### RAYNAL Audran SFL5

|  |
| --- |
| Groupe Olivier |
| Dossier technique du projet – Partie personnelle |
| Etudiant 4 : Visualisation de l’état du système et acquisition de mesure par capteur |

Table des matières

[I - Situation dans le projet 3](#_Toc9509275)

[A – Synoptique de la réalisation 3](#_Toc9509276)

[B-Rappel des tâches de l’étudiant 4](#_Toc9509277)

[II - Réalisation de l’acquisition des mesures 5](#_Toc9509278)

[A – fonctionnement du capteur 5](#_Toc9509279)

[B – Le montage 6](#_Toc9509280)

[C- L’acquisition des mesures 7](#_Toc9509281)

[D – Etalonnage 8](#_Toc9509282)

[III - Réalisation de l’Application Android 9](#_Toc9509283)

[A - Questionnement 9](#_Toc9509284)

[B - Fonctionnement de l’application 11](#_Toc9509285)

[C – Code Android Studio 12](#_Toc9509286)

[IV – Diagramme de classes 14](#_Toc9509287)

[V - Test Unitaire 15](#_Toc9509288)

[VI - Fiche recette 16](#_Toc9509289)

[CONCLUSION 17](#_Toc9509290)

[Communication avec le groupe 17](#_Toc9509291)

[Connaissances apportées 17](#_Toc9509292)

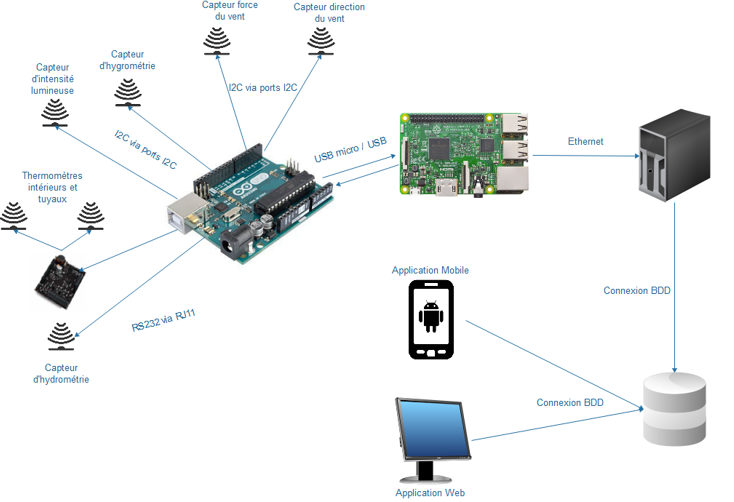
[Taches restantes 17](#_Toc9509293)

[Difficultés rencontrés 18](#_Toc9509294)

[Regard critique du projet 18](#_Toc9509295)

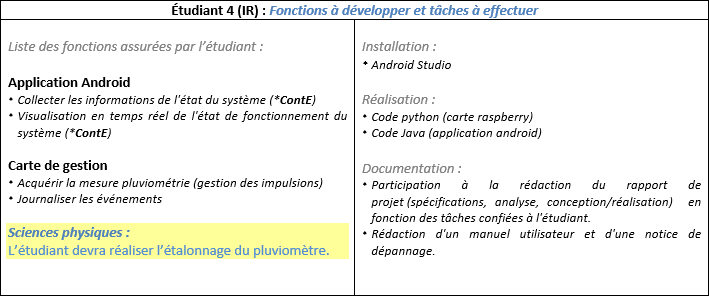
# I - Situation dans le projet

## A – Synoptique de la réalisation



*Voici les parties de la synoptique me concernant.*

## B - Rappel des tâches de l’étudiant



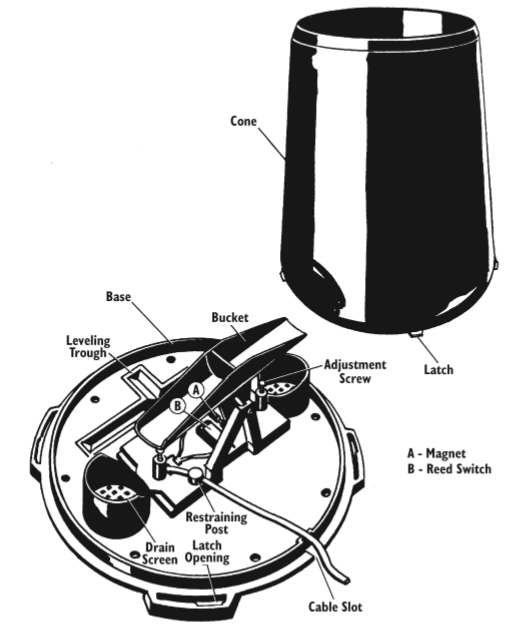
Dans le projet, ma principale tâche est le développement d’une application sur smartphone Android. L’application est développée avec le logiciel Android Studio, l’IDE officiel de développement sous Android. L’application aura comme objectif de permettre à l’utilisateur de visualiser en temps réel l’état du système qui est composé des capteurs, de la carte de gestion et du serveur. Elle récupère les valeurs de l’état du système et les affiches.

Un capteur d’Hydrométrie m’a été confié. Il s’agit du Pluviomètre Rain Collector II de la société Davis. Je dois acquérir la mesure pluviométrie grâce à ses impulsions. De plus un étalonnage est à réaliser, c’est-à-dire que pour une certaine quantité d’eau, je dois connaitre la hauteur au sol que cela représente en millimètre.

Pour réaliser cette tâche, il est nécessaire d’utiliser une carte d’acquisition Arduino. Le programme a été développé grâce à l’IDE Arduino.

# II - Réalisation de l’acquisition des mesures

## A - fonctionnement du capteur

Ceci est le schéma de la documentation officiel de Davis Instruments. Il présente les différents composants de mon capteur, le Rain Collector II.

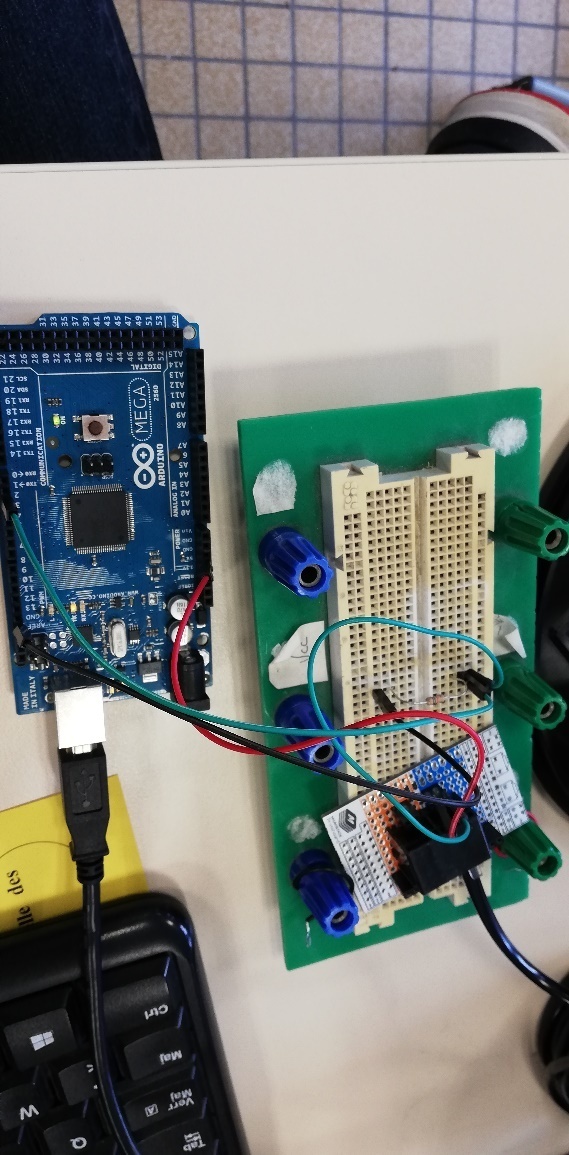
Les composants permettant la mesure d’Hydrométrie sont le switch magnétique et l’aimant situé sous la bascule. Ils sont représentés sur le schéma en position A et B. L’aimant permet sous certaines conditions de réagir avec le switch magnétique qui enregistrera une impulsion électrique qui sera transmise à la carte d’acquisition Arduino via un câble RJ11.

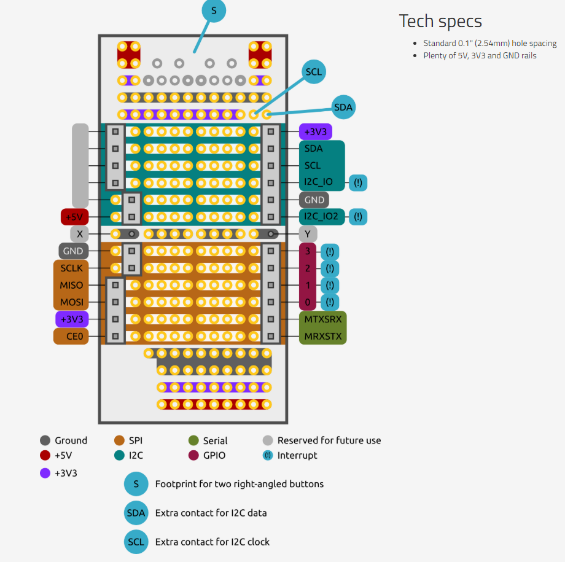
Le cône sert à collecter l’eau de pluie. Elle sera dirigée vers la bascule qui, avec suffisamment d’eau basculera en la laissant couler, et entraînera dans son mouvement l’aimant. En changeant de position, il permettra d’effectuer une impulsion sur le switch magnétique.

## https://image.tubefr.com/upload/3/b5/3b501c1d7e445b24090de40c90b2b6a5.jpgB - Le montage

Source : https://www.instructables.com

Ce schéma est le modèle sur lequel le montage final a été basé. La carte d’acquisition est une carte Arduino, le montage à pu être réalisé sur un modèle MEGA ou UNO. J’ai eu besoin d’une platine d’essai et d’une Microstack protoboard. Cette dernière est un petit module facilitant la connectique. On y a soudé des connecteurs RJ11 femelles et des broches permettant de recevoir le RJ11 mâle du pluviomètre.

De plus, il est nécessaire d’ajouter au montage, pour corriger la différence de tension : une résistance entre 10K et 15k Ohms. Celle utilisée ici est de 15K Ohm. Le montage étant simple, il n’y a pas besoin de Shield, il suffit de bien brancher les pins.

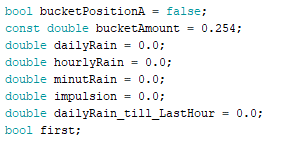


Ci-dessus à gauche la photo du montage, et à droite le schéma technique de la protoboard.

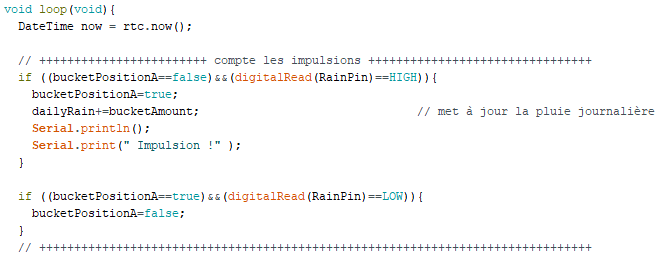
## C - L’acquisition des mesures

Après le montage terminé, j’ai pu entreprendre le programme en lui-même.

Le programme repose sur un principe assez simple, la position de l’aimant. En effet, d’après le fonctionnement du pluviomètre, l’aimant en changeant de position dû à la pluie présente sur la bascule, fait émettre une impulsion au switch magnétique. C’est ainsi qu’on saura qu’une quantité de pluie précise est tombé.



Voici sur cet extrait la déclaration des variables du programme. Il y a plusieurs double qui signifie le taux de pluie sur une minute, sur une heure, et sur la journée. Mais aussi le taux de pluie de la dernière heure.



Sur cet extrait du programme, on peut voir le Boolean bucketPositionA. C’est lui qui représente la position de l’aimant quand il passe devant le switch. Si il prend la valeur true, c’est-à-dire qu’il vient de bouger, et donc qu’il pleut. On incrémente donc le double dailyRain, la pluie journalière et on affiche un texte signifiant qu’il y a une impulsion.

Enfin, à minuit chaque jour, le taux de pluie de la journée est remis à zéro. Et le taux de pluie de la dernière heure aussi, pour éviter d’avoir un taux de pluie négatif à 1h du matin.

## D - Etalonnage

Pour réaliser l’étalonnage du pluviomètre, il a fallu effectuer quelques tests et calcul. Voici la procédure servant à simuler les conditions réelles de la pluie, le capteur n’était pas connecté :

* Prendre une bouteille d’eau de 1L (totalement remplie mais la quantité d'eau totale peut varier dans de très légère proportion)
* Verser la totalité de l’eau dans le cône du pluviomètre
* Compter le nombre d’impulsions (la bascule changeant de position émet un bruit distinctif)

Le test à été effectué 3 fois, dans un lavabo de petite taille, en changeant le débit d’eau pour vérifier si le résultat évoluait avec la vitesse de versement. Le premier test à compté 143 impulsions avec un débit lent. Le deuxième test 150 avec un débit moyen et le troisième et dernier 154 avec un débit un peu plus rapide.

Il est possible que lors des tests des impulsions ont été manqués, ou que le nombre total ait une erreur.

En prenant le résultat du deuxième test, 150, comme base il y a une différence de -5% et de 2 % avec le premier et dernier tests. Donc compte tenu que les nombres sont proches, une moyenne à été établie avec ces 3 résultats.

143 + 150 + 154 = 447

447 / 3 = 149

De plus, selon la documentation, pour chaque impulsion, le pluviomètre considère que 0.01 inches, soit 0.254mm d’eau est tombée. Un adaptateur métrique peut être installé car le capteur est une version américaine, mais il n’a pas été installé.

149 \* 0.254 = 37.846 millimètres

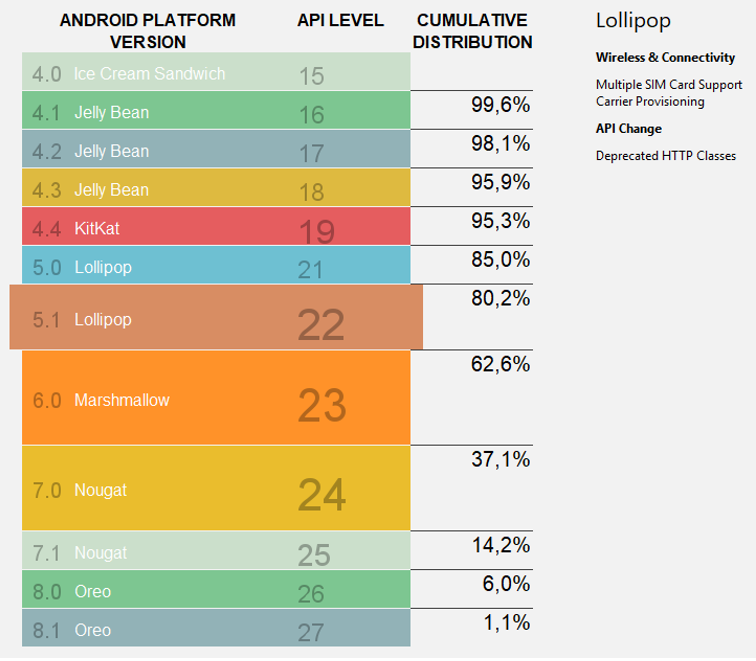
Donc avec un litre d’eau, il y a 3.7846 centimètres d’eau dans le lavabo de test.

# III - Réalisation de l’Application Android

## A - Questionnement

La première question a été de savoir sur quel niveau de l’API Android allait être développé l’application. Un API de développement *(Application Programming Interface)* est une interface de programmation constitué d’un ensemble de fonction.

Le choix de l’API de programmation minimum pour mon application s’est porté sur 2 critères : le plus important est la distribution globale de l’API.

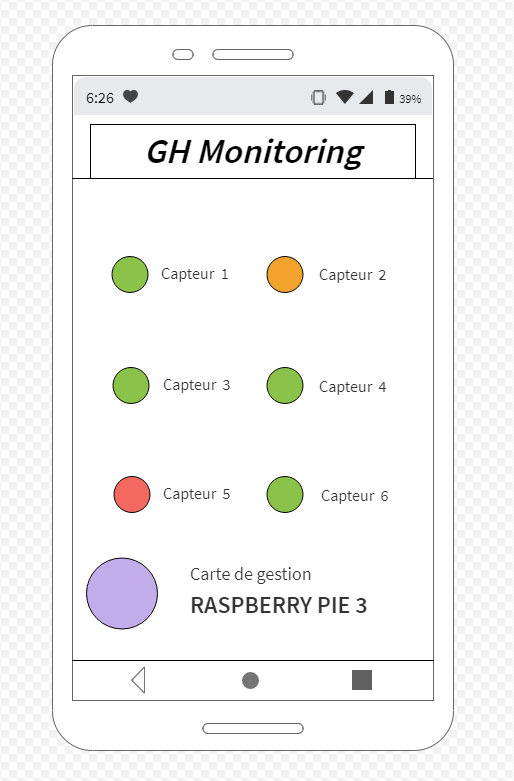
L’application n’étant pas destiné au grand public mais à seulement quelques utilisateurs professionnels, on doit pouvoir l’utiliser sur le plus de modèle de smartphone Android possible. Le second critère est l’âge de l’API, car il doit être assez récent pour que je puisse profiter d’un maximum de fonctionnalité et de méthode de développement. Il fallait faire un compromis et donc l’API de niveau 22, Android 5.1 Lollipop a été choisie, car largement répandu, et assez récente car cette version d’Android est sortie en 2015.

Ce tableau récapitulatif peut être trouvé au démarrage d’un projet sur Android Studio mais aussi sur le site officiel de développeurs Android.

Ensuite, la deuxième question était de savoir comment présenter visuellement l’application.

L’objectif de l’Application est simple : il faut afficher en temps réel l’état du système et de ses composants : La carte de Gestion et les 6 capteurs la composant. C’est-à-dire que l’utilisateur de l’application doit être informé rapidement et simplement si un composant ne fonctionne plus.

Ainsi, il a été décidé de réaliser des indicateurs avec un code couleur. Si la pastille de couleur est verte, il n’y a pas de problème détecté, le composant fonctionne. Si elle est rouge, il y a un problème et une intervention de maintenance est requise. Il n’y aura qu’une seule fenêtre, pour garder la simplicité.

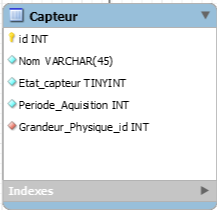


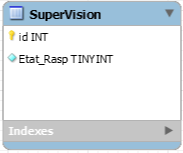
GH monitoring signifie Surveillance de Serre *(Greenhouse monitoring)*

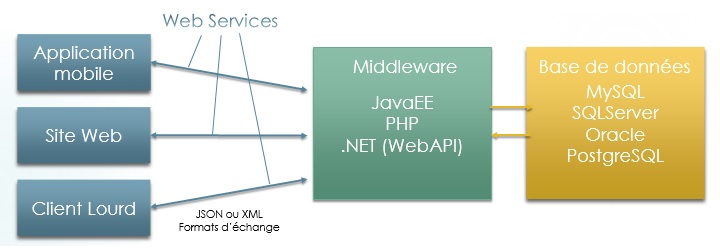
Le prototype d’IHM présenté ci-dessus a servi de base pour le développement de l’application.

## B - Fonctionnement de l’application

Le projet SFL5 et le projet SFL6 partagent la même base de données. Les états des capteurs y sont stockés tout comme l’état de la Raspberry. Le champ *Etat\_capteur* fait partie de la table Capteur et le champ *Etat\_Rasp* fait partie de la table SuperVision.





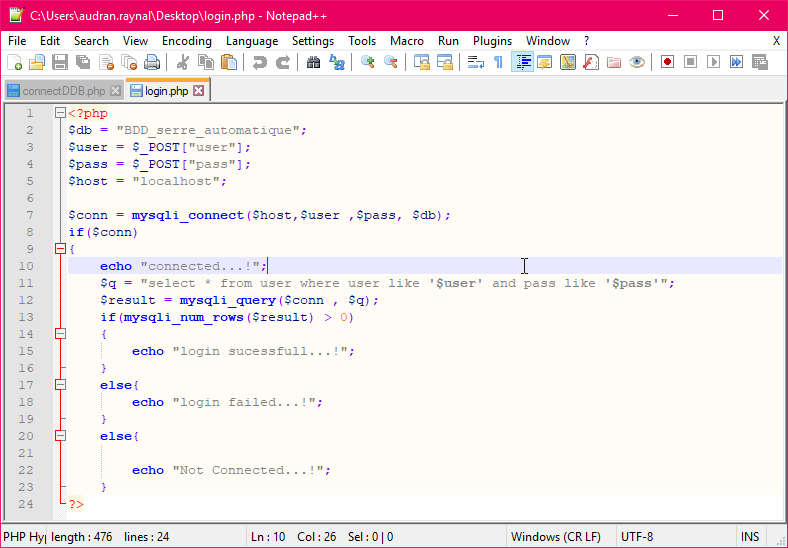
Connecter une application Java directement à une base de données est impossible. C’est pourquoi je suis obligé de passer par un middleware, ici ce sera un service web, une page PHP que je devrai écrire.

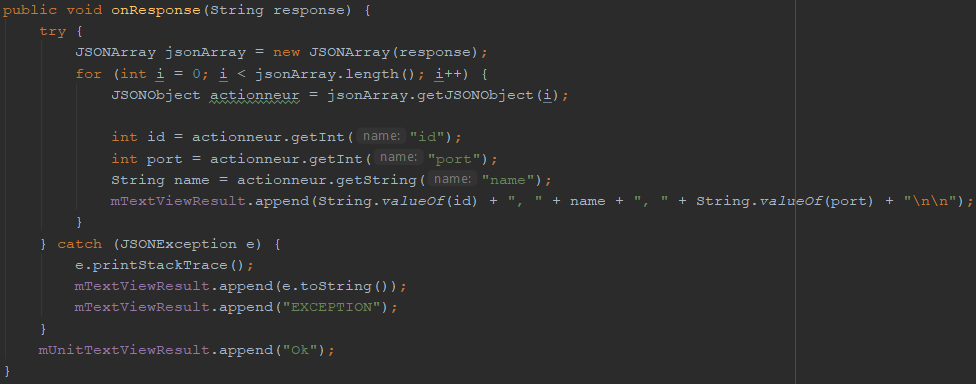
Source : http://tutorielandroid.francoiscolin.fr/index.php

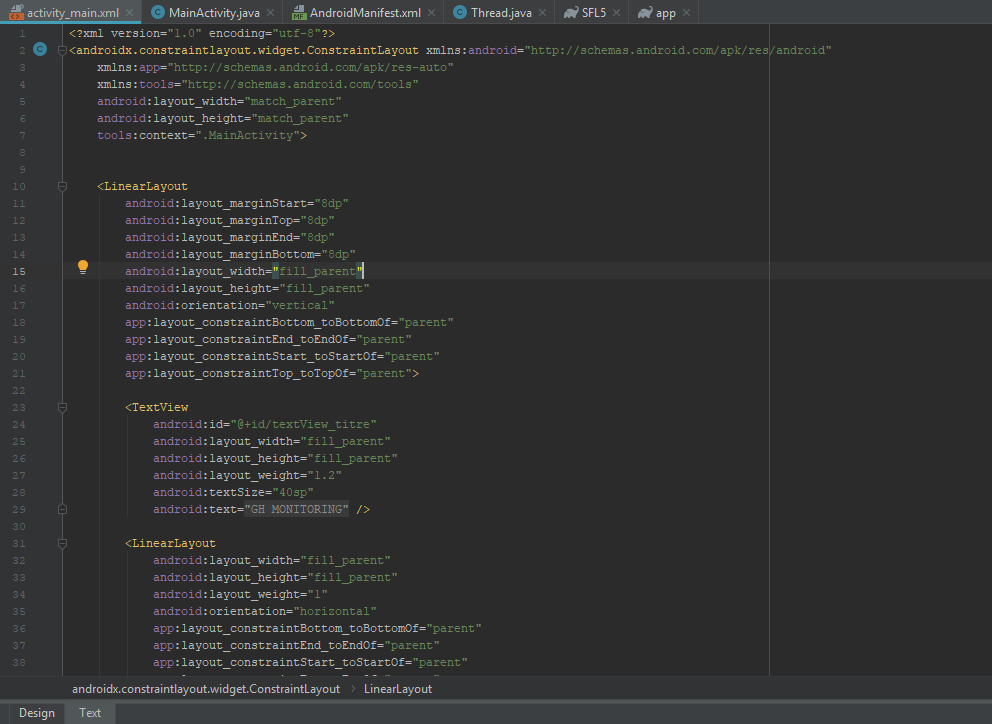
L’application web en PHP fera office d’intermédiaire entre l’application Android et la base de données. Elle contiendra les requête SQL qui permettra de se connecter à la base de données pour pouvoir récupérer les valeurs : les états du système. Ensuite, la page PHP enverra les données en format JSON *(JavaScript Object Notation)* à l’application Android qui pourra les interpréter.

## C - Code Android Studio

Ci-dessous est un extrait de mon fichier PHP. Il sert à se connecter en tant qu’utilisateur à la base de données. On peut voir plusieurs echo permettant de savoir si on est connecté et identifié. Le nom de la base de données est « BDD\_serre\_automatique ».



Ci-dessous est un extrait du code sous Android Studio, il sert à traduire les données JSON envoyé par la page PHP.



Cet extrait du code en XML, produit l’affichage de l’application. Il y a trois balises LinearLayout horizontal pour 3 lignes avec 2 capteurs chacune. Pour chaque capteur il y a une balise TextView (pour le nom du capteur) et une balise ImageView (pour la pastille de couleur).

Ces 3 lignes sont contenues dans un LinearLayout vertical, qui contient lui-même un TextView pour le nom de l’application.

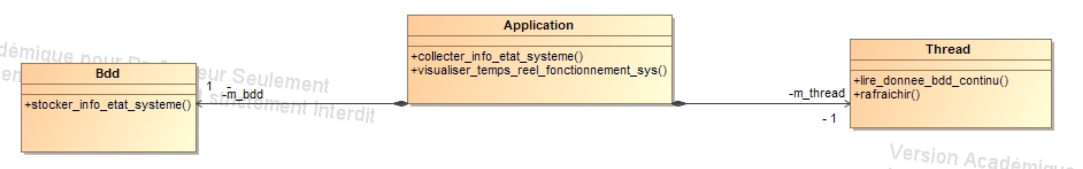
De plus, la méthode de développement des poids a été utilisé pour que l’application s’adapte aux différentes tailles et définitions des écrans de smartphone.

 Le poids d’un TextView est égal à 1.2

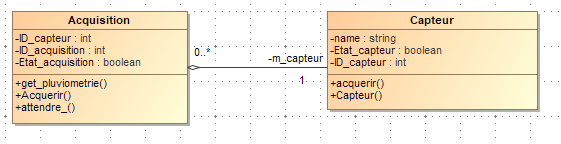
La méthode utilisé pour faire changer les pastilles de couleurs est setBackground. Pour une condition, la couleur de fond (background) de la pastille changera. La condition ici est l’état du composant, récupéré en JSON, qui est un boolean.

# IV - Diagramme de classes

Ces diagrammes ont été réalisés en début de projet, en même temps que l’analyse globale.

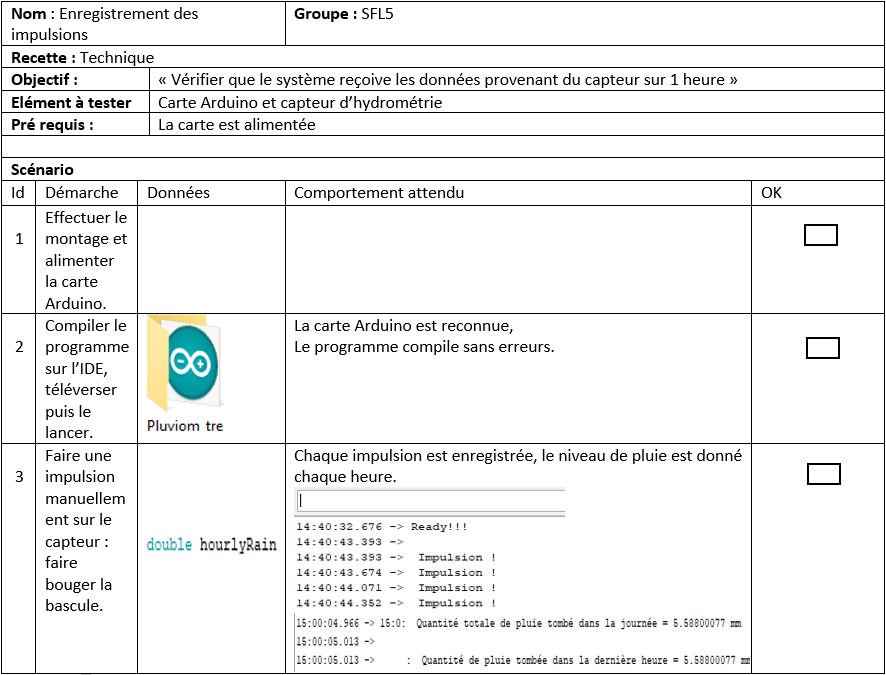


Voici le diagramme de classe de l’application Android. Elle utilisera un thread pour que les valeurs des états des capteurs soit mis à jour constamment. On peut retrouver les objectif de l’application dans les méthodes de sa classe : collecter et afficher des valeurs de la base de données. La classe Bdd à une méthode pour stocker les valeurs voulues.



Le diagramme ci-dessus présente l’acquisition des mesures du capteur. La classe capteur possède un attribut Etat\_capteur en boolean, car il ne peut que marcher, ou ne pas marcher.

# V - Test Unitaire



# VI - Fiche recette

# CONCLUSION

## Communication avec le groupe

La réussite d’un projet en groupe requiert une bonne cohésion d’équipe mais aussi d’une bonne communication entre tous les membres du groupe. Nous avions plusieurs moyens de communiquer et de partager :

* Nous avions GitHub, outil de gestion de version, extrêmement utile pour un projet de groupe.
* Nous avions un groupe de discussion Messenger pour pouvoir discuter à distance, informer les autres membres d’éventuels retards ou absences.
* Le diagramme de Gantt pour se tenir informé de l’avancement du projet.

Il faut préciser qu’étant donné que notre projet SFL5 et le projet SFL6 sont très proches, il y avait aussi un travail commun, et une communication était requise, notamment sur la création de la base de données qui est commune, mais aussi sur l’assemblage de la serre. Heureusement nos groupe respectifs étaient proche, et le travail s’est déroulé presque sans accrocs.

## Connaissances apportées

Grâce au travail sur mon capteur, j’ai pu découvrir l’univers Arduino, car je n’y avais jamais touché auparavant. Bien que je connaissais le langage utilisé dans l’IDE Arduino, car c’est du C/C++, j’ai découvert son écosystème, ainsi que les montages à effectuer dans lesquels j’avais peu d’expérience auparavant.

Par le passé j’avais développé des applications Android, mais pas en Java et par Android Studio. De plus je me suis familiarisé avec le principe du middleware et le format d’échange JSON.

## Taches restantes

A ce jour, certaine de mes tâches ne sont pas terminées.

En effet, mon application Android n’est pas finie, il manque encore beaucoup de ligne de codes essentielles au fonctionnement. Le thread n’est pas programmé. La partie PHP n’est pas tout à fait terminée.

## Difficultés rencontrés

J’ai rencontré plusieurs difficultés lors du projet, qui étaient souvent reliés au peu de connaissance que j’ai par rapport à Android Studio, et qui était dur à prendre en main. De plus j’ai beaucoup de difficulté car la programmation n’est pas mon point fort.

La page PHP était complexe à réaliser car j’ai mis du temps à comprendre comment fonctionnait le JSON et un middleware en général. J’ai essayé plusieurs méthode de connexion à la base de données avant de trouver celle qui m’a convenu.

Je pourrai aussi parler de l’état de notre réseau, qui nous ralentissaient considérablement, car nous n’avions plus accès à nos lecteurs réseaux, mais aussi à Internet. De plus, la vitesse de téléchargement était faible, ce qui était embêtant pour les nombreuses mise à jour d’Android Studio.

## Regard critique du projet

Il existe une critique d’ordre matérielle : les ordinateurs que nous avons ont une configuration modeste, suffisante pour la plupart des tâches. Mais pour l’utilisation d’Android Studio, personnellement je trouve que nos ordinateurs étaient un peu dépassés. En effet le temps de lancement de l’application, ainsi que les temps de chargement et de compilation, peuvent être assez long et atteindre plusieurs minutes, ce qui est frustrant à la longue.

Ce qui aurait pu être une solution à ce problème, c’est d’avoir le choix de langage et/ou d’IDE. Bien qu’Android Studio soit complet et gratuit, il n’en reste pas moins lourd et requiert beaucoup de ressources. De plus il est assez difficile à prendre en main pour la première fois. Il existe d’autre moyen de développer une application sous Android, par exemple Microsoft et son environnement .NET et le langage C#. Le langage Kotlin est depuis 2017 le second langage nativement pris en charge par Android, et il aurait été intéressant de le découvrir ou d’avoir le choix de l’utiliser.