se ressemblent déjà beaucoup de par l'utilisation des mêmes composants:

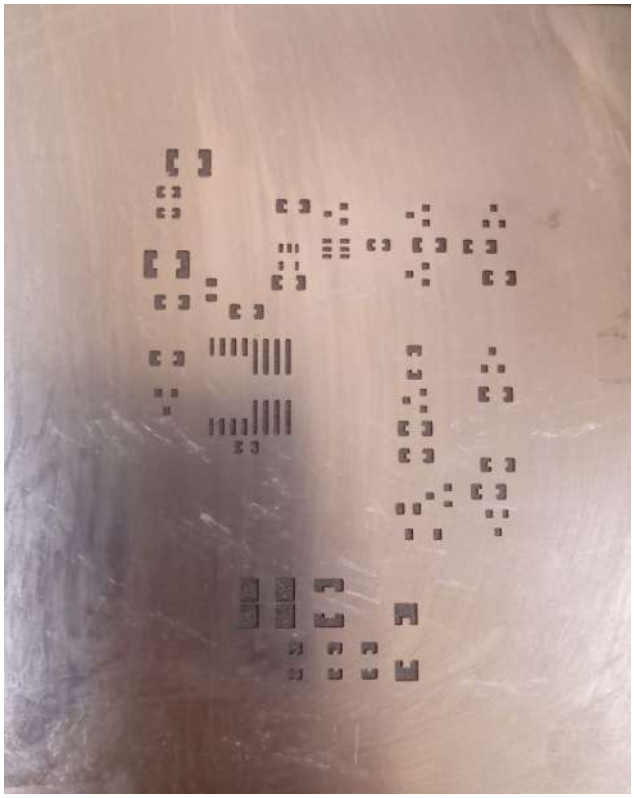
(bleu:William ,Rouge:Théo, Vert:Nathan)



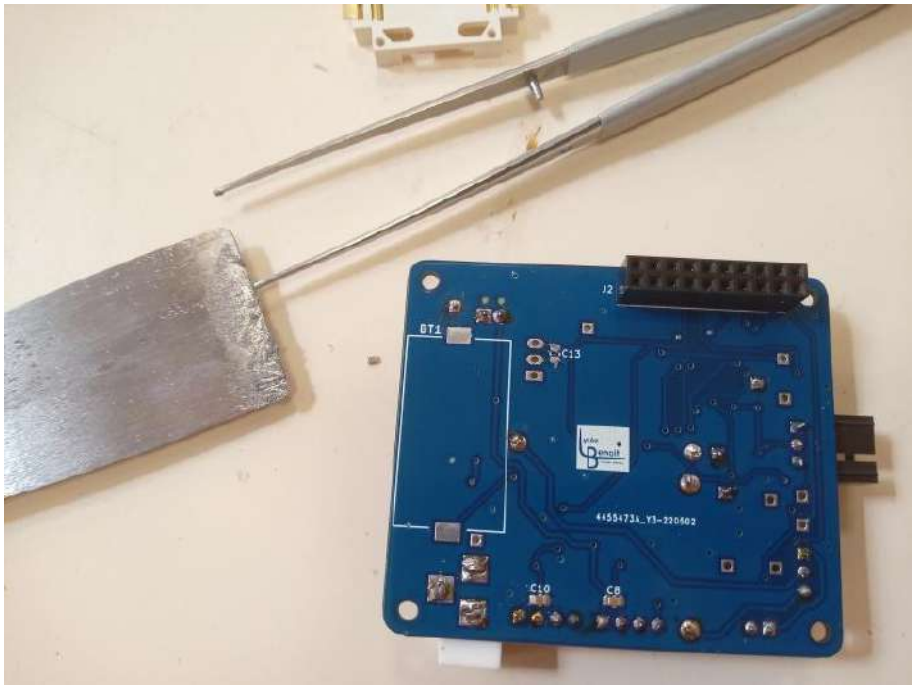


Placement du Stencil

Premièrement j'ai posé un pochoir (ou stencil), c'est une feuille métallique imprimable percé de sorte a ce que l'on puisse ensuite disposer de la pâte à braser sur les broches sans déborder afin d'y mettre par la suite les composants CMS qui sont des composants miniatures et non traversant.



Pour les placer, on vient déposer de la pâte à braser sur les broches de nos CMS



Chauffage dans le Four a Refusion

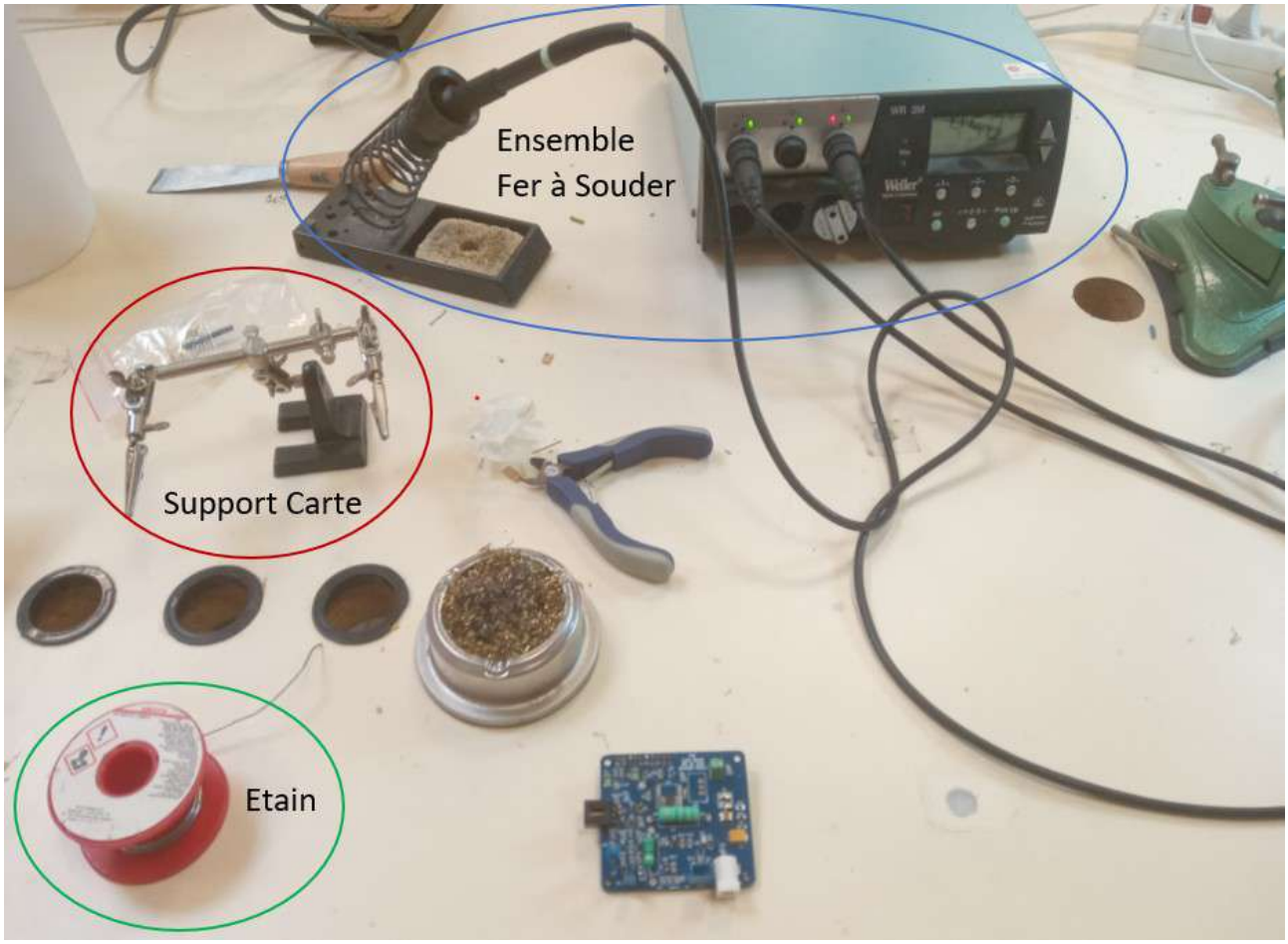
Une fois cela fait, les composants sont chauffés dans un four a refusion pendant une durée de 2m50 et a une température variant de 95 à 275 °C lors du chauffage de la carte, la pâte se solidifie et fixe les composants à leur broche.

Soudage a l’air Chaud

Pour ce qui est des 3 condensateurs CMS qui sont placés à l’arrière de ma carte et pour donc éviter de réchauffer l’entièreté de sa surface pour seulement trois composants, j’ai utilisé la méthode de la soudure a l’air chaud à l’aide de l’appareil ci-dessous, elle a l’avantage de pouvoir chauffer des endroits localisés , idéal donc lorsque l’on dispose de peu de composant à souder comme ici

Soudure a l’Étain

Quand tout cela est finis, on s’occupe de souder les composants traversants tel que les connectiques à broches ou seront par la suite branchés les capteurs de détection, anémomètre et buzzer avec de l’étain qui est le matériau utilisé ici:



A noter que peut après cela ,nous avons remarqué que la valeur de la résistance R1 n'était pas la bonne , nous l'avons donc changée pour une résistance de 10kΩ .

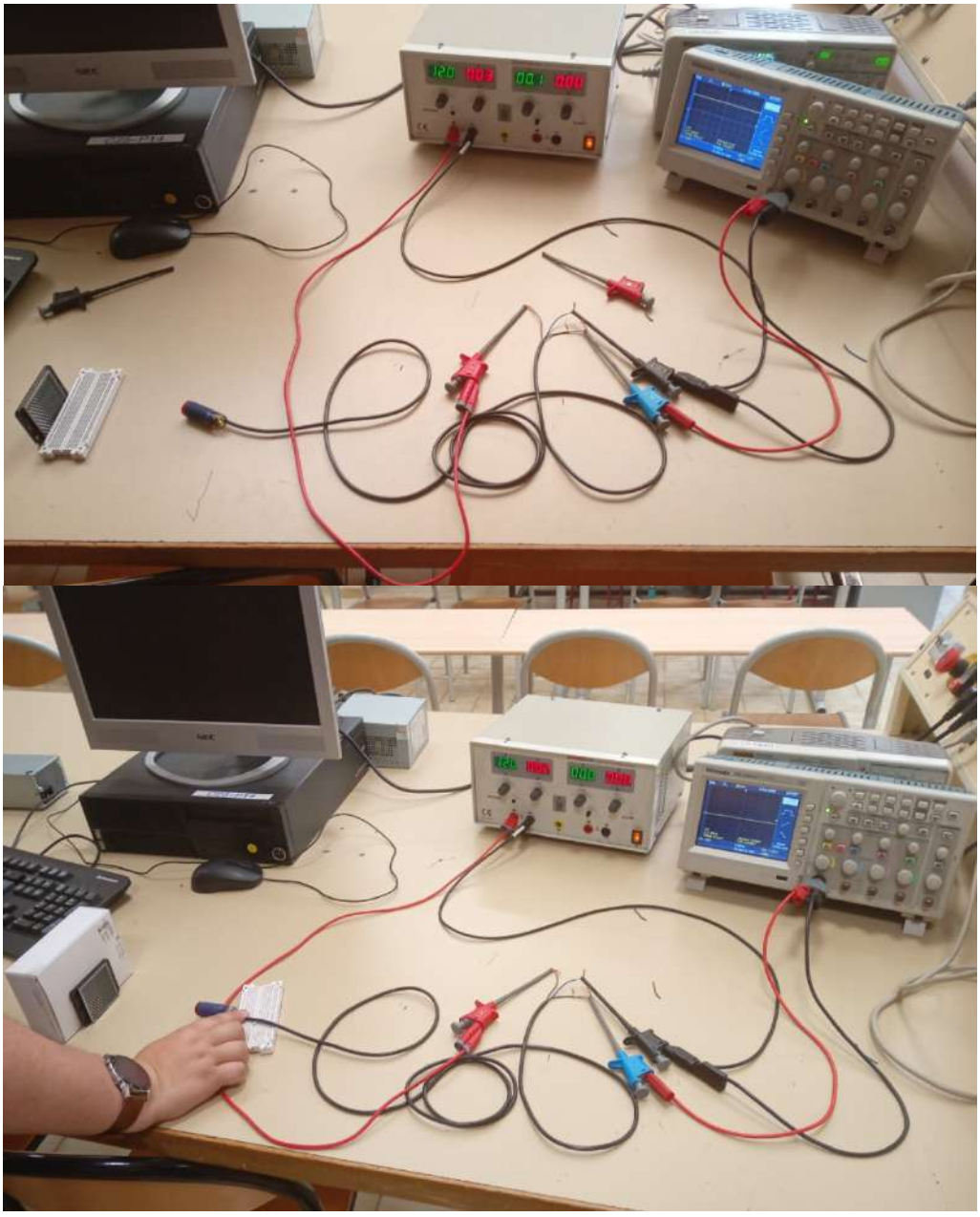
Nettoyage des broches

Pour finir, on nettoie les différentes broches ou l'on a soudé avec de l'alcool isopropylique pour y enlever les bavures d'étains entre deux broches et ainsi éviter les court circuit.

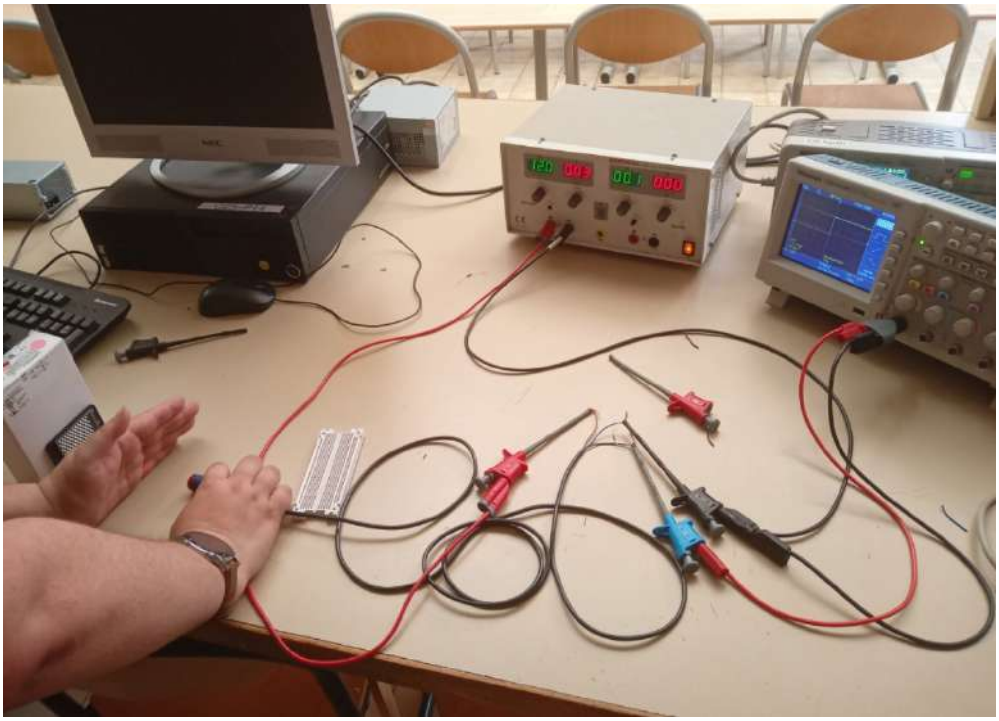
Partie Physique:

Étude de trame capteur XUB

J'ai ensuite été amené à la demande du professeur de physique à étudier le fonctionnement de la trame du capteur photoélectrique XUB9APANL2 ,pour cela je l'ai branché en suivant le schéma de raccordement



Trame état bas, pas de détection



Trame état haut, passage d'un coureur (Signal à +12V)

Ensuite, j'ai déterminé si le capteur était comme le disait la documentation, capable de détecter le passage d'un coureur sur la distance d'un couloir d'athlétisme soit 1.22m je l'ai donc placé a cette distance pour effectuer les tests suivant:

L’oscilloscope est bien à l’état bas (0V) signifiant que le réflecteur (à gauche) renvoie bien le signal infra-rouge du capteur qui ne détecte pas d’obstacle et cela a une distance de 1,22m, le capteur respecte donc le cahier des charges

Le réflecteur étant vraiment contraignant a mettre en œuvre de par le fait qu’il nécessite un alignement parfait avec le capteur pour la détection et donc que au plus la distance est élevée, au plus cela devient difficile a calibrer.

Colorimétrie

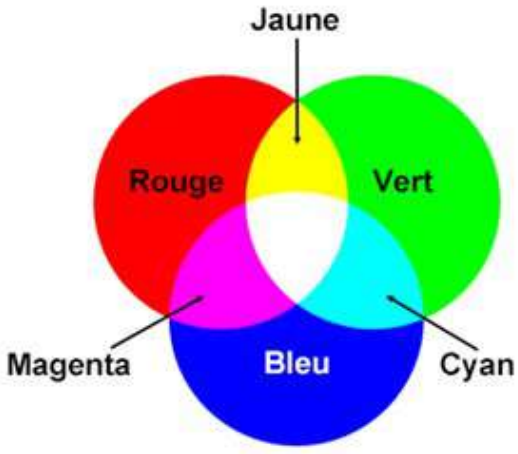
Mises en Contexte:

Lors d'un test commun IR & EC avec les présences des professeurs, nous avons testés le modèle plug du capteur PA18 en attendant de recevoir la version câble que j'ai mis en œuvre ,à savoir le PA18CAD10NAM1SA , nous nous sommes rendus compte lors des tests que lorsque Julien est passés devant le capteur avec son survêtement de couleur noir , l'infrarouge ne le détecté pas , après une bonne vingtaines de minutes de questionnement , nous en sommes venu a la conclusion que cela venait du fait que la couleur noire absorbée le faisceau infra-rouge .

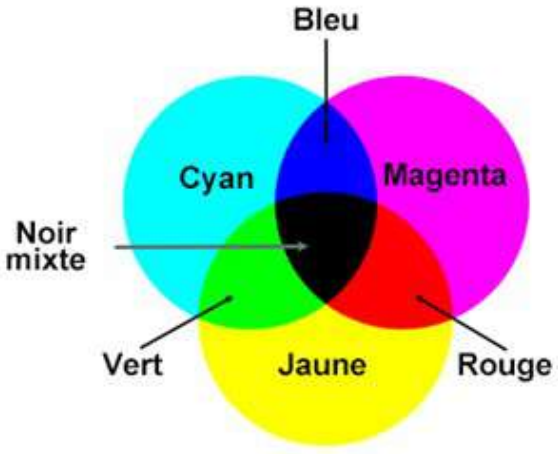
Explication :

La manière dont nous voyons les couleurs dépend de la longueur d'onde que notre œil perçoit , la longueur d'onde qu'un objet reflète définis ainsi sa couleur.

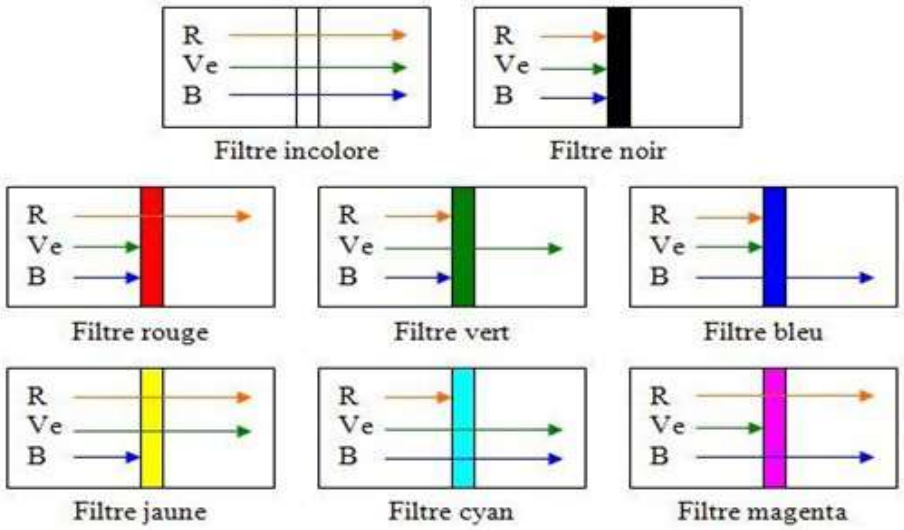
Si de la lumière blanche arrive sur une surface de couleur bleue , seule les longueurs d'onde correspondant au bleu seront renvoyées a notre œil et les autres absorbés.Hors , sur une surface noire , toutes les longueurs d'onde sont absorbées , on ne reçoit plus que du noir , expliquant pourquoi un faisceau infra-rouge ne peut être émis sur un habit de cet couleur .



Couleurs additives



Couleurs soustractives



La solution proposé pour contourner ce problème est de mettre a disposition des chasubles pour les coureurs pour qu'ils puissent être détectés même si ils venaient a

porter du noir.

Conclusion

A ce jour , il me reste encore à résoudre le problème de saturation sur le capteur PA18 faisant qu'il n'atteint pas les 3,3V , à réalisé des test d'alimentation de mon capteur sur la carte électronique et si le temps me le permet , de réaliser une fiche de dépannage pour expliquer que faire en cas de problème sur la carte . Pour ce qui est des possibilités d'évolution du projet , on peut imaginer que le système de fixation des capteurs infrarouge se fasse a l'aide de barrières fixes, facilitant son utilisation pour le professeur de sport comme pour les coureurs.

PARTIE INDIVIDUELLE JULLIAN ALEXANDRE

J'ai choisi dans ce projet d'être l'étudiant IR 3 et m'occuper de la partie Base de donnée et Serveur. Mon but est, à terme, d'établir la gestion du feu de signalisation, la lecture de l'anémomètre qui permettra de connaître la vitesse moyenne du vent sur la course, et l'accès à la BDD. Il me faudra donc, faire une BDD avec SQLite, permettant d'économiser des performances sur le raspberry PI qu'on utilisera, et éviter de créer un serveur avec PHPMyAdmin, une BDD avec MYSQL qui prendrait trop de place sur notre calculateur.

Après une première réunion effectuée rapidement après le début du projet, nous avons décidé de changer la mise en œuvre de ma partie, au lieu de la mise en place de Web Services, je m'occuperais de mettre en place un serveur qui permettra la communication entre la tablette qui servira de télécommande dont l'IR 1 s'occupe, et le RPI dont l'IR 3 s'occupe.

Partie Professionnelle

Base de données

Spécifications base de données:

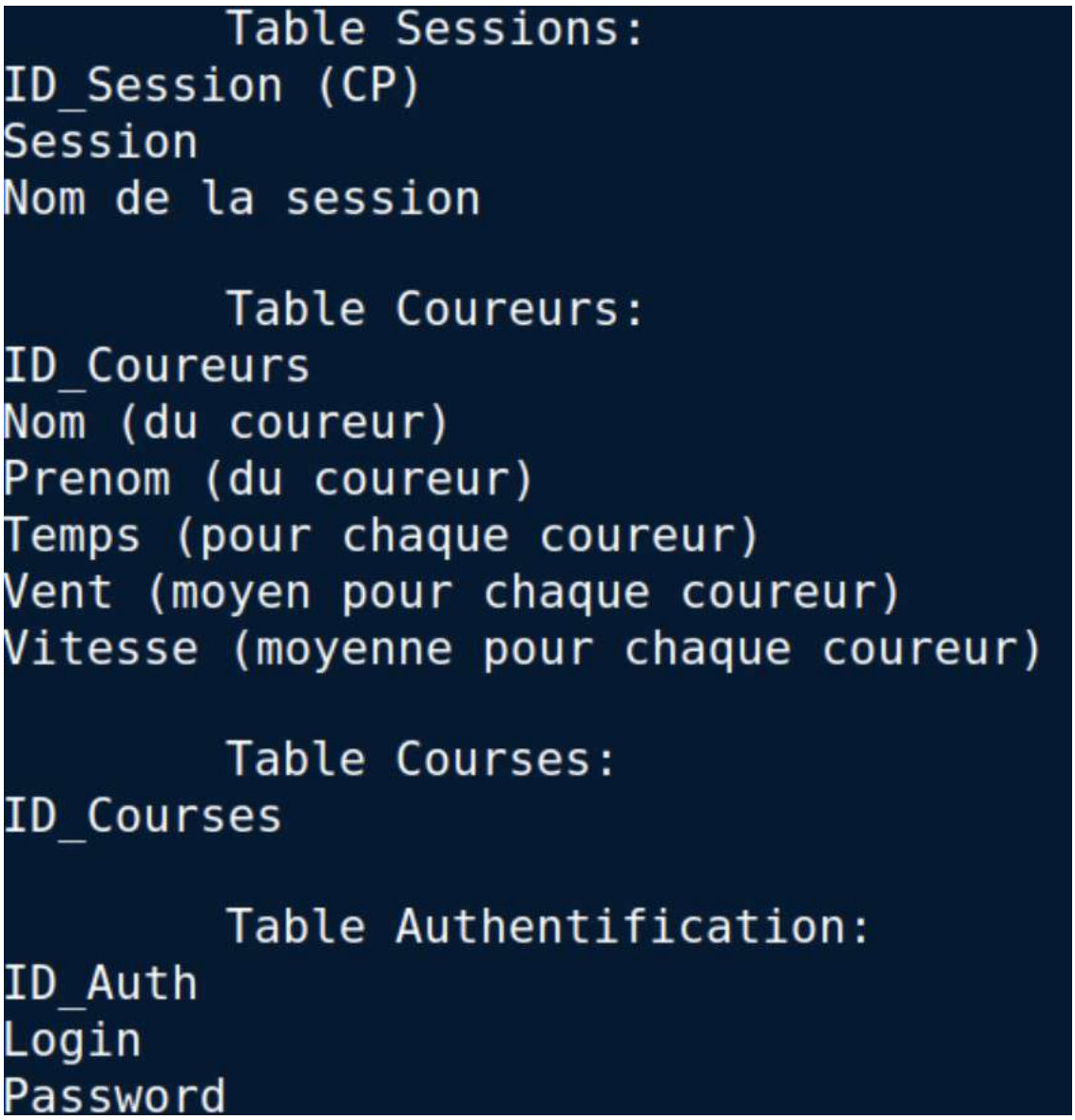
Au départ, je voulais créer la base de données en partant de ces spécifications.

```
ID (chaque table)
Nom (Coureurs)
Prenom (Coureurs)
Session
Date
Identifiant
Mot de passe
Temps (par coureur)
Vent moyen (par coureur)
Vitesse moyenne (par coureur)
CSV
```

Malheureusement, suite à des discussions avec le chef de projet et le client, nous avons décidé que l'exportation du fichier au format CSV permettant l'acquisition et sauvegarde des données se fera directement lorsque l'utilisateur cliquera sur le bouton, l'enregistrement du fichier se fera directement sur la clé USB insérée.

J'ai donc dû repartir sur une bonne base et revoir mes spécifications et ai décidé de retirer certaines tables.

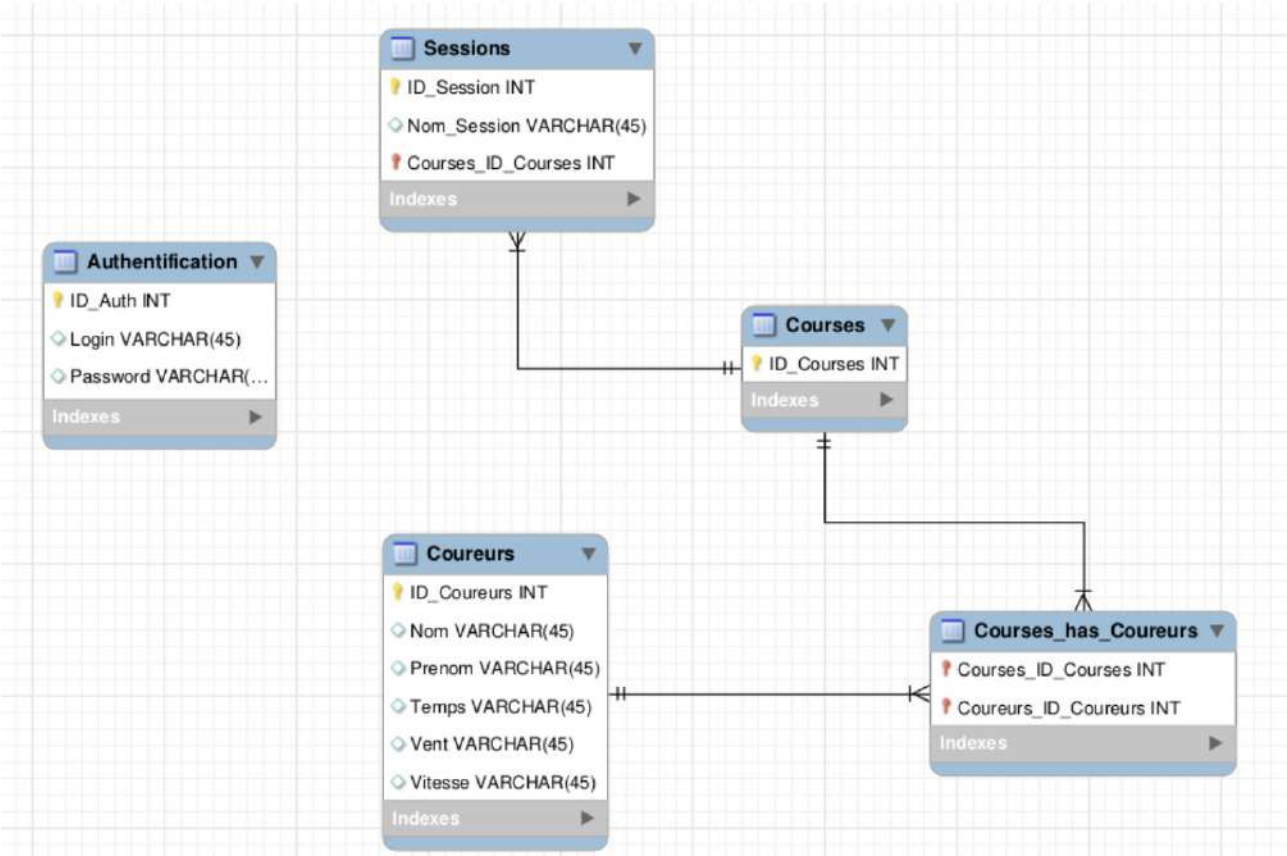
Voici au final ce qui doit être apparent dans les tables de la BDD :



Dans ma BDD, j'ai créé 4 tables. Une table « Sessions », une « Courses », une « Coureurs » et enfin une table « Authentification ». La table Sessions, permettra de gérer le nom de sauvegarder le nom de la session en cours. La table Courses, permettra, elle, d'avoir l'ID de mes courses, clé primaire de celle-ci, et clé étrangère de la table Coureurs, qui contiendra le nom, prénom, Temps, Vent (moyen) et la Vitesse (moyenne) de chaque

coureur. La clé étrangère « ID_Courses » permettra d'avoir un suivi sur la Course « 1 » dans laquelle courent les coureurs 1 et 2 assignés.

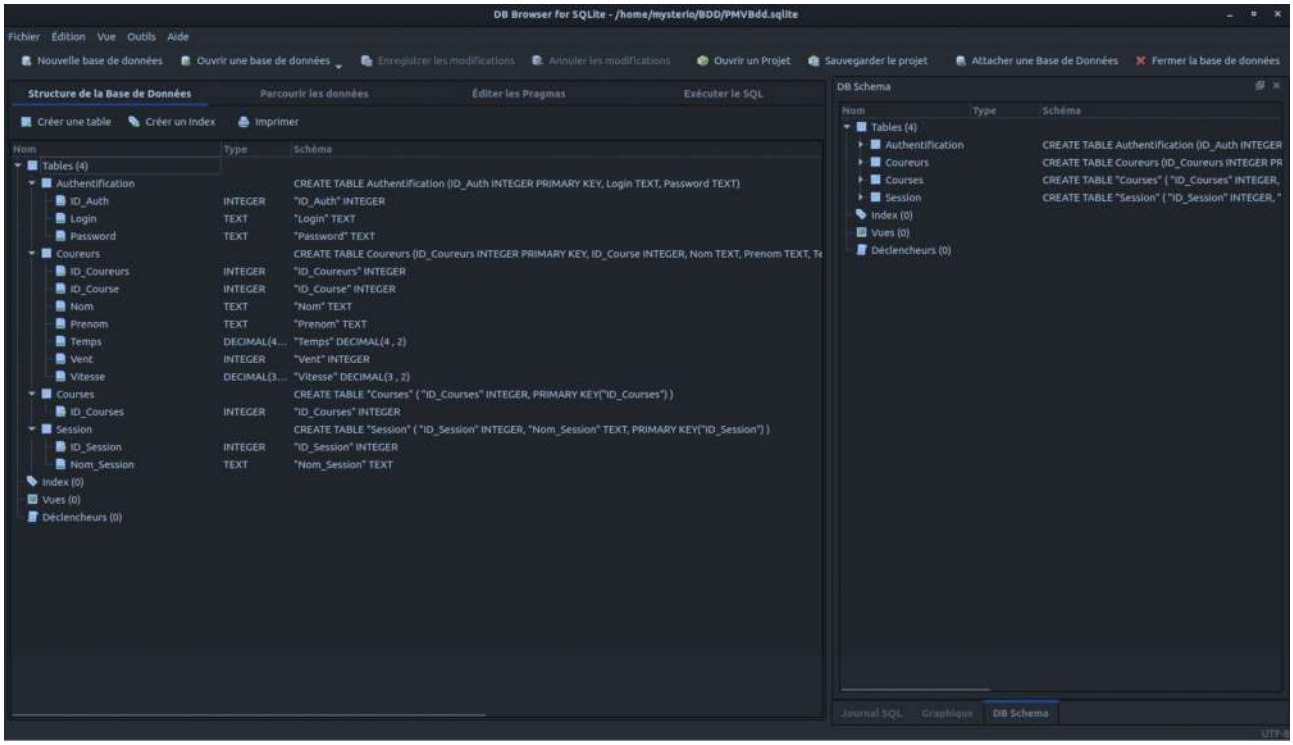
Enfin, la table Authentification permettra de sauvegarder l'identifiant, ainsi que le mot de passe du professeur utilisant l'application.



Je me suis servi de SQLite afin de créer ma base de données, j'ai utilisé des requêtes SQL pour la création de mes tables comme le par exemple pour la table Sessions :

```
CREATE TABLE "Sessions" (
        "ID_Session" PRIMARY KEY INTEGER,
        "Nom_Session" TEXT
        ) ;
```

Et ceci, pour chacune des tables. J'ai fini par trouver une application graphique permettant de visualiser ma base de données plus facilement qu'en ligne de commandes. Ce logiciel s'appelle DBBrowser et facilite grandement la vision ainsi que la création de la base de données en elle-même.



Suite à des discussions, des modifications ont été effectuées afin de ne plus passer par la BDD afin d'enregistrer l'état des boutons.

Afin de pouvoir utiliser la base de données pour modifier, stocker ou vérifier certaines informations, j'ai dû inclure des requêtes SQL dans le code.

```
bool·CBdd::verifConnection(QString·Login,·QString·Pass)·{

·····/*·Récupération·du·nombre·de·lignbe·correspondant·dans·la·BDD·*/

····sqlQuery.prepare("SELECT·*·FROM·Authentification·WHERE·Login·=·:Login·AND·Password·=·:Password;");
····sqlQuery.bindValue(":Login",·Login);
····sqlQuery.bindValue(":Password",·Pass);
····sqlQuery.exec();
```

Ces lignes permettent d'initialiser une requête SQL afin de sélectionner dans la table "Authentification" toutes les informations où il y a une correspondance avec le login et le mot de passe. Elle est ensuite exécutée.

Point d'accès WIFI

Il à fallu configurer le Raspberry PI comme point d'accès wifi afin que la connexion avec la tablette puisse être effectuée. Pour cela, j'ai suivi un tutoriel disponible sur le site https://www.framboise314.fr/raspap-creez-votre-hotspot-wifi-avec-un-raspberry-pi-de-facon-express/, cela m'a permis de le faire facilement et rapidement afin que ça puisse être opérationnel. Il suffisait d'installer quelques paquets et de configurer le nom et la clé de sécurité réseau. Il état ensuite possible de se connecter a la PI et accéder au réseau.

Suite à ça, nous avons pu effectuer quelques essais afin de connaître la distance approximative à laquelle la tablette peut être connectée au point d'accès.

Protocole de communication

Nous avons par la suite, établit un protocole de communication avec Erwan afin de pouvoir faire transiter certaines données dont il pouvait avoir besoin par le biais du réseau et de trames JSON. Nous avons décidés de noms de trames ainsi que de leur composition.

Par exemple, cette trame se nomme, selon notre protocole, "timeRun", et dans le champ de données "data", nous y mettons l'ID de la course, ainsi que le temps, le vent, et la vitesse des deux coureurs.

timeRun

Permet de transmettre le temps d'une course spécifiée (uni-directionnel)

```
{
  "command": "timeRun",
  "data": {
    "id": <id de la course>,
    "time1": "<temps sous forme de chaîne de caractères>",
    "wind1": "<vent sous forme de chaîne de caractères>",
    "speed1": "<vitesse sous forme de chaîne de caractères>",
    "time2": "<temps sous forme de chaine de caractères>",
    "wind2": "<vent sous forme de chaîne de caractères>",
    "speed2": "<vitesse sous forme de chaîne de caractères>"
  }
}
```

Anémomètre

Pour l'anémomètre, suite à des discussions avec le professeur d'électronique, nous avons eu plus de détails sur son fonctionnement.

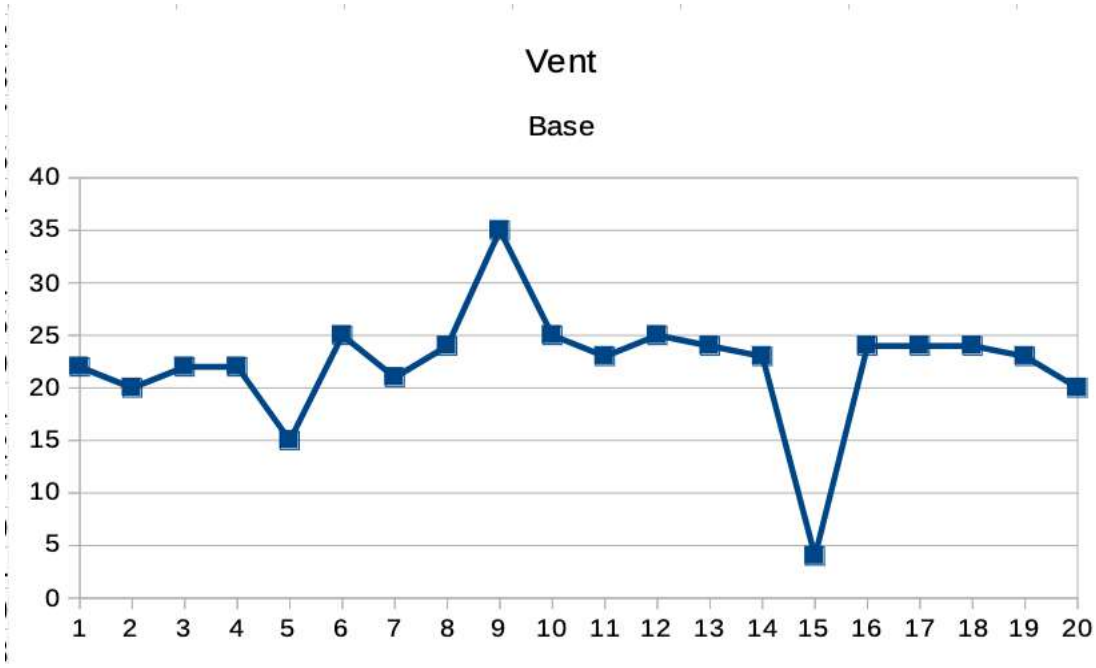
Les valeurs de celui-ci iront de 0 à 50km/h de vent, ce qui représente le retournement de parapluie exposés au vent sur l'échelle de Beaufort.

A cette vitesse, on peut estimer que la course n'aura pas lieu puisque la marche peut commencer à être compliquée. A titre comparatif, en milieu professionnel, il faut que le vent soit inférieur à 7.2km/h afin que le temps soit valable et comptabilisé.

L'anémomètre peut acquérir une valeur proche de 120km/h au maximum. Ici, on limite à 3.3V ce qui, dans notre cas correspond à peu près à 50km/h. La limitation se fait car la carte ne peut pas recevoir plus de 3.3V en entrée. Ce qui, automatiquement, empêche une sortie plus élevée.

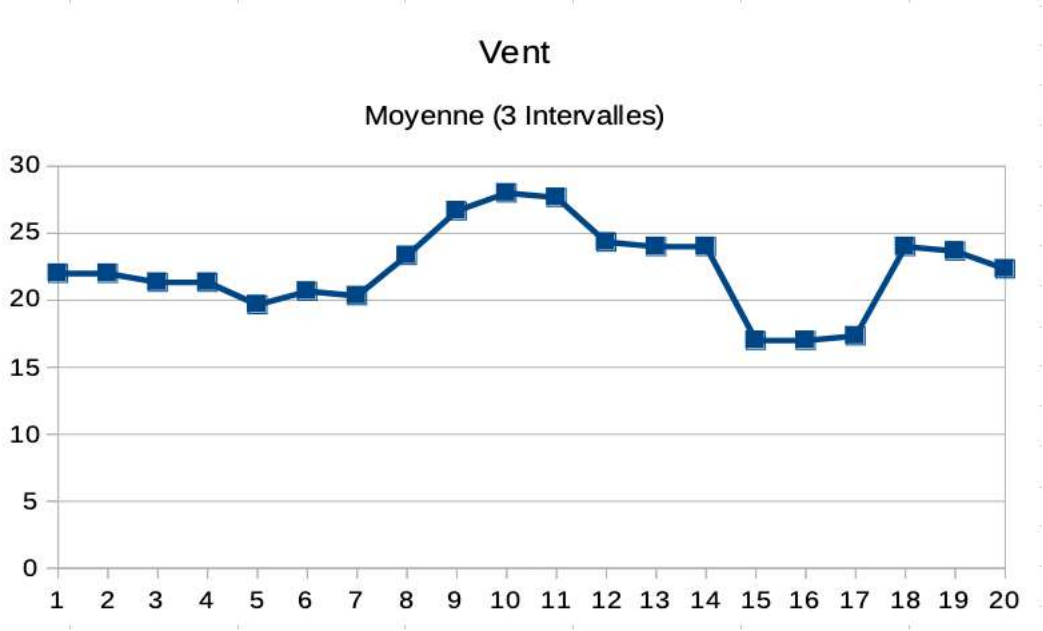
Partie Physique

Le point physique traité dans ma partie sera par rapport à la moyenne du vent relevée par l'anémomètre.



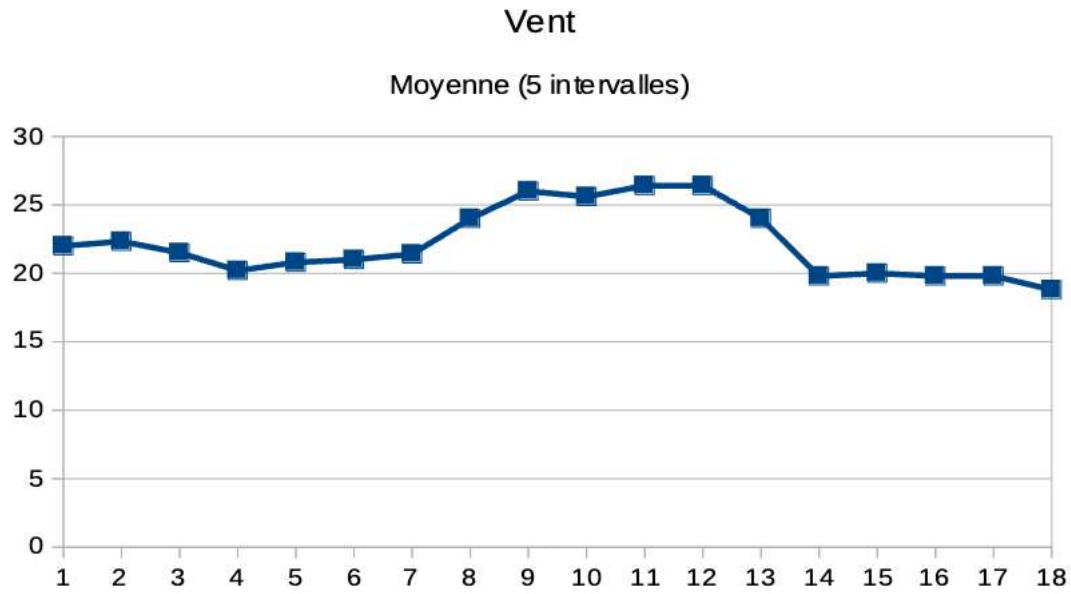
Une simulation à été faite dans un tableur avec des valeurs comprises entre 20 et 25, ainsi que quelques erreurs de mesures permettant de comprendre le but de ce moyenneur.

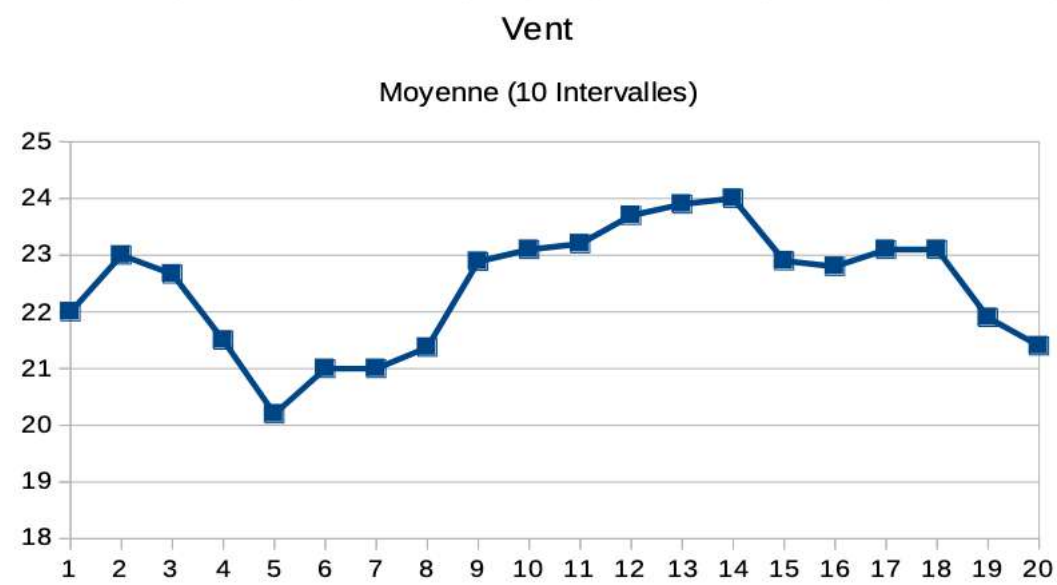
Pour cette première moyenne glissante, j'ai utilisé 3 intervalles différentes avec de voir laquelle pouvait être le plus efficace. Le premier, se base sur 3 valeurs.



On voit que, par rapport aux valeurs de base, le vent a été filtré pour donner des valeurs plus équilibrées ce qui donne donc une moyenne correcte sur les valeurs relevées.

J'ai ensuite décidé de faire une moyenne glissante avec 5 valeurs, puis 10 valeurs afin de comparer celle qui serait le mieux à utiliser et qui garderait une bonne précision.





On voit que, sur la moyenne à 5 intervalles, on se retrouve avec un graphique plutôt constant, et qui corrige bien les erreurs potentielles de mesures. Les valeurs calculées restent raisonnablement proche de celles relevées dans la réalité et nous permet d'avoir un relevé assez précis du vent. De même manière que, pour la moyenne à 10 intervalles, la correction est faite, mais les valeurs se retrouvent encore plus atténuées étant donné que la plage de valeurs est plus importante. Ce qui nous amène donc à penser que, l'utilisation de 5 valeurs afin de calculer la moyenne reste le mieux afin d'avoir quelque chose de correcte et proche de la réalité.

On se rend finalement compte que, le moyenneur agit ici comme un filtre passe bas. Il vient couper les valeur les plus grandes, et les égaliser avec celles qui sont plus petites.

PARTIE INDIVIDUELLE ROBILLARD THÉO

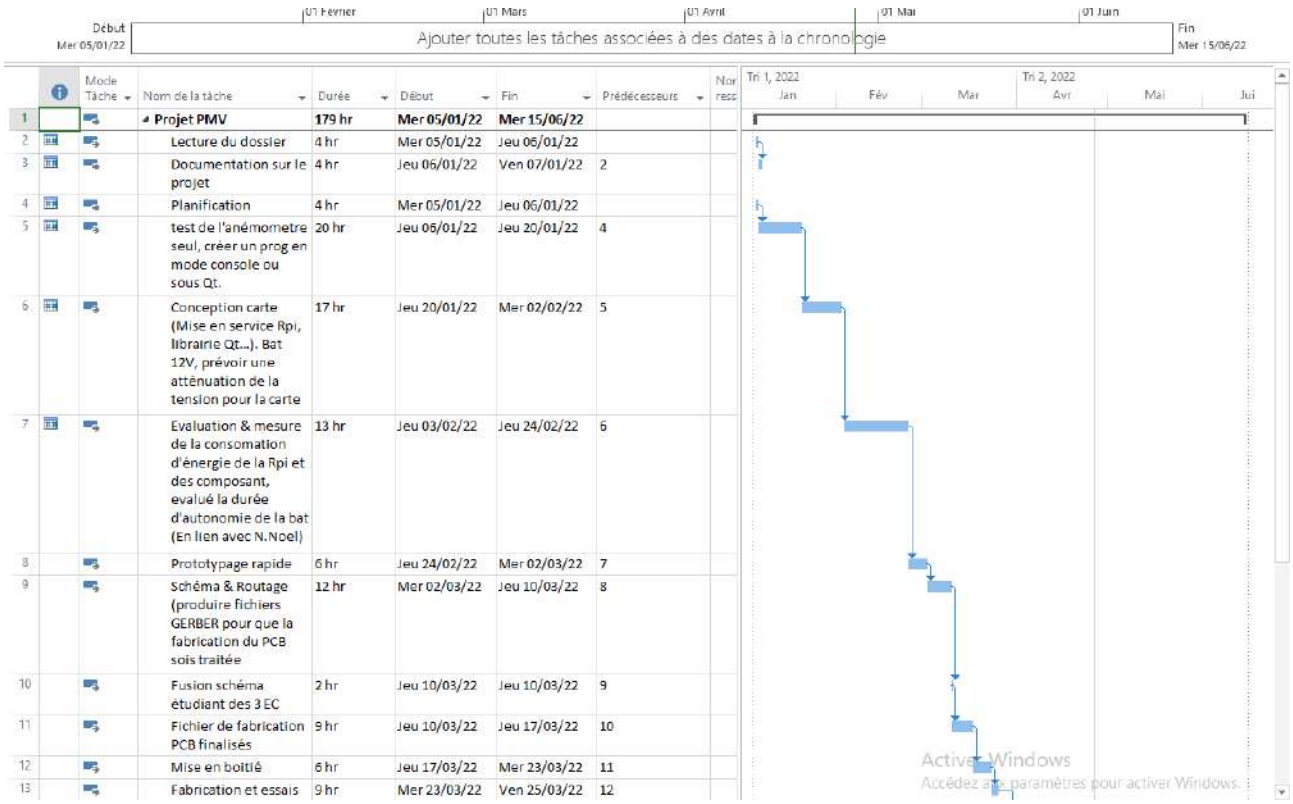
On m’a attribué le rôle de l’EC3 qui consiste à la mise en place de la batterie ainsi que d’un capteur de vent (anémomètre)

Le but étant d’alimenter la Raspberry Pi 3 via la batterie et de relever la puissance du vent avant de démarrer les courses (celui-ci pouvant altéré la vitesse du coureur en fonction de sa puissance).

Partie professionnelle

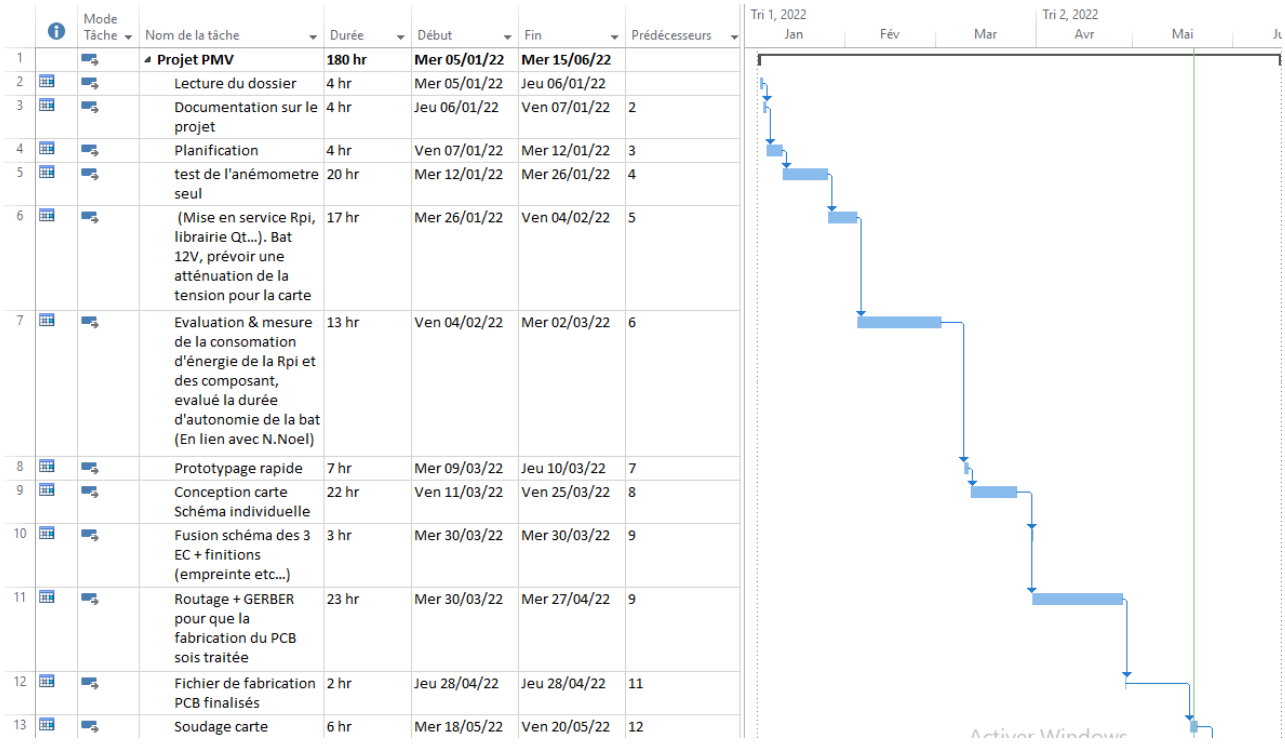
Gestion du projet

Après avoir pris connaissance des documents présentant le projet PMV j’ai créer mon diagramme de Gantt prévisionnel celui-ci me permettant de visualisé le temps qu’il me faudrait pour accomplir les tâches qui m’était demander, le but est de le comparer avec le diagramme de Gantt final.



	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		**Projet PMV**	**179 hr**	**Mer 05/01/22**	**Mer 15/06/22**	
2		Lecture du dossier	4 hr	Mer 05/01/22	Jeu 06/01/22	
3		Documentation sur le projet	4 hr	Jeu 06/01/22	Ven 07/01/22	2
4		Planification	4 hr	Mer 05/01/22	Jeu 06/01/22	
5		test de l'anémometre seul, créer un prog en mode console ou sous Qt.	20 hr	Jeu 06/01/22	Jeu 20/01/22	4
6		Conception carte (Mise en service Rpi, librairie Qt...). Bat 12V, prévoir une atténuation de la tension pour la carte	17 hr	Jeu 20/01/22	Mer 02/02/22	5
7		Evaluation & mesure de la consomation d'énergie de la Rpi et des composant, evalué la durée d'autonomie de la bat (En lien avec N.Noel)	13 hr	Jeu 03/02/22	Jeu 24/02/22	6
8		Prototypage rapide	6 hr	Jeu 24/02/22	Mer 02/03/22	7
9		Schéma & Routage (produire fichiers GERBER pour que la fabrication du PCB sois traitée	12 hr	Mer 02/03/22	Jeu 10/03/22	8
10		Fusion schéma étudiant des 3 EC	2 hr	Jeu 10/03/22	Jeu 10/03/22	9
11		Fichier de fabrication PCB finalisés	9 hr	Jeu 10/03/22	Jeu 17/03/22	10
12		Mise en boitié	6 hr	Jeu 17/03/22	Mer 23/03/22	11
13		Fabrication et essais	9 hr	Mer 23/03/22	Ven 25/03/22	12

Diagramme de Gantt prévisionnel



	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		**Projet PMV**	**180 hr**	**Mer 05/01/22**	**Mer 15/06/22**	
2		Lecture du dossier	4 hr	Mer 05/01/22	Jeu 06/01/22	
3		Documentation sur le projet	4 hr	Jeu 06/01/22	Ven 07/01/22	2
4		Planification	4 hr	Ven 07/01/22	Mer 12/01/22	3
5		test de l'anémometre seul	20 hr	Mer 12/01/22	Mer 26/01/22	4
6		(Mise en service Rpi, librairie Qt...). Bat 12V, prévoir une atténuation de la tension pour la carte	17 hr	Mer 26/01/22	Ven 04/02/22	5
7		Evaluation & mesure de la consomation d'énergie de la Rpi et des composant, evalué la durée d'autonomie de la bat (En lien avec N.Noel)	13 hr	Ven 04/02/22	Mer 02/03/22	6
8		Prototypage rapide	7 hr	Mer 09/03/22	Jeu 10/03/22	7
9		Conception carte Schéma individuelle	22 hr	Ven 11/03/22	Ven 25/03/22	8
10		Fusion schéma des 3 EC + finitions (empreinte etc...)	3 hr	Mer 30/03/22	Mer 30/03/22	9
11		Routage + GERBER pour que la fabrication du PCB sois traitée	23 hr	Mer 30/03/22	Mer 27/04/22	9
12		Fichier de fabrication PCB finalisés	2 hr	Jeu 28/04/22	Jeu 28/04/22	11
13		Soudage carte	6 hr	Mer 18/05/22	Ven 20/05/22	12

Diagramme de Gantt final

En soit, le prévisionnel et le final se rapprochent beaucoup. On a beaucoup changé le schéma structurel, et j'ai été un peu lent sur le routage de la carte.

Il manque certains composants pour dire que le soudage de la carte est fini. J'ai indiqué 6h car on est pas tous passés en même temps pour le soudage. Étant passé en dernier, j'ai travaillé sur la liste de matériel et mon dossier pendant que les autres soudaient. Mon soudage à moi a pris environ 2h.

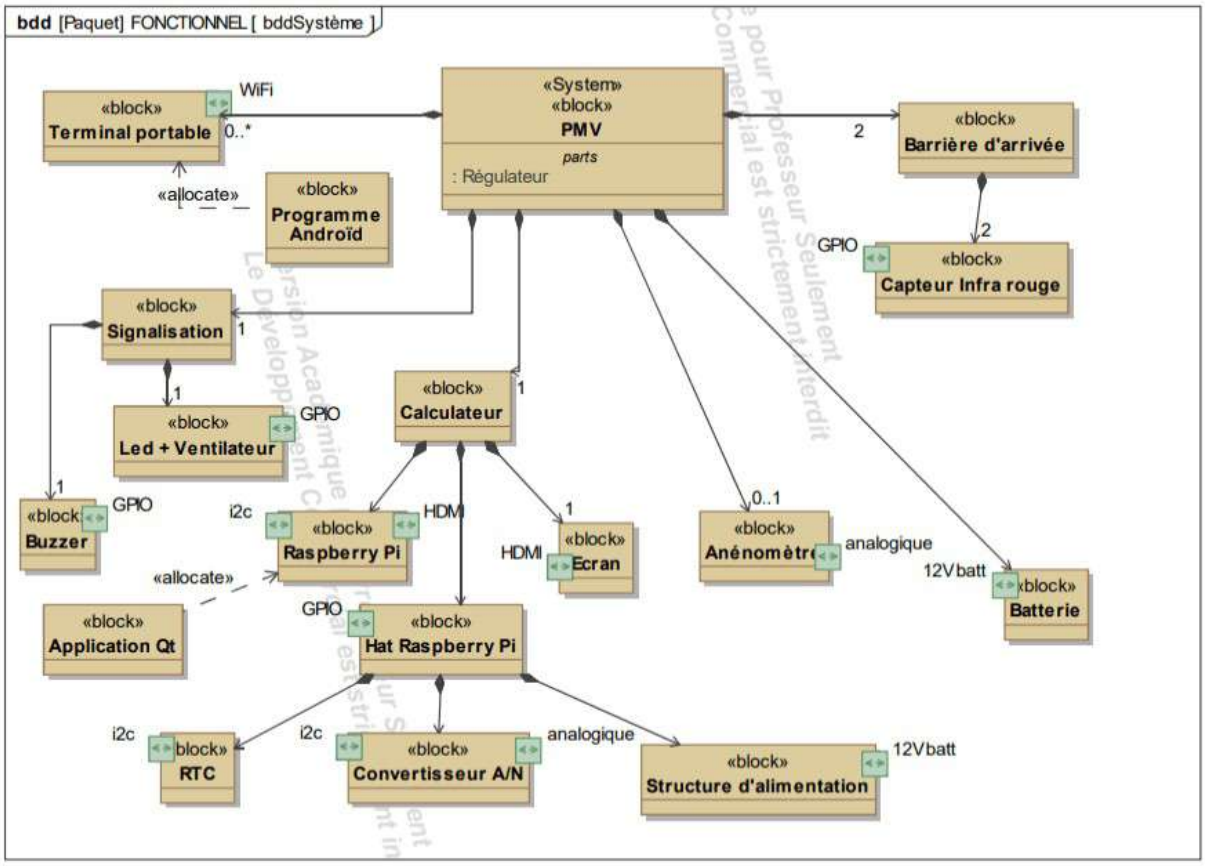


Diagramme de block

Anémomètre

L'anémomètre est un appareil de mesure permettant de mesurer la vitesse du vent. En compétition un temps n'est pas pris en compte si le vent souffre à + de 2 m/s

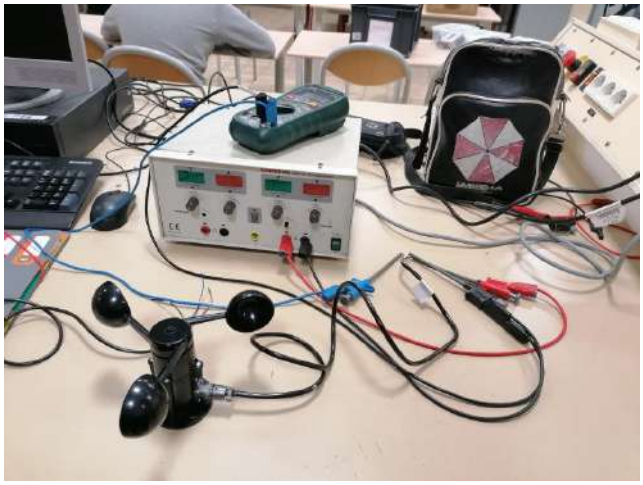
Mon anémomètre sera suivi d'un ADC il fonctionnera en I2C et convertira les données de l'Anémomètre (analogique) en données numériques.

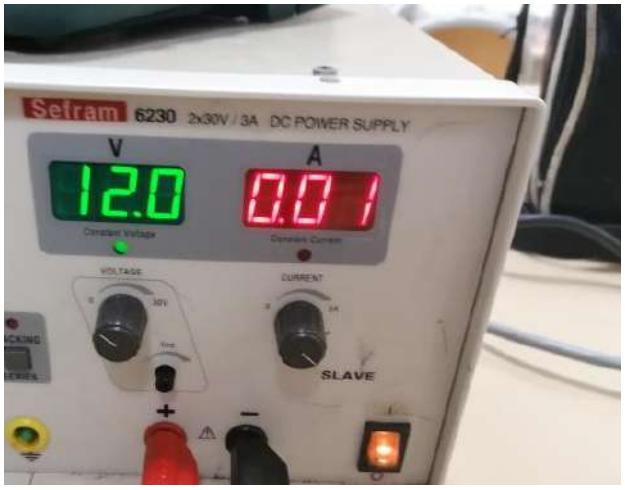
Les valeurs de celui-ci iront de 0 à 50km/h. Ce qui est déjà très puissant, à titre comparatif, dans le milieu compétitif les records ne sont pas comptés si le vent dépasse 7,2 km/h

Dans notre cas on limite à 3.3V ce qui correspond à peu près à 50km/h pour cet anémomètre.

Phase de test: (Sans programme)

Dans les spécifications de l'anémomètre, il y est indiqué qu'il génère une tension entre 0,4V et 2V en fonction de la vitesse à laquelle il tourne.





Phase de test: (Avec programme)

Ce test expérimente l'ADC, l'anémomètre à été par un potentiomètre

```cpp
#include <iostream>
#include <bcm2835.h>
using namespace std;

#define clk_div BCM2835_I2C_CLOCK_DIVIDER_2500

// g++ ADC101_V3.cpp -l bcm2835 -o ADC101_V3

void init_I2c_bcm2835()
{

int main(void)
{
        double mesure;
        unsigned int slave_address=0x54; // Adresse du DS1621
        char i2cOut[3];         // Tableau des données I2C à sortir
        char i2cIn[2];          // Tableau des données I2C entrées
        int resultat_ADC =0;
        double tension=0;

        init_I2c_bcm2835();
        bcm2835_i2c_setSlaveAddress(slave_address);     // Affectation de l'adresse de l'esclave

        i2cOut[0] = 0x00;
        bcm2835_i2c_write(i2cOut, 1);   // Prépositionne la valeur du pointeur 0x00 Pour être en mode "Conversion Result)
        delay(1000);                                    // 1 seconde avant première lecture

        while(1)
                {
                bcm2835_i2c_read(i2cIn,2);
                //cout<<"i2cIn[0] = "<< (int)i2cIn[0]<< "      i2cIn[1] = "<< (int)i2cIn[1]<<endl;
                resultat_ADC = ((i2cIn[1]>>2)& 0b00111111) + (64* (i2cIn[0] & 0b00001111));
                tension = resultat_ADC*3.3/1023; // tension/quantum
                cout<<"Sortie ADC :  "<<resultat_ADC<<"  Tension de sortie : "<<tension<<"V"<<endl;
                //if (i2cIn[1]!=0) mesure = (double)i2cIn[0] + 0.5;     // Commenter
                //else mesure = (double)i2cIn[0];
                //cout<<"Temperature du DS1621 : "<<mesure<<" deg."<<endl;
                delay(1000);                                                            // 1 seconde entre chaque conversion
                }

        bcm2835_close();
        return 0;
}
```

```
pi@raspberrypi:~/Desktop/ADC101_&_Rpi $ sudo ./ADC101_V3
Sortie ADC :  554  Tension de sortie : 1.7871V
Sortie ADC :  554  Tension de sortie : 1.7871V
Sortie ADC :  554  Tension de sortie : 1.7871V
Sortie ADC :  554  Tension de sortie : 1.7871V
Sortie ADC :  554  Tension de sortie : 1.7871V
Sortie ADC :  554  Tension de sortie : 1.7871V
SSortie ADC :  554  Tension de sortie : 1.7871V
Sortie ADC :  554  Tension de sortie : 1.7871V
Sortie ADC :  554  Tension de sortie : 1.7871V
Sortie ADC :  554  Tension de sortie : 1.7871V
```

Batterie

Pour ce projet on m'a confié une batterie YUASA 12V

Data Sheet

NP SERIES - **NP7-12**

Reliability is your Security

Yuasa NP, NPC and NPH Batteries. Utilising the latest advance design Oxygen Recombination Technology, Yuasa have applied their 80 years experience in the lead acid battery field to produce the optimum design of Sealed Lead Acid batteries.

FEATURES

- Superb recovery from deep discharge.
- Electrolyte suspension system.
- Gas Recombination.
- Multipurpose: Float or Cyclic use.
- Usable in any orientation (except continuous inverted).
- Superior energy density.
- Lead calcium grids for extended life.
- Manufactured World wide.
- Application specific designs.

Technical Features

Sealed Construction
Yuasa's unique construction and sealing technique ensures no electrolyte leakage from case or terminals

Electrolyte Suspension System
All NP batteries utilize Yuasa's unique electrolyte suspension system incorporating a microfine glass mat to retain the maximum amount of electrolyte in the cells. The electrolyte is retained in the separator material and there is no free electrolyte to escape from the cells. No gels or other contaminants are added.

Control of Gas Generation
The design of Yuasa's NP batteries incorporates the very latest oxygen recombination technology to effectively control the generation of gas during normal use.

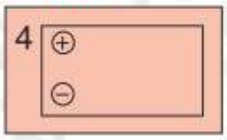
Low Maintenance Operation
Due to the perfectly sealed construction and the recombination of gasses within the cell, the battery is almost maintenance free.

Terminals
NP batteries are manufactured using a range of terminals which vary in size and type. Please refer to details as shown.

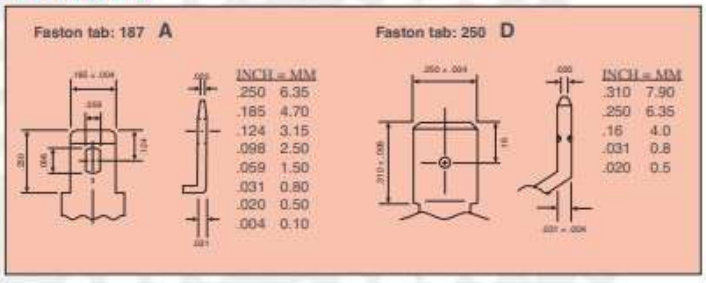
Operation in any Orientation
The combination of sealed construction and Yuasa's unique electrolyte suspension system allows operation in any orientation, with no loss of performance or fear of electrolyte leakage. (Excluding continuous use inverted)

Valve Regulated Design
The batteries are equipped with a simple, safe low pressure venting system which releases excess gas and automatically reseals should there be a build up of gas within the battery due to severe overcharge. Note. On no account should the battery be charged in a sealed container.

Layout



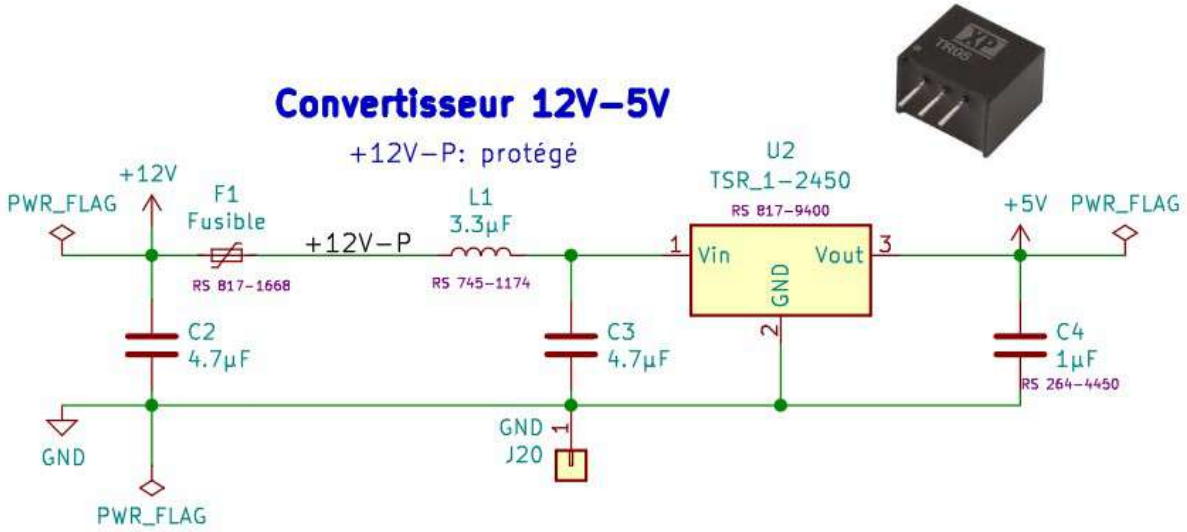
Terminals



General Specifications

Nominal Capacity (Ah)	NP7-12
20hr to 1.75vpc 30°C	7
10hr to 1.75vpc 20°C	6.4
5hr to 1.70vpc 20°C	5.9
1hr to 1.60vpc 20°C	4.2
Voltage	12
Energy Density (Wh.t.20hr)	91
Specific Energy (Wh.kg.20hr)	32
Int. Resistance (m.Ohms)	25
Maximum discharge (A)	40/75
Short Circuit current (A)	210
Dimensions (mm)	
Length	151
Width	65
Height overall	97.5
Weight (Kg)	2.65
Terminal	A/D
Layout	4
Terminal Torque Nm	-

Le but de la batterie est d'alimenter la RPi pour cela on a besoin de convertir le 12V en 5V je me servirai pour les test de ce schéma utiliser sur rushball.



On s'en servira pour faire des test dehors, ces test permettant de voir l'alimentation de la RPi mais aussi de voir la puissance de la led de Nathan

Schéma & routage KICAD

Notre carte sera un HAT pour une Raspberry qu'on connectera via les broches GPIO le but du HAT(chapeau en anglais) étant d'être brancher au dessus de la Rpi.

Nous sommes 3 dans le projet, et nous avons tous une partie différente, donc nous avons fait chacun une partie du schéma structurel qui correspond à notre contrat, ces parties seront ensuite, après approbation du professeur d'électronique fusionner.

On à fais une mini réunion avec les IR pour ce mettre d'accord sur l'utilisation des GPIO

voici le plan final réalisé par William.

Liste des GPIO du HAT Raspberry PI (Projet PMV)

Pour William RICHARD :

-GPIO17 : Capteur Couloir 1

-GPIO27 : Capteur Couloir 2

Pour Théo ROBILLARD :

-GPIO2 : SDA(Données)

-GPIO3 : SCL(Horloge)

Pour Nathan NOEL :

-GPIO18: Led ROUGE --GPIO22: Buzzer

-GPIO23: Led VERTE

Schema HAT RPI3 (2x10):

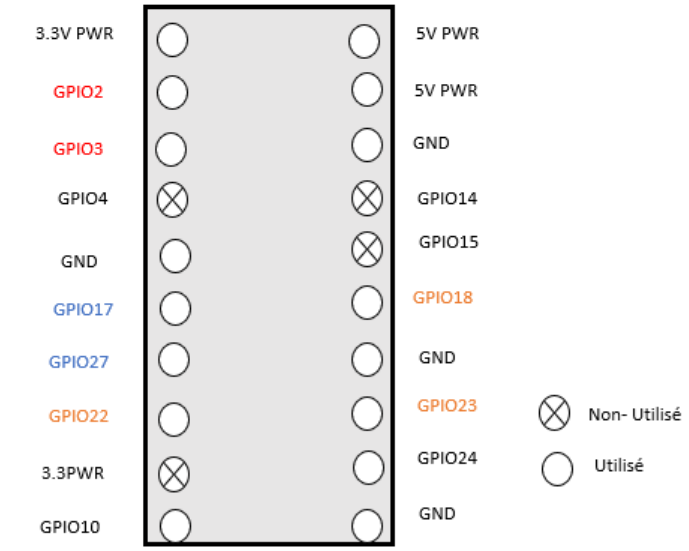
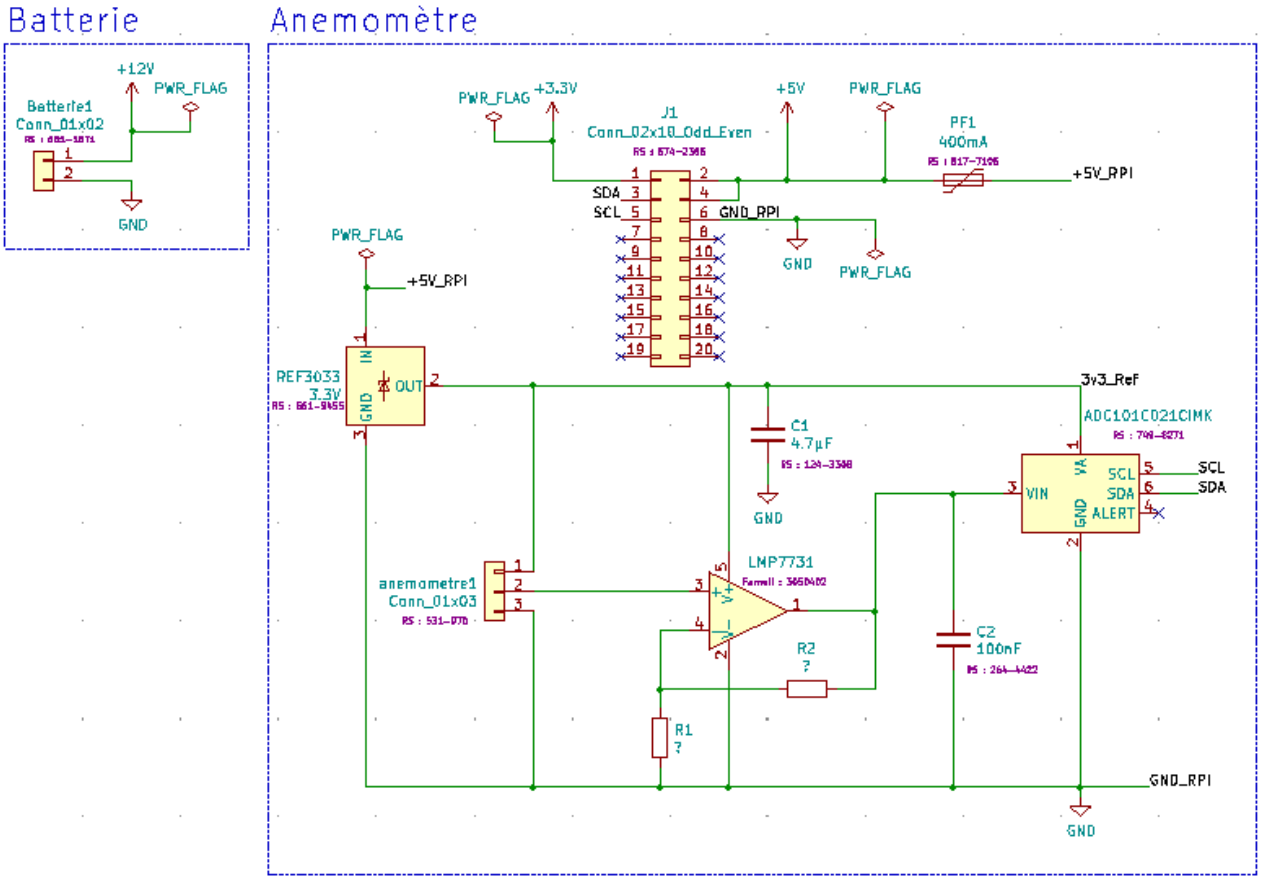
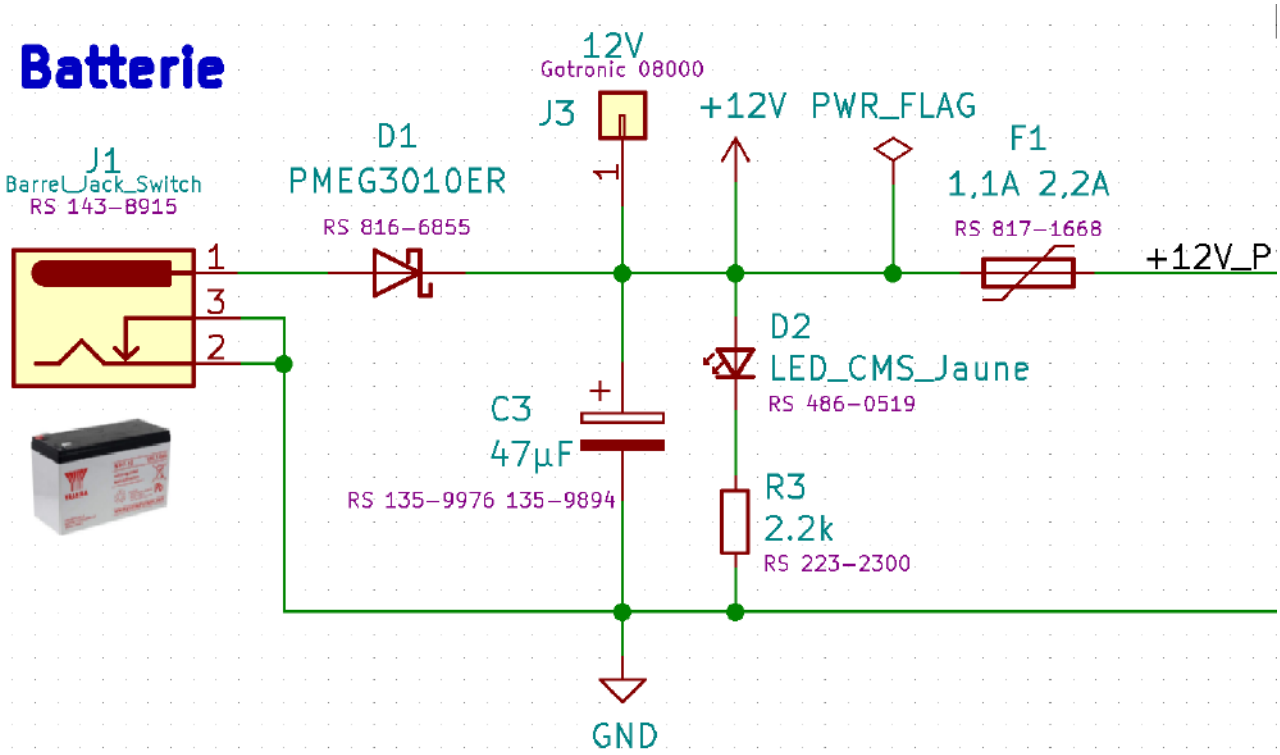


Schéma structurel

La première version de ma partie ressemble à ceci

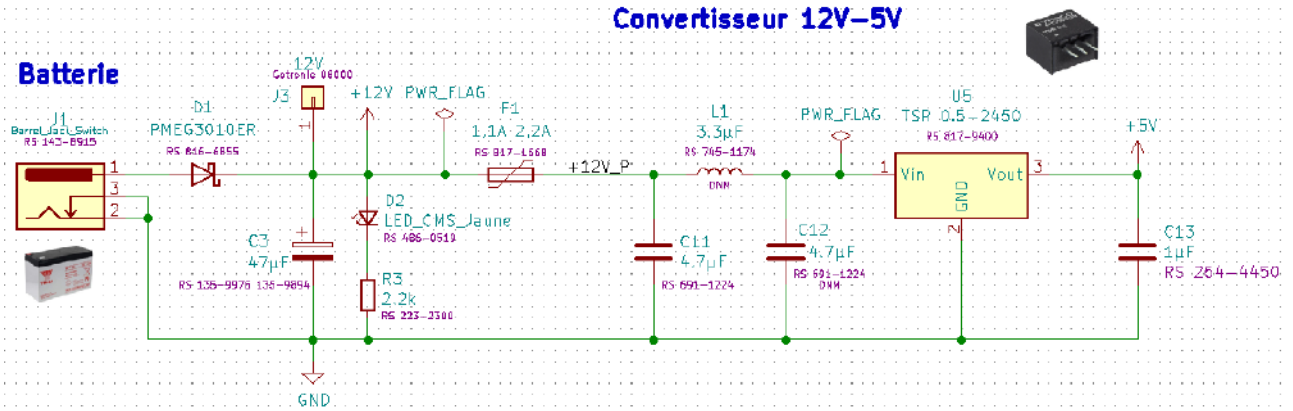


Ensuite on a rajouter une protection pour la batterie

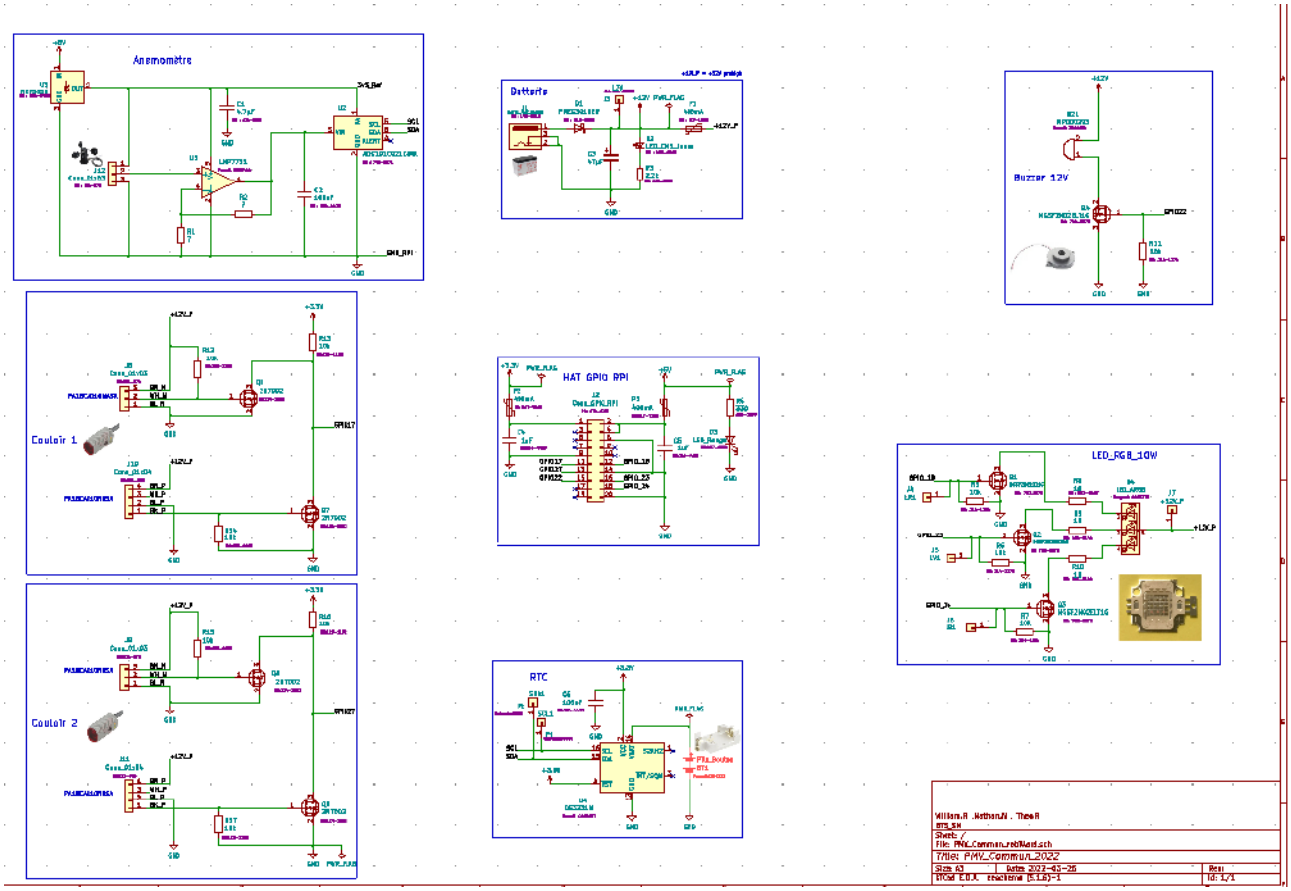


La diode D1 est une diode de Schottky. Elle a pour but de protéger les circuits des régulateurs.

Et pour finir on a rajouter le convertisseur 12V → 5V provenant du schéma rushball :



Après avoir eu l'approbation du professeur et quelques changement sur certaines parties, le schéma final avec les parties de Nathan (Buzzer, Led) et de William (RTC + détecteurs)



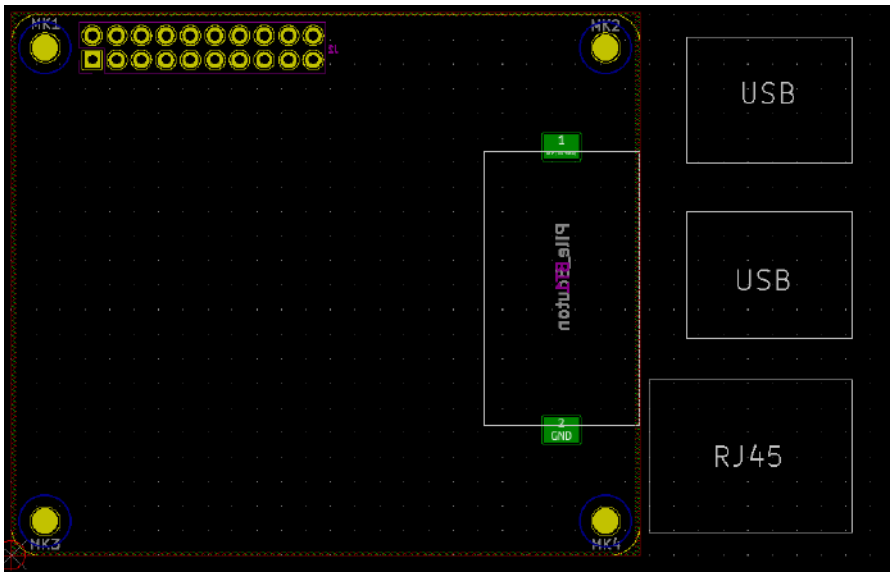
Pour finir, on a affecter les empreintes, celles-ci sont importantes pour le routage puis-ce qu'elles sont les modèles des composants.

Symbole: Attribution Empreintes

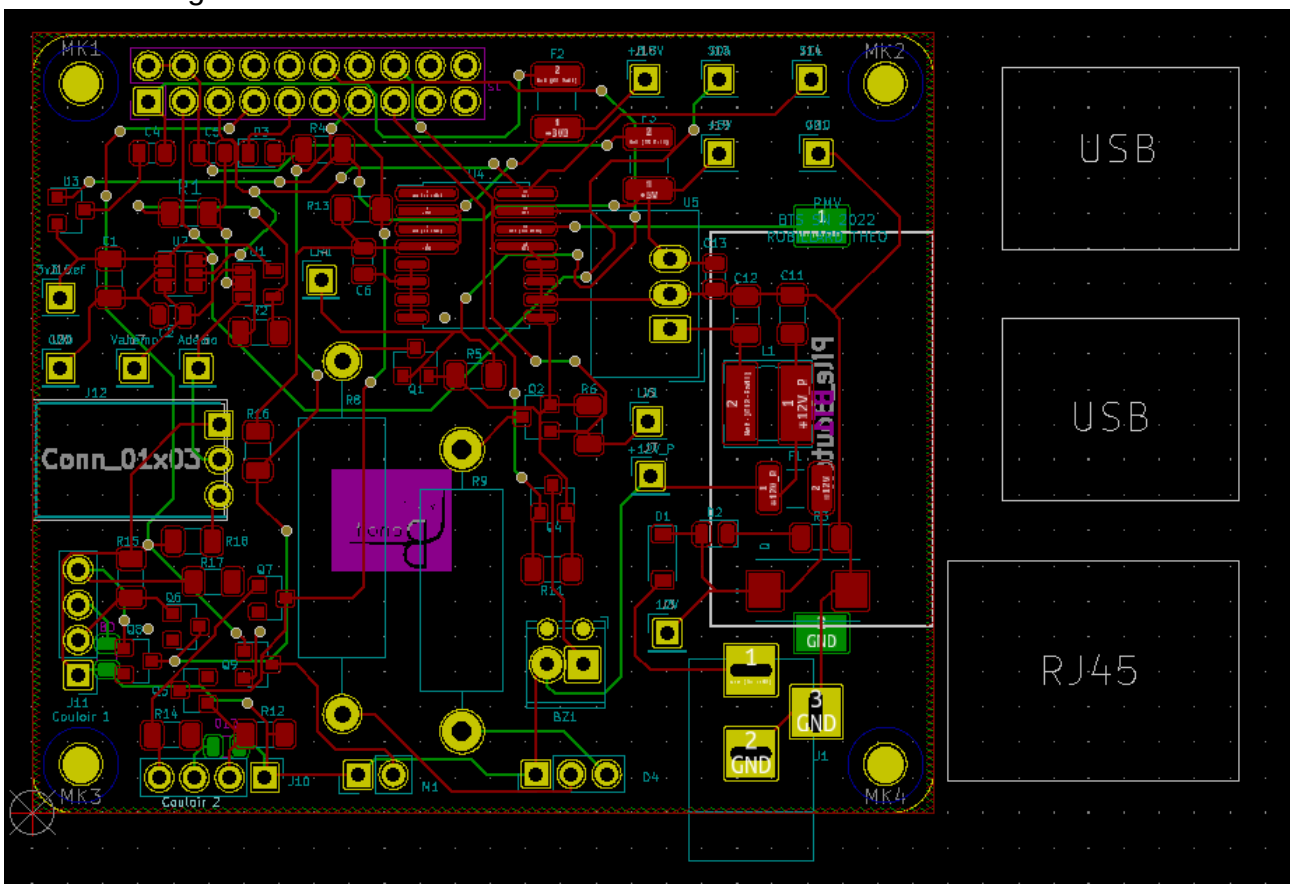
1	BT1	Pile_Bouton	0_ModulesProjet:Support_pile_bouton_Keystone_1060
2	BZ1	MP000293	0_ModulesProjet:Bornier_Phoenix_Contact_x2cts_172565
3	C1	4.7µF	Capacitor_SMD:C_1206_3216Metric
4	C2	100nF	Capacitor_SMD:C_0805_2012Metric
5	C3	47µF	0_ModulesProjet:Capacitor_AVX_E_CASE_2917
6	C4	1µF	Capacitor_SMD:C_0805_2012Metric
7	C5	1µF	Capacitor_SMD:C_0805_2012Metric
8	C6	100nF	Capacitor_SMD:C_0805_2012Metric
9	C8	1µF	Capacitor_SMD:C_0805_2012Metric
10	C10	1µF	Capacitor_SMD:C_0805_2012Metric
11	C11	4.7µF	Capacitor_SMD:C_1206_3216Metric
12	C12	4.7µF	Capacitor_SMD:C_1206_3216Metric
13	C13	1µF	Capacitor_SMD:C_0805_2012Metric
14	D1	PMEG3010ER	Diode_SMD:D_SOD-123
15	D2	LED_CMS_Jaune	LED_SMD:LED_0805_2012Metric
16	D3	LED_Rouge	LED_SMD:LED_0805_2012Metric
17	D4	LED_ARGB	0_ModulesProjet:Embase_AMPMODU_HE14_Droite_3
18	F1	1,1A 2,2A	0_ModulesProjet:Polyswitch_MICROSMD110F-2
19	F2	400mA	0_ModulesProjet:Polyswitch_MICROSMD110F-2
20	F3	400mA	0_ModulesProjet:Polyswitch_MICROSMD110F-2
21	J1	Barrel_Jack_Switch	0_ModulesProjet:Fiche_alimentation_DC
22	J2	Conn_GPIO_RPI	Connector_PinSocket_2.54mm:PinSocket_2x10_P2.54mm_Ve
23	J3	12V	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
24	J4	LR1	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
25	J5	LV1	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
26	J7	+12V_P	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
27	J10	Conn_01x04	0_ModulesProjet:Embase_AMPMODU_HE14_Droite_4
28	J11	Conn_01x04	0_ModulesProjet:Embase_AMPMODU_HE14_Droite_4
29	J12	Conn_01x03	0_ModulesProjet:Embase_Molex_SL_3
30	J13	SDA	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
31	J14	SCL	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
32	J15	Anemo	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
33	J16	3v3_Ref	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
34	J17	Vanémo	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
35	J18	+3,3V	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
36	J19	+5V	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
37	J20	GND	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
38	J21	GND	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Ve
39	L1	3.3µF	0_ModulesProjet:Choke_TCK-062
40	M1	Fan	0_ModulesProjet:Embase_AMPMODU_HE14_Droite_2
41	Q1	MGSF2N02ELT1G	Package_TO_SOT_SMD:SOT-23

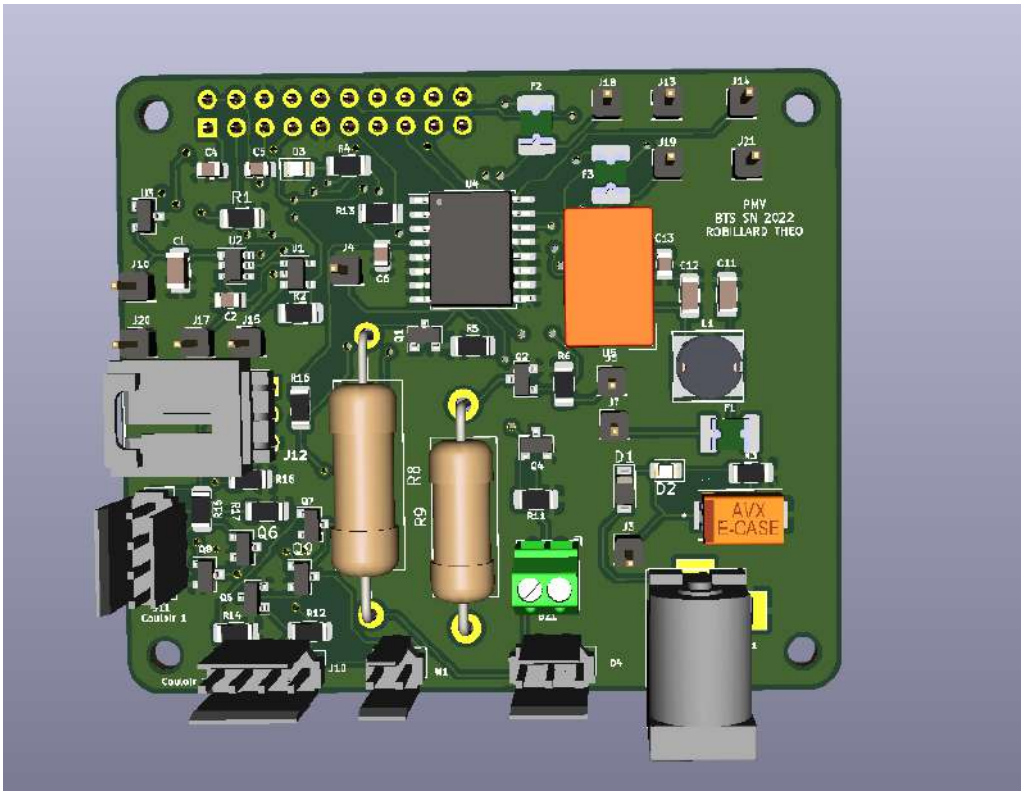
Routage

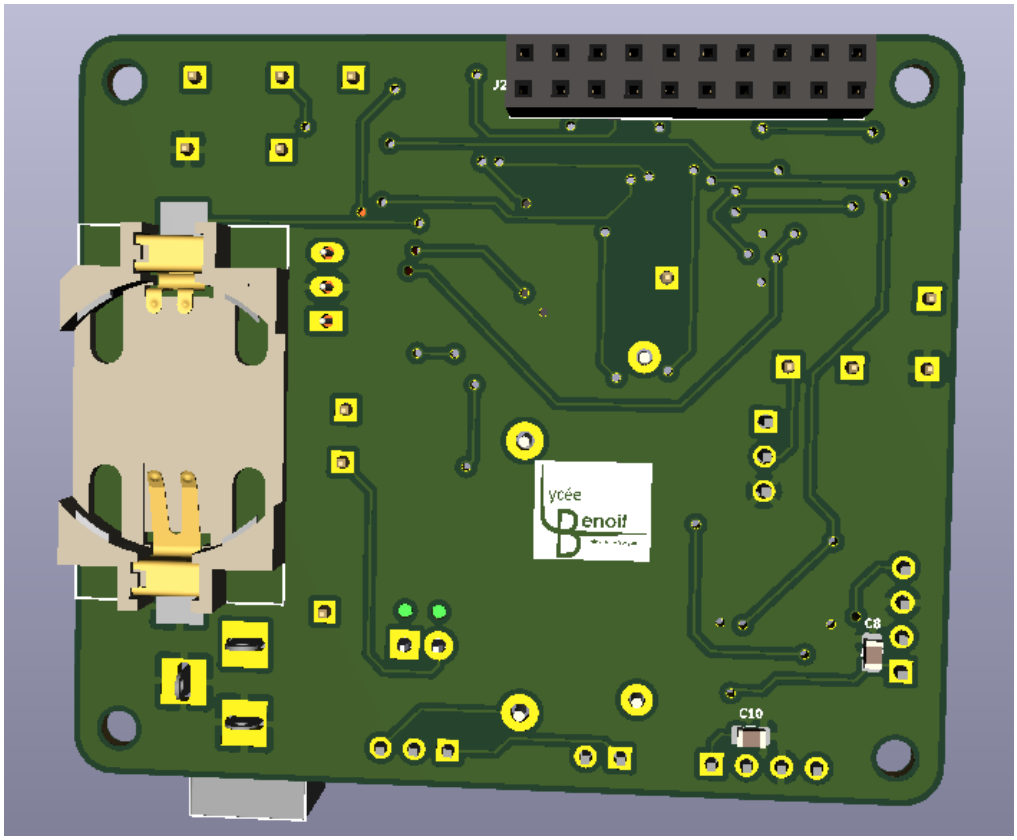
Pour le routage on a dû faire des mesures sur la Rpi pour prévoir le placement du support pile qui sera soudée à l'arrière de la carte, il fallait donc prévoir son placement pour qu'il ne touche pas les composants de la Rpi.



Voici le routage terminé :




PMV
BTS SN 2022
ROBILLARD THEO
AVX
E-CASE


ycée
Benoit

Liste de matériel

On nous a demandé de faire une liste de matériel pour faire une estimation du coût de notre carte électronique.

Repère	Désignation du matér	Valeur (Référence)	Tol.±	Equivalent	Boîtier	Remarques	Qté	Condition	ref rapide	Prix U HT	Prix T HT
BT1	Support Pile			RS 237-4222			1		Farnell 1704232	2,850	2,850
BZ1	Buzzer			Gotronic 05480			1		Farnell 3106924	3,920	3,920
C1, C11, C12	Condensateur	4,7µF	10%	Farnell 1908177	1206 (3216M)		1	Par 10	RS 691-1224	1,102	1,102
C2, C6	Condensateur	1µF	20%	Farnell 1759037	0805 (2012M)		3	Par 25	RS 264-4422	0,140	0,420
C3	Condensateur	47µF	10-20%	RS 135-9976	2917		1	Par 5	RS 135-9894	2,292	2,292
C4, C5, C13,C8, C10	Condensateur	1µF	(-20)-80%	Farnell 7865556	0805(2012)		5	Par 25	RS 264-4450	0,042	0,210
D1	Diode Schottky CMS	30V 1A		Farnell 3869974RL	SOD-123		1	Par 50	RS 816-6855	0,317	0,317
D2	LED Jaune CMS	2, 1V		Farnell 1561241	2012(0805)		1	Par 25	RS 486-0519	0,340	0,340
D3	LED Rouge CMS	2V		Farnell 2099236	2012(0805)		1	Par 5	RS 497-4804	0,278	0,278
D4	LED ARGB			Aliexpress : 4000168221143			1		Farnell 1160379	6,144	6,144
F1	Fusible réarmable	2,2A, 1,1A,24V		Farnell 1345925			1	Par 10	RS 817-1668	0,390	0,390
F2, F3	Fusible réarmable	400mA,0,2A,30V		Farnell 1596997			2	Par 10	RS 517-7105	0,279	0,558
J3,J4,J5,J7,J13,J14,J15,J16,J17,J18,19,J20,J21	Point de test HE14			RS:547-3166			13		GOTRONIC 08000	0,500	6,500
J10,J11	Connecteur_01x04			Farnell 1205984			2	Par 10	RS 531-986	0,899	1,798
J12	Connecteur_01x03			Farnell 1205983			1	Par 10	RS 531-970	0,739	0,739
L1	Inductance			Farnell 1864096			1		RS 745-1174	3,370	3,370
M1	Fan w/ Dissipateur	12v		Ebay: 333565747772			1		AliExpress 4001331919760	3,040	3,040
Q1,Q2,Q4,Q9	MOSFET N CMS	2,8A 20V		Farnell 1463618	SOT-23		4	Par 20	RS 792-5675	0,377	1,508
Q5,Q6,Q7,Q8	MOSTFET N CMS	115mA 60V		RS 124-1692	SOT-23		4	Par 10	Farnell 1865556	0,020	0,081
R5, R6, R11, R12, R13, R14, R15, R16, R17, R18	Resistance CMS	10k Ohms	5%	Farnell 2331740		Changer ref	10	Par 10	668-8524	0,386	3,860
R1	Resistance CMS	10k Ohms	1%	Farnell 1663050	1206(3216M)		1	Par 100	RS 125-1192	0,030	0,030
									TOTAL HT FEUILLE		39,747
									TOTAL HT GLOBAL		39,747
									COEFFICIENT TVA	1,2	47,697

	Nom	Date	Folio
Crée :	Robillard T.	05/05/2022	1/2
Mis à jour :	Robillard T.	05/05/2022	

Carte PMV

Repère	Désignation du matér	Valeur (Référence)	Tol.±	Equivalent	Boîtier	Remarques	Qté	Condition	ref rapide	Prix U HT	Prix T HT
R2	Resistance CMS	21K Ohms	1%	Farnell 2139529	1206(3216M)		1	Par 10	Farnell 2139529	0,068	0,068
R3	Resistance CMS	2.2K Ohms	1%	Farnell 9336168	1206(3216M)		1	Par 50	RS 223-2300	0,078	0,078
R4	Resistance CMS	330 Ohms	1%	Farnell :9240918	1206(3216M)		1	Par 50	RS 679-2049	0,106	0,106
R8	Resistance	18 Ohms	5%	Farnell 9339221	PR02		1	Par 10	RS 683-5657	0,366	0,366
R9	Resistance	10 Ohms	5%	Farnell 1219213	CFR16		1	Par 10	RS 131-716	0,045	0,045
U1	LMP7731	22MHz		Mouser 926-LMP7731MF/NOPB	SOT-23		1		Farnell 3050402	2,370	2,370
U2	ADC101C021	188,9ksps,10bits		Mouser 926-AD101C021CIMKNPB	SOT-23-5		1		RS 749-8271	0,920	0,920
U3	REF3133	3,3V,10mA		Mouser 595-REF3133AIDBZT	SOT-23		1		RS 661-9455	4,500	4,500
U4	DS3231M	2,2V à 5,5V		RS 518-299	SOIC		1		Farnell 2515487	8,040	8,040
U5	TSR 0,5-2450	6,5V à 32Vcc ,500mA		Mouser 495-TSR0.5-2450	SIP 3		1		RS 817-9400	5,370	5,370
J1	Conn alim DC			Farnell 1854512			1		Farnell 1854512	2,420	2,420
J2	Conn femelle 2x20			Farnell 2311676					RS 674-2365		

Nous avons pu utiliser 3 façons différentes de souder notre carte :

CMS (Four à refusion) : 

Les composants CMS se soudent sur la carte directement. Ils n'ont pas de broches et sont très petits, donc l'utilisation du fer à souder est peu recommandée

Le soudage des composants CMS via le four à refusion ce fait en 3 parties :

- La mise en place de la pâte à brasée grâce au stencil

  Pour cette partie, on utilise le stencil pour placer avec précision la à brasée

- <u>Le placement des composants CMS</u>

  Ensuite, à l'aide d'une pince, on place les composants CMS délicatement sur la carte.

- <u>Le four à refusion</u>

  Pour finir, on met la carte dans le four à refusion de 95°C à 275°C. Cependant, celui-ci étant dysfonctionnel, il faut le stopper manuellement après 2m50s .

<u>Résultat final :</u>

CMS (Air chaud)

Le soudage à l'air chaud est aussi fait pour les composants CMS, mais celui-ci nous évite l'utilisation du four à refusion.

Je m'en sert pour souder les 2 condensateur CMS à l'arrière de ma carte.

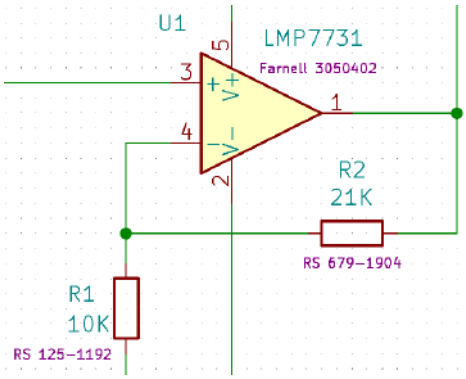
Traversant (Fer à souder) 

Le fer à souder est plus utilisé pour les composants traversants, ils sont plus grands que les CMS et sont soudés via des broches qui passent au travers de la carte.

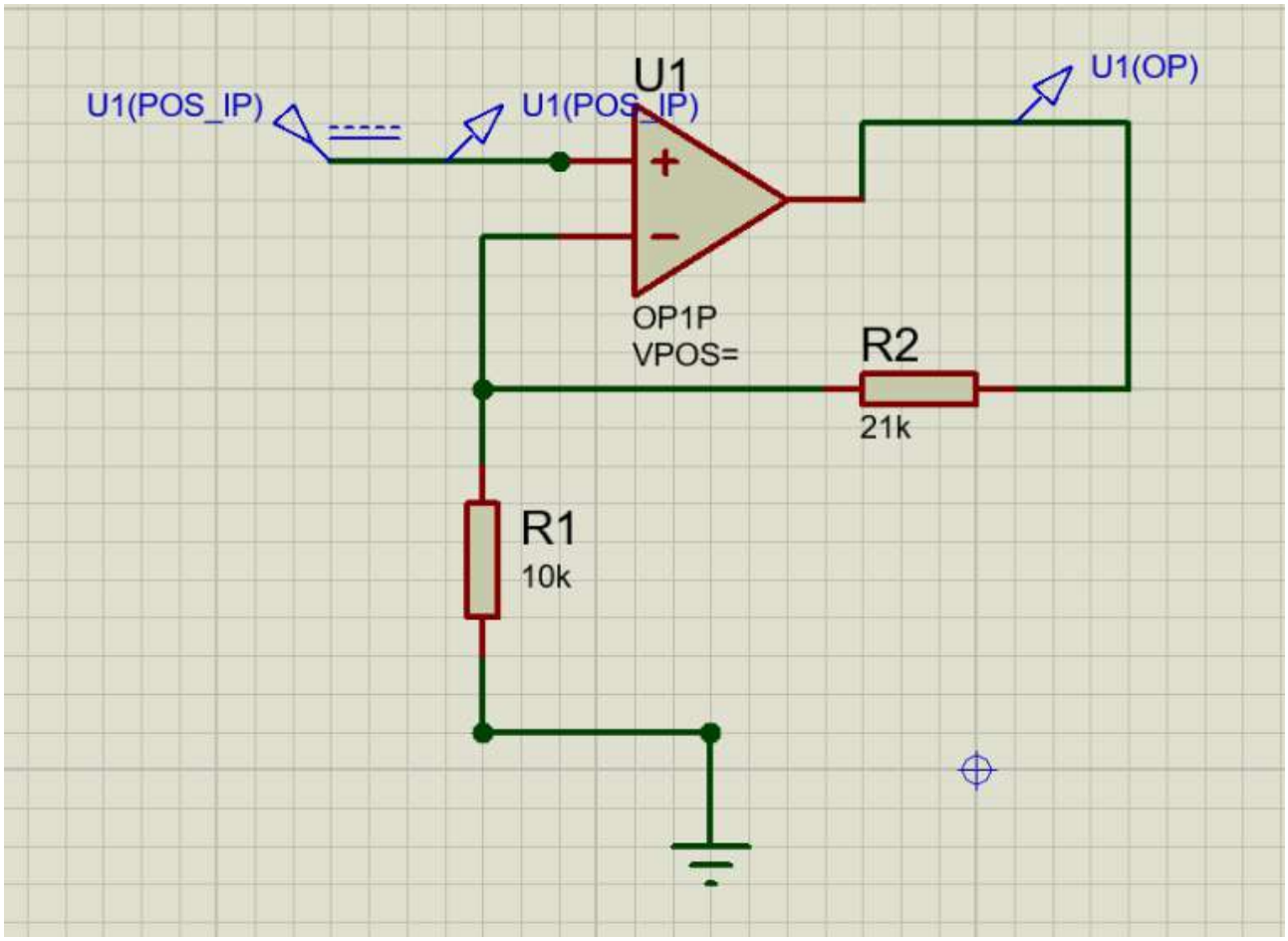
Les DEL sont un bon exemple de composants traversants.

Partie Physique

Dans le montage de mon anémomètre est utilisé un Amplificateur non inverseur.



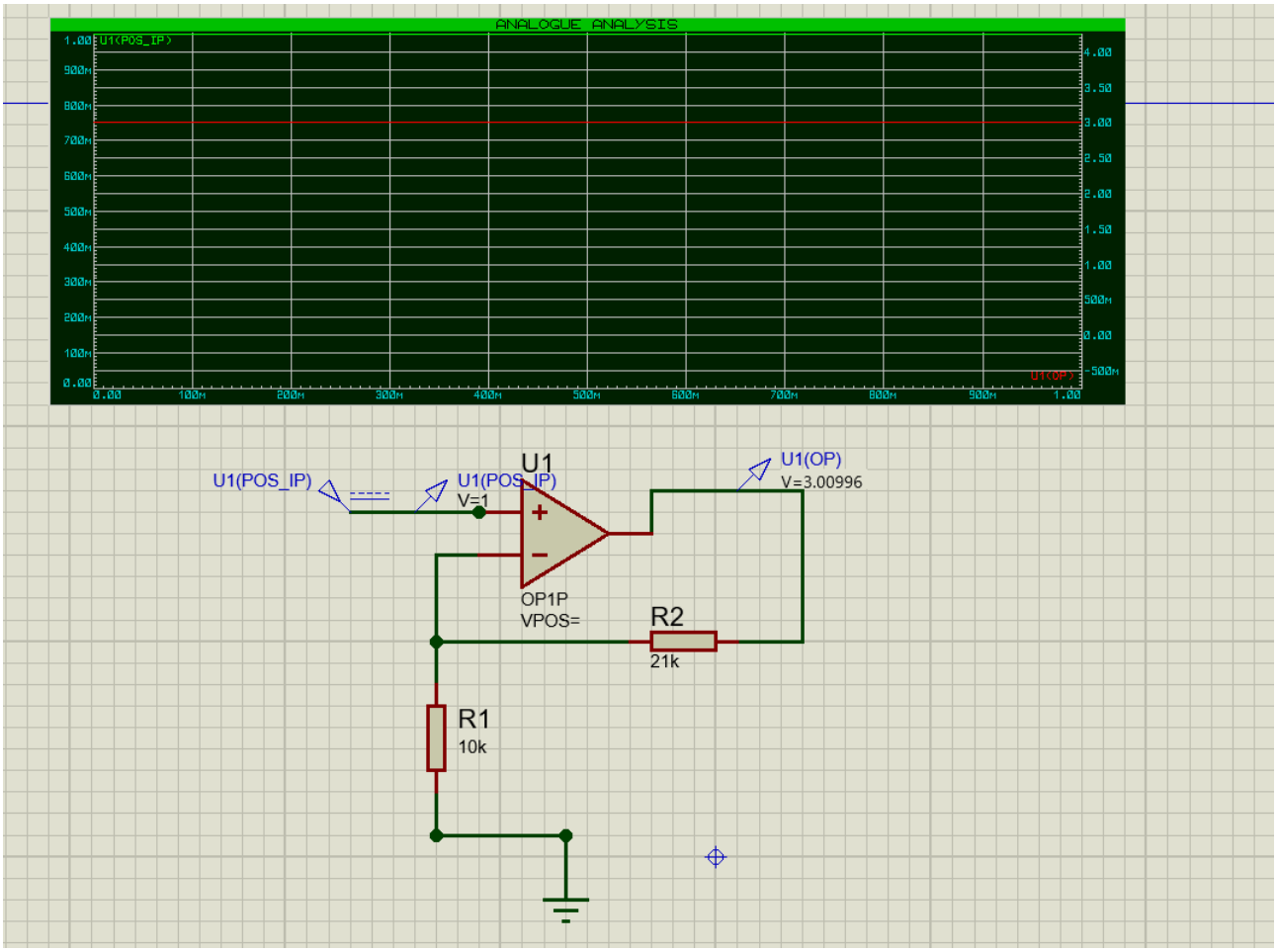
Comme son nom l'indique, le but de ce montage est d'amplifier une tension.



La formule :

$$Vs = Ve.\left(1+\frac{R2}{R1}\right) = 1*\left(1+\frac{21000}{10000}\right) = 3.1$$

Voici le résultat sur Isis, on a envoyé 1V et en sortie de l'amplificateur opérationnel on retrouve ~3V



Conclusion

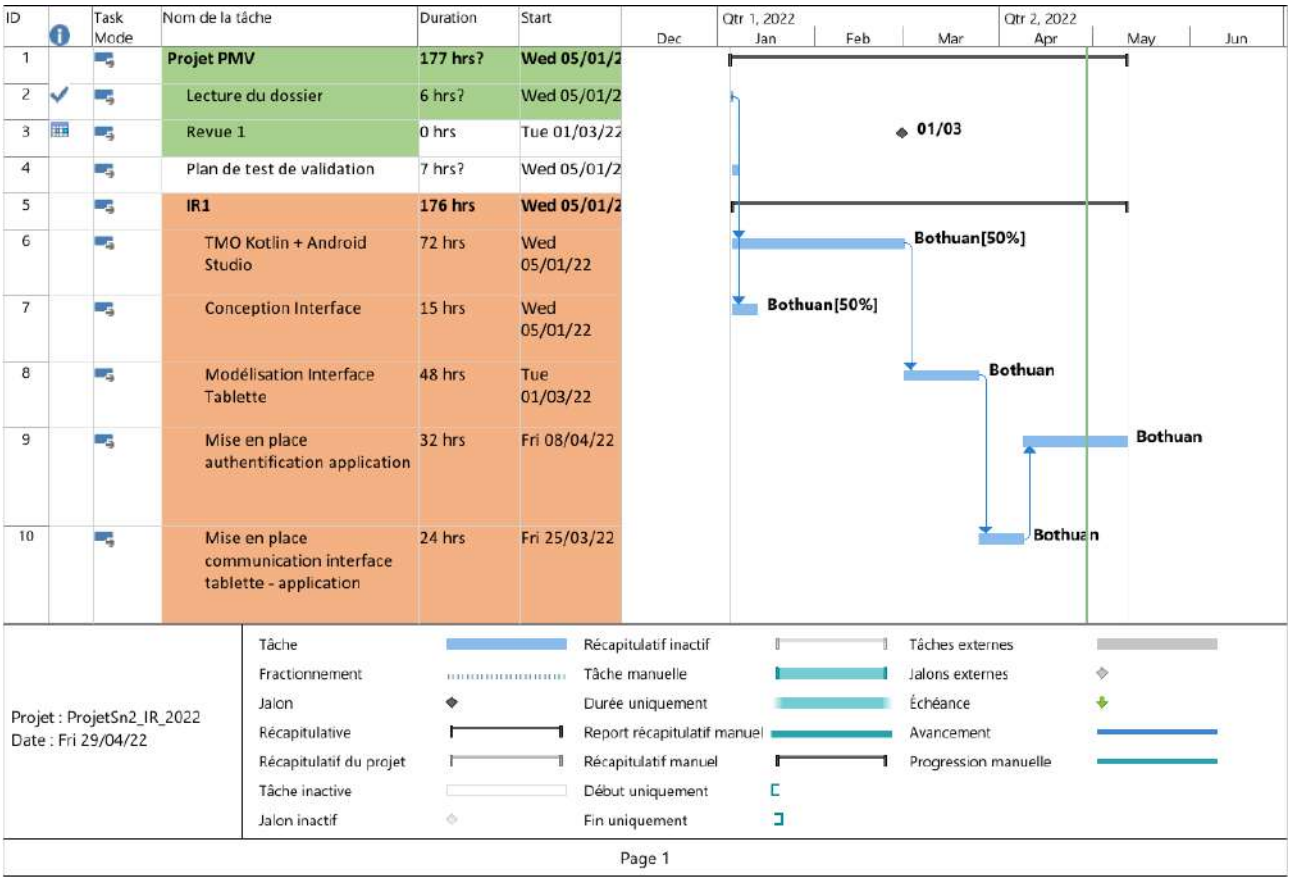
Mon but premier avant le passage de la revue 3 est de faire des test de trame i2c et de faire fonctionner la carte. Ainsi on sera en mesure de faire le "*mode d'emploi*"

PARTIE INDIVIDUELLE BOTHUAN ERWAN

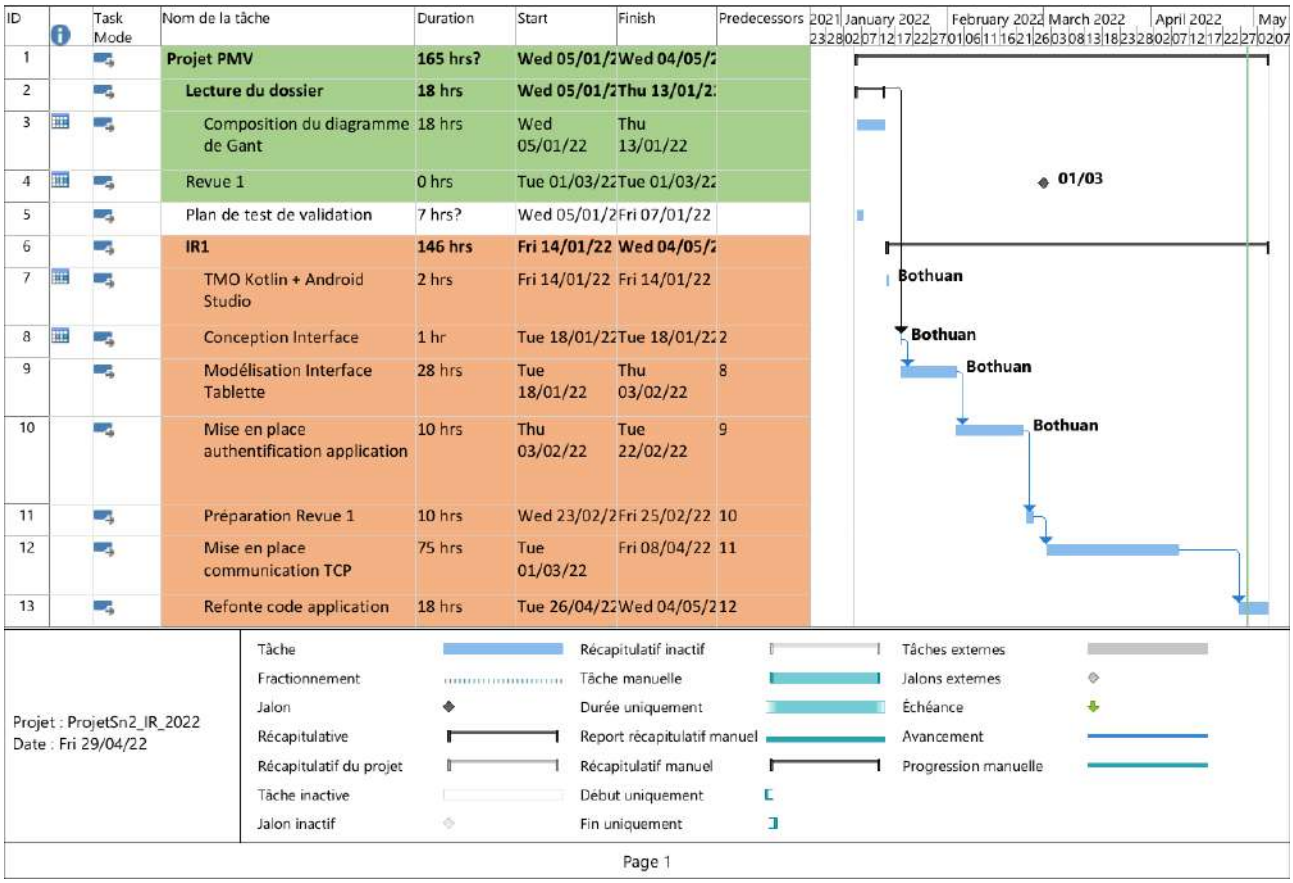
Durant ce projet, j'ai occupé le rôle d'IR1. Ce rôle est chargé du développement de l'application Android afin de bénéficier d'une interface de commande déportée par rapport au système principal. Mon choix s'est porté sur ce rôle grâce à l'utilisation d'un nouveau langage de programmation : le Kotlin. La découverte de ce langage me permet d'ajouter une nouvelle corde à mon arc en matière de connaissances et de compréhension en programmation et développement.

Partie Professionnelle

Gestion du projet



Planification prévisionnelle établie en début de projet



ID	Task Mode	Nom de la tâche	Duration	Start	Finish	Predecessors
1		**Projet PMV**	**165 hrs?**	**Wed 05/01/2**	**Wed 04/05/2**	
2		**Lecture du dossier**	**18 hrs**	**Wed 05/01/2**	**Thu 13/01/2**	
3		Composition du diagramme de Gant	18 hrs	Wed 05/01/22	Thu 13/01/22	
4		Revue 1	0 hrs	Tue 01/03/22	Tue 01/03/22	
5		Plan de test de validation	7 hrs?	Wed 05/01/2	Fri 07/01/22	
6		**IR1**	**146 hrs**	**Fri 14/01/22**	**Wed 04/05/2**	
7		TMO Kotlin + Android Studio	2 hrs	Fri 14/01/22	Fri 14/01/22	
8		Conception Interface	1 hr	Tue 18/01/22	Tue 18/01/22	2
9		Modélisation Interface Tablette	28 hrs	Tue 18/01/22	Thu 03/02/22	8
10		Mise en place authentification application	10 hrs	Thu 03/02/22	Tue 22/02/22	9
11		Préparation Revue 1	10 hrs	Wed 23/02/2	Fri 25/02/22	10
12		Mise en place communication TCP	75 hrs	Tue 01/03/22	Fri 08/04/22	11
13		Refonte code application	18 hrs	Tue 26/04/22	Wed 04/05/2	12

Planification réelle mise à jour pour la revue 2

De plus, tout au long du projet, le code source de l'application est envoyé sur un dépôt GitHub, afin de conserver une trace des différentes modifications successives, tout en donnant accès aux différents membres du projet aux sources depuis n'importe quel lieu disposant d'une connexion Internet.

Découverte du langage Kotlin

Afin d'accomplir ma tâche, il m'a fallut découvrir un nouveau langage de programmation : le Kotlin. Ce langage, développé par JetBrains (les développeurs d'Intellij Idea) et par Google depuis 2009, s'est récemment vu mettre en avant par Google pour le développement de son système d'exploitation mobile : Android.

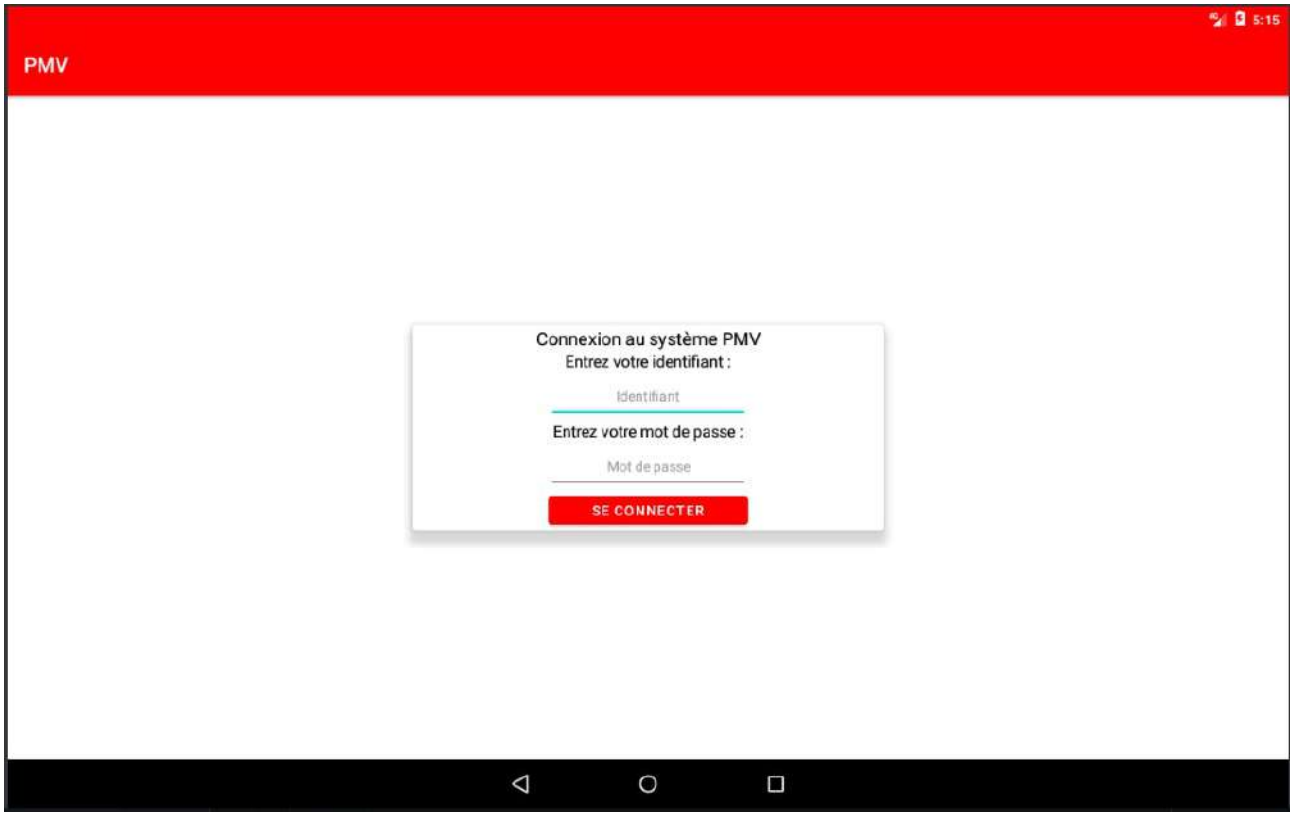
À l'heure actuelle, de nombreux composants système et applications sont développées en Java. Mais depuis quelques temps, l'utilisation du Java recule au profit du Kotlin, ce dernier permettant de nombreuses manipulations impossibles en Java, tout en ajoutant de nouvelles sécurités au dessus de ce dernier.

Cependant, une compatibilité totale entre Java et Kotlin est assurée, Kotlin étant interprété par la Machine Virtuelle Java (JVM), ce qui permet de maintenir une interopérabilité entre différents composants d'une même application, dont certains peuvent être développés en Java, alors que des composants plus récents peuvent quand à eux être développés en Kotlin.

Authentification de l'utilisateur

Durant la lecture du cahier des charges et après discussion avec le professeur en charge du projet, il a été décidée de mettre en place une authentification des utilisateurs. Cette authentification à pour but de prévenir l'utilisation non autorisée de l'application, et ainsi fausser les résultats.

Pour se faire, lors du lancement de l'application, une interface demandant un identifiant et un mot de passe apparaît. Sans les bons identifiants, il est impossible d'accéder au reste de l'application (l'interface de contrôle est lancée seulement après vérification des identifiants.



Interface d'authentification de l'utilisateur au lancement de l'application

Lorsque l'utilisateur cliquait sur le bouton "Se connecter", une connexion réseau s'établissait avec le serveur PMV afin de comparer les identifiants stockés dans la base de données pour lancer ou non l'activité suivante.

Le diagramme de séquence suivant présente le cheminement des informations lors du lancement de l'application et de l'interaction entre l'utilisateur et l'interface :

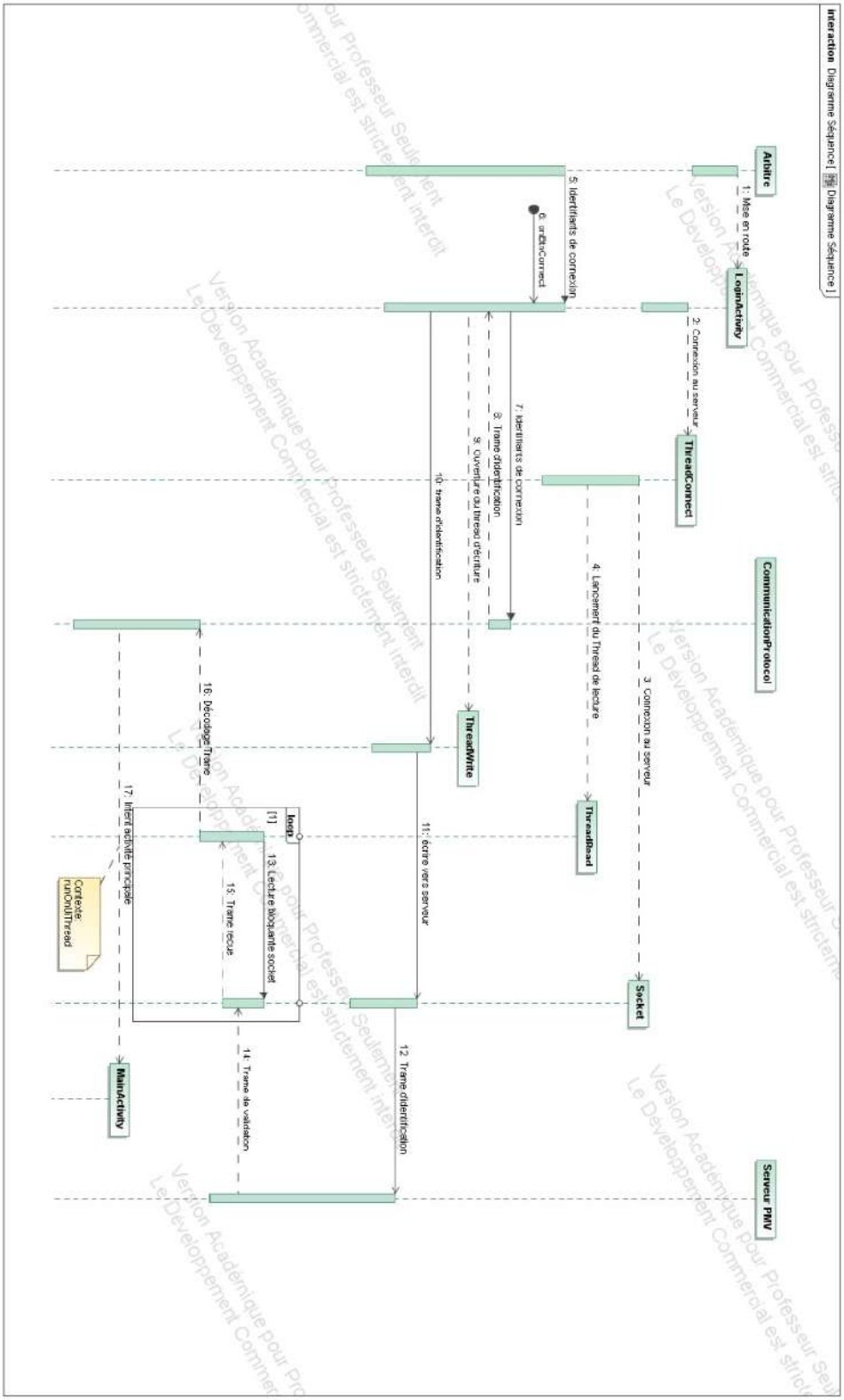


Diagramme de séquence ancienne version de l'application

Contrôle de la session de course

L'activité principale, qui présente l'ensemble des éléments permettant la gestion de course est lancée après la vérification des identifiants dans l'interface d'authentification.

Cette interface permettant un contrôle total sur la session de course en cours, a été composée de manière à être intuitive, tout en étant relativement proche de l'interface présente sur le Raspberry Pi et codée en Qt.

L'interface est composée de principalement deux parties :

- La partie gestion de course, avec les boutons permettant d'effectuer des actions sur l'ensemble du système.

Il est possible par exemple de lancer la session de course, puis de passer aux différentes étapes de la course : Préparation, À Vos Marques (AVM), Prêt, Partez (Go).

De plus, le bouton Go passe en Stop lorsqu'une course est en cours, afin d'arrêter la course en cas d'accident ou d'interruption.

Les boutons de la barre de gestion ne sont pas tous accessible en même temps. Un bouton ne sera cliquable que si le bouton précédent à été actionné. Cela évite ainsi de procéder à un lancement de course trop rapide en brulant les étapes.

Pour finir, l'ensemble des éléments de la barre de gestion de course ont été agrandi suite à la demande du commanditaire, la taille de base des éléments ayant été jugée trop petite.

```
fun onBtnPrepare(v: View){
    val toast = Toast.makeText( context: this, "Préparez-vous !", Toast.LENGTH_SHORT) // Création d'une infobulle
    toast.show() // Affichage de l'infobulle de message
    val buttons = arrayListOf<Button>(
        findViewById(R.id.btnAVM),
        findViewById(R.id.btnReady),
        findViewById(R.id.btnGo)
    ) //création d'une liste contenant les différents boutons de gestion de course
    buttons[0].isEnabled=true // Activation du bouton "À vos marques!"
    buttons[1].isEnabled=false // Désactivation du bouton "Prêt !"
    buttons[2].isEnabled=false //Désactivation du bouton "Partez !"
} // Fonction exécutée lors du clic sur le bouton "Préparation
```

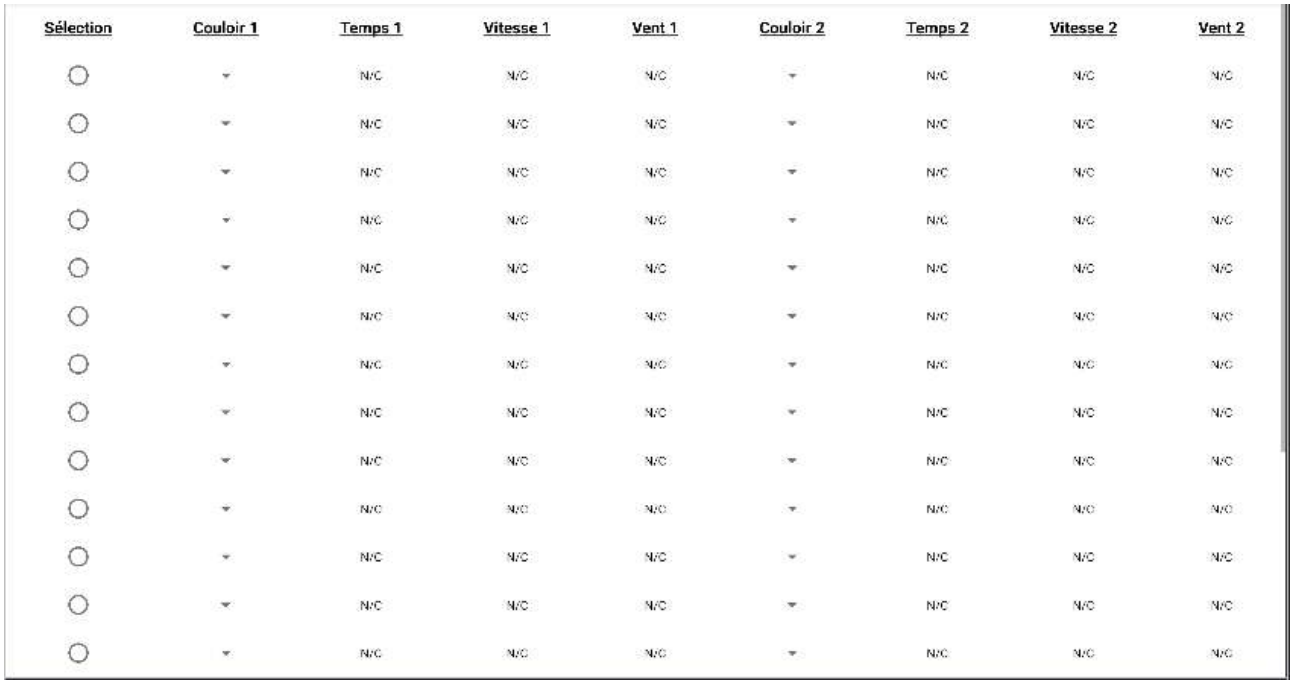
Exemple de code d'un des boutons de la barre de gestion

Interface de la barre de gestion de session

- La partie composition des courses et affichage des résultats en temps réel

La partie gauche présente des boutons radio qui permettent de sélectionner la ligne de course désirée, puis des boites de sélection permettent de choisir les élèves qui doivent effectuer la dite course. Ensuite, une fois l'élève arrivé en fin de course, son temps, sa vitesse moyenne et la vitesse du vent pour ce coureur sont affiché.

Il est possible d'effectuer 20 course, ce qui permet de faire passer au total 40 élèves par session.



Refonte complète du code de l'application

Après avoir développé une première version de l'application, il a été décidé avec Monsieur Antoine d'effectuer une refonte profonde du code source de l'application. En effet, même si fonctionnelle, l'application n'était pas très optimisée, et le code source était illisible. De plus, l'ancien code ne bénéficiait pas de la puissance du langage Kotlin.

Dans un premier temps, j'ai réduit le nombre d'activité à une (contre deux précédemment), et la gestion de la connexion puis de la session auparavant effectuée par deux activités distinctes et maintenant gérée par une seule activité composée de fragments interchangeables. Les fragments sont des morceaux d'interface, permettant de composer un affichage avec des groupes complets d'éléments, et comportant des fonctions qui leurs sont propres.

De plus, le fonctionnement de la communication TCP est retravaillé. Auparavant, la communication se faisait au moyen de trois thread (un pour la connexion, un pour la lecture, et un pour l'écriture). Cette technique était l'une des options possibles pour contourner le blocage qu'effectue Android, celui-ci interdisant les communications réseau sur le thread principal de l'application. Dorénavant, la communication se fera grâce à des coroutines, un concept propre à Kotlin qui permet une gestion plus propre des processus parallèles au processus principal.

Les changements mentionnés ci-dessus permettront, a terme, de faciliter les évolutions du code de l'application (*cf. Diagrammes de séquence en annexe)*

Protocole de communications

La communication des informations entre les éléments est effectuée en suivant un protocole au format JSON, contenant une commande et des données supplémentaires au besoin.

Le format JSON a été choisi pour sa facilité d'organisation des données et sa lisibilité. Grâce au JSON, les différentes valeurs se trouvent sous forme d'objets, ce qui rend le traitement de ces valeurs bien plus simple grâce aux différentes bibliothèques disponibles dans la plupart des langages de programmation actuels.

Le squelette d'une trame ressemble à ceci :

Modèle de base d'une trame protocolaire

```
{
  "command": "<command>",
  "data": {
    "<some data>": "<data>"
  }
}
```

Par exemple, la trame d'authentification ressemble à ceci :

authCheck

Permet de vérifier les identifiants de connection (bi-directionnel)

Cas Application vers Raspberry:

```
{
  "command": "authCheck",
  "data": {
    "login": "<identifiant>",
    "pass": "<mot de passe>"
  }
}
```

(Une liste des différentes trames disponibles est située en Annexe.)

Traitement des trames entrantes

Lorsque le fragment de gestion de session est lancé, l'activité principale lance en parallèle la lecture constante des trames entrantes sur la socket de connexion entre le serveur et l'application.

Pour se faire, au lancement du fragment de gestion de session, le fragment définit comme listener l'activité principale. C'est donc cette dernière qui va avoir pour rôle de récupérer les informations envoyées sur l'interface de la classe du fragment, pour ensuite exécuter les fonctions nécessaires au bon fonctionnement de l'application.

Par exemple, voici comment est défini le listener de la classe FragmentSession, qui a en charge la gestion du fragment de gestion de course.

Tout d'abord, nous définissons une interface. C'est grâce à elle que nous allons pouvoir mettre en place des fonctions qui permettront une communication entre l'activité principale et le fragment en cours.

```
interface OnSessionManagement{
    fun onSessionRunning(settings: DataApp)
    fun onSendCommand(data: DataSend)
    fun onEndSession(settings: DataApp)
    fun onUpdateSession(type: ReceiveActions, data: ArrayList<Any>)
}
```

Cette interface dispose de 4 fonctions avec un rôle précis pour chacune d'elles, et toutes prennent des paramètres afin d'agrandir le champs des actions réalisables par celles-ci.

Nous noterons que chaque fonction de cette interface prend des paramètres dont les types sont fait maison, comme le type DataApp, qui est un type de classe permettant de stocker des données facilement. La classe est définie ci-dessous :

```
data class DataApp(
    val login: String = "login",
    val password: String = "passwd",
    val SERVER_ADDRESS: String = "192.168.13.226",
    val SERVER_PORT: Int = 4444,
    var message: String = "Nothing"
)
```

Grâce à cette classe, il est plus simple de transmettre des données récurrentes, sans pour autant avoir à multiplier la liste des paramètres d'une fonction.

Un autre exemple de type de données fait maison est le type ReceiveActions. Ce paramètre utilise une classe d'énumérateur. Cela signifie qu'elle ne peux avoir que certains valeurs précises définies préalablement. Voici la déclaration de cette classe enum :

```
enum class ReceiveActions{
    CONTROL,
    BUTTON,
    RUNS, //Servira dans une version future de l'application
    RUNNERS,
    SESSION
}
```

Cette classe a pour objectif de réunir les différents types de commandes pouvant être reçues par l'application.

Une fois l'interface déclarée dans la classe de gestion du Fragment, nous l'implémentons dans la classe de l'activité principale.

Cette dernière aura donc pour but de définir les actions qui seront effectuées après réception des données, grâce à l'interface.

Voici un exemple d'implémentation d'une des fonctions de l'interface dans l'activité principale :

```
override fun onSessionRunning(settings: DataApp) {
    _client.alwaysReadFromServer()
}
```

```
override fun onUpdateSession(type: ReceiveActions, data: ArrayList<Any>) {
    val f: FragmentSession = supportFragmentManager.findFragmentByTag( tag: "FrgmSess") as FragmentSession
    when(type) {
        ReceiveActions.CONTROL -> {
            f.disableInterface()
        }
        ReceiveActions.RUNNERS -> {
            f.setRunnersList(data)
        }
        ReceiveActions.SESSION -> {
            f.restoreSession(data)
        }
        ReceiveActions.BUTTON ->
            f.controlButtons(data)
        else -> {}
    }
}
```

Lorsque l'interface est implémentée sur l'activité principale, il est nécessaire de définir cette implémentation comme listener de notre fragment lors de l'instanciation, comme ci-dessous :

```
private lateinit var _listener: OnSessionManagement
private lateinit var _rootView: View

override fun onAttach(context: Context) {
    super.onAttach(context)
    _listener = if(context is OnSessionManagement) context else let {
        throw ClassCastException("$context must implement params")
    }
}
```

Puis lors de l'affichage du fragment, nous lançons une des fonctions de l'interface, à savoir celle qui lance la lecture continue des trames sur la socket.

```
Log.d( tag: "FragmentSession", msg: "Starting onSessionRunning")
_listener.onSessionRunning(DataApp(_login!!, _password!!, _serverAddress!!, _serverPort!!.toInt(), message: "onSessionRunning"))
```

La fonction alwayReadFromServer va, au sein d'une Coroutine, lire en permanence les trames puis grâce à une instance de la classe du protocole de communication, déchiffrer les trames pour pouvoir en extraire et traiter les données.

```
fun alwaysReadFromServer(){
    _endThread = false
    _end = false
    CoroutineScope(Dispatchers.Main).launch(Dispatchers.IO){ this: CoroutineScope
        while(!_end){
            val response = readWithTimeout(TIMEOUT)
            val lines = _protocol.decodeData(response)
            if(lines[0].toString().isNotEmpty()){
                _activity.runOnUiThread{
                    when(lines[0].toString()) {
                        "getControl" -> _activity.onUpdateSession(ReceiveActions.CONTROL, arrayListOf())
                        "transfertAllRunners" -> {
                            lines.removeAt( index: 0)
                            _activity.onUpdateSession(ReceiveActions.RUNNERS, lines)
                        }
                        "sessionTransfert" -> {
                            lines.removeAt( index: 0)
                            _activity.onUpdateSession(ReceiveActions.SESSION, lines)
                        }
                        "btnState" -> {
                            lines.removeAt( index: 0)
                            _activity.onUpdateSession(ReceiveActions.BUTTON, lines)
                        }
                    }
                }
            }
        }
        _endThread = true
    }
}
```

Toutefois, la lecture sur le socket étant bloquante, nous ne pouvons pas nous permettre de prendre de risque d'essayer de lire sur une socket vide et donc de bloquer une Coroutine. C'est pour prévenir ce problème que nous utilisons une fonction qui détecte la présence

de données sur la socket, et de les lires, tout en conservant un délai maximum en cas d'absence de données et ainsi éviter de bloquer un processus.

```
private fun readWithTimeout(time: Long): String{
    var response = "NOTHING"
    var tempo: Long = 0
    val delta: Long = 50
    val bReader = BufferedReader(InputStreamReader(_reader))
    while(tempo < time){
        val nb = _reader.available()
        if(nb > 0){
            response = bReader.readLine()
            break
        } //if
        runBlocking{ this: CoroutineScope
            delay(delta)
        } // runBlocking
        tempo += delta
    } // while
    if(tempo == time){
        response = "NO RESPONSE"
    } //if
    return response
} //readWithTimeout
```

Une fois les données reçues, elles sont transmises au protocole, qui les déchiffres grâce à la fonction suivante, qui a pour but d'analyser la trame au format json :

```
fun decodeData(data: String): ArrayList<Any>{
    Log.d( tag: "decodeData1", data)
    var command: String
    var jsonObject: JSONObject
    try{
        jsonObject = JSONTokener(data).nextValue() as JSONObject
        command = jsonObject.getString( name: "command")
        Log.d( tag: "decodeData2", command)
    }catch(e: Exception){
        Log.d( tag: "decodeData2E", data)
        jsonObject = JSONTokener( in: "{\"command\": \"NO JSON\"}").nextValue() as JSONObject
        command = jsonObject.getString( name: "command")
    }

    return when(command){
        "authCheck" -> {
            val jsonData = jsonObject.getJSONObject( name: "data")
            Log.d( tag: "decodeData3", jsonData.toString())
            val authStatus = jsonData.getInt( name: "success")
            Log.d( tag: "decodeData4", authStatus.toString())
            if(authStatus == 1) arrayListOf("authTrue") else arrayListOf("authFalse")
        }
        "getControl" -> {
            Log.d( tag: "decodeData5", msg: "getControl")
            arrayListOf("getControl")
        }
        "transfertAllRunners" -> {
            Log.d( tag: "decodeData7", msg: "transfertAllRunners")
            val runners: ArrayList<Any> = decodeTransfertAllRunners(jsonObject.getJSONObject( name: "data")) as ArrayList<Any>
            runners.add( index: 0, element: "transfertAllRunners")
            runners
        }
        "sessionTransfert" -> {
            Log.d( tag: "Prot::decodeData", msg: "sessionTransfert")
            val session: ArrayList<Any> = decodeSessionTransfert(jsonObject.getJSONObject( name: "data")) as ArrayList<Any>
            session.add( index: 0, element: "sessionTransfert")
            session
        }
        "btnState" -> {
            Log.d( tag: "Prot::decodeData", msg: "btnState")
            val btnState: ArrayList<Any> = decodeBtn(jsonObject.getJSONObject( name: "data")) as ArrayList<Any>
            btnState.add( index: 0, element: "btnState")
            btnState
        }
        else -> {
            Log.d( tag: "decodeData6", msg: "REC UNKNOWN")
            arrayListOf("UNKNOWN !")
        }
    }
}
```

Les données une fois traitées sont transmises sous forme de Liste de Données (ArrayList) à la fonction onUpdateSession de l'interface de notre fragment, pour modifier l'affichage de notre application.

```
override fun onUpdateSession(type: ReceiveActions, data: ArrayList<Any>) {
    val f: FragmentSession = supportFragmentManager.findFragmentByTag( tag: "FrgmSess") as FragmentSession
    when(type) {
        ReceiveActions.CONTROL -> {
            f.disableInterface()
        }
        ReceiveActions.RUNNERS -> {
            f.setRunnersList(data)
        }
        ReceiveActions.SESSION -> {
            f.restoreSession(data)
        }
        ReceiveActions.BUTTON ->
            f.controlButtons(data)
        else -> {}
    }
}
```

Enfin, la fonction de notre interface lancera une fonction de notre fragment afin de modifier l'affichage de notre application. Par exemple, voici la fonction qui permet de récupérer le nom de la session ainsi que de mettre en place la liste des coureurs dans les boites de sélections de courses :

```
fun setRunnersList(runners: ArrayList<Any>){
    val runnersList: ArrayList<String> = runners as ArrayList<String>
    _rootView.findViewById<TextView>(R.id.txtSessionName).text = String.format("%1$s", runnersList[0])
    runnersList.removeAt( index: 0)
    val arrayAdapter: ArrayAdapter<String> = ArrayAdapter<String>(_rootView.context, android.R.layout.simple_spinner_dropdown_item, runnersList)
    initSpinners(getSpinners(_rootView), arrayAdapter)
}
```

Pour finir, vous pourrez trouver en annexe le diagramme de classe de cette nouvelle version de l'application.

Partie Physique

Protocole WIFI

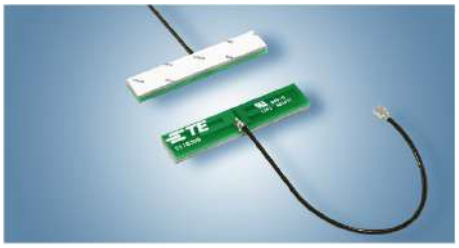
Afin d'assurer la communication entre les deux éléments du système, une communication TCP est établie sur une connexion WIFI. Le WIFI (ou Wireless Fidelity), est un ensemble de protocoles de communication sans fil.

Il permet de relier plusieurs appareils au moyen d'ondes radio, dans les bandes de fréquence 2.4GHz et 5GHz (pour les dernières normes les plus répandues). Actuellement, la norme WIFI-5 utilise des canaux de 80MHz sur la bande des 5GHz et 2.4GHz, atteignant des débits théoriques de 1 300 Mbits/s par canal, ce qui permet d'obtenir un débit théorique total de 7Gbits/s. La puissance des ondes WIFI est d'environ 30mW, ce qui les rend bien moins dangereuses que les ondes émises par nos téléphones mobiles, qui atteignent les 600mW. Cette différence de puissance à permis de classer ces ondes comme inoffensive pour la santé.

La plupart des périphériques utilisent des antennes omnidirectionnelle pour capter les ondes WIFI, avec une PIRE maximale de 100mW ou 20dBm.

Antenne WIFI

Les antennes WIFI présentes dans la Raspberry et dans la tablette Lenovo TB-X605F (tablette fournie par la région aux professeurs de lycée, et qui est utilisée pour ce projet) sont des antennes internes, dont la documentation n'est pas disponible. Afin de vous présenter une antenne correspondant aux fréquences sus-mentionnées, nous prendrons l'exemple de l'antenne *Antenne WiFi Omnidirectionnelle MCIS, MHF, U.FL Externe WiFi (Dual Band) (ref.2118903-1)* de chez *TE CONNECTIVITY*, qui est une antenne compatible avec l'ensemble des bandes de fréquences utilisées par les différents périphériques de notre système.

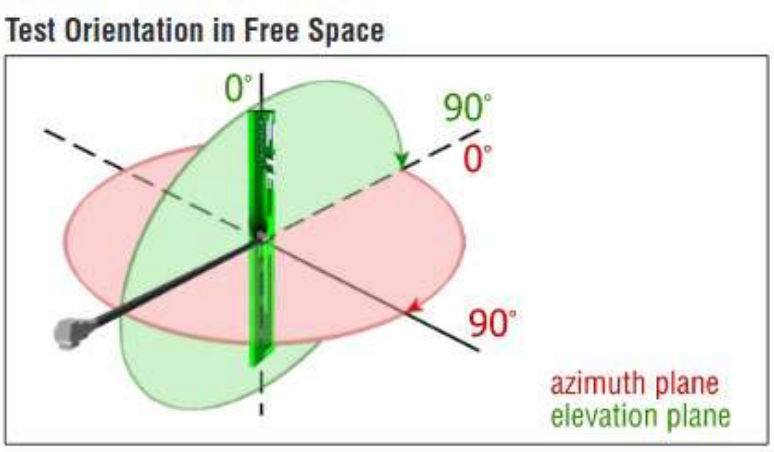


Specifications

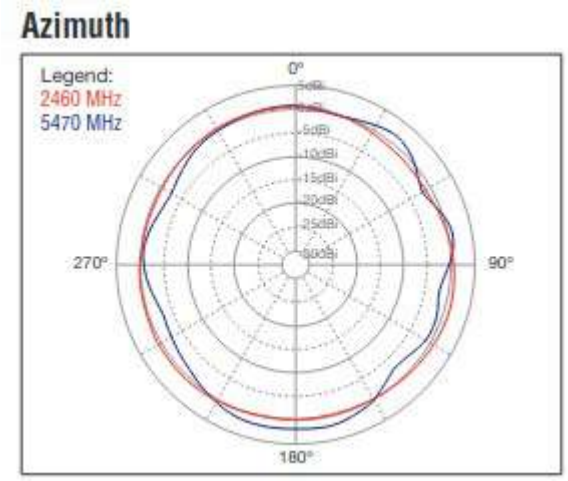
Frequency Range (MHz)	2400–2483.5; 4900–5875
Peak Gain	+3.7 dBi
VSWR	< 2.0:1
Polarization	Linear
Azimuth Beamwidth	Omni-directional
Power Handling	3 Watt cw
Feed Point Impedance	50 Ohms unbalanced
Size	40.0 mm x 8.0 mm x 1.0 mm
Weight	1.4 g.
Mounting	Adhesive. See diagram on page 2.
Keep Out Area	See diagram on page 2.
Cable / Connector	120 mm length. 1.13 mm dia. with U.FL connector

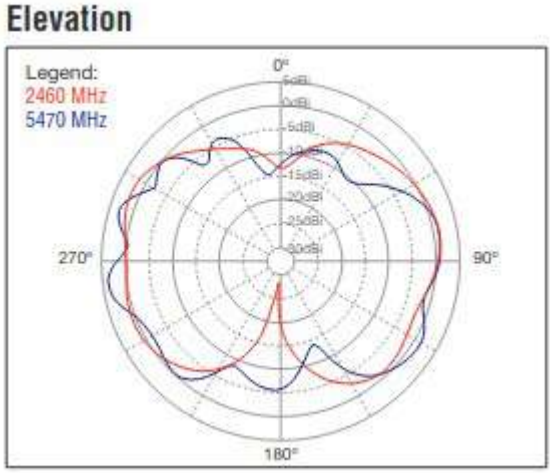
Cette antenne peut émettre sur les bandes de fréquences 2,4GHz et 5GHz et dispose des caractéristiques suivantes :

Le gain G_{dBi} de +3.7dBi est égal à un gain G de 2,344.



<u>Diagrammes de rayonnement :</u>





Sur les schémas ci-dessus, nous pouvons remarquer que sur l'azimuth, le rayonnement de cette antenne est équivalent à une antenne isotrope, malgré la baisse de puissance par endroit sur la bande de fréquences de 5GHz, cette bande étant plus sensible aux interférences.

Pour ce qui est de l'émission dans un plan spatial, nous pouvons remarquer que l'émission est bien moins régulière, ce qui l'éloigne du modèle de l'antenne isotrope. Comme précédemment, nous remarquons que l'émission sur la bande des 5GHz est plus sensible et plus fluctuante que la bande des 2,4GHz.

L'ensemble de la documentation est fournie par le constructeur : TE CONNECTIVITY

Annexes

Protocole.md

Type de protocole:

Protocole TCP avec formatage JSON

Le Protocole

Modèle de base d'une trame protocolaire

```
{
  "command": "<command>",
  "data": {
    "<some data>": "<data>"
  }
}
```

Liste des commandes disponibles:

transfertRunner

Permet de transmettre la liste des coureurs sélectionnés pour une course (bi-directionnel)

```
{
  "command": "transferRunner",
  "data": {
    "id": <numéro de course>,
    "runners": [
      <id coureur 1>,
      <id coureur 2>
    ]
  }
}
```

transfertAllRunners

Permet de transmettre l'ensemble des coureurs disponibles à l'application (uni-directionnel)

```
{
  "command": "transfertAllRunners",
  "data": {
    "runnersCnt": <nombre de coureurs de la session>,
    "runner1": {
      "id": <id coureur dans la BDD>,
      "name": "<Nom + Initiale du prénom du coureur>"
    },
    "runner2": {
      "id": <id coureur dans la BDD>,
      "name": "<Nom + Initiale du prénom du coureur>"
    },
    ...
  }
}
```

timeRun

Permet de transmettre le temps d'une course spécifiée (uni-directionnel)

```
{
  "command": "timeRun",
  "data": {
    "id": <id de la course>,
    "time1": "<temps sous forme de chaine de caractères>",
    "wind1": "<vent sous forme de chaine de caractères>",
    "speed1": "<vitesse sous forme de chaine de caractères>",
    "time2": "<temps sous forme de chaine de caractères>",
    "wind2": "<vent sous forme de chaine de caractères>",
    "speed2": "<vitesse sous forme de chaine de caractères>"
  }
}
```

getCsv

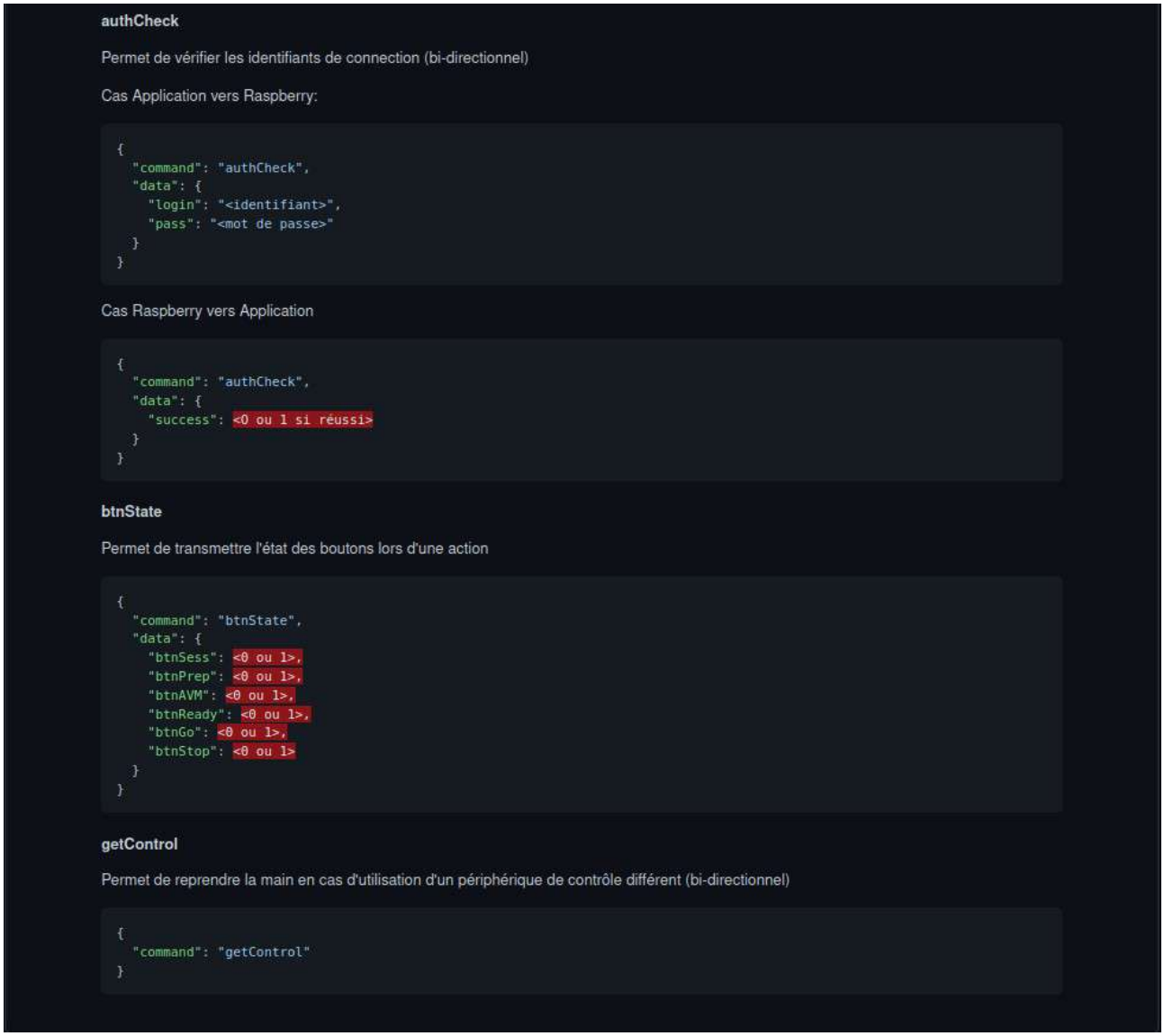
Permet de demander un fichier CSV contenant l'ensemble de la course (uni-directionnel)

```
{
  "command": "getCsv"
}
```

transfertCsv

Permet de transmettre les données pour le fichier CSV à la tablette (uni-directionnel)

```
{
  "command": "transfertCsv",
  "data": {
    "runnersCnt": <nombre de coureurs>,
    "runner1": {
      "idRun": <id course>,
      "name": "<nom coureur>",
      "time": "<temps en CDC>",
      "wind": "<vent en CDC>",
      "speed": "<vitesse en CDC>"
    },
    "runner2": {
      "idRun": <id course>,
      "name": "<nom coureur>",
      "time": "<temps en CDC>",
      "wind": "<vent en CDC>",
      "speed": "<vitesse en CDC>"
    },
    ...
  }
}
```



authCheck

Permet de vérifier les identifiants de connection (bi-directionnel)

Cas Application vers Raspberry:

```
{
  "command": "authCheck",
  "data": {
    "login": "<identifiant>",
    "pass": "<mot de passe>"
  }
}
```

Cas Raspberry vers Application

```
{
  "command": "authCheck",
  "data": {
    "success": <0 ou 1 si réussi>
  }
}
```

btnState

Permet de transmettre l'état des boutons lors d'une action

```
{
  "command": "btnState",
  "data": {
    "btnSess": <0 ou 1>,
    "btnPrep": <0 ou 1>,
    "btnAVM": <0 ou 1>,
    "btnReady": <0 ou 1>,
    "btnGo": <0 ou 1>,
    "btnStop": <0 ou 1>
  }
}
```

getControl

Permet de reprendre la main en cas d'utilisation d'un périphérique de contrôle différent (bi-directionnel)

```
{
  "command": "getControl"
}
```

Liste des commandes disponibles au sein du protocole

Diagramme de séquence ancienne version application

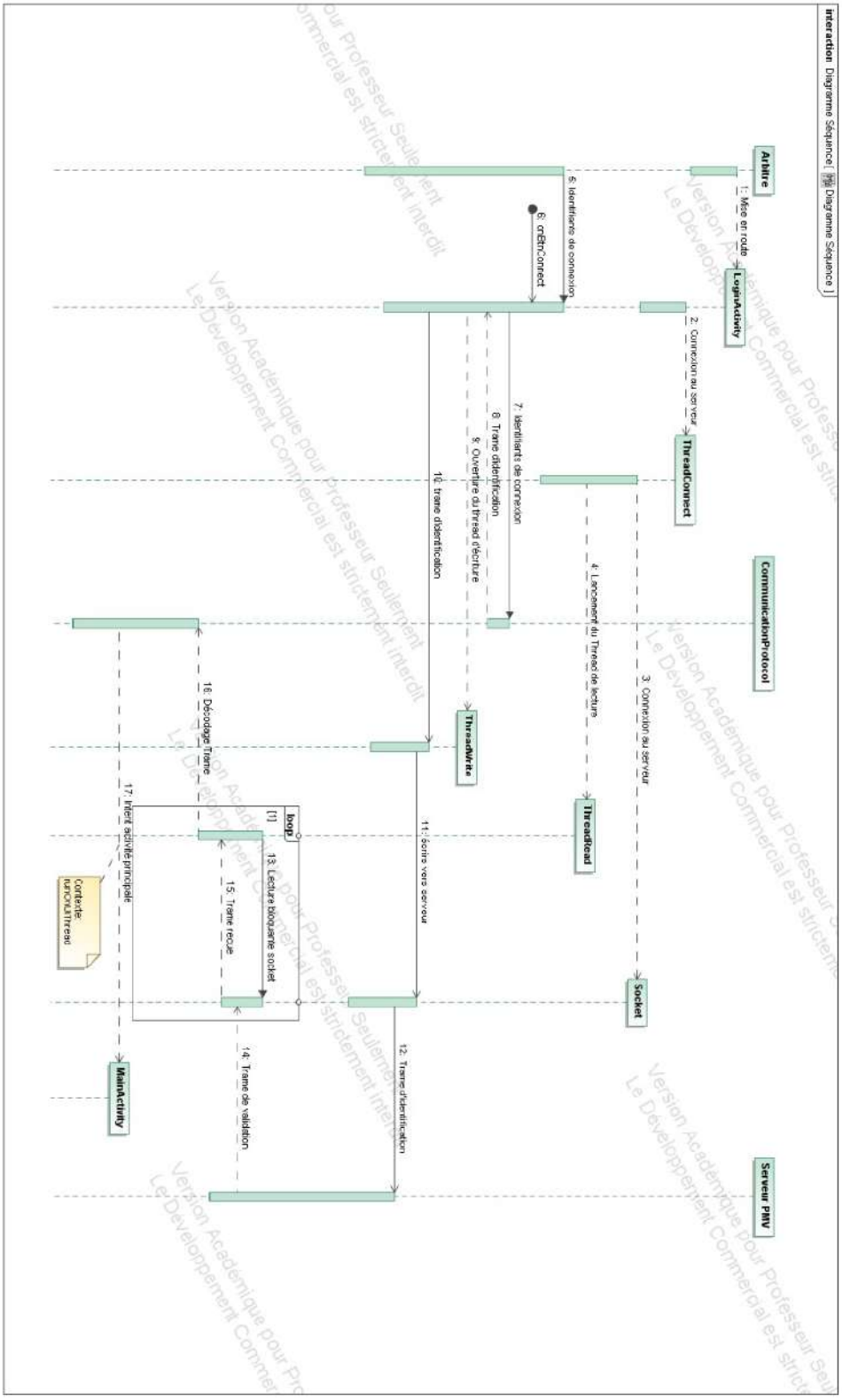


Diagramme de séquence nouvelle version de l'application

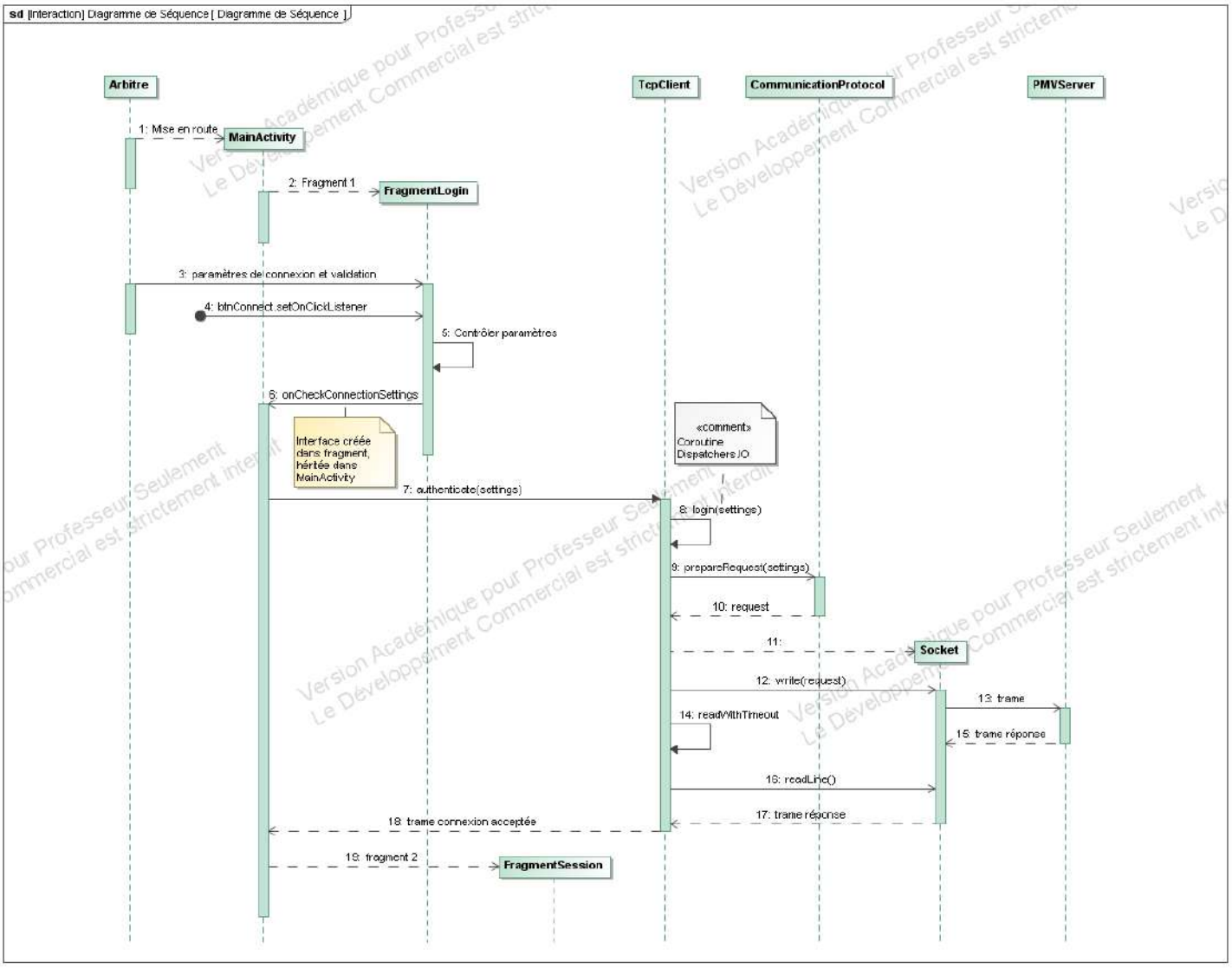


Diagramme de classe avant refonte du code

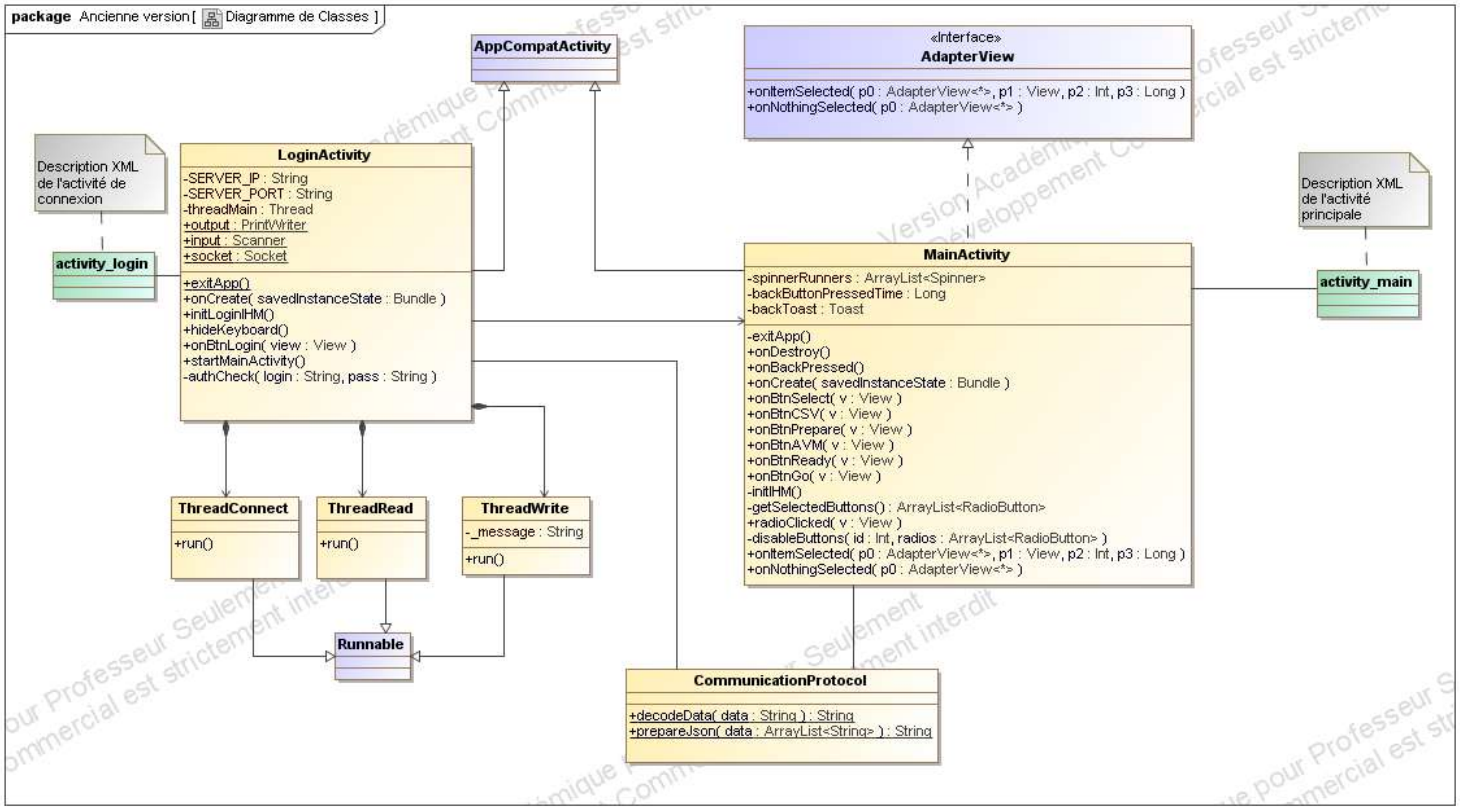


Diagramme de classe après refonte du code