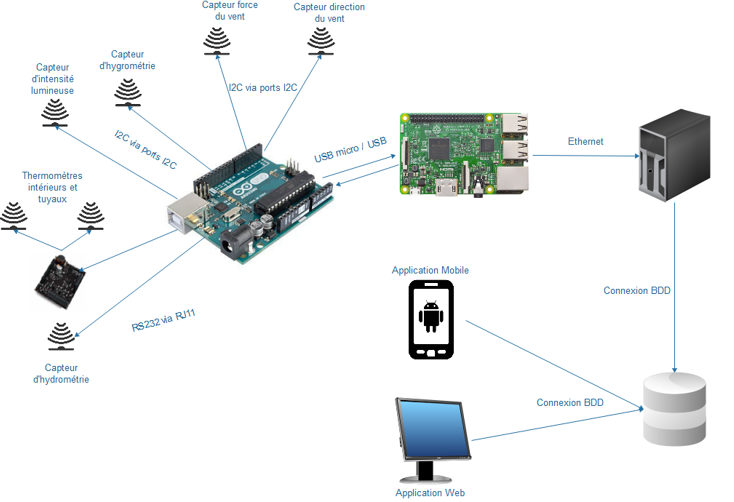
|  |
| --- |
| Groupe Olivier |
| Dossier technique du projet – Partie personnelle |
| Etudiant 4 : Visualisation de l’état du système et acquisition de mesure par capteur |

# RAYNAL Audran

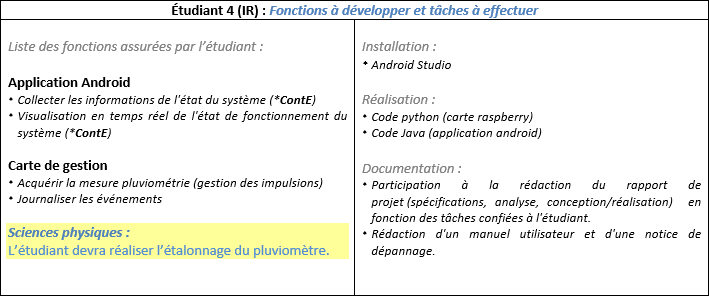
# Situation dans le projet

### A – Synoptique de la réalisation



*Voici les parties de la synoptique me concernant.*

### B-Rappel des tâches de l’étudiant



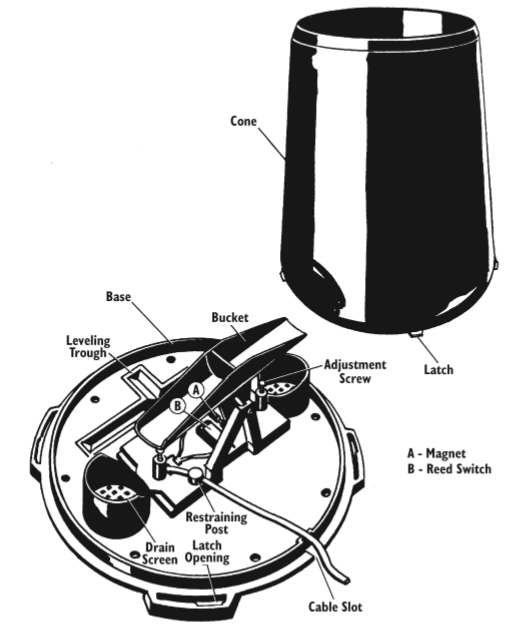
Dans le projet, ma principale tâche est le développement d’une application sur smartphone Android. L’application est développée avec le logiciel Android Studio, l’IDE officiel de développement sous Android. L’application aura comme objectif de permettre à l’utilisateur de visualiser en temps réel l’état 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.

On m’a confié le capteur d’Hydrométrie. Il s’agit du Pluviomètre Rain Collector II de la société Davis. Je dois acquérir la mesure pluviométrie grâce à ses impulsions. Je réaliserai son étalonnage, 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.

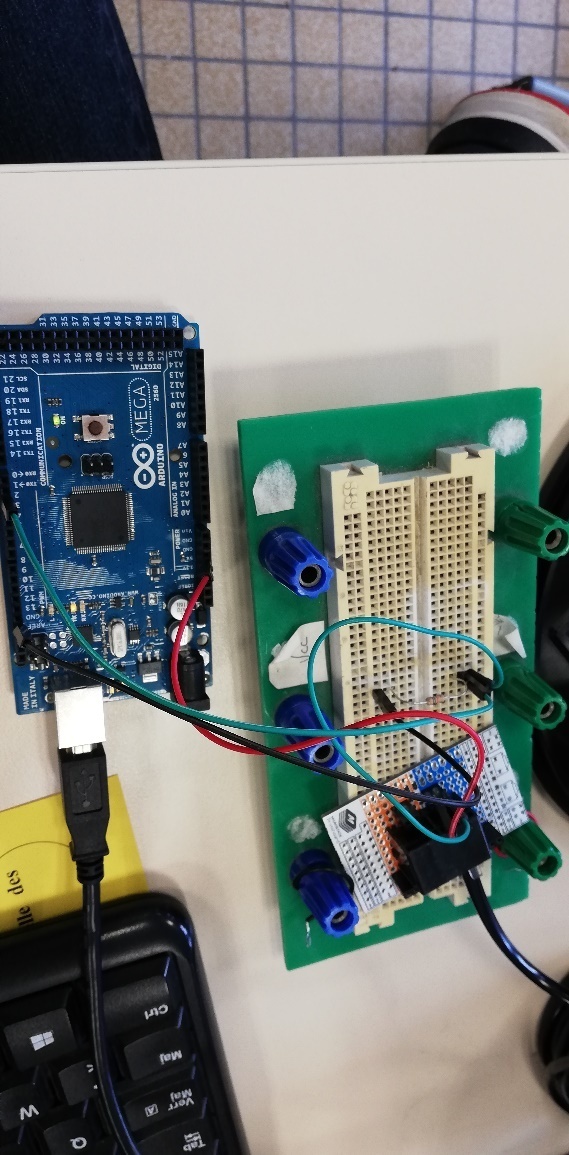
Les composants permettant la mesure d’Hydrométrie sont le switch magnétique et l’aimant situé sous la bascule. Ils sont représentés sur le schéma en position A et B. L’aimant permet sous certaines conditions de réagir avec le switch magnétique qui enregistrera une impulsion électrique qui sera transmise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

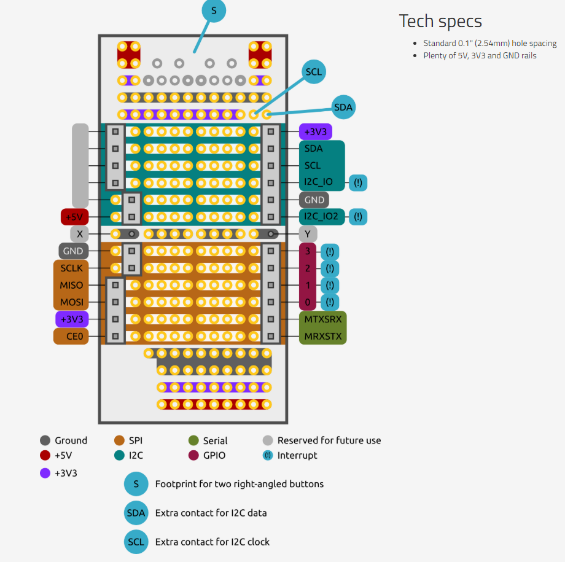
Le cône sert à collecter l’eau de pluie. Elle sera dirigée vers la bascule qui, avec suffisamment d’eau basculera en la laissant couler, et entraînera dans son mouvement l’aimant. En changeant de position, il permettra d’effectuer une impulsion sur le switch magnétique.

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

Source : https://www.instructables.com

Ce schéma est le modèle sur lequel je me suis basé pour réaliser mon montage. 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.

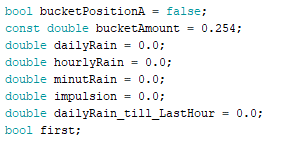
De plus, il est nécessaire d’ajouter au montage : une résistance entre 10K et 15K Ohms, celle utilisée ici est de 15K Ohm. Le montage étant simple, il n’y a pas besoin de Shield, il suffit de bien brancher les pins.



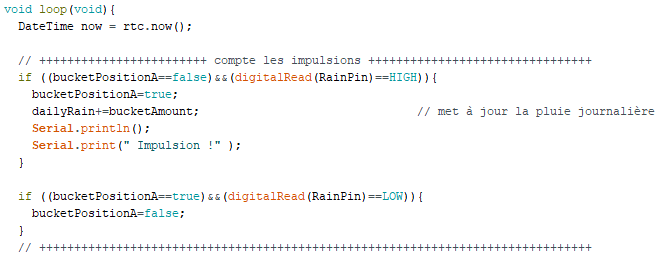
### C- L’acquisition des mesures

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.



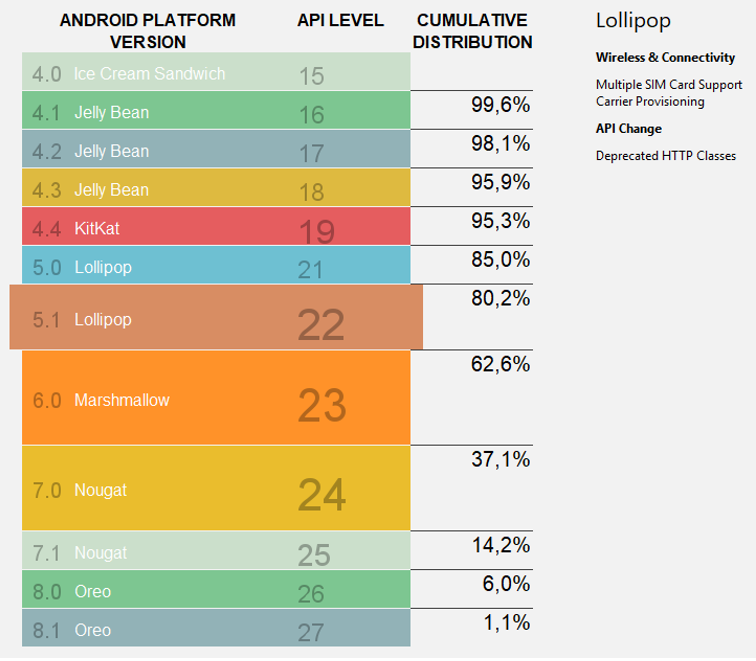
Sur cet extrait de mon programme, on peut voir le Boolean bucketPositionA. C’est lui qui représente la position de l’aimant quand il passe devant le switch. Si il prend la valeur true, c’est-à-dire qu’il vient de bouger, et donc qu’il pleut. On incrémente donc le double dailyRain, la pluie journalière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

# III - Réalisation de l’Application Android

Un API de développement *(Application Programming Interface)* est une interface de programmation constitué d’un ensemble de fonction.

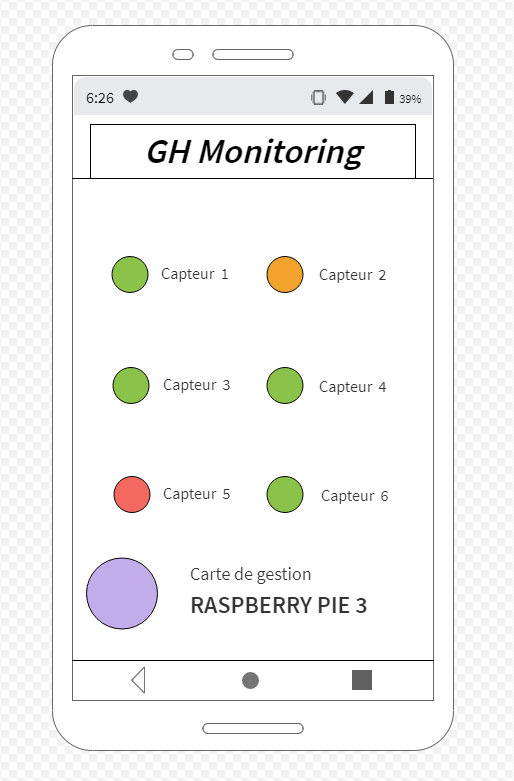
Le choix de l’API de programmation minimum pour mon application s’est porté sur 2 critères : le plus important est la distribution globale de l’API.

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 à 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 développement Android.

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.

J’ai donc 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.

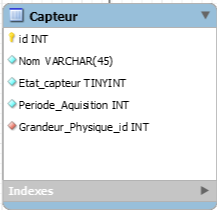


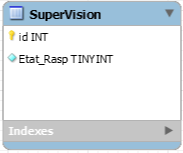
GH monitoring signifie Surveillance de Serre *(Greenhouse monitoring)*

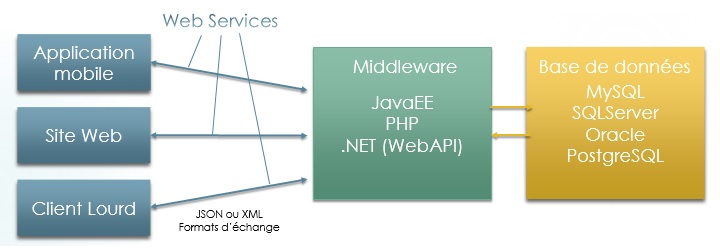
Le prototype d’IHM présenté ci-dessus a servi de base pour le développement de l’application.

## 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.



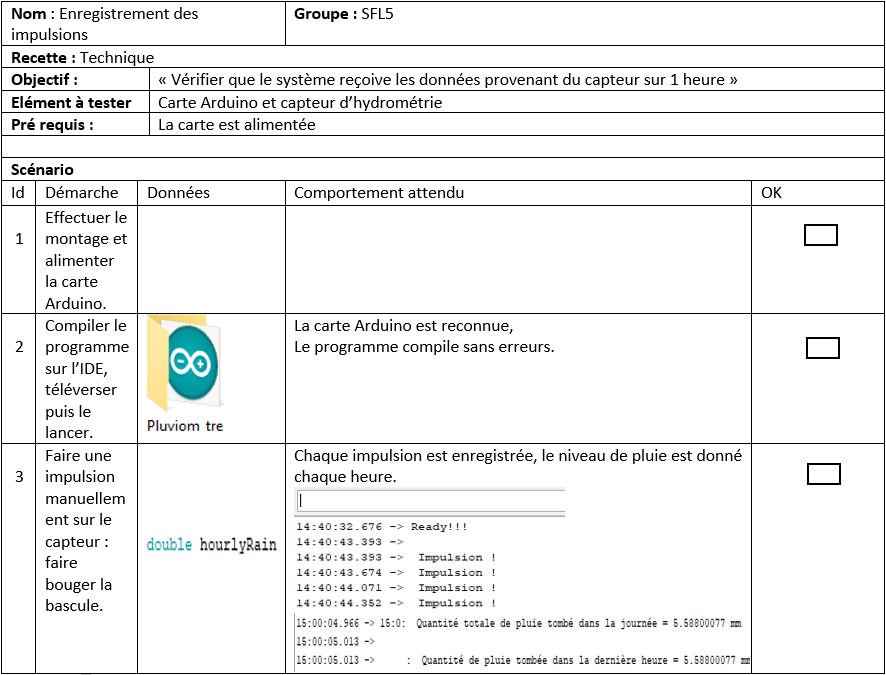


Connecter une application Java directement à une base de données est impossible. C’est pourquoi je suis obligé de passer par un middleware, ici ce sera un service web, une page PHP que je devrai écrire.

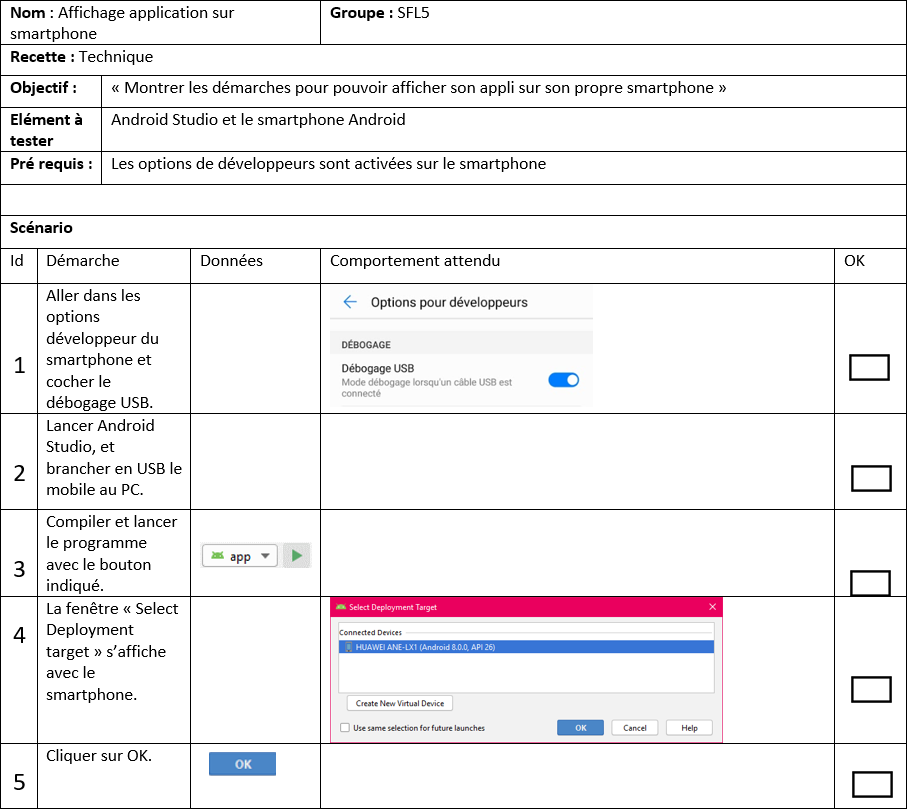
Source : http://tutorielandroid.francoiscolin.fr/index.php

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.

## Test Unitaire



## 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.

## 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.

## 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.