académie d'aix-marseille

B.T.S SN 2018

Projet :**2SP**

Revue de projet finale

2SP ELECTRONIC
Solutions & Systèmes Personnalisés

Lycée Benoit
L'Isle Sur La Sorgue

Etudiants :

GRIMOUD *Yohan*

CHOLEZ-GLAB *Corentin*

GALATIOTO *Enzo*

SOUMILLE *Rémy*

Sommaire

PartieCommune

I/ Le projet

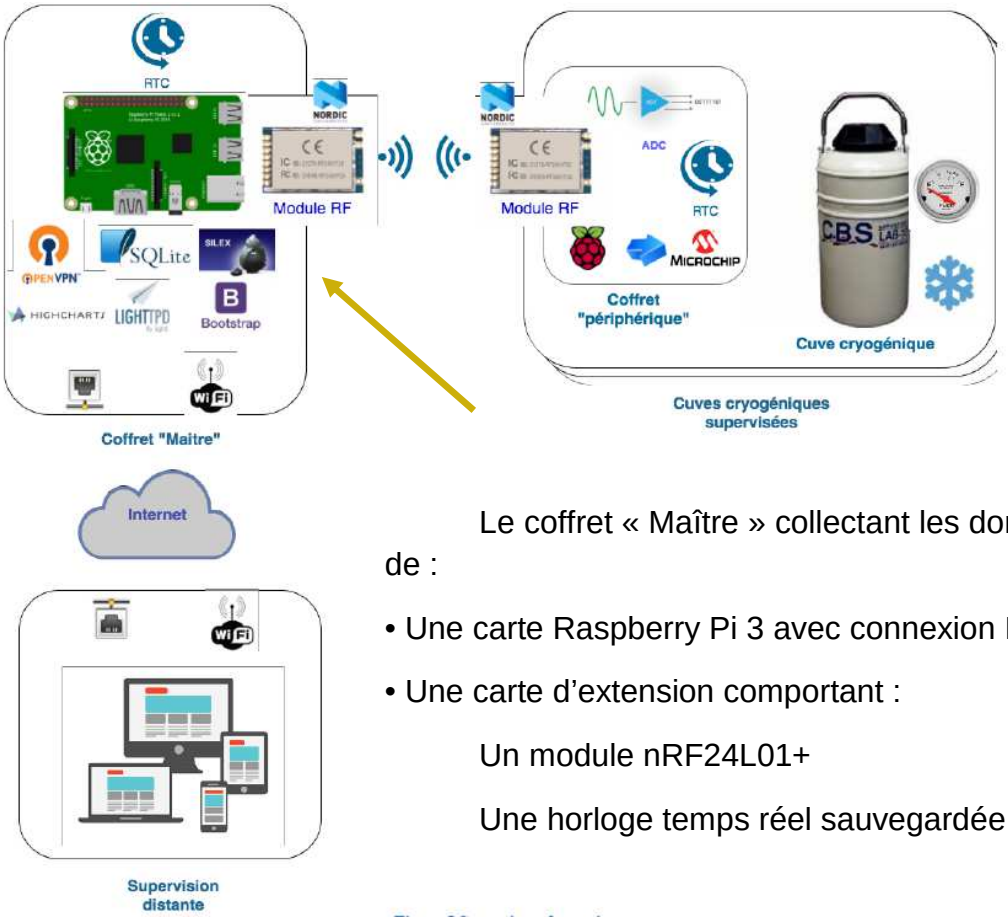
L'objectif initial du projet consiste à proposer une alternative à une application matérielle et logicielle qui a été développée par la société 2SP et qui consiste à surveiller les niveaux d'un parc de cuves qui contiennent de l'azote liquide.

Ce prototype propose déjà des fonctionnalités de contrôle/commande à distance mais souffre d'imperfections auxquelles la société 2SP souhaite remédier.

L'évolution envisagée consiste à :

- Assurer une télémétrie sans fil à l'aide de *breakouts* à base de composant RF de chez Nordic (Référence nRF24L01+).
- Baisser le coût de fabrication du système de supervision par utilisation de cartes électroniques existantes et répandues
- Limiter la consommation électrique du dispositif de mesure alimenté par batterie
- Protéger contre toute falsification les informations de traçabilité du niveau de chaque cuve

Un synoptique de l'ensemble figure ci-dessous :



Le coffret « Maître » collectant les données sera composé de :

- Une carte Raspberry Pi 3 avec connexion Ethernet et WiFi
- Une carte d'extension comportant :

  Un module nRF24L01+

  Une horloge temps réel sauvegardée

II/ L'entreprise

http://2spelectronic.fr/presentation/

2SP Electronic conçoit et développe des systèmes électroniques sur mesure

Ils réalisent des prototypes et petites séries.

Dans son atelier, spécialement conçu pour le prototypage et la petite série, 2SP Electronic apporte une solution mécanique pour l'intégration personnalisée des systèmes électroniques à votre outil de production et procède aux installations.

III/ Cahier des charges

• La mesure du niveau ne fait pas partie du projet. Le projet ne concerne que l'exploitation des différentes mesures.

• Chaque cuve est équipée d'un système de mesure du niveau d'azote

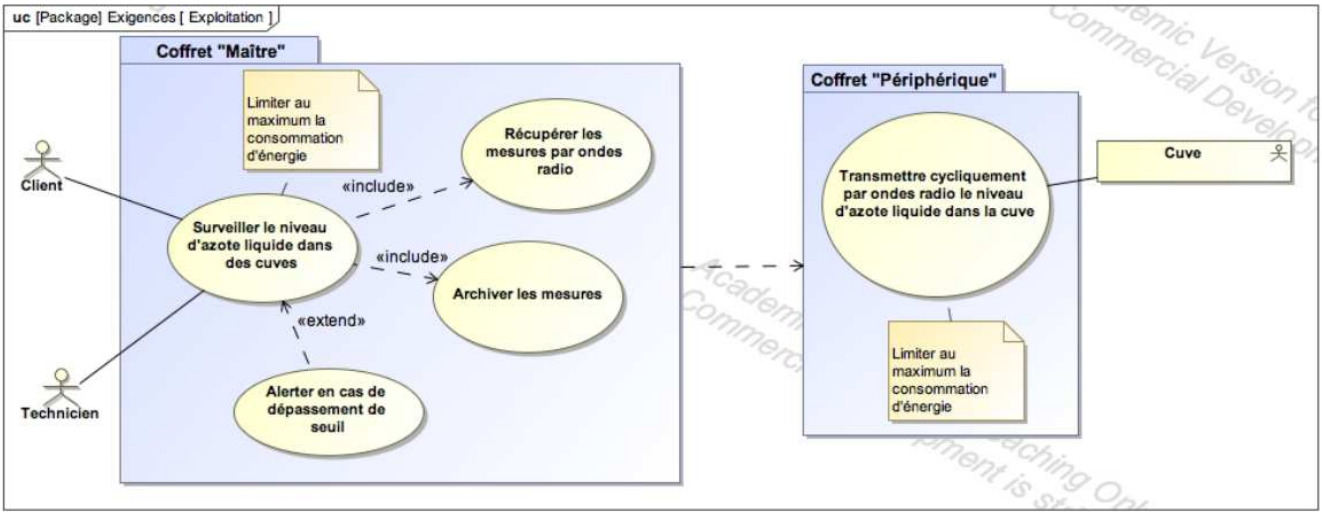
• Chaque cuve doit transmettre, à intervalle de temps régulier, une mesure de niveau par liaison sans fil à une centrale. Celle-ci recueille chaque mesure et la stocke.

• Une surveillance à distance doit être possible (site embarqué, alerte par mail).

• La consommation doit être aussi faible que possible.

• L'objectif est que cet avant-projet soit transposable dans un autre contexte que la mesure de niveau dans les cuves.

IV /Diagramme SYSML



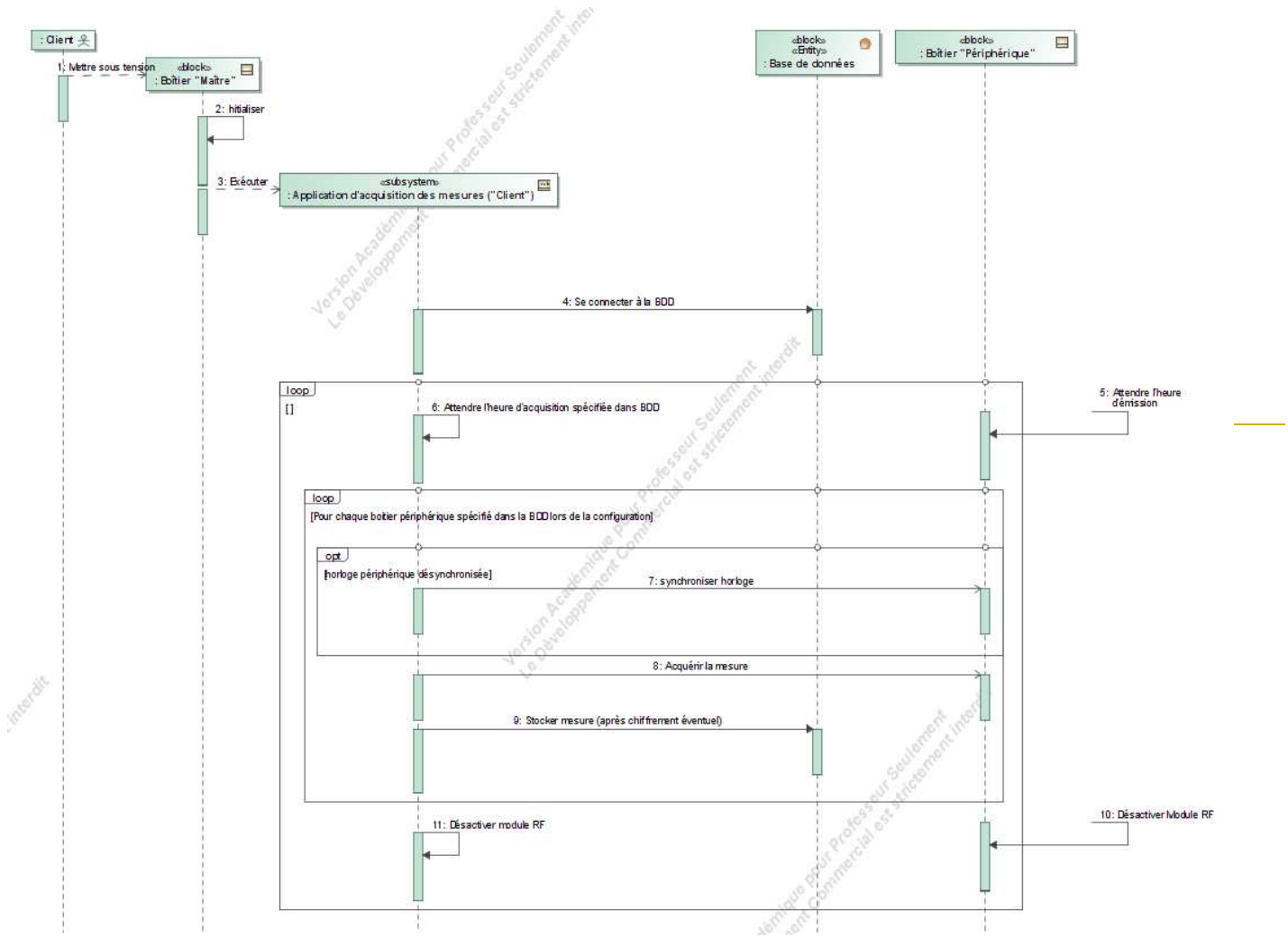


Diagramme de séquence commun

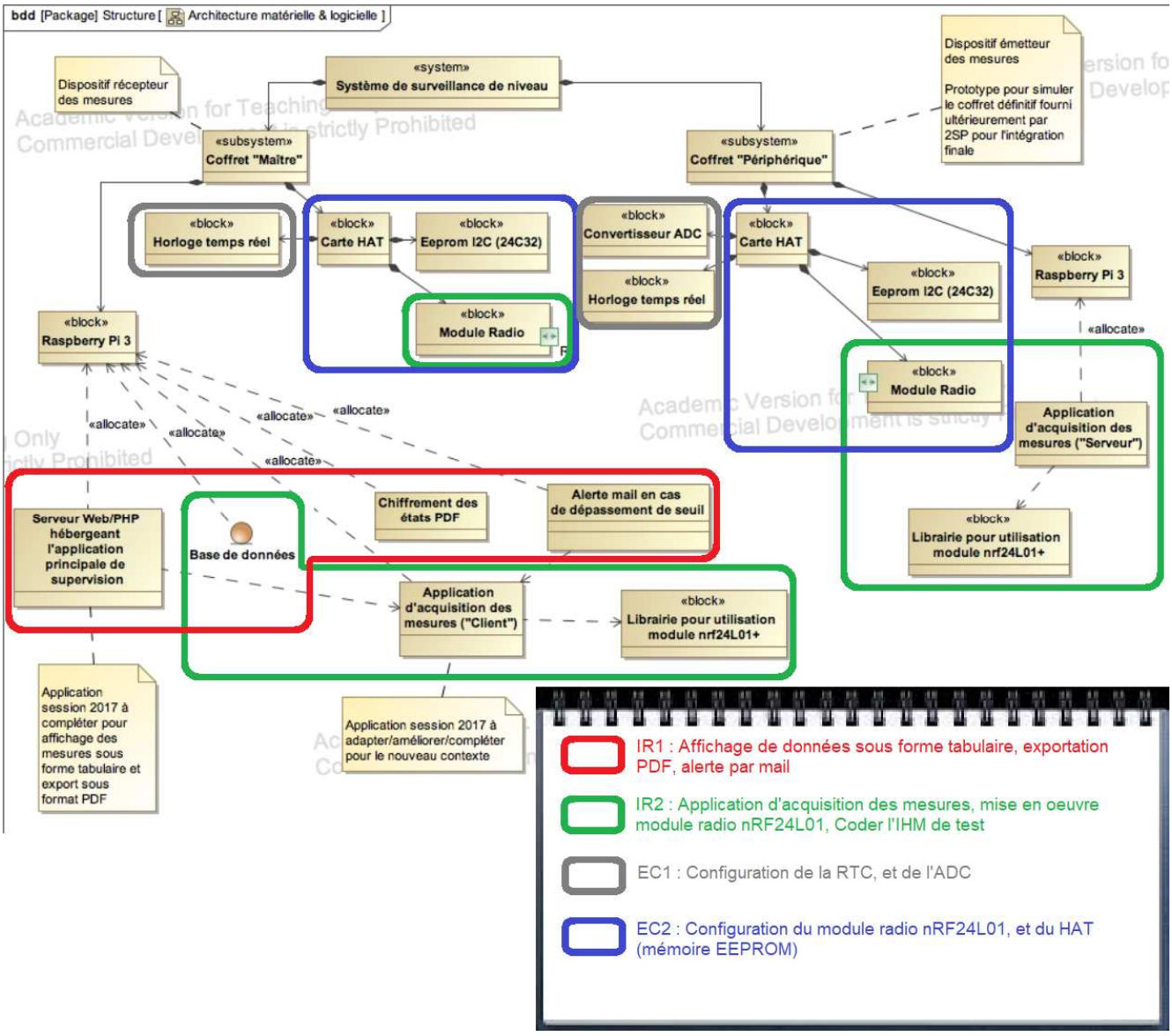
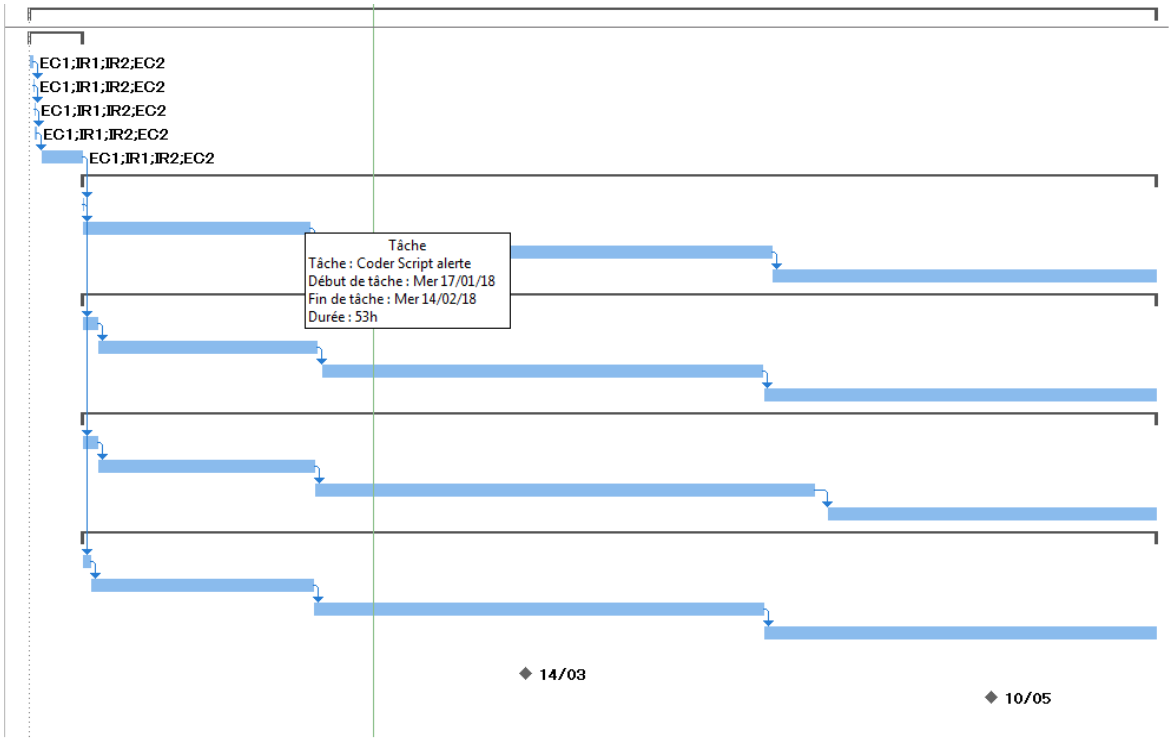


Diagramme de définition de blocs

	Projet	200 heures	Jeu 11/01/18	Jeu 31/05/18		
	Tâches communes	**10 heures**	**Jeu 11/01/18**	**Mer 17/01/18**		
	Lecture du dossier	2 heures	Jeu 11/01/18	Jeu 11/01/18		EC1;IR1;IR2;EC2
	Analyse de l'existant	1 heure	Jeu 11/01/18	Jeu 11/01/18	3	EC1;IR1;IR2;EC2
	Réunion de synthèse	1 heure	Jeu 11/01/18	Jeu 11/01/18	4	EC1;IR1;IR2;EC2
	Réaliser un plan des itérations	2 heures	Jeu 11/01/18	Jeu 11/01/18	5	EC1;IR1;IR2;EC2
	Réaliser un planning prévisionnel	4 heures	Ven 12/01/18	Mer 17/01/18	6	EC1;IR1;IR2;EC2
	Tâches Enzo	**190 heures**	**Mer 17/01/18**	**Jeu 31/05/18**		**IR1**
	Visualiser/filtre mesures BDD	2 heures	Mer 17/01/18	Mer 17/01/18	7	
	Coder Script alerte	53 heures	Mer 17/01/18	Mer 14/02/18	9	
	Conception site Web	80 heures	Jeu 15/02/18	Ven 13/04/18	10	
	Solution pour chiffrer PDF	55 heures	Ven 13/04/18	Jeu 31/05/18	11	
	Tâches Rémy	**190 heures**	**Mer 17/01/18**	**Jeu 31/05/18**		**IR2**
	Utilisation du module nRF	10 heures	Mer 17/01/18	Ven 19/01/18	7	
	Concevoir librairie C++	50 heures	Ven 19/01/18	Jeu 15/02/18	14	
	Coder protocole Maitre/Esclave	69 heures	Ven 16/02/18	Jeu 12/04/18	15	
	Coder une IHM	60 heures	Jeu 12/04/18	Jeu 31/05/18	16	
	Tâches Corentin	**190 heures**	**Mer 17/01/18**	**Jeu 31/05/18**		**EC1**
	Choix du convertisseur (A/N)	10 heures	Mer 17/01/18	Ven 19/01/18	7	
	Carte extension pour Raspberry	50 heures	Ven 19/01/18	Jeu 15/02/18	19	
	Conception librairie	80 heures	Jeu 15/02/18	Mer 18/04/18	20	
	Procédure de synchronisation	50 heures	Ven 20/04/18	Jeu 31/05/18	21	
	Tâches Yohan	**190 heures**	**Mer 17/01/18**	**Jeu 31/05/18**		**EC2**
	Choix Convertisseur (Com)	7 heures	Mer 17/01/18	Jeu 18/01/18	7	
	Carte Extension Rasp	50 heures	Jeu 18/01/18	Jeu 15/02/18	24	
	Conception de la librairie	74 heures	Jeu 15/02/18	Jeu 12/04/18	25	
	Configurer mémoire morte pour le HAT	59 heures	Jeu 12/04/18	Jeu 31/05/18	26	
	Revue 1	0 heure	Mer 14/03/18	Mer 14/03/18		EC1;EC2;IR1;IR2
	Revue 2	0 heure	Jeu 10/05/18	Jeu 10/05/18		EC1;EC2;IR1;IR2
	Revue finale	0 heure	Jeu 14/06/18	Jeu 14/06/18		EC1;EC2;IR1;IR2

Diagramme de GANTT Réalise sous PROJECT



Partie personnelle étudiant 1 (GALATIOTO Enzo)

1/ Tâches à effectuer

La tâche qui m'a été confié consiste essentiellement à faire évoluer l'application Web développée l'an dernier par des étudiants de BTS SN afin qu'elle offre un affichage amélioré des mesures effectuées sur les cuves (mesures stockées dans une base de données) tout en permettant leur filtrage ainsi que leur exportation au format PDF. 2SPELectronic souhaiterait également que lui soit proposé une solution pour protéger de toute falsification les PDFs générés.

En dehors de cela, le client souhaite être informé des mesures anormales qui pourraient survenir en phase d'exploitation.

Un résumé des tâches avec leurs solutions techniques envisagées figure ci-après :

- Mettre en œuvre le plug-in jqGrid ou JQuery Datatables pour visualiser/filtrer les mesures de la BDD SQLite sous forme tabulaire et proposer l'export des données au format PDF.
- Coder un script ou programme qui permet d'alerter par mail en cas de dépassement de seuil (haut ou bas) de remplissage d'une cuve.
- Proposer une solution qui chiffre les PDF générés (script shell, javascript, programme C++…)
- Concevoir/Coder/Tester l'évolution du site Web existant développé avec le Framework PHP Silex pour intégrer le nouvel affichage des mesures

2/ Plan des itérations

3/ Itération 1

3.1) MISE EN ŒUVRE DE L'APPLICATION WEB 2SP EXISTANTE

Mon 1er travail a consisté à mettre en œuvre ce qui a été réalisé l'an dernier.

Pour cela, il a fallu :

- Installer/configurer l'OS de la Raspberry Pi
- Permettre un accès au système de fichiers de la Raspberry Pi depuis ma machine de développement Windows
- Cloner l'archive du projet contenant les sources de l'application Web
- Installer l'environnement Web sur la Raspberry Pi (PHP, librairies tierces)
- Installer le gestionnaire de BDD (SQlite)
- Installer/configurer le serveur Web (Lighttpd)

3.1.1) INSTALLATION DE L'OS DE LA RASPBERRY PI

Premièrement, il a fallu préparer la carte SD c'est à dire y copier une image compressée de l'OS "Raspbian Stretch Lite".

Ensuite, j'ai dû :

- configurer la carte Raspberry, en activant la connexion SSH à distance, ainsi que l'option permettant l'accès bus i2C
- changer le mot de passe pour plus de sécurité
- faire en sorte de pouvoir à accéder à internet depuis le lycée sans avoir a reconfigurer cet accès à chaque mise sous tension. Ceci se résume à :
- installer le paquet resolvconf avec la commande *sudo apt-get update ; sudo apt-get install resolvconf.*
- renseigner ensuite le fichier */etc/dhcpcd.conf* avec une *directive static domain_name_servers* à laquelle on fournit l'adresse IP du serveur DNS

3.1.2) MISE EN PLACE D'UN PARTAGE DE FICHIERS

L'étape suivante a consisté à créer un partage de fichiers avec le service **SAMBA** sur la Raspberry PI afin d'accéder à ses fichiers directement depuis l'explorateur de fichier de ma machine de développement Windows.

Samba est une application permettant d'utiliser sous Linux le protocole SMB (Session Message Block), qui est un protocole réseau utilisé pour partager des fichiers.

J'ai utilisé l'outil **SAMBA,** pour partager un disque Linux (ma Raspberry PI) pour ma machine WINDOWS.

Grâce à ce partage, j'ai pu accéder à mes fichiers et codes sources étant sur ma Raspberry.

La procédure d'installation/configuration de Samba sur la Raspberry Pi est la suivante :

- Installer les paquets nécessaires
  ↳sudo apt-get install samba samba-common-bin
- Faire une copie de sauvegarde du fichier de configuration original
  ↳*sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.backup*
- Editer le fichier de configuration /etc/samba/smb.conf pour **créer un nouveau partage** protégé par mot de passe en y ajoutant :

```
[Partage 2SP 2018]
comment = Partage Samba sur Raspberry Pi pour le projet 2SP session 2018
path = /home/pi/2sp_2018
writable = yes
```

```
#guest ok = yes
#guest only = yes
create mode = 0777
directory mode = 0777
share modes = yes
valid users = pi
```

3.1.3) CLONAGE DE L'ARCHIVE DU PROJET

Les sources de l'application Web sont disponibles sur un service d'hébergement de fichiers qui prend en charge également la gestion de version : ***BitBucket*** *(*https://bitbucket.org/product*).*

L'utilisation de ce site nécessite d'installer en local sur la machine le logiciel ***git***.

Une fois *git* installé et un compte créé sur *Bitbucket* (soit en s'inscrivant soit par l'intermédiaire d'une invitation), il suffit de récupérer les logiciels du projet par une opération de clonage.

```
$ sudo apt-get update
$ sudo apt-get install git
$ cd /home/pi
$ git clone https://nom_utilisateur_compte_bitbucket@bitbucket.org/cdefrance/2sp_2017.git
```

À présent le répertoire /home/pi/2sp_2017 contient l'ensemble des fichiers archivés du projet de 2017.

Il suffit alors de déplacer l'application Web contenu dans cette arborescence dans un répertoire plus commode/approprié qui sera utilisé pour le développement de cette année

```
$ mkdir /home/pi/2sp_2018 /home/pi/2sp_2018/public_html
$ cp -R /home/pi/2sp_2017/ressources/codebase/silex/dashboard/
/home/pi/2sp_2018/public_html/
$ cd /home/pi/2sp_2018
```

3.1.4) INSTALLATION DE L'ENVIRONNEMENT WEB

L'application web 2SP existante s'appuie sur un ensemble de librairies tierces dont le framework PHP ***Silex*** qui constitue le cœur de l'application.

L'installation de ces librairies va s'effectuer depuis une application PHP nommée ***Composer***.

Composer est une application PHP qui permet de gérer les librairies dont dépend une application PHP.

On spécifie dans un fichier texte (→ composer.json) les librairies utilisées dans notre application et *Composer* se charge de les télécharger avec leurs dépendances et de les installer dans le répertoire vendor/ à la racine du site.

Composer peut donc être considéré comme une sorte de gestionnaire de package comme *apt* dans Linux.

Étant donné que *Composer* est une application PHP, il faut en premier lieu installer ce langage.

```
↳sudo apt-get install php7.0-common php7.0-cgi php7.0-sqlite3 php7.0
```

Ensuite, il suffit de suivre la procédure indiquée sur le site web de *Composer* pour l'installer (https://getcomposer.org/download/).

```
$ curl https://getcomposer.org/installer -o composer-setup.php
$  php -r "if (hash_file('SHA384', 'composer-setup.php') ===
'544e09ee996cdf60ece3804abc52599c22b1f40f4323403c44d44fdfdd586475ca9813a858088
ffbc1f233e9b180f061') { echo 'Installer verified'; } else { echo 'Installer corrupt';
unlink('composer-setup.php'); } echo PHP_EOL;"
Installer verified
$ php composer-setup.php
All settings correct for using Composer
Downloading...

Composer (version 1.6.2) successfully installed to:
/home/pi/2sp_2018/public_html/dashboard/composer.phar
Use it: php composer.phar

$ rm composer-setup.php
$ sudo mv composer.phar /usr/local/bin/composer
```

Il reste alors à installer les librairies spécifiées dans le fichier composer.json de l'application clonée ainsi que leurs dépendances.

```
$ cd /home/pi/2sp_2018/public_html/dashboard/
$ composer install
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 35 installs, 0 updates, 0 removals
  - Installing twig/twig (v1.21.2): Downloading (100%)
  - Installing twig/extensions (v1.3.0): Downloading (100%)
  - Installing symfony/routing (v2.8.33): Downloading (100%)
  - Installing paragonie/random_compat (v1.4.2): Downloading (100%)
  - Installing symfony/polyfill-mbstring (v1.6.0): Downloading (100%)

  [...]

monolog/monolog suggests installing sentry/sentry (Allow sending log messages to a Sentry
server)
symfony/monolog-bridge suggests installing symfony/console (For the possibility to show log
messages in console commands depending on verbosity settings. You need version ~2.3 of
the console for it.)
```

```
Writing lock file
Generating autoload files
```

Les librairies sont alors installées dans le répertoire vendor/ de l'application.

3.1.5) INSTALLATION DE SQLITE STUDIO

L'installation du gestionnaire de base de données SQLite utilisée par l'application se fait simplement via le gestionnaire de paquets de Raspbian.

```
$ sudo apt-get install sqlite3 libsqlite3-dev
```

Aucune configuration n'est necessaire

3.1.6) INSTALLATION / CONFIGURATION DU SERVEUR WEB LIGHTTPD

Ici, il faut non seulement installer le serveur Web mais aussi le configurer pour :

- mettre en place un hôte virtuel pour l'application Web 2SP
- mettre en place le mécanisme de **réécriture d'adresses** qui consiste à transformer les URLs reçues par le serveur de façon à les faire correspondre à des ressources réellement présentes sur le site
-

La procédure est la suivante :

1. Activer le module mod_rewrite de reécriture d'adresse.

Il suffit d'ajouter le module mod_rewrite à la suite de ceux déjà présents dans la liste de l'option server.modules du fichier /etc/lighttpd/lighttpd.conf

2. Activer le module simple-vhost.

```
$ sudo lighty-enable-mod simple-vhost
Enabling simple-vhost: ok
Run /etc/init.d/lighttpd force-reload to enable changes
```

3. En tant qu'administrateur, configurer les hôtes virtuels pour le site en modifiant comme suit le fichier /etc/lighttpd/conf-enabled/10-simple-vhost.conf :

```
# !! Mettre le nom d'hôte en minuscule dans le motif de correspondance (<- server.force-lowercase-filenames ??)
$HTTP["host"] == "dashboard" {
        simple-vhost.server-root          = "/home/pi/2sp_2018/public_html/"
        simple-vhost.document-root        = "/web/"
url.rewrite-once = (
            # configure some static files
            "^/favicon\.ico$" => "$0",
```

```
            "^/css/.+" => "$0",
            "^/img/.+" => "$0",
            "^/lib/.+" => "$0",
            "^(/[^\?]*)(\?.*)?" => "/index.php$1$2"
        )
} else $HTTP["host"] == "default" {
        simple-vhost.server-root = "/home/pi/2sp_2018/public_html/"
        simple-vhost.document-root = "/"
}
## the default host if no host is sent
simple-vhost.default-host        = "dashboard"
```

4. Redémarrer le serveur web pour prendre en compte les modifications

```
$ sudo systemctl restart lighttpd
```

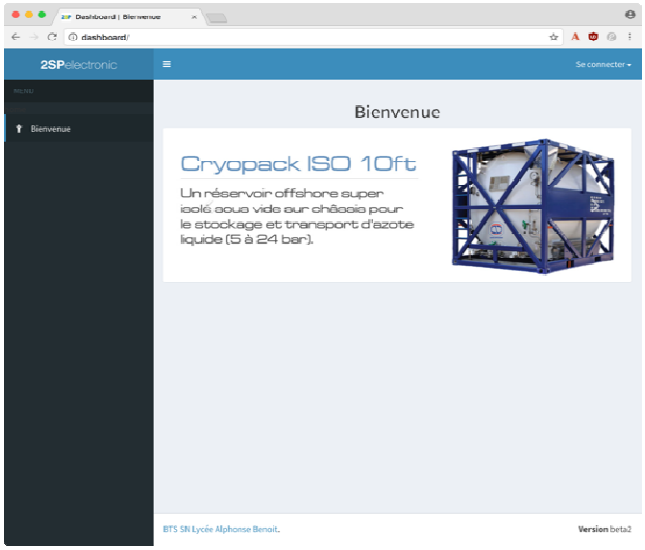
5. Ajouter 1 entrée pour le site dashboard dans le fichier C:\Windows\System32\drivers\etc\hosts de l'ordinateur depuis lequel sera lancé le navigateur internet en spécifiant l'adresse IP réelle de la Raspberry Pi

```
192.168.1.20       dashboard
```

6. Ajuster les autorisations sur les répertoires

```
$ cd /home/pi/2sp_2018
$ sudo chgrp -R www-data public_html/
$ chmod g+w public_html/dashboard/db/db2sp.db
public_html/dashboard/var/log/dashboard.log
```

L'application web est désormais accessible depuis le navigateur :

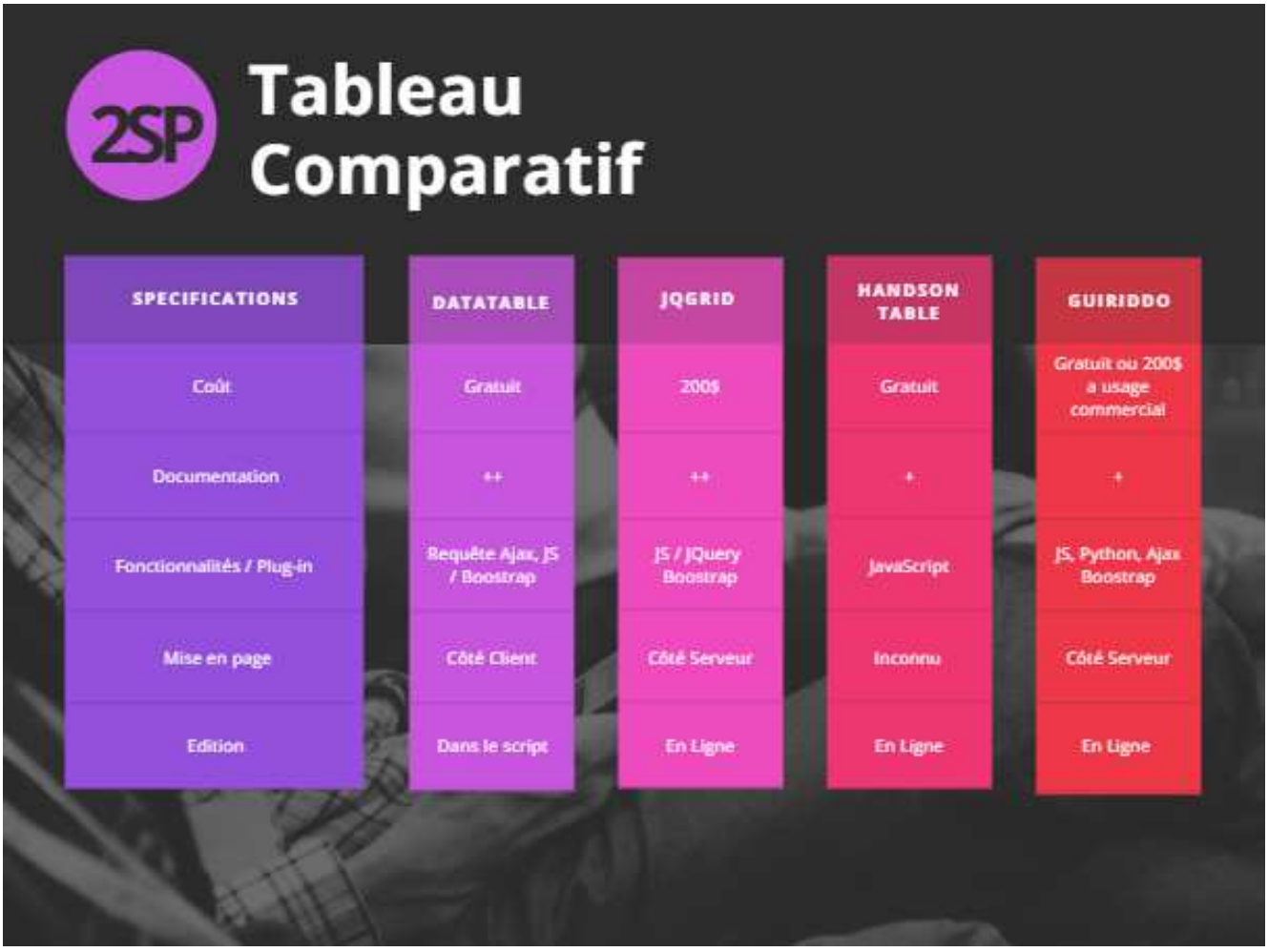


3.2/ MISE EN ŒUVRE DE L'AFFICHAGE DE DONNEES SOUS FORME TABULAIRE

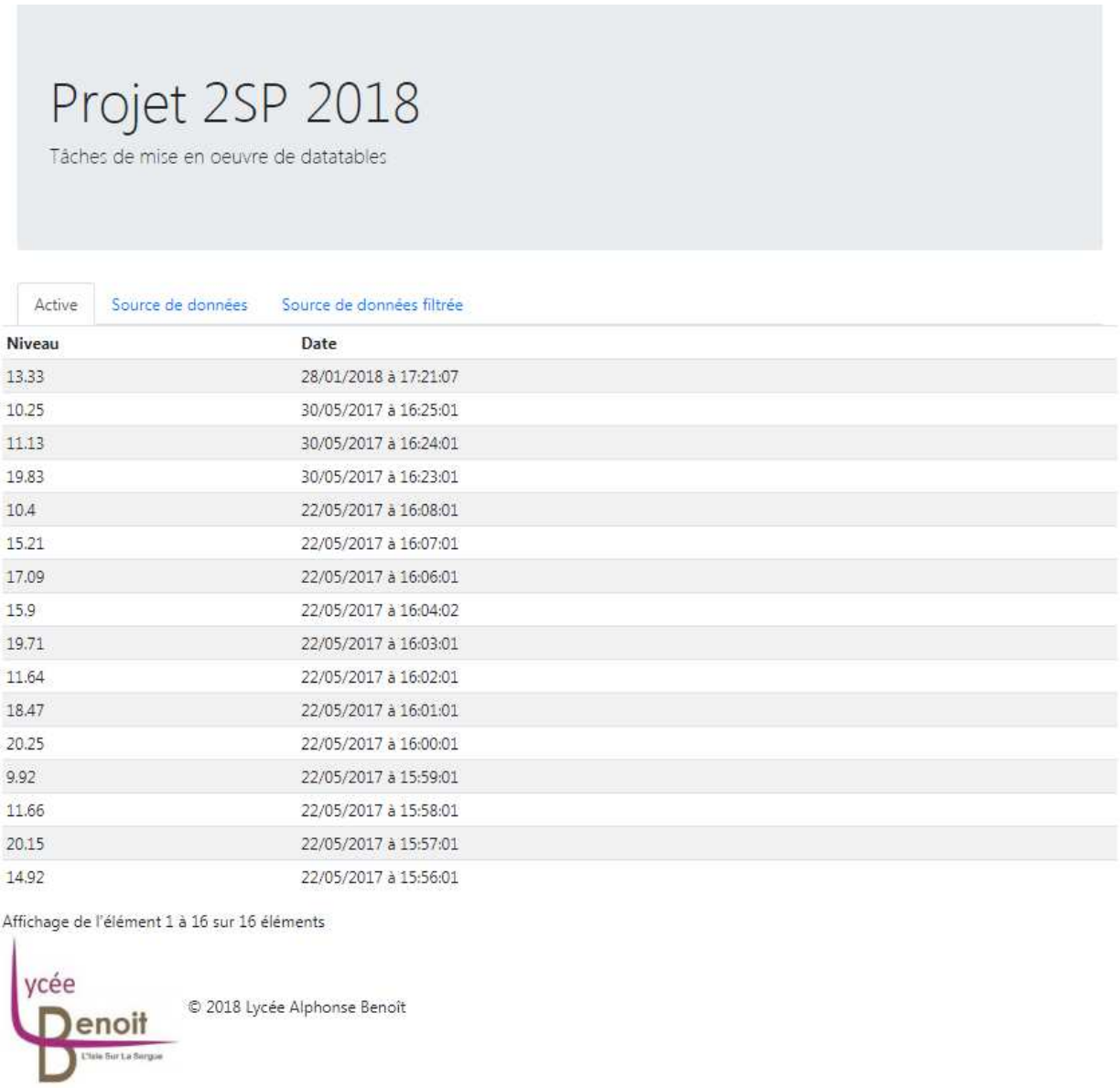
3.2.1) CHOIX D'UNE LIBRAIRIE

Suite à une réunion de vie de projet, avec notre professeur référent et les membres du groupe de projet, pendant laquelle j'ai fait une synthèse sur les différentes solutions possibles, nous nous sommes mis d'accord sur le fait d'utiliser la librairie JavaScript ***DataTables*** qui permet d'afficher des données sous forme tabulaire. Cette librairie propose de nombreuses fonctionnalités (Ajax, tris par colonne, export PDF etc…).

Voici un tableau comparatif des librairies explorées lors de mon étude :



Voici l'onglet numéro (1) du site web correspondant à l'itération 1.



Projet 2SP 2018

Tâches de mise en oeuvre de datatables

Active | Source de données | Source de données filtrée

Niveau	Date
13.33	28/01/2018 à 17:21:07
10.25	30/05/2017 à 16:25:01
11.13	30/05/2017 à 16:24:01
19.83	30/05/2017 à 16:23:01
10.4	22/05/2017 à 16:08:01
15.21	22/05/2017 à 16:07:01
17.09	22/05/2017 à 16:06:01
15.9	22/05/2017 à 16:04:02
19.71	22/05/2017 à 16:03:01
11.64	22/05/2017 à 16:02:01
18.47	22/05/2017 à 16:01:01
20.25	22/05/2017 à 16:00:01
9.92	22/05/2017 à 15:59:01
11.66	22/05/2017 à 15:58:01
20.15	22/05/2017 à 15:57:01
14.92	22/05/2017 à 15:56:01

Affichage de l'élément 1 à 16 sur 16 éléments

Les valeurs affichées proviennent de valeurs brutes rentrées sous forme de tableau JavaScript dans la page html affichée :

```
$(document).ready(function() {
    var dataSet = [
        [13.33, 1517156467],
        [10.25, 1496154301],
        [11.13, 1496154241],
        [19.83, 1496154181],
        [10.4,  1495462081],
        [15.21, 1495462021],
        [17.09, 1495461961],
        [15.9,  1495461842],
        [19.71, 1495461781],
        [11.64, 1495461721],
        [18.47, 1495461661],
        [20.25, 1495461601],
        [9.92,  1495461541],
        [11.66, 1495461481],
        [20.15, 1495461421],
        [14.92, 1495461361]
    ];
```

13.33	28/01/2018 à 17:21:07
10.25	30/05/2017 à 16:25:01

Le niveau (1ère valeur du tableau) est retranscrit tel quel tandis que la date (2ème valeur du tableau) exprimée sous forme d'un timestamp (nombre de secondes écoulées depuis le 01/01/1970 00 :00 :00) est convertie grâce au code suivant inclus dans l'attribut render de l'objet Datatable :

```
render: function(data, type, row, meta) {
if( type === 'display' || type === 'filter')        {
var dateTime = new Date(data*1000);
var dateString = dateTime.toLocaleString('fr-FR');
return dateString;
 }
```

4/ Itération 2

- Adaptation du site web pour afficher les données de la BDD
- Concevoir un script permettant d'alerter par mail lors d'un dépassement de seuil de remplissage de cuve

4.1.1) CREATION DU SECOND ONGLET

Dans un premier temps, j'ai adapté mon site web pour y afficher les données provenant de la base de données db2sp.db

J'ai donc créé un second onglet appelé "Source de données", grâce a la fonction « navs » de l'outil de design WEB → Boostrap

Source : « *https://getbootstrap.com/docs/4.0/components/navs* »

```
    <!-- Différent onglets -->
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" href="index.php">Active</a>
  <li class="nav-item">
    <a class="nav-link" href="ajax_data_source.php">Source de données</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="ajax_data_source_filtered.php">Source de données filtrée<
  </li>
</ul>
</ul>
```

4.1.2) CONNEXION A LA BDD

Grâce à la fonction suivante qui établit une connexion avec db2sp.db étant notre BDD et se trouvant dans le même répertoire que mon code.

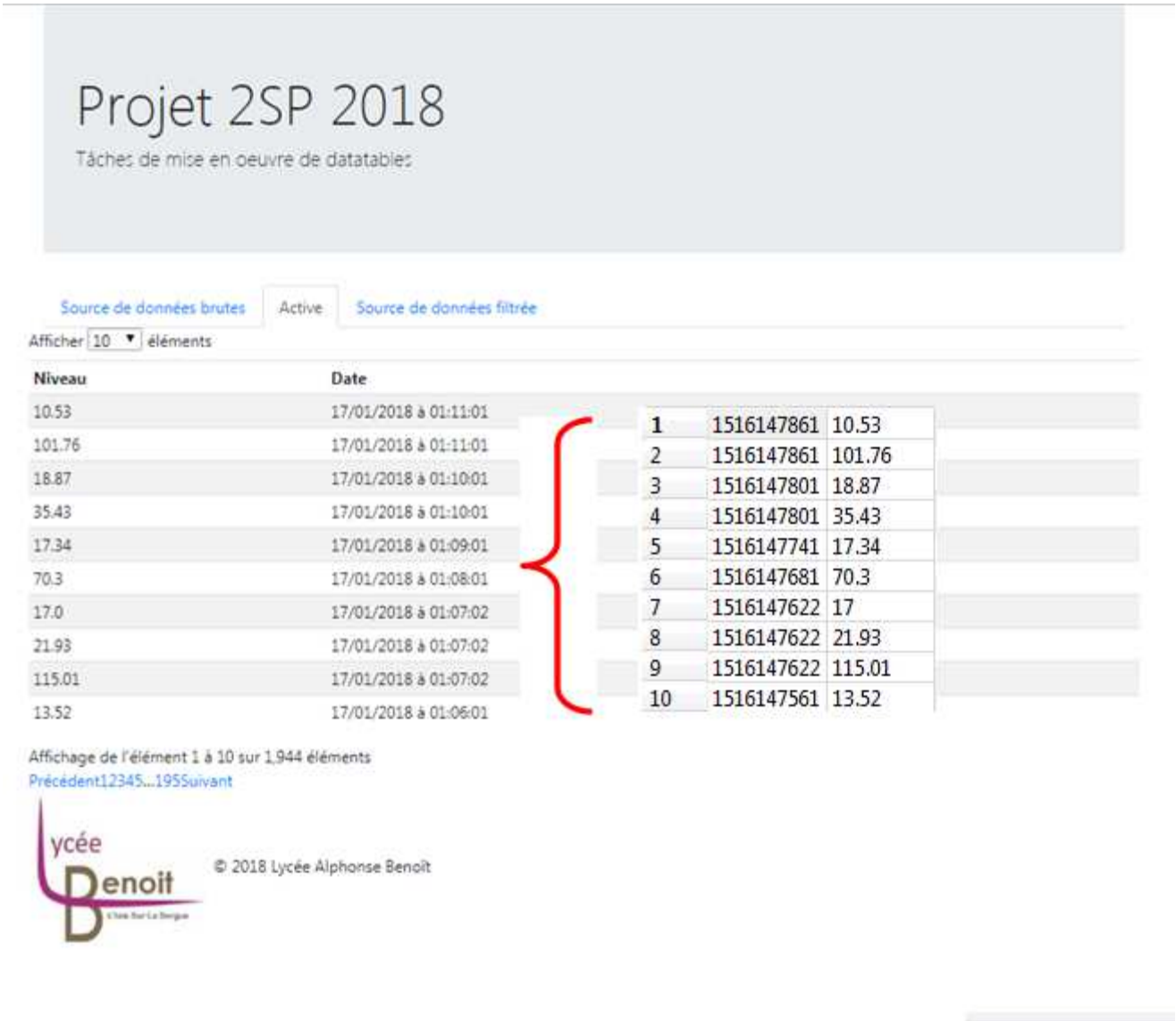
```
define("PDO_HOST", "tmo-datatables");

$myPDO = new PDO('sqlite:./db2sp.db');
```

J'ai pu m'y connecter et ensuite créer des requêtes SQL permettant l'affichage de données choisies de la base de données

Suite à la connexion a la base de données, Une jointure est créée permettant l'affichage de la date venant de la table **RelevesMesures** ainsi que le niveau des Cuves provenant de la table **Cuves Monitorées.**

```
<?php

define("PDO_HOST", "tmo-datatables");

$myPDO = new PDO('sqlite:./db2sp.db');

$stmt = $myPDO->prepare("SELECT R.date, RC.niveau, C.nom FROM RelevesMesures AS R, RelevesMesures_cibler_CuvesMonitorees AS RC WHERE R.idReleveMesure = RC.RelevesMesures_idReleveMesure = RC.CuvesMonitorees_idCuveMonitoree");

$stmt->execute();

$result = $stmt->fetchAll(PDO::FETCH_ASSOC);

$str = json_encode($result);

echo $str;

?>
```

Voici l'onglet numéro (2) du site web correspondant à l'itération 2.



Les données venant de la BDD **db2sp.db** sont bien retranscrites dans le second onglet de ma page WEB & triées de la date la plus récentes a la plus anciennes.

4.2.1) UTILISATION D'UN BOUTON

J'ai choisi d'utiliser le **bouton "pdfHtml5",** qui peut être configuré pour que le PDF s'ouvre automatiquement dans une nouvelle fenêtre / onglet au lieu de télécharger automatiquement le fichier (qui est l'action par défaut).

Cette option est contrôlée par l'option de téléchargement du bouton, qui peut être définie pour être ouverte, j'ai suivi l'exemple donné de la librairie utilisée (Datatables)

https://datatables.net/extensions/buttons/examples/html5/pdfOpen.html

J'y ai inséré le code en JavaScript suivant :

```
$(document).ready(function() {
        $('#zonesB').DataTable( {

    dom: 'Bfrtip',
    buttons: [
      {
          extend: 'pdfHtml5',
          download: 'open'
      }
  ],
```

En plus du code ci-dessus, les fichiers de bibliothèque Javascript suivants sont chargés pour permettre l'affichage du bouton.

https://code.jquery.com/jquery-1.12.4.js

https://cdn.datatables.net/1.10.16/js/jquery.dataTables.min.js

https://cdn.datatables.net/buttons/1.5.1/js/dataTables.buttons.min.js

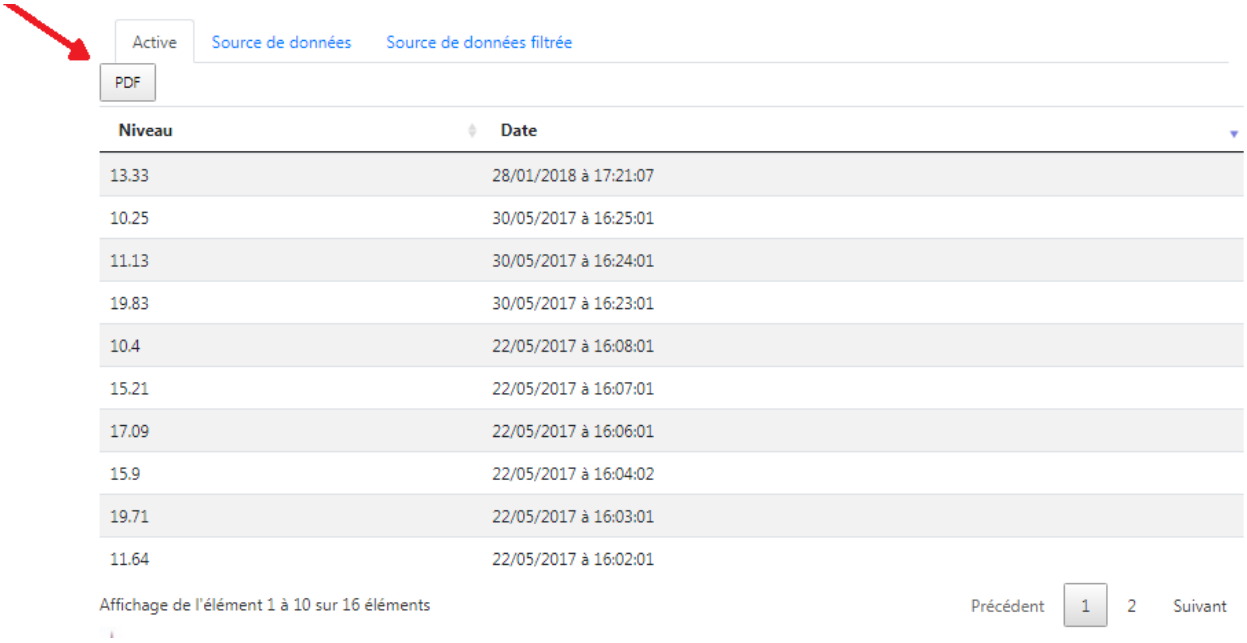
https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.32/pdfmake.min.js

https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.32/vfs_fonts.js

https://cdn.datatables.net/buttons/1.5.1/js/buttons.html5.min.js

4.2.2) RENDU SUR LE SITE WEB

Voici ci dessous l'aperçu de l'ajout du bouton PDF sur l'onglet de mon site WEB

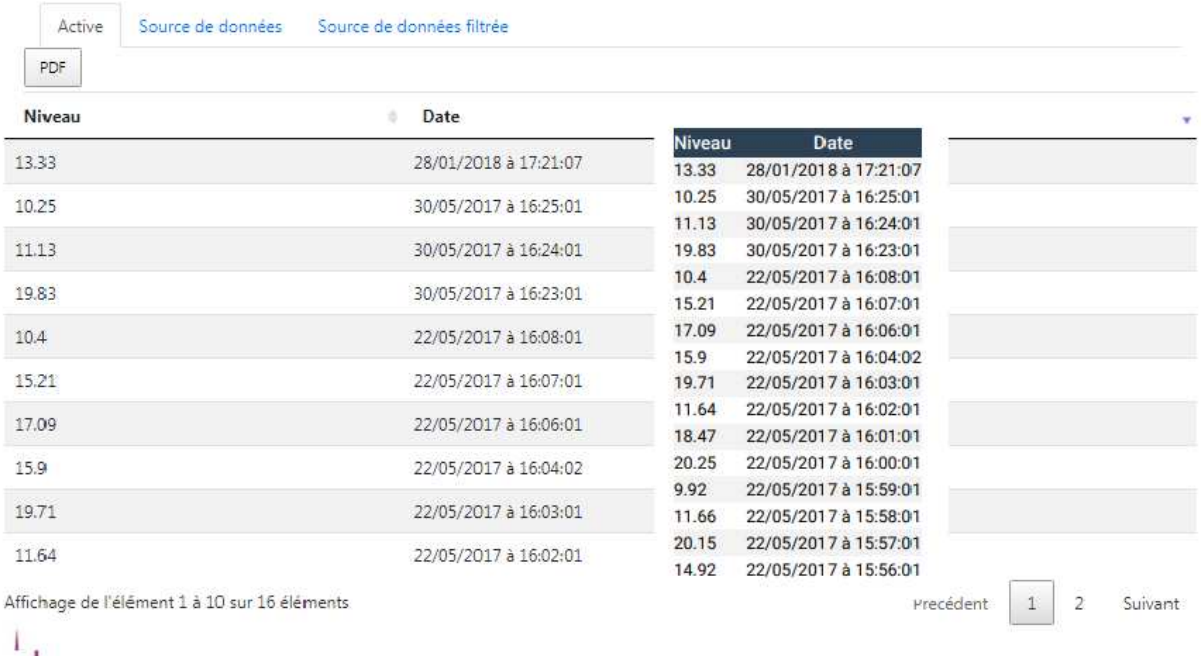


La principale utilitée du bouton PDF étant l'export de données.

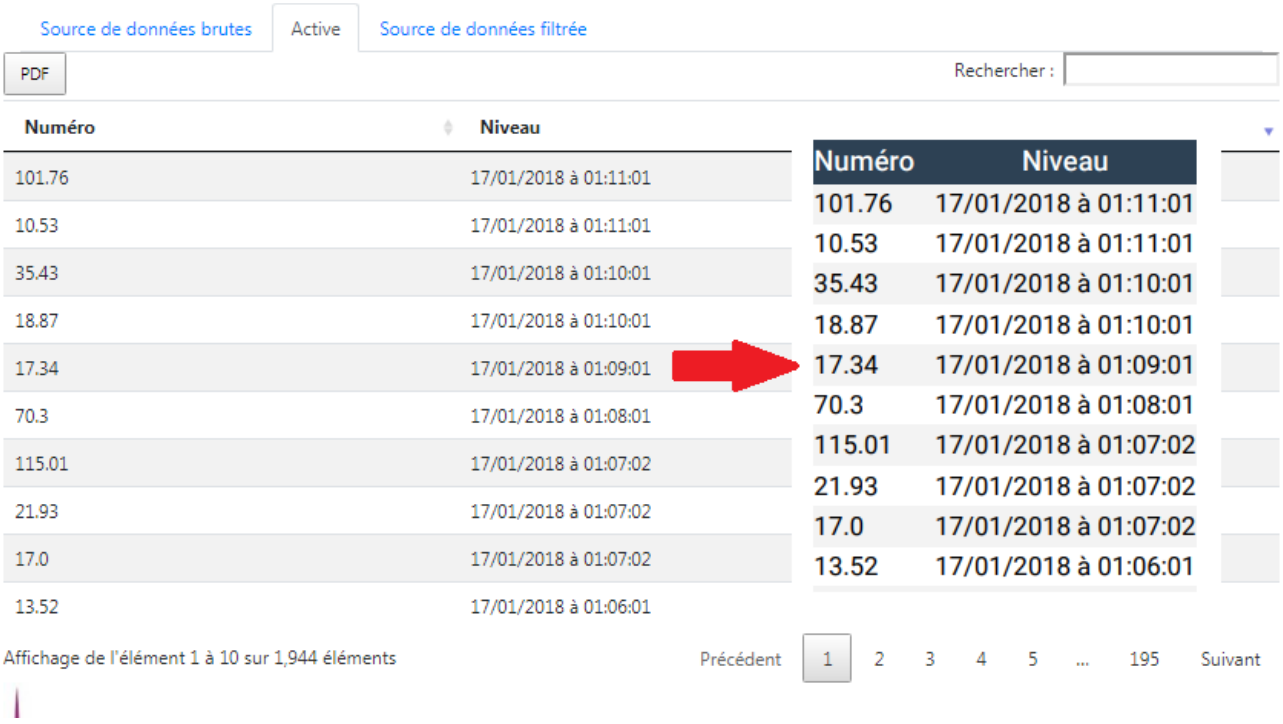
Voici ci joint le rendu du tableau avec les valeurs brutes du tableau (1)

Ainsi qu'une capture d'écran des valeurs exportées venant de la base de données (2)

(1)



(2)


Source de données brutes
Active
Source de données filtrée
PDF
Rechercher :
Numéro
Niveau
101.76 17/01/2018 à 01:11:01
10.53 17/01/2018 à 01:11:01
35.43 17/01/2018 à 01:10:01
18.87 17/01/2018 à 01:10:01
17.34 17/01/2018 à 01:09:01
70.3 17/01/2018 à 01:08:01
115.01 17/01/2018 à 01:07:02
21.93 17/01/2018 à 01:07:02
17.0 17/01/2018 à 01:07:02
13.52 17/01/2018 à 01:06:01
Affichage de l'élément 1 à 10 sur 1,944 éléments
Précédent 1 2 3 4 5 ... 195 Suivant

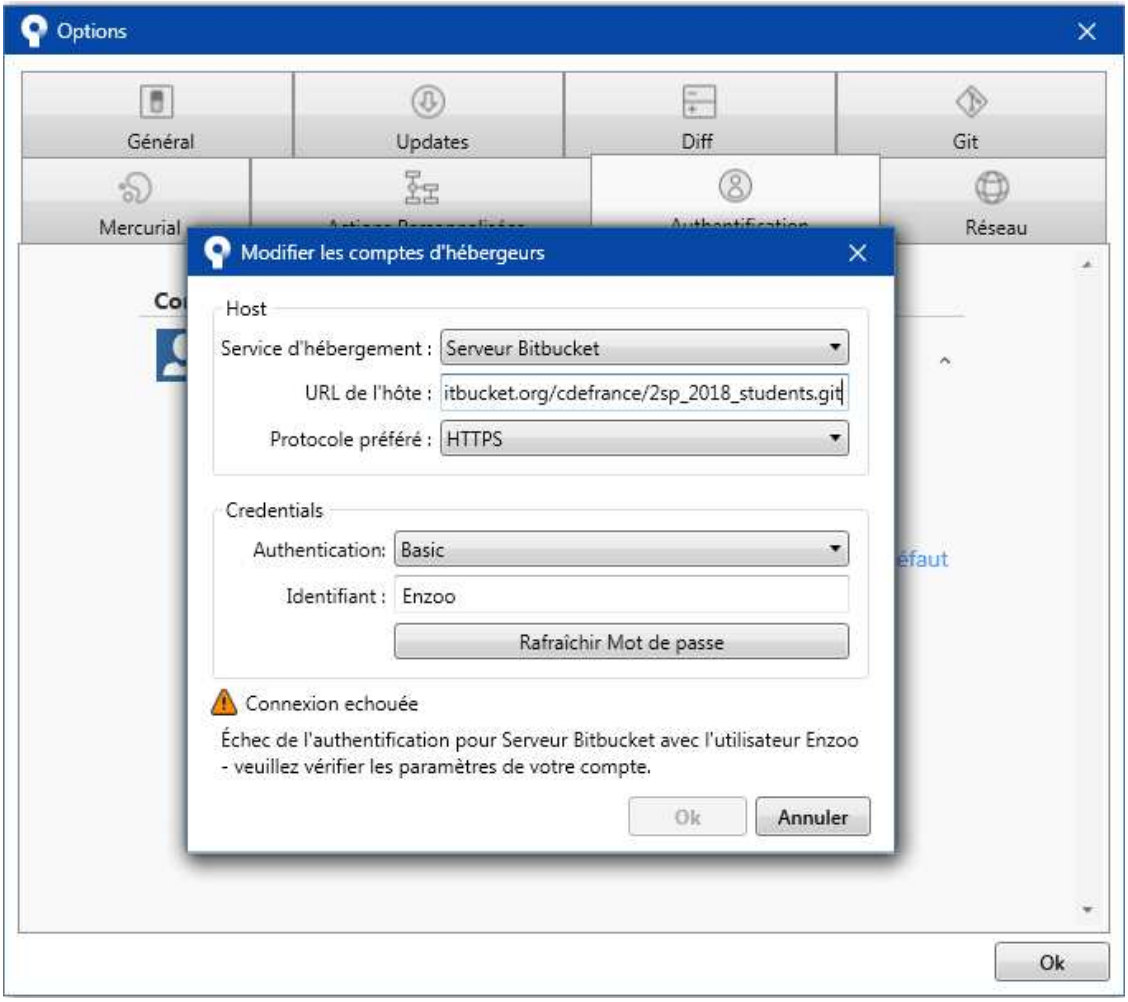
5/ Stockage de fichiers

5.1.1 INSTALLATION / PARAMETRAGE SOURCETREE

Notre professeur a configuré un dépôt logiciel sur le service WEB BitBucket pour héberger nos codes source et suivre leurs versions.

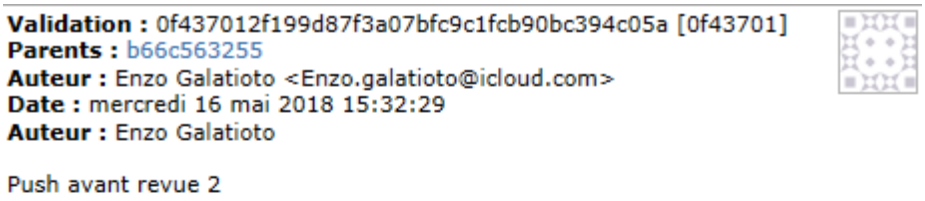
Ce service repose sur le logiciel de gestion de version Git

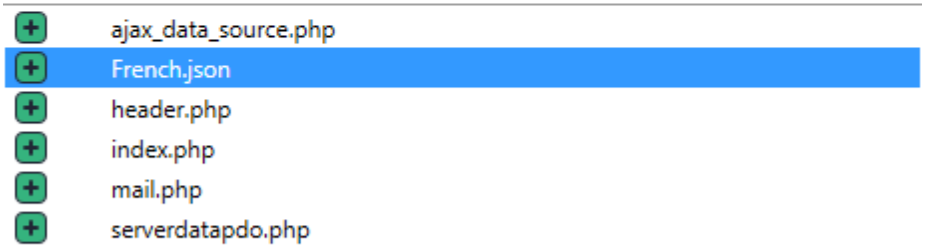
J'ai ensuite installé puis utilisé le logiciel SourceTree qui est un client Git graphique pour simplifier les intéractions avec le dépôt BitBucket du projet 2SP.



En créant un compte et en me connectant depuis l'URL de l'hôte de notre dépôt.

J'ai pu y déposer et contrôler les différentes versions de mon site WEB.





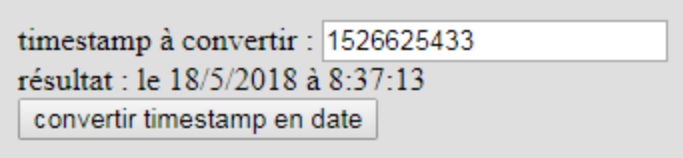
6/ Partie physique

J'ai choisi de développer la conversion **horodatage** (en anglais *timestamping*), qui est un mécanisme qui consiste à associer une date et une heure à un événement, une information ou une donnée informatique.

Il a pour but d'enregistrer l'instant auquel une opération a été effectuée. Une mesure dans le cas des cuves.

Le timestamp est le nombre de secondes écoulées depuis le 1er janvier 1970

Ci-joint un exemple du timestamp, de la date du jour.



Pour retrouver la date et l'heure du jour à partir de cette donnée, on convertit le nombre de secondes depuis l'époque appelé UNIX (depuis le 1er janvier 1970) pour retrouver la date d'aujourd'hui.

Human readable time	Secondes
1 heure	3600 sec
1 jour	86400 sec
1 semaine	604800 sec
1 mois (30.44 jour)	2629743 sec
1 year (365.24 jours)	31556926 sec

7/ Conclusion

Pour conclure, le site est désormais adapté pour afficher les données de la BDD.

et également équipé d'un bouton permettant l'export au format PDF

Il me reste à créer une alerte mail lorsqu'il y aura un dépassement de seuil.

VI/ Partie personnelle de l'étudiant 2 (SOUMILLE Rémy)

1/ Présentation de ma partie

Mon rôle principal consiste à mettre en place une télémétrie par radio-fréquence en utilisant un module de communication nrf24L01+ de chez NordicSemiConductor.

Je dois adapter et faire évoluer le code réalisé l'an dernier qui assurait cette télémétrie par Bluetooth avec le module RN4020 de chez Microchip.

Les données de télémétrie concernent les niveaux de réfrigérant dans des cuves cryogéniques de différentes capacités disposées au sein d'une entreprise.

Les dispositifs implantés côté cuve, nommés « coffrets périphériques », et qui transmettent sur demande les données sont constitués d'une Raspberry Pi et d'une carte d'extension disposant d'un module NRF24L01+ pour la transmission RF, d'une RTC pour disposer d'un moyen d'horodatage même si la Raspberry Pi n'est pas connectée à internet et d'un convertisseur analogique/numérique auquel est relié la sonde développée par 2SPElectronic.

Le dispositif central, nommé « coffret maître », qui interroge les cuves est constitué d'une Rapsberry Pi connectée à internet ainsi que de la même carte d'extension que celle présente sur les coffrets périphériques sauf que celle-ci ne sera pas équipée avec la RTC et le convertisseur analogique/numériques car ils sont inutiles dans ce contexte.

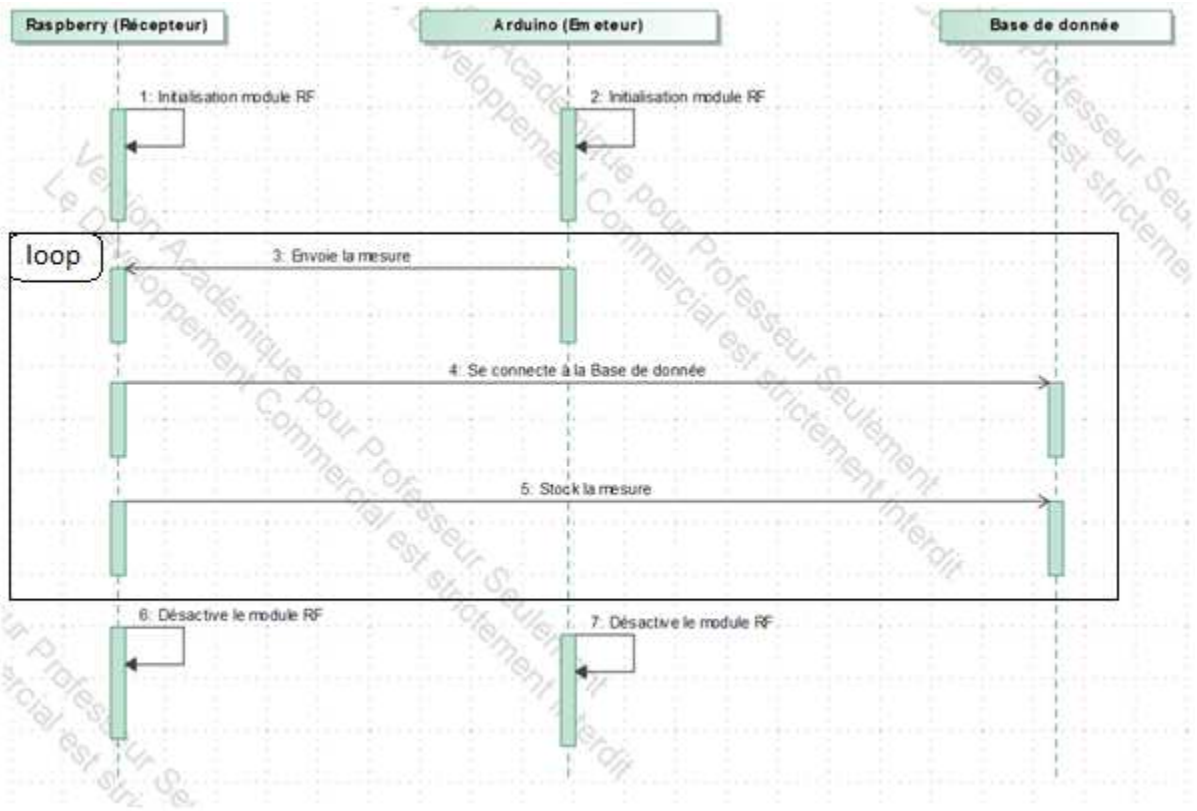
Les mesures effectuées à l'initiative du coffret maître sont archivées dans une BDD à des fins de traçabilité.

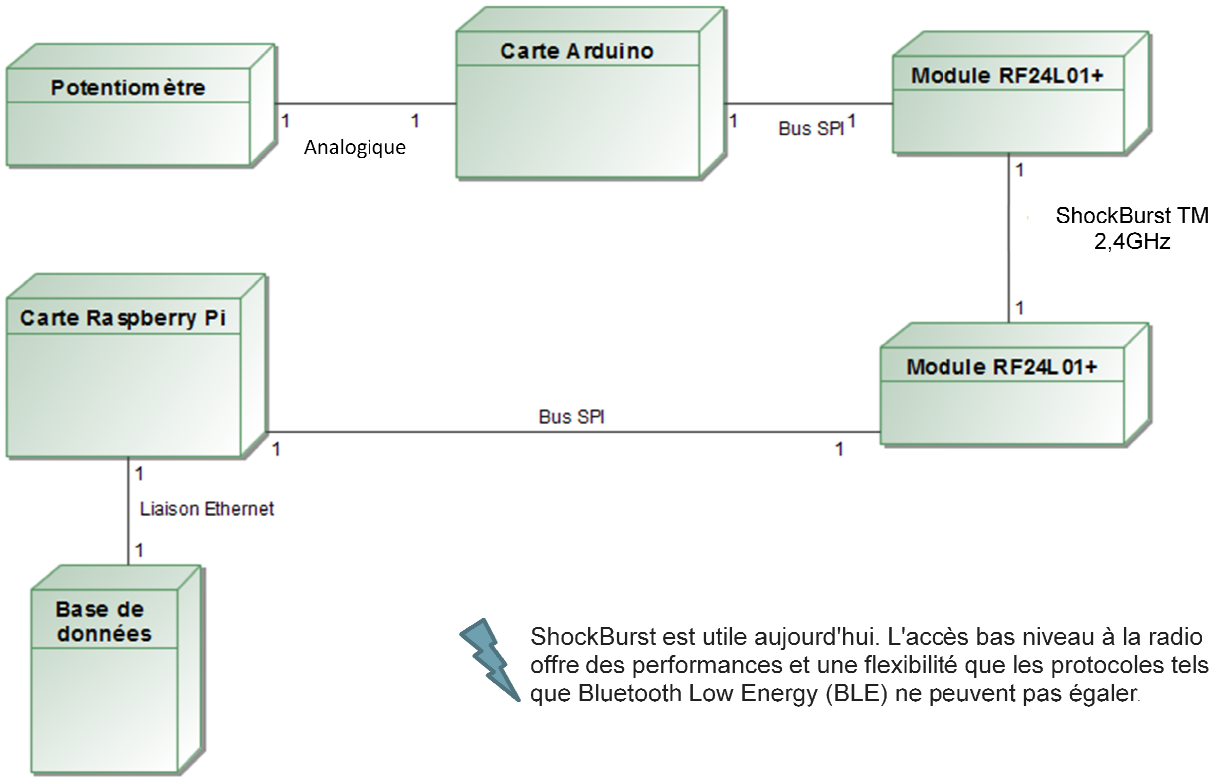
L'exploitation des données est réalisée à l'aide d'une application web sur laquelle travaille mon camarade Enzo.

Afin de pouvoir développer/tester mon code avant de pouvoir disposer de la carte d'extension réalisée par mes camarades d'EC, j'utilise un module breakout NRF24L01+ câblé en filaire sur ma Raspberry Pi Maître.

Le coffret périphérique est quant à lui émulé par une carte Arduino à laquelle sont reliés un module breakout NRF24L01+ et un potentiomètre qui joue alors le rôle de l'ensemble sonde cryogénique/convertisseur analogique/numérique.

2/ Diagrammes pour mieux comprendre mon travail





ShockBurst est utile aujourd'hui. L'accès bas niveau à la radio offre des performances et une flexibilité que les protocoles tels que Bluetooth Low Energy (BLE) ne peuvent pas égaler.

3/ Plan des itérations

Itérations	Désignation	Justification
1	• Concevoir/Coder/Tester la librairie nRF24 ➢ Installer la librairie du module RF24 ➢ Tester les capteurs avec les exemples ➢Trame/écriture dans la base de donnée	• Vérifier la bonne communication entre les deux modules • Montrer le début du raisonnement grâce aux méthodes fonctionnelles présentées et ainsi avoir une vision globale sur la partie à réaliser
2	• Coder le protocole Maitre<X<Esclave ➢ Les informations des cuves doivent s'écrire dans la base de donnée correctement ➢Création des classes pour le module RF24 ainsi que la base de donnée	• Assurer la bonne communication des deux modules • Assurer une écriture dans la base de donnée correcte pour pouvoir avoir accès aux informations
3	• Coder IHM ➢Démonstration fonctionnel du projet final ➢Diagramme de classe final	• Assurer une communication complète des données entre les modules et la base de donnée •L'Inteface Homme Machine doit être opérationnel pour que l'utilisateur puisse voir toutes les informations des cuves

4/ Itérations 1 et 2

Installation de l'environnement de travail

Pour commencer, Il faut installer une image compressée de l'OS "Raspbian Stretch Lite" sur une carte SD, qui par la suite ira dans la Raspberry Pi.

Après avoir installé l'image compressée de l'OS sur la carte SD il faut:

- Configurer la carte Raspberry, en activant les connexions SSH à distance, ainsi que l'option de prise en charge du bus SPI
- Changer le mot de passe pour plus de sécurité
- faire en sorte de pouvoir accéder à internet depuis le lycée sans avoir à reconfigurer cet accès à chaque mise sous tension. Ceci se résume à :
  - installer le paquet `resolvconf` avec la commande `sudoapt-get update ; sudoapt-getinstallresolvconf`.
  - renseigner ensuite le fichier `/etc/dhcpcd.conf` avec une directive `staticdomain_name_servers` à laquelle on fournit l'adresse IP du serveur DNS

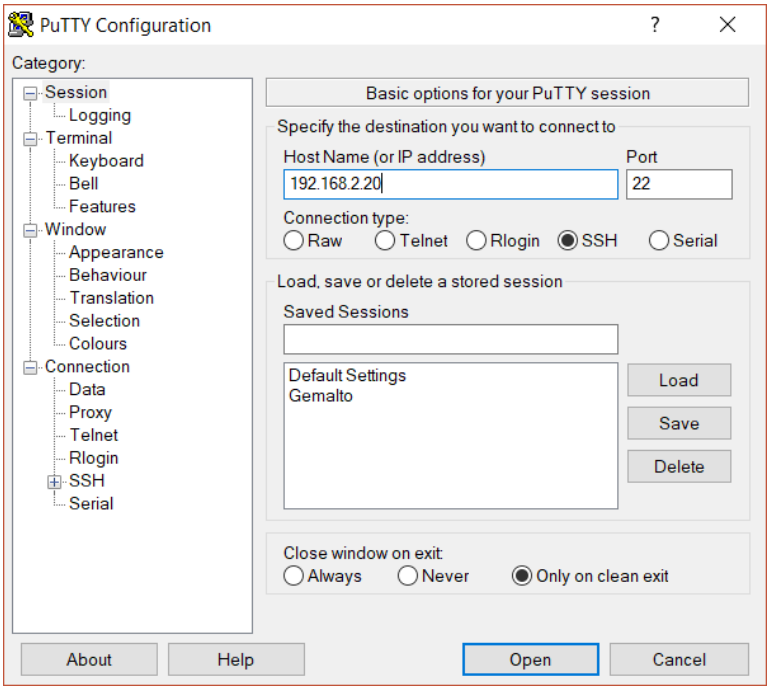
Pour pouvoir accéder au fichier de la RaspberryPi plus facilement depuis l'explorateur de fichier de ma machine de développement Windows, j'ai crée un partage de fichiers.

Pour cela il faut :

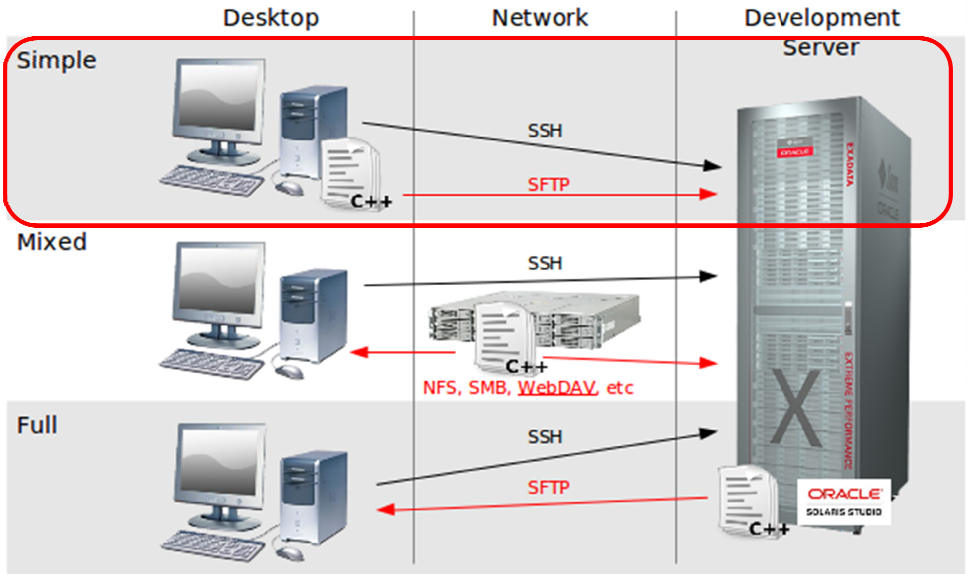
- Installer le serveur Samba sur la Raspberry PI
- Faire une copie de sauvegarde du fichier de configuration original
- Editer le fichier de configuration de samba `/etc/samba/smb.conf`pour créer un nouveau partage protégé par mot de passe

```
[Partage 2SP 2018]
comment = Partage Samba sur Raspberry Pi pour le projet 2SP session 2018
path = /home/pi/2sp_2018
writable = yes
#guest ok = yes
#guest only = yes
create mode = 0777
directory mode = 0777
share modes = yes
validusers = pi
```

Une connexion SSH est préférable pour l'accès à distance sur la Raspberry Pi. J'ai donc installé PuTTY qui fait office de client SSH et me permettra d'exécuter des commandes à partir du shell.



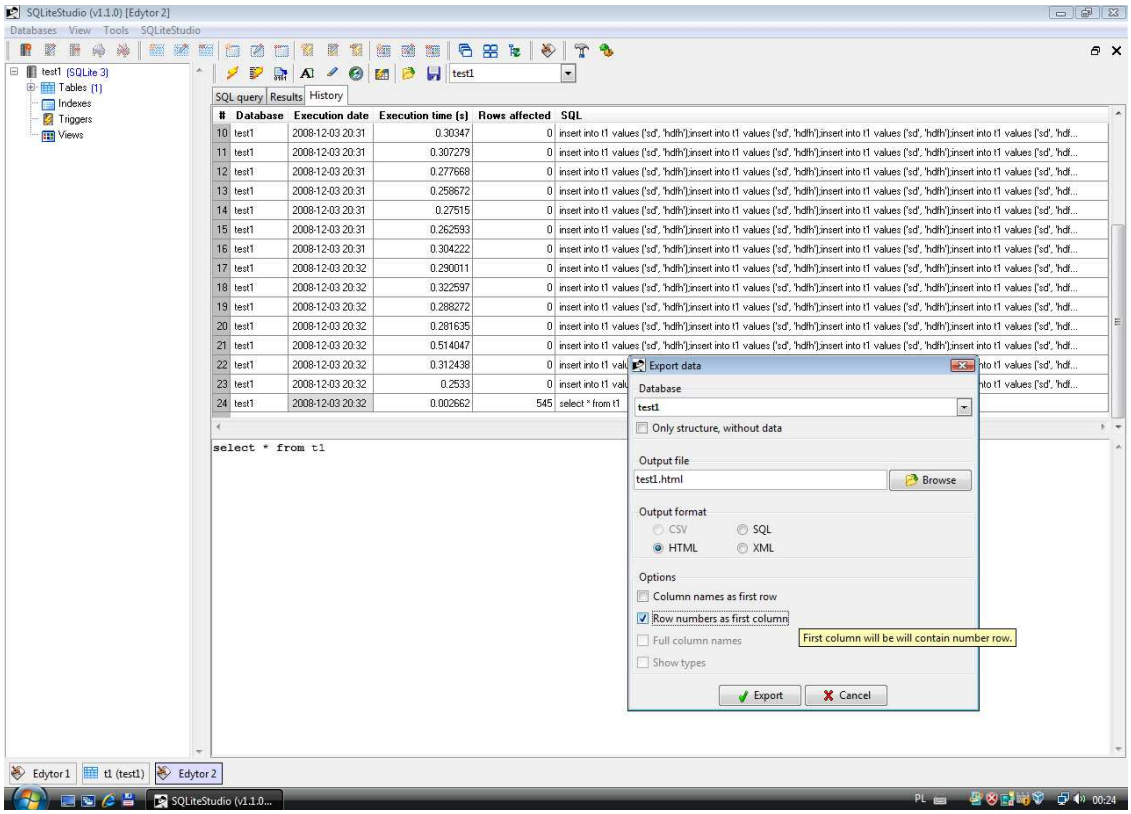
Le codage devant être réalisé en C++ standard, j'ai installé Netbeans qui est un environnement de développement intégré C++ d'Oracle (passé sous licence Apache récemment) qui autorise le développement à distance selon différents modes.



Dans le cadre du projet, nous utilisons le mode « Simple » :
Les fichiers source résident sur la machine de développement (mon PC). Lorsqu'on demande de construire l'exécutable, les fichiers sources sont tranmis sur la Raspberry Pi (« Development server » sur le schéma ci-dessus) via le protocole SFTP (protocole de transfert de fichiers de SSH).

Pour le développement côté Arduino, j'ai simplement installé l'IDE Arduino dans sa version 1.8.5

Pour visualiser/contrôler/maintenir la BDD SQLite, j'ai d'une part installé un client en ligne de commande sur la Raspberry Pi ainsi qu'un client graphique sur mon poste Windows (SQLite Studio).



Comparaison de deux librairies :

Essentiellement, 2 librairies C++ de gestion du NRF24L01+ compatibles Arduino/Raspberry existent : **RF24** (développée par TMHR20) et (développée par Mike McCauley).

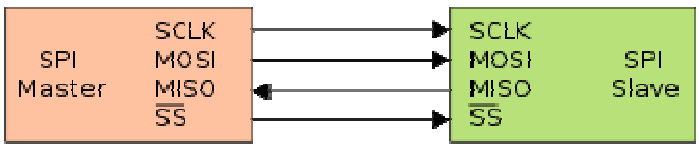
J'ai donc comparé ces 2 librairies pour déterminer celle que j'allais utiliser dans le projet

	RadioHead	TMRH20
Aperçut	Pilotes (RH_NRF24)/Gestionnaires	
Avantages	-Instancier + de un pilote et gestionnaire -RH_NRF24 compatible -28 octets sur n'importe quelle fréquence	-Fiable, réactif, sans bug et riche en fonctionnalités -Simple utilisation -format adressage de 24,32,40 bits -RF24MESH
Inconvénient	-Transport non adressé et non fiable via les installations du Driver	

Lorsqu' 'il est utilisé sur la Raspberry Pi, les 2 bibliothèques s'appuient sur la librairie bcm2835 de Mike McCauley pour la communication pas bus SPI avec le NRF24L01+.

Etude du bus SPI (Serial Peripheral Interface)

Le SPI est un bus de données série synchrone full-duplex qui fonctionne en mode maître-esclave. Le maître peut communiquer avec plusieurs esclaves moyennant l'utilisation d'une broche de sélection (Slave Slelect)

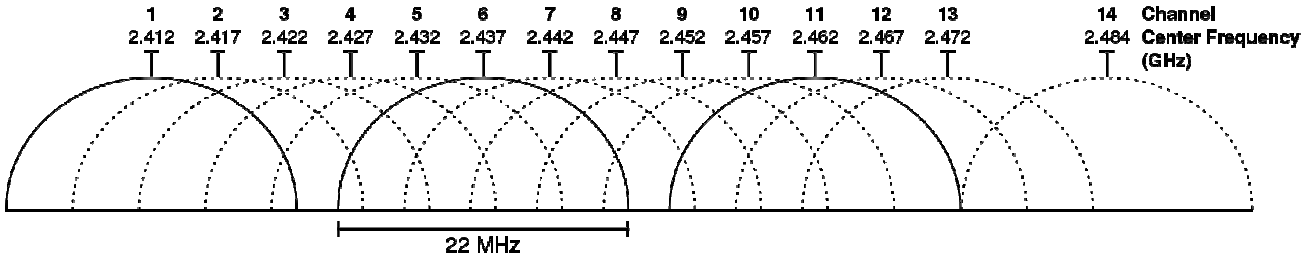


Avantages : Il dispose du Full Duplex, son débit peut être plus élevé : il est possible d'atteindre 100Mbit/s.

Inconvénients : Monopolise plus de broche que I2C ou UART, il n'y a pas d'adressage, pas d'acquittement et ne peut fonctionner que sur de courtes distances.

Etude du module NRF24L01+

Le module fonctionne sur la bande de fréquence 2,4GHz



Il offre un bon compromis entre la consommation d'énergie, le débit et la portée.

Le module NRF24L01+ s'alimente de 1.9V à 3.6V.

Il permet une communication sur une distance d'environ 100m à 800m.

Le module dispose de plusieurs broches qui sont les suivantes :

MOSI : Données de la carte (Arduino/RPi) vers le module (broche 6)

MISO : Données provenant du module vers la carte maître (Aruino/RPi) (broche 7)

SCK : Horloge (broche 5)

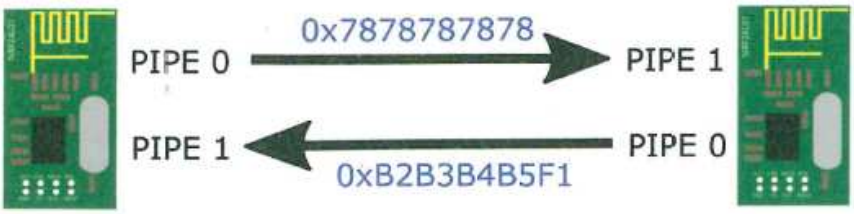
CSN : Demande au module de prendre compte les instructions SPI (broche 4)

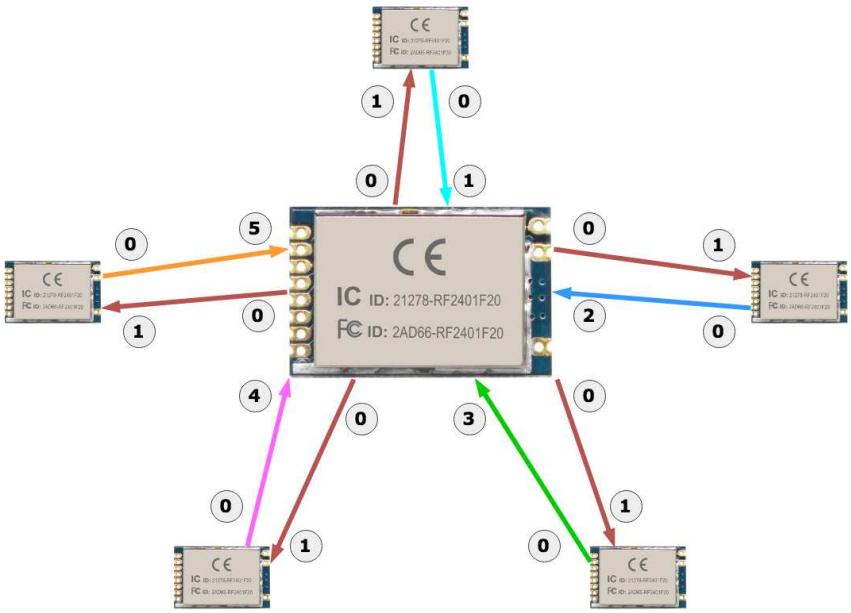
CE : Active l'émetteur radio (broche 3)

VCC : 3.3V (broche 2)

GND : (broche 1)

Le module dispose de 6 pipes qui sont des canaux de communication pour établir des connexions avec d'autres modules. Le pipe 0 est utilisé pour l'émission alors que les autres ne peuvent être utilisés qu'en réception.

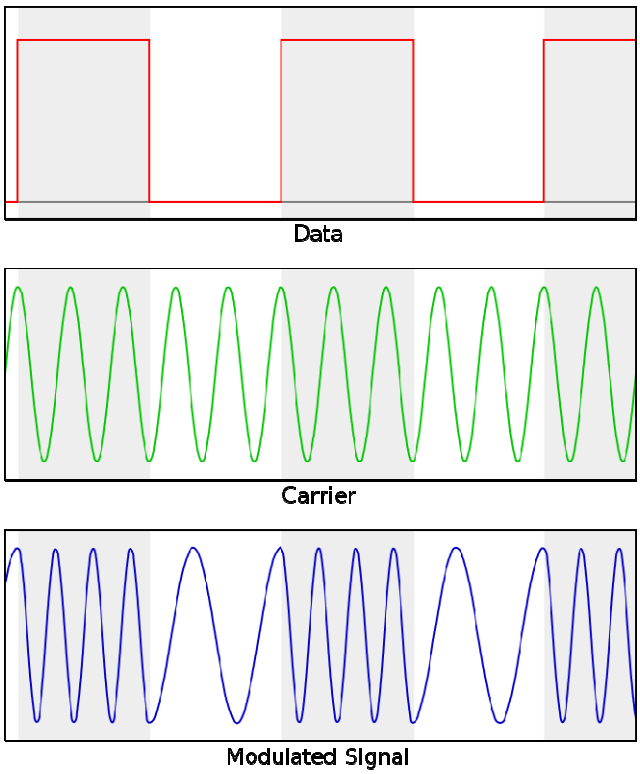


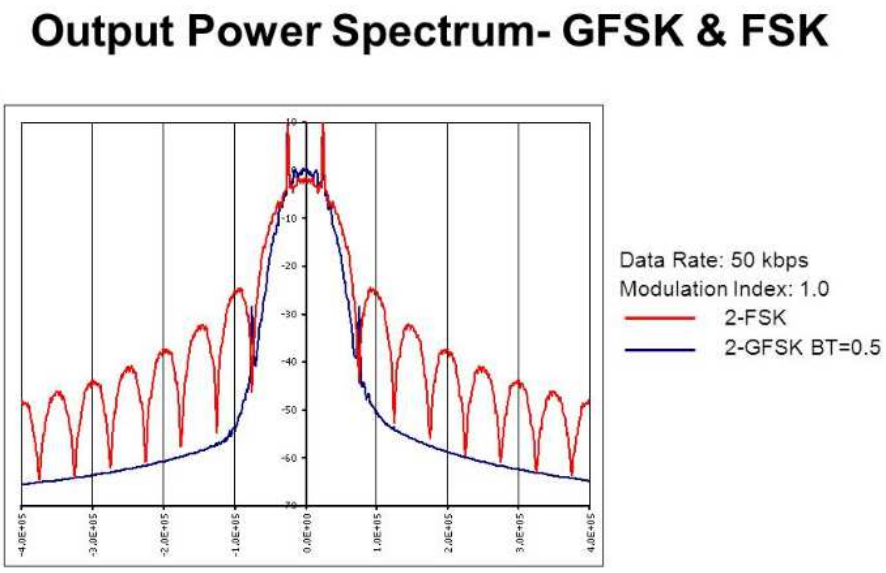


Le module utilise la modulation GFSK dite fréquence gaussienne qui filtre les impulsions de données avec un filtre gaussien pour rendre les transitions plus lisses. Ce filtre a l'avantage de réduire la bande passante.

Un modulateur GFSK diffère d'un simple modulateur de modulation par déplacement de fréquence car avant que la forme d'onde de base (niveaux -1 et +1) ne passe dans le modulateur FSK, il passe par un filtre gaussien pour rendre les transitions plus fluides afin de limiter sa largeur spectrale.

Exemple théorique du FSK :





La GFSK permet donc d'atténuer les ondes perturbantes et par la suite de se focaliser uniquement sur la raie principale.

Tableau comparatif des valeurs théoriques :

	WIFI	BLUETHOOTH	SHOCKBURST
Bande de fréquence (GHz)	2.4	2.4	2.4
Débits (Mbps/s)	2	1	1
Canaux de communication	Canaux 7 à 10	79 canaux	126 canaux
Nombre de bits		48	32
Portée (m)	140	100	100

La sensibilité pour 1Mbps est de -85dBm

G=20 log (p/10^-3) -> P=10^-3*10^G/20

Pour une écoute minimal noté P on a donc 0.056µs

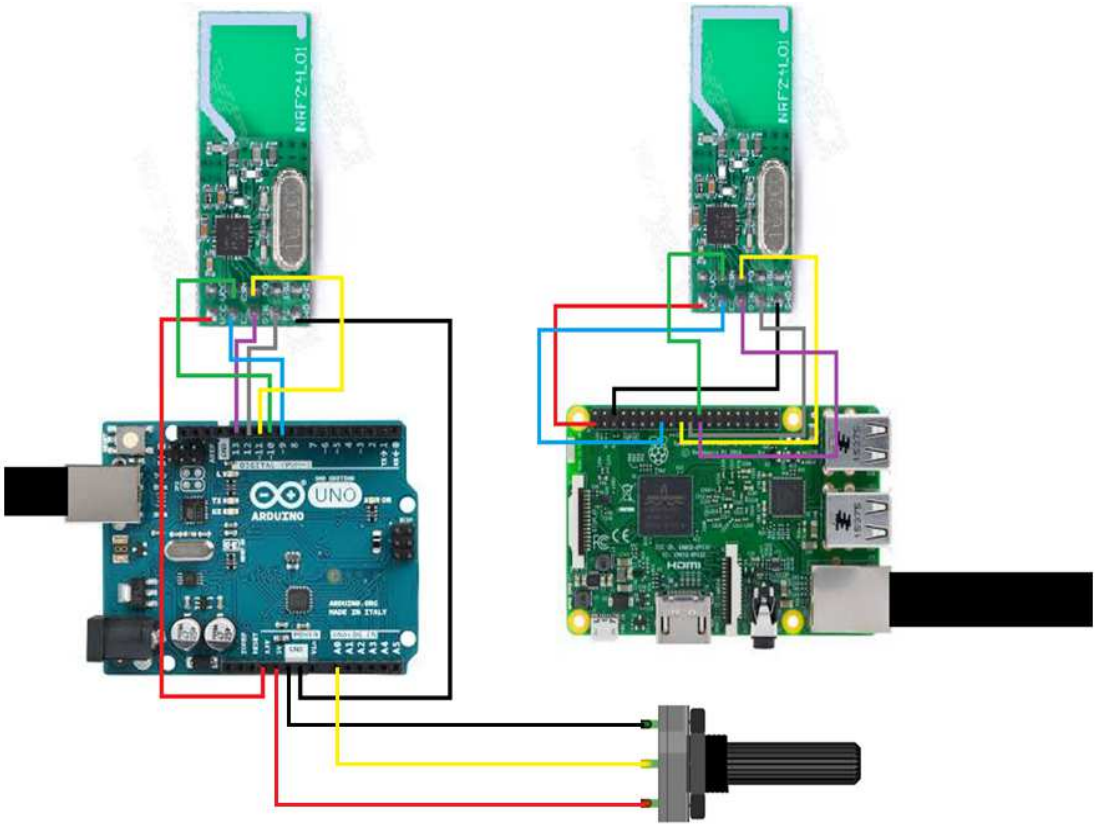
Câblage des modules et communication

Pour commencer concrètement mon travail, j'ai mis en œuvre l'exemple de base livré avec la librairie, j'ai câblé un module NRF24L01+ sur une carte ArduinoUno qui jouait le rôle d'émetteur de mesures. Un deuxième module a été câblé sur une carte Raspberry Pi qui jouait le rôle de récepteur.

A terme, la Raspberry Pi doit envoyer les données reçues dans la BDD SQlite du projet.

Une carte Arduino a été utilisée pour l'émetteur car une fois le programme flashé, il suffit de mettre la carte sous tension pour disposer d'une source de données transmises avec le protocole Shockburst. Cela m'a permis de mettre en place plus facilement l'architecture d'application du coffret maître qui reçoit les données, les extraits, puis les sauvegardes dans la BDD.

Un potentiomètre est raccordé à la carte Arduino pour pouvoir simuler des variations de mesures de cuves.



Choix du canal de communication.

Le nrf24l01 opère sur un canal radiofréquence de la bande des 2.4GHz. Il y a en tout 125 canaux possibles qui occupent une largeur de 1MHz lorsqu'on utilise une vitesse de 1Mbps. Par défaut, dans la librairie RF24, le canal utilisé est le n°76 mais il est possible de le changer avec la méthode `setChannel()`.

Ci dessous l'extrait de la datasheet du composant qui traite de ce sujet :

[...]
RF channel frequency (§6.3, p.23 de la datasheet)

The RF channel frequency determines the center of the channel used by the nRF24L01. The channel occupies a bandwidth of 1MHz at 1Mbps and 2MHz at 2Mbps. nRF24L01 can operate on frequencies from 2.400GHz to 2.525GHz. The resolution of the RF channel frequency setting is 1MHz.

At 2Mbps the channel occupies a bandwidth wider than the resolution of the RF channel frequency setting. To ensure non-overlapping channels in 2Mbps mode, the channel spacing must be 2MHz or more. At 1Mbps the channel bandwidth is the same as the resolution of the RF frequency setting.

The RF channel frequency is set by the RF_CH register according to the following formula:
F0= 2400 + RF_CH [MHz]

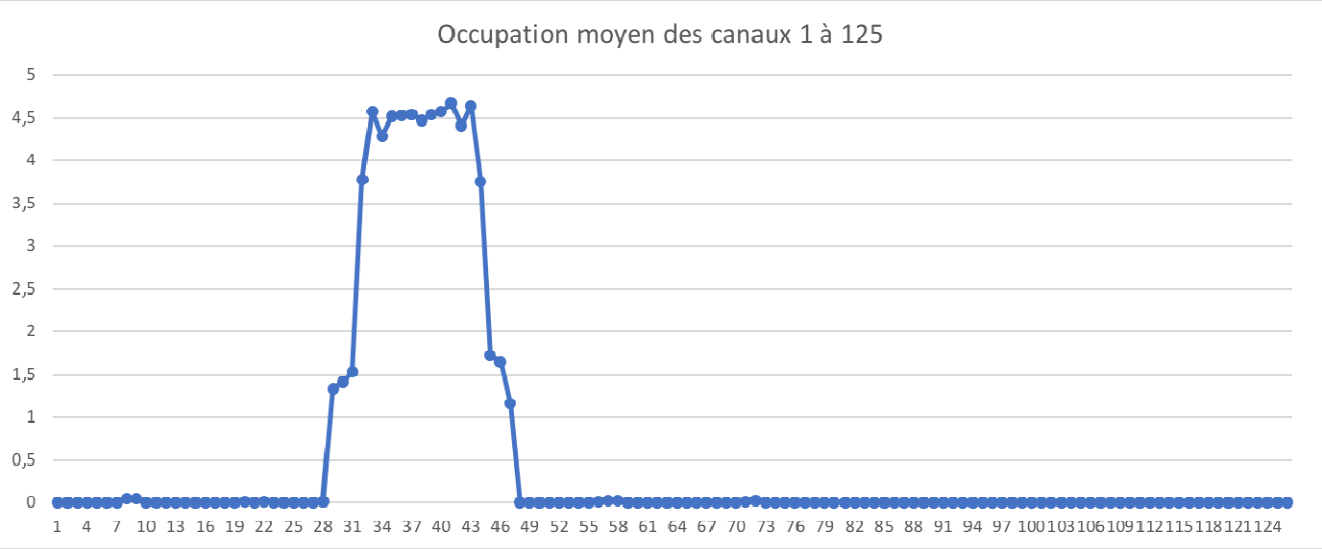
A transmitter and a receiver must be programmed with the same RF channel frequency to be able to communicate with each other.
[...]

Tout cela pour dire qu'il faut s'assurer que le canal utilisé par le NRF24L01+ n'est pas déjà pris par autre chose (Routeur WiFi par exemple ou alors équipement Bluetooth) sous peine d'avoir des interférences.

A cette fin, la librairie RF24 vient avec un exemple appelé "scanner". Ce programme parcourt en boucle les 125 canaux possibles et détecte la puissance des signaux présents sur chacun d'eux puis affiche ce niveau sous forme d'un nombre allant de 0x00(pas de signal détecté) à 0x0F(signal fort détecté).

On peut alors exploiter ces résultats dans Excel pour tracer une courbe d'occupation des canaux.

Ci-dessous la courbe exprime l'occupation moyenne des canaux sur un ensemble de 68 mesures (68 boucles dans le programme) et on voit nettement que les canaux 27 à 48 sont très utilisés donc à éviter pour l'utilisation du NRF24L01+.

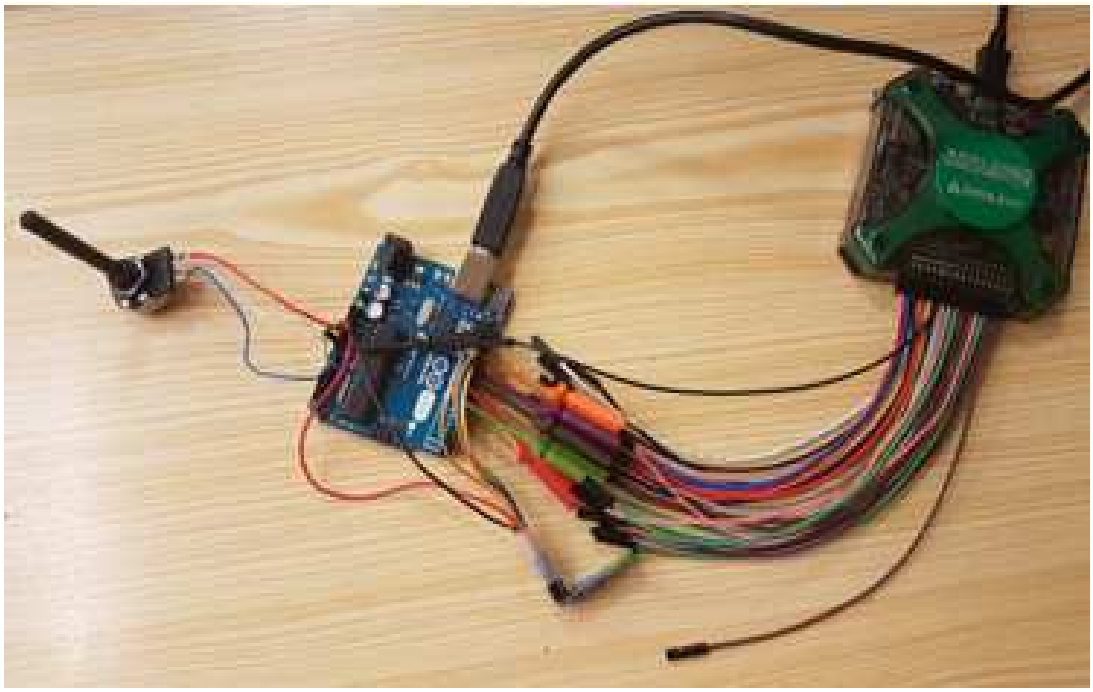


Tests du module NRF avec AnalogDiscover

Nous disposons d'un analyseur logique appelé **AnalogDiscovery**.

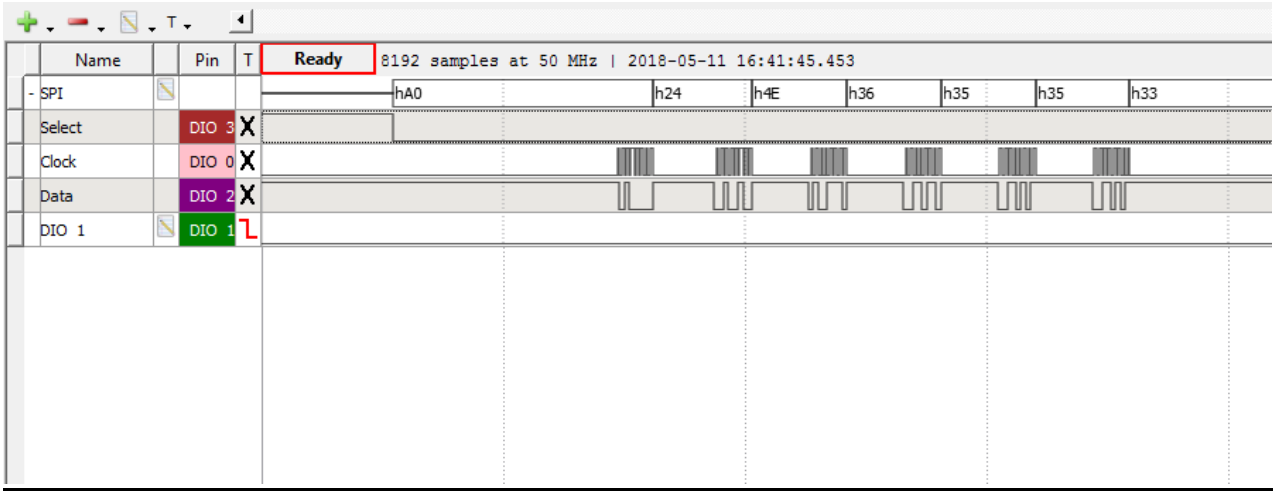
Celui-ci nous a permis d'observer les trames SPI que l'ArduinoUno envoie au module nrf24.

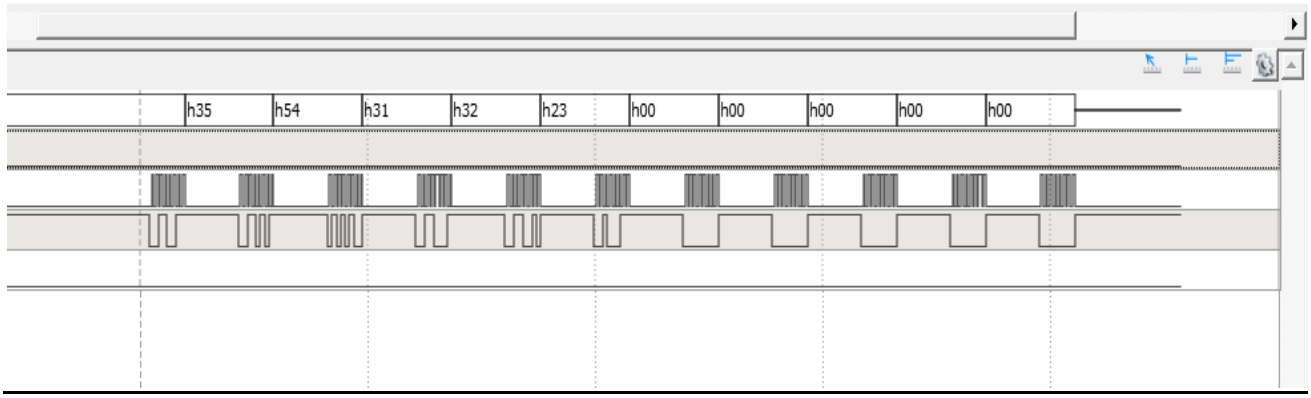
Le câblage est montré ci-dessous :



Pour que l'on puisse pouvoir obtenir la partie de la trame désirée, j'ai dû générer une impulsion sur une broche digitale de l'Arduino qui m'a servi pour déclencher la capture de trame.

On peut observer ci-dessous le résultat obtenu avec le logiciel Waveforms :





Ci-dessus on peut observer que la trame a été relevée correctement et notée en hexadécimal. Depuis la table ASCII on peut traduire les données :

hA0 : début trame	h24 : caractère $	h4E : caractère N	h36 : caractère 6	h35 : caractère 5	h35 : caractère 5
h33 : caractère 3	h35 : caractère 5	h54 : caractère T	h31 : caractère 1	h32 : caractère 2	h23 : caractère #

Ci-dessus est la trame qu'Arduino envoie au module nrf24 pour qu'il puisse la transmettre.

Partie de code rédigée en langage C

Premièrement il a fallu m'approprier le code d'exemple fournit avec la librairie rf24, télécharger et porter quelques modifications pour qu'il puisse fonctionner.

Raspberry

```
//On déclare les librairies pour que les fonctions soient reconnues.
#include <cstdlib>
#include <iostream>
#include <sstream>
#include <string>
#include <unistd.h>
#include <stdio.h>
#include <RF24/RF24.h>

using namespace std;

int main(int argc, char** argv){
    //Le tableau temps qui contient les données de la trame
    char temps[12];
    //Définition des canaux de communication
    const uint8_t pipes[][6] = {"1Node","2Node"};

    RF24 radio(22,0);
    radio.begin();
```

Arduino

```
const int analogPin = A0;
//On definie le début de la trame avec $N et la fin de trame T123# qui reseront
fixe et seulement data varie avec la valeur du potentiomètre.
char frame[16];
char prefix[] = "$N";
char data[5 + 1];
char queue[] = "T12#";
int sensorValue;
unsigned long niveau;

void setup() {
  Serial.begin(115200);
```

Raspberry

```
//Réglage des canaux (pipe) qui ne faut surtout pas oublier de croiser car sans
cela, on aurait des confrontations et les données seraient mal lues.
  radio.openWritingPipe(addresses[0]);
  radio.openReadingPipe(1, addresses[1]);
```

Arduino

```
// Réglage des canaux (pipe)
    radio.openWritingPipe(pipes[1]);
    radio.openReadingPipe(1,pipes[0]);
```

Arduino

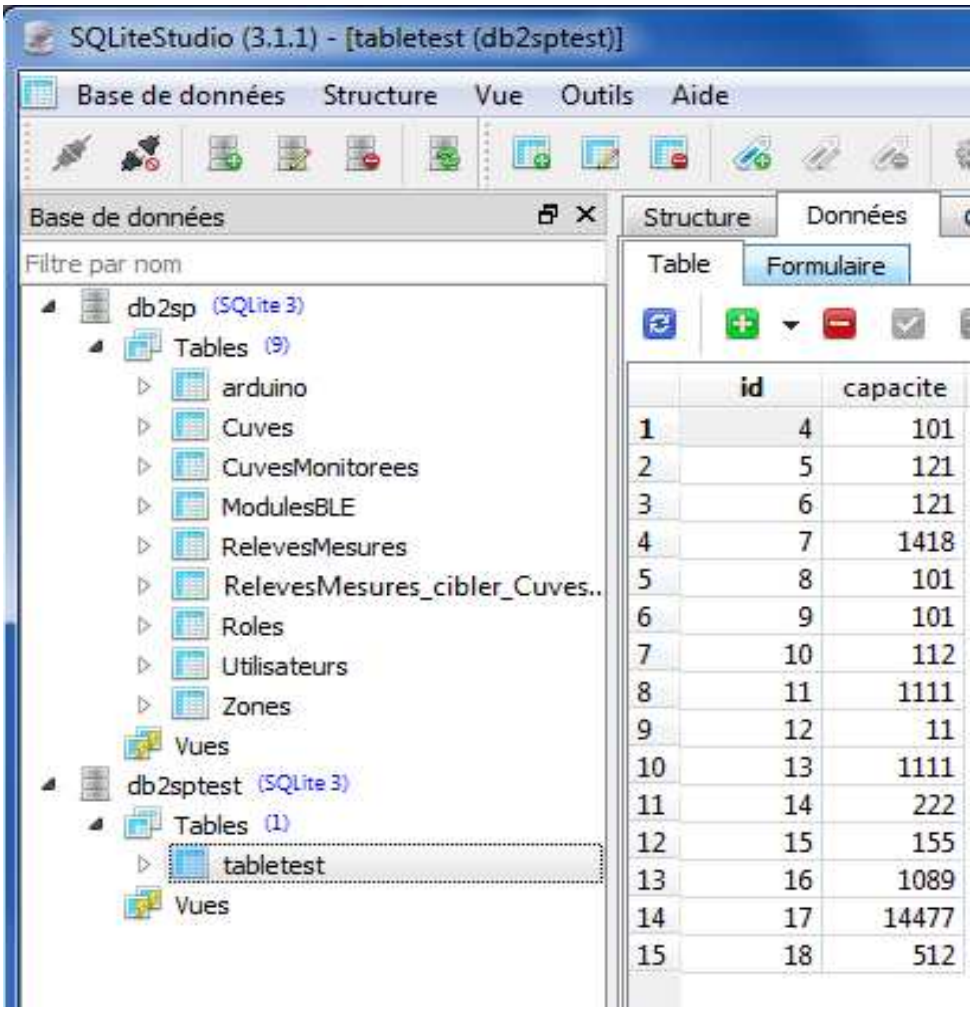
```
void loop() {
  // Arrêt écoute
  void loop() {
  // Arrêt écoute
  sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  //La valeur du potentiomètre est codée sur 10 bits donc varie de 0    à 1023. Le convertisseur analogique/numérique utilisé sur la carte d'extension est un convertisseur 16 bits donc on utilise la fonction map() pour la faire varier de 0 à 65 535
  niveau = map(sensorValue, 0, 1023, 0, 65535);
  Serial.println(niveau);
  ltoa(niveau, data, 10);

  strncpy(&frame[2], data, 5);

  sprintf(frame, "%s%s%s", prefix, data, queue);

  radio.stopListening();
  Serial.println(F("Envoi"));
  Serial.println(frame);
```

Raspberry

```
namespace {
    //On accède à la base de données.
    const char * DB_NAME = "/home/pi/2sp_2018/db2sptest.db";
}


//On ouvre la base de données.
ret = sqlite3_open_v2(DB_NAME, &db, SQLITE_OPEN_READWRITE, NULL);

    if (ret != SQLITE_OK) {
        cerr << "[" << __FILE__ << ", " << __func__ << "()] Echec sqlite3_open() : " << sqlite3_errstr(ret) << endl;
        throw runtime_error("Echec ouverture de la BDD");
        //return 0;
    } else {
        cerr << "BDD ouverte" << endl;
    }
```

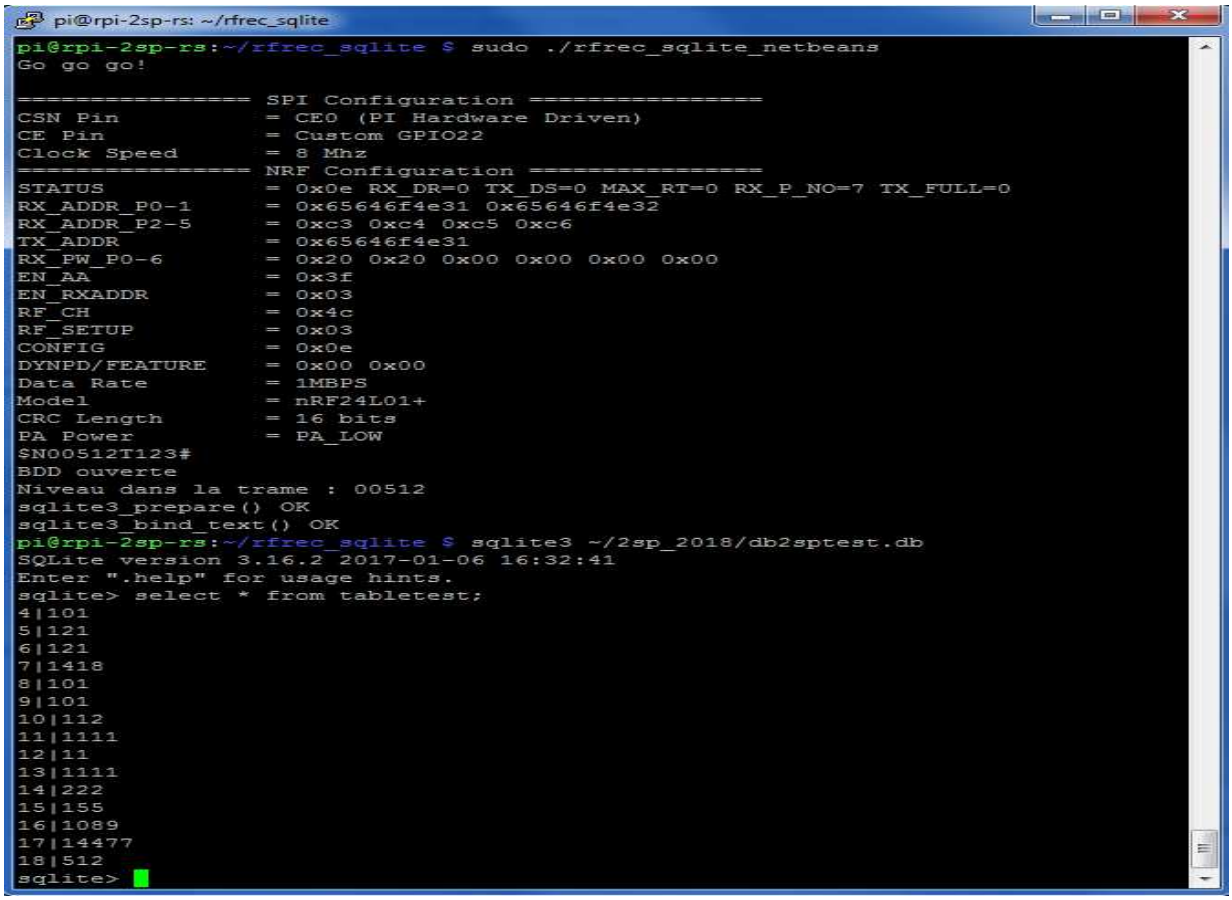
```
//On prepare requête SQL
sqlite3_prepare_v2(db, "insert into tabletest (capacite) values (?1);", -1, &stmt,
NULL);
    if (ret != SQLITE_OK) {
        cerr << "[" << __FILE__ << ", " << __func__ << "()] Echec
sqlite3_prepare() : " << sqlite3_errstr(ret) << endl;
        //return 0;
    } else {
        cerr << "sqlite3_prepare() OK" << endl;
    }
    sqlite3_bind_text(stmt, 1, strNiveau, -1, SQLITE_STATIC);
    if (ret != SQLITE_OK) {
        cerr << "[" << __FILE__ << ", " << __func__ << "()] Echec
sqlite3_bind_text() : " << sqlite3_errstr(ret) << endl;
        //return 0;
    } else {
        cerr << "sqlite3_bind_text() OK" << endl;
    }


// On exécute la requete
    ret= sqlite3_step(stmt);
    if (ret != SQLITE_DONE) {
    printf("ERROR inserting data: %s\n", sqlite3_errmsg(db));
    sqlite3_close(db);
    printf("Fermeture BDD\n");
```

Ci-dessous on observe le résultat de la requête SQL avec l'application SQLiteStudio :



	id	capacite
1	4	101
2	5	121
3	6	121
4	7	1418
5	8	101
6	9	101
7	10	112
8	11	1111
9	12	11
10	13	1111
11	14	222
12	15	155
13	16	1089
14	17	14477
15	18	512

Ci-dessous figure le même résultat mais cette fois-ci en utilisant le clientsqlite3 depuis le terminal de la raspberry pi :

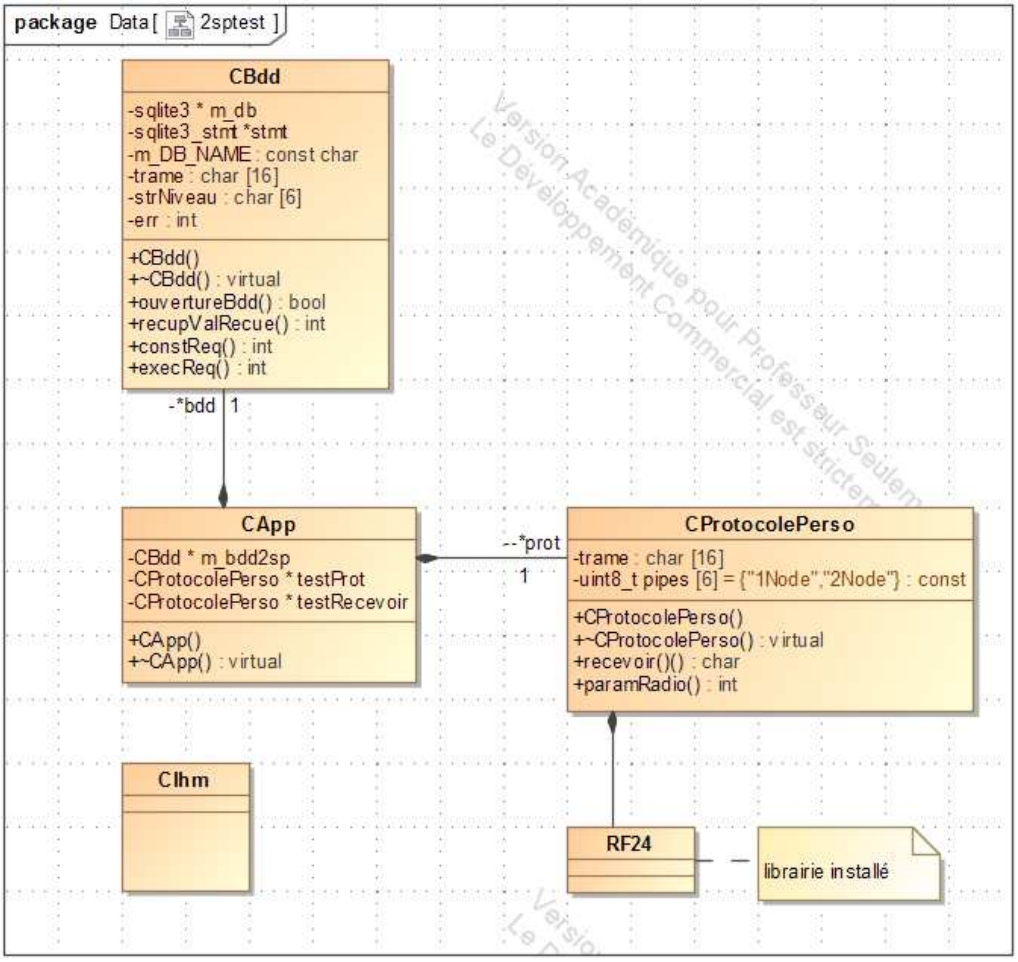


On peut constater sur cette capture d'écran de la Raspberry pi, que la trame de la carte Arduino a bien été transmise et reçu par la Raspberry pi. Ici, on relève un niveau de trame qui est égale à 512 et cette valeur a bien été écrite dans la base de données sqlite3 à l'id 18.

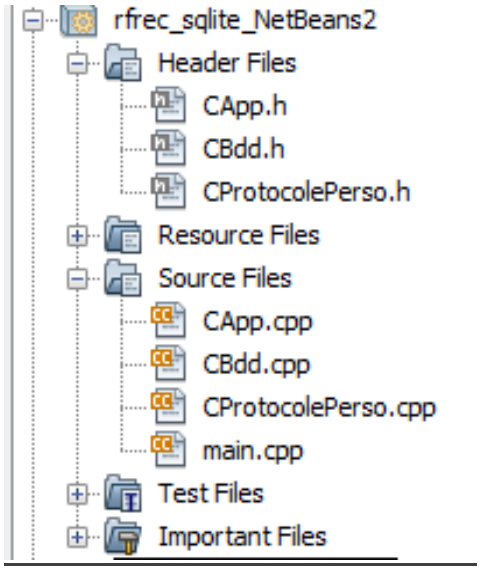
Partie code rédigé en langage C++

Après avoir eu à coder en langage C, il faut absolument coder en langage C++ qui est un langage bien plus structuré.

Voici ci-dessous le diagramme de classe :



Voici l'arborescence de fichiers correspondante sous Netbeans :



Ci-dessous un exemple de code provenant de CProtocolePerso.h :

```
#ifndef CPROTOCOLEPERSO_H
#define CPROTOCOLEPERSO_H

#include "RF24/RF24.h"

class CProtocolePerso {
public:
    CProtocolePerso();
    virtual ~CProtocolePerso();
    char* recevoir();
    int paramRadio();
    int NumtoTemp(char *strNvieau);
private:
 //   RF24 * radio;
    char trame[16];
    const uint8_t pipes[2][6] = {"1Node","2Node"};
};

#endif /* CPROTOCOLEPERSO_H */
```

Le fichier .h permet de déclarer les classes avec leurs méthodes et attributs. On retrouve le bout de code en C précédemment qui a été repris et structuré dans le .h .

Ci-dessous un exemple de code provenant de CProtocolePerso.cpp :

```
#include <iostream>
#include "CProtocolePerso.h"

using namespace std;

RF24 radio(22,0);

//Ceci est le constructeur de CProtocolePerso et qui va lancer la méthode
paramRadio() avec son contenu qu'il en contient.
CProtocolePerso::CProtocolePerso() {
    paramRadio();
}

//Ceci est le desctructeur qui va arreter l'écoute en cours des modules.
CProtocolePerso::~CProtocolePerso() {
    radio.stopListening();
}
```

Ci-dessous on peut observer le contenu de la méthode paramRadio() qui consiste à initialiser le module nrf24 pour qu'il puisse par la suite communiquer.

```
int CProtocolePerso::paramRadio(){
    radio.begin();
    radio.setRetries(15,15);
    // Affichage d'un résumé de la configuration
    radio.printDetails(); //to stdout
    // Activer l'accusé réception
    radio.enableAckPayload(); // 0&1
    // L'accusé réception est dynamique
    radio.enableDynamicPayloads(); // ture
    // Réglage de l'amplificateur
    radio.setPALevel(RF24_PA_LOW); // 1 à 4 niveau
    // Réglage des canaux (pipe)
    radio.openWritingPipe(pipes[0]);
    radio.openReadingPipe(1,pipes[1]);
    // reprise de la réception
    radio.startListening();
    cerr << "Module en communication" << endl;

}
```

Ci-dessous, le fichier .cpp permet d'implémenter ce que l'on a déclaré dans le .h. Cette méthode permet simplement de recevoir la trame.

```
char* CProtocolePerso::recevoir() {
    bool quit = false;

    do{
        if (radio.available() ) {
            //oui, lecture
            radio.read(&trame, 16);

            cerr << "Valeur de la trame = " << trame << endl;
            cerr <<"caractere = "<< trame[0]<< endl;
        }

    }while(trame[0] != '$');

    int i = 1;
    cerr << "do while ok" << endl;
    cerr << "debut while " << endl;
    while( !quit ) {
            cout << "c[ " << i << "] = " << trame[i] << endl;
            if(trame[ i ]  == '#') {
                quit = true;
            }
            i++;
    }
```

Le résultat donne ceci :

```
Module en communication
Valeur de la trame = $N65535T12#
caractere = $
do while ok
debut while
c[ 1] = N
c[ 2] = 6
c[ 3] = 5
c[ 4] = 5
c[ 5] = 3
c[ 6] = 5
c[ 7] = T
c[ 8] = 1
c[ 9] = 2
c[ 10] = #
trame: $N65535T12#
$.N.6.5.5.3.5.T.Niv dans la trame : 65535
BDD ouverte
sqlite3_prepare() OK
sqlite3_bind_text() OK
Requete OK!
BDD fermée
Suppression Instruction Préparé
pi@rpi-2sp-rs:~/rfrec_sqlite2 $
```

On peut voir la trame qui a bien été retransmise, on a bien récupéré la partie de la trame qui nous intéresse et on l'a bien écrite dans la base de données :

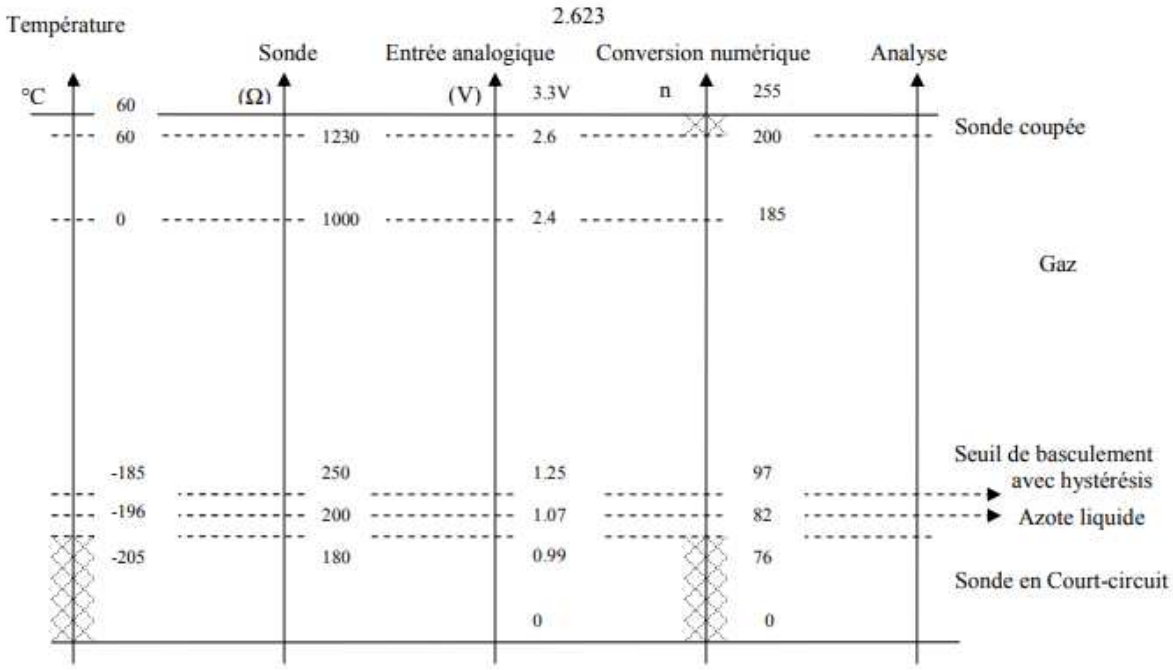
```
pi@rpi-2sp-rs:~/rfrec_sqlite2 $ sqlite3
SQLite version 3.16.2 2017-01-06 16:32:41
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open /home/pi/2sp_2018/db2sptest.db
sqlite> select * from tabletest;
37|65535
38|46893
39|30301
40|0
41|2882
42|13324
43|21652
44|65535
sqlite>
```

Après avoir fini de retranscrire mon programme du langage C en langage C++ il faut désormais passer au concret.

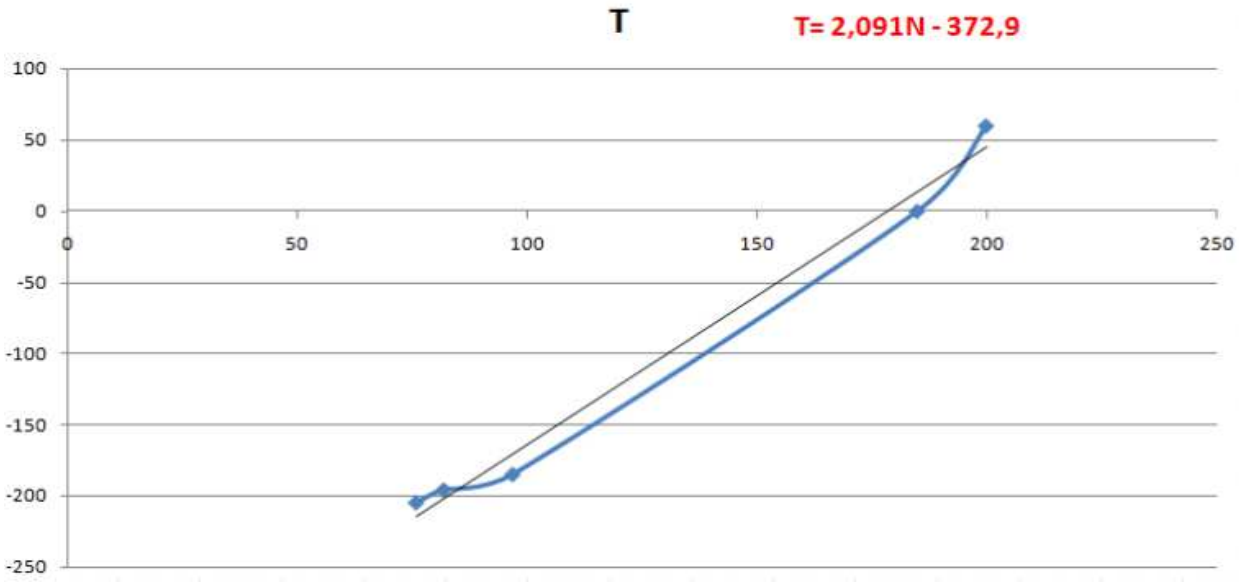
La sonde intégrée dans la cuve nous transmet des valeurs numériques. Pour pouvoir comprendre ce à quoi elles correspondent, il faut les convertir en température.

Ci-dessous on peut voir un tableau de correspondance fournit par 2SPElectronic :

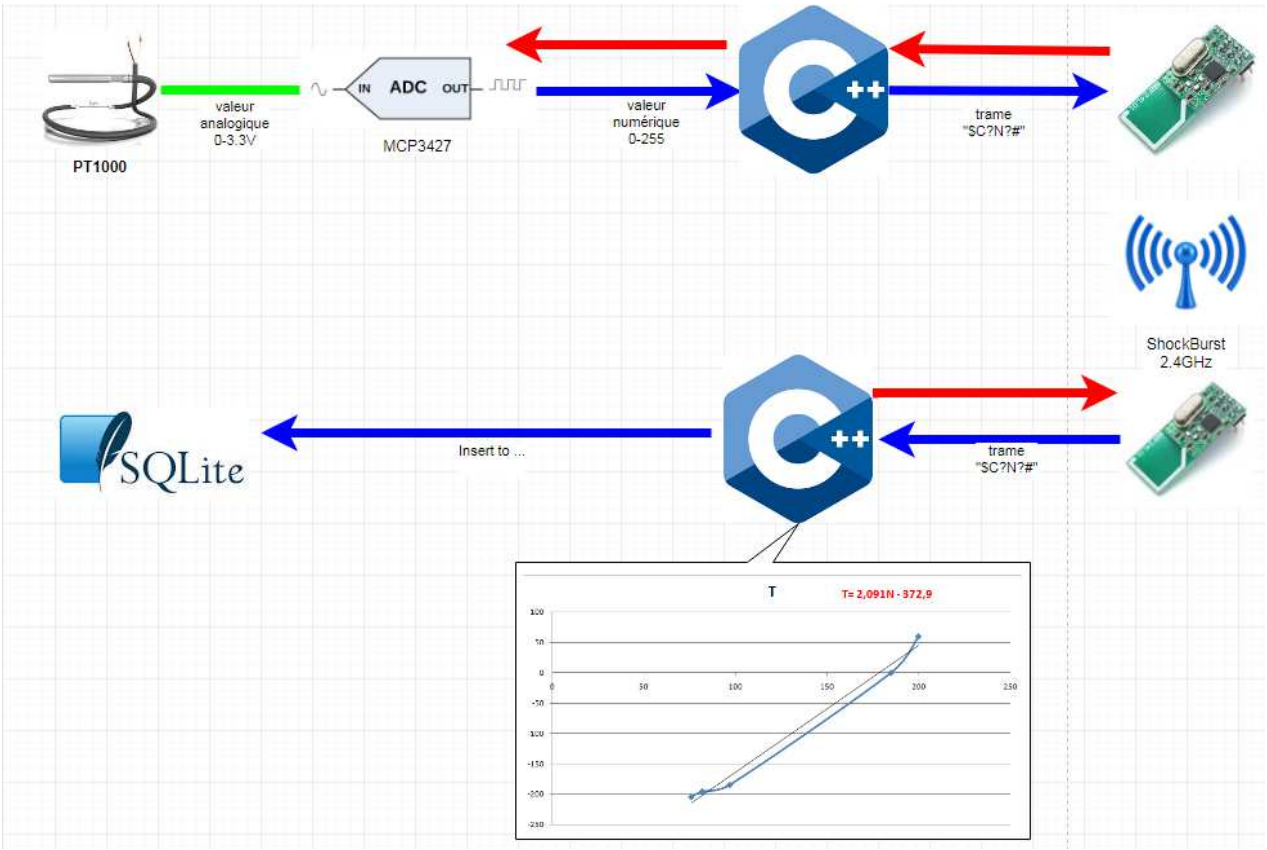
Sondes de niveau



Après avoir analysé ce tableau, j'ai réalisé un tableau Excel qui m'a montré que la correspondance suivait une courbe linéaire. Excel m'a permis de récupérer l'équation de la droite. J'ai ensuite utilisé cette équation dans mon programme pour réaliser le calcul de conversion :



Après avoir obtenu la fonction, il faut désormais remettre en ordre les idées et j'ai réalisé un synoptique que l'on peut voir ci-dessous :



La partie du code qui consiste à réaliser la conversion se trouve dans le fichier CProtocolePerso.cpp qui est ci-dessous :

```
int CProtocolePerso::NumtoTemp(char *strNiveau) {
    int T = 2.091*atoi(strNiveau) - 372.9;
    cout << "Temp: " << T << endl;

    if (T <= -205){
        cerr << "Sonde en court-circuit!" << T << endl;
    }
    else if(T <=-196){
        cerr << "Niveau trop haut!" << endl;
    }
    else if(T <=-140){
        cerr << "Température OK!" << endl;
    }
    else if(T <=59){
        cerr << "Alarme Température Haute!" << endl;
    }
    else if(T>=60){
        cerr << "Sonde Coupé" << endl;
    }
    return T;
}
```

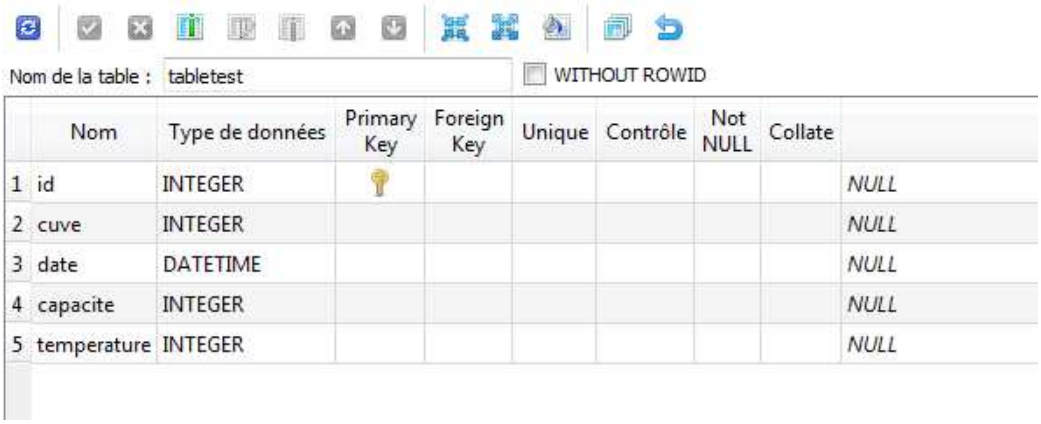
Et donc modifier la requête SQL pour écrire les données dans la BDD :

```
string query = "insert into tabletest (capacite, temperature,date) values
(?1,?2,datetime('now'));";
```

Pour porter une petite précision, datetime('now') n'est utilisable qu'avec sqlite3.

Après avoir modifié le code, j'ai dû rajouter les colonnes dans la table qui sont pour la date, la capacité et la température.

Ci-dessous on peut voir la modification avec le logiciel SQLiteStudio :



Nom de la table : tabletest WITHOUT ROWID

	Nom	Type de données	Primary Key	Foreign Key	Unique	Contrôle	Not NULL	Collate	
1	id	INTEGER							NULL
2	cuve	INTEGER							NULL
3	date	DATETIME							NULL
4	capacite	INTEGER							NULL
5	temperature	INTEGER							NULL

Ci-dessous on voit l'exécution du programme et l'on remarque bien que dès que l'on trouve le caractère '$ ', on exécute la lecture des caractères un par un jusqu'à arriver au caractère 'T' pour ensuite convertir la valeur numérique 108 en température ce qui donne -147°C et nous avertit que la température dans la cuve est correcte.

```
int CBdd::constReq(int T, char* N){

    bool ret = false;
    int err;

    string query = "insert into tabletest (capacite, temperature,date) values
(?1,?2,datetime('now'));";
    //cout << query << endl;
    err = sqlite3_prepare_v2(m_db, query.c_str(), -1, &stmt, NULL);
    if (ret != SQLITE_OK) {
        cerr << sqlite3_errmsg(m_db) << sqlite3_errstr(ret) << endl;
        //return 0;
    } else {
        cerr << "sqlite3_prepare() OK" << endl;
    }
    err = sqlite3_bind_text(stmt, 1, N, -1, SQLITE_STATIC);
    err = sqlite3_bind_int(stmt, 2, T);
    if (ret != SQLITE_OK) {
        cerr << sqlite3_errmsg(m_db) << sqlite3_errstr(ret) << endl;
        //return 0;
    } else {
        cerr << "sqlite3_bind_text() OK" << endl;
    }
}
```

```
Module en communication
Valeur de la trame = $N108T12#
caractere = $
do while ok
debut while
c[ 1] = N
c[ 2] = 1
c[ 3] = 0
c[ 4] = 8
c[ 5] = T
c[ 6] = 1
c[ 7] = 2
c[ 8] = #
trame: $N108T12#
$.N.1.0.8.T.Niv dans la trame : 108
BDD ouverte
Temp: -147
Température OK!
sqlite3_prepare() OK
sqlite3_bind_text() OK
Requete OK!
BDD fermée
Suppression Instruction Préparé
pi@rpi-2sp-rs:~/rfrec_sqlite2 $
```

On constate bien par la suite que cela est écrit dans la base de données avec les nouvelles informations :

```
69||2018-05-18 07:46:05|255|160
70||2018-05-18 07:46:16|195|34
71||2018-05-18 07:46:37|159|-40
72||2018-05-18 07:46:44|130|-101
73||2018-05-18 07:46:50|81|-203
74||2018-05-18 07:46:55|28|-314
75||2018-05-18 07:47:06|72|-222
76||2018-05-18 07:47:11|109|-144
77||2018-05-18 07:48:14|109|-144
78||2018-05-18 07:49:25|108|-147
sqlite>
```

On voit bien la date avec le temps grâce à datetime('now') dans la requête SQL et ensuite 108 la valeur numérique et -147 la température de la cuve. On peu aussi remarquer que la température est OK avec un -147°C.

5/ Itération 3

La dernière tâche que j'ai a réaliser est de faire communiquer les deux modules en utilisant le protocole de communication maître/esclave car jusqu'à présent je n'ai fait qu'utiliser le code fournit en exemple et améliorer ce dernier pour en arriver à un programme proche du fonctionnement attendu.

Ci-dessous tout le code réalisé pour pouvoir faire communiquer les modules en maître esclave :

Arduino

Simu_cuve.ino

```
#include <Time.h>     // Librairie tierce permettant de garder une trace de
l'heure avec ou sans RTC
#include <TimeLib.h>  //
#include <SPI.h>
#include <RF24.h>
#include "nRF24L01.h"
#include "printf.h"

/*bool radioNumber = 0;*/
#define TRIGGER 7
#define PIN_CS  10
#define PIN_CE  9
#define SPI_SPEED 1000000

#define REQUEST_FORMAT "$T????????#"
#define RESPONSE_FORMAT "$T????????N??#"

RF24 radio(PIN_CE, PIN_CS);

// Base des adresses de pipe choisie pour projet 2SP : 0xe931a846 (c'est en fait
le crc32 de la chaîne "2SPELECTRONIC")
// Le dernier octet de l'adresse sera codé comme suit : 'P' (comme pipe) codé sur
5 bits en code Baudot alphabet international + numéro de pipe (0 à 5)
//
//pipe 0 : 0xe931a846 0b01101 000 => 0xe931a84668
//pipe 1 : 0xe931a846 0b01101 001 => 0xe931a84669
//pipe 2 : 0xe931a846 0b01101 010 => 0xe931a8466a
//pipe 3 : 0xe931a846 0b01101 011 => 0xe931a8466b
//pipe 4 : 0xe931a846 0b01101 100 => 0xe931a8466c
//pipe 5 : 0xe931a846 0b01101 101 => 0xe931a8466d


uint64_t addresses[] = {0xe931a84668LL, 0xe931a84669LL};

const int analogPin = A0;


void setup() {

  Serial.begin(115200);
  printf_begin();

  Serial.println(F("Go go go!"));
```

```
  pinMode(TRIGGER, OUTPUT);

  radio.begin();

  // Activer l'accusé réception
  radio.enableAckPayload();

  // L'accusé réception est dynamique
  radio.enableDynamicPayloads();

  // Réglage de l'amplificateur
  radio.setPALevel(RF24_PA_LOW);

  //Réglage des canaux (pipe)
  radio.openWritingPipe(addresses[1]);
  radio.openReadingPipe(1, addresses[0]);

  radio.printDetails();

  // On fixe la taille des données utiles sur la  taille de la plus grande des
trames
  // entre celle de quête et celle de réponse
  // Attention : celle-ci doit rester inférieure à 32 (imposée par le matériel)
  radio.setPayloadSize( max( strlen(REQUEST_FORMAT)+1, strlen(RESPONSE_FORMAT)+1 )
);

  // En écoute
  radio.startListening();
}

void loop() {
  int sensorValue;
  char request[] = REQUEST_FORMAT;
  char response[] = RESPONSE_FORMAT;

  // Attendre trame de requête
  if (radio.available()) {
    // Arrêter l'écoute
    radio.stopListening();

    // Lire la requête reçue
    radio.read(&request, strlen(REQUEST_FORMAT)+1);
    Serial.print("Requete : ");
    Serial.println(request);

    // Extraire l'heure et la programmer sur l'Arduino
    // Note : la fonction setTime() ne semble programmer que les 4 octets de
    // poids faible !?
    time_t ts;
    ts = (time_t)strtol(&request[2], NULL, 16);
    Serial.print("ts : ");
    Serial.println(ts);
    setTime(ts);

    // Lire la valeur du potentiomètre (niveau de cuve) et construire la partie de
la trame de réponse associée
    char niveauStr[] = "Nxx";
    sensorValue = analogRead(A0);
    sprintf(niveauStr, "N%02x", map(sensorValue, 0, 1023, 0, 255));
```

```
    // Lire l'heure système et construire la partie de la trame de réponse
associée
    char timestampStr[] = "Txxxxxxxx";
    sprintf(timestampStr, "T%08x", now());

    // Construire la trame entière de réponse
    // Note : sprintf(frame, "$T%08xN%02x#", now(), map(sensorValue, 0, 1023, 0,
255)) ne fonctionne pas !?
    // => ceci explique pourquoi on utilise 2 sprintf() séparés pour le timestamp
et le niveau de cuve
    sprintf(response, "$%s%s#", timestampStr, niveauStr);
    Serial.print("Response : ");
    Serial.println(response);

    // Envoyer la réponse
    if (!radio.write(&response, strlen(response)+1)) {
      Serial.println("Erreur : échec envoi réponse");
    }

    // Reprendre l'écoute
    radio.startListening();
  }

  delay(100);

}
```

Raspberry

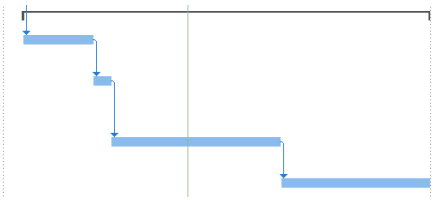
Rfsend_centrale.cpp

```
#include <unistd.h>
#include <ctime>
#include <RF24/RF24.h>

#define REQUEST_FORMAT "$T????????#"
#define RESPONSE_FORMAT "$T????????N??#"

using namespace std;

RF24 radio(22,0);

// Base des adresses de pipe choisie pour projet 2SP : 0xe931a846 (c'est en fait
le crc32 de la chaîne "2SPELECTRONIC")
// Le dernier octet de l'adresse sera codé comme suit : 'P' (comme pipe) codé sur
5 bits en code Baudot alphabet international + numéro de pipe (0 à 5)
//
//pipe 0 : 0xe931a846 0b01101 000 => 0xe931a84668
//pipe 1 : 0xe931a846 0b01101 001 => 0xe931a84669
//pipe 2 : 0xe931a846 0b01101 010 => 0xe931a8466a
//pipe 3 : 0xe931a846 0b01101 011 => 0xe931a8466b
//pipe 4 : 0xe931a846 0b01101 100 => 0xe931a8466c
//pipe 5 : 0xe931a846 0b01101 101 => 0xe931a8466d
const uint64_t pipeW = 0xe931a84668LL;
const uint64_t pipeR = 0xe931a84669LL;

int main(int argc, char** argv){
      unsigned long debut;
      bool timeout;

      cout << "Go go go!" << endl;

      radio.begin();

      // Activer l'accusé réception
      radio.enableAckPayload();

      // L'accusé réception est dynamique
      radio.enableDynamicPayloads();

      // réglage de l'amplificateur
      radio.setPALevel(RF24_PA_LOW);

      // réglage des canaux (pipe)
      radio.openWritingPipe(pipeW);
      radio.openReadingPipe(1,pipeR);

      // affichage d'un résumé de la configuration
      radio.printDetails();

      // On fixe la taille des données utiles sur la  taille de la plus
      grande des trames
      // entre celle de quête et celle de réponse
      // Attention : celle-ci doit rester inférieure à 32 (imposée par le
matériel)
  radio.setPayloadSize( max( strlen(REQUEST_FORMAT)+1, strlen(RESPONSE_FORMAT)+1 )
);
```

```
	while(1) {
		// Arrêt écoute
		radio.stopListening();

		time_t t = time(nullptr);

		stringstream request;
		request << "$T" << setfill('0') << setw(8) << hex << t << '#';
		auto s = request.str();
		const char * frame = s.c_str();
		cout << "Trame : [" << frame << "]" << endl;

		printf("Envoi\n");

		// Envoi avec vérification d'erreur
		if( !radio.write(frame, strlen(frame)+1) ) {
			printf("Erreur : write.\n");
		}

		// Mise en écoute
		radio.startListening();

		// Enregistrement du début du délai
		debut = millis();
		timeout = false;

		// Tant que rien à lire
		while (!radio.available()) {
			if (millis()-debut > 1000) {
				timeout = true;
				break;
			}
		}

		// abandon pour cause de délai écoulé ?
		if(timeout){
			printf("Erreur : timeout.\n");
		} else {
			char response[] = RESPONSE_FORMAT;

			// lecture des données reçues
			radio.read(&response, strlen(RESPONSE_FORMAT)+1);

			// Affichage
			cout << "Reponse : [" << response << "]" << endl;
		}

		// dodo 5 secondes
		sleep(5);
	}
	return 0;
}
```

5/ Planning

		Tâche Rémy	190 heures	Mer 17/01/18	Jeu 31/05/18	
9		Concevoir/Coder/Tester la librairie nRF24	50 heures	Mer 17/01/18	Ven 09/02/18	7
10		Modifier l'application d'acquisition de mesures	10 heures	Ven 09/02/18	Jeu 15/02/18	9
11		Coder le protocole Maitre<X<Esclave	70 heures	Jeu 15/02/18	Jeu 12/04/18	10
12		Coder l'IHM	60 heures	Jeu 12/04/18	Jeu 31/05/18	11



6/ Journal de bord :

Jeudi 11 janvier

Pris connaissance des documents ≈ 2h

Recherche de nRF24L01+

Vendredi 12 janvier

Installation magicdraw, µproject

Planification de projet

Mercredi 17 janvier

Installation de netbeans

Recherches sur le projet

Jeudi 18 janvier

Conception de la carte raspberry

Configurer et activer SPI

Installation de la bibliothèque nfr24l01 Radiohead

Vendredi 19 janvier

Installation de samba

Mercredi 24 janvier

Comparer les 2 libraires Radiohead et TMRH20 Avantage/inconvénients

Jeudi 25 janvier

Chercher comment rajouter + de 5cuves avec les 2 librairies.

Vendredi 26 janvier

Définir un protocole pour la synchro boitier maitre/esclave

Mercredi 31 janvier

Regarder comment écrire dans les bases de données

MULTITANKS MONITOR interroge le module et le sauvegarde dans la BDD.

Jeudi 1 février

Tester les 2 capteurs radio avec la libraire choisi serveur (arduino), client (raspi)

Vendredi 2 Mercredi 7 jeudi 8 février

Interpréter/Coder pour pouvoir faire communiquer les 2 capteurs + lire valeur potentiomètre

Vendredi 9 février

Diaporama + Documentation

Mercredi 14 février

Envoyer par trame les donnes date, heure, température, niveau d'azote

Jeudi 15 Vendredi 16 Mercredi 21 Vendredi 23 février

Installer la base de données

Coder pour écrire dans la base de données la trame reçut

Mois de mars et Avril

Tout le mois de mars, il faut retranscrire le code C en C++, intégrer les nouvelles valeurs pour les convertir de numérique en température et réaliser la partie science physique.

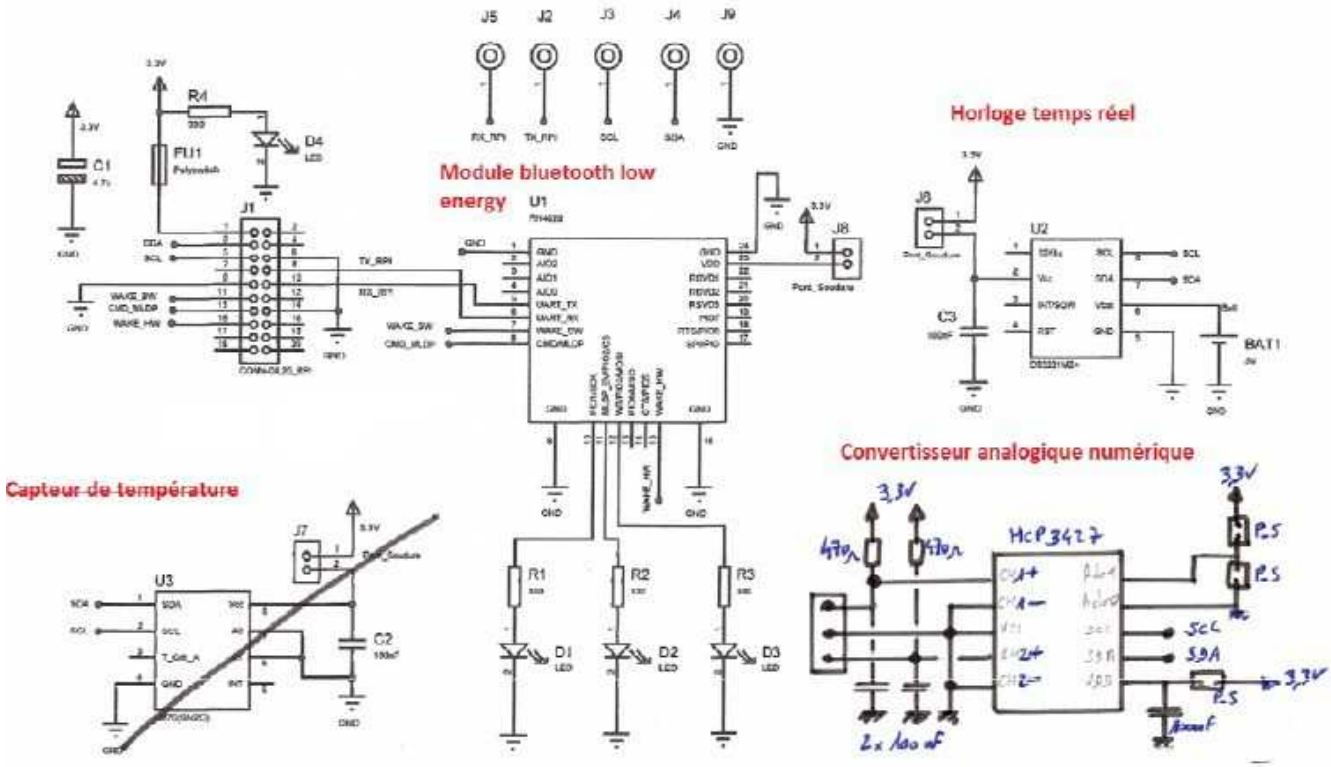
VII/ Partie personnelle étudiant 3 (CHOLEZ-GLAB Corentin)

1/ Présentation du travail à réaliser

- Analyser la version précédente.
- Valider le choix du convertisseur analogique/numérique.
- Concevoir/Réaliser/Tester une carte d'extension pour Raspberry conforme à la spécification HAT
- Concevoir/Coder/Tester une librairie d'acquisition des grandeurs analogiques
- Configurer l'horloge temps réel

Intégrer une procédure de synchronisation des RTCs au cours des acquisitions

Schéma proposé par le client en 2017 :

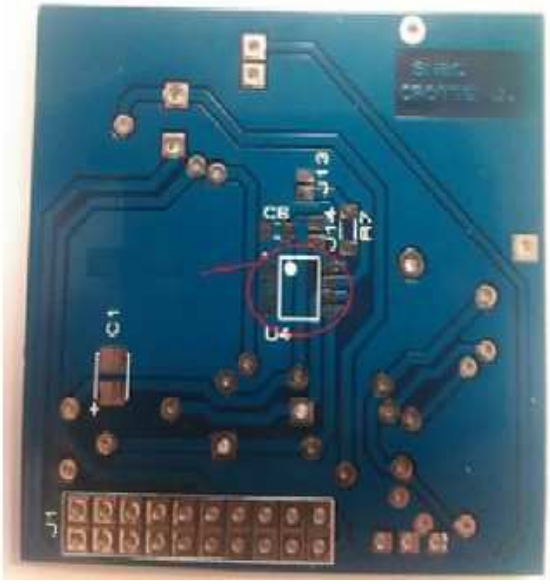


2/ Développement de la carte

Carte réalisée en 2017

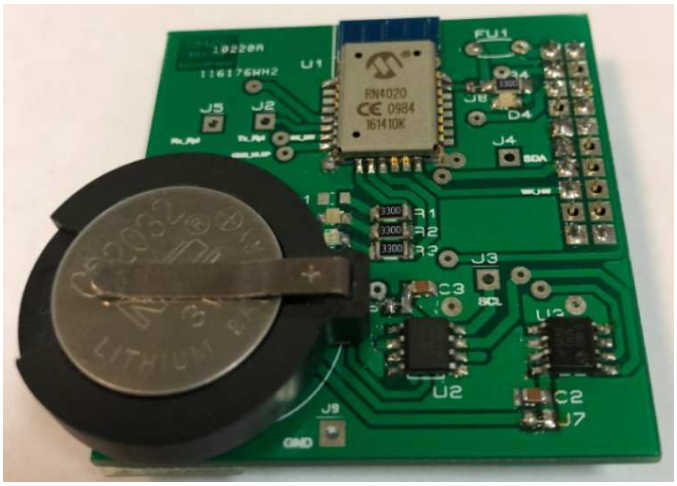
- Module Bluetooth
- Capteur de température
- RTC
- Cavalier pour le mode optimisation
- Mémoire HAT

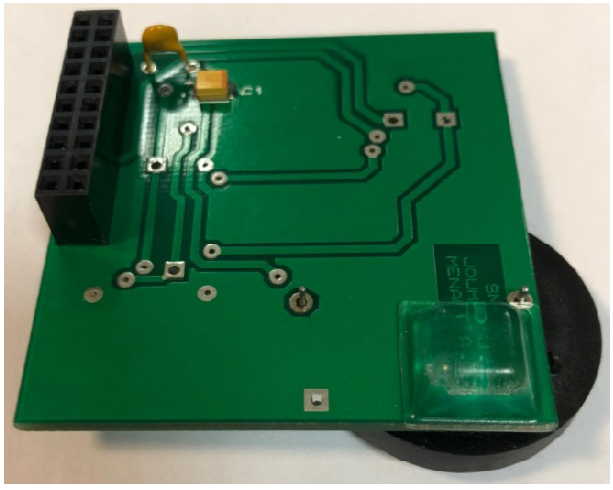

Convertisseur + Bornier



Carte réalisée en 2016

- Module Bluetooth
- Capteur de température
- RTC





3/ Diagramme de Gantt prévisionnel

		Projet	218 heures	Jeu 11/01/18	Ven 08/06/18
2		Tâches communes	8 heures	Jeu 11/01/18	Ven 12/01/18
3		Lecture du dossier	2 heures	Jeu 11/01/18	Jeu 11/01/18
4		Analyse de l'existant	1 heure	Jeu 11/01/18	Jeu 11/01/18
5		Réunion de synthèse	1 heure	Jeu 11/01/18	Jeu 11/01/18
6		Réaliser un plan des itérations	2 heures	Jeu 11/01/18	Jeu 11/01/18
7		Réaliser un planning prévisionnel	4 heures	Jeu 11/01/18	Ven 12/01/18
8		Tâches Corentin	218 heures	Jeu 11/01/18	Ven 08/06/18
9		Recherche sur le DS3231	4 heures	Ven 12/01/18	Mer 17/01/18
10		Essai horloge temps réel + modification du code	20 heures	Mer 17/01/18	Jeu 25/01/18
11		Essai et test du nRF24L01	20 heures	Mer 31/01/18	Jeu 08/02/18
12		Rédaction dossier revue 1	11 heures	Ven 09/02/18	Jeu 15/02/18
13		Routage de la carte	12 heures	Ven 16/02/18	Mer 14/03/18
14		Rédaction Dossier revue 2 et finale	17 heures	Jeu 15/03/18	Jeu 22/03/18
15		Recherche sur le MCP3427	8 heures	Ven 23/03/18	Jeu 29/03/18
16		Mise en oeuvre du MCP3427	25 heures	Ven 30/03/18	Jeu 12/04/18
17		Partie physique sur le MCP3427	17 heures	Ven 13/04/18	Jeu 10/05/18
18		Rédaction finale et rendu de dossier	14 heures	Ven 11/05/18	Ven 18/05/18
19		Soudage de la carte et essai	38 heures	Mer 23/05/18	Ven 08/06/18
20		Revue 1	0 heure	Mer 14/03/18	Mer 14/03/18
21		Revue 2	0 heure	Jeu 10/05/18	Jeu 10/05/18
22		Revue finale	0 heure	Jeu 14/06/18	Jeu 14/06/18

Diagramme de Gantt final :

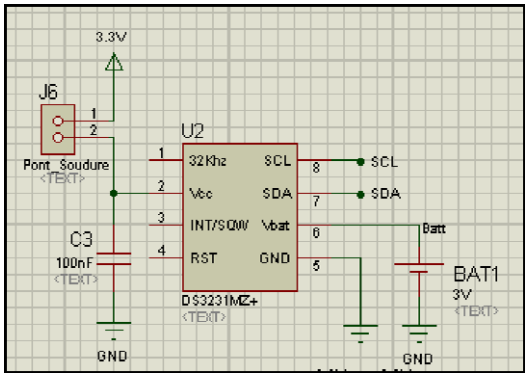
		Projet	200 heures	Jeu 11/01/18	Jeu 31/05/18
2		Tâches communes	10 heures	Jeu 11/01/18	Mer 17/01/18
3		Lecture du dossier	2 heures	Jeu 11/01/18	Jeu 11/01/18
4		Analyse de l'existant	1 heure	Jeu 11/01/18	Jeu 11/01/18
5		Réunion de synthèse	1 heure	Jeu 11/01/18	Jeu 11/01/18
6		Réaliser un plan des itérations	2 heures	Jeu 11/01/18	Jeu 11/01/18
7		Réaliser un planning prévisionnel	4 heures	Ven 12/01/18	Mer 17/01/18
8		Tâches Corentin	190 heures	Mer 17/01/18	Jeu 31/05/18
9		Recherche sur le DS3231	5 heures	Mer 17/01/18	Jeu 18/01/18
10		Essai horloge temps réel + modification du code	50 heures	Jeu 18/01/18	Mer 14/02/18
11		Recherche sur le MCP3427	80 heures	Jeu 15/02/18	Ven 13/04/18
12		Mise en oeuvre du MCP3427	55 heures	Ven 13/04/18	Jeu 31/05/18
13		Revue 1	0 heure	Mer 14/03/18	Mer 14/03/18
14		Revue 2	0 heure	Jeu 10/05/18	Jeu 10/05/18
15		Revue finale	0 heure	Jeu 14/06/18	Jeu 14/06/18

4/ Les composants

L'horloge temps réel DS3231MZ+

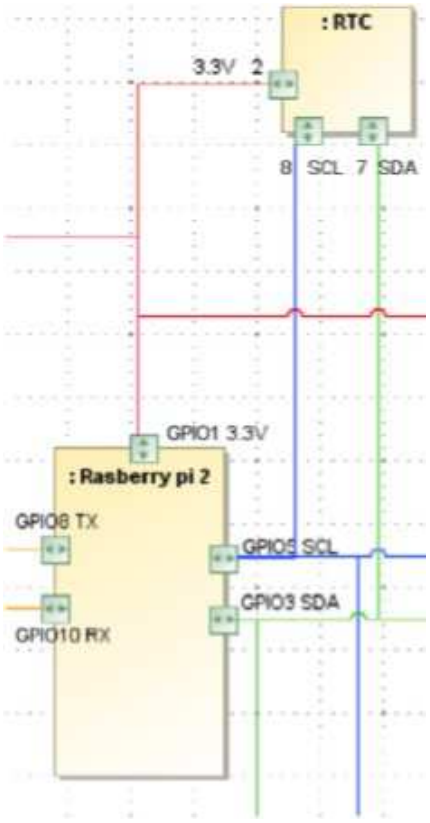
RTC (Real Time Clock)

DS3231MZ+



L'horloge temps réel DS3231MZ+ est la même utilisée que l'année précédente.

Lorsque la Raspberry n'est pas connectée à un réseau, l'heure de la Raspberry se remet à zéro ou à la dernière utilisation de cette dernière. Cependant pour l'utilisation que nous voulons faire de la Raspberry dans ce projet, nous avons besoin que l'heure soit à jour pour enregistrer l'heure et la date à laquelle les mesures de températures sont effectuées pour suivre l'évolution de ces dernières. Pour cela j'ai donc suivi le tutoriel qui a été rédigé par l'élève de l'année précédente (Voir ANNEXE). Je dispose de la carte de l'année 2016 pour effectuer le test sur ma Raspberry.
J'ai rencontré un petit problème elle n'était pas reconnu par ma carte. Il m'a fallut changer la pile de la carte pour ensuite qu'elle soit reconnu à l'adresse i2c 0x68.



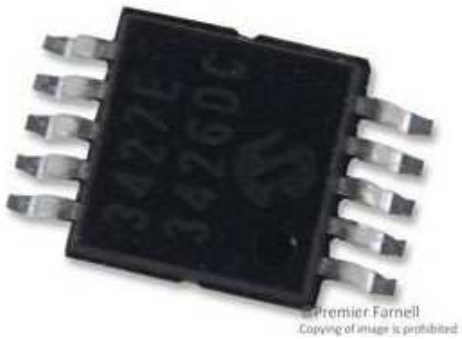
Extrait de la documentation de l'Horloge temps réel

Table 2. Timekeeping Registers

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hours	10 Hours	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hours	10 Hours	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hours	10 Hours	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	NA	NA	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32KHZ	BSY	A2F	A1F	Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	81h–7Fh
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Temperature MSB	—
12h	DATA	DATA	0	0	0	0	0	0	Temperature LSB	—

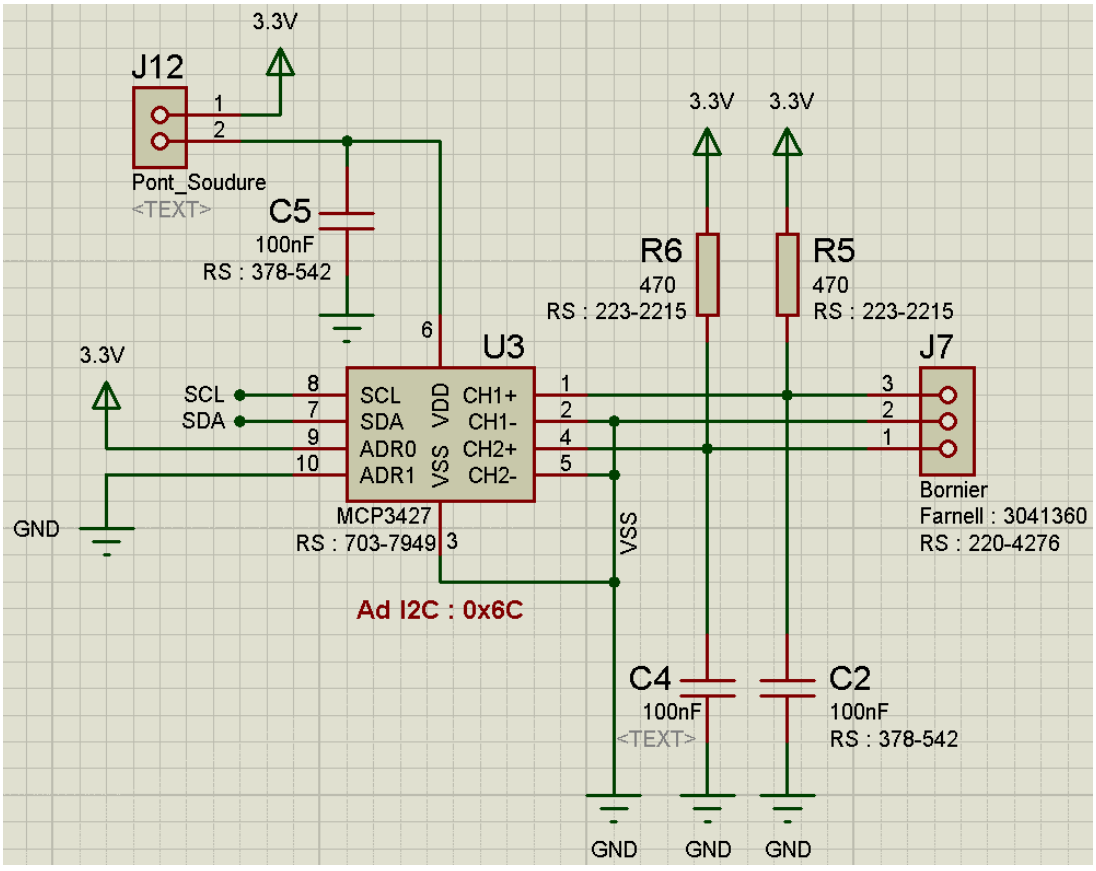
Note: *Unless otherwise specified, the registers' state is not defined when power is first applied.*

Le convertisseur analogique numérique MCP3427



Le convertisseur analogique numérique MCP3427 va permettre de récupérer le niveau de la cuve et de le convertir afin de pouvoir le transmettre via le module RF24L01 au coffret maître. Ce composant a pour particularité de ne pas êtreun convertisseur à approximations successives mais un convertisseur sigma-delta qui permet une meilleure résolution.

Schéma structurel du composant sur Proteus



5/ Mise en œuvre du MCP3427

```
double getMesure(int CH)
{

    int temp=0;
    double mesure = 0;
    char i2cOut[2];
    char i2cIn[3];

    if (CH==1)          i2cOut[0]=0x88;         // Mesure sur CH1
    else if (CH==2)     i2cOut[0]=0xA8;         // Mesure sur CH2
        bcm2835_i2c_write(i2cOut, 1);           // Configuration en 16bits et mesure sur CH1+ ou CH2+ en one shoot (économie d'énergie)
        bcm2835_delay(100);                     // Délais indispensable pour laisser le temps à la conversion
        bcm2835_i2c_read(i2cIn, 3);             // Lecture des 3 octets
        mesure = (256*i2cIn[0] + i2cIn[1])*2.048/32767;    // Calcul
        return mesure;
}


int main(void)
{
    unsigned int slave_address=0x6C;

    if (!bcm2835_init() || !bcm2835_i2c_begin())
    {
        printf("init failed. Are you running as root??\n");
        _exit(-1);
```

L'octet de configuration est en mode one shot sur le channel 1 en mode 16 bits sans gain soit « 10001000 » donc « 0x88 »
Pour le channel 2 nous avons la même chose sauf que c'est « 10101000 » soit « 0xA8 »

Le quantum pour une alimentation de 2,048V de la RaspBerry Pi pour une résolution de 16 bits.

Les PINs Adr0 et Adr1 sont à l'état haut ce qui crée l'adresse i2c : 0x6C

I^2C Device Address Bits			Logic Status of Address Selection Pins	
A2	A1	A0	Adr0 Pin	Adr1 Pin
0	0	0	0 (Addr_Low)	0 (Addr_Low)
0	0	1	0 (Addr_Low)	Float
0	1	0	0 (Addr_Low)	1 (Addr_High)
1	0	0	1 (Addr_High)	0 (Addr_Low)
1	0	1	1 (Addr_High)	Float
1	1	0	1 (Addr_High)	1 (Addr_High)
0	1	1	Float	0 (Addr_Low)
1	1	1	Float	1 (Addr_High)
0	0	0	Float	Float

Voici l'extrait de la documentation permettant le calcul du quantum.

EQUATION 4-3:

$$LSB = \frac{2 \times V_{REF}}{2^N} = \frac{2 \times 2.048\,V}{2^N}$$

Where:

N = Resolution, which is programmed in the Configuration Register: 12, 14, or 16.

bit 7 **$\overline{\text{RDY}}$:** Ready Bit

This bit is the data ready flag. In read mode, this bit indicates if the output register has been updated with a latest conversion result. In One-Shot Conversion mode, writing this bit to "1" initiates a new conversion.

Reading $\overline{\text{RDY}}$ bit with the read command:

1 = Output register has not been updated.
0 = Output register has been updated with the latest conversion result.

Writing $\overline{\text{RDY}}$ bit with the write command:

Continuous Conversion mode: No effect

One-Shot Conversion mode:
1 = Initiate a new conversion.
0 = No effect.

bit 6-5 **C1-C0:** Channel Selection Bits

00 = Select Channel 1 **(Default)**
01 = Select Channel 2
10 = Select Channel 3 **(MCP3428 only, treated as "00" by the MCP3426/MCP3427)**
11 = Select Channel 4 **(MCP3428 only, treated as "01" by the MCP3426/MCP3427)**

bit 4 **$\overline{\text{O}}$/C:** Conversion Mode Bit

1 = Continuous Conversion Mode **(Default)**. The device performs data conversions continuously.
0 = One-Shot Conversion Mode. The device performs a single conversion and enters a low power standby mode until it receives another write or read command.

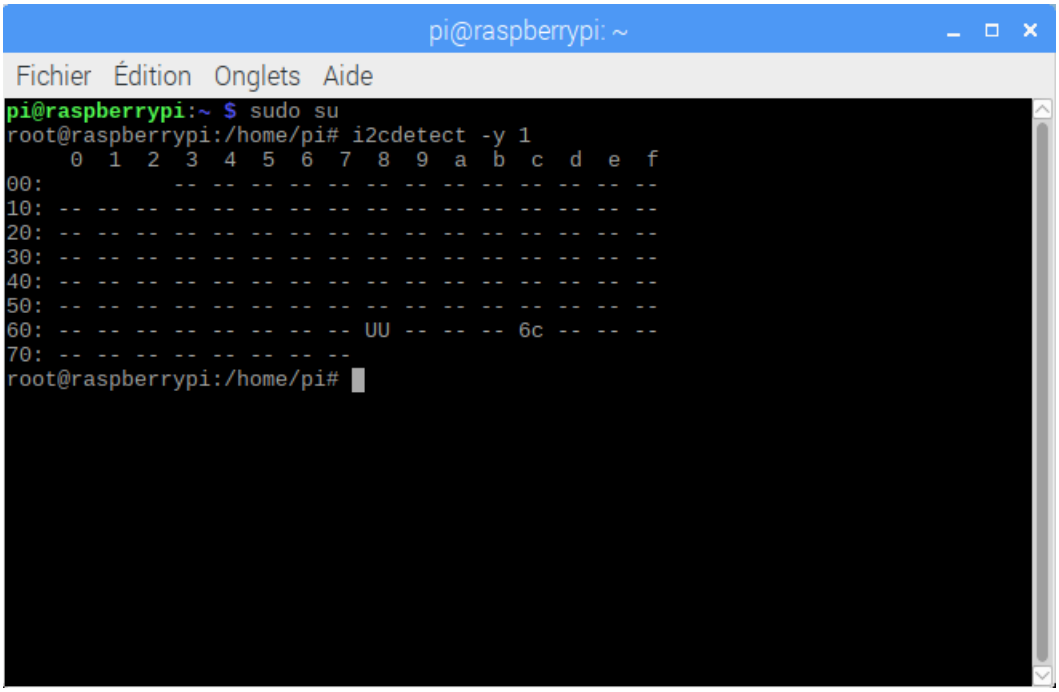
bit 3-2 **S1-S0:** Sample Rate Selection Bit

00 = 240 SPS (12 bits) **(Default)**
01 = 60 SPS (14 bits)
10 = 15 SPS (16 bits)

bit 1-0 **G1-G0:** PGA Gain Selection Bits

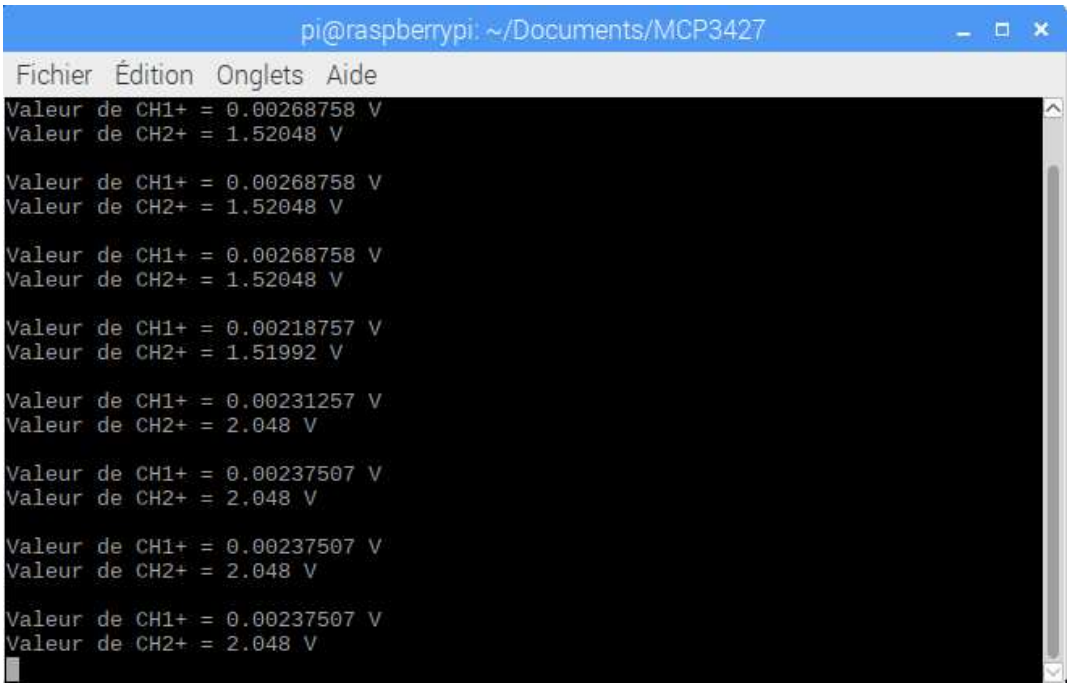
00 = x1 **(Default)**
01 = x2
10 = x4
11 = x8

En faisant la commande : sudo i2cdetect –y 1
Nous pouvons voir que deux composants I2C sont détectés sur la RaspBerry, l'adresse 68 visible ici en « UU » car le composant est pris en charge par la RaspBerry, ce composant correspond à la RTC. Nous pouvons voir que le convertisseur analogique numérique est à l'adresse 6c.

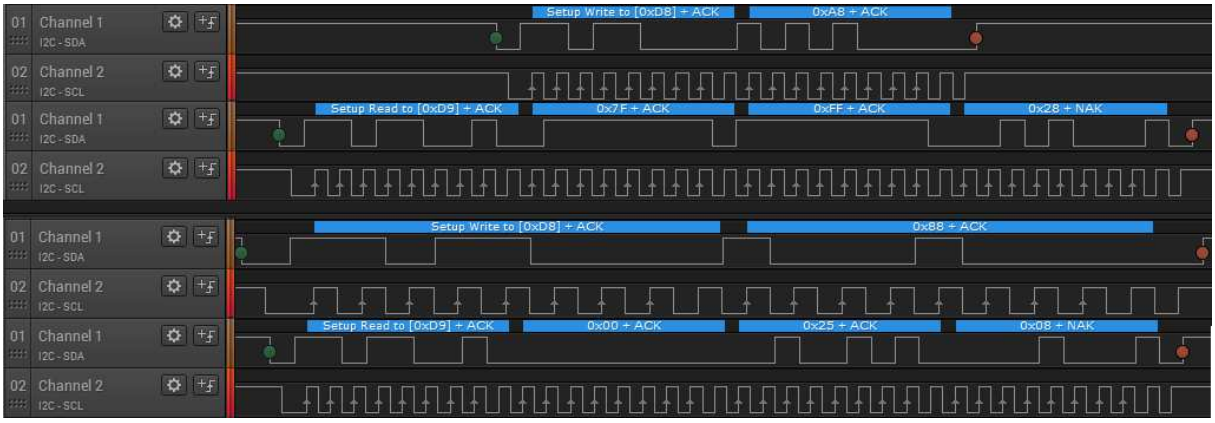


En faisant varier le potentiomètre nous optenons des valeurs comprises entre 0V et 2,048V

Lors des acquisitions de mesures j'ai fait varier le potentiomètre relié sur le CH2+ nous pouvons constater une augmentation de la valeur.



Analyse des trames 0xA8, 0x88



L'écriture se fait sur l'adresse 0xD8 et la lecture se fait sur l'adresse 0xD9. Sur ces trames les premières écritures se font sur le CH2 à l'adresse 0xA8. Ensuite les deuxièmes écritures se font sur le CH1 à l'adresse 0x88. Le Channel 2 sur cette trame est à 2.048V et le Channel 1 est pratiquement au minimum, proche de 0V.

Montage pour obtenir les trames I2C

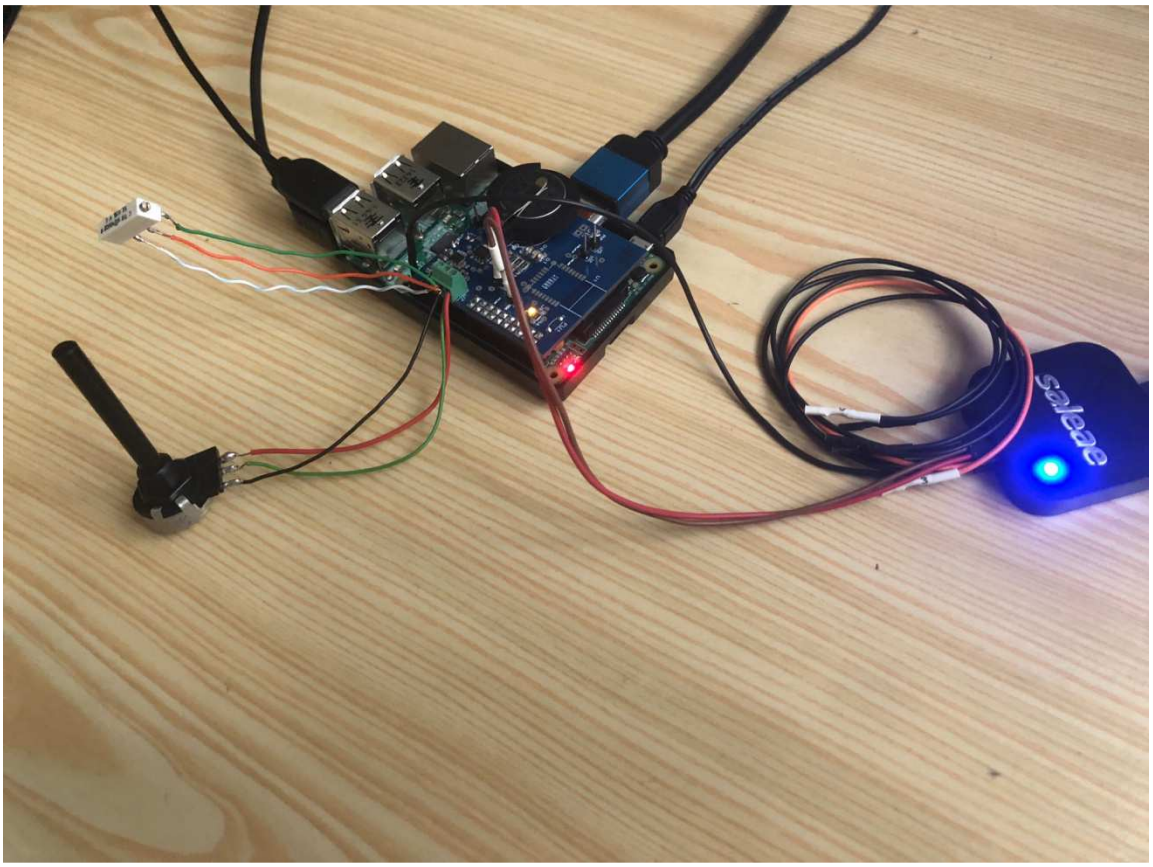
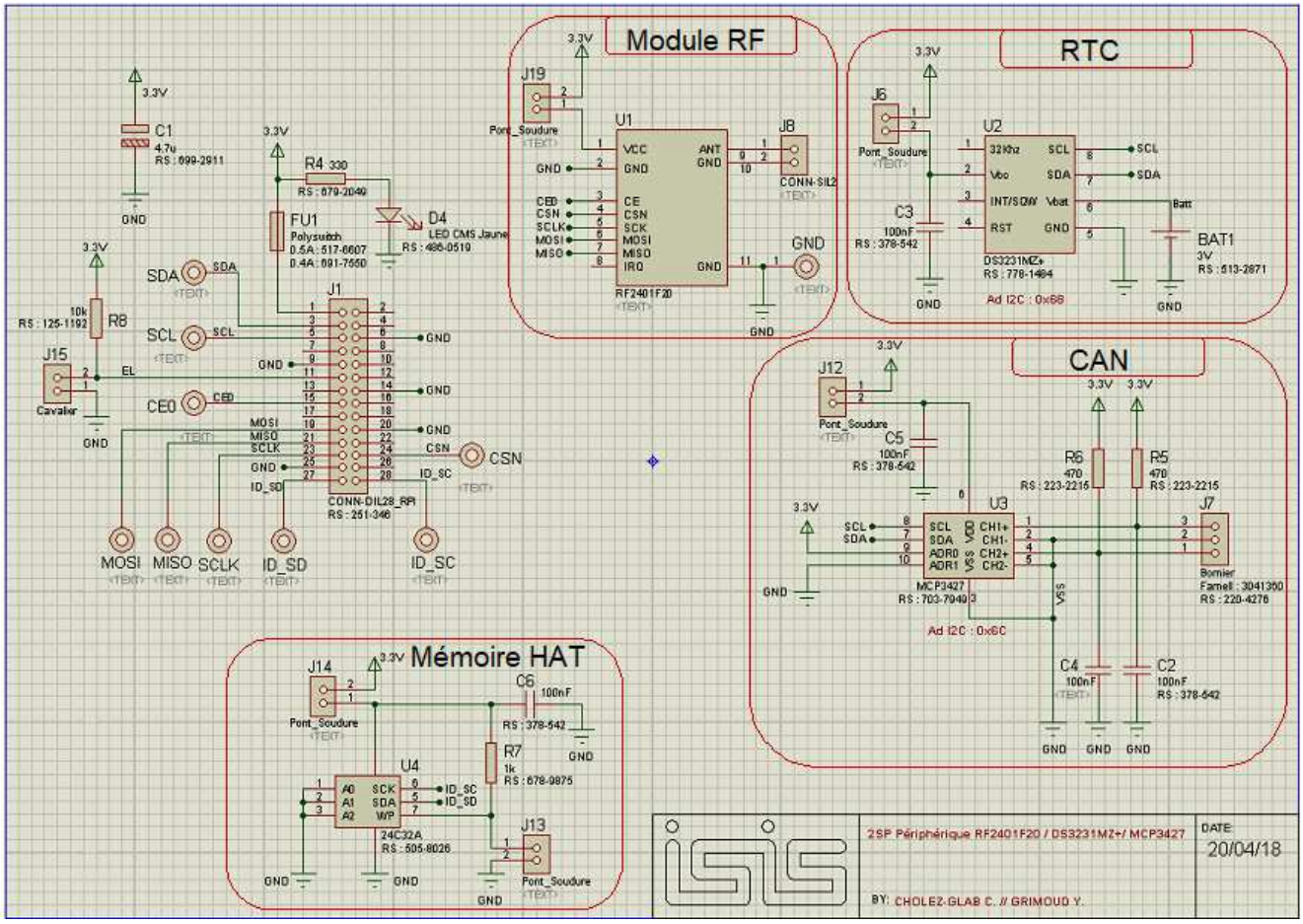
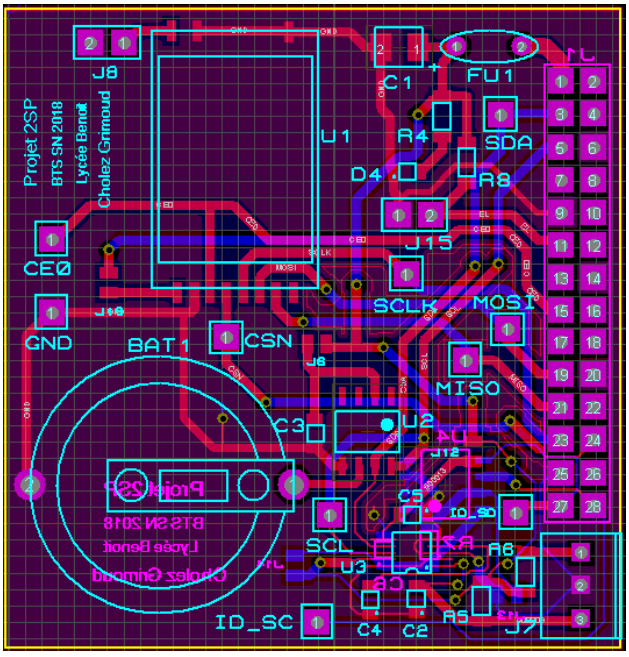


Schéma structurel de la carte sur Proteus



Routage de la carte sur ARES



Le routage de la carte a été fait en commun à cause d'un retard pris sur le projet. Nous avons donc décidé de faire le routage en commun pour gagner du temps sur le reste du projet.

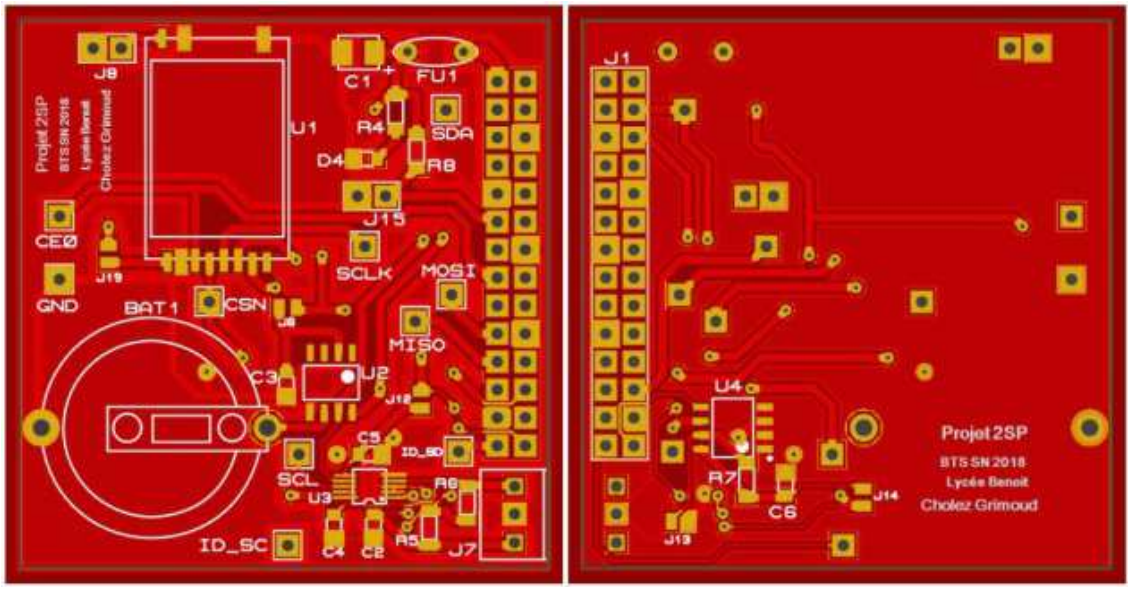
Commande de la carte sur SeeedStudio

PCB Cost	USD$4.90
Base Material	FR-4 TG130
No. of Layers	2 layers
PCB Dimensions	100mm * 100mm
PCB Quantity	10
No. of Different Designs	1
PCB Thickness	1.6mm
PCB Color	Red
Surface Finish	HASL
Minimum Solder Mask Dam	0.4mm↑
Copper Weight	1oz.
Minimum Drill Hole Size	0.3mm
Trace Width / Spacing	6/6 mil
Blind or Buried Vias	No
Plated Half-holes / Castellated Holes	No
Impedance Control	No
Sub-Total	USD$4.90
Production Time	5 – 6 Working Days
Weight	0.32kg
Shipping	Calculated at Checkout

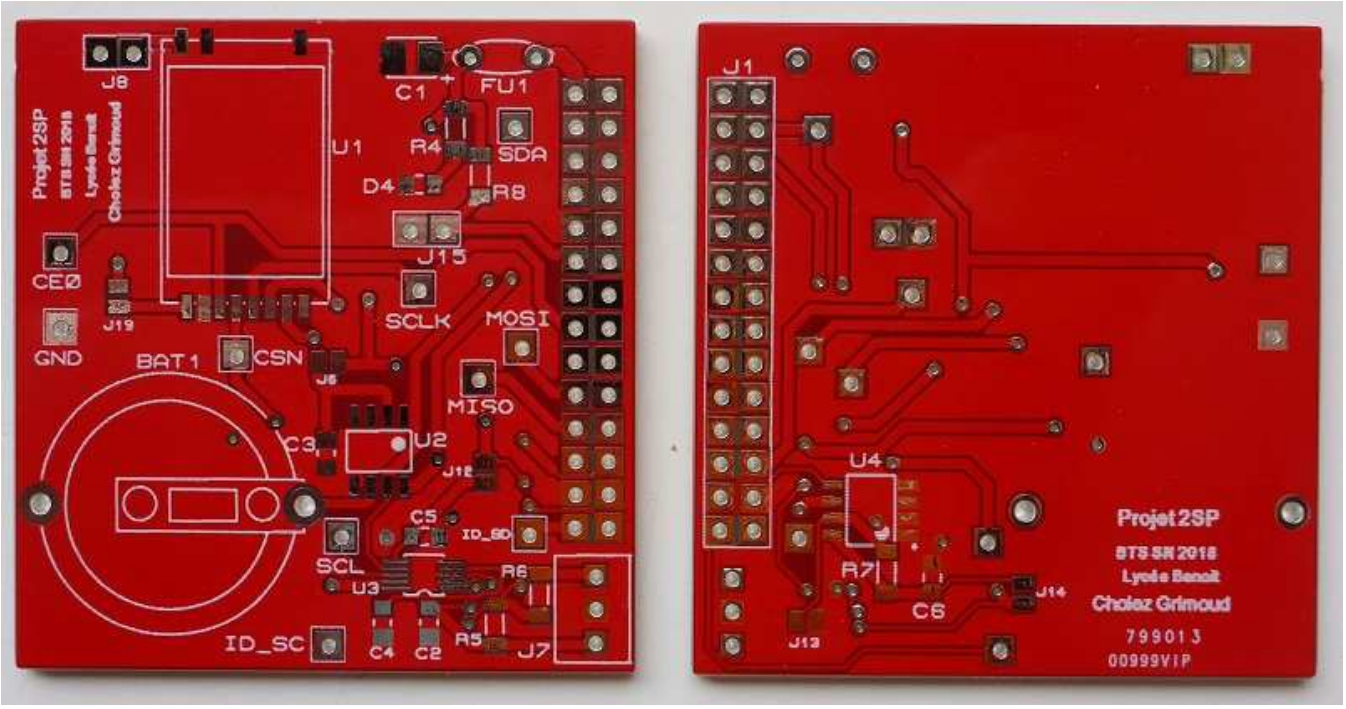
Add to Cart

La carte a été sous-traitée sur le site SeeedStudio qui a fabriqué nos cartes en Asie. Pourquoi l'Asie ? Car le prix de la même carte en France aurait surement coûté 10 fois plus cher. Donc pour une raison économique nous avons préféré la faire sous-traiter en Asie. Pour un coût de 4,90$ nous avons 10 cartes.

Rendu des cartes sur le site SeeedStudio avec Gerber Viewer :



Réception de la carte 2018



Le côté top de la carte **Le côté bottom de la carte**

Nous avons reçu les cartes le 11 Mai 2018.

Pochoir pour les soudures CMS de la carte

Ce pochoir va permettre de souder plus facilement les composants CMS de la carte, les trous sont exactement à l'endroit des composants CMS nous avons juste à étaler de la pâte à braser.

6/ Conclusion

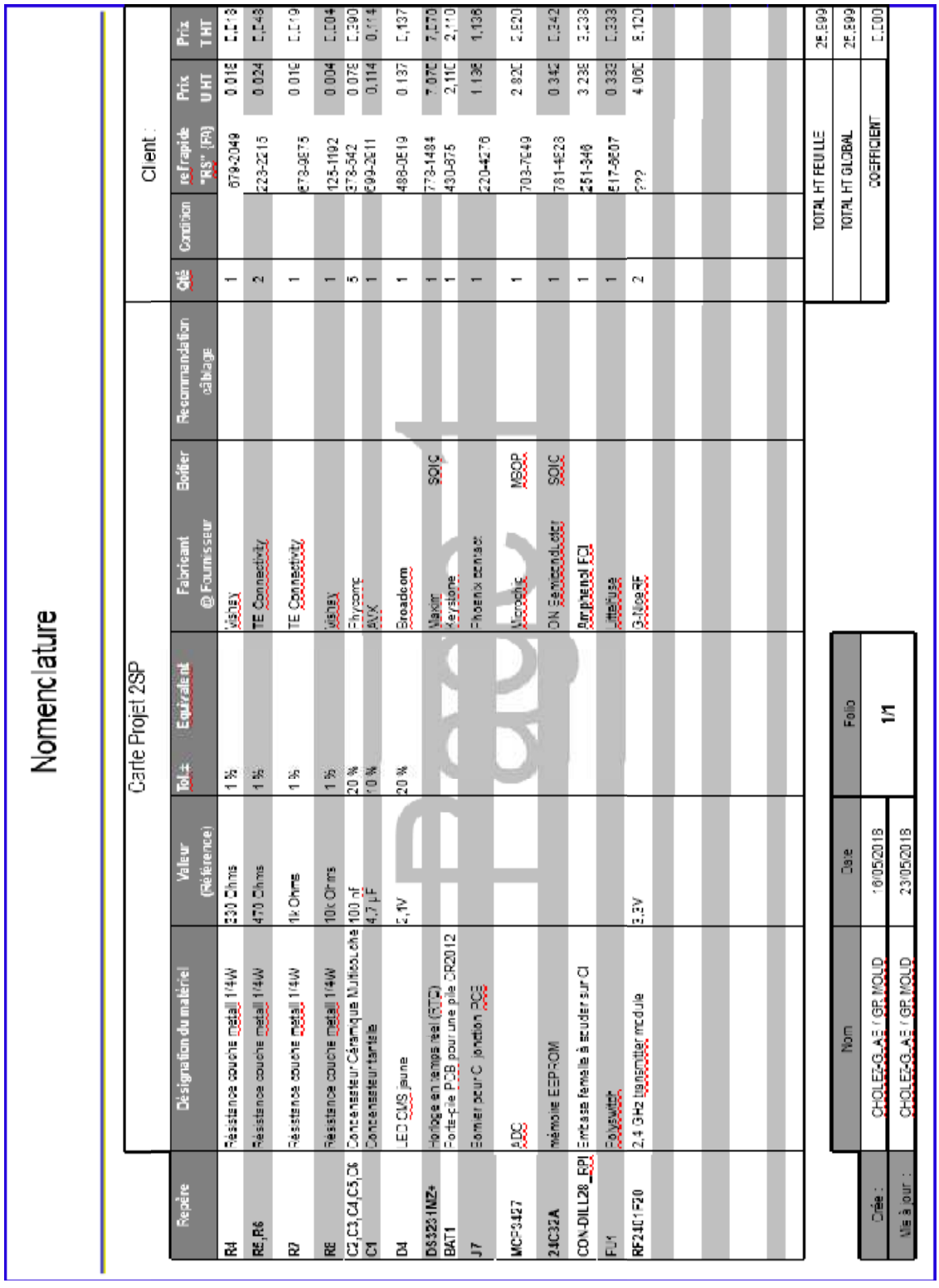
1) Diagramme de Gantt final

		Projet	218 heures	Jeu 11/01/18	Ven 08/06/18
2		**Tâches communes**	**8 heures**	**Jeu 11/01/18**	**Ven 12/01/18**
3		Lecture du dossier	2 heures	Jeu 11/01/18	Jeu 11/01/18
4		Analyse de l'existant	1 heure	Jeu 11/01/18	Jeu 11/01/18
5		Réunion de synthèse	1 heure	Jeu 11/01/18	Jeu 11/01/18
6		Réaliser un plan des itérations	2 heures	Jeu 11/01/18	Jeu 11/01/18
7		Réaliser un planning prévisionnel	4 heures	Jeu 11/01/18	Ven 12/01/18
8		**Tâches Corentin**	**218 heures**	**Jeu 11/01/18**	**Ven 08/06/18**
9		Recherche sur le DS3231	4 heures	Ven 12/01/18	Mer 17/01/18
10		Essai horloge temps réel + modification du code	20 heures	Mer 17/01/18	Jeu 25/01/18
11		Essai et test du nRF24L01	20 heures	Mer 31/01/18	Jeu 08/02/18
12		Rédaction dossier revue 1	11 heures	Ven 09/02/18	Jeu 15/02/18
13		Routage de la carte	12 heures	Ven 16/02/18	Mer 14/03/18
14		Rédaction Dossier revue 2 et finale	17 heures	Jeu 15/03/18	Jeu 22/03/18
15		Recherche sur le MCP3427	8 heures	Ven 23/03/18	Jeu 29/03/18
16		Mise en oeuvre du MCP3427	25 heures	Ven 30/03/18	Jeu 12/04/18
17		Partie physique sur le MCP3427	17 heures	Ven 13/04/18	Jeu 10/05/18
18		Rédaction finale et rendu de dossier	14 heures	Ven 11/05/18	Ven 18/05/18
19		Soudage de la carte et essai	38 heures	Mer 23/05/18	Ven 08/06/18
20		Revue 1	0 heure	Mer 14/03/18	Mer 14/03/18
21		Revue 2	0 heure	Jeu 10/05/18	Jeu 10/05/18
22		Revue finale	0 heure	Jeu 14/06/18	Jeu 14/06/18

2) Conclusion et connaissances acquises

Ce projet m'a apporté beaucoup de connaissance dans plusieurs domaines. Notamment, sur l'importance de l'organisation dans un groupe de projet.
J'ai compris comment bien analyser une documentation d'un composant ainsi que le fonctionnement de mes composants notamment de l'horloge temps réel.
En effet, j'ai élargi mes connaissances en routage grâce au routage de la carte avec ARES, de nombreux problèmes m'ont aidé à me perfectionner dans le routage.

Annexes :



Nomenclature

Carte Projet 2SP

Client :

Repère	Désignation du matériel	Valeur (Référence)	Tol.±	Equivalent	Fabricant @ Fournisseur	Boîtier	Recommandation câblage	Qté	Condition	ref rapide "RS" (FR)	Prix U HT	Prix THT
R4	Résistance couche métal 1/4W	330 Ohms	1 %		Vishay			1		679-2049	0.015	0.015
R5,R6	Résistance couche métal 1/4W	470 Ohms	1 %		TE Connectivity			2		623-2615	0.024	0.048
R7	Résistance couche métal 1/4W	1k Ohms	1 %		TE Connectivity			1		679-9675	0.015	0.015
R8	Résistance couche métal 1/4W	10k Ohms	1 %		Vishay			1		125-1192	0.004	0.004
C2,C3,C4,C5,C6	Condensateur Céramique Multicouche	100 nF	20 %		Phycomp			5		278-542	0.078	0.390
C1	Condensateur tantale	4,7 µF	10 %		AVX			1		509-2611	0,114	0,114
D4	LED CMS jaune	2,1V	20 %		Broadcom			1		488-0619	0.137	0,137
DS3231MZ+	Horloge en temps réel (RTC)				Maxim	SOIC		1		778-1484	7.070	7,070
BAT1	Porte-pile PCB pour une pile CR2032				Keystone			1		430-875	2,110	2,110
J7	Barrier pour C jonction 2C5				Phoenix contact			1		220-4276	1.138	1,138
MCP3427	ADC				Microchip	MSOP		1		703-7649	2.820	2,820
24C32A	mémoire EEPROM				ON Semiconductor	SOIC		1		781-4528	0.342	0,342
CON-DILL28_EPI	Embase femelle à souder sur CI				Aprimatel EPI			1		451-348	3.238	3,238
FU1	Polyswitch				Littelfuse			1		617-3607	0.338	0,338
RF24G1F30	2.4 GHz transmitter module	3,3V			G-NiceRF			2		692	4.060	8,120

TOTAL HT FEUILLE			25,899
TOTAL HT GLOBAL			25,899
COEFFICIENT			1,000

	Nom	Date	Folio
Créé :	CHOLEZ-GLAB / GP MOLD	16/05/2018	1/1
Mis à jour :	CHOLEZ-GLAB / GP MOLD	23/05/2018	

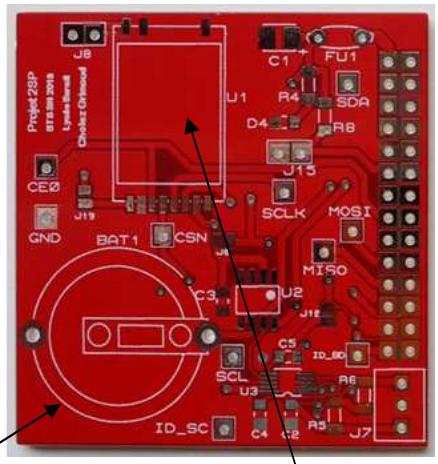
Procédure de soudure de la carte

Nous allons souder les composants CMS en étalant la pâte à braser grâce au pochoir sur le côté Bottom de la carte, car c'est sur ce côté qu'il y'a moins de composants CMS.

Dans un second temps nous allons étaler la pâte à braser grâce au pochoir, sur le côté top de la carte.

Ensuite nous allons souder les composants traversant en commençant par les plus petits.

Puis nous allons rajouter l'antenne au module RF2401F20 puis la pile sur le support de pile.



Support de pile

Module RF2401F20

7/ CONFIGURATION DE LA RASPBERRY ET DU MATÉRIEL

Premièrement nous devons activer l'I2C sur la carte Raspberry Pi

Utilisez votre Rpi en mode Super User :

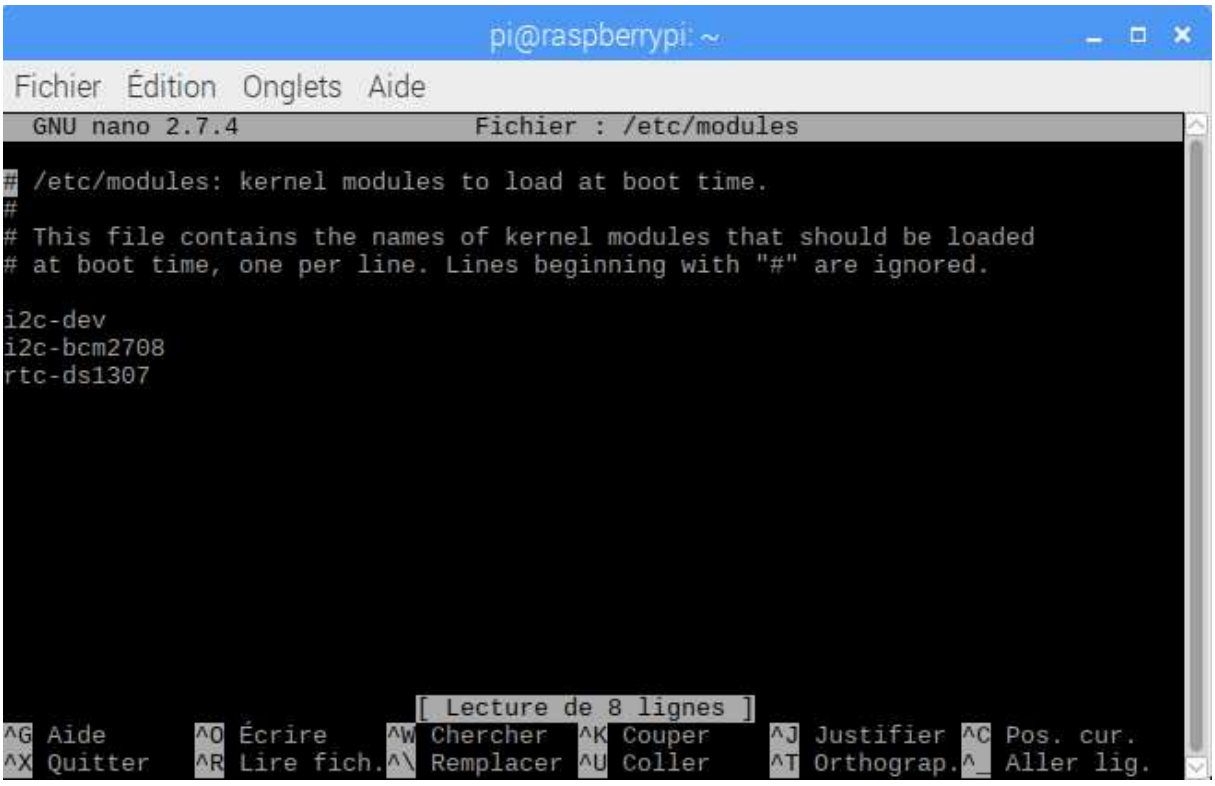
Une fois activé nous allons éditer le fichier /etc/modules avec l'éditeur nano (Nano est un simple éditeur de texte qui n'offre pas de fantaise).

Pour cela nous allons écrire la commande :
root@RaspBerrypi:/home/pi# nano /etc/modules

Nous allons ajouter deux lignes :

i2c-bcm2708 et i2c-dev

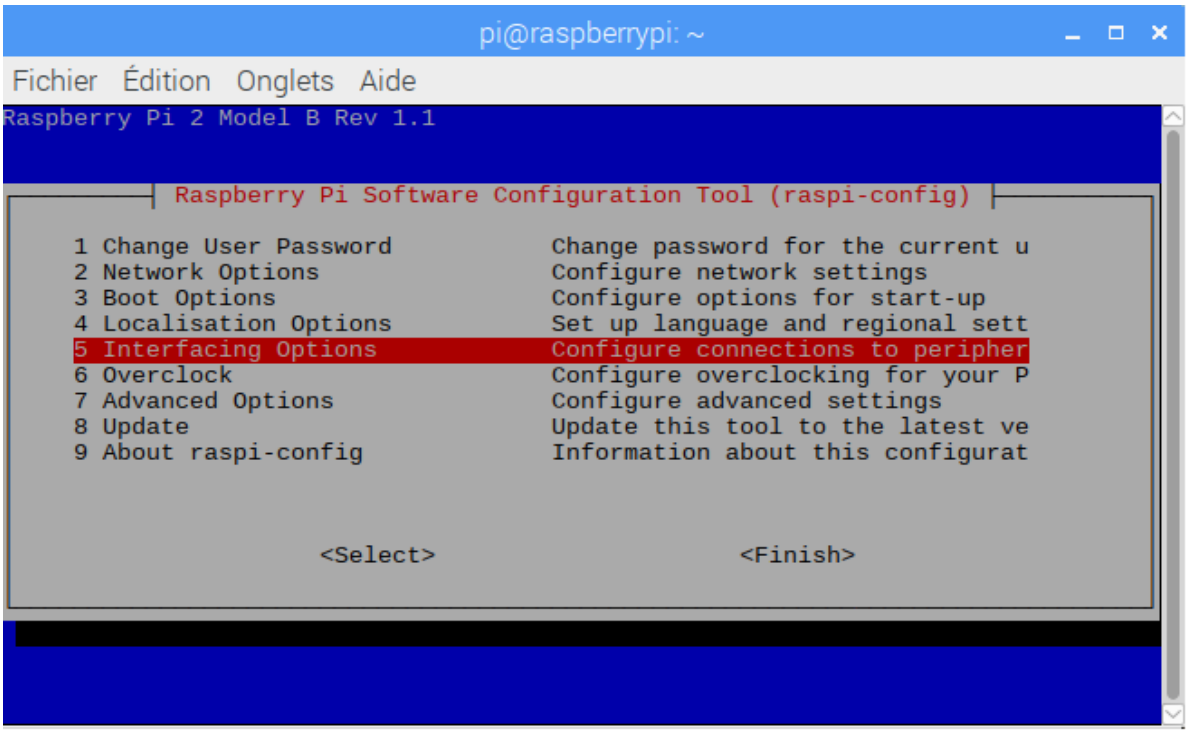
Comme cela :



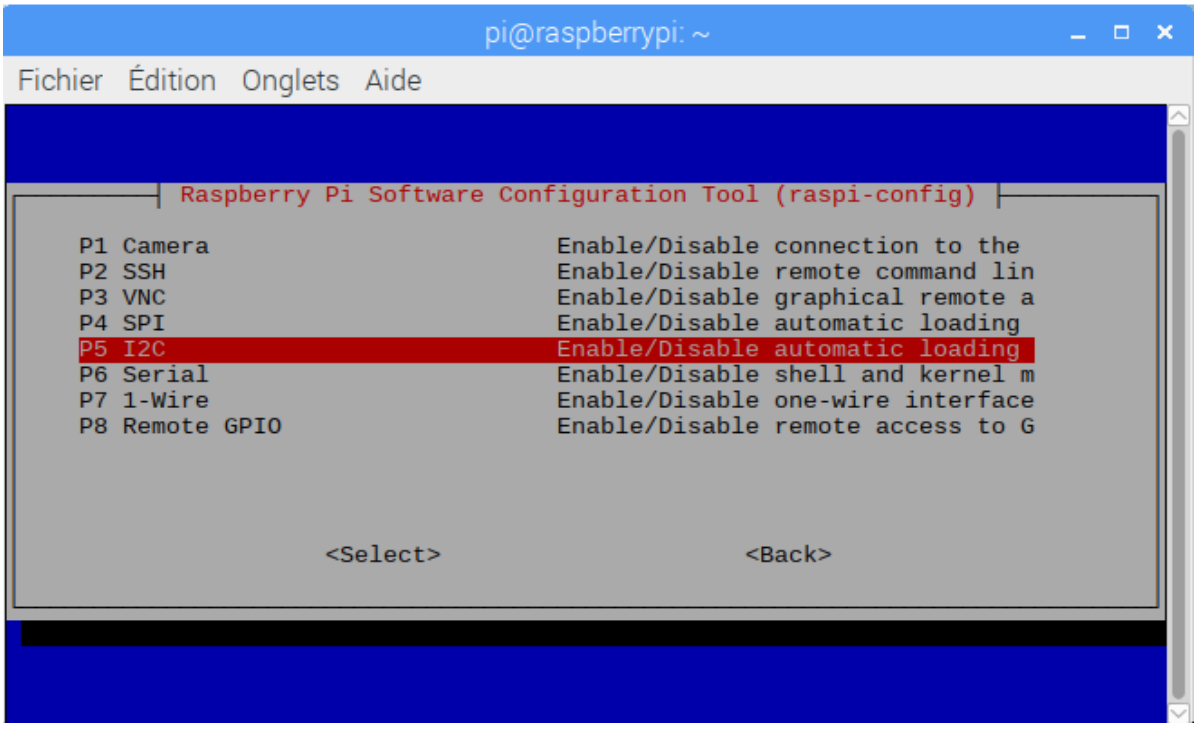
La ligne « rtc-ds1307 » est à rajouter ultérieurement dans le tutoriel.
Lorsque ces deux lignes sont rajoutées nous pouvons sauvegarder et quitter : Ctrl+X puis O et Entrée.

Sauvegardez puis lancer l'interface de configuration de la RaspBerry :
root@RaspBerrypi:/home/pi# raspi-config

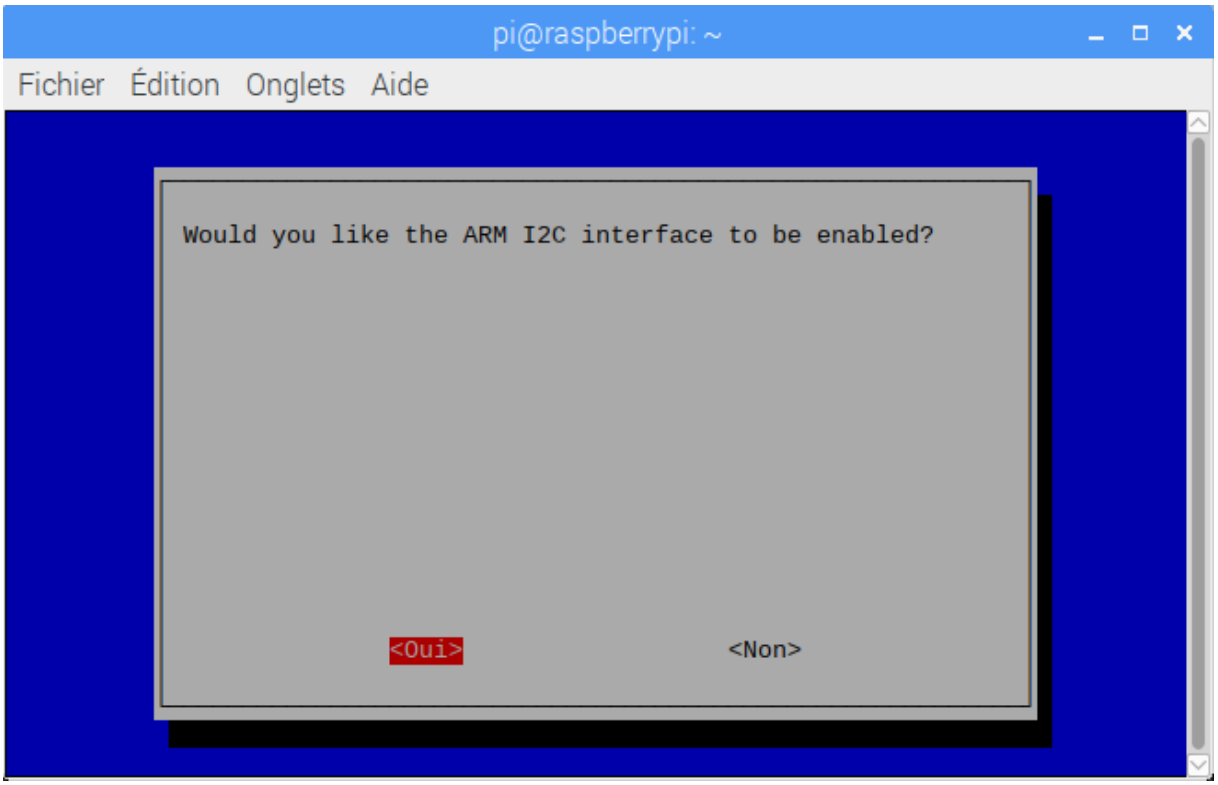
Entrez dans le menu « Interfacing Options ».



Sélectionnez P5 I2C (Cette option nous permet d'activer l'initialisation de l'I2C dès le démarrage de la RaspBerry)



Confirmer l'activation



Quitter puis redémarrer la RaspBerry :
root@RaspBerrypi:/home/pi# reboot

Nous allons ensuite vérifier le bon fonctionnement de l'I2C sur la carte :

root@RaspBerrypi:/home/pi# dmesg|grep i2c

```
root@raspberrypi:/home/pi# ls /dev/i2c*
/dev/i2c-1
```

root@RaspBerrypi:/home/pi# ls /dev/i2c*

```
root@raspberrypi:/home/pi# dmesg|grep i2c
[    4.326719] i2c /dev entries driver
```

Nous voyons donc que le Bus I2C fonctionne correctement comme le montrent ces commandes.

Initialisation de la RTC (Adaptation du tutoriel de CROTTE Johann)

Pour cela, nous allons écrire la valeur 0x04 dans le registre « control » qui est le 0x0E. Ainsi que la valeur 0x08 dans le registre « status » qui est le 0x0F.

root@RaspBerrypi:/home/pi# i2cset –y 1 0x68 0x0E 0x04 b root@RaspBerrypi:/home/pi# i2cget –y 1 0x68 0x0F b 0x88
root@RaspBerrypi:/home/pi# i2cset –y 1 0x68 0x0F 0x08 b root@RaspBerrypi:/home/pi# i2cget –y 1 0x68 0x0F b 0x08

```
root@raspberrypi:/home/pi# i2cset -y 1 0x68 0x0E 0x04b
Error: Could not set address to 0x68: Device or resource busy
root@raspberrypi:/home/pi# i2cget -y 1 0x68 0x0F b 0x88
Error: Could not set address to 0x68: Device or resource busy
root@raspberrypi:/home/pi# i2cset -y 1 0x68 0x0F 0x08 b
Error: Could not set address to 0x68: Device or resource busy
root@raspberrypi:/home/pi# i2cget -y 1 0x68 0x0F b 0x08
Error: Could not set address to 0x68: Device or resource busy
```

La ligne d'erreur est présente car les modifications ont déjà été effectuer au-préalable.

Nous allez maintenant pour voir installer l'horloge temps réel sur la RaspBerry afin que cet dernière démarre avec l'heure de celle-ci et non celle d'origine. Sans être raccorder à internet.

Pour cela nous allons brancher la RaspBerry à internet et taper la commande :

root@RaspBerrypi:/home/pi# echo ds13070x68 > /sys/class/i2c-adapter/i2c-1/new_device (Déclaration du nouvelle esclave i2c)

(Notez que dans la commande figure la référence du composant DS1307 qui est une autre horloge temps réel mais qui est compatible avec notre DS3231MZ+)

root@RaspBerrypi:/home/pi# hwclock –r (Lecture de l'heure sur la RTC)
root@RaspBerrypi:/home/pi# date (Lecture de l'heure sur la RaspBerry)
root@RaspBerrypi:/home/pi# hwclock –w (Ecriture de l'heure de la RaspBerry sur la RTC)
root@RaspBerrypi:/home/pi# hwclock –r (Lecture de l'heure sur la RTC pour vérifier le bon changement d'heure)

```
root@raspberrypi:/home/pi# hwclock -r
2018-05-23 08:55:34.002931+0200
root@raspberrypi:/home/pi# date
mercredi 23 mai 2018, 08:55:43 (UTC+0200)
root@raspberrypi:/home/pi# hwclock -w
root@raspberrypi:/home/pi# hwclock -r
2018-05-23 08:56:00.309017+0200
```

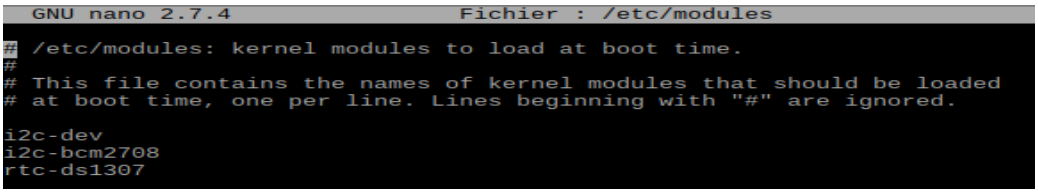
La RTC est désormais à l'heure et le restera même si elle est déconnectée de la Raspberry Pi.

Nous allons affecter cette RTC au démarrage de la RaspBerry pour que celle-ci dispose de l'heure y compris lorsqu'elle est déconnecté du réseau.

Vous allez éditer le fichier /etc/modules avec l'éditeur Nano comme vue précédemment.

root@RaspBerrypi:/home/pi# nano /etc/modules

Vous rajoutez la ligne : rtc-ds1307

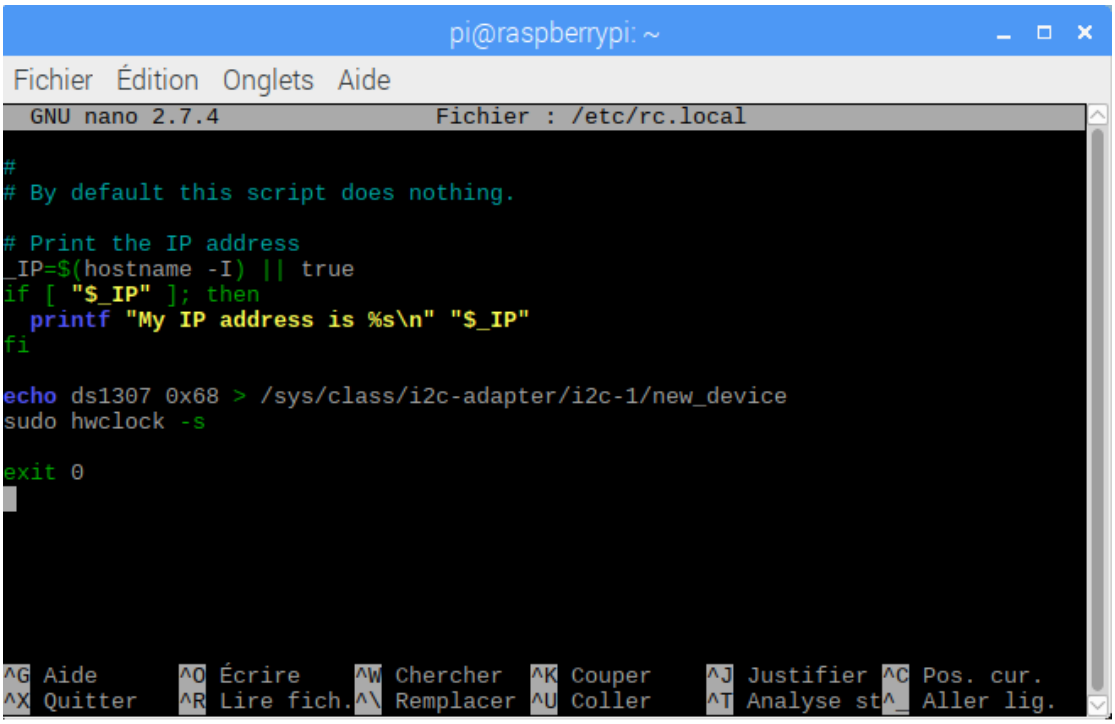


Lorsque cette ligne est rajoutée nous pouvons sauvegarder et quitter : Ctrl+X puis O et Entrée.
Vous éditez le fichier /etc/rc.local avec l'éditeur Nano

root@RaspBerrypi:/home/pi# nano **/etc/rc.local**
Avant la ligne : **Exit 0**
Rajoutez les lignes :

echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device sudo hwclock -s

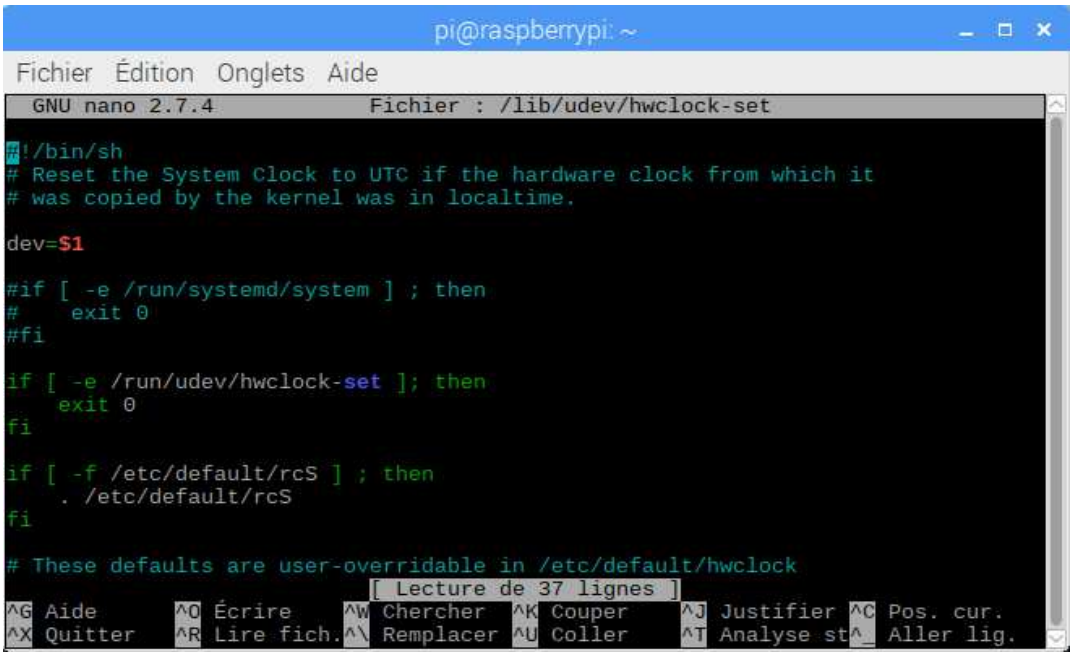
Comme ci-dessous :



Lorsque cette ligne est rajoutée nous pouvons sauvegarder et quitter : Ctrl+X puis O et Entrée.
Inclure dans le fichier /lib/udev/hwclock-set :
root@RaspBerrypi:/home/pi# nano /lib/udev/hwclock-set

if [/run/systemd/system] ; then
exit 0 fi
Et mettre en commentaire :
#if [e /run/systemd/system] ; then #exit 0
#fi

Comme ci-dessous :

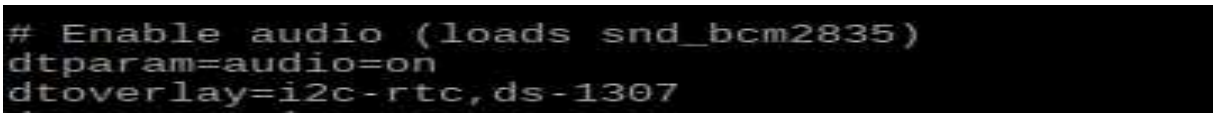


Lorsque cette ligne est rajoutée nous pouvons sauvegarder et quitter : Ctrl+X puis O et Entrée.

Modifier le fichier /boot/config.txt

root@RaspBerrypi:/home/pi# nano /boot/config.txt

Rajouter la ligne : *dtoverlay=i2c-rtc, ds-1307*



Lorsque cette ligne est rajoutée nous pouvons sauvegarder et quitter : Ctrl+X puis O et Entrée.

Vous pouvez maintenant essayer :

Éteignez votre RaspBerry attendez un moment, déconnecté la du réseau puis redémarrez là pendant qu'elle est toujours connecté à la RTC. La RaspBerry sera à la bonne heure et non à l'heure où elle s'est éteinte durant la dernière utilisation.

Journal de bord.		
Date	***Temps en heures***	***Objet***
Mercredi 10 Janvier 2018	**4 Heures**	- Présentation des projets (2h) - Prise de connaissance des documents (2h)
Jeudi 11 Janvier2018	**7 Heures**	- Prise de connaissance des documents (1h) - Réunion EC/IR pour discuter du projet (2h)
Vendredi 12 Janvier 2018	**10 Heures**	- Explication du projet par Mr Hortolland - Recherche sur le matériel utilisé
Mercredi 17 Janvier 2018	**14 Heures**	- Analyse du document 2SP 2017 (4h)
Jeudi 18 Janvier 2018	**17 Heures**	- Recherche sur l'horloge temps réel
Vendredi 19 Janvier 2018	**20 Heures**	- Mise en oeuvre de l'horloge temps reel depuis le tutoriel crée par des élèves de l'année précédente - Changement de la pile de l'horloge
Mercredi 24 Janvier 2018	**24 Heures**	- Suite mise en oeuvre de l'horloge temps réel
Jeudi 25 Janvier 2018	**27 Heures**	- Recherche sur le CAN (Convertisseur Analogique Numérique)
Vendredi 26 Janvier 2018	**30 Heures**	- Installation de la librairie BCM2835 - Prise d'informations sur le module NRF24l01
Mercredi 31 Janvier 2018	**34 Heures**	- Téléchargement des librairies Hackable - Cablâge du 1 er module et premier test
Jeudi 01 Février 2018	**37 Heures**	- Communication Arduino à Arduino 1 er module (OK) (3h)
Vendredi 02 Février 2018	**40 Heures**	- Absent
Mercredi 08 Février 2018	**43 Heures**	- Conception du code pour liaison de Raspberry à Raspberry (3h)
Vendredi 09 Février 2018	**46 Heures**	- Communication Raspberry à Raspberry 1 er module (3h)
Jeudi 15 Février 2018	**49 Heures**	- Communication Raspberry à Raspberry 1 er module (OK) (3h)
Vendredi 16 Février 2018	**52 Heures**	- Communication Arduino à Arduino 2 ème module (OK) (1h) - Communication Arduino à Raspberry2 ème module (OK) (2h)
Mercredi 21 Février 2018	**56 Heures**	- Communication Raspberry à Raspberry2 ème module(OK)
Vendredi 23 Février 2018	**59 Heures**	- Préparation dossier commun - Mise en forme dossier personnel
Mercredi 14 Mars 2018	**63 Heures**	- Réception nouveau CAN - Dossier personnel
Jeudi 15 Mars 2018	**66 Heures**	- Dossier personnel
Vendredi 16 Mars 2018	**69 Heures**	- Dossier personnel

Mercredi 21 Mars	**73 Heures**	- Dossier personnel
Jeudi 22 Mars	**76 Heures**	- Dossier personnel
Vendredi 23 Mars	**79 Heures**	- Revue N°1
Mercredi 28 Mars	**83 Heures**	- Correction et analyse des remarques de la première revue.
Jeudi 29 Mars	**86 Heures**	- Analyse de la documentation du MCP3427
Vendredi 30 Mars	**89 Heures**	- Câblage de la carte test du module MCP3427 de l'année dernière
Mercredi 4 Avril	**93 Heures**	- Essai du MCP3427
Jeudi 5 Avril	**96 Heures**	- Essai du MCP3427
Vendredi 6 Avril	**99 Heures**	- Test du MCP3427 sur la carte de l'année dernière
Mercredi 11 Avril	**103 Heures**	- Correction du programme par monsieur Hortolland et ajout de 2 potentiomètre sur la carte pour effectuer les tests.

Jeudi 12 Avril	**106 Heures**	`Valeur de CH1+ = 0.00231257 V`
		`Valeur de CH2+ = 2.048 V`
		
		`Valeur de CH1+ = 0.00237507 V`
		`Valeur de CH2+ = 2.048 V`
		
		`Valeur de CH1+ = 0.00237507 V`
		`Valeur de CH2+ = 2.048 V`
		
		`Valeur de CH1+ = 0.00237507 V`
		`Valeur de CH2+ = 2.048 V`
		Premier résultat obtenu avec les potentiomètre sur le MCP3427

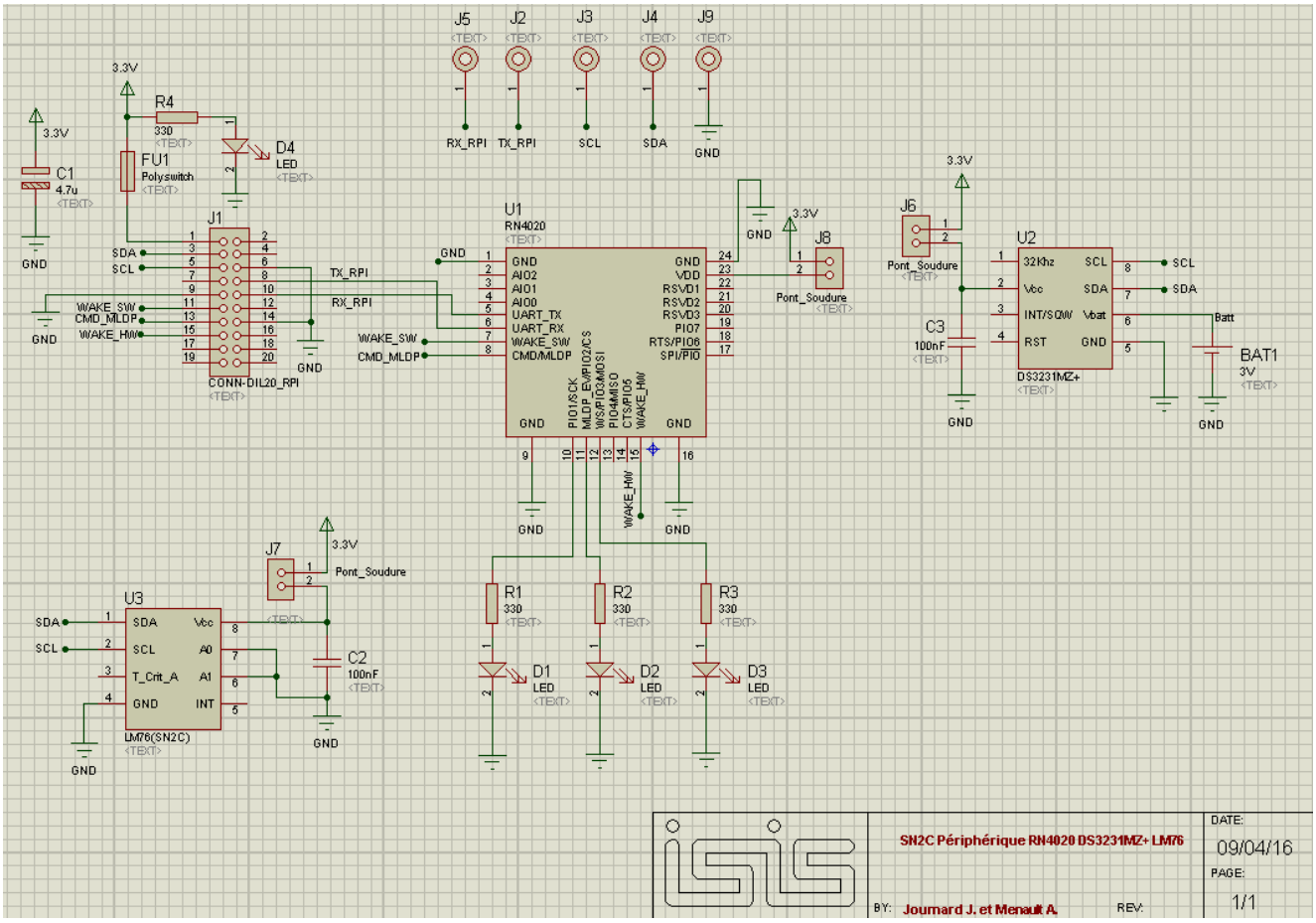
Vendredi 13 Avril	**109 Heures**	- Routage
Mercredi 18 Avril	**113 Heures**	- Routage de la carte sur ARES - Mise a jour des references des composants sur Proteus
Jeudi 19 Avril	**116 Heures**	- Routage de la carte sur ARES
Vendredi 20 Avril	**119 Heures**	- Routage de la carte sur ARES
Mercredi 9 Mai	**123 Heures**	- Nomenclature du projet
Vendredi 11 Mai	**126 Heures**	- Reception de la carte 2018 - Analyse de trame avec Saleae du MCP3427
Mercredi 16 Mai	**130 Heures**	- - Analyse de trame avec Saleae du MCP3427 et de la documentation pour décrypter une trame
Vendredi 18 Mai	**133 Heures**	- Dossier de la revue final
Mercredi 23 Mai	**137 Heures**	- Dossier de la revue final
Jeudi 24 Mai	**140 Heures**	- Dossier de la revue final
Vendredi 25 Mai	**143 Heures**	- Revue N°2

VIII/ Partie personnelle étudiant 4 (GRIMOUD Yohan)

1/ Présentation du travail à réaliser

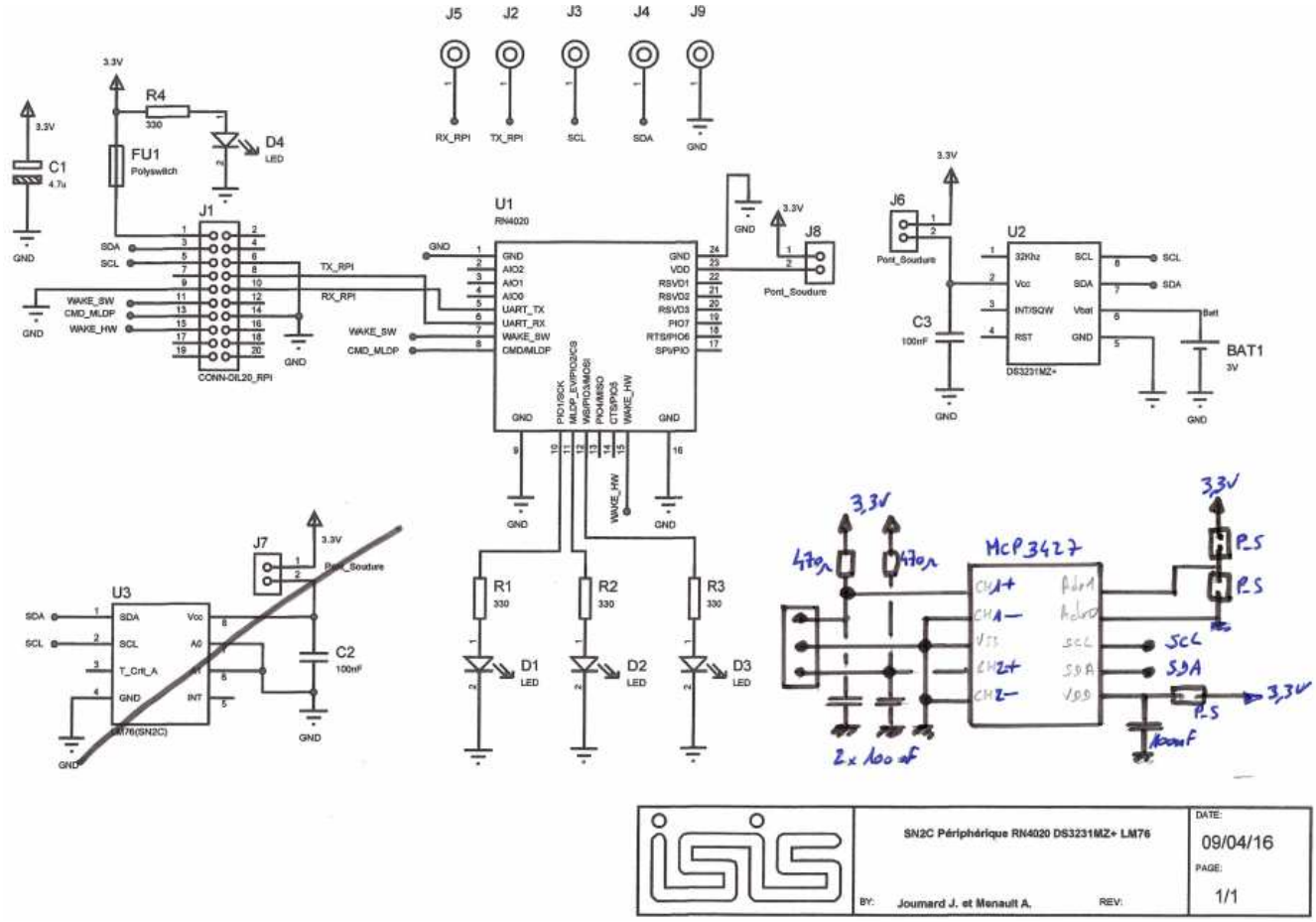
I) Travail réalisé en 2016 :

Le projet 2SP de l'année 2016 avait pour but de transférer des données en Bluetooth entre 2 Raspberry pi ainsi que l'horaire de ces transferts, les données étaient générées par un capteur de température.



II) Améliorations apportées en 2017 :

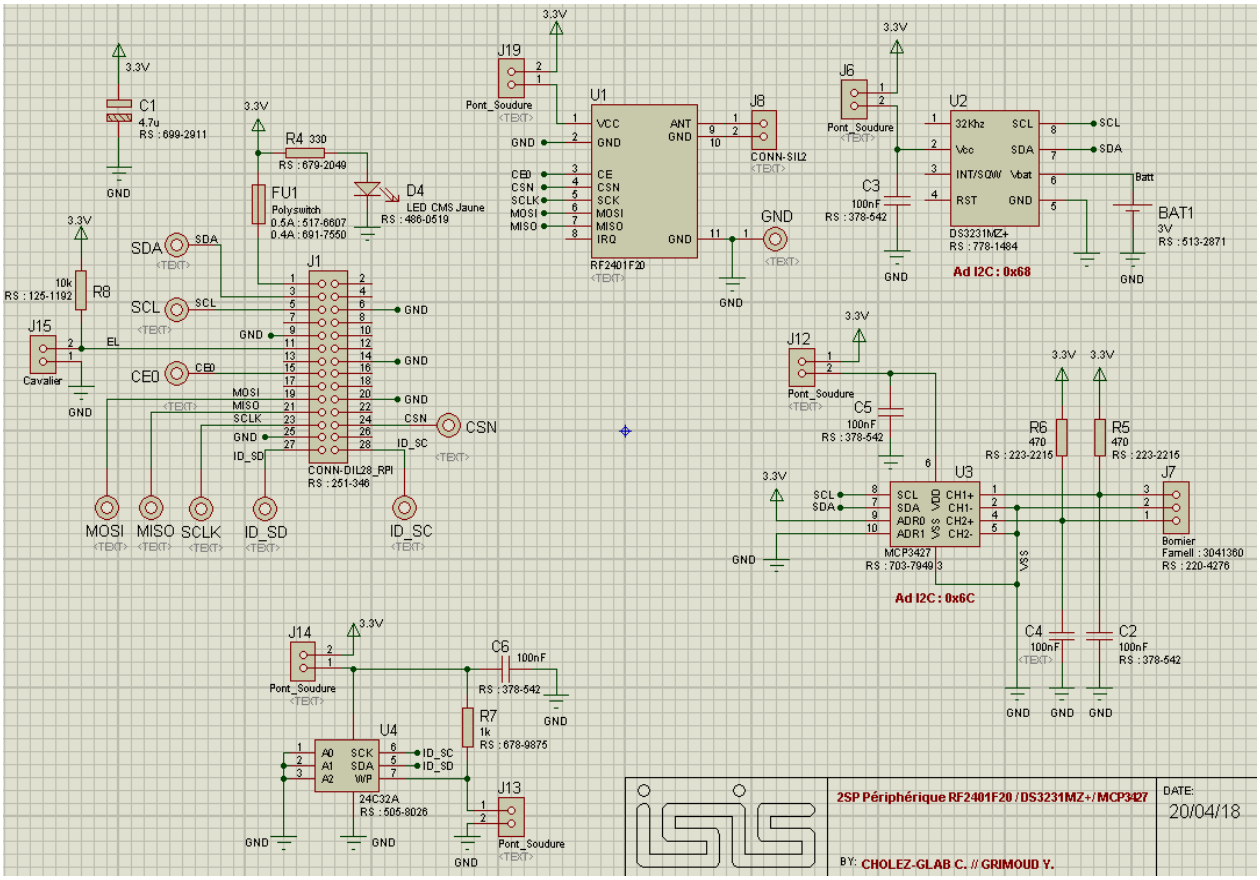
Schéma proposé par le client :



Le projet avait pour but d'être utilisé dans des conditions réelles, pour cela un convertisseur analogique numérique et un bornier ont été ajoutés pour récupérer et convertir les informations du capteur de température.

Au cours du projet il y a eu d'autres changements sur le schéma structurel, par exemple, le module Bluetooth est remplacé par un module de radio fréquence pour que la communication soit possible à plus grande distance. Les changements effectués sont présents sur le schéma ci-dessous.

Schéma structurel final :

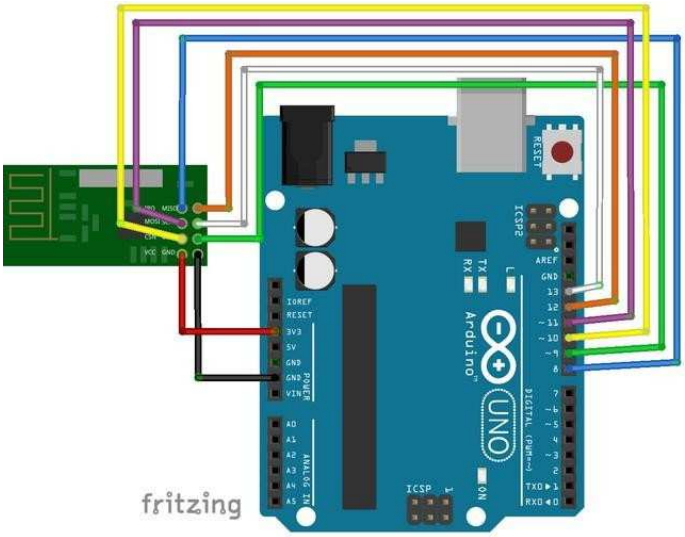


2/ Travail réalisé

Cette année la communication change de format puisqu'elle était en Bluetooth les années précédentes mais désormais elle sera par Radio Fréquence avec des modules à tester par prototypage sur Raspberry pi 2.

J'ai donc commencé par faire communiquer 2 Arduinos avec les modules nRF24l01 en m'aidant de l'exemplaire Hackable n°16 dans lequel le câblage du composant est schématisé puis j'ai récupéré les codes d'émissions et de réceptions des modules sur le site github.

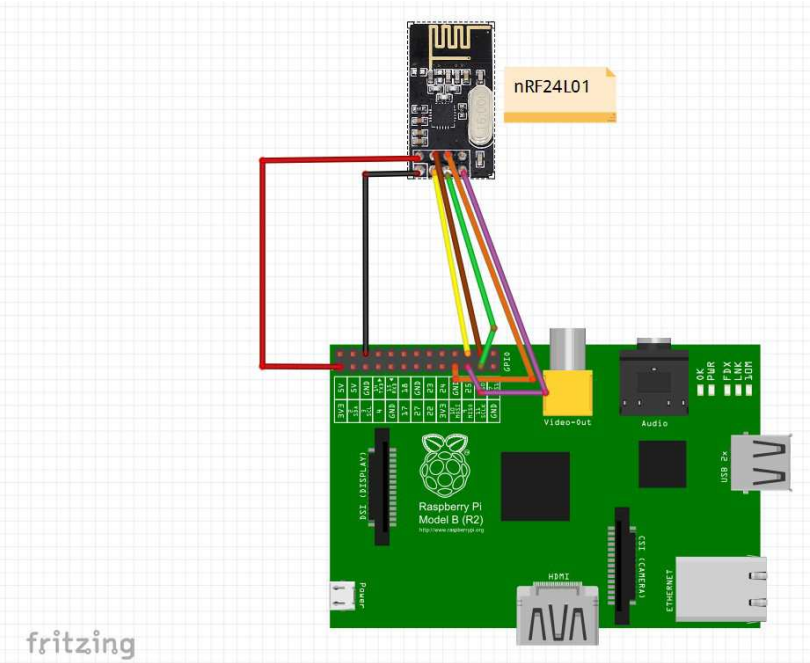
Schéma du câblage du module nRF24l01 sur Arduino :



La liaison entre 2 Arduinos est relativement simple puisque tout était à ma disposition mais ça a pris plus de temps que nécessaire à cause d'unmauvais câblage des différentes broches.

La seconde étape était de faire communiquer une Raspberry pi 2 avec une Arduino (avec la Raspberry pi 2 en émetteur et l'Arduino en récepteur). Encore une fois le code ainsi que le schéma des connexions étaient présents sur le manuel, de ce fait il n'y a eu aucune difficulté pour faire communiquer les modules.

Schéma du câblage du module nRF24l01 sur Raspberry pi 2 :



Code émetteur sur Arduino :

La troisième phase est donc de faire communiquer 2 Raspberry pi entre elles, contrairement aux 2 premières étapes, le code pour la partie réceptrice ne se trouve pas sur le site github, il faut donc prendre les deux codes de la partie émettrice pour trouver les différences entre ces programmes pour 'créer' un code récepteur compatible avec la Raspberry pi 2.

Code récepteur Arduino:

rfrec §

```
#include <SPI.h>
#include <RF24.h>
#define PIN_CS  10
#define PIN_CE  9
RF24 radio(PIN_CE,PIN_CS);
const uint64_t pipeA = 0xF0F0F0F0E1LL;
const uint64_t pipeB = 0xF0F0F0F0D2LL;

void setup() {
  Serial.begin(115200);
  Serial.println(F("Go go go!"));
    radio.begin();
    // Activer l'accusé réception
  radio.enableAckPayload();
    // L'accusé réception est dynamique
  radio.enableDynamicPayloads();
    // réglage de l'amplificateur
    //radio.setPALevel(RF24_PA_LOW);
  radio.setPALevel(RF24_PA_HIGH);
    // réglage des canaux (pipe)
  radio.openWritingPipe(pipeA);
  radio.openReadingPipe(1,pipeB);
    // En écoute
  radio.startListening();
}
void loop() {
  unsigned long temps;
     // Des données sont-elle disponibles ?
  if(radio.available()){
     // oui, lecture
  while (radio.available()) {
  radio.read(&temps, sizeof(unsigned long));
  }
     // arrête émission
  radio.stopListening();
     // envoi de la réponse
  radio.write(&temps, sizeof(unsigned long));
     // reprise de la réception
  radio.startListening();
     // Affichage
  Serial.print(F("pong !  "));
  Serial.println(temps);
  }
}
```

Code émetteur Arduino:

rfsend §

```
#include <SPI.h>
#include <RF24.h>
#define PIN_CS  10
#define PIN_CE  9

RF24 radio(PIN_CE,PIN_CS);

const uint64_t pipeA = 0xF0F0F0F0E1LL;
const uint64_t pipeB = 0xF0F0F0F0D2LL;

void setup() {
  Serial.begin(115200);
  Serial.println(F("Go go go!"));

  radio.begin();

  // Activer l'accusé réception
  radio.enableAckPayload();
  // L'accusé réception est dynamique
  radio.enableDynamicPayloads();

  // réglage de l'amplificateur
  radio.setPALevel(RF24_PA_LOW);
  //radio.setPALevel(RF24_PA_HIGH);

  // réglage des canaux (pipe)
  radio.openWritingPipe(pipeB);
  radio.openReadingPipe(1,pipeA);

  // En écoute
  radio.startListening();
}

void loop() {
  // Arrêt écoute
  radio.stopListening();
  Serial.println(F("Envoi"));

  // Utilisation de micro comme messahe
  unsigned long message = micros();
  // envoi
  if (!radio.write(&message, sizeof(unsigned long) )){
    Serial.println(F("erreur"));
  }
```

Code récepteur Rpi:

```
#include <cstdlib>
#include <iostream>
#include <sstream>
#include <string>
#include <unistd.h>
#include <RF24/RF24.h>


RF24 radio(22,0);


const uint64_t pipeA = 0xF0F0F0F0E1LL;
const uint64_t pipeB = 0xF0F0F0F0D2LL;


int main(int argc, char** argv){
//  unsigned long message;
//  unsigned long debut;
//  bool timeout;
//  unsigned long reception;
//  unsigned long fin;


printf("Go go go!\n");


radio.begin();


// Activer l'accusé réception
radio.enableAckPayload();
// L'accusé réception est dynamique
radio.enableDynamicPayloads();
// réglage de l'amplificateur
//radio.setPALevel(RF24_PA_LOW);
radio.setPALevel(RF24_PA_HIGH);
// réglage des canaux (pipe)
//radio.openWritingPipe(pipeA);
//radio.openReadingPipe(1,pipeB);


// affichage d'un résumé de la configuration
//radio.printDetails();


// En écoute
radio.startListening();


while(1) {
unsigned long temps;



// Des données sont-elles disponibles ?
 if(radio.available())
```

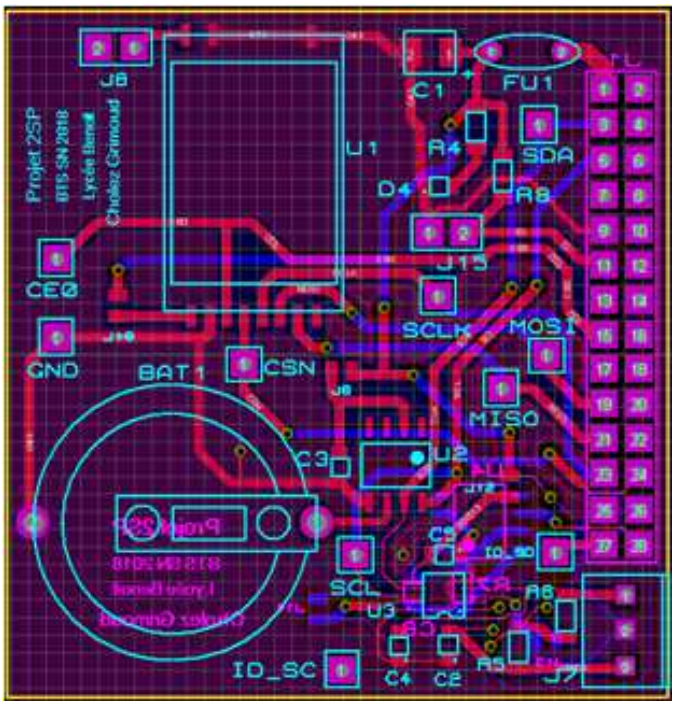
```
      {

//oui, lecture
while (radio.available())

                  {

                  radio.read(&temps, sizeof(unsigned long));

                   }


//Arrêt émission
radio.stopListening();


//Envoie de la réponse
radio.write(&temps, sizeof(unsigned long));
//Reprise de la réception
 radio.startListening();


//Affichage
 printf("pong ! \n");
 printf("temps: %lu\n", temps);
            }
      }

return 0;

}
```

Code émetteur Rpi:

```
#include <cstdlib>
#include <iostream>
#include <sstream>
#include <string>
#include <unistd.h>
#include <RF24/RF24.h>

RF24 radio(22,0);


const uint64_t pipeA = 0xF0F0F0F0E1LL;
const uint64_t pipeB = 0xF0F0F0F0D2LL;


int main(int argc, char** argv){
unsigned long message;
unsigned long debut;
bool timeout;
unsigned long reception;
unsigned long fin;


printf("Go go go!\n");
radio.begin();


// Activer l'accusé réception
 radio.enableAckPayload();
// L'accusé réception est dynamique
radio.enableDynamicPayloads();
// réglage de l'amplificateur
//radio.setPALevel(RF24_PA_LOW);
radio.setPALevel(RF24_PA_HIGH);


// réglage des canaux (pipe)
radio.openWritingPipe(pipeB);
radio.openReadingPipe(1,pipeA);


// affichage d'un résumé de la configuration
radio.printDetails();


// En écoute
radio.startListening();


while(1) {
  // Arrêt écoute
  radio.stopListening();


 printf("Envoi\n");
 message = millis();
```

```
// Envoi avec vérification d'erreur
if(!radio.write(&message, sizeof(unsigned long))) {
        printf("Erreur : write.\n");

}

// Mise en écoute
radio.startListening();


// Enregistrement du début du délai
debut = millis();
timeout = false;


// Tant que rien à lire
while (!radio.available()) {
if (millis()-debut > 600) {
timeout = true;
break;

      }
 }

// abandon pour cause de délai écoulé ?
if(timeout){

  printf("Erreur : timeout.\n");

}
else {
  // lecture des données reçues
radio.read(&reception, sizeof(unsigned long));
// enregistrement du moment de la réception
fin = millis();


// Affichage
      printf("Message: %lu, reponse: %lu, temps complet: %lu\n", message,
reception, fin-message);
}

// dodo 1 seconde
sleep(1);

 }

 return 0;

}
```

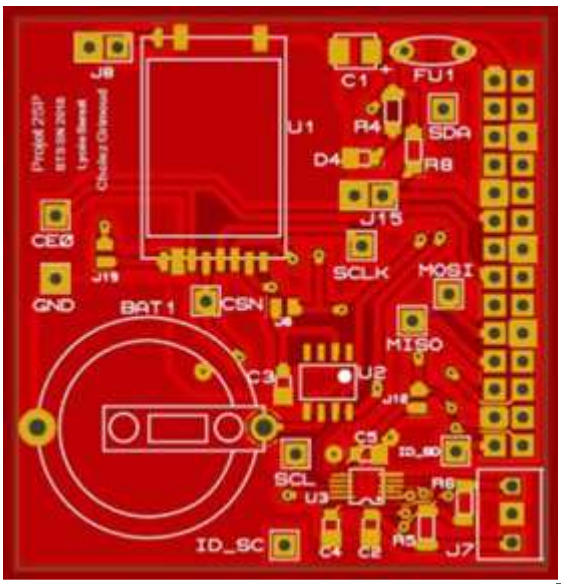
3/ Routage et visualisation de la carte



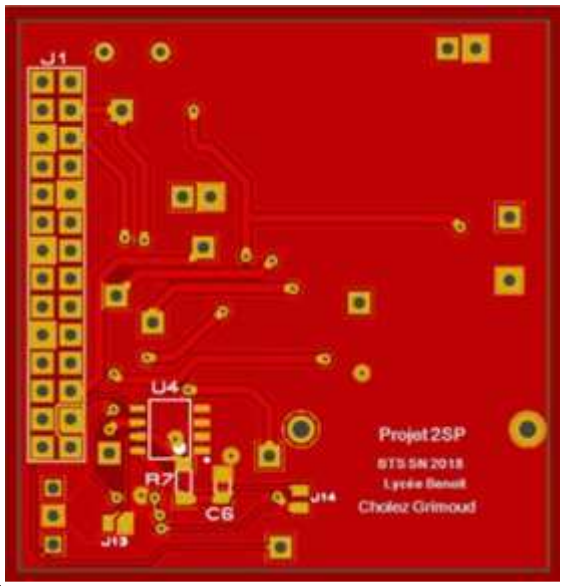
Le routage est similaire à celui de CHOLEZ-GLAB par manque de temps, nous avons dû le réaliser en commun.

Version de la carte avec Gerber Viewer

Côté top



Côté bottom



4/ Commande de la carte

PCB Cost	USD$4.90
Base Material	FR-4 TG130
No. of Layers	2 layers
PCB Dimensions	100mm * 100mm
PCB Quantity	10
No. of Different Designs	1
PCB Thickness	1.6mm
PCB Color	Red
Surface Finish	HASL
Minimum Solder Mask Dam	0.4mm↑
Copper Weight	1oz.
Minimum Drill Hole Size	0.3mm
Trace Width / Spacing	6/6 mil
Blind or Buried Vias	No
Plated Half-holes / Castellated Holes	No
Impedance Control	No
Sub-Total	USD$4.90
Production Time	5 – 6 Working Days
Weight	0.32kg
Shipping	Calculated at Checkout

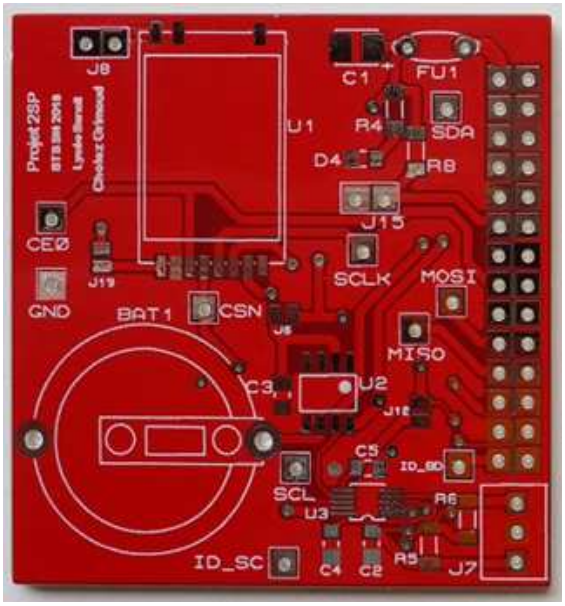
Add to Cart

La commande a été faite en Asie pour des questions financières, en effet le fait de faire sous-traiter ce modèle de carte en France ou en Europe coûte trop cher.

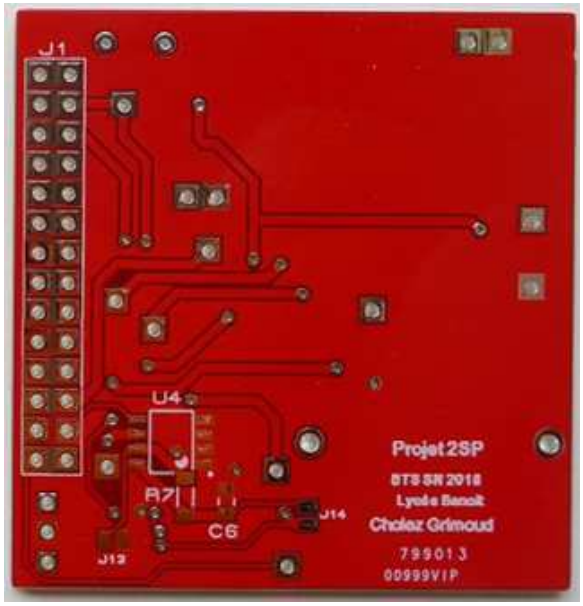
Le pochoir ci-contre est utile pour souder les composants CMS avec de la pâte à braser.

5/ Carte finale

Nous avons reçu la version PCB terminée de la carte le 11 mai 2018.

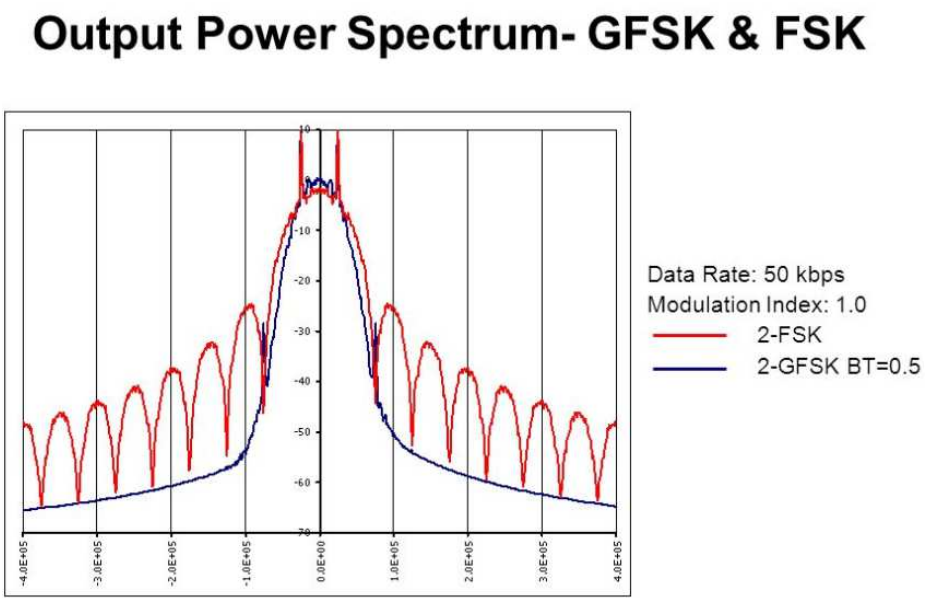


Côté top
de la carte



Côté bottom
de la carte

6/ Partie physique



7/ Conclusion

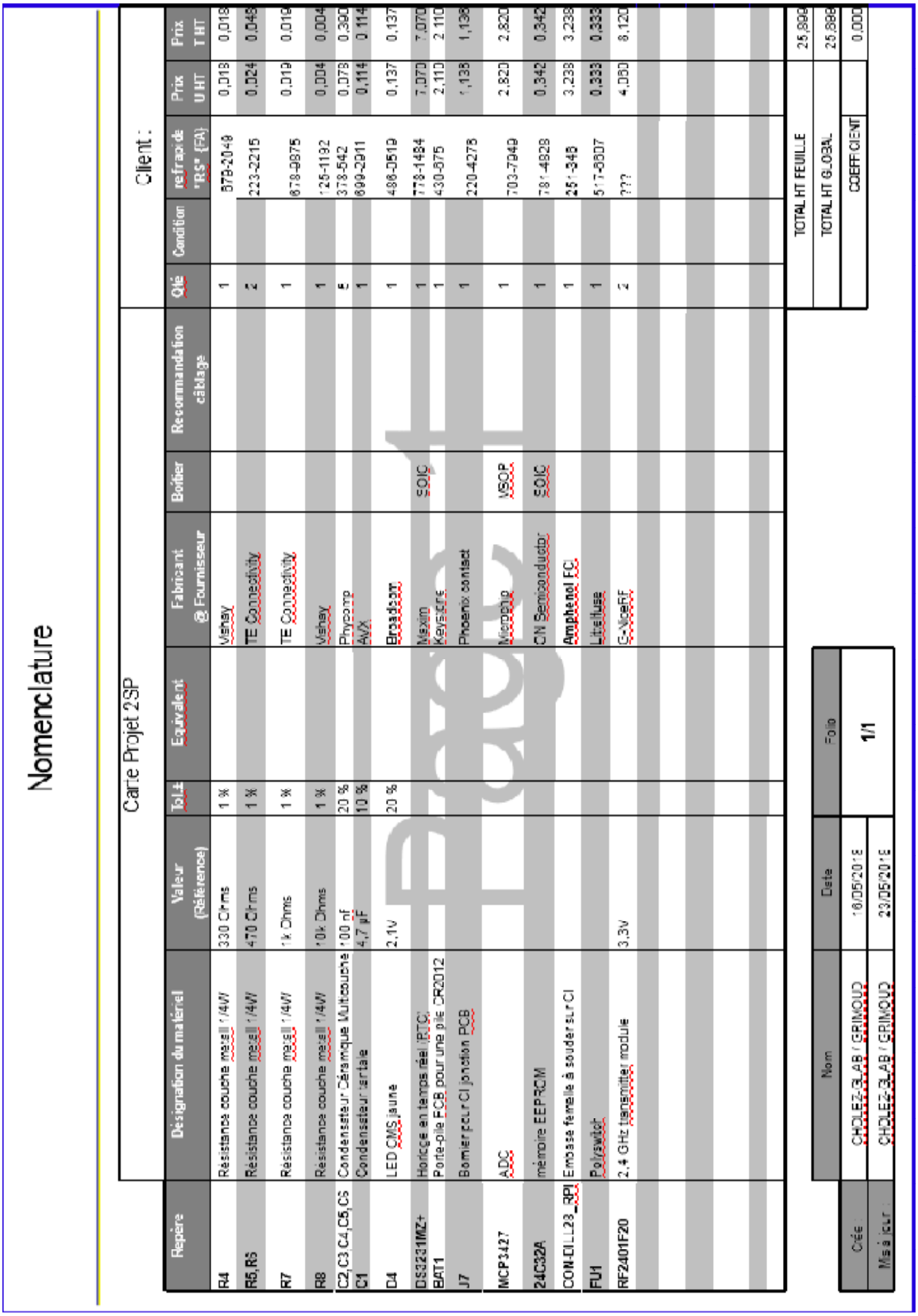
Le projet est très intéressant mais à cause de nombreux problèmes rencontrés et un manque d'organisation de ma part je ne me suis pas occupé de toute ma partie personnelle. Malgré ça j'ai quand même appris beaucoup en soudure ou encore en routage.

Bilan du projet : - Schéma structurel de la carte terminé

- Routage de la carte terminé
- Carte reçu le 11 mai 2018
- Mémoire EEPROM (HAT) pas terminé

8/ Annexes

Nomenclature :



Nomenclature

Carte Projet 2SP

Client :

Repère	Désignation du matériel	Valeur (Référence)	Tol±	Equivalent	Fabricant & Fournisseur	Boîtier	Recommandation câblage	Qté	Condition	refrac de "RS" (FA)	Prix U HT	Prix T HT
R4	Résistance couche metall 1/4W	330 Ohms	1 %		Vishay			1		678-2048	0,018	0,018
R5,R6	Résistance couche metall 1/4W	470 Ohms	1 %		TE Connectivity			2		223-2215	0,024	0,048
R7	Résistance couche metall 1/4W	1k Ohms	1 %		TE Connectivity			1		678-9875	0,019	0,019
R8	Résistance couche metall 1/4W	10k Ohms	1 %		Vishay			1		125-1132	0,004	0,004
C2,C3,C4,C5,C6	Condensateur Céramique Multicouche	100 nf	20 %		Phycomp			5		378-542	0,078	0,390
C1	Condensateur tantale	4,7 µF	10 %		AVX			1		699-2911	0,114	0,114
D4	LED CMS jaune	2,1V	20 %		Broadcom	SOIC		1		486-0519	0,137	0,137
DS3231MZ+	Horloge en temps réel (RTC)				Maxim			1		778-1484	7,070	7,070
BAT1	Porte-pile PCB pour une pile CR2012				Keystone			1		430-875	2,110	2,110
J7	Bornier pour CI jonction PCB				Phoenix contact			1		220-4278	1,138	1,138
MCP3427	ADC				Microchip	MSOP		1		703-7949	2,820	2,820
24C32A	mémoire EEPROM				ON Semiconductor	SOIC		1		781-4828	0,342	0,342
CON-DILL23_3PI	Embase femelle à souder sur CI				Amphenol FCI			1		251-348	3,238	3,238
FU1	Polyswitch				Littelfuse			1		517-3807	0,333	0,333
RF2401F20	2.4 GHz transmitter module	3.3v			G-NiceRF			2		???	4,060	8,120

	Nom	Date	Folio
Crée	CHOLEZ-BLAB / GRIMOUD	18/05/2018	1/1
Modifié	CHOLEZ-BLAB / GRIMOUD	23/05/2018	

TOTAL HT FEUILLE	25,998
TOTAL HT G.03AL	25,998
COEFFICIENT	0,000

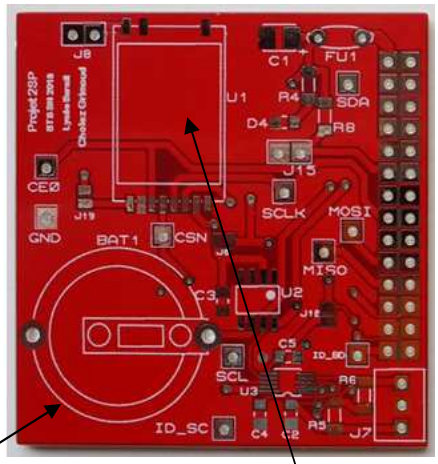
Procédure de soudure de la carte

Nous allons souder les composants CMS en étalant la pâte à braser grâce au pochoir sur le côté Bottom de la carte, car c'est sur ce côté qu'il y'a moins de composants CMS.

Dans un second temps nous allons étaler la pâte à braser grâce au pochoir, sur le côté top de la carte.

Ensuite nous allons souder les composants traversant en commençant par les plus petits.

Puis nous allons rajouter l'antenne au module RF2401F20 puis la pile sur le support de pile.



Support de pile

Module RF2401F20

Journal de bord :

10/01/2018	- Présentation des projets - Prise de connaissance des documents
11/01/2018	- Prise de cnnaissance des documents - Réunion EC/IR pour discuter du projet
12/01/2018	- Explication du projet par Mr Hortolland
17/01/2018	- Analyse du document 2SP 2017
18/01/2018	- Prise d'informations sur le module nRF24l01 +
19/01/2018	- Initialisation de la carte SD
24/01/2018	- Installation des outils Raspberry
25/01/2018	- Installation de la librairie bcm2835 - Prise d'informations sur le module nRF24l01 +
26/01/2018	- Téléchargement des librairies Hackable - Câblage du 1 er module et premier test
31/01/2018	-Absent
01/02/2018	- Communication Arduino à Arduino 1 er module (OK)
02/02/2018	- Communication Arduino à Raspberry 1 er module (OK)
08/02/2018	- Conception du code pour liaison de Raspberry à Raspberry
09/02/2018	- Communication Raspberry à Raspberry 1 er module
15/02/2018	- Communication Raspberry à Raspberry 1 er module (OK)
16/02/2018	- Communication Arduino à Arduino 2 ème module (OK) - Communication Arduino à Raspberry 2 ème module (OK)
21/02/2018	- Communication Raspberry à Raspberry 2 ème module (OK)
23/02/2018	- Dossier personnel
14/03/2018	- Dossier personnel
15/03/2018	-Dossier personnel
16/03/2018	- Dossier personnel

23/03/2018	- 1 ère Revue
28/03/2018	- Correction et analyse des remarques de la Revue n°1
29/03/2018	- Conception du schéma stucturel de la carte
30/03/2018	- Don du sang
04/04/2018	- Scéma structurel
05/04/2018	- Scéma structurel
06/04/2018	- Scéma structurel
11/04/2018	- Scéma structurel
12/04/2018	- Scéma structurel
13/04/2018	- Début du routage sur ARES
18/04/2018	- Routage sur ARES - Mise à jour de la carte des composants sur Proteus
19/04/2018	- Routage sur ARES
20/04/2018	- Fin du routage sur ARES - Commande de la carte
09/05/2018	- Conception de la nomenclature de la carte
11/05/2018	- Réception de la carte (version 2018)
16/05/2018	- Dossier personnel
17/05/2018	- Dossier personnel
18/05/2018	- Dossier personnel
23/05/2018	- Dossier personnel
24/05/2018	- Dossier personnel + diapo