BTS systèmes numériques

Projet AIES

BTS Systèmes Numériques

2018 - 2020

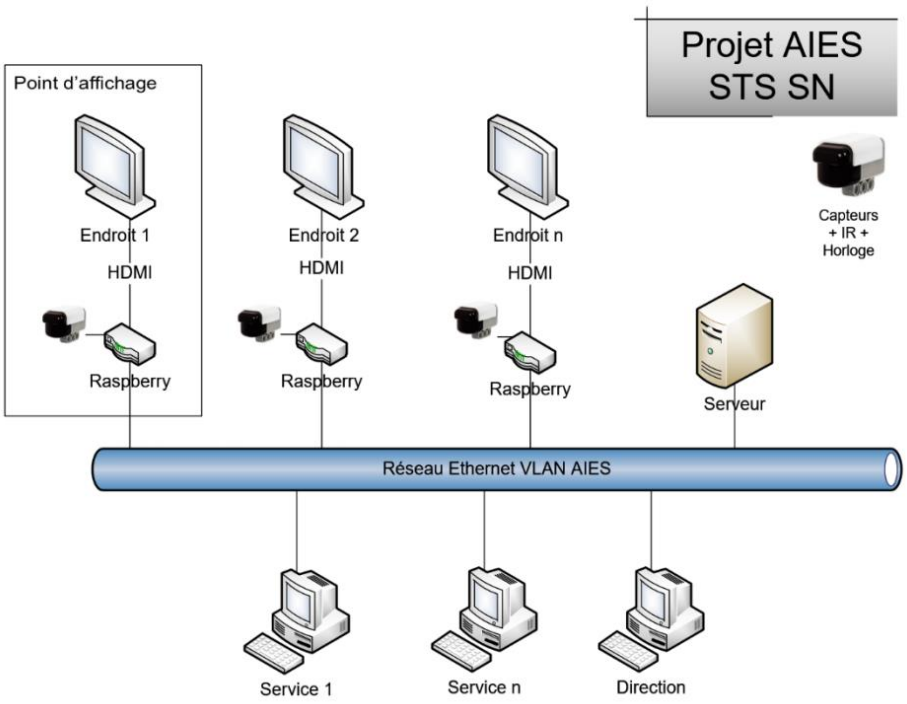
Zehani Ryan-Jacques, Jacques Rémi, Harduin Tom, Grosset Killian

Sommaire

1 Présentation et situation du projet

1.1 Présentation du projet

Le projet consiste en l'amélioration d'un système d'affichages dynamiques d'informations contrôlables par l'outil informatique. Le système nous a été demandé par le Lycée Benoit. Ce système est composé de plusieurs points d'affichages répartis dans des endroits clé du lycée. Chaque service a la possibilité de publier des informations. Des informations flashs peuvent être possiblement affichées dans la partie basse des écrans d'affichage ainsi que des vidéos et des images pour prévenir d'événements spéciaux.

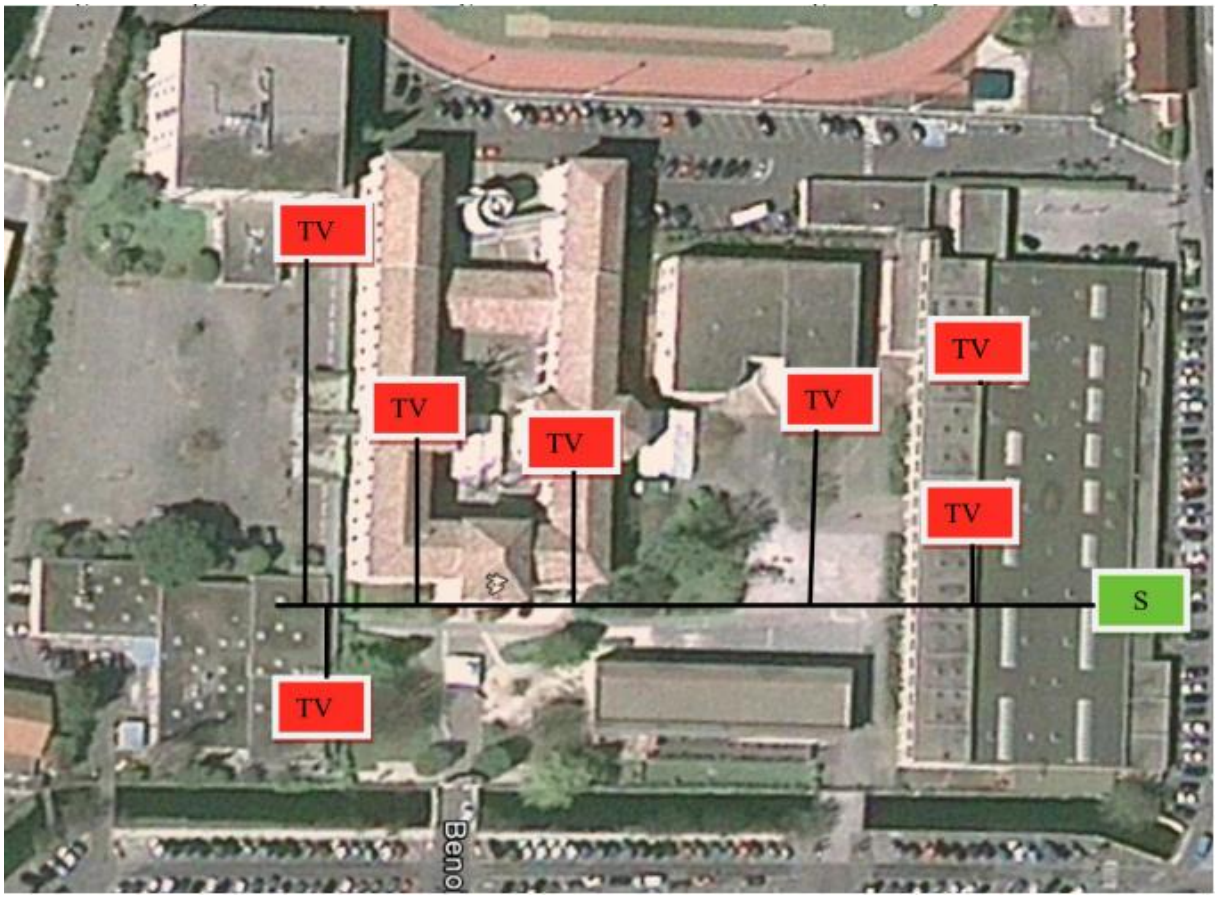


1.2 Cahier des charges – Expression du besoin

La direction du lycée BENOIT est soucieuse de délivrer une information plus dynamique, moderne, attractive et plus pertinente auprès des personnels et des élèves. De ce fait, le lycée désire se doter d'un système d'affichages dynamiques d'informations, installés dans les endroits passants.
Le projet se voudra évolutif, il sera possible dans l'avenir d'ajouter des points d'affichage (en fonction des besoins, du budget et de la couverture réseau de l'établissement).

Le système a été déployé durant l'année 2016-2017,2018 et enfin 2020. Sept points d'affichages sont fonctionnels, le huitième sera ajouté en cours d'année.

Les points d'affichages seront utilisés afin de renseigner les lycéens, les enseignants et le personnel sur les événements du lycée, les professeurs absents, ainsi que des valeurs fournies par des capteurs connectés à une carte Raspberry Pi (Température, qualité de l'air, détection de fumée).



2 Spécification

2.1 Global

Un point d'affichage (PA) est composé :

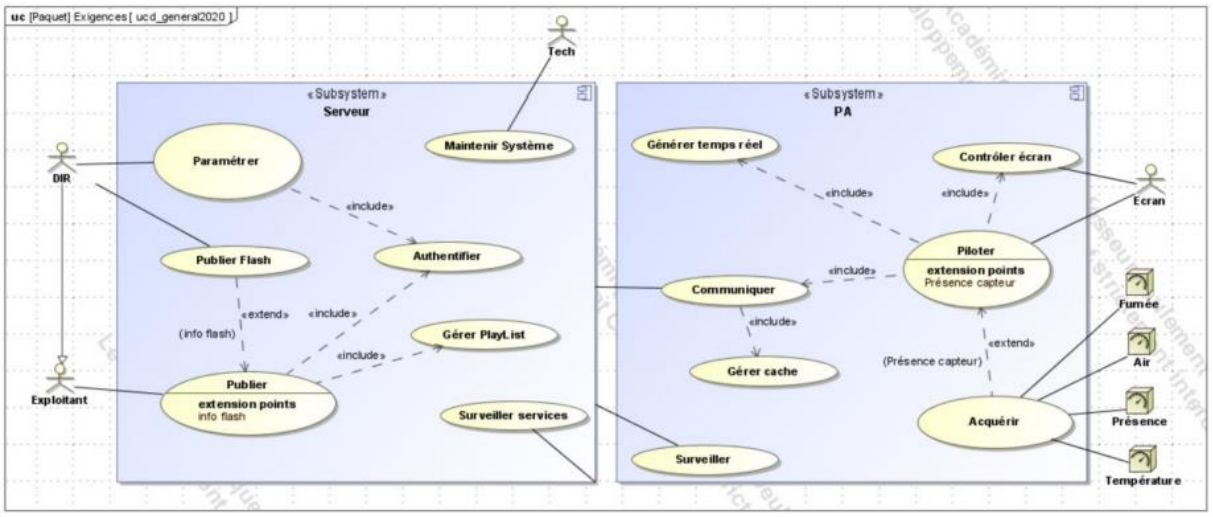
- D'un écran de télévision.
- D'une carte RASPBERRY PI 2 B+ reliée par la prise HDMI.
- Un capteur de présence afin de permettre l'extinction automatique des écrans lorsque personne ne passe (temporisé 1h).
- Un capteur de température.
- Un capteur de qualité de l'air.
- Un détecteur de fumée.
- Carte de réception d'une horloge (synchronisée ou pas).
- Un émetteur infrarouge pour commander les TV non compatibles avec la norme CEC (nouveauté 2017).

2.2 Ajouts pour l'année 2020

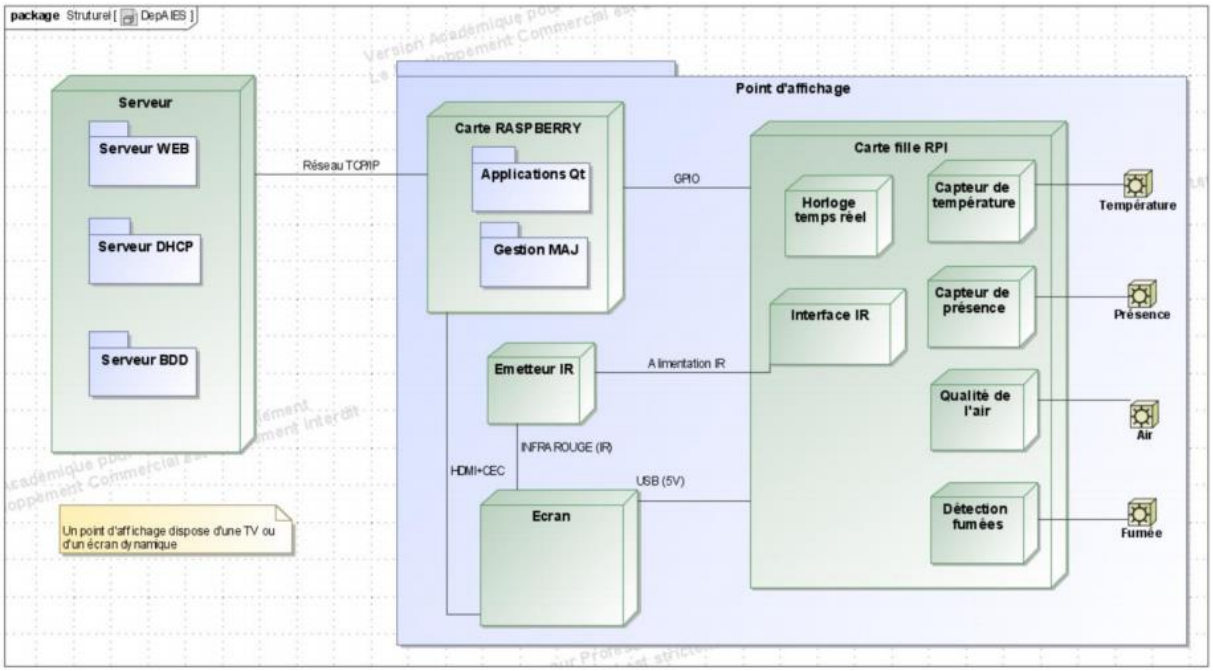
Ce projet aura une mise à jour pour 2020. Un nouveau point d'affichage sera installé, l'application sera mise à jour ainsi que le site web. Il sera possible d'afficher des images et des vidéos dans les diapositives. L'espace de stockage sur la marte micro SD sera visible depuis le logiciel. Une automatisation au lancement du logiciel si le serveur n'est pas disponible, ainsi qu'un relancement en cas de perte de connexion avec le serveur. Le « Shield » subira une refonte afin de pouvoir accueillir les deux nouveaux capteurs ainsi que la suppression d'une partie des composants.

3 Diagramme UML/SYSML

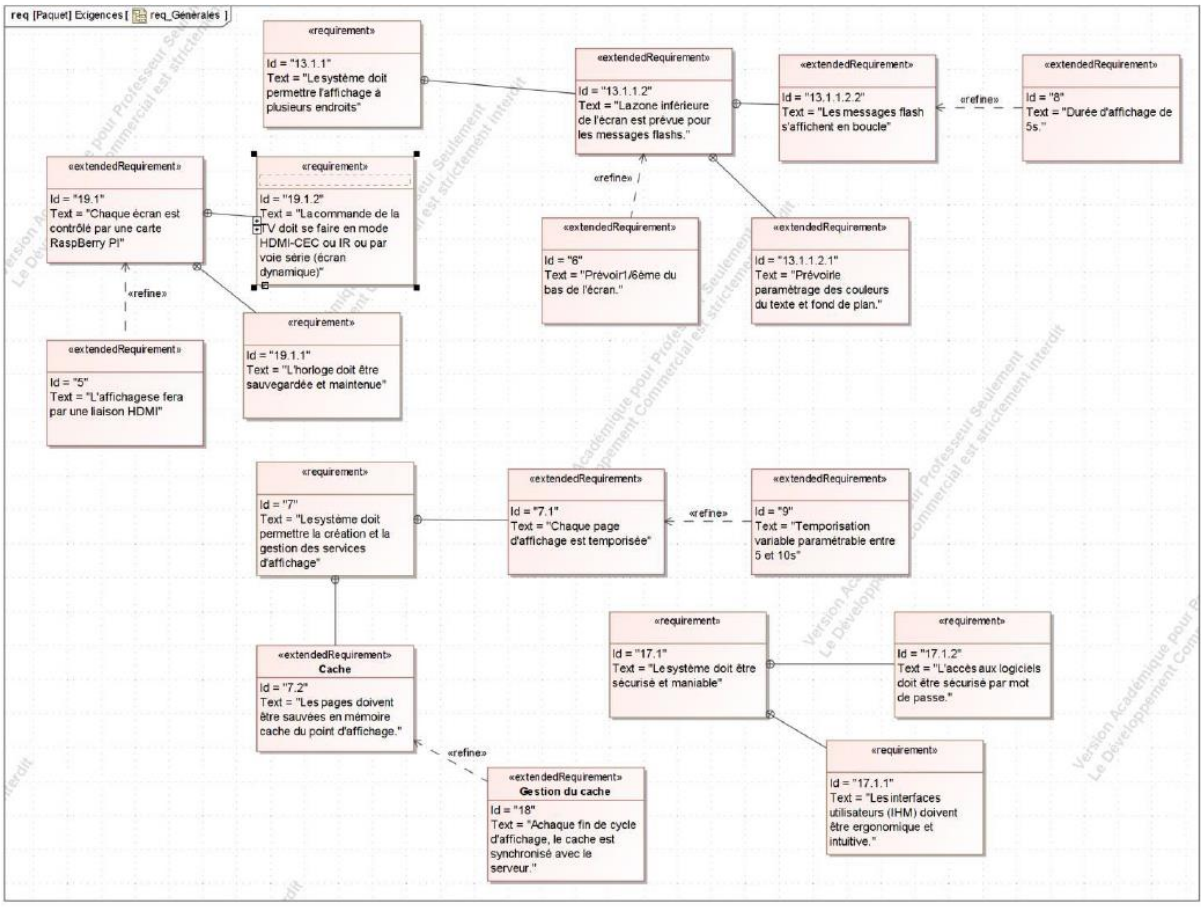
3.1 Diagramme de cas d'utilisation



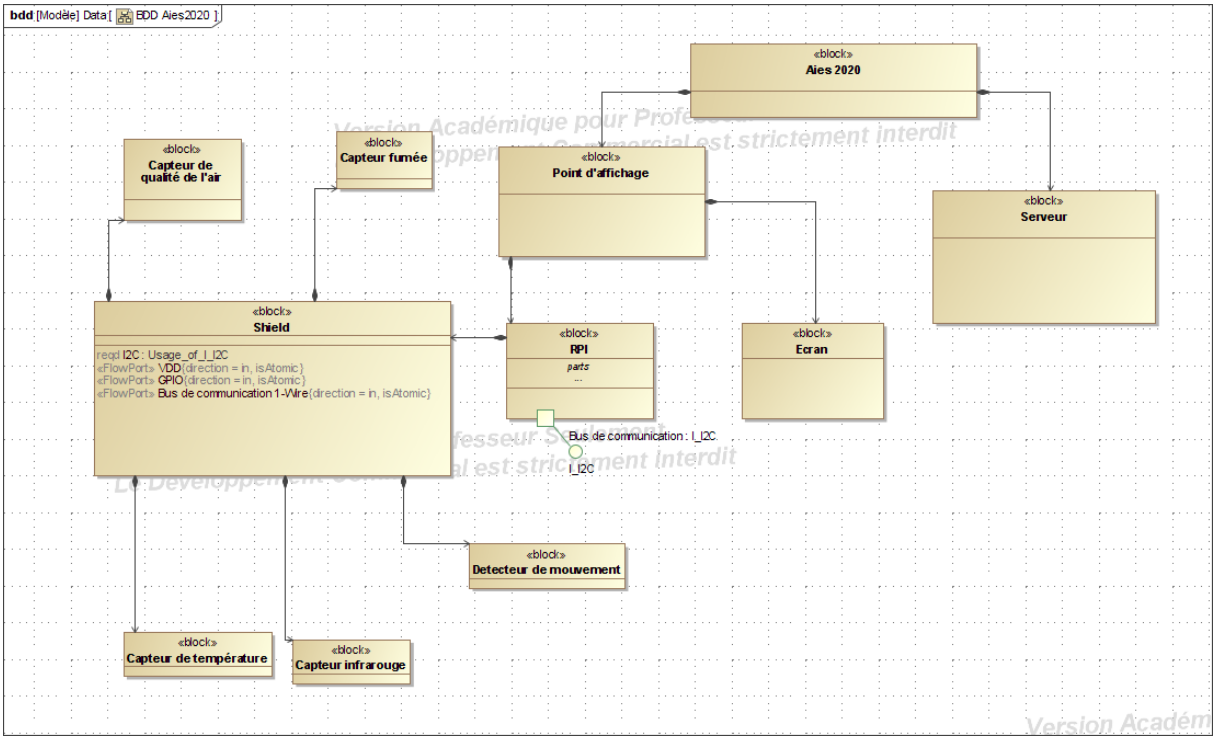
3.2 Diagramme de déploiement



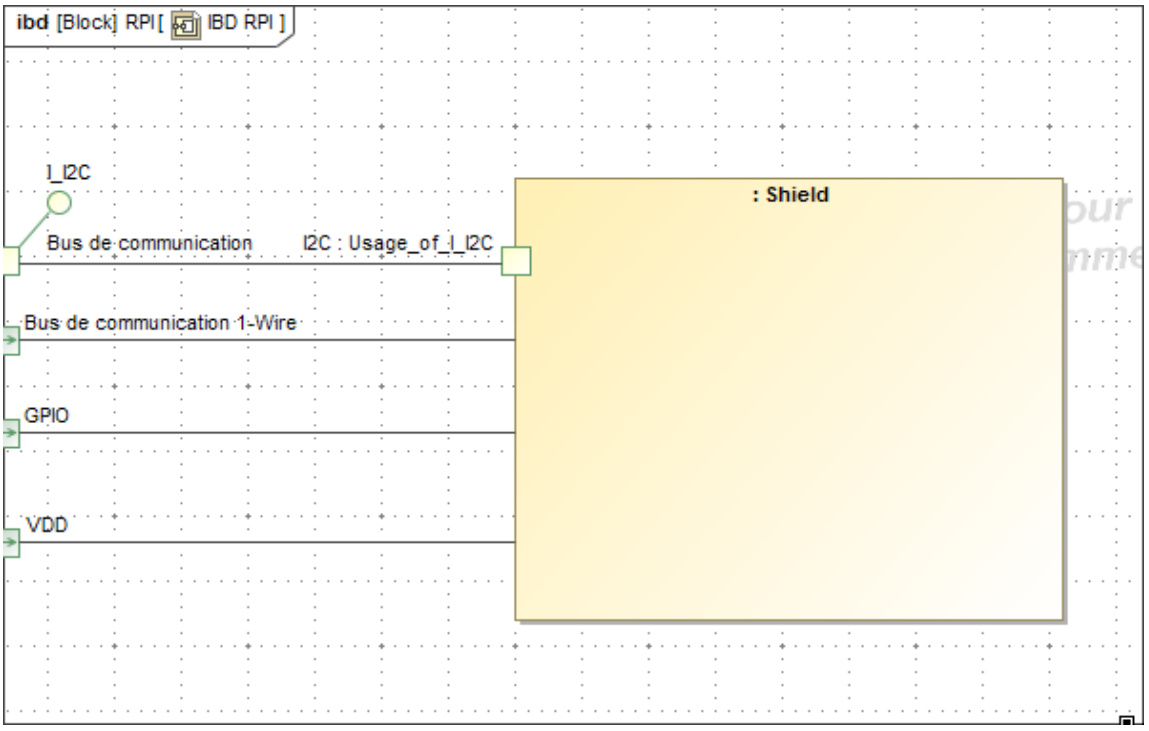
3.3 Diagramme d'exigences



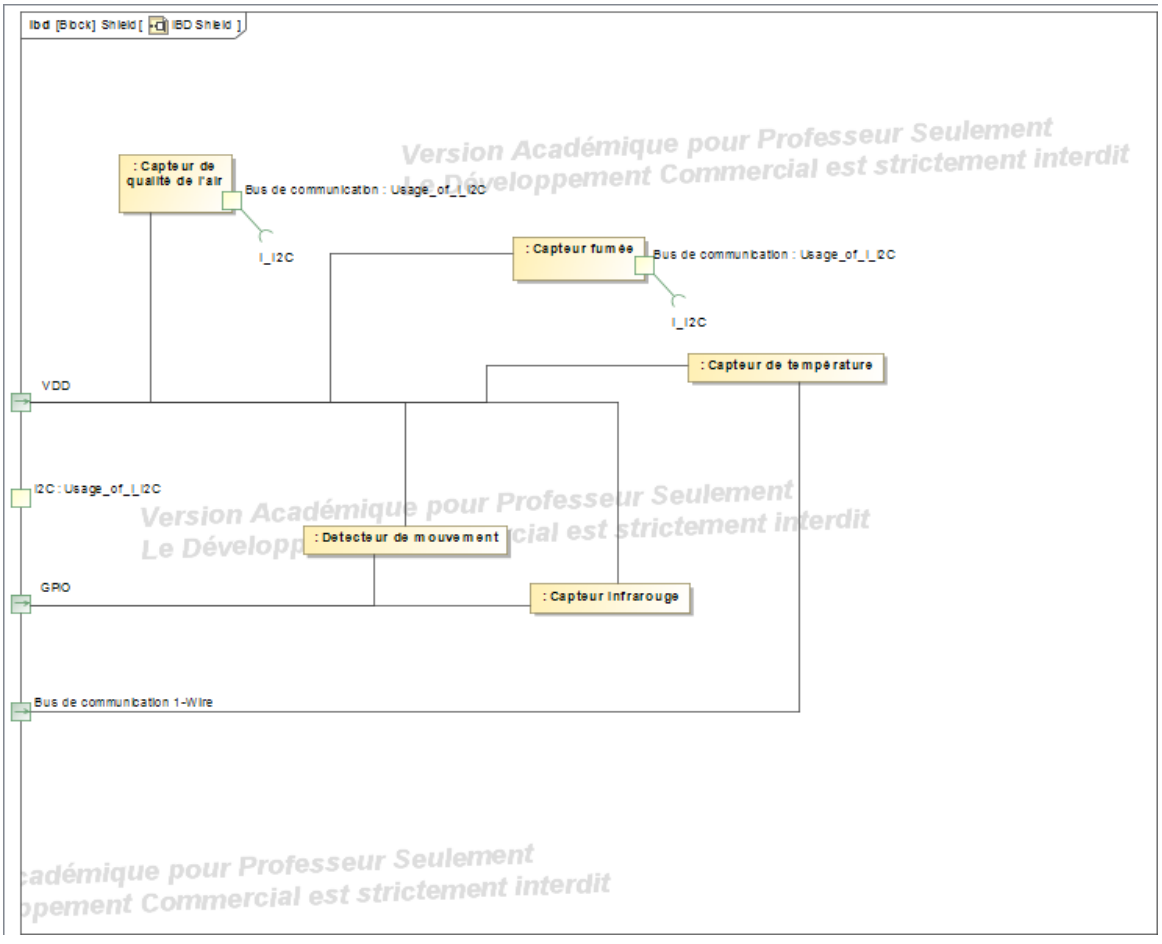
3.4 BDD



3.5 IBD RPI



3.6 IBD SHIELD



4 Répartition des tâches

4.1 Ir 1

	Fonctions à développer et tâches à effectuer	
Étudiant 1 (Serveur)	*Liste des fonctions assurées par l'étudiant* Cas d'utilisation côté serveur (voir UCD). **Tester/comprendre/corriger** la partie logicielle existante. **Amélioration** : Prise en compte dans la BDD des capteurs de fumée et qualité de l'air. Mise en mémoire cache (SD) des informations à afficher. Intégration du format vidéo (WEB). Création des masques de saisie pour la vidéo. Uniformisation du code PHP/CSS.	Installation : - *Serveur Linux virtualisé* - *Services WEB sur serveur* - *Services SGBD* Mise en œuvre : Configuration : - *Serveurs Linux, WEB, MySQL* Réalisation : - *BDD* - *Logiciel PHP client WEB* - *Script Shell de surveillance* Documentation : - *Logiciel client* - *Documentation logicielle*

4.2 Ir 2

Étudiant 2 (PA)	*Liste des fonctions assurées par l'étudiant* Cas d'utilisation côté PA (voir UCD). **Tester/comprendre/corriger** la partie logicielle existante. **Amélioration** : Surveillance du remplissage de la carte SD des PA. Mise en mémoire cache (SD) des informations (côté PA). Affichage des capteurs de fumée et qualité de l'air. Intégration d'informations vidéo (PA). Surveillance et relance du PA en cas de perte de connexion avec le serveur.	Installation : - *Services clients MySQL sur RPI* - *Qt et cross développement* Mise en œuvre : - *Script SHELL de surveillance* Configuration : - *Système RPI pour accès BDD* Réalisation : - *Logiciel C/C++ RPI (client MySQL)* Documentation : - *Configuration totale de la RPI* - *Documentation logicielle*

4.3 Ec 1

Étudiant 3 (PA) **EC 1**	*Liste des fonctions assurées par l'étudiant* **Mesure de la qualité de l'air** Tous les documents des versions antérieures sont à disposition de l'étudiant, qui doit en prendre connaissance de façon à être capable d'expliquer le rôle de chacune des structures, ainsi que l'évolution des différentes versions. Concevoir/Réaliser/Tester une nouvelle carte d'extension optimisée par rapport aux versions antérieures, et intégrant un capteur de qualité de l'air qui sera à choisir parmi plusieurs possibles. Développer une petite application en mode console, ou graphique, permettant à minima de communiquer avec le capteur retenu.	**Installation** : Mise en service *(initialisation/configuration)* d'un Raspberry Pi *(librairie BCM2835, Qt Creator, autre si nécessaire)*. **Mise en œuvre :** Tester les capteurs de qualité de l'air mis à votre disposition dans leur version breakout. Choisir le mieux adapté, en exposant clairement les critères ayant mené à ce choix *(performances, disponibilité, prix, encombrement, facilité de câblage, etc ...)* **Réalisation :** Prototypage rapide pour tester les capteurs. Conception une carte intégrant la structure optimisée *(mesure de température, détection de mouvement, horloge temps réel)* à laquelle s'ajouteront le capteur de qualité de l'air retenu, et le détecteur de fumée retenu par l'étudiant EC2. Le schéma structurel final devra être le même pour les étudiants EC1 et EC2. Prévoir la possibilité de réaliser un boîtier sur mesure pour le RPI associé à la carte d'extension. Le routage de la carte sera individuel, mais la disposition des connecteurs identique. Le format de la carte devra être compatible Rpi2/3/4. **Documentation :** Schéma de câblage rapide *(Fritzing)*. Documents de fabrication *(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. Programme en C/C++ d'acquisition de la qualité de l'air, accompagné des commentaires et diagrammes nécessaires à sa compréhension. Fiche de mise en service. Fiche de dépannage.

4.4 Ec 2

Étudiant 4 (PA) **EC 2**	*Liste des fonctions assurées par l'étudiant* **Détection de fumée** Tous les documents des versions antérieures sont à disposition de l'étudiant, qui doit en prendre connaissance de façon à être capable d'expliquer le rôle de chacune des structures, ainsi que l'évolution des différentes versions. Concevoir/Réaliser/Tester une nouvelle carte d'extension optimisée par rapport aux versions antérieures, et intégrant un détecteur de fumée qui sera à choisir parmi plusieurs possibles. Développer une petite application en mode console, ou graphique, permettant à minima de communiquer avec le détecteur retenu.	**Installation** : Mise en service *(initialisation/configuration)* d'un Raspberry Pi *(librairie BCM2835, Qt Creator, autre si nécessaire)*. **Mise en œuvre :** Tester les détecteurs de fumée mis à votre disposition. Choisir le mieux adapté, en exposant clairement les critères ayant mené à ce choix *(performances, disponibilité, prix, encombrement, facilité de câblage, etc ...)* **Réalisation :** Prototypage rapide pour tester les capteurs. Conception une carte intégrant la structure optimisée *(mesure de température, détection de mouvement, horloge temps réel)* à laquelle s'ajouteront le détecteur de fumée retenu, et le capteur de qualité de l'air retenu par l'étudiant EC1. Le schéma structurel final devra être le même pour les étudiants EC1 et EC2. Prévoir la possibilité de réaliser un boîtier sur mesure pour le RPI associé à la carte d'extension. Le routage de la carte sera individuel, mais la disposition des connecteurs identique. Le format de la carte devra être compatible Rpi2/3/4. **Documentation :** Schéma de câblage rapide *(Fritzing)*. Documents de fabrication *(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. Programme en C/C++ de détection de fumée, accompagné des commentaires et diagrammes nécessaires à sa compréhension. Fiche de mise en service. Fiche de dépannage.

4.5 Commun

Tous les étudiants	*Liste des fonctions assurées par les étudiants* Contenu à étudier en sciences physiques : (Références extraites du référentiel) • 2 Les ondes 2.4 Lignes de transmission 2.5 Fibre optique • 5 Colorimétrie et images numériques • 6 Oscillateurs 6.1 Production de signaux (EC)	

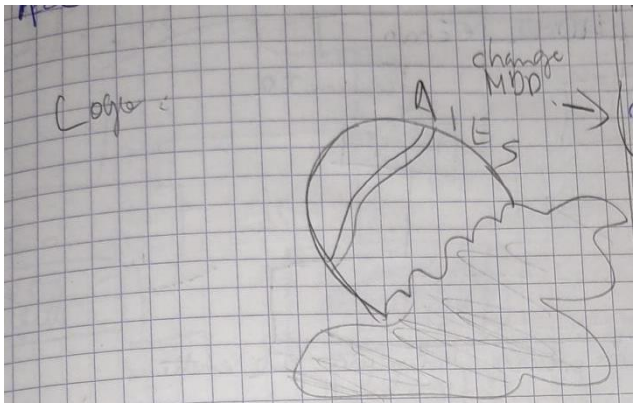
5 Réunions

Fiche de rapport de réunion

Projet	*Auteur*	*Date*
AIES	Groupe AIES	07/01/2020
Produit	***Durée de la réunion***	
Logo	1h	

Déroulement

Réflexion et questionnement sur la création du logo et son rapport avec notre projet (Couleur bleu pour représenter la pureté de l'air et la fumée pour représenter le détecteur de fumée).



Fiche de rapport de réunion

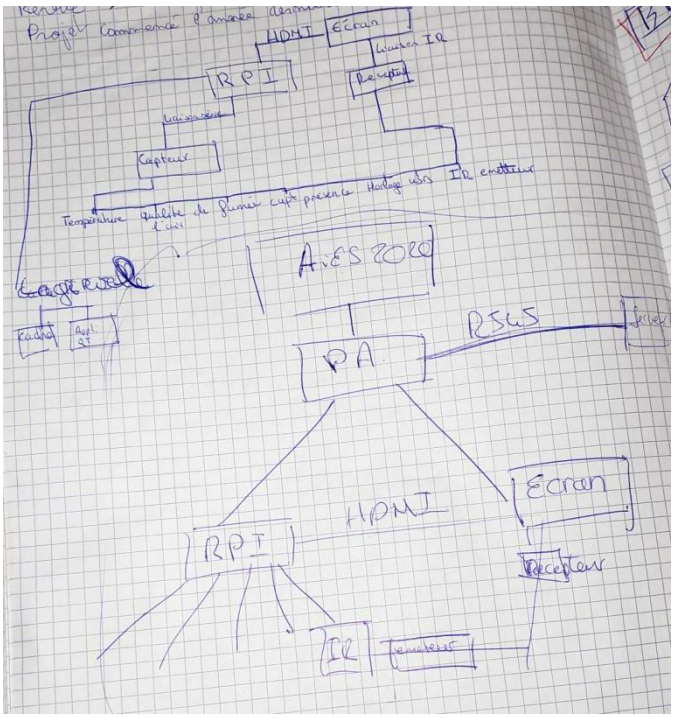
Projet	*Auteur*	*Date*
AIES	Groupe AIES	13/01 /2020

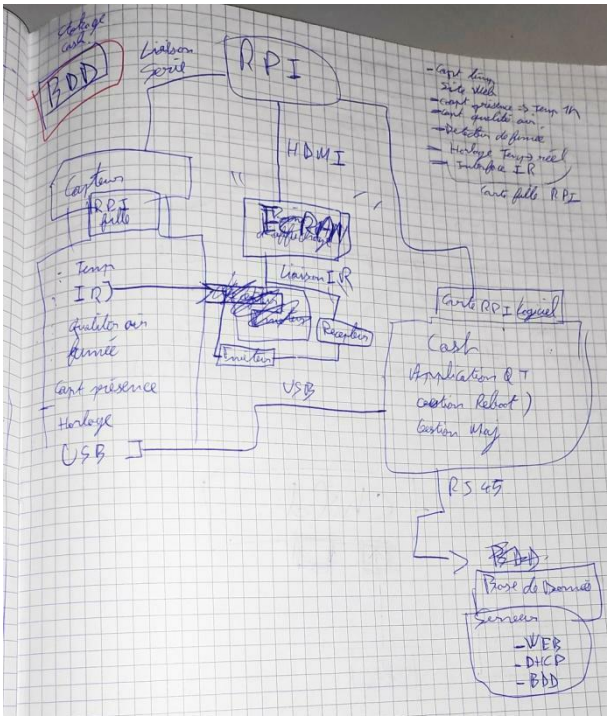
Produit	*Durée de la réunion*
BDD et IBD	2h

Déroulement

Réflexion du groupe pour la création du BDD et de l'IBD.

Recherche écrite :



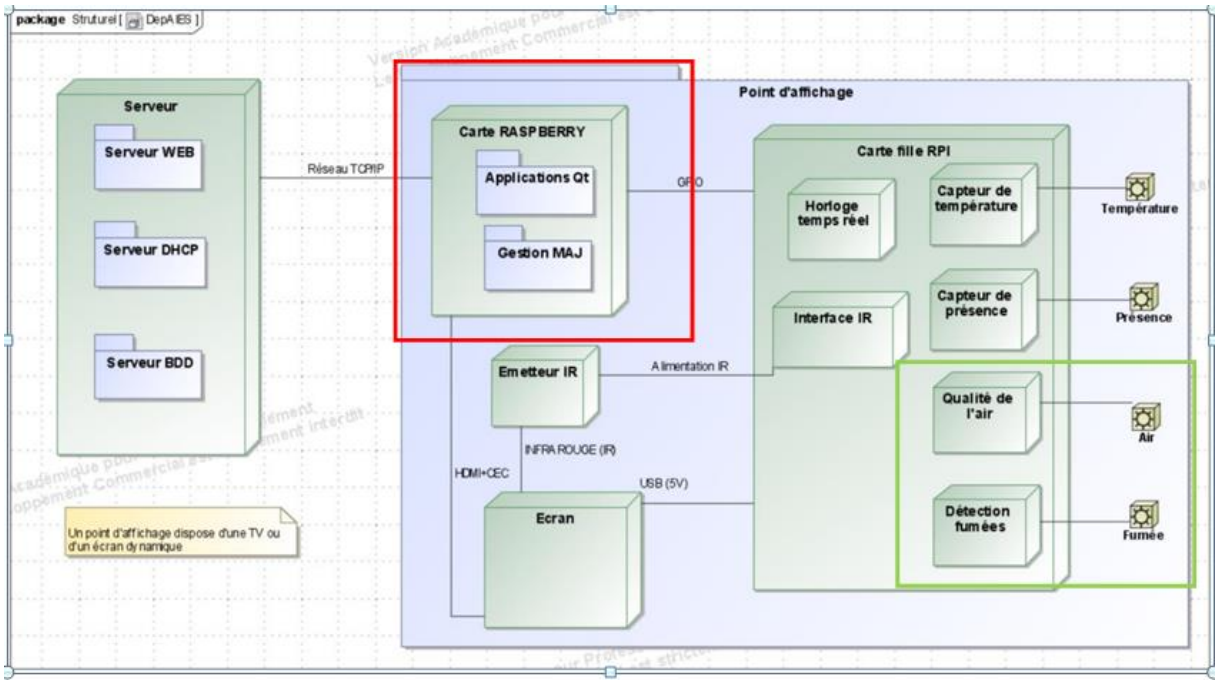


6 Partie Personnelle Killian Grosset

6.1 Présentation du projet orienté vers ma partie

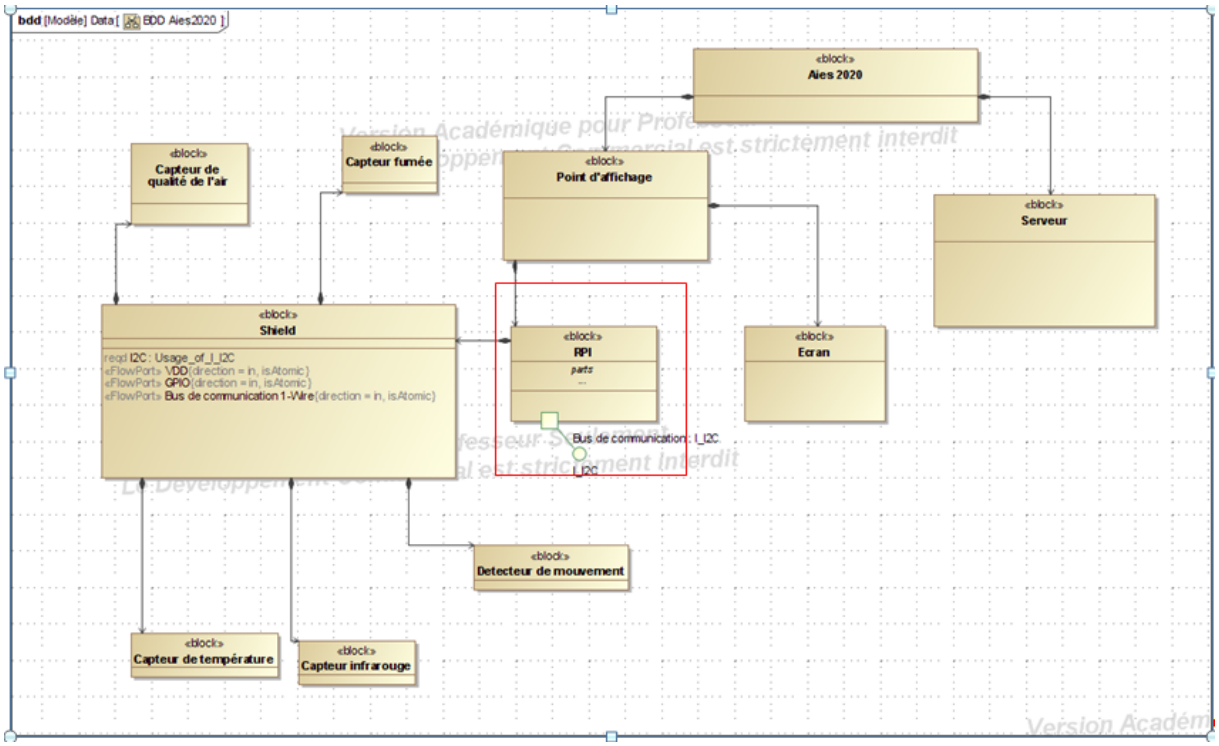
Le projet AIES est un projet qui nous ai demandé par le Lycée Alphonse Benoît. C'est un projet qui a pour but d'informer les lycéens, les professeurs et le personnel des différents événements qui ont lieu dans l'enceinte du lycée. Sept point d'affichage sont déjà installé dans le lycée est une huitième va être ajouté par notre groupe pendant le cours de notre projet. Ma partie dans ce projet consiste à mettre à jour le programme qui a été faite les années précédentes. Je dois pour cela m'occupé du démarrage des Raspberrys pour évier qu'elle ce connecte en cas de problème avec le serveur et de redémarrer le programme les Raspberrys perdent le signal. Une des autres partie de mon projet et de crée un système afin de vérifier l'espace mémoire de chaque Carte afin de savoir la place restante sur celle-ci et de vidé la mémoire en cas de sur-stockage. Et enfin de crée une mémoire cache qui permettrais aux Cartes de ne plus avoir à questionner le serveur entre chaque diapositive à afficher ce qui permettrais de fluidifier le réseau et limiter les problèmes en cas de perte de connexion.

6.2 Diagramme de déploiement



La partie entouré en rouge est la partie sur laquelle je vais travailler au long de mon projet
La partie entouré en vert est c'elle qui est faite par mes camarades d'EC que je devrais implémenter au programme à la fin de leurs parties

6.3 Ma Partie sur le BDD



6.4 Cahier des charges

Étudiant 2 (PA) **IR 2**	*Liste des fonctions assurées par l'étudiant* Cas d'utilisation côté PA (voir UCD). **Tester/comprendre/corriger** la partie logicielle existante. **Amélioration** : Surveillance du remplissage de la carte SD des PA. Mise en mémoire cache (SD) des informations (côté PA). Affichage des capteurs de fumée et qualité de l'air. Intégration d'informations vidéo (PA). Surveillance et relance du PA en cas de perte de connexion avec le serveur.	Installation : - *Services clients MySQL sur RPI* - *Qt et cross développement* Mise en œuvre : - *Script SHELL de surveillance* Configuration : - *Système RPI pour accès BDD* Réalisation : - *Logiciel C/C++ RPI (client MySQL)* Documentation : - *Configuration totale de la RPI* - *Documentation logicielle*

Les contraintes rencontrées par rapport à ce cahier des charges sont dans un premier temps la compréhension du programme déjà existant sur lequel j'ai travaillé pendant 1 semaine est sur lequel j'ai ensuite crée un diagramme de classe avec le logiciel MagicDraw. L'une des autres contraintes à laquelle j'ai du faire face est le fait de devoir gêner mon camarade qui s'occupe de la machine virtuel afin de pourvoir faire des tests sur le serveur. La dernière contrainte à laquelle j'ai du faire face est le développement sur Raspberry Pi.

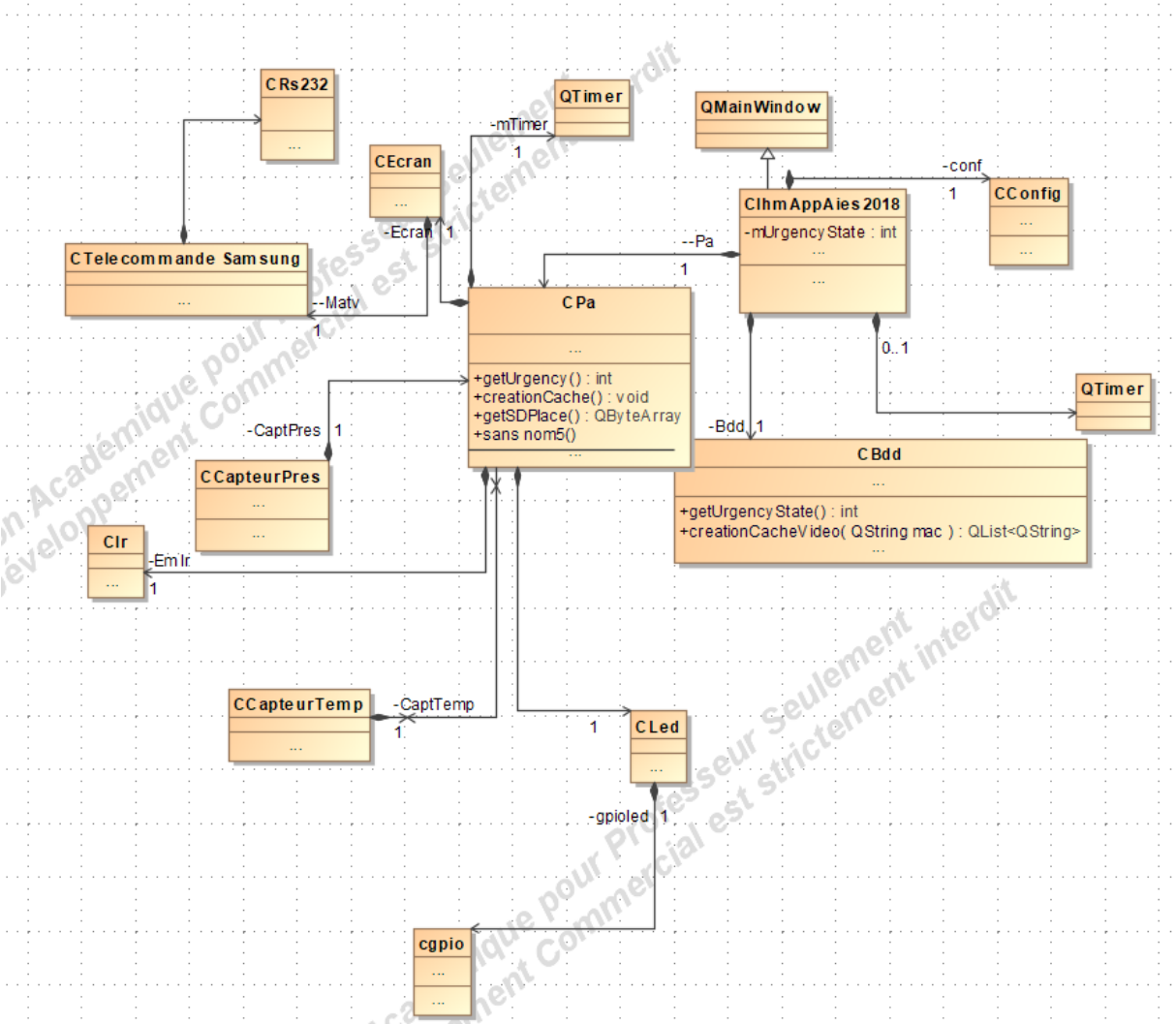
6.5 Planification

Nom de la tâche	Durée	Début	Fin
Projet AIES	225 h?	Lun 06/01/20	Jeu 14/05/20
Prise en main du sujet	4 h	Lun 06/01/20	Lun 06/01/20
Analyse du cahier des charges ainsi que les diagrammes UML/SYSML	8 h	Mar 07/01/20	Jeu 09/01/20
Analyse du système déjà existant	15 h	Jeu 09/01/20	Jeu 16/01/20
Création des diagrammes SYSML	20 h	Jeu 16/01/20	Lun 27/01/20
Programmation de la fonctionnalité de reboot	30 h	Lun 27/01/20	Lun 10/02/20
Rédaction du compte rendu	3 h	Lun 10/02/20	Mar 11/02/20
programmer la vérification du stockage de la carte SD	50 h	Mar 11/02/20	Mer 18/03/20
programmation de la fonctionnalité cache	90 h	Mer 18/03/20	Mer 13/05/20
Rédaction du compte rendu	3 h	Mer 13/05/20	Jeu 14/05/20
Revue 1	0 h	Lun 10/02/20	Lun 10/02/20
Revue 2	0 h	Lun 06/04/20	Lun 06/04/20

6.6 Travaux effectués

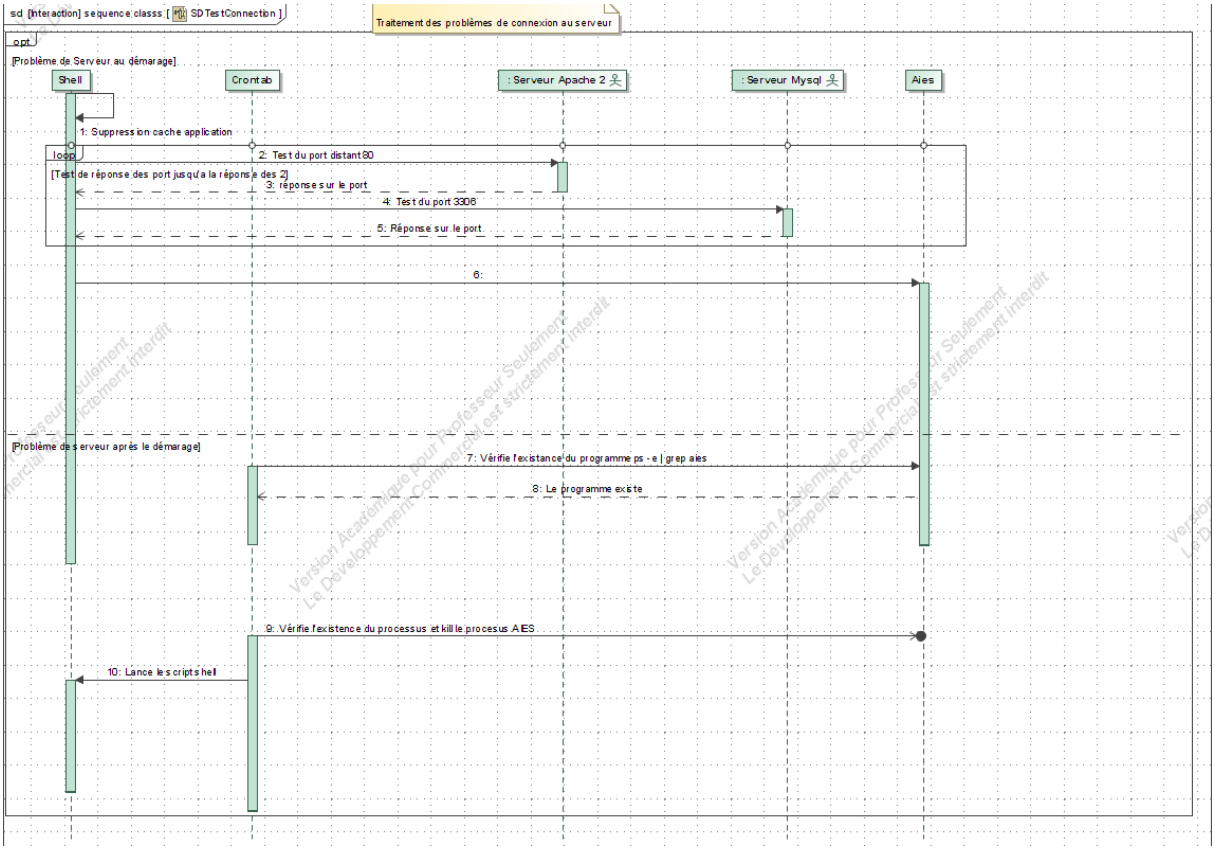
6.6.1 Les diagrammes

6.6.1.1 Diagramme de classe 2020

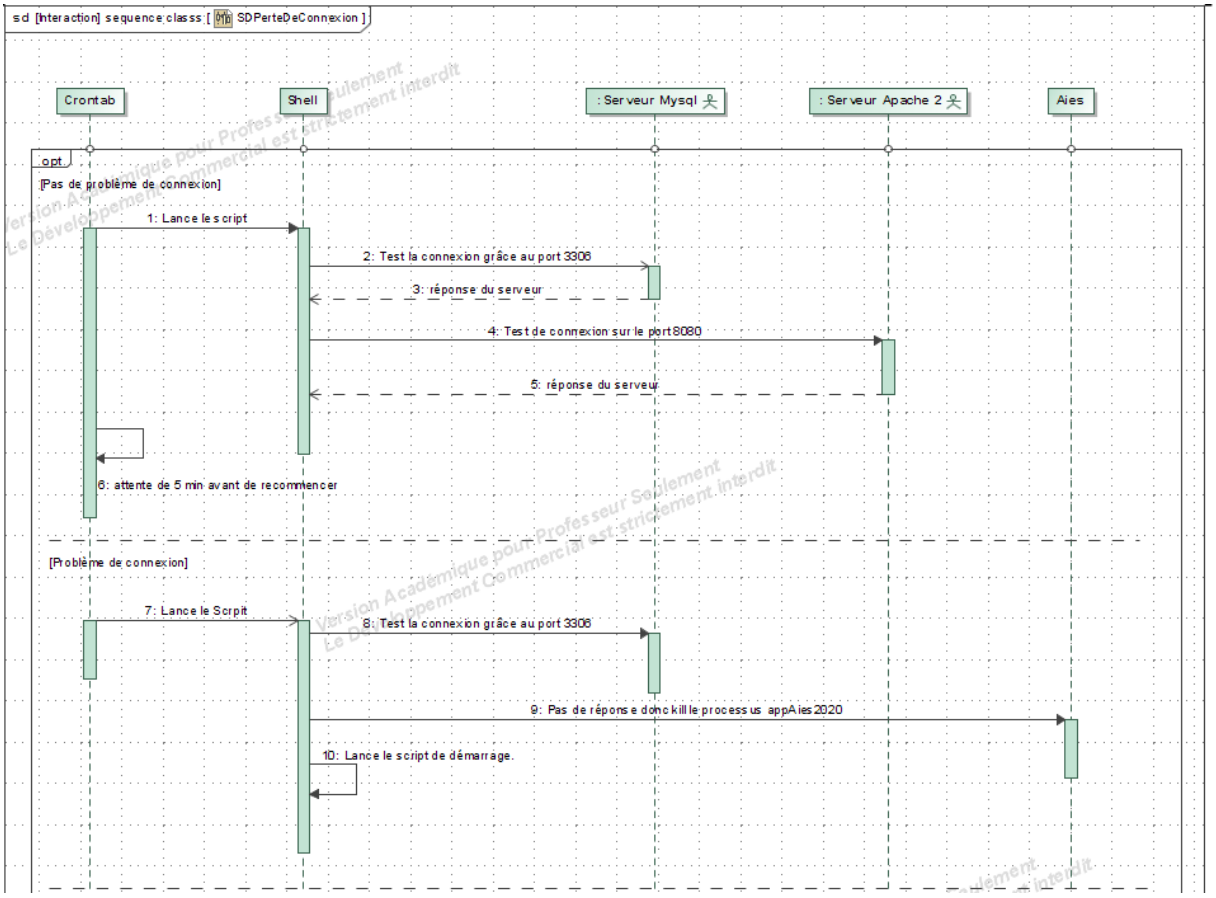


J'ai fais apparaitre sur ce diagramme seulement les nouvelles méthodes que j'ai utilisé pour le projet.

6.6.1.2 Diagramme de Séquence Test de connexion



6.6.1.3 Diagramme de Séquence perte de connexion



6.6.2 Programmation de la fonctionnalité de boot et de reboot du programme

6.6.2.1 Script de démarrage du programme

```
ip=$(cat /home/pi/aies/aies2020/aies.ini|grep ip|tr -d 'i'|tr -d 'p'|tr -d '=') #Recupère l ip du serveur
echo $ip

while [ "$resultat" != 0 ] #Boucle d'attente de la disponibilité des serveurs
do
        nc -vvv -w 10 -z $ip 80 #Test la disponibilité du port 80
        resultat=$?

        if [ "$resultat" != 0 ] #Si la commande échoue attente de 5 minute
        then
                sleep 5m

        else #Réussite de la connexion

                nc -vvv -w 10 -z $ip 3306 #Test la disponibilité du port 3306
                resultat=$?
        fi

        if [ "$resultat" != 0 ] #Si la commande échoue attente de 5 minute
        then
                sleep 5m

        fi #Réussite de connexion au deux serveur donc on lance l application

done

sudo systemctl restart cron.service
cd /home/pi/aies/aies2020
./appAies2020
```

6.6.2.2 Script en cas de perte du serveur

```
ip=$(cat /home/pi/aies/aies2018/aies.ini|grep ip|tr -d 'i'|tr -d 'p'|tr -d '=') #Récupère l'adresse ip remplit dans le dossier aies.ini
echo $ip #


        nc -vvv -w 10 -z $ip 80
        resultat=$?

        if [ "$resultat" != 0 ]
        then
                sudo killall appAies2019 #Si la connexion échoue on ferme l application
                sleep 5m
                sudo reboot #On redémarre la RPI

        else
                nc -vvv -w 10 -z $ip 3306
                resultat=$?
        fi
        if [ "$resultat" != 0 ]
        then
                sudo killall appAies2019 #Si la connexion échoue on ferme l application
                sleep 5m
                sudo reboot #On redémarre la RPI
        fi
```

6.6.2.3 Script en cas de maintenance

```
sudo systemctl stop cron.service #arrêt du service Cron
```

6.7 Application Qt

6.8 Stockage carte SD

Pour commencer je vais montrer comment j'ai fais pour le stockage de la carte SD.

```
QByteArray CPa::getSDPlace()
{
 QProcess sh; //on crée ici sh qui est le processus qui va nous permettre de faire une commande en shell
 sh.start("sh", QStringList() << "-c" << "df / |tr -s ' ' |cut -d ' ' -f5 |head -n 2 |tail -n 1 |tr -d '%'\l");
 // nous fesons ci-dessus la commande qui nous permet de récuperer seulement le pourcentage de la place restante dans la SD
 sh.waitForFinished(); //nous attendons le retour de la commande
 QByteArray output = sh.readAll();
 sh.close();//kill le processus
 qDebug() << output;
 return output;
}
```

Cette portion de code nous permet juste de faire la commande shell dans notre application QT dans le but de renvoyer ces informations à la base de données gérée par Tom.

```
QByteArray pourcentage = pa->getSDPlace();
```

Ici je récupère le pourcentage dans le main () pour l'envoyer à la BDD grâce à un QTimer

```
bdd->setCapteurs("UPDATE pas SET temp='"+realtemp+"', Pourcentage_SD ='"+pourcentage+"',
```

Et enfin grâce à cette ligne j'envoie les données dans la BDD

6.9 Programmation de la fonctionnalité d'arrêt des points d'affichages

Pour commencer je récupère la valeur de mon attribue Urgency dans la base de données

```
int CBdd::getUrgencyState()
{
     if(!Aies_bdd.isOpen())
    {

        exit(1);
    }

     else
     {
         int urgencyIsOn = 0;
         QSqlQuery Aies_query("SELECT isOn FROM urgency"); //prend la valeur dans la BDD
         Aies_query.next();
         urgencyIsOn = Aies_query.value(0).toBool(); //transforme la valeur en bool
 //      qDebug() << "Urgency: " << urgencyIsOn;
         return urgencyIsOn;
     }
}
```

Cette partie de code me permet seulement d'avoir accès à mon attribue Urgency dans la classe principale de mon application

```
int CPa::getUrgency()
{
     int mUrgency = mBdd->getUrgencyState();
     return mUrgency;
}
```

Ensuite je fais un test pour savoir à quoi correspond la valeur contenue dans Urgency. Si la valeur est de 1 j'arrête les points d'affichage et je diffuse une diapositive pour dire qu'aucun contenue n'est actuellement disponible. Puis je réessaye toutes les 10 secondes.

```
    mUrgencyState = pa->getUrgency();
    qDebug() << "urgence"<<  mUrgencyState << endl;

if(mUrgencyState != 0)
{
        QUrl url("http://" + mServeur + docs + "/rpi/slide/oups.php");
        ui->webViewStarter->load(url);
        timerIdleSlide->start(10000);

}
```

6.10 Création du cache pour les vidéos

```
QList<QString> CBdd::creationCacheVideo(QString mac){

bool newData = false;
QString req = "SELECT pathimg FROM slidezone WHERE state = 'actif' AND (zone=0 OR zone="+getNoZone(mac)+") AND pathimg IS NOT NULL";
QSqlQuery Aies_query(req); // envoie de la requette a la BDD
QList<QString> videoSlide;

qDebug() << "video " << req << " " << Aies_query.size();
while(Aies_query.next()) // la requette nous renvoi un resultat
{
    newData = true;
    QString tempQList;
    tempQList.append(Aies_query.value(0).toString()); //on met les diapositive dans une liste pour les traiter

    videoSlide.append(tempQList);
}
if(newData == false) //aucune diapositive ne correspond a la requette
{
    QString tempQList;
    tempQList.append("none");
    videoSlide.append(tempQList);
}
 return videoSlide;
}
```

Ici je récupère toutes les diapositives contenant une vidéo pour pouvoir les traiter autre part.

```
id CPa::creationCache()

int i =0;
QList<QString> videoSlide = mBdd->creationCacheVideo(mMac);
//QString command("scp root@192.168.1.39:"+path+" /home/pi/cache/");
QProcess scp;

for(i =0;i < videoSlide.size();i++)
{
  scp.startDetached("sh", QStringList() << "-c" << "scp root@192.168.1.39:/srv/www/htdocs/2020"+videoSlide.at(i)+" /home/pi/cache/");
  //ici nous utilison la commande scp pout récupérer les vidéos stocker sur le serveur selons leurs chemin d accès
  qDebug() << "Je suis passé" << i;
  scp.close();
}
```

C'est grâce à cette partie de code que je télécharge les vidéos du serveur vers la RPI. Le seul problème que j'ai avec cette partie de code est la commande SCP. En temps normal pour utiliser la commande SCP il nous faut renseigner les identifiants et mot de passe de la machines entrantes et de la machines host. Pour contourner ceci j'ai du crée des clés que j'ai renseigné sur le serveur et sur la RPI qui me permet ici d'effectuer une connexion sans avoir à renseigner les informations de connexion.

```
sudo rm -R /home/pi/cache #Supression du fichier nommé cache
sudo mkdir /home/pi/cache #Création d'un fichier nommé cache
```

6.11 Récapitulatif des recherches sur les codecs

Au cours du projet nous avons eu à utiliser des formats vidéo dans d'avoir un nouveau moyen pour afficher nos information.

C'est pour cela qu'avec mon camarade d'IR nous avons étudié certain formats Vidéos.

J'ai fais ici un tableau récapitulatif de certain format dont 2 que j'ai utilisé sur la RPI qui sont le MP4 et le MOV.

Points	MP4	MP3	MPEG4	MOV
Media Supporté	Vidéo, Audio et Textes	Audio	Vidéo, audio, et textes	Vidéo, audio, et textes
Compression	Avec et sans Perte	Sans perte	Avec et sans Perte	Avec et sans Perte
Open Source	Oui	Oui	Oui	No
Compatibilité	Universel	Universel	Universel	Limité
Application	Stream de Vidéo, Compression de Vidéo, Jouer des vidéos avec sous-titres	Compression, stockage et transfert de fichier audio	Il s'agit d'un conteneur utilisé pour la compression et la gestion des fichiers audio/vidéo	Lecture de vidéos avec sous-titres ou compression de vidéos
Codec	H.264, H.265, AAC, MP3			Quick Time

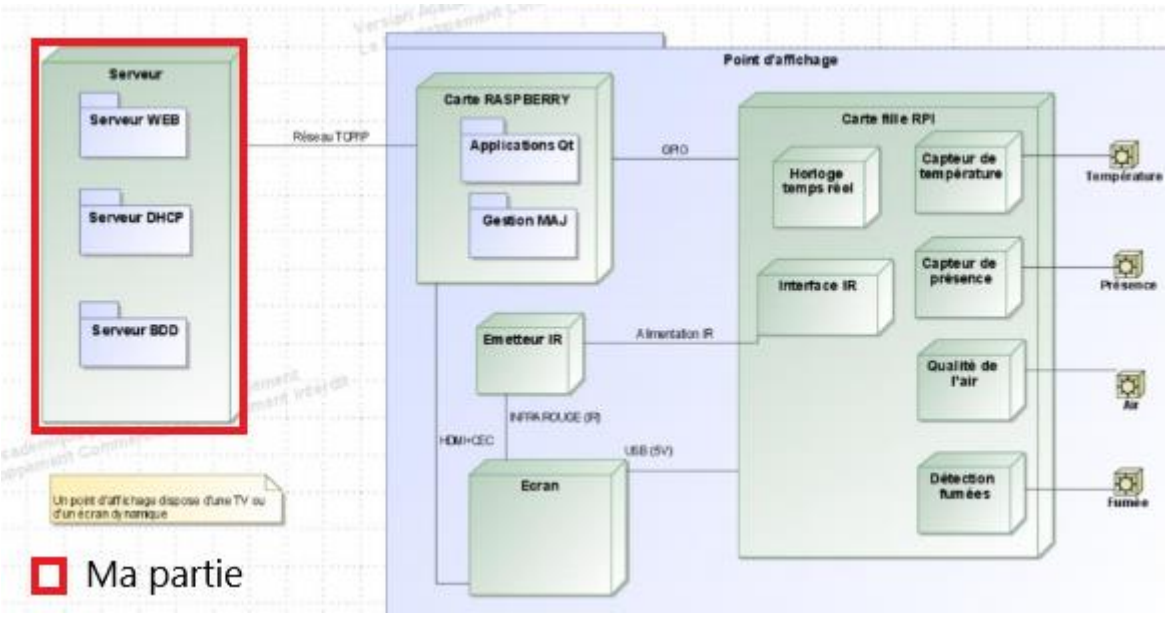
Après avoir effectué les testes nous nous somme rendus compte que le fichier MP4 étais le meilleur Format vidéo que l'on pouvait avoir car il utilise les mêmes codecs que détiens la RPI. De plus Après le testes de plusieurs format vidéo il est le seul à s'afficher de manière correcte dans l'application Qt du au faite qu'il peut êtres plus ou moins compressé et qu'il utilise les bons code

7 Introduction partie personnelle Tom Harduin

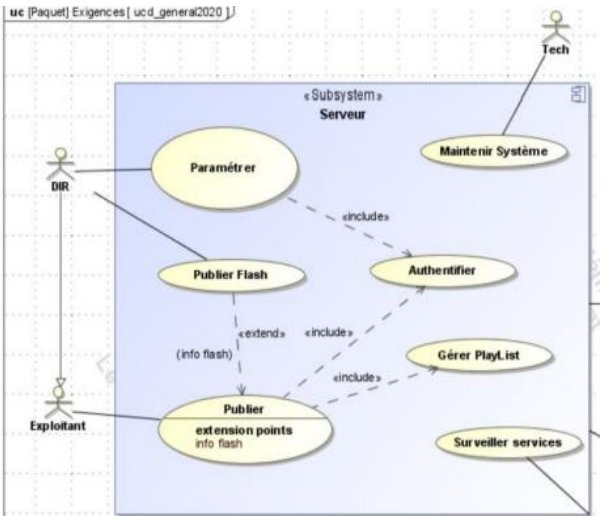
7.1 Introduction projet

Le projet AIES a commencé en 2016, il a été poursuivi en 2017 et 2018. On a donc pour mission de l'améliorer cette année. Le projet consiste à améliorer la partie déjà existante d'un système d'affichages dynamiques. Il existe déjà plusieurs points d'affichages dans le lycée. Cette année un nouveau point d'affichage sera ajouté ainsi qu'une mise à jour de l'application et du site web. Je m'occupe personnellement de la partie serveur (Base de données, web et DHCP).

7.2 Diagramme de déploiement



7.3 Diagramme de cas d'utilisation (partie serveur)



7.4 Tâches à effectuer

J'ai plusieurs objectifs à réaliser lors de ce projet. Les différentes tâches sont d'ajouter la possibilité du format vidéo pour les diaporamas. Intégration des capteurs de fumée et de qualité de l'air dans la base de données. De nouvelles tâches ont aussi été demandées, comme l'ajout d'un tableau pour modifier l'organisation des zones ainsi qu'un bouton qui permet l'arrêt de toutes les diapos en cours.

7.5 Cahier des charges

	Fonctions à développer et tâches à effectuer	
Étudiant 1 (Serveur) **IR 1** IR	*Liste des fonctions assurées par l'étudiant* Cas d'utilisation côté serveur (voir UCD). **Tester/comprendre/corriger** la partie logicielle existante. **Amélioration** : Prise en compte dans la BDD des capteurs de fumée et qualité de l'air. Mise en mémoire cache (SD) des informations à afficher. Intégration du format vidéo (WEB). Création des masques de saisie pour la vidéo. Uniformisation du code PHP/CSS.	Installation : - *Serveur Linux virtualisé* - *Services WEB sur serveur* - *Services SGBD* Mise en œuvre : Configuration : - *Serveurs Linux, WEB, MySQL* Réalisation : - *BDD* - *Logiciel PHP client WEB* - *Script Shell de surveillance* Documentation : - *Logiciel client* - *Documentation logicielle*

7.6 Planning prévisionnel

Evaluations du projet	**Lun 10/02/20**	**Jeu 18/06/20**			**209 h**
Revue 1	Lun 10/02/20	Lun 10/02/20			2 h
Revue 2	Lun 04/05/20	Lun 04/05/20	7		2 h
Remise du dossier	Mer 27/05/20	Mer 27/05/20	15		1 h
Soutenance finale	Lun 15/06/20	Lun 15/06/20			1 h
Vacances d'hiver	Lun 17/02/20	Lun 02/03/20	2;11		4,14 jrs
Vacances printemps	Lun 13/04/20	Dim 26/04/20	13		3,43 jrs
Projet AIES	**Lun 06/01/20**	**Jeu 18/06/20**			**259 h**
Prise en main du projet	Lun 06/01/20	Lun 13/01/20			19 h
Analyse du code existant	Mar 14/01/20	Lun 27/01/20	9		20 h
Début de la page de vidéo	Mar 28/01/20	Jeu 13/02/20	10		27 h
Intégration format vidéo (web)	Mar 03/03/20	Lun 06/04/20	6		50 h
Ajout des capteurs de fumée et qualité de l'air (BDD)	Mar 07/04/20	Lun 13/04/20	12		10 h
Uniformisation code php/css	Lun 27/04/20	Jeu 14/05/20	7		30 h
Rédaction du dossier	Jeu 09/01/20	Mer 27/05/20			29,57 jrs
BTS blanc	**Lun 10/02/20**	**Mar 07/04/20**			**86 h**
1er BTS Blanc	Lun 10/02/20	Lun 10/02/20			6 h
2nd BTS Blanc	Lun 06/04/20	Mar 07/04/20			6 h

7.7 Travail accompli

Avant de m'attaquer aux différentes tâches qui m'ont été confiées, j'ai d'abord analysé le code du site WEB ainsi que la base données. j'ai aussi tester le site WEB (création de diapositives etc..)

7.8 Problème d'affichage des photographies

Lors de mes tests sur le site existant, je me suis rendu compte que les photos des diaporamas ne voulait pas s'insérer alors que c'était le cas dans la version précédente. J'ai donc cherché dans le code d'où pouvait venir le problème, en allant dans le dossier où était proposé les différents thèmes de diapos puis dans le dossier des diapos d'image au milieu du code j'ai trouvé cette ligne :

```
$docRoot = '/2019/rpi/slide/images/';
```

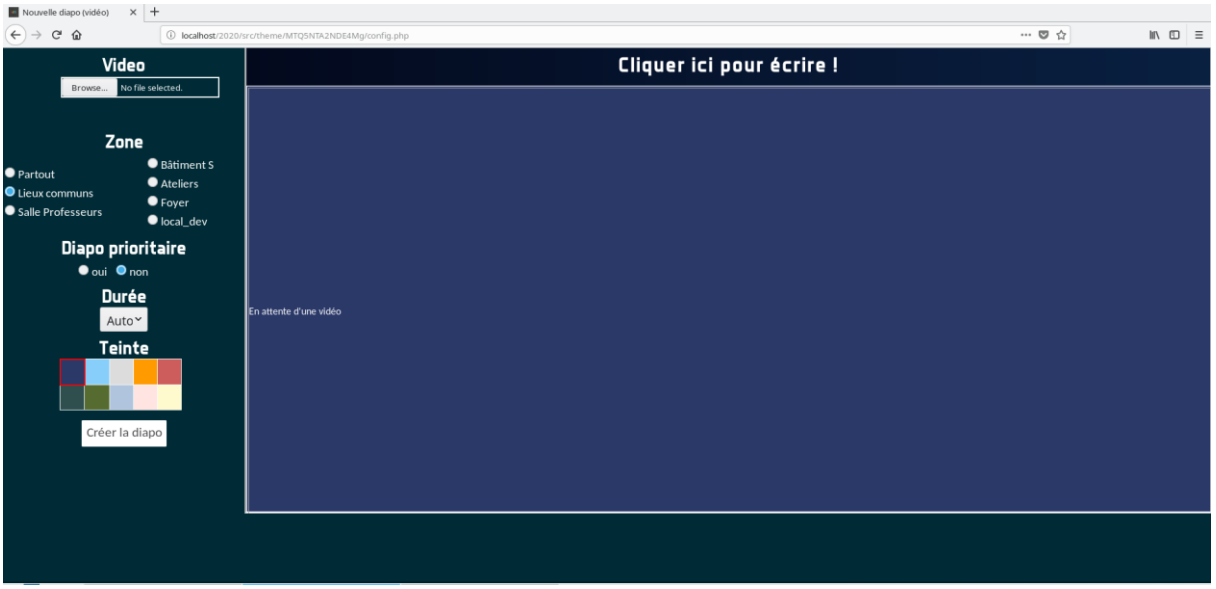
Le chemin d'accès n'était donc plus le même qu'avant car le dossier avait été renommé "2020" il a suffit de modifier cette partie pour que cela refonctionne correctement.

7.9

```
$docRoot = '/2020/rpi/slide/images/';
```

7.10 Début de la page des créations de diaporamas vidéo

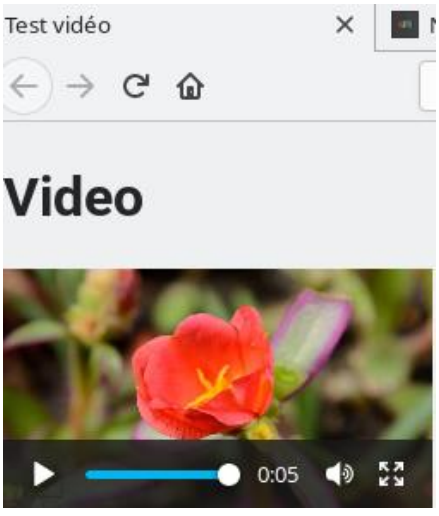
Création de la page pour créer les diaporamas “vidéos” en se basant sur les modèles existant (plus précisément celui des photos).



7.11 Début de test pour mettre des vidéos sur le site

```
<!DOCTYPE html>

<html>
    <head>
        <title>Test vidéo</title>

    </head>

    <body>

        <h1>Video</h1>

        <video controls width="250" id=video controls="controls">

            <source src="/srv/www/htdocs/2020/rpi/slide/videos/flower.webm" type="video/webm">

            <source src="/srv/www/htdocs/2020/rpi/slide/videos/flower.mp4" type="video/mp4">

            Le navigateur ne supporte pas le format vidéo.
        </video>

        <script>
            var vid = document.getElementById("video");

            if(window.addEventListener){//Utilise la fonction dès que la page charge
                window.addEventListener('load', enableAutoplay, false);
            }else{
                window.attachEvent('onload', enableAutoplay);
            }

                function enableAutoplay() { //permet de mettre la vidéo automatiquement sur play
                    vid.autoplay = true;
                    //vid.loop = true;
                    vid.load();
                }
        </script>

    </body>
</html>
```

Le fichier est externe au dossier du projet. Il permet pour l'instant de lancer la vidéo dès que la page web a fini de charger. L'objectif final serait que la vidéo se lance au moment où sa diapo arrivera. Ainsi que de l'insérer dans la page de création de nouvelle diapo.



Page web du code ci-dessus

7.12 Insertion vidéo

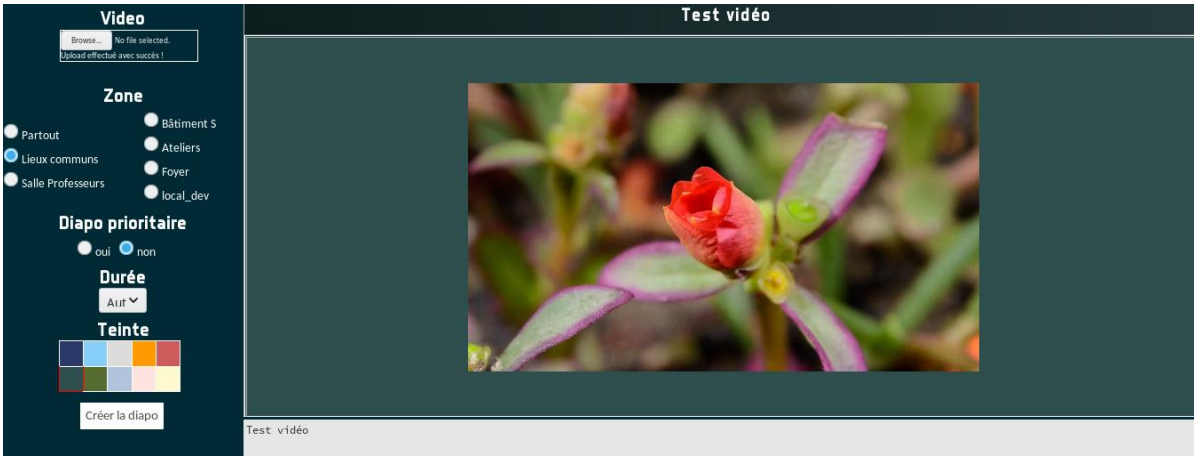
Depuis la première revue j'ai pu avancer sur la partie de l'insertion vidéo. Je peux maintenant insérer des vidéos depuis le site WEB. Pour cela j'ai procédé à une analyse :

J'ai été amené à choisir un format vidéo. J'ai donc décidé d'en parler pour ma partie physique car je trouve cela intéressant d'expliquer le choix d'un format plutôt qu'un autre.

Après quelques recherches j'ai rapidement choisi le MP4 et le WebM à approfondir :

	WebM	MP4
Année	2010	2004
CODEC	VP8, VP9, Vorbis, Opus	H.264, H.265, AAC, MP3
Qualité	bonne qualité	meilleure qualité
Taille des fichiers	petits fichiers	grands fichiers
Diffusion	promu principalement par Google	très courant
Brevets	open source	soumis à brevet

Même si sur le papier le format WebM paraît plus adapté pour notre utilisation que le format MP4. (car sur la RPI nous avons besoin des fichiers les plus petits possible) nous nous sommes rendu compte que finalement la RPI avait beaucoup de mal à lire les fichiers au format WebM et aucun problème lors du test avec le MP4. Les Codecs de la RPI devaient sûrement avoir du mal avec le WebM car c'est un format plus récent que le MP4.



7.13 Intégration capteurs BDD

Dans mon cahier des charges il m'a été demandé d'intégrer les capteurs dans la base de données. J'ai donc consulté mes camarades EC pour savoir comment fonctionnait leurs capteurs.

Pour le capteur de fumée, il renvoie une valeur booléenne (à 0 pas de fumée et à 1 le capteur détecte de la fumée, j'ai donc créé une colonne pour le détecteur de fumée).

Le capteur de qualité de l'air renvoie lui deux valeurs. Une valeur correspondant à un taux de CO2 et une deuxième qui correspond COV (Composé organique volatil) j'ai donc créé une colonne pour chacune de ses valeurs. Mon binôme IR peut maintenant m'envoyer les valeurs dans la BDD et je peux donc les réutiliser dans l'onglet "État système".

J'ai aussi ajouté la colonne "Pourcentage_SD" (dernière colonne) pour mieux gérer le cache.

La table ci-dessous représente les différents points d'affichages (nom de la table: "pas").

ID	name	ip	mac	version	zone_id	temp	CO2	COV	fumee	CtrlTv	ModeFonc	hStart	hStop	presence	idleTime	freqIr	Pourcentage_SD
1	PA_VIESCO	10.73.254.12	b8:27:eb:27:8c:8e	2.2	1	15.6	1000	0	0	cec	heure	07:00	19:00	N	15	0	0
2	PA_PROFS	10.73.254.10	b8:27:eb:aa:70:09	2.2	2	22.6	451	0	0	ir	heure	07:00	19:00	N	15	0	0
3	PA_C12	10.73.254.11	b8:27:eb:12:db:13	2.2	1	25.0	300	0	0	rs	presence	16:14	16:12	O	5	0	0
4	PA-FOYER	10.73.254.20	b8:27:eb:e9:17:d9	2.1	1	20.7	0	0	0	ir	heure	07:00	19:00	N	15	0	0
5	PA-SELF	10.73.254.13	b8:27:eb:93:da:8d	2.2	1	21.1	0	0	0	ir	heure	07:00	19:00	N	15	0	0
6	PA-S	10.73.254.14	b8:27:eb:c1:a6:2b	2.2	3	-100.0	0	0	0	ir	heure	16:00	15:55	N	15	0	0
7	PA_AtelierSII	10.73.254.15	b8:27:eb:c3:65:be	2.3	4	24.1	0	0	0	rs	heure	07:55	17:45	N	15	0	0
8	PA_AtelierPRO	10.73.254.16	b8:27:eb:87:ce:cd	2.3	4	24.1	0	0	0	ir	heure	16:55	16:53	N	60	15	41

7.14 Amélioration de l'onglet "État système"

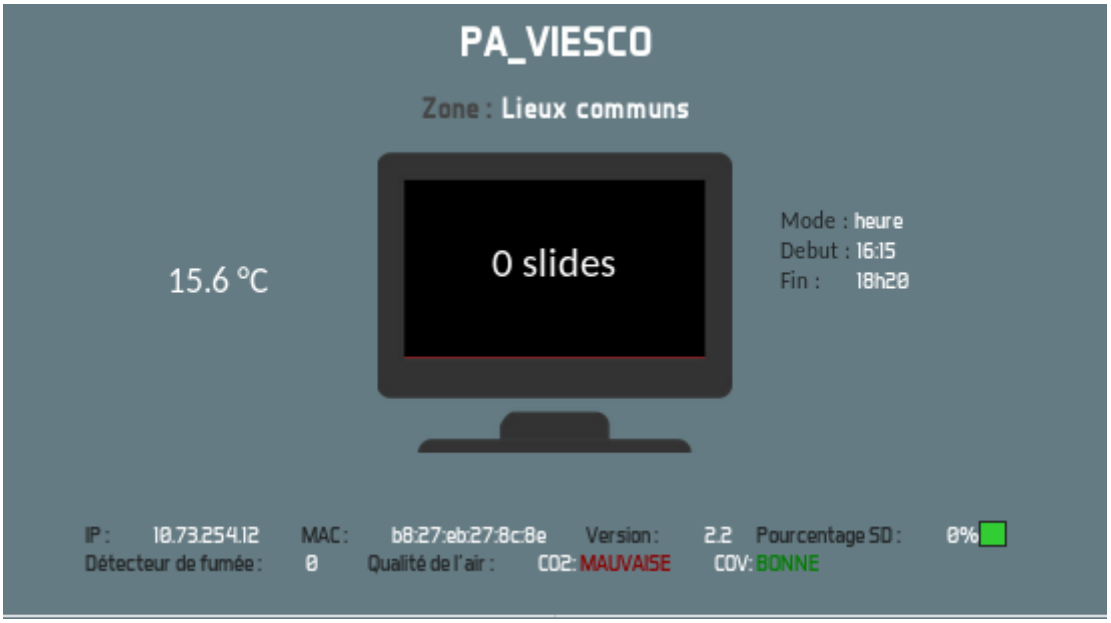
Sur le screen ci-dessous j'ai ajouté plusieurs nouveaux éléments:

-Le pourcentage de la carte SD. Mon camarade IR m'envoie le pourcentage de la carte SD de chaque point d'affichage. En fonction de la valeur j'ai décidé d'ajouter un code couleur, vert jusqu'à 60% inclus, orange de 61% à 89% inclus et rouge jusqu'à 100%. Cela a pour but de mieux gérer le cache de la carte SD.

-Le détecteur de fumée. La valeur passe à 1 en cas de détection de fumée.

-La qualité de l'air. Il y a trois états pour chaque valeur avec le même principe et code couleur que pour le pourcentage de la carte SD. Bonne en vert, Médiocre en orange et mauvaise en rouge.

J'ai aussi modifié après consultation du professeur la temporisation de rafraîchissement de la page elle était de 5 secondes, elle est maintenant de 20 secondes. Un rafraichissement aussi régulier n'était pas nécessaire de plus à chaque rafraichissement la page était remonté en haut ce qui peut être gênant pour la lecture des informations



7.15 Ajout du chemin vidéo

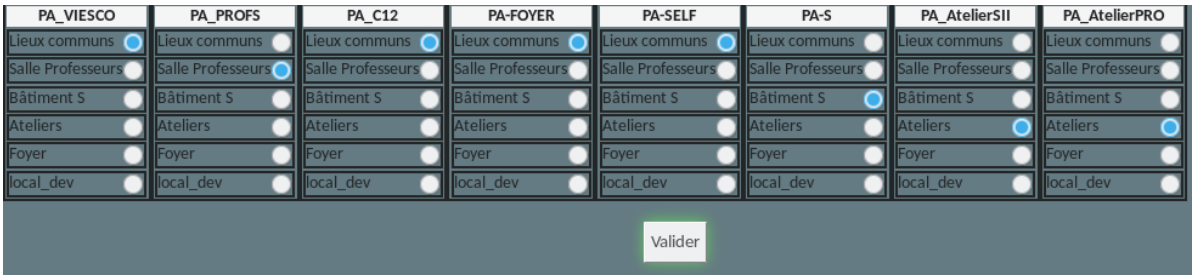
Dans la base de données nous avons décidé avec mon binôme IR de créer une colonne “pathimg” (le chemin d’accès aux images et aux vidéos) dans la table “slidezone” (qui gère les différentes diapos créé) avec le code ci-dessous:

```
$req = $db->query("INSERT INTO slidezone
(title, path, time, zone, priority, state, date_create, theme, creator, pathimg)
VALUES ('" . $_POST['title'] . "', '/rpi/slide/" . $newfile . "', " . $_POST['time'] . ",
" . $_POST['zone'] . ", '" . $_POST['priority'] .
"','arch', '" . $date_create . "', 6, '" . $_SESSION['common'] . "', '/rpi/slide/videos/" . $_SESSION['pathimg']. "') ");
```

J’ai ajouté ce qui est encadré en noir. J’insère donc dans le “pathimg” le chemin où est ma vidéo et à la suite ma variable superglobales récupère le nom du fichier. Cela permet à mon binôme de pouvoir récupérer les vidéos pour mieux gérer le cache.

7.16 Tableau organisation des zones

Sur demande en cours de projet, j’ai ajouté une fonctionnalité à l’onglet administrateur. Elle permet la modification de la zone de chaque point d’affichages.



Code qui permet l’affichage du tableau:

```
while ($data_name = $req_namepas->fetch())//créer une ligne avec le nom de chaque PA présent dans la BDD
{
    $req_zones = $db->query("SELECT zones, idzone FROM zones WHERE id != 0");// select la colonne de toutes zones sauf "partout"
    $nameValue = 0;
    $nbZone = 0;
    $checkZone = $checkZone+1;

    echo "<div class=tableau> <table>
            <form action='#' method='post'>
            <tr>
                <th>". $data_name['name'] ."</th>
            </tr>";

            while ($data_zone = $req_zones->fetch())//créer une ligne par zones existante dans chaque colonne (PA)
            {
                $nameValue = $nameValue+1;
                $nbZone = $nbZone+1;

                    //coche par défaut la zone actuel du PA
                    if($data_name['zone_id'] == $data_zone['idzone']){
                        $verifChecked = "checked";
                    }else{
                        $verifChecked = "";
                    }//else

               echo "<tr>
                        <td>
                            ". $data_zone['zones'] ."<input type='radio' class='radio' name='checkzone". $checkZone .
                            "'value='". $nameValue . "'$verifChecked/>
                        </td>
                    </tr>";
            }//while colonne
```

Requête SQL :

```
/**************PA_VIESCO(n°1)*************/
$answer1 = isset($_POST['checkzone1']) ? $_POST['checkzone1'] : NULL;
if ($answer1 == "1") {
    $req_test = $db->query("UPDATE pas SET zone_id ='1' WHERE id ='1'");
}else if($answer1 == "2") {
    $req_test = $db->query("UPDATE pas SET zone_id ='2' WHERE id ='1'");
}else if($answer1 == "3") {
    $req_test = $db->query("UPDATE pas SET zone_id ='3' WHERE id ='1'");
}else if($answer1 == "4") {
    $req_test = $db->query("UPDATE pas SET zone_id ='4' WHERE id ='1'");
}else if($answer1 == "5") {
    $req_test = $db->query("UPDATE pas SET zone_id ='5' WHERE id ='1'");
}else if($answer1 == "6") {
$req_test = $db->query("UPDATE pas SET zone_id ='6' WHERE id ='1'");
}
```

7.17 Arrêter toutes les diapositives

J'ai ajouté un bouton qui permet l'arrêt ou non de toutes les diapositives en cours. Soit avec l'état "diapos en cours" soit "diapos à l'arrêt". En fonction de l'état je modifie un champ dans la base de données, ensuite mon camarade IR utilise cette valeur la dans son programme pour que les diapositives puissent être tous à l'arrêt.

```
<?php
    $req_urgency = $db->query("SELECT isOn FROM urgency");
    $data_urgency = $req_urgency->fetch();


$etat = isset($_POST['checkzone']) ? $_POST['checkzone'] : NULL;
if ($etat == "0") {
    $req_start = $db->query("UPDATE urgency SET isOn ='0'");
}else if($etat == "1") {
    $req_stop = $db->query("UPDATE urgency SET isOn ='1'");
}

if(0 == $etat)
    $verifcheckzone1 = "checked";
else
    $verifcheckzone1 = "";

if(0 != $etat)
    $verifcheckzone2 = "checked";
else
    $verifcheckzone2 = "";

echo"
<form action='stopDiapo.php' method='post'>
    <p><span><td>Diapos en cours<input type='radio' class='radio' name='checkzone' value='0'". $verifcheckzone1 ."/></td></span>
    <td>Diapos à l'arrêt<input type='radio' class='radio' name='checkzone' value='1'".$verifcheckzone2 ."/></td></p>
    <input type='submit' class='submit' value='Valider'/>
</form>"
?>
```

Screen correspondant au code ci-dessus.

7.18 Bilan

7.19 Journal de bord

	Début du projet
06-janv.	Prise en main du sujet
07-janv.	Première réunion avec les professeurs. Création du logo et installation de project
08-janv.	Installation serveur Linux (Apache et BDD)
13-janv.	Création du diagramme de gantt
16-janv.	BDD inaccessible en dehors du réseau local (modification fichier my.cnf)
20-janv.	Apprentissage du code existant
03-févr.	Problème VM avec internet (réinstallation). Réunion rédaction partie commune
06-févr.	Rédaction partie personnelle
07-févr.	Revue 1
1er semaine Mars	Ajouts de différents champs dans la BDD + Programmation (php)
2e semaine Mars	Modif rafraîchissement + Ajout pourcentage de la SD dans "état système"
3e semaine Mars	Recherche solution vidéo sur RPI + Ajout qualité de l'air dans "état système"
4e semaine Mars	Problème d'affichage de la vidéo sur le navigateur QT de la RPI
1er semaine Avril	Réunion IR avec les professeurs + Début travail sur la gestion des zones
2e semaine Avril	Fin du travail sur la gestion des zones + Bouton arrêt des diapositives + Réunion IR et EC
3e semaine Avril	Rédaction revue 2
4e semaine Avril	Finalisation du travail

7.20 Conclusion

Ce projet m'aura permis d'approfondir mes compétences en programmation tout particulièrement en php où j'avais très peu de connaissance, mais aussi en compréhension du fonctionnement des bases de données. J'ai appris à créer un diagramme de bloc (BDD). Mais une des parties les plus intéressantes était la communication qu'elle soit entre membres du même projet ou avec nos professeurs. L'organisation en équipe était importante pour pouvoir rendre nos différentes revues sans retard ainsi qu'avancer convenablement tout en sachant où en était les autres.

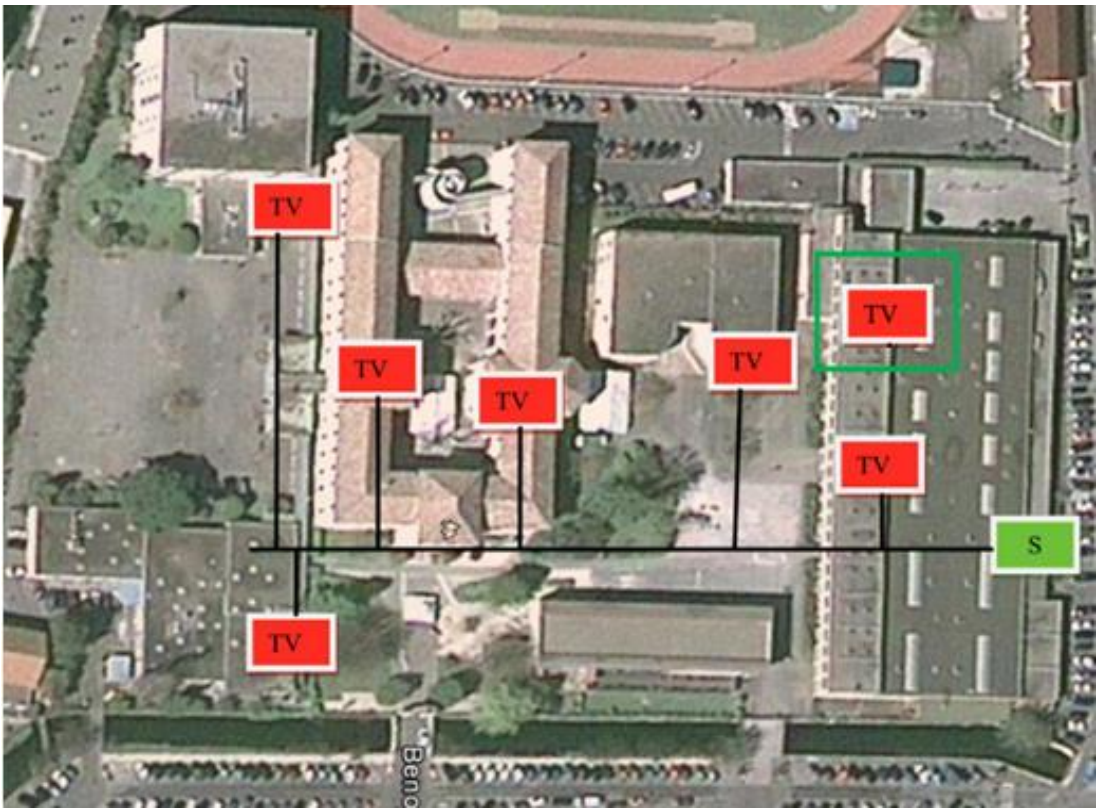
8 Partie personnelle Ryan Jacques ZEHANI

8.1 Introduction :

Je travaille sur le projet AIES (Affichage d'Informations pour Etablissement Scolaire) à la suite de trois années précédentes de réalisation en 2016, 2017 et 2018.

Ce projet a pour but de transmettre des informations aux élèves ainsi qu'aux professeurs à l'aide d'un système d'affichages dynamiques d'informations répartis dans les endroits passants du lycée.

Ce système est composé de 8 points d'affichage dont 7 actifs plus un dernier (Celui encadré en vert) qui sera rajouté au cours de l'année.



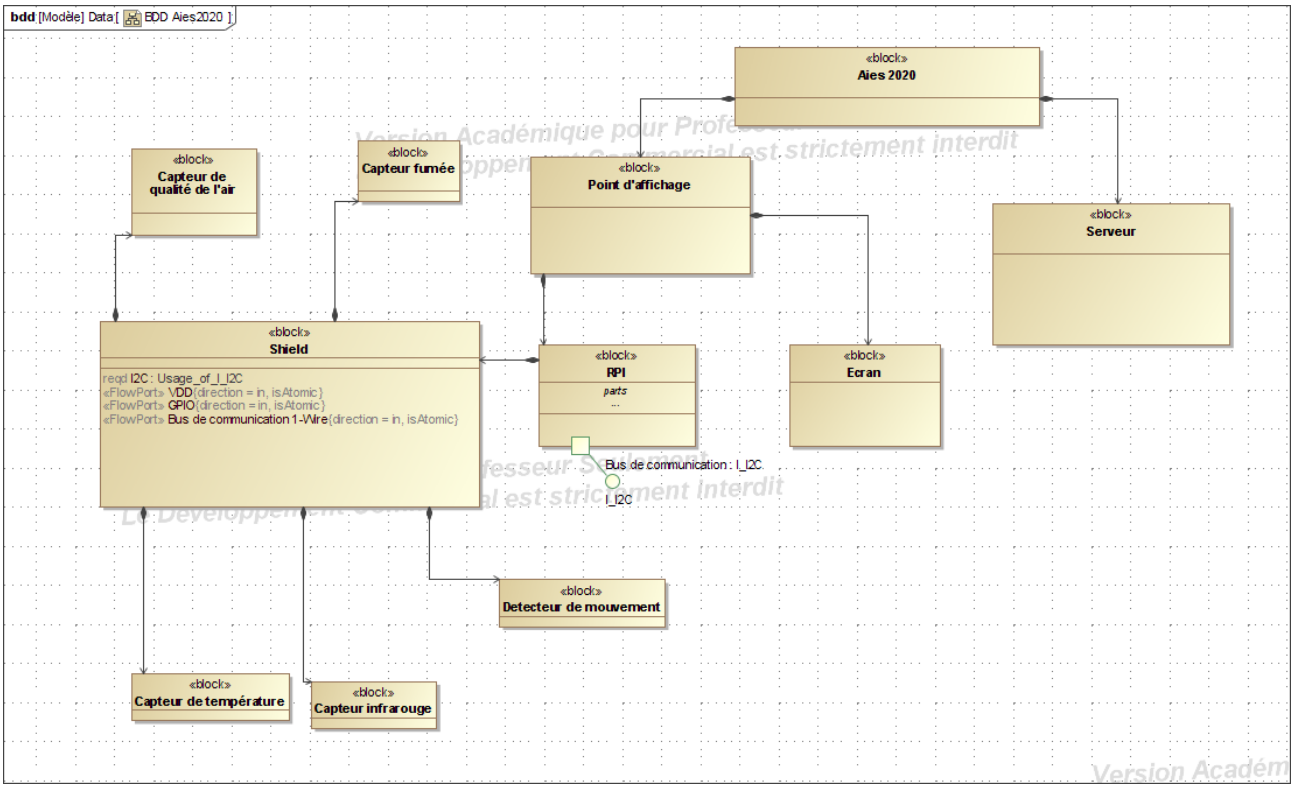
Cette année l'évolution cette année consiste à ajouter un détecteur de fumée ainsi qu'un capteur de qualité de l'air. Les logiciels seront aussi améliorés.

Un point d'affichage est composé :

• D'un écran de télévision.
• D'une carte RASPBERRY PI 2 B+.
• Un capteur de présence afin de permettre l'extinction automatique des écrans lorsque aucune personne ne passe (temporisé 1h).
• Un capteur de température.
• Un capteur de qualité de l'air (rajouté cette année).
• Un détecteur de fumée (rajouté cette année).
• Une carte de réception d'une horloge.
• Un émetteur infrarouge pour commander les TV.

Mon but dans ce projet est l'installation du capteur de fumée.
Je travaille en collaboration avec deux techniciens IR qui s'occupent de l'amélioration de la base de données, du site web et du programme sur la Raspberry. Je suis également accompagné d'un technicien EC qui a en charge l'installation du capteur de qualité de l'air.
BDD du système

Nous avons réalisé un BDD (Diagramme de Bloc) permettant de représenter toutes les parties physiques de notre système.



Et comme dit précédemment, j'ai principalement en charge la partie capteur de fumée.

8.2 Travail effectué :

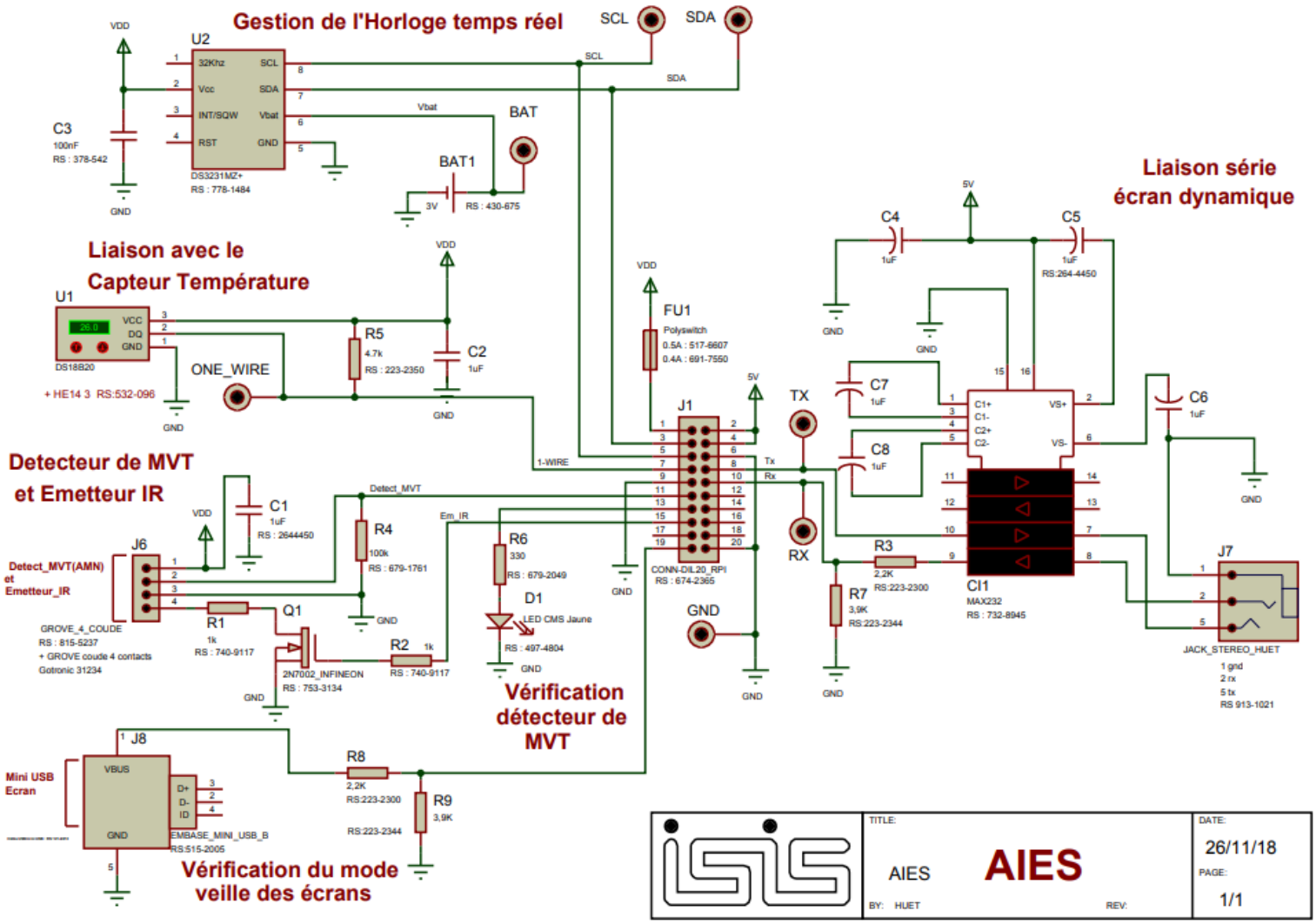
Prise en main du Projet et des parties déjà présentes

Réalisation d'un Gantt

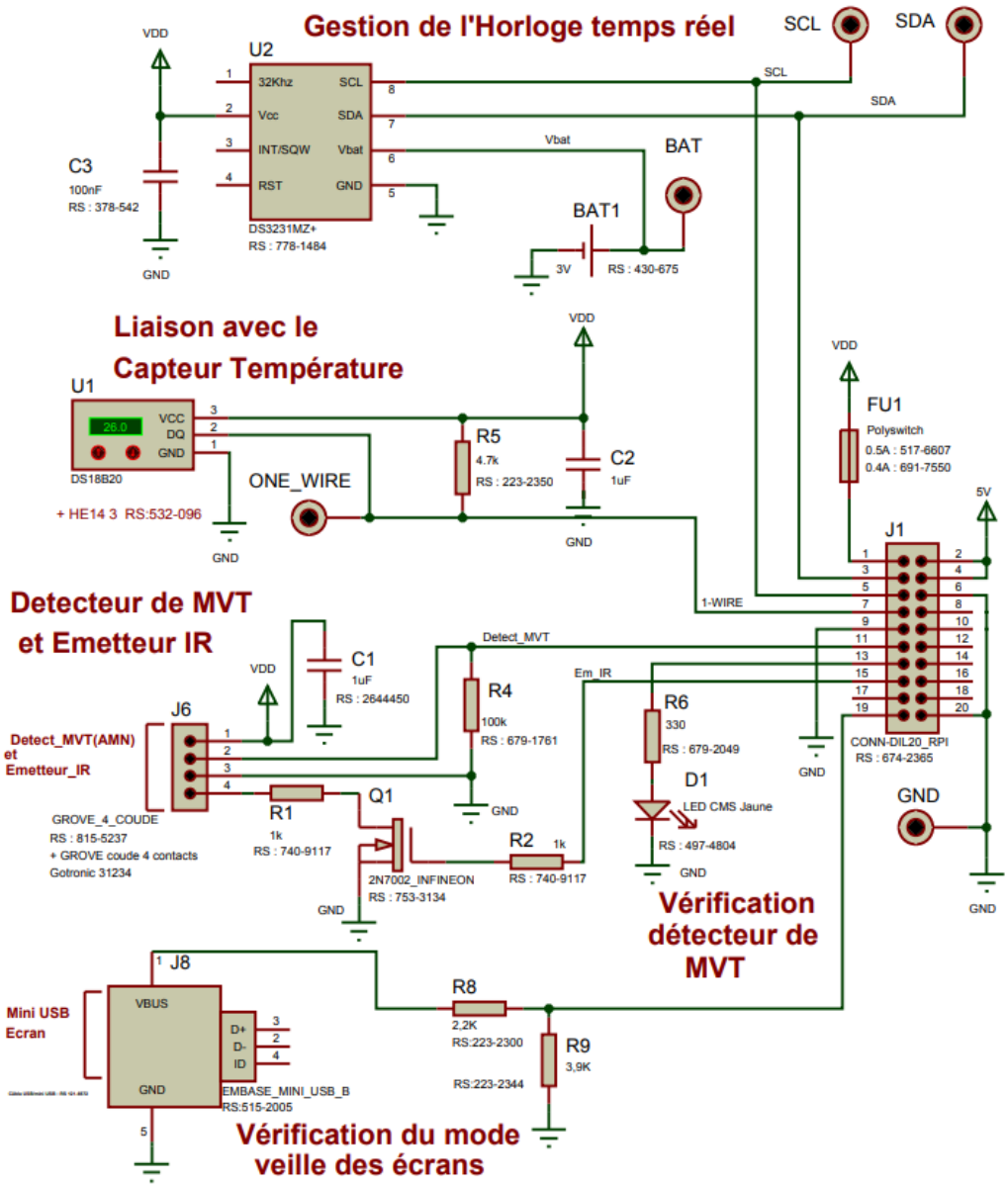
Test des différents capteurs mis à ma disposition

8.3 Prise en main du projet:

J'ai commencé par analyser le schéma structurel de la carte de 2018, ensuite je me suis renseigné sur toutes les parties physiques du système déjà présentes (capteurs, détecteurs, horloge, ect…) et j'en ai fait un résumé, que j'ai inséré dans ma partie physique présentée après.



Dans un premier temps sur cette carte des année précédentes, il nous a été demandé de supprimer toute la partie de droite (Liaison série écran dynamique) comme montré ci-joint, pour la remplacer par une liaison HDMI et par la suite, ajouter notre capteur de fumée ainsi que celui de qualité de l'air, pour ce qui est de la partie EC.



Après plusieurs réflexions ainsi que plusieurs difficultés, nous avons décidé avec nos professeurs, de ne pas changer la carte du projet 2018 mais, de simplement changer le connecteur « J1 » et, d'y installer à la place un connecteur adapté pour la fixation d'un HAT, auquel nos capteurs de fumée et de qualité de l'air y seront soudés.

8.4 Réalisation d'un Gantt :

J'ai réalisé un Gantt prévisionnel de mes tâches à réaliser avant la revue 1, que je présente ci-contre.

Projet	63 hr	Lun 06/01/20	Jeu 05/03/20		
Specifications générales	**17 hr**	**Lun 06/01/20**	**Jeu 16/01/20**		
Prise en main du Projet	2 hr	Lun 06/01/20	Lun 06/01/20		EC1
Creation du diagrame de gantt	8 hr	Lun 06/01/20	Jeu 09/01/20	10	EC1
Convenir d'une charte graphique pour le dossier	2 hr	Lun 13/01/20	Lun 13/01/20	11	EC1
Analyser le schema structurel de la carte de 2018	5 hr	Lun 13/01/20	Jeu 16/01/20	12	EC1
Essais et validation des structures (prototypage rapide)	**38 hr**	**Jeu 16/01/20**	**Mar 11/02/20**	**13;9**	
BDD IBD	7 hr	Jeu 16/01/20	Mar 21/01/20		EC1
Effectuer des essais comparatifs sur les capteurs de détection de fumée	20 hr	Mar 21/01/20	Mar 04/02/20	15	EC1
Illustration des essais de prototypage avec un schema	4 hr	Mar 04/02/20	Jeu 06/02/20	16	EC1
Saisie completes du schema sur KICAD	10 hr	Jeu 06/02/20	Lun 02/03/20	17	EC1
Dresser la listes des composant pour la nouvelles structure	5 hr	Lun 02/03/20	Jeu 05/03/20	18	EC1
Revue 1	0 hr	Lun 06/01/20	Lun 06/01/20		

Puis voici les tâches que j'ai pu réaliser, pour le moment.

Projet	44 hr	Lun 06/01/20	Jeu 30/01/20		
Specifications générales	**11 hr**	**Lun 06/01/20**	**Lun 13/01/20**		
Prise en main du Projet	4 hr	Lun 06/01/20	Mar 07/01/20		EC1
Creation du diagrame de gantt	3 hr	Mar 07/01/20	Jeu 09/01/20	10	EC1
Convenir d'une charte graphique pour le dossier	1 hr	Jeu 09/01/20	Jeu 09/01/20	11	EC1
Analyser le schema structurel de la carte de 2018	3 hr	Jeu 09/01/20	Lun 13/01/20	12	EC1
Essais et validation des structures (prototypage rapide)	**23 hr**	**Jeu 16/01/20**	**Jeu 30/01/20**	**13;9**	
BDD IBD	6 hr	Jeu 16/01/20	Lun 20/01/20		EC1
Effectuer des essais comparatifs sur les capteurs de détection de fumée	15 hr	Mar 21/01/20	Jeu 30/01/20	15	EC1
Illustration des essais de prototypage avec un schema	2 hr	Jeu 30/01/20	Jeu 30/01/20	16	EC1
Saisie completes du schema sur KICAD	0 hr	Jeu 30/01/20	Jeu 30/01/20	17	EC1
Dresser la listes des composant pour la nouvelles structure	0 hr	Jeu 30/01/20	Jeu 30/01/20	18	EC1
Dossier Pdf	10 hr	Lun 06/01/20	Jeu 09/01/20		
Revue 1	0 hr	Lun 06/01/20	Lun 06/01/20		

8.5 Test des Différent Capteurs.

Pour mon projet j'ai eu le choix entre 3 capteurs différents donnés ci-joint :

Le MQ-135

Le MAX30105

Le MQ-2

Dans un premier temps, j'ai réalisé des recherches et j'ai pu observer que la programmation ainsi que le câblage des deux capteurs MQ étaient identique. J'en ai donc profité pour tester ces deux capteurs avec le même programme, ainsi que le même câblage.
Une différence constatable entre le MQ-2 et le MQ-135 est le nombre de broches des capteurs.
Le MQ-2 dispose de 3 broches (5 V, GND, Sortie Analogique).
Le MQ-135 dispose de 4 broches (5 V, GND, Sortie Analogique, Sortie Numérique).
Par la suite je me suis penché sur le MAX30105 et j'ai pu constater qu'il est généralement utilisé via une carte et un programme Arduino, j'ai donc dû réaliser mes tests pour ce capteur via Arduino.

8.6 Capteur MQ :

J'ai fait des recherches internet sur ce capteur et j'ai pu trouver un site qui fournissait une très bonne explication, ainsi qu'un programme muni de son câblage pour les deux capteurs MQ. J'ai pu constater quelques problèmes dans le câblage et j'ai donc appliqué mes correctifs.

Schéma capteur MQ-2

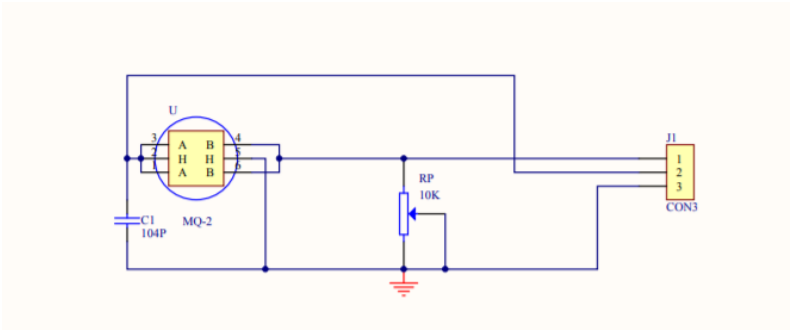
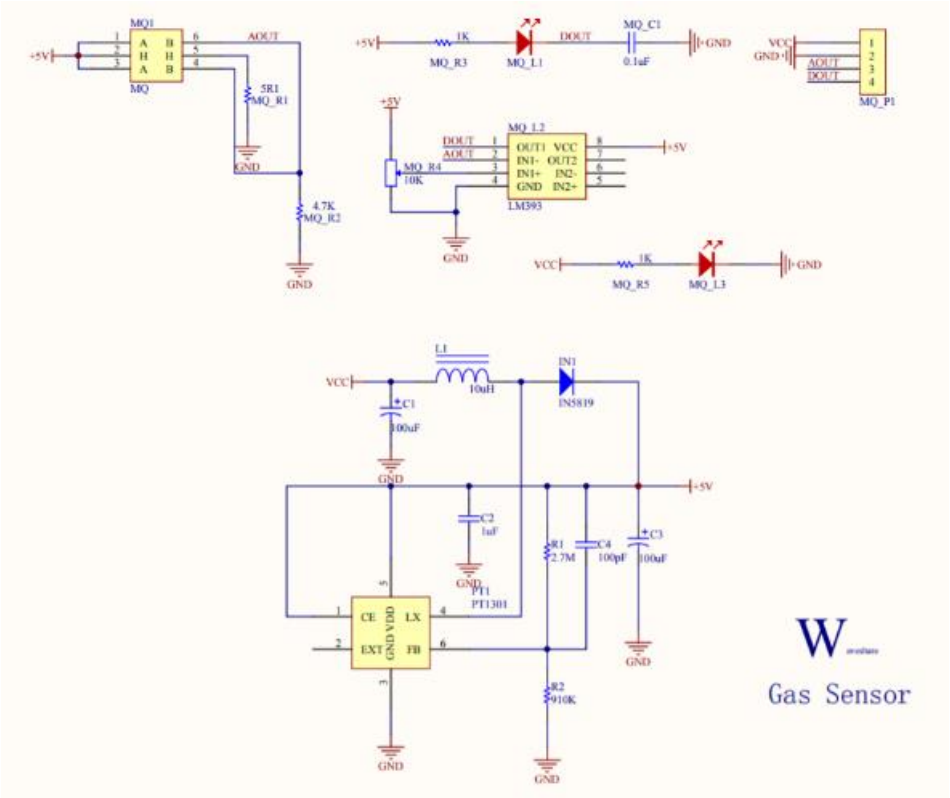
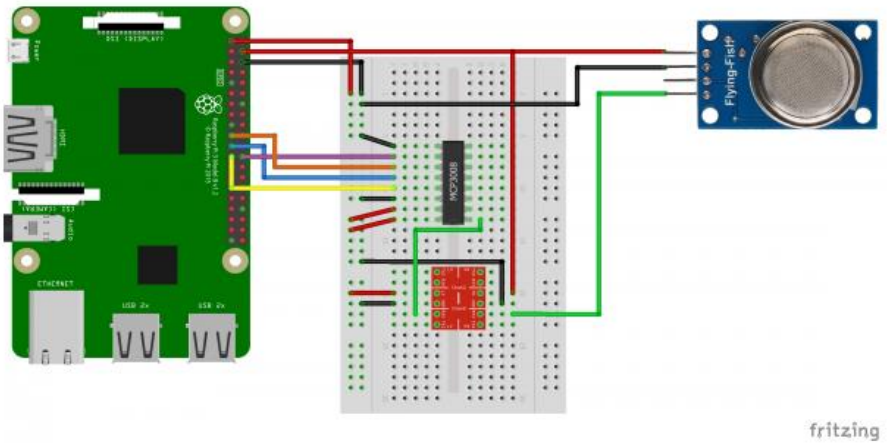


Schéma capteur MQ-135

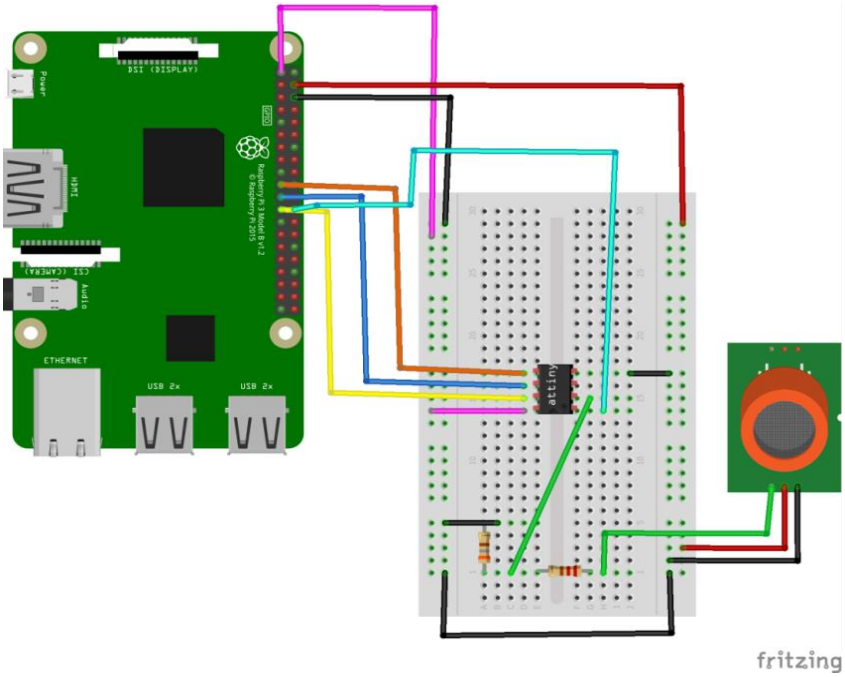


8.7 Câblage :

Câblage fourni sur internet :



Correctifs apportés au câblage :



Je me suis renseigné sur les Convertisseurs Analogiques Numériques (CAN), celui demandé dans le câblage initial était de référence MCP3008 et j'ai constaté que seulement un seul channel du CAN était utilisé pour ce montage. J'en ai discuté avec mon professeur et il m'a conseillé d'utiliser un MCP3002 constitué de seulement 2 channels, donc plus petit et plus optimisé.

J'ai aussi inséré à mon montage un pont diviseur de tension sur les bus reliant mon capteur au CAN en remplacement du level shifter, que nous n'avions pas en stock au lycée.

8.8 Programme :

Un Programme Python sur GitHub était fourni pour le test de ce capteur j'ai donc téléchargé ce programme sur ma RPI, via la commande suivante.

```
git clone https://github.com/tutRPi/Raspberry-Pi-Gas-Sensor-MQ
```

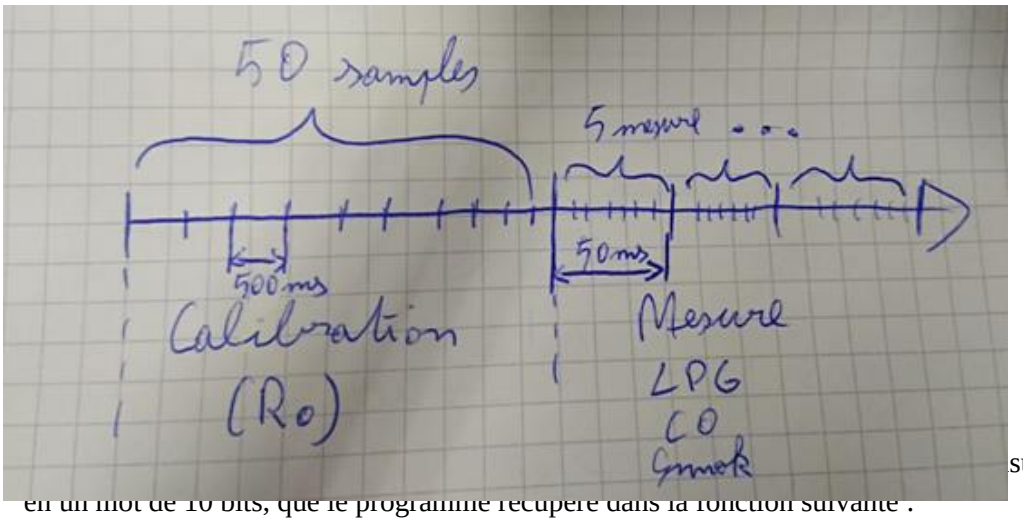
J'ai étudié le programme ainsi que la documentation du capteur pour comprendre le fonctionnement du capteur, et je vais expliquer les parties les plus importantes liées à la mesure et au calcul de la mesure.

```
class MQ():

    ######################### Hardware Related Macros #########################

    MQ_PIN                      = 0  # define which analog input channel you are going to use (MCP3008)
    RL_VALUE                    = 5         # define the load resistance on the board, in kilo ohms
    RO_CLEAN_AIR_FACTOR         = 9.83      # RO_CLEAR_AIR_FACTOR=(Sensor resistance in clean air)/RO,
                                            # which is derived from the chart in datasheet

    ######################### Software Related Macros #########################
    CALIBARAION_SAMPLE_TIMES    = 50        # define how many samples you are going to take in the calibration phase
    CALIBRATION_SAMPLE_INTERVAL = 500       # define the time interval(in milisecond) between each samples in the
                                            # cablibration phase
    READ_SAMPLE_INTERVAL        = 50        # define the time interval(in milisecond) between each samples in
    READ_SAMPLE_TIMES           = 5         # define how many samples you are going to take in normal operation
                                            # normal operation

    ######################### Application Related Macros ######################
    GAS_LPG                     = 0
    GAS_CO                      = 1
    GAS_SMOKE                   = 2
```

On peut voir sur ces quelques lignes la fonction liée à la calibration du capteur dans un endroit sain (un endroit où le taux de particule dans l'air est « normal »).

J'ai pu comprendre que le capteur, pendant la calibration, fait 50 mesures de toutes les 500 ms donc sur un temps de 25 000 ms, et par la suite passe en mode mesure normal, et là on peut observer qu'il fait 5 mesures toutes les 50 ms.

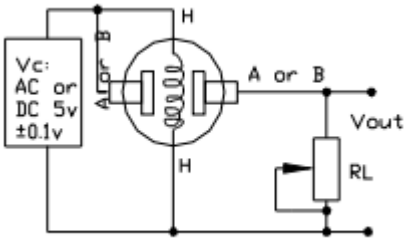
J'ai donc défini une frise chronologique du fonctionnement du capteur :

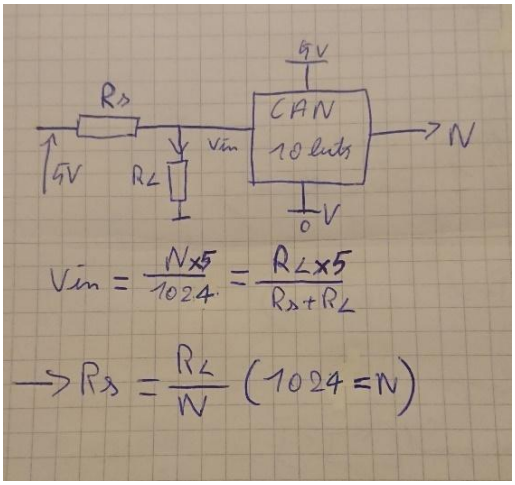


suite la convertit en un mot de 10 bits, que le programme récupère dans la fonction suivante :

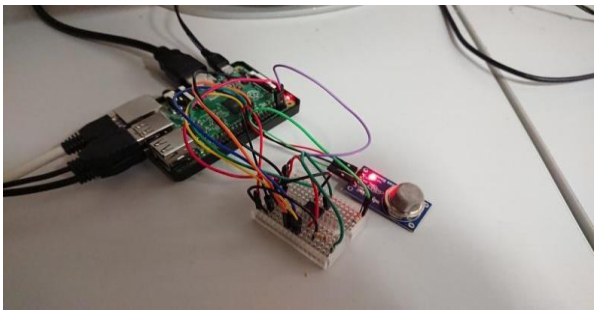
```
69      def MQResistanceCalculation(self, raw_adc):
70          return float(self.RL_VALUE*(1023.0-raw_adc)/float(raw_adc));
71
```

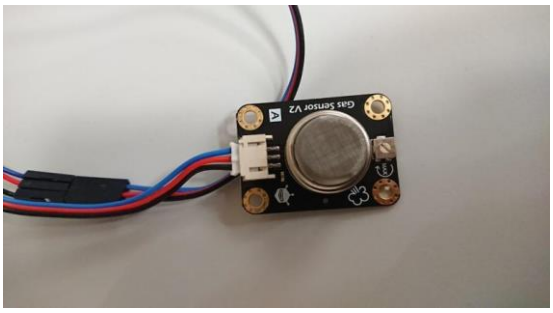
Explication par un calcul de la fonction :

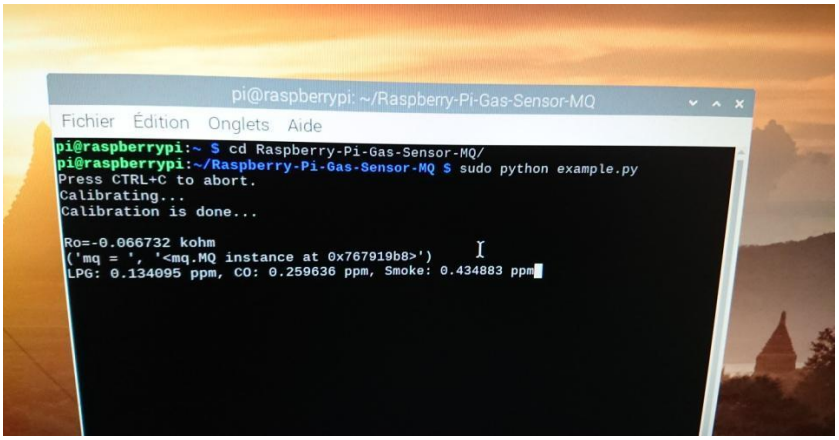




Quelques photos des deux capteurs câblés à la RPI ainsi que l'information affichée à l'écran, à la suite du lancement du programme.







On peut donc observer le taux de particule par Million (pmp) pour le LPG, le CO, et le Smoke qui nous intéresse plus particulièrement dans mon projet.

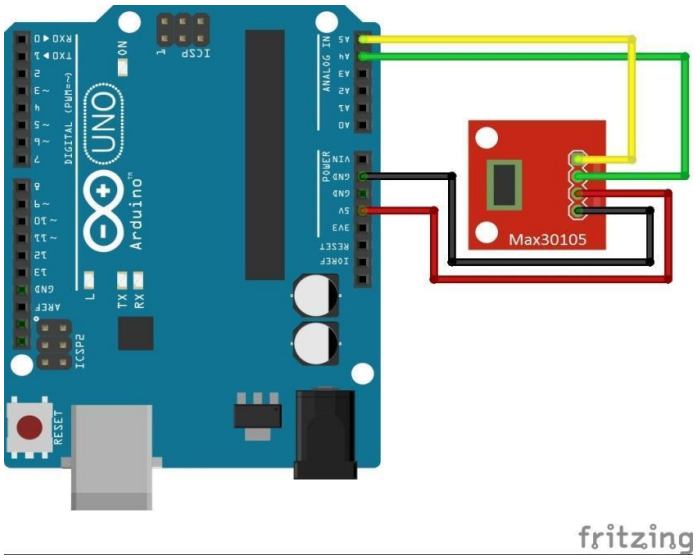
Ro signifie la résistance que mon capteur se fixe à son démarrage, celle-ci est calculée en fonction du taux de particule dans le milieu où il fonctionne à son démarrage.

8.9 Capteur MAX30105 :

Comme présenté précédemment j'ai réalisé mes tests pour ce capteur sur une carte Arduino.

J'ai trouvé un programme ainsi qu'un câblage pour réaliser mes tests.

Voici le câblage



J'ai exposé à la suite le programme Arduino.

```
#include <Wire.h>
#include "MAX30105.h"

MAX30105 particleSensor;
void setup()
{
Serial.begin(115200);
Serial.println("MAX30105 Basic Readings Example");
if (particleSensor.begin() == false)
{
Serial.println("MAX30105 was not found. Please check wiring/power. ");
while (1);
}
void loop()
{
Serial.print(" R[");
Serial.print(particleSensor.getRed());
Serial.print("] IR[");
Serial.print(particleSensor.getIR());
Serial.print("] G[");
Serial.print(particleSensor.getGreen());
Serial.print("]");
Serial.println();

}
```

Après mes tests, j'ai réalisé un tableau de confrontation pour mes différents capteurs.
Cela a permis de m'orienter vers un capteur en particulier, qui à mon sens est le plus approprié pour mon projet : Le MQ-2

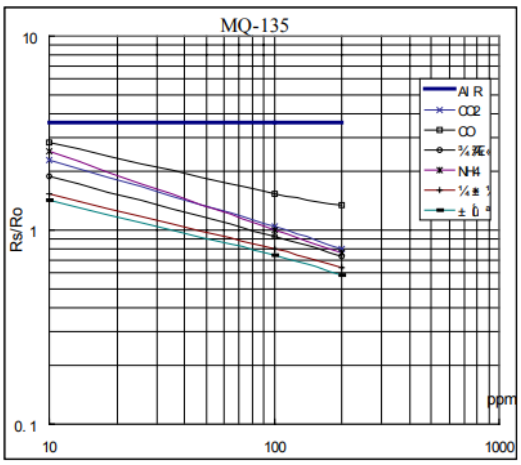
Max 30105	MQ-135	MQ-2
Câblage simple	Besoin convertisseur analogique numérique	Besoin convertisseur analogique numérique
	arret du capteur quand mise en veille RPI	
8,14 TTC	6,20 TTC	7,60 TTC
Utilisation généralement faite via Arduino	RPI Liaison SPI	RPI Liaison SPI
N'arrive pas à faire la différence entre gaz et objet		
Perturbation par la lumière ambiante		
Compact CMS	Compact mais assez gros capteur	Compact mais assez gros capteur
Ne détecte pas les particules, détecte l'opacité de la fumée	Détection taux de particules plus faible	Détection taux de particules 50 fois supérieur au MQ-135
5V alimentation signal 3,3V	5V alimentation signal 3,3V	5V alimentation signal 3,3V
8	11	14
2,99 mA	2,9 mA	2,92 mA
	2 PTS	
	1 PTS	
	0 PTS	

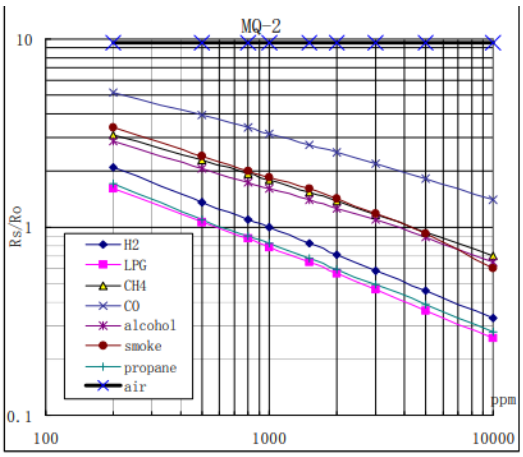
Raison de mon choix :

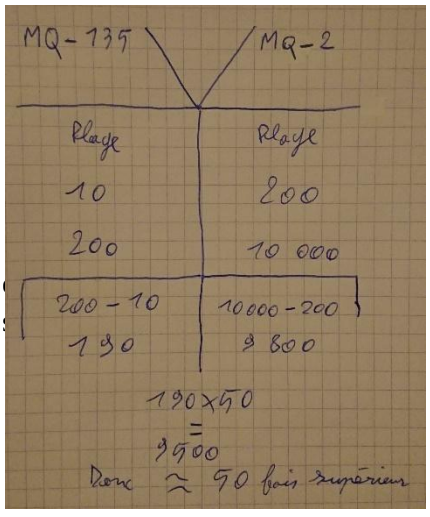
Pour le Max 30105, son plus gros problème est que ce capteur ne détecte pas vraiment le nombre de particules dans l'air, mais détecte l'opacité de la fumée.
Dans l'établissement, selon moi ce choix n'est pas le plus approprié.

Mon choix s'est donc réduit aux deux capteurs MQ et j'ai remarqué que pour le capteur MQ-135, la mise en veille de la RPI coupe la mesure du taux de particules.

Mais j'ai aussi observé que la détection du taux de particules du MQ-2 est 50 fois supérieur à celui du MQ-135 comme montré ci-joint.







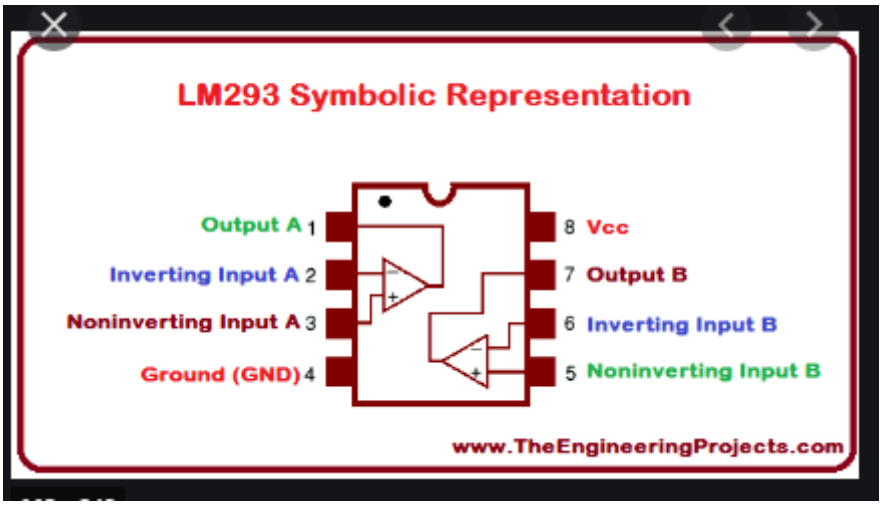
8.10 Prototypage du projet

Après en avoir discuté avec mes professeurs, nous avons déduit que le plus approprié pour notre projet est la création d'un HAT que l'on pluggera sur la carte de 2018 après la modification de son connecteur, cela sera plus simple, plus rapide, et surtout plus économique.

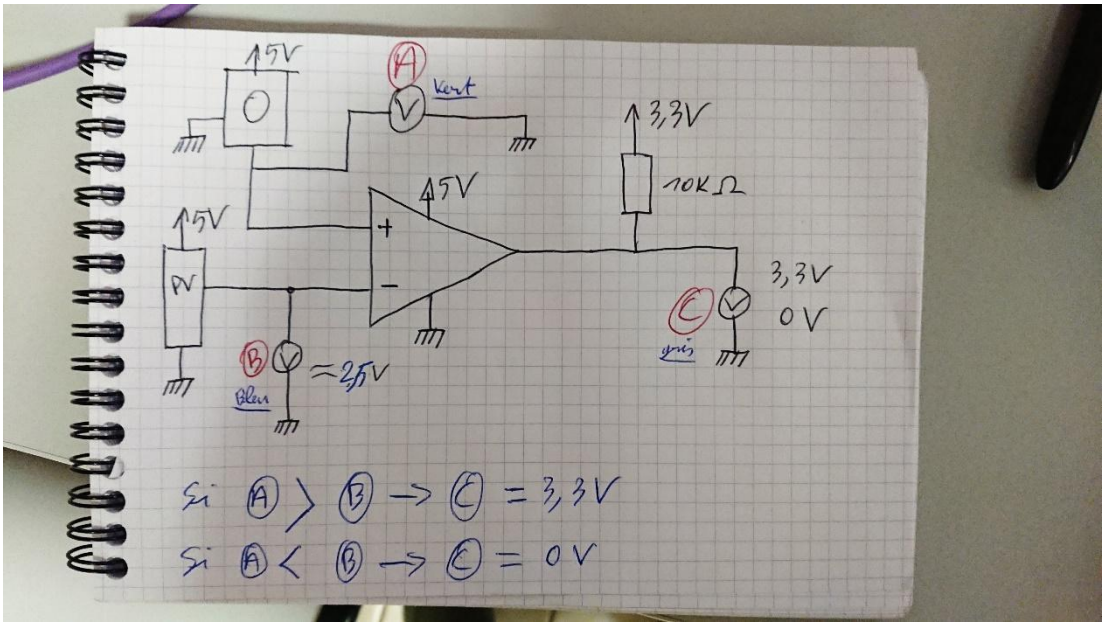
Pour ma partie personnelle (détecteur de fumée) nous avons convenu de remplacer mon convertisseur analogique numérique, par un comparateur a collecteur ouvert plus simple, plus compact, plus approprié.

A collecteur ouvert était un critère de choix, par le fait de la non nécessité a y introduite un level shifter pour adapté sa tensions d'entrer à celle du système (5V au lieur du 3,3V).

J'utilise donc pour mon prototypage, un LM 293 N :



8.11 Schéma de mon prototype



8.12 Explication du fonctionnement

Le potentiomètre représenté par le multimètre B, me permet de fixer un seuil dans mon montage, en entrant une tension sur la broche moins du comparateur.

Mon capteur étant fixé par son potentiomètre personnel a une tension de base en environnement sain, est représenté par le multimètre A.

Le multimètre C représente la tension de sortie du comparateur.

Après avoir fixé un seuil sur mon capteur à l'aide de son potentiomètre personnel, on peut fixer le seuil du comparateur avec le potentiomètre relié à la broche - du comparateur.

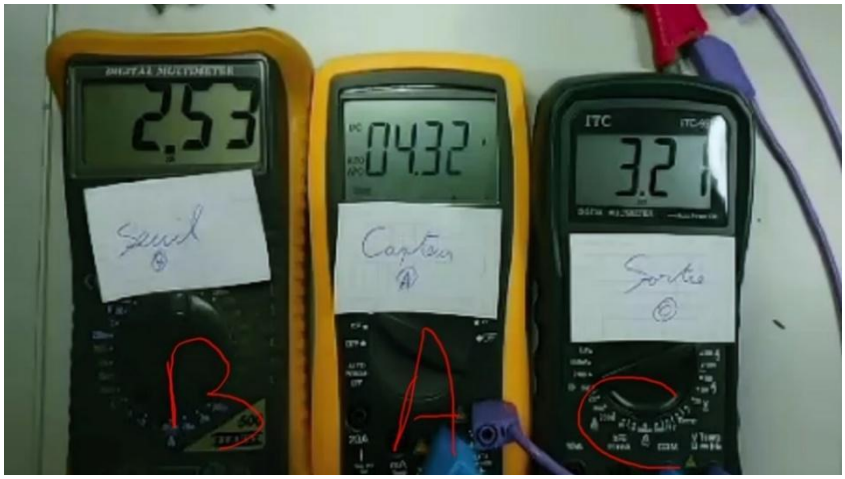
J'ai donc fixé mon seuil du comparateur à 2.5 V et le seuil de mon capteur à 1.15 V en environnement sain, on voit donc que la sortie est à 0 V.

Quand mon capteur détecte de la fumée, sa tension de sortie s'élève à 4,3 V et par la suite le comparateur ne sort rien sur sa sortie, le 3,3V prend donc le dessus sur la sortie, sa sortie passe à 3.3V, il y a un incendie.

Le potentiomètre permet donc de fixer un seuil à dépasser par le capteur, pour que le comparateur puisse sortir une tension ou non, à sa sortie.

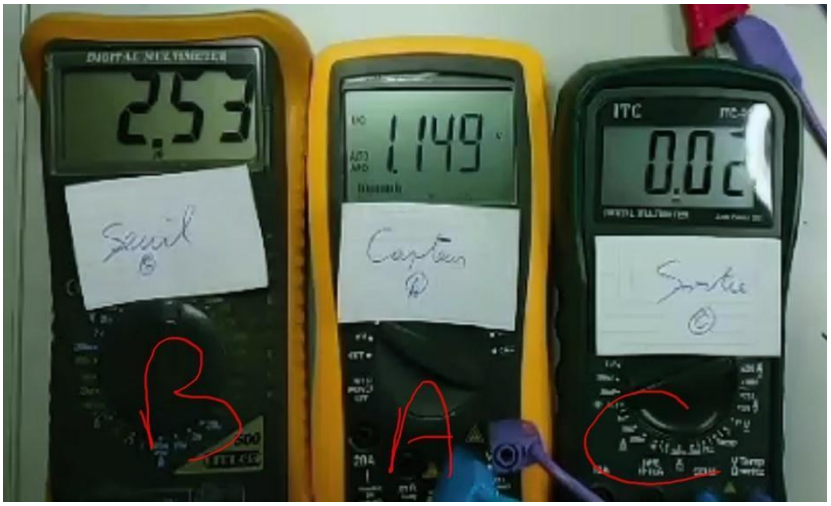
Environnement enfumé

Sortie 3.3V

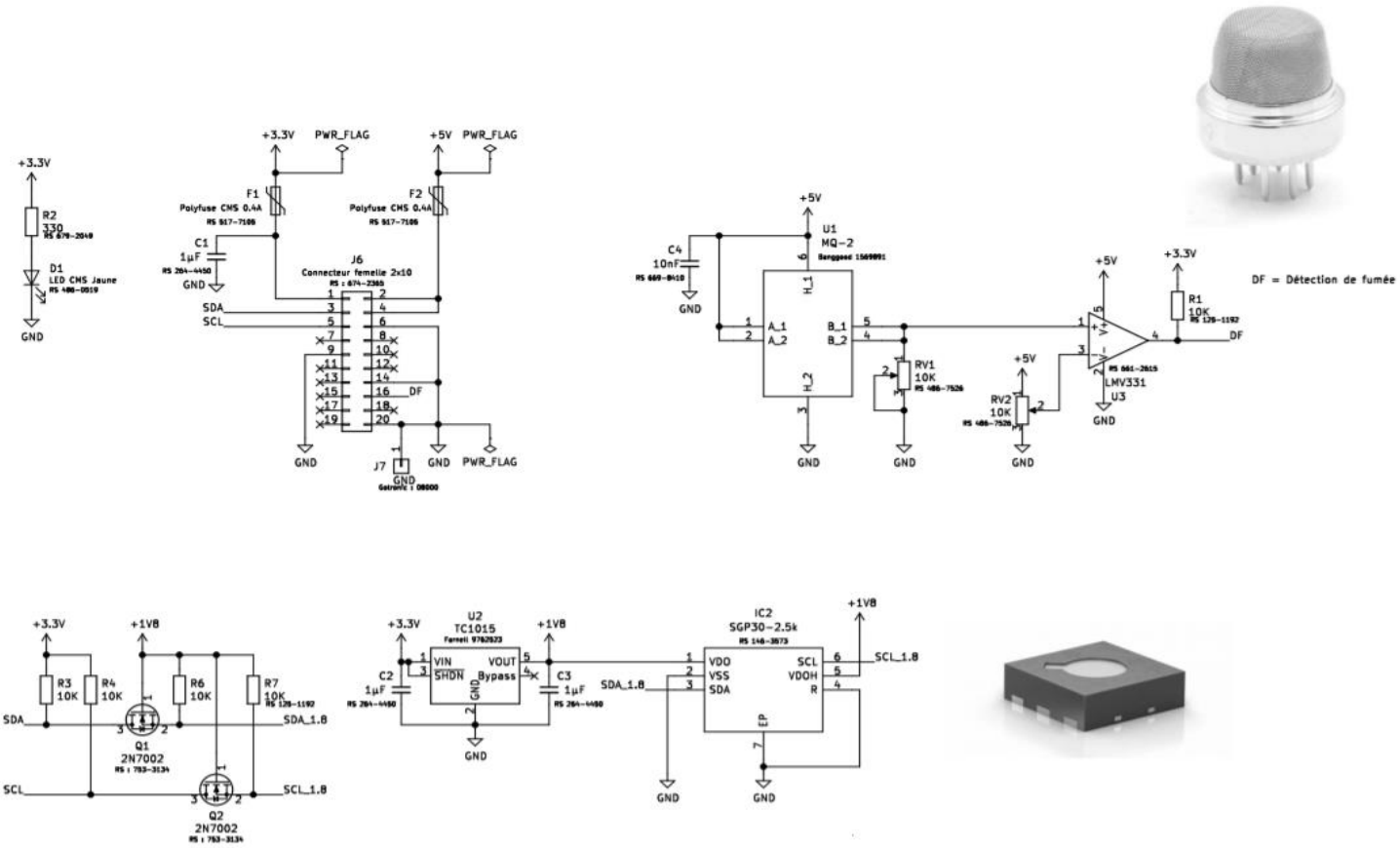


Environnement sain

Sortie 0V



8.13 Représentation du HAT à l'aide de Kikad



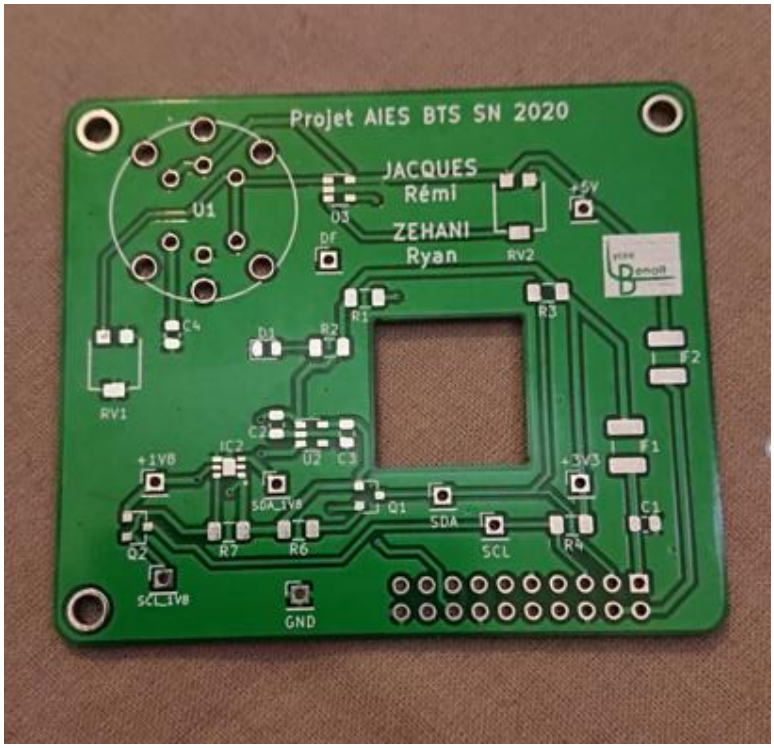
Voici le schéma structurel pour le HAT comportant le détecteur de fumée ainsi que le capteur de qualité de l'air et tous les composants permettant leurs fonctionnements.

Après avoir réalisé ce schéma nous sommes passés au routage toujours à l'aide de Kikad pour crer ce HAT en question en modèle 3D.

8.14 Création physique du HAT et réalisation final de la carte

A la suite du routage nous l'avons enregistré au format Gerber que nous avons transmis à une entreprise qui nous a fabriqué notre HAT.

Apres plusieurs semaines voici le résultat :



Après la réception des composants nous avons commencé la réalisation de notre carte.

Cela se réalise en 4 étapes :

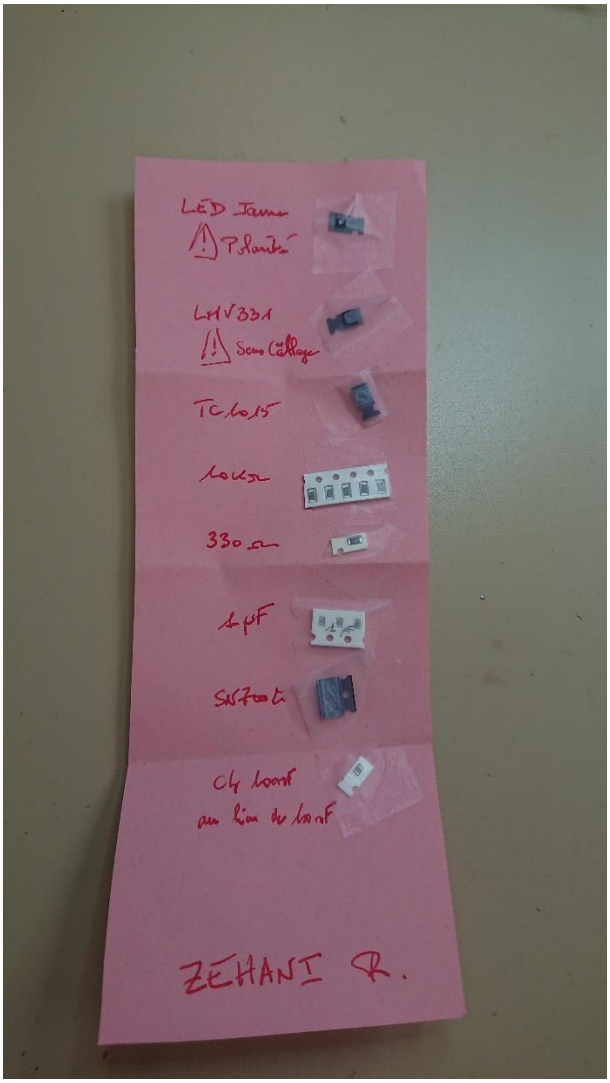
Etape 1 = Repérage des composants ;

Etape 2 = Mise en place de la pâte à Brasé ;

Etape 3 = Installation des composants et Passage au four à refusion ;

Etape 4 = Soudure des composants traversant et test de la carte ;

Etape 1

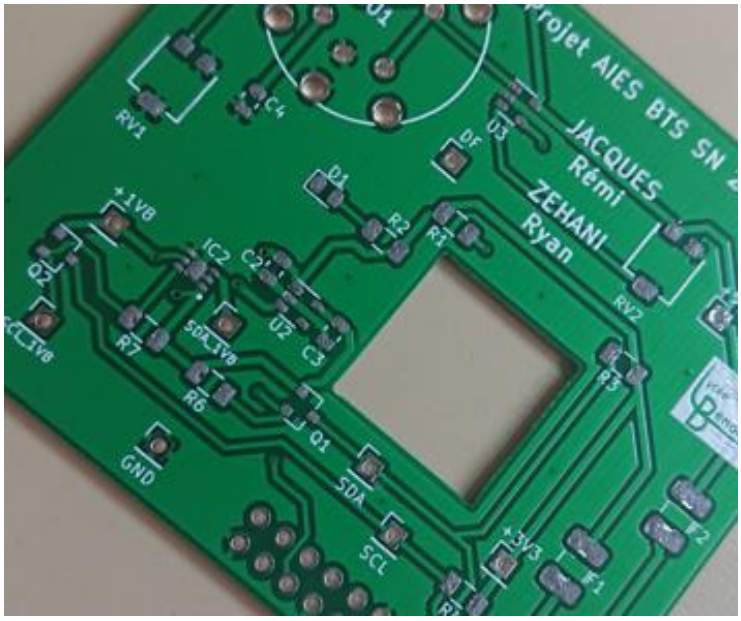


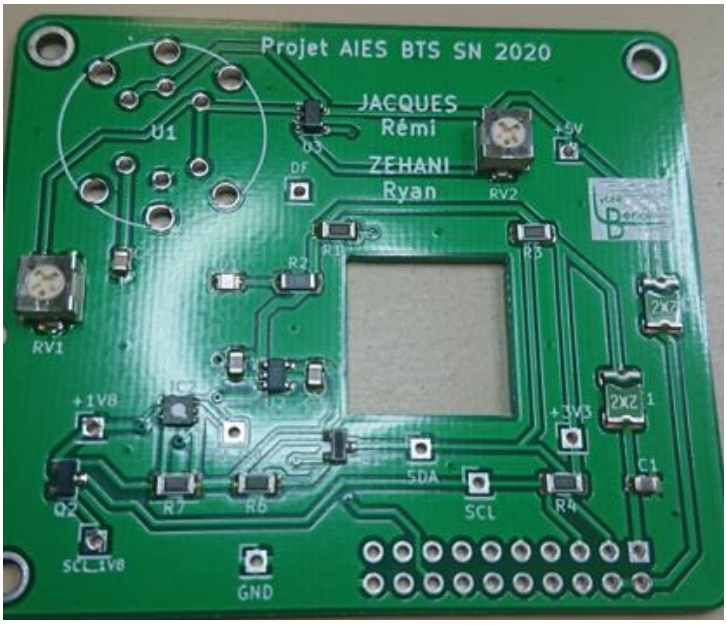
Etape 2

A l'aide d'un support nous maintenons la carte à son emplacement pour pouvoir y poser au-dessus le Stencil (plaque grise) pour y insérer la pâte à Brasé, à l'emplacement des trous pour les composants.

En enlevant le Stencil, nous nous retrouvons avec une carte ayant de la pâte à Brasé à tous les bons emplacements de soudure. Il ne reste plus qu'à installer les composants au bon endroit en suivant le routage sur Kikad.

Etape 3

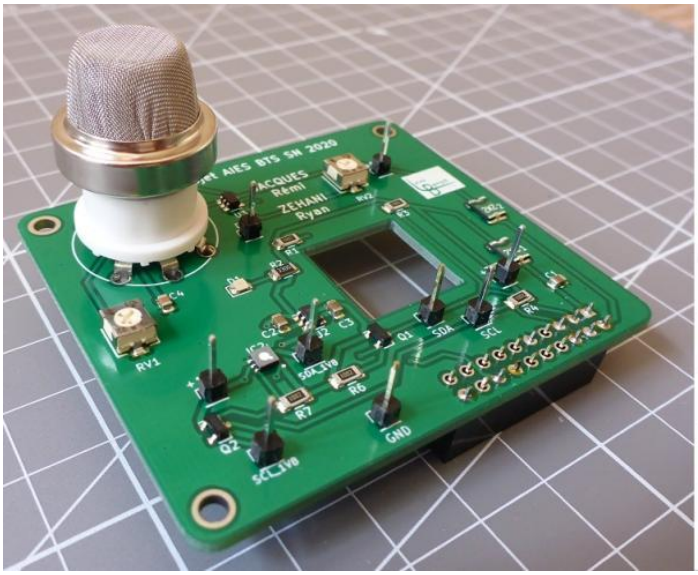




Après avoir retiré le Stencil, il suffit de vérifier au microscope si les points de pâte à Brasé ne font pas de court-circuit entre eux. Par la suite l'installation de composant est suivie d'un passage au four à refusion afin de les souder.

Etape 4

Les composants traversant comme les points de test ou les borniers de connexion sont soudés à la main avec un fer à souder, afin de finaliser la carte. Les tests sont réalisés à la suite pour vérifier que tous les composants sont fonctionnels, ainsi que la carte dans son ensemble.



9 Partie physique :

9.1 Un capteur MQ-2 qu'est-ce c'est ?

Le capteur MQ-2 est un micro-capteur de fumée et de gaz, je l'utilise comme capteur de fumée dans mon projet, je vais donc généralement expliquer la partie fumée.

Le capteur MQ-2 est un capteur de fumée avec une sortie analogique (AOut) qui signale la présence de fumée en élevant la tension en sortie.

Plus il y a de fumée et plus la tension monte.

Il est possible de régler la sensibilité du capteur à l'aide du potentiomètre se trouvant à l'arrière du module, ce dernier permet d'ajuster un seuil d'activation pour le signal de sortie.

Le capteur ne nécessite qu'une seule broche d'entrée analogique d'un microcontrôleur, ce qui le rend très pratique.

Le capteur de fumée est un micro capteur, dans mon cas un semi-conducteur qui détecte la présence de fumée à des concentrations de 200 ppm à 10 000 ppm (Partie par million c'est le nombre de particule de fumée dans 1 millions de particules d'air)

Le capteur détecte la concentration de fumée dans l'air et sort le résultat comme une tension analogique.

Le capteur peut fonctionner à des températures allant de -10 à 50 ° C et consomme moins de 150 mA à 5 V.

9.2 Les particules qu'il mesure

Le CO (monoxyde de carbone) : résultat d'une combustion incomplète, quel que soit le combustible utilisé (bois, butane, charbon, essence, gaz naturel).

Le CO2 (dioxyde de carbone) : résulte d'une combustion complète.

Le gaz de pétrole liquéfié (GPL) ou (LPG) : est un mélange d'hydrocarbures légers, issu du raffinage du pétrole pour 40 % et de traitement du gaz naturel pour 60 %. Les hydrocarbures constituant le GPL sont essentiellement le propane et le butane.

9.3 Procédé de mesure

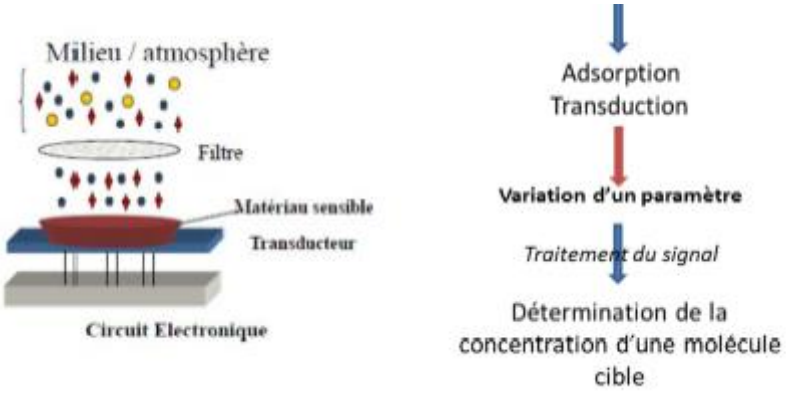
9.4 Qu'est-ce qu'un micro-capteur ?

Principe général

Un capteur est un composant dont le fonctionnement est basé sur un phénomène physique, chimique ou biologique permettant de transformer une grandeur physico-chimique en un signal, généralement électrique.

Un capteur de pollution de l'air transforme la concentration gazeuse ou particulaire dans une atmosphère en une variation de tension, de courant ou d'impédance électrique.

Principe général d'un exemple de capteur de gaz



La couche sensible permet la reconnaissance d'un gaz cible avec lequel elle interagit.

Le système transducteur transforme l'interaction chimique/physique, en un signal électrique.

Un filtre peut être appliqué en amont du dispositif, pour améliorer la spécificité de détection

9.5 Mieux comprendre mon capteur, ce qu'il mesure

Capteur semi-conducteur

Ces capteurs contiennent une couche sensible semi-conductrice qui possède une conductivité dépendante de la composition de l'air qui l'entoure.

Le signal obtenu correspond donc à une résistance variable, en fonction de l'évolution de la conductivité.

Des défauts dans le réseau cristallin sont associés aux propriétés électriques des oxydes métalliques. En effet, ces composés métalliques sont souvent isolants à l'état de cristal parfait.

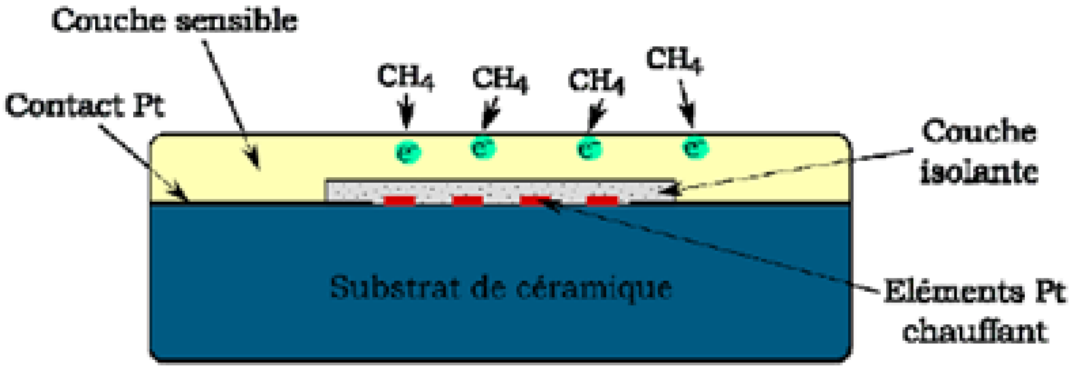


Schéma d'un capteur semi-conducteur

Il existe deux types de semi-conducteurs :

· De type n : il possède des atomes de métal en excès qui peuvent s'ioniser facilement et libérer des électrons permettant le transport du courant.

· De type p : certains nœuds du métal restent vacants, c'est pourquoi des ions perdent un électron de manière a conservé l'électroneutralité du composé. La conductivité électrique est permise par les trous ainsi formés.

Par ces défauts des oxydes métalliques, on introduit des porteurs de charge (électrons ou trous vacants) qui conditionnent les propriétés électriques et optiques du matériau.

Le retrait ou le don par un gaz d'un porteur de charge, modifie le nombre de ceux-ci et donc la conductivité.

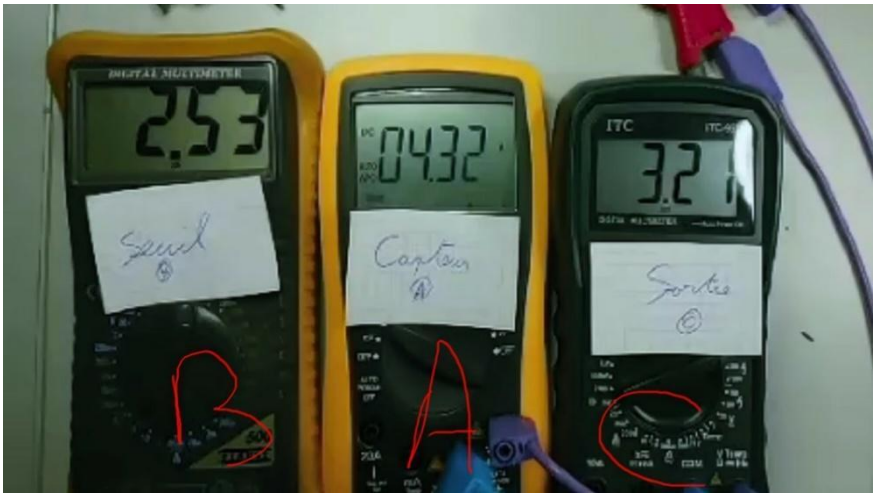
C'est pourquoi la mesure de la conductivité d'un matériau semi-conducteur permet de détecter des gaz.

De manière générale, les gaz oxydants diminuent la conductivité alors que les gaz réducteurs l'augmentent pour les semi-conducteurs de type n, et l'effet inverse est observé avec des semi-conducteurs de type p.

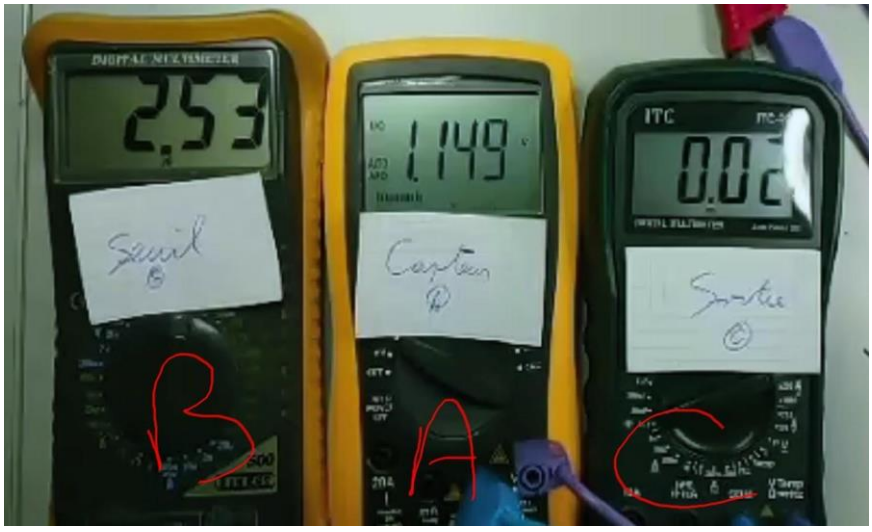
Limites et points forts

Ce type de capteur démontre une forte sensibilité mais est peu sélectif à un gaz spécifique. Il est donc nécessaire d'utiliser des matériaux qui filtreront le gaz cible.
Pour qu'un changement de conductivité soit observé en présence de gaz, il est nécessaire de chauffer le matériau. Une thermo-résistance est donc nécessaire.
Le temps de réponse est de quelques minutes et le temps de désorption a tendance à être lent. Une dérive dans le temps (due à l'accumulation de gaz non désorbé) peut être observée après une trop forte exposition aux polluants. Il est donc important de prendre en compte une maintenance régulière, et il est conseillé d'utiliser ces capteurs en parallèle d'une méthode de référence.
La température et le taux d'humidité sont également des paramètres qui interfèrent.

9.6 Potentiomètres



Environnement enfumé sortie 3.3V



Environnement sain sortie 0V

Le potentiomètre représenté par le multimètre B, me permet de fixer un seuil dans mon montage, en entrant une tension sur la broche moins du comparateur.

Mon capteur étant fixé par son potentiomètre personnel a une tension de base en environnement sain, est représenté par le multimètre A.

Le multimètre C représente la tension de sortie du comparateur.

Après avoir fixé un seuil sur mon capteur à l'aide de son potentiomètre personnel, on peut fixer le seuil du comparateur avec le potentiomètre relié à la broche - du comparateur.

J'ai donc fixé mon seuil du comparateur à 2.5 V et le seuil de mon capteur à 1.15 V en environnement sain, on voit donc que la sortie est à 0 V.

Quand mon capteur détecte de la fumée, sa tension de sortie s'élève à 4,3 V et par la suite le comparateur ne sort rien sur sa sortie, le 3,3V prend donc le dessus sur la sortie, sa sortie passe à 3.3V, il y a un incendie.

Le potentiomètre permet donc de fixer un seuil à dépasser par le capteur, pour que le comparateur puisse sortir une tension ou non, à sa sortie.

9.7 Principe de fonctionnement des autres capteurs

9.7.1 Horloge quartz

Une horloge à quartz utilise un oscillateur à quartz.

Le quartz a la propriété d'osciller à une fréquence précise, lorsqu'il est stimulé électriquement (à l'aide d'une pile par exemple).

Cette stimulation électrique dans le quartz engendre des vibrations mécaniques, on obtient ainsi un oscillateur électrique. Par un calcul, on obtient l'unité de temps.

9.7.2 Capteur de température

Thermistance qui est basée sur la variation de la résistance électrique en fonction de la température.

Notre capteur est doté d'un CAN, il sort donc une valeur binaire à sa sortie qui transmet l'information de température au programme, qui la transforme en valeur décimale.

10 Partie du candidat JACQUES Rémi

10.1 Présentation du rôle du candidat dans ce projet

Pour les deux candidats en électronique et communication, la conception et la fabrication de la nouvelle carte électronique du point d'affichage a été demandée afin d'y intégrer les deux nouveaux capteurs qui sont pour le candidat M.ZEHANI un capteur de fumée et pour le candidat M.JACQUES un capteur de qualité de l'air. Il sera également demandé au candidat de choisir entre trois capteurs différents qui seront détaillés plus bas.

Pour cette première revue, mon but était de tester les trois capteurs et de les mettre en confrontation afin de choisir celui qui serait le mieux adapté pour l'intégration sur le *shield* de la carte Raspberry PI.

Ensuite il me sera demandé de créer un code en langage C qui communiquera avec le capteur. La saisie du schéma de routage se fera sur le logiciel KiCad en prenant soin de retirer une partie qui ne sera plus utilisée dans cette nouvelle version. Ensuite nous enverrons notre commande pour en effectuer la fabrication.

La dernière étape sera de souder tout les composants sur le nouveau *shield* et d'effectuer les tests finaux à l'aide de la Raspberry Pi.

Photographie d'une Raspberry Pi

Photographie du shield

10.2 Partie Physique

Durant cette partie du dossier, je vais éclaircir certains points de physique en ce qui concerne le fonctionnement ou la communication des éléments du projet.

10.2.1 Fonctionnement de l'horloge DS3231M

Le DS3231M est une horloge I2C à temps réel (Real Time Clock) peu coûteuse et extrêmement précise.

Elle fonctionne à l'aide d'un oscillateur à quartz, ce minéral possède la capacité d'osciller à une fréquence stable lorsqu'il est stimulé électriquement.

Les propriétés piézoélectriques remarquables du minéral de quartz permettent d'obtenir des fréquences d'oscillation très précises, qui en font un élément important en électronique numérique ainsi qu'en électronique analogique.

Un cristal de quartz

10.2.2 Fonctionnement du capteur de température DS18B20

Ce capteur de température fonctionne à l'aide d'une thermistance. Il communique la température en Celcius à l'aide d'un bus 1-wire. Il est capable de mesurer des températures allant de -55°C à +125°C.

La fonctionnalité principale de ce capteur c'est de faire du « direct-to-digital ». Sa sortie n'est qu'en numérique via le bus 1-wire. Il possède donc un Convertisseur Analogique Numérique (CAN).

Le capteur est une thermistance, c'est l'un des principaux capteurs de température. Il consiste en la variation de la résistance électrique en fonction de la montée ou de la baisse de température.

10.2.3 Fonctionnement du capteur de mouvement

Ce capteur possède 4 récepteurs pyroélectiques. La pyroélectricité est la propriété d'un matériau dans lequel un changement de température entraine une variation de polarisation électrique. Cette variation de polarisation crée une différence de potentiel temporaire, celle-ci disparaissant après le temps de relaxation diélectrique. Cette variation peut générer un courant électrique, ce qui rend ces matériaux utiles pour la détection de radiations.

10.2.4Étude du principe physique des capteurs

Dans leurs documentations, il est stipulé que les capteurs CCS811 et SGP30 fonctionnent avec des capteurs MOX (Metal OXide gas sensor ou capteur de gaz d'oxyde métallique) également appelés capteurs semi-conducteurs. Ce type de capteur est utilisé pour détecter les COV ou TVOC.

Ces capteurs contiennent une couche sensible semi-conductrice (oxyde métallique MOX) qui possède une conductivité dépendante de la composition de l'air qui l'entoure. Le signal obtenu correspond donc à une résistance variable en fonction de l'évolution de la conductivité.

Le matériau support de la réaction d'oxydo-réduction est un oxyde métallique semi-conducteur comme le Dioxyde d'étain (SnO2) ou l'oxyde de Zinc (ZnO) par exemple.

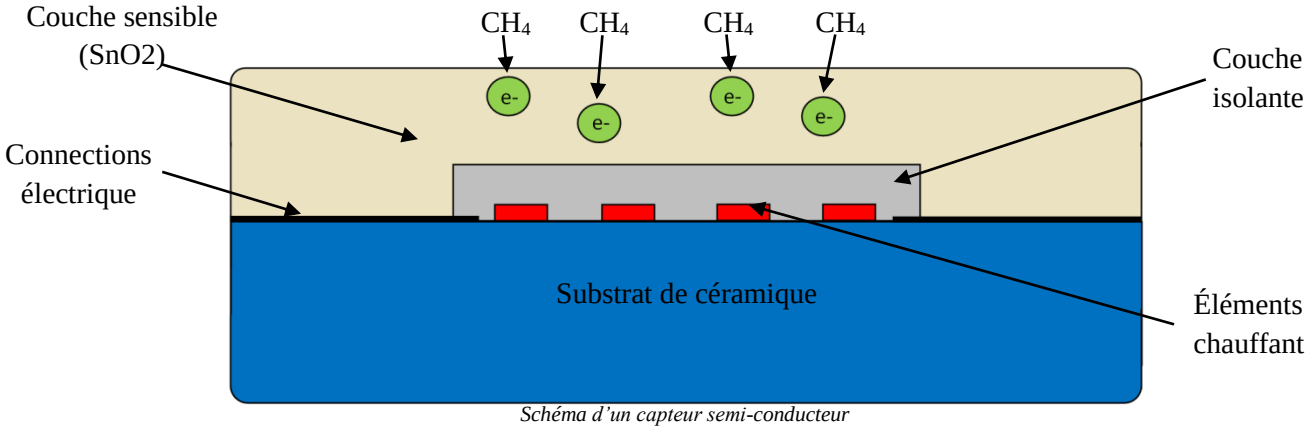


Schéma d'un capteur semi-conducteur

La valeur de l'humidité de l'air influence les capteurs semi-conducteurs en changeant le niveau de base de la résistance et la sensibilité. Il faut donc prendre en compte ce paramètre lors de l'utilisation des capteurs. Dans la plupart des cas, les caractéristiques sont assurées pour une température de 20 °C ainsi qu'un taux d'humidité suffisant.

Le processus général de détection peut être résumé ainsi : le gaz à détecter s'adsorbe à la surface de l'oxyde ou réagit avec des espèces préabsorbées (essentiellement les espèces oxygénées) sur l'oxyde, ceci entraîne une variation de la charge d'espace de la couche de surface, qui est convertie en variation de résistance électrique du matériau.

A propos de la détection du CO2 :

Les capteurs MOX ne permettent pas de mesurer un taux de CO2. La documentation du CCS811 stipule :

« Le CCS811 prend en charge des algorithmes intelligents pour traiter les mesures brutes pour produire une valeur de TVOC ou un équivalent CO2 (eCO2), car la principale cause de TVOC est d'origine humaine. »

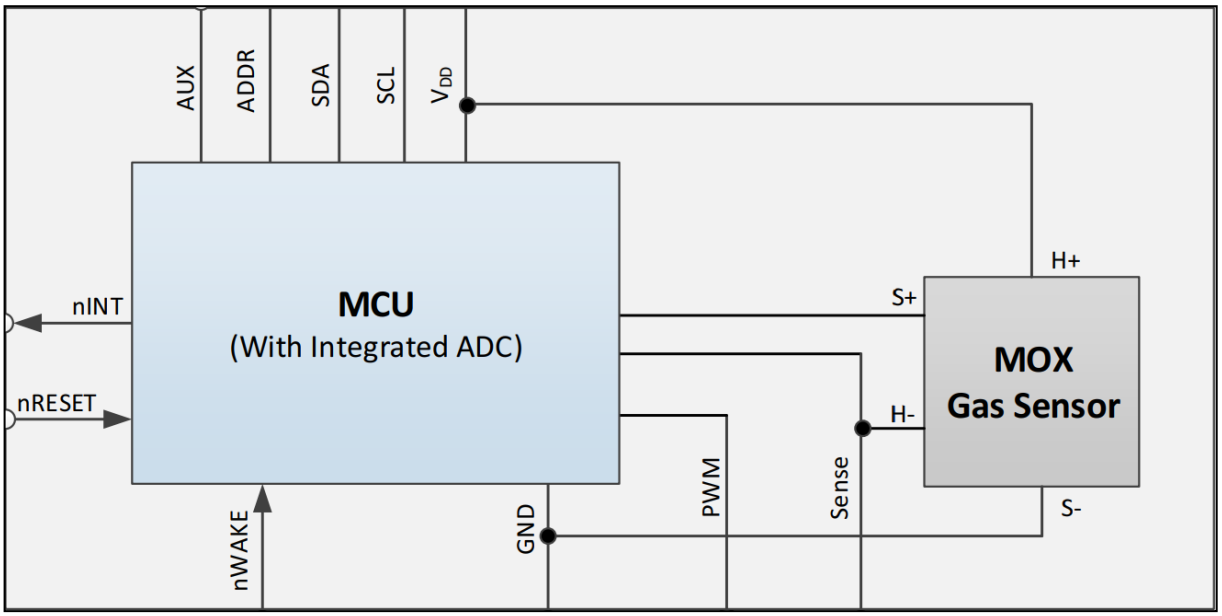


Diagramme de Block du capteur CCS811

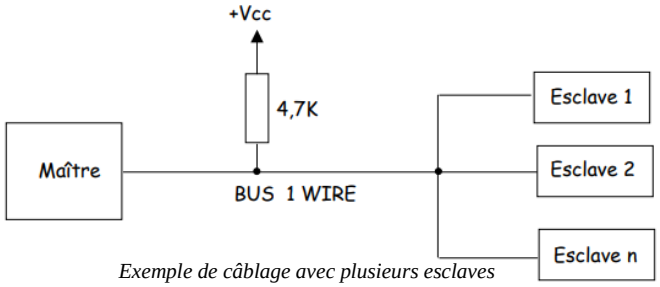
Le CCS811 se compose d'un capteur MOX qui va relever les valeurs de TVOC puis les envoyer vers le microcontrôleur (microcontroller unit / MCU). Qui va effectuer un calcul de l'équivalent CO2 (eCO2). Puis qui va convertir toutes ces données grâce à son convertisseur analogique numérique (Analog to Digital Converter / ADC) et qui va les envoyer via le bus I2C vers notre Raspberry Pi.

Le capteur SGP30 possède un BDD du même type hors mis qu'il est d'après sa documentation le premier capteur de gaz à oxyde métallique comportant plusieurs éléments de détection sur une seule puce, le SGP30 fournit des informations plus détaillées sur la qualité de l'air qui sont des informations sur le taux d'éthanol et de dihydrogène.

10.2.5Le bus 1-Wire

1-Wire (aussi connu sous le nom de bus Dallas ou OneWire) est un bus conçu par Dallas Semiconductor qui permet de connecter (en série, parallèle ou en étoile) des composants avec seulement deux fils (un fil de données et un fil de masse).
Il est généralement utilisé en domotique pour des thermomètres.

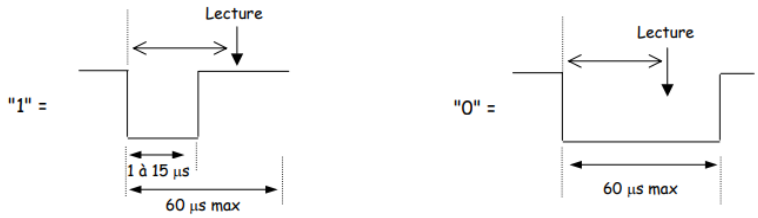
Le fil unique du bus doit être tiré au +Vcc par une résistance de 4,7KΩ. L'état repos du bus est donc un état haut.



Exemple de câblage avec plusieurs esclaves

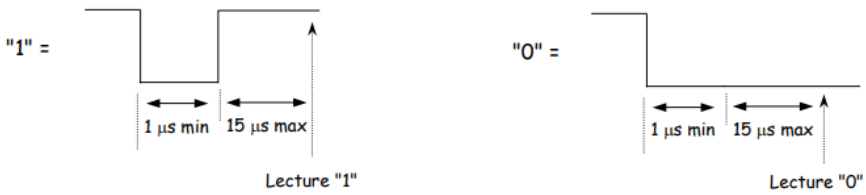
Emission d'un bit du maître vers l'esclave :

Le maître force le bus à "0" pendant 1 à 15 µs. L'esclave va lire le bus entre 15 et 45 µs après le front descendant (valeur typique 30 µs). Si on veut émettre un "1", il faut repasser le bus à "1" immédiatement, et ne plus rien faire jusqu'à t = 60 µs. Pour émettre un "0" il faut laisser le bus à "0" jusqu'à t = 60 µs, puis repasser le bus à "1". La durée du bit est donc de 60 µs, ce qui donne un débit de 16 kbits/sec.



Réception d'un bit par le maître :

Le maître force le bus à "0" pendant au moins 1 µs. Si l'esclave veut émettre un "1", il laisse le bus libre donc tiré à "1". Pour émettre un "0", l'esclave doit tirer le bus à "0" pendant 15 µs au minimum. Le maître devra donc dans tous les cas lire le bus 15 µs maximum après avoir tiré le bus à "0" pendant 1 µs. L'état du bus donnera alors le bit transmis par l'esclave.



10.2.6Le bus I2C

I2C (signifie : Inter-Integrated Circuit, en anglais) est un bus informatique conçu par Philips permettant de transmettre des informations de façon asynchrone. Le protocole de la liaison est du type maître/esclave.

Il existe d'innombrables périphériques exploitant ce bus, il est même implémentable par logiciel dans n'importe quel microcontrôleur.

Il permet de faire communiquer entre eux des composants grâce à trois fils :

- Le SDA pour Signal de données
- Le SCL pour Signal d'horloge
- La masse

Il y'a une adresse unique pour chaque périphérique. C'est un bus bi-directionnel qui possède différentes vitesses de transmission :

- 100kbps (Standard mode)
- 400kbps (fast mode)
- 3.2 Mbps (high-speed mode)

Le bus doit être au repos avant la prise de contrôle donc on a SDA et SCL à 1.

- Le bit de start est défini par un passage à 0 pour le SDA tandis que le SCL reste à 1.
- Le bit de stop est défini par un passage à 1 pour le SDA tandis que le SCL reste à 1.

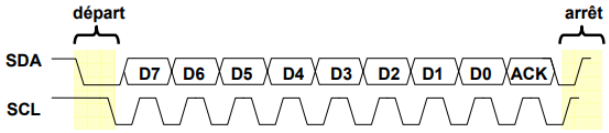


Schéma démontrant le bit de start et de stop

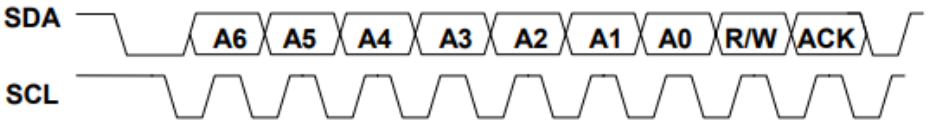
Pour transmettre un octet, le maître transmet le bit de poids fort (D7) sur SDA et il valide la donnée en appliquant un niveau 1 sur SCL. Lorsque SCL retombe à 0, il poursuit avec D6, etc. jusqu'à ce que l'octet complet soit transmis.

Pour le ACK (Acknowledge / Aquitement), le maître libère la ligne SDA, l'esclave force la ligne SDA au niveau bas (en gras sur le schéma), le maître envoie une impultion sur l'horloge SCL et lorsque l'impulsion retombe à zéro, l'esclave libère SDA.

Schéma démontrant le fonctionnement du bit ACK

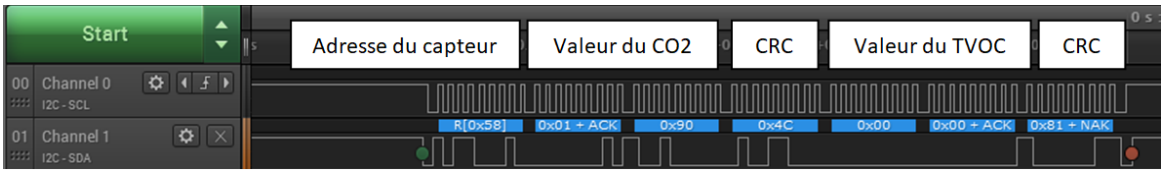
Transmission d'une Adresse :

Nos capteurs ont une adresse codée sur 7 bits. Nous avons ici les 7 bits d'adresse A6 à A0 suivi du bit R/W pour Read/Write qui détermine si le maître veut lire ou écrire (W=0 et R = 1):



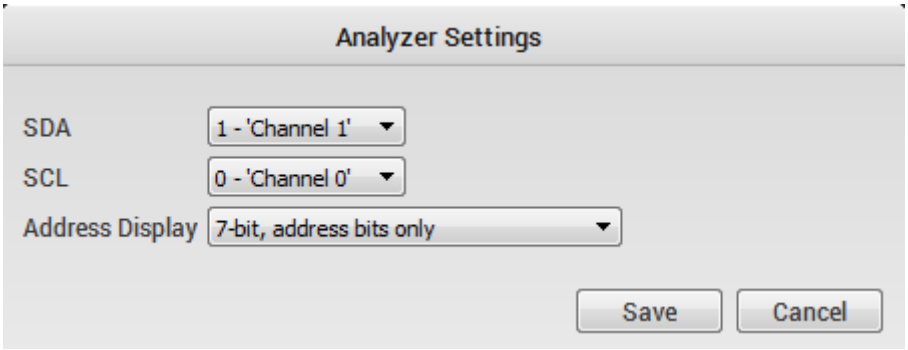
Exemple d'un octet d'adresse

De nombreuses captures de trames I2C ont étés détaillées dans la partie électronique de mon dossier. En voici une ou nous pouvons constater les bits de start et de stop marqués avec une pastille verte et rouge.



Exemple de trame I2C

On notera également que pour éviter de compter le bit R/W, nous avons sélectionnés de ne garder que l'adresse sur 7 bits, le huitième étant le bit R/W.



Menu de sélection de l'analyseur de trame

10.3 Présentation des capteurs de qualité de l'air

10.3.1 Généralités

Les trois capteurs proposés ont des similitudes, elles seront citées ici :

Ils communiquent à l'aide du bus I2C et fonctionnent avec une tension de 3.3V fournie par la Raspberry Pi. Sauf le SGP30 qui fonctionne en 1.8V mais qui possède un *level-shifter* pour le 3.3V.

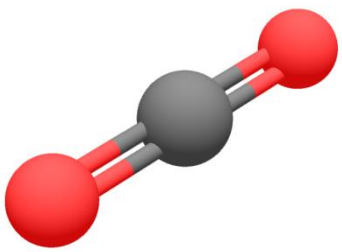
Leurs différences de taille est négligeable pour le choix d'un capteur sur les trois proposés.

10.3.2 Informations sur les valeurs relevées

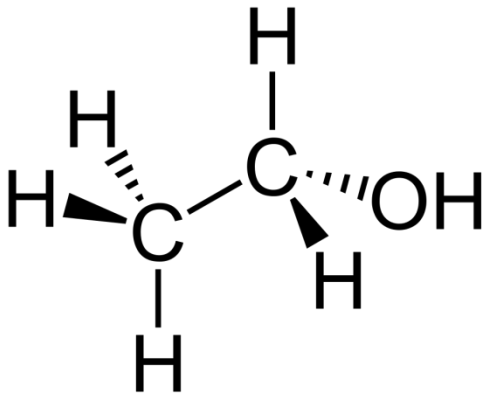
Le dihydrogène et l'éthanol sont des gaz classé comme étant extrêmement inflammable.

Dans le cas du CO2, à partir d'une certaine concentration dans l'air, ce gaz s'avère dangereux voire mortel à cause du risque d'asphyxie ou d'acidose, bien que le CO2 ne soit pas chimiquement toxique. En France, le taux de CO2 dans l'air intérieur des bâtiments est habituellement compris entre 350 et 2 500 ppm environ. Séjourner toute la journée dans un air dont le taux de CO2 atteint ou dépasse 600 ppm dégrade nos capacités cognitives (penser, raisonner, se souvenir, décider).

Le TVOC ou COVT est l'acronyme de Composés Organiques Volatils Totaux, c'est-à-dire la quantité totale de gaz émis ayant des effets à court ou à long terme sur la santé. Les COVT peuvent causer de graves problèmes de santé à court et à long termes. Les effets sur la santé varient, allant de petites irritations au niveau des yeux, du nez et de la gorge aux lésions ou cancers provoqués au niveau du foie et des reins, selon le niveau d'exposition.



Représentation en 3D du CO2

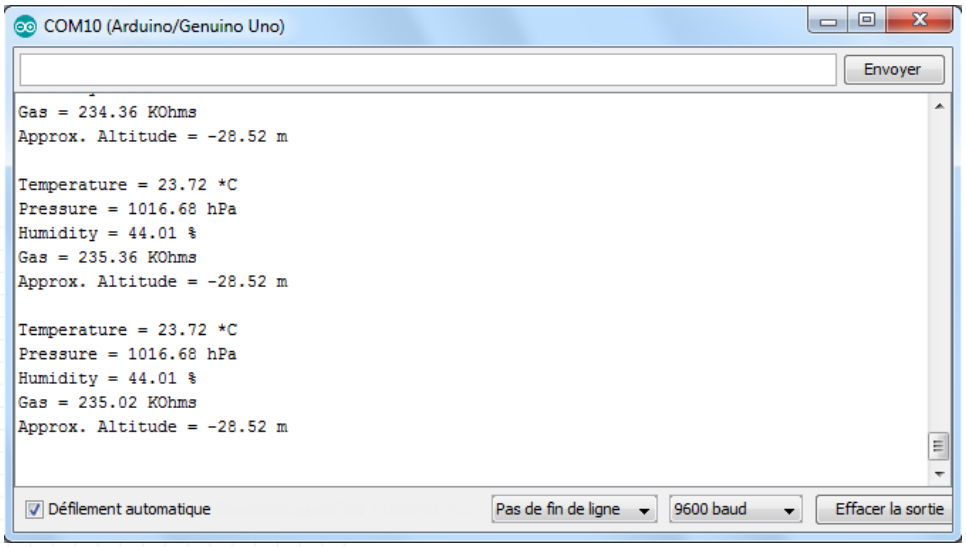


Structure de l'éthanol

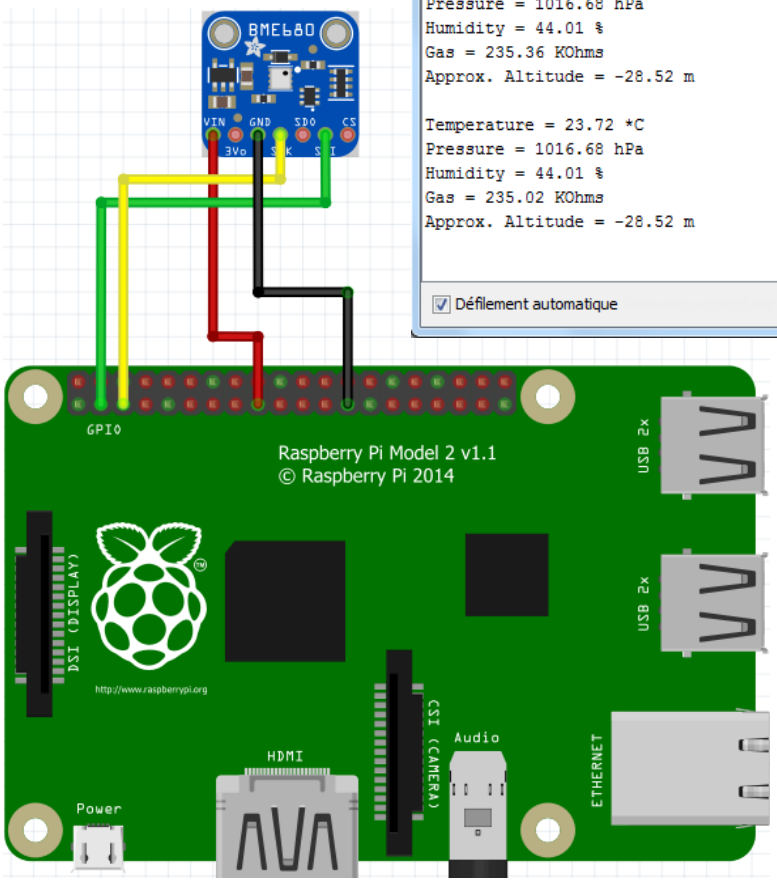
10.3.3BME680

Le BME680 est fabriqué par BOSH, sa taille est de 3.0 mm x 3.0 mm x 0.93 mm. Ce capteur nous renvoie des informations sur la température, la pression, l'humidité, le gaz et fait une approximation de l'altitude.

Pour le câblage, on connecte le 3.3V au Vin, la masse, le SDA de la Rpi/Arduino au SCI et le SCL.



Valeurs envoyées par le capteur



Câblage du breakout sur une Raspberry Pi

10.3.3.1 Code Arduino du BME680

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"

#define BME_SCK 13
#define BME_MISO 12
#define BME_MOSI 11
#define BME_CS 10
#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME680 bme; // I2C

void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println(F("BME680 test"));

  if (!bme.begin()) {
    Serial.println("Could not find a valid BME680 sensor, check wiring!");
    while (1);
  }

  // Set up oversampling and filter initialization
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150); // 320*C for 150 ms
}

void loop() {
  if (! bme.performReading()) {
    Serial.println("Failed to perform reading :(");
    return;
  }
  Serial.print("Temperature = ");
  Serial.print(bme.temperature);
  Serial.println(" *C");

  Serial.print("Pressure = ");
  Serial.print(bme.pressure / 100.0);
  Serial.println(" hPa");

  Serial.print("Humidity = ");
  Serial.print(bme.humidity);
  Serial.println(" %");

  Serial.print("Gas = ");
  Serial.print(bme.gas_resistance / 1000.0);
  Serial.println(" KOhms");

  Serial.print("Approx. Altitude = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");

  Serial.println();
  delay(2000);
}
```

Configuration du suréchantillonnage et initialisation du filtre

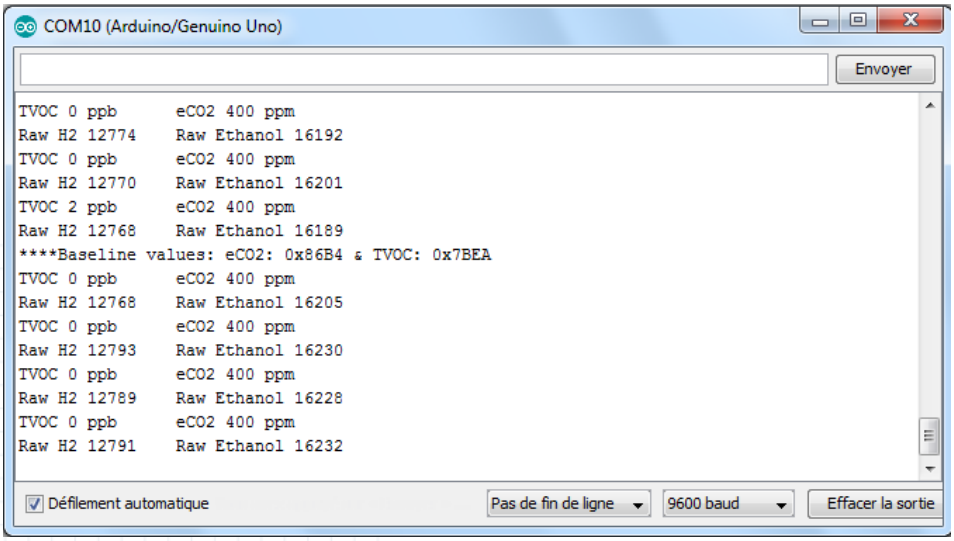
Affichage des valeurs

Puis délais de 2 secondes

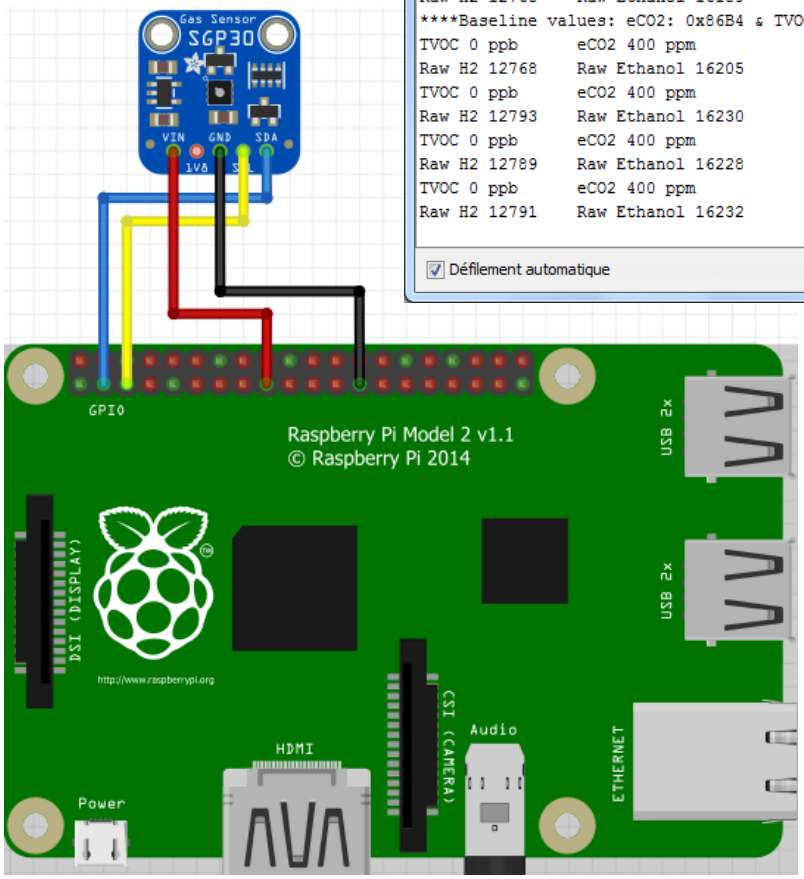
10.3.4SPG30

Le SPG30 est fabriqué par SENSIRION, sa taille est de 2.45 mm x 2.45 mm x 0.9 mm. Ce capteur nous renvoie des informations sur le CO2, le TVOC et le taux de dihydrogène (H2) et d'éthanol dans l'air.

Pour le câblage, on connecte le 3.3V au Vin, la masse, le SDA et le SCL.



Valeurs envoyées par le capteur

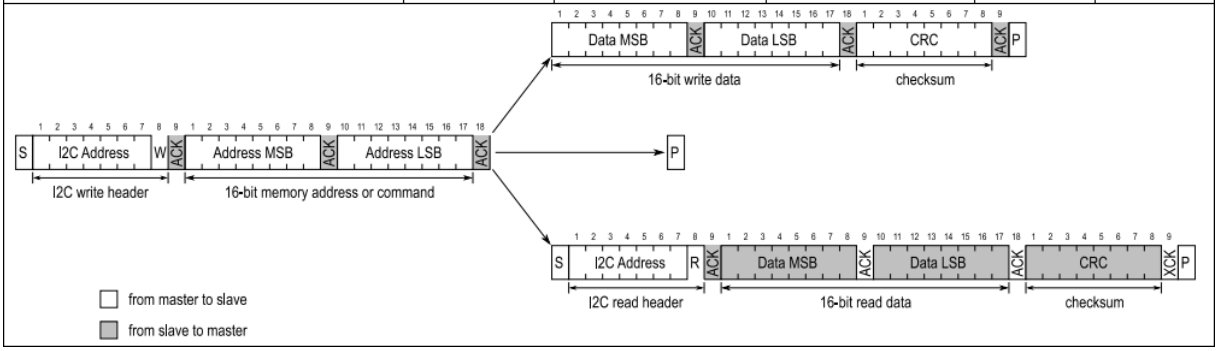


Câblage du breakout sur une Raspberry Pi

10.3.4.1 Décodage de trames I2C avec Saleae

Pour recevoir les quatre valeurs du SGP30, deux commandes sont appelées. Pour obtenir les valeurs du CO2 et du TVOC, la commande sgp30_measure_iaq est envoyée au capteur avec un code en hexadécimal de 0x2008. Pour obtenir les valeurs brutes de dihydrogène et d'éthanol, la commande sgp30_measure_raw sera envoyée au capteur avec un code en hexadécimal de 0x2050.

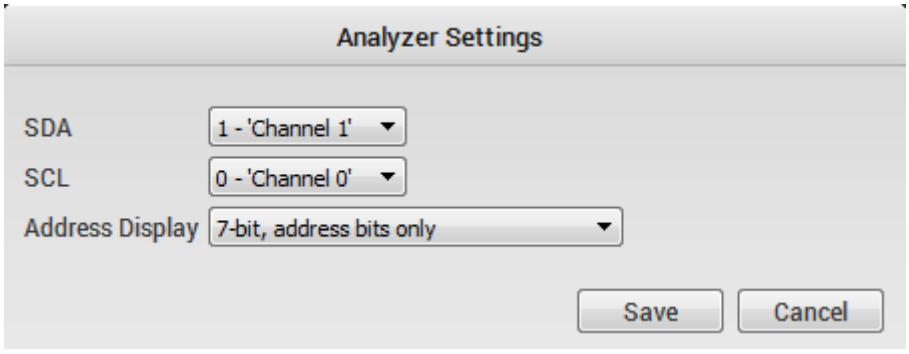
Feature Set	**0x0022**				
Command	**Hex. Code**	**Parameter length including CRC [bytes]**	**Response length including CRC [bytes]**	**Measurement duration [ms]**	
				Typ.	**Max.**
sgp30_iaq_init	0x2003	-	-	2	10
sgp30_measure_iaq	0x2008	-	6	10	12
sgp30_get_iaq_baseline	0x2015	-	6	1	10
sgp30_set_iaq_baseline	0x201e	6	-	1	10
sgp30_set_absolute_humidity	0x2061	3	-	1	10
sgp30_measure_test[10]	0x2032	-	3	200	220
sgp30_get_feature_set	0x202f	-	3	1	10
sgp30_measure_raw	0x2050	-	6	20	25
sgp30_get_tvoc_inceptive_baseline	0x20b3	-	3	1	10
sgp30_set_tvoc_baseline	0x2077	3	-	1	10



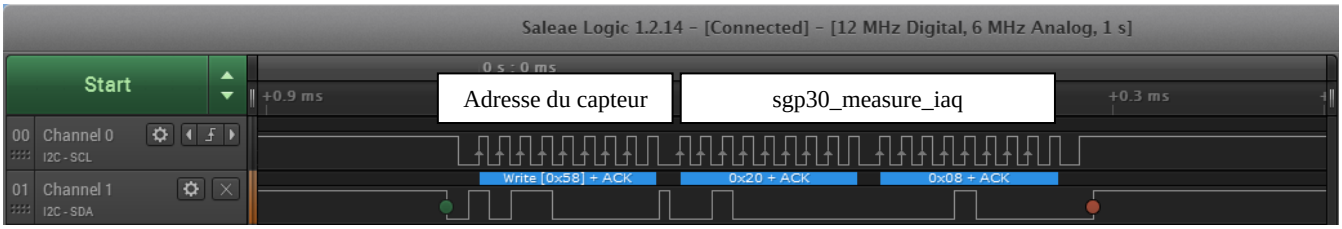
Extrait de Doc. Sur les différentes commandes pouvant être envoyées au capteur

Le « write » sera codé sur 3 octets. Un pour l'adresse du capteur et deux pour envoyer la commande.

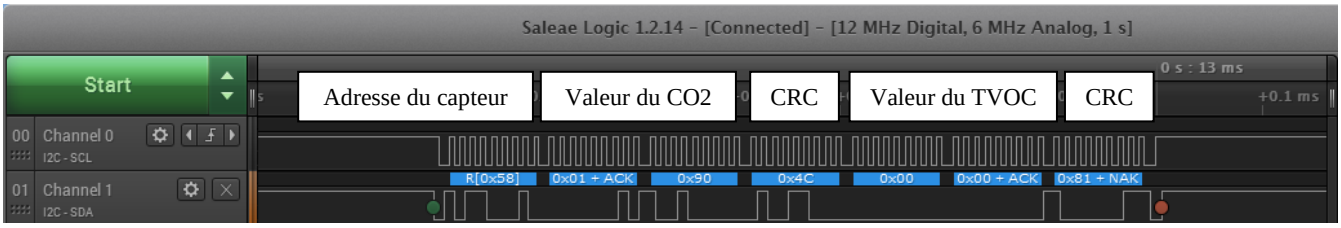
Le « read » sera codé sur 7 octets. Un pour l'adresse puis deux pour la mesure du CO2 suivi d'un octet pour le CRC. Puis 2 octets pour la valeur du TVOC avec un dernier octet pour le CRC.



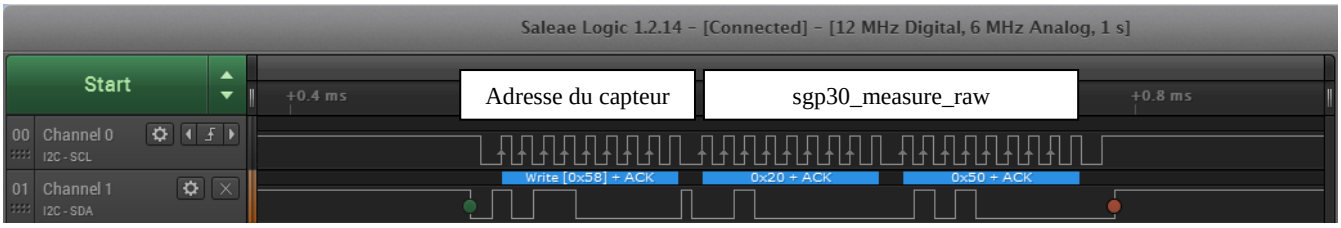
Paramètres de toutes les mesures effectuées



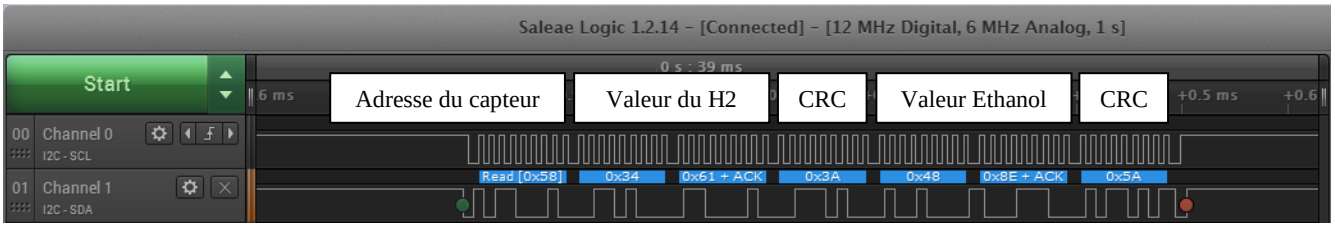
Trame envoyée au capteur



Trame envoyée à l'Arduino



Trame envoyée au capteur



Trame envoyée à l'Arduino

10.3.4.2 Code Arduino du SGP30

```
#include <Wire.h>
#include "Adafruit_SGP30.h"

Adafruit_SGP30 sgp;

uint32_t getAbsoluteHumidity(float temperature, float humidity) {

    const float absoluteHumidity = 216.7f * ((humidity / 100.0f) * 6.112f
* exp((17.62f * temperature) / (243.12f + temperature)) / (273.15f +
temperature)); // [g/m^3]
    const uint32_t absoluteHumidityScaled = static_cast<uint32_t>(1000.0f
* absoluteHumidity); // [mg/m^3]
    return absoluteHumidityScaled;
}

void setup() {
  Serial.begin(9600);
  Serial.println("SGP30 test");

  if (! sgp.begin()){
    Serial.println("Sensor not found :(");
    while (1);
  }
  Serial.print("Found SGP30 serial #");
  Serial.print(sgp.serialnumber[0], HEX);
  Serial.print(sgp.serialnumber[1], HEX);
  Serial.println(sgp.serialnumber[2], HEX);
}

int counter = 0;
```

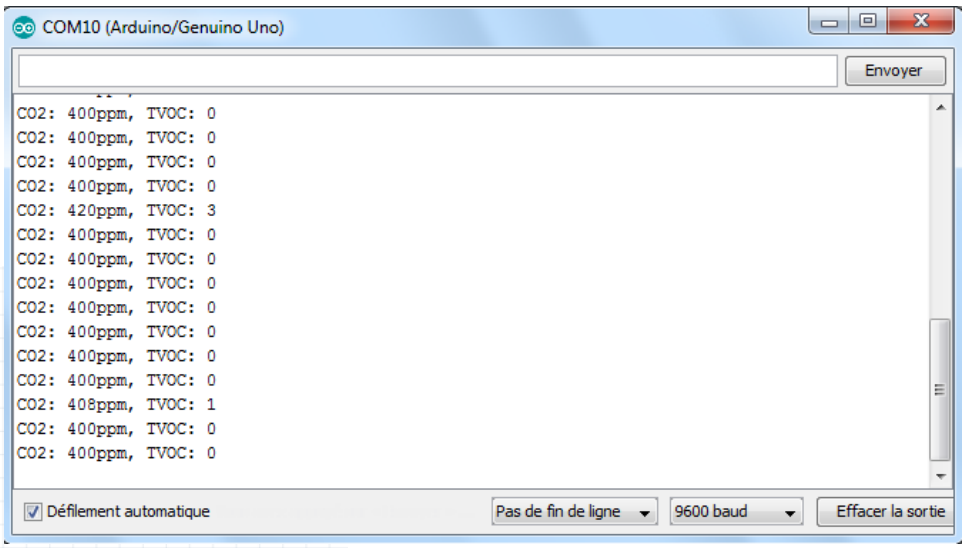
Appel d'une fonction permettant d'obtenir la valeur d'humidité absolue

```
void loop() {

  if (! sgp.IAQmeasure()) {
    Serial.println("Measurement failed");
    return;
  }
  Serial.print("TVOC "); Serial.print(sgp.TVOC
  Serial.print("eCO2 "); Serial.print(sgp.eCO2

  if (! sgp.IAQmeasureRaw()) {
    Serial.println("Raw Measurement failed");
    return;
  }
  Serial.print("Raw H2 "); Serial.print(sgp.ra
  Serial.print("Raw Ethanol "); Serial.print(sgp.rawEthanol);
Serial.println("");

  delay(1000);

  counter++;
  if (counter == 30) {
    counter = 0;

    uint16_t TVOC_base, eCO2_base;
    if (! sgp.getIAQBaseline(&eCO2_base, &TVOC_base)) {
      Serial.println("Failed to get baseline readings");
      return;
    }
    Serial.print("****Baseline values: eCO2: 0x"); Serial.print(eCO2_base,
HEX);
    Serial.print(" & TVOC: 0x"); Serial.println(TVOC_base, HEX);
  }
}
```

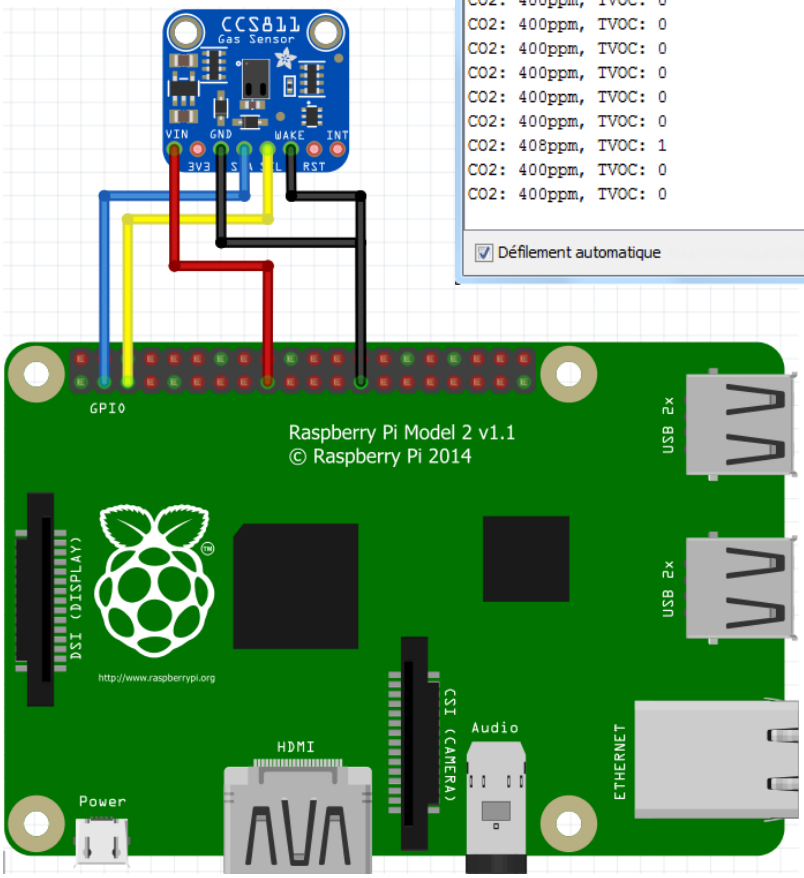
10.3.5 CCS811

Le SPG30 est fabriqué par AM5, sa taille est de 2.7 mm x 4.0 mm x 1.1 mm. Ce capteur nous renvoie des informations sur le CO2 et le TVOC.

Pour le câblage, on connecte le 3.3V au Vin, la masse, le SDA et le SCL. La broche Wake doit être reliée à la masse.



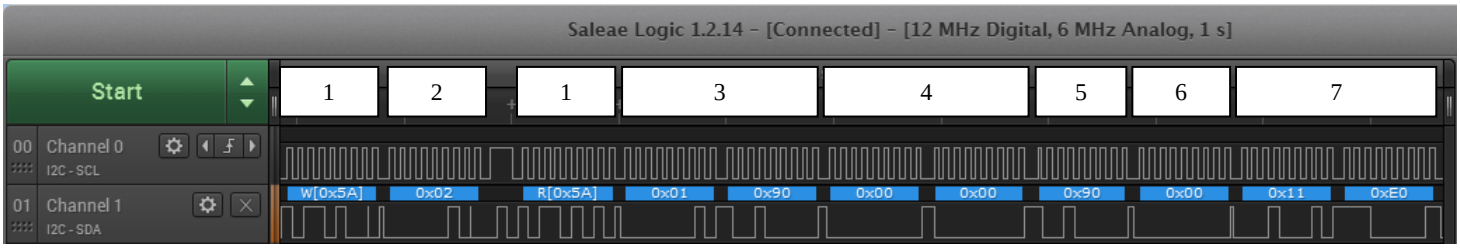
Valeurs envoyées par le capteur



Câblage du breakout sur une Raspberry Pi

10.3.5.1 Décodage de trames I2C avec Saleae

Pour recevoir les valeurs du CCS811, une commande est appelée. Pour obtenir les valeurs du CO2 et du TVOC, la commande ALG_RESULT_DATA est envoyée au capteur avec un code en hexadécimal de 0x02.



Trame envoyée au capteur avec réponse vers l'Arduino

Légende :

1 : Adresse du capteur

2 : ALG_RESULT_DATA

3 : Valeur du CO2, 0x0190

4 : Valeur du TVOC, 0x0000

5 : Status, 0x90

6 : Error ID, 0x00

7 : RAW DATA, 0x11E0

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6 & 7
eCO_2 High Byte	eCO_2 Low Byte	eTVOC High Byte	eTVOC Low Byte	STATUS	ERROR_ID	See RAW_DATA

Spécifications de la Doc. sur le contenu d'une trame

La signification du bit de status en hexadécimal est renseignée dans la documentation.

7	6	5	4	3	2	1	0
FW_MODE	APP_ERASE	APP_VERIFY	APP_VALID	DATA_READY	-		ERROR

Bit(s)	Field	Description
7	FW_MODE	0: Firmware is in boot mode, this allows new firmware to be loaded 1: Firmware is in application mode. CCS811 is ready to take ADC measurements
6	APP_ERASE	Boot Mode only. 0: No erase completed 1: Application erase operation completed successfully (flag is cleared by APP_DATA and also by APP_START, SW_RESET, nRESET and APP_VERIFY) After issuing the ERASE command the application software must wait 500ms before issuing any transactions to the CCS811 over the I^2C interface.
5	APP_VERIFY	Boot Mode only. 0: No verify completed 1: Application verify operation completed successfully (flag is cleared by APP_START, SW_RESET and nRESET) After issuing a VERIFY command the application software must wait 70ms before issuing any transactions to CCS811 over the I^2C interface
4	APP_VALID	0: No application firmware loaded 1: Valid application firmware loaded
3	DATA_READY	0: No new data samples are ready 1: A new data sample is ready in ALG_RESULT_DATA, this bit is cleared when ALG_RESULT_DATA is read on the I^2C interface
2:1	-	Reserved
0	ERROR	This bit is cleared by reading ERROR_ID (it is not sufficient to read the ERROR field of ALG_RESULT_DATA and STATUS) 0: No error has occurred 1: There is an error on the I^2C or sensor, the ERROR_ID register (0xE0) contains the error source

Spécifications de la Doc. sur l'octet de Status

1	0	0	1	0	0	0	0

0x90 en Binaire

Donc ici, le capteur est prêt à effectuer des mesures, une application valide de *firmware* est chargée, il n'y a pas de nouvelles données et il n'y a pas d'erreurs rencontrées.

RAW_DATA	R	2 bytes	Raw ADC data values for resistance and current source used.

Spécifications de la Doc. sur RAW_DATA

RAW_DATA donne des données ADC brutes pour la résistance et la source de courant utilisée.

10.3.5.2 Code Arduino du CCS811

```
#include "Adafruit_CCS811.h"

Adafruit_CCS811 ccs;

void setup() {
  Serial.begin(9600);

  Serial.println("CCS811 test");

  if(!ccs.begin()){
    Serial.println("Failed to start sensor! Please check your wiring.");
    while(1);
  }

  // Wait for the sensor to be ready
  while(!ccs.available());
}

void loop() {
  if(ccs.available()){
    if(!ccs.readData()){
      Serial.print("CO2: ");
      Serial.print(ccs.geteCO2());
      Serial.print("ppm, TVOC: ");
      Serial.print(ccs.getTVOC());
    }
    else{
      Serial.println("ERROR!");
      while(1);
    }
  }
  delay(500);
}
```

10.3.6 Comparatifs

	BME680	SGP30	CSS811
Prix (*10)	9,41 €	9,65 €	9,15 €
Mesures	TEMPERATURE, PRESSION, HUMIDITE, GAZ, APPROX. ALTITUDE	CO2, TVOC, H2 et ETHANOL	CO2 ET TVOC
Communication	I2C	I2C	I2C
Taille	3.0 x 3.0 x 0.93	2.45x 2.45 x 0.9	2.7 x 4.0 x 1.1
Tension	3.3V	1.8V	3.3V
Complexité du code (Arduino)			

Tableau comparatif des trois capteurs

Sur le tableau, les avantages sont en vert et les désavantages sont en rouge. Les trois capteurs se ressemblent. Seul le BME680 se démarque avec des valeurs renvoyées n'ayant rien à voir avec le taux de CO2 ou le TVOC. La différence de prix est minime et ne peux pas, pour moi être un facteur de choix par rapport au besoin (une dizaine de capteurs répartis sur le lycée). Nous ne sommes pas ici en situation de production de masse ou dans ce cas ces quelques centimes d'euros auraient fait une grosse différence sur des milliers de capteurs intégrés à des cartes puis vendus. Les capteurs communiquent tous via le protocole I2C donc le bus de communication n'est pas non plus un critère de choix. Leurs tailles, très réduites ne posent de problème pour aucun des trois capteurs. En voulant pousser la comparaison, j'ai intégré au tableau la longueur du code Arduino test qui permet de les contrôler ; étant donné que les codes créés en langage C seront sensiblement similaires et simples en excluant le BME680 qui possède un code un peu plus complexe. Les codes se valent, ils sont parsemés de « *Serial.print* » ce qui rend une impression de code long alors que pour le CCS811 par exemple, seules quelques commandes sont envoyées au capteur. Il suffit de survoler le code du CSS811 pour le comprendre. L'intérêt de posséder un capteur ayant pour avantage de détecter des gaz considérés pour leurs compositions inflammables n'a d'après mes recherches pas vraiment d'intérêt étant donné qu'un détecteur de fumée sera intégré à cette carte par un autre candidat. Les valeurs récupérées par l'un des trois capteurs seront affichées sur le point d'affichage. Pour moi, les valeurs de CO2 et de TVOC suffisent à l'affichage d'informations de qualité de l'air. La consommation de ces capteurs n'est pas à prendre en compte étant donné que les capteurs ne fonctionnent pas sous batterie, ils sont peu énergivores. Le plus gros défaut du SGP30 est qu'il doit fonctionner, contrairement aux deux autres capteurs en 1.8V ce qui implique la présence d'un *level-shifter* sur le *shield*.

10.4 Tâches effectuées

10.4.1 Remarques

Mon travail jusqu'à cette première revue aura été de modifier un diagramme de block initialement créé par le candidat M.GROSSET, de créer des diagrammes de block internes et également de créer le diagramme de Gantt prévisionnel.

En rapport avec l'intégration d'un capteur, j'ai eu à me renseigner sur les documentations fournies par les enseignants, j'ai également eu à créer des schémas de câblage pour effectuer mes tests à l'aide du logiciel Fritzing.

Une fois les capteurs branchés, l'utilisation d'un code en langage C était nécessaire à leurs fonctionnements. Afin de gagner du temps, j'ai cherché un code test sur le net afin de tester rapidement les trois capteurs pour me renseigner sur leurs fonctionnements et sur les valeurs qu'ils reverraient. N'ayant pas réussi à mètre la main sur un code en langage C adapté à ces capteurs, l'utilisation d'une carte Arduino Uno était intéressante afin de les tester rapidement. Pour se faire, dans le logiciel Arduino, il m'a fallu ajouter la librairie du capteur en question et un code appelé test était fourni.

La création du logo pour ce projet s'est faite lors d'une petite réunion, après avoir donnés nos avis et nos idées, nous nous sommes rapidement mis d'accord sur son design. J'ai eu l'occasion de réaliser ce logo en dehors des cours sur le logiciel Paint.net afin de ne pas empiéter sur mes heures de projet en classe.

J'ai eu à souder des pins aux *breakouts* également.

J'ai terminé cette première session en effectuant des comparatifs entre les capteurs et en interprétant les trames I2C.

10.4.2 Diagrammes de Gantt

Projet	Lun 06/01/20	Jeu 14/05/20			142 hr
Specifications générales	**Lun 06/01/20**	**Lun 27/01/20**			**33 hr**
Création/modification du diagrame de Gantt	Lun 06/01/20	Jeu 09/01/20		EC1	10 hr
Convenir d'une charte graphique pour le dossier et le Diaporama	Lun 13/01/20	Lun 13/01/20	12	EC1	1 hr
Création/modification du BDD et IBD	Lun 13/01/20	Jeu 16/01/20	13	EC1	6 hr
Analyse du schéma structurel	Jeu 16/01/20	Lun 27/01/20	14	EC1	16 hr
Essais et validation des structures (prototypage rapide)	**Mar 28/01/20**	**Jeu 06/02/20**			**16 hr**
Comparation des documentations des capteurs	Mar 28/01/20	Jeu 30/01/20	11	EC1	4 hr
Calculs de durée de la pile	Jeu 30/01/20	Jeu 30/01/20	17	EC1	2 hr
Essais comparatifs sur les capteurs de qualité de l'air	Jeu 30/01/20	Mar 04/02/20	18	EC1	6 hr
Choix d'un capteur parmis les trois proposés	Mar 04/02/20	Mar 04/02/20	19	EC1	1 hr
Illustration des essais de câblage rapide avec un shcéma Fritzing	Jeu 06/02/20	Jeu 06/02/20	20	EC1	3 hr
Prototypage rapide et routage	**Lun 09/03/20**	**Lun 30/03/20**			**31 hr**
Saisie complète du schéma de la carte sur KiCAD. Le second capteur sera également intégré	Lun 09/03/20	Jeu 26/03/20	16	EC1	30 hr
Dresser une liste des composants avec les références et distributeurs	Lun 30/03/20	Lun 30/03/20	23	EC1	1 hr
Fabrication et essais	**Lun 30/03/20**	**Lun 06/04/20**			**11 hr**
Envoi du fichier KiCAD pour fabrication	Lun 30/03/20	Lun 30/03/20	22	EC1	1 hr
Soudage des composants sur la carte	Lun 30/03/20	Jeu 02/04/20	26	EC1	6 hr
Tests de la carte	Jeu 02/04/20	Lun 06/04/20	27	EC1	4 hr

Diagramme de Gantt prévisionnel du projet complet

Tâches effectuées jusqu'à revue 1	35 hr
Diagramme de Gantt	2 hr
Etude et prise en main du projet	1 hr
Réunion de lancement	1 hr
Diagramme de Block et IBD	9 hr
Analyse schéma structurel	1 hr
Etude des capteurs	1 hr
Création table des matières du dossi	1 hr
Fritzing schéma de câblage	1 hr
Recherge de code pour test des capteurs	1 hr
Soudure des capteurs + recherches sur leurs fonctionnements	1 hr
Modification d'un code en C pour I2C	1 hr
Communiation avec les capteurs en I2C via Cmd	1 hr
Tests pour le fonctionnement des capteurs en I2C	8 hr
Comparatifs relatifs aux capteurs	2 hr
Création de la revue de projet	1 hr
Interprétations de trames I2C	3 hr

Diagramme de Gantt réel jusqu'à revue1

Les deux diagrammes de Gantt sont similaires, le diagramme réel est plus précis. La prévision de mes tâches avait un peu trop de marge. Sauf pour les diagrammes de Block et de Block Interne ou la manipulation du logiciel MagicDraw s'est avérée complexe au début ce qui a fait que j'ai du passer plus de temps à travailler dessus. Les comparaisons des documentations des capteurs s'est faite tout au long de mes tâches. Le temps attribué aux tests des capteurs était presque égal aux prévisions (6 heures prévues contre 8 heures dans la réalité). La réalisation des schémas de câblage avec le logiciel Fritzing s'est avérée plus rapide que prévue avec une heure d'avance comparé aux prévisions. Cela est dû à la facilité que j'ai eu à trouver les fichiers relatifs aux capteurs, mais le travail effectué en travaux pratiques avant le projet m'a grandement aidé à me faciliter l'utilisation de ce logiciel. Au final mon objectif pour cette première revue est atteint. Les parties *spécifications générales* et *essais et validation des structures (prototypage rapide)* sont complétées. Les dernières séances ont étés consacrées à la création du dossier et à l'interprétation de trames I2C ce qui n'avait pas été prévu lors de la création du diagramme de Gantt prévisionnel.

10.5 **Tests des capteurs**

Après avoir choisis le capteur CCS811 et ayant un routage à faire, pendant le contrôle de la première revue de projet, mes professeurs et moi-même nous sommes posés la question : est t-il préférable d'intégrer le capteur au *hat* de la RPI ou de le déporter plus loin comme le capteur de température afin d'avoir une meilleure « précision » sur la qualité de l'air ?

Nous avions donc imaginé l'expérience de mettre en œuvre deux captures simultanément dans un couloir du lycée : un capteur serait positionné derrière une TV comme si elle était dans le *hat* de la RPI, et l'autre déporté plus loin dans le couloir.

Le but aurait été d'enregistrer les valeurs et de les mettre dans un tableur comme exel et de comparer deux graphiques afin de déterminer qu'elle solution serait la meilleure.

Il me fallait alors transposer le programme Arduino qui communiquait avec le capteur en langage C pour le Raspberry Pi. C'était la méthode la plus rapide pour créer un programme en C et la plus sûre étant donné que branché à une carte Arduino, le capteur fonctionne correctement.

J'ai donc effectué une capture de trame I2C sur 10 secondes à l'aide d'un analyseur logique nommé Saleae afin de voir comment se déroule l'initialisation du capteur et pour recréer toutes les commandes I2C dans mon programme en C qui devait fonctionner en mode console de la RPI pour aller au plus simple.

Après avoir créé un programme entier en langage C basé sur le code Arduino, le capteur répond mis à part que souvent, des valeurs erronées apparaissent en passant par exemple de 400ppm à 32 000ppm (qui est le seuil de CO2 correspondant à un risque asphyxie chez l'homme). Mon professeur avait émis l'hypothèse qu'il y ait possiblement un problème de vitesse de transmission.

Ensuite, du au confinement pour éviter la propagation du COVID-19, j'ai du continuer de modifier mon code en C afin de trouver une potentielle erreur dans le code. Je correspondais avec mes professeurs par mail uniquement, et je n'avais pas d'analyseur logique avec moi.

Je devais effectuer les tests chez moi. Il m'est impossible de reproduire la présence de dizaines d'élèves passant et stagnant dans le couloir, n'ayant qu'une plaquette pour pluguer mes breakout, ils sont côte à côte. Si le test se limite à mettre du plastique sur l'un des deux, je ne pourrais pas conclure sur la différence entre les valeurs relevées. Je cherchais quand même à faire fonctionner mon capteur avec un programme en C.

Le capteur me donnais une valeur de Bit d'erreur = 2 ce qui correspond au MEASMODE et parfois une valeur de 130 qui ne correspond à rien du tout.

En effectuant des demandes de status, le capteur me donnait également des valeurs erronées (0x91 qui ne correspond à rien dans la documentation du capteur au lieu de 0x90 ou 0x98 qui certifient le bon fonctionnement du capteur).

En cherchant un code en C je suis tombé sur un code en C pour RPI et l'allure du code était grandement similaire au mien.

En poursuivant mes recherches, une personne sur le net avait trouvé une solution au problème de code erreur 2 sauf que la solution était dans une librairie faire par Arduino. Donc inutilisable pour ma RPI.

Adafruit, le constructeur explique sur un site web :

« *This sensor is not well supported on Raspberry Pi. This is because it uses I2C clock stretching witch the Pi cannot do without drastically slowing down the I2C speed. CircuitPython and Arduino are supported.* »

Le problème vient peut être de la vitesse trop importante du RPI. Il aurait fallu ralentir la vitesse de l'horloge I2C. Mais la vitesse que j'utilisais était déjà la plus lente proposée par cette librairie (100kHz).

J'ai alors passé un très long moment à la recherche d'un code, aucun ne marchait à cause de la vitesse de transmission du RPI qui reste trop importante, j'ai eu des difficultés à tester tout les codes proposés car les mettre en œuvre me prenais un certain temps.

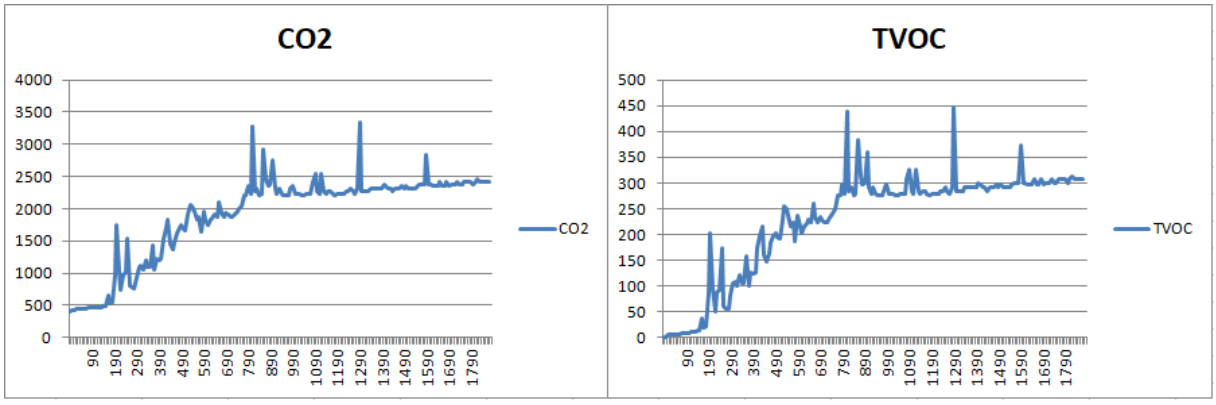
Mon professeur, a testé un programme C++ que je ne parvenais pas à compiler sur une carte RPI.

Je possédais une version Buster et la compilation me donnait des erreurs. J'ai eu à réinstaller une version Jessie en flashant ma carte micro SD avec le logiciel balenaEtcher après avoir téléchargé le .img de la version Jessie.

Le code proposé par mon professeur mettait 20 minutes à se lancer et se ferme anormalement en plus de valeurs erronées toujours présentes mais un peu moins rare.

J'ai cependant effectué quelques mesures afin de vous permettre de visualiser le temps de chauffe du capteur et de remarquer si il y'a des différences notables avec les deux capteurs côte à côte dans le même environnement :

Remarque : ne pouvant pas copier les valeurs directement depuis le moniteur série de l'Arduino, j'ai utilisé le logiciel 'coolterm' afin de récupérer les valeurs via le port COM relié à l'Arduino.

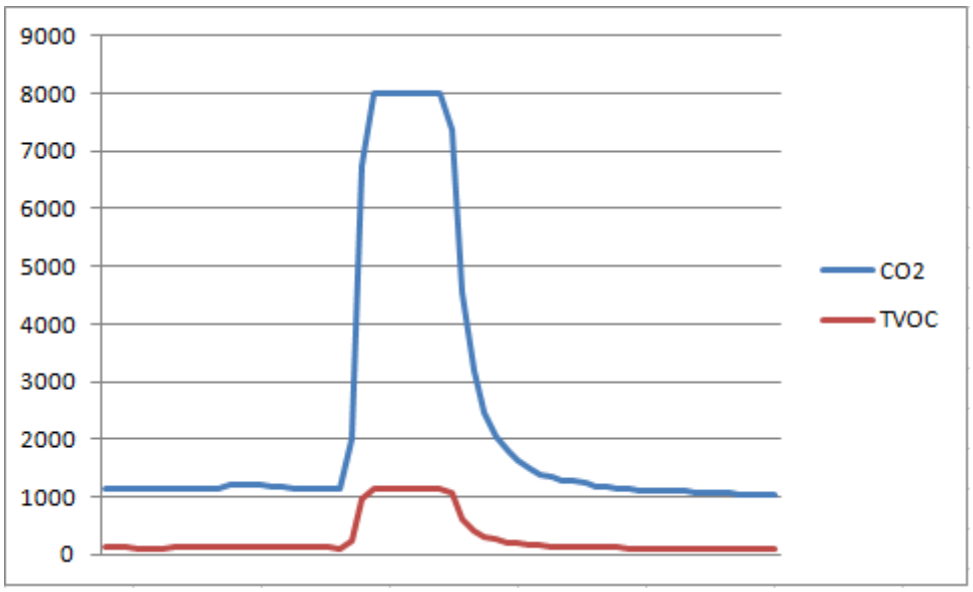


Capture du taux de CO2 et TVOC pendant 30 minutes chaque 10 secondes du CCS811 fonctionnant sous Arduino.

Les valeurs de CO2 sont toujours en ppm (parties par million) et les valeurs de TVOC sont toujours en ppb (parties par billion) sur mes graphiques. On remarque que le capteur se stabilise vers les 770 secondes

ce qui correspond à environ 12 minutes. Ce temps correspond au temps de chauffe du capteur, je détaillerais son principe de fonctionnement dans la partie Physique de mon rapport.

Je trouvais également intéressant de vous présenter les seuils du CCS811.



Capture du taux de CO2 et TVOC du CCS811 fonctionnant sous Arduino.

Ces seuils m'ont intéressé car ils ne sont pas égaux à ceux décrit dans la documentation. Ici nous avons un seuil de CO2 à 7992ppm et un seuil de TVOC à 1156ppb.

eCO_2

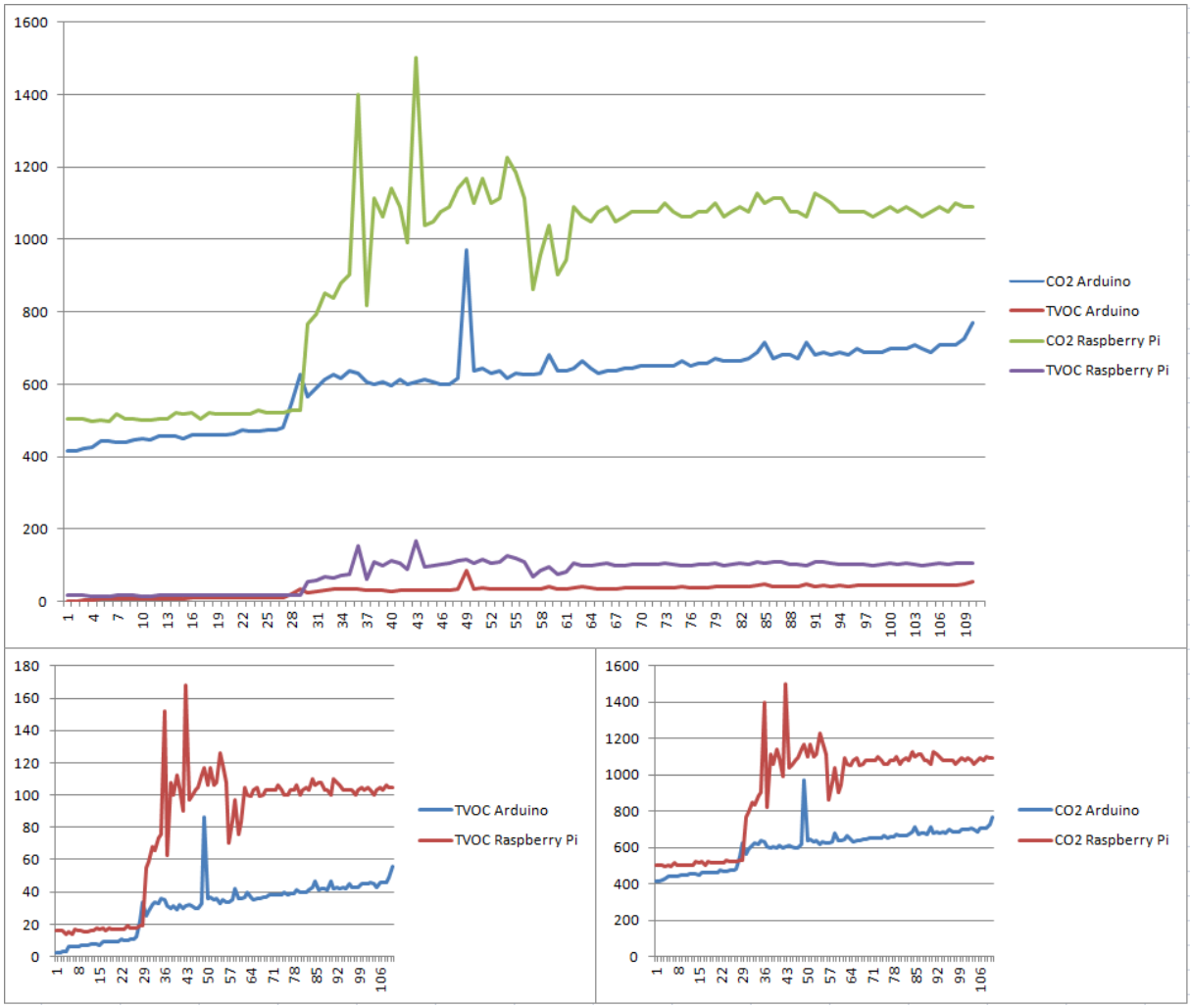
The equivalent CO_2 (eCO_2) output range for CCS811 is from 400ppm to 8192ppm. Values outside this range are clipped.

TVOC

The Total Volatile Organic Compound (TVOC) output range for CCS811 is from 0ppb to 1187ppb. Values outside this range are clipped.

Extraits du datasheet du CCS811

Possédant deux capteur CCS81, j'ai effectué des mesures de TVOC et de CO2 en simultané en reliant un capteur au RPI et l'autre sur l'Arduino.

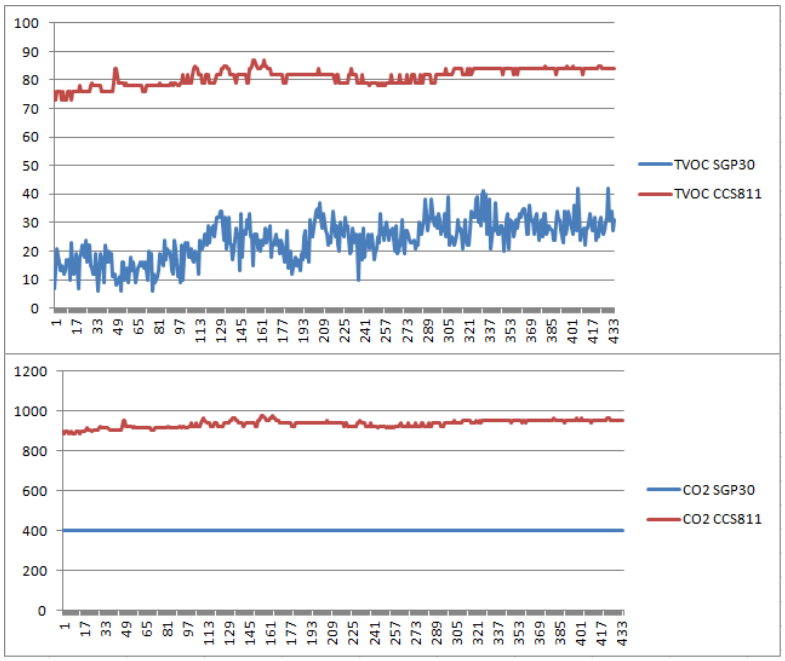


Capture du taux de CO2 et TVOC pendant 25 minutes chaque 20 secondes du CCS811 fonctionnant sous Arduino et RPI.

On constate ici que les valeurs des deux capteurs CCS811 différent en fonction soit si programme, soit de l'environnement. Les deux capteurs étaient à quelques centimètres l'un de l'autre pendant les mesures. Le programme en C utilisé était celui qui se fermait sans raisons au bout d'un certain temps.

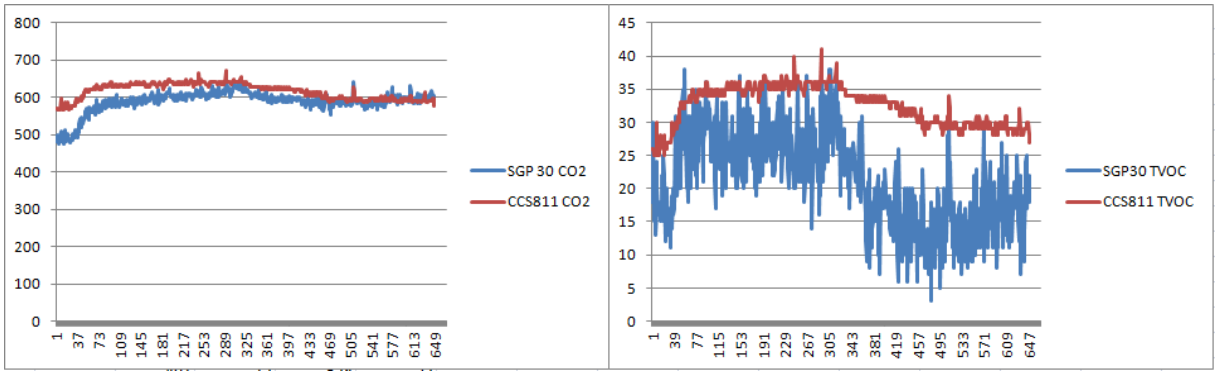
Avec tous nos problèmes sur le CCS811 nous avions eu l'idée de nous rabattre sur le SGP30, un capteur qui renvoie des valeurs similaires au CCS811, fonctionnant avec une tension de 1.8V et nécessitant une adaptation de tension dans le *hat*.

Ne possédant toujours pas d'analyseur logique, la création de code en C à partir du code Arduino m'était impossible. J'ai mis la main sur un programme en Python que mes camarades de projet en option Informatique et Réseaux auraient eu à transposer en langage C par la suite.



Capture du taux de CO2 et TVOC pendant 10 minutes chaque seconde du CCS811 fonctionnant sous Arduino et du SGP30 sous RPI.

Le problème ici était que même si le TVOC varie, le taux de CO2 reste à 400ppm ce qui est le minimum du capteur. On m'a ensuite demandé d'effectuer des captures avec deux capteurs simultanés sous Arduino. J'ai donc du créer un programme Arduino permettant de mettre en œuvre les deux capteurs en simultané. Voici les mesures que j'ai relevées :



Capture du taux de CO2 et TVOC chaque 5 secondes du CCS811 et du SGP30 fonctionnant sous Arduino.

On constate ici que les valeurs de CO2 des deux capteurs sont similaires. Le CCS811 a des mesures de TVOC plus stables que le SGP30.

Vous trouverez à la page suivante mon programme Arduino détaillé.

```
#include "Adafruit_CCS811.h"
#include "SparkFun_SGP30_Arduino_Library.h"
#include <Wire.h>

SGP30 mySensor;
Adafruit_CCS811 ccs;



void setup() {


  Serial.begin(9600);
  Wire.begin();

  if (mySensor.begin() == false) {
    Serial.println("No SGP30 Detected. Check connections.");
    while (1);
  }


  //Initializes sensor for air quality readings
  mySensor.initAirQuality();
  Serial.println("CCS811 test");
  if(!ccs.begin()){
    Serial.println("Error");
    while(1);
  }


  // Wait for the sensor to be ready
  while(!ccs.available());
}


void loop() {

  delay(10000); //Wait 10 second


  mySensor.measureAirQuality();
  Serial.print(mySensor.CO2);
  Serial.print(" ");
  Serial.print(mySensor.TVOC);
  Serial.print(" ");


  if(ccs.available()){
    if(!ccs.readData()){
      Serial.print(ccs.geteCO2());
      Serial.print(" ");
      Serial.print(ccs.getTVOC());
      Serial.println();
    }
    else{
      Serial.println("ERROR!");
      while(1);
    }
  }
}
```

On inclut les librairies des deux capteurs et Wire.h

On créé des objets des objets pour les classes SGP30 et CCS811

void setup

On définit le port série

Initialisation du SGP30

Initialisation du CCS811

On attend que le capteur CCS811 soit prêt

void loop

10 secondes de délais à chaque début de boucle

Mesure du CO2 et du TVOC du SGP30 avec un espace entre chaque valeurs

Mesure du CO2 et du TVOC du CCS811 avec un espace entre chaque valeurs puis un retour à la ligne

Voici le schéma que j'ai réalisé pour mettre en œuvre simultanément mes deux capteurs. Il a été réalisé avec le logiciel Fritzing. Le brakout du CCS811 est un équivalent de celui que je possède.

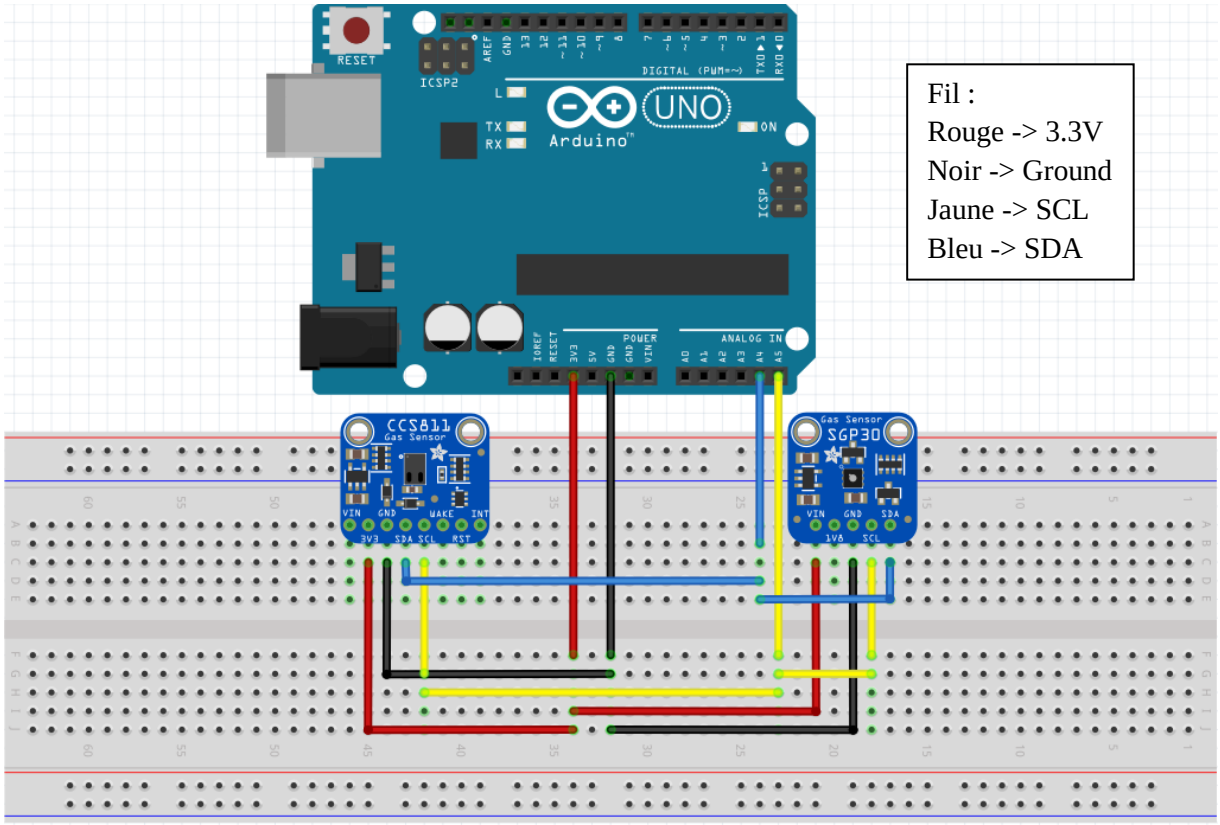
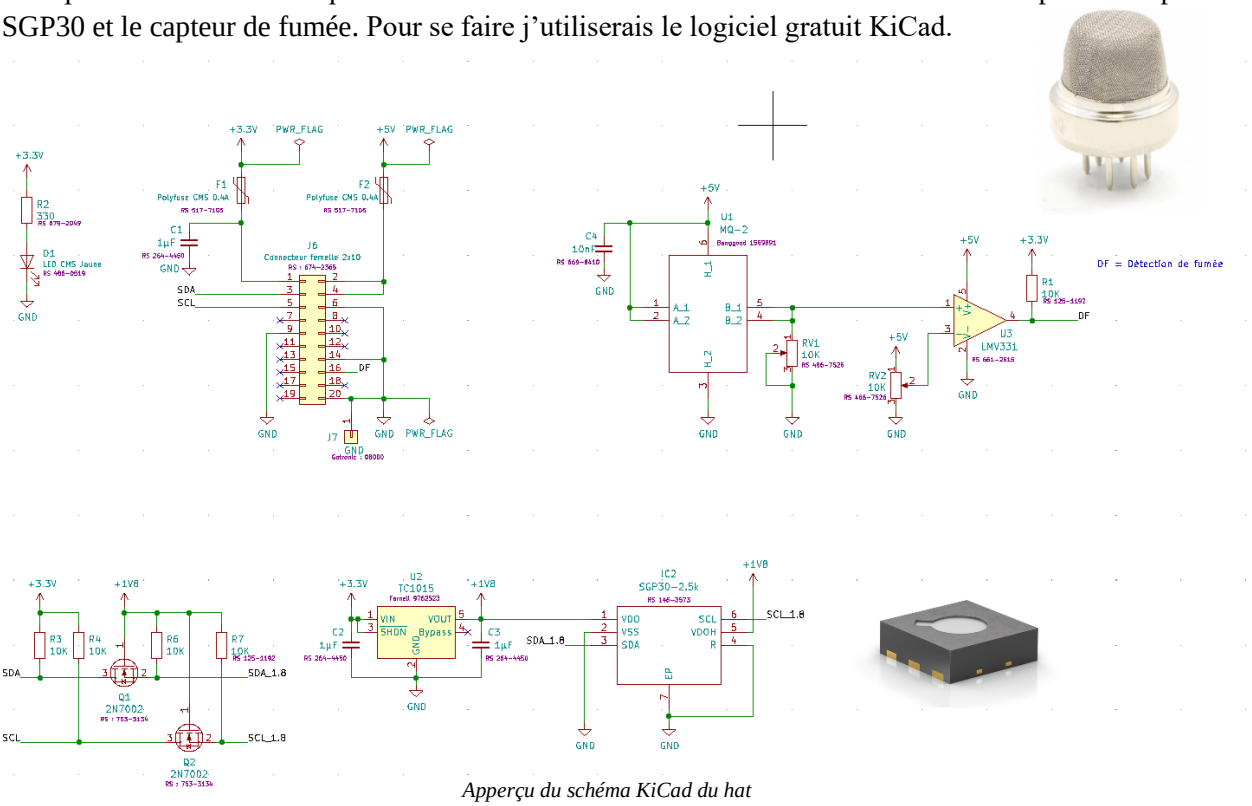


Schéma de mise en œuvre du CCS811 et du SGP30 fonctionnant simultanément sous Arduino.

On rappelle que le breakout du SGP30 possède un level-shifter.

10.6 **Routage du pcb**

Mon professeur d'élèctronique ma demandé de router le schéma du *hat*. Ce schéma comporte le capteur SGP30 et le capteur de fumée. Pour se faire j'utiliserais le logiciel gratuit KiCad.

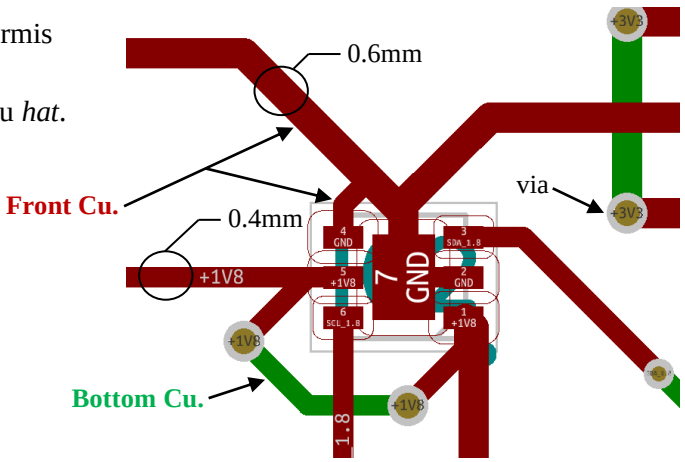


Apperçu du schéma KiCad du hat

J'ai donc commence à faire le routage de la carte. La première étape consiste à placer correctement et ingenieusement les composants. Nous sommes aidés par le *chevelu* qui nous indique les liens entre les composants. Une fois les composants placés, il nous faut les relier entre eux par ce qu'on appelle des pistes qui peuvent avoir différentes largeurs, ici nous avons pris 0.6mm pour les fils qui concernent l'alimentation (sauf exeption), et des fils de 0.4mm pour ce qui est par exemple SDA/SCL etc.

Des exeptions ont étés rencontrées lors du routage de la carte avec de petits composants qui avaient des borniers sur lesquels nous ne pouvions nous connecter seulement avec des pistes de 0.4mm.

J'ai également eu à utiliser des *via* ce qui ma permis de passer du côté *bottom* de la carte afin de grandement simplifier le routage et le câblage du *hat*.



Capture d'un extrait du schéma de routage détaillé