Rapport intermédiaire pour le projet long
Système d'arrosage intellectuel

Natalia Bragina, Nicolas Graff

28 février 2023

Présentation générale du projet

Le but de ce projet est la mise en place d'un système d'arrosage qui permet d'alimenter en eau les plantes selon leurs besoins. Ce projet trouve son utilité pour les personnes qui ne peuvent pas être systématiquement présentes pour l'arrosage.

Le système est basé sur un ordinateur miniature Raspberry Pi. Les données sur l'humidité du sol sont fournies par des capteurs d'humidité. Le ravitaillement et la coupure de l'eau sont assurés par des solénoïdes, auxquels l'eau est amenée à l'aide d'une pompe submersible. Raspberry Pi gère l'ouverture/fermeture des solénoïdes et le branchement/débranchement de la pompe.

En outre, Raspberry Pi possède une base de données et un serveur qui fournira à l'utilisateur une interface Web avec certaines fonctionnalités :

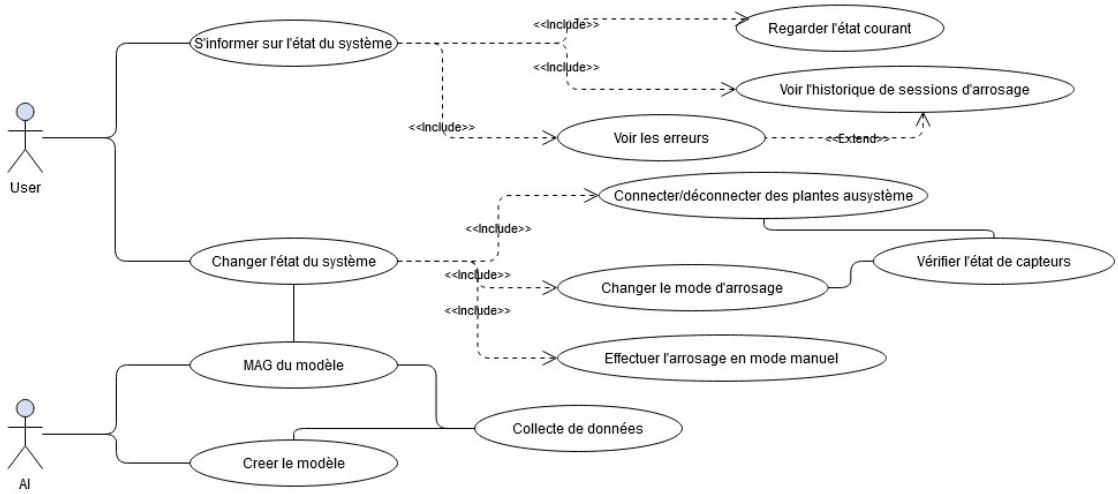
1. Information sur l'état actuel du système et possibilité de voir ses logs.
2. Connexion et déconnexion des plantes au système en cas de changement dans le jardin.
   Il faut informer le système de la nouvelle plante afin que le programme en tienne compte lors de l'arrosage. De la même manière, il faut informer le système si une plante n'est plus présente dans le jardin.
3. Possibilité de s'arranger en cas de dysfonctionnement du système ou d'une de ses parties.

Par défaut, le système fonctionne en mode automatique. Cependant, en cas d'erreurs ou si nécessaire, il sera possible d'effectuer certaines actions à distance en « mode manuel » : vérifier le fonctionnement des capteurs, effectuer l'arrosage des plantes, etc.

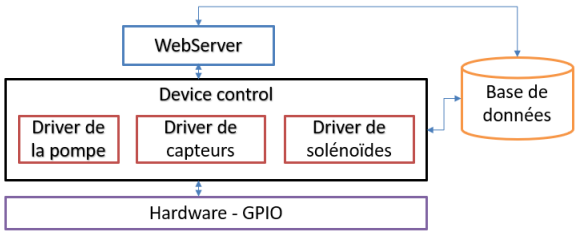
Pour assurer le bon fonctionnement du système même en cas de défaillance possible de certains capteurs, nous souhaitons mettre en place une IA qui permettra de détecter les erreurs et de procéder à l'arrosage malgré la défaillance éventuelle des capteurs.

Parties du projet réalisés

1. Analyse de besoins, élaboration de spécifications.
   Les résultats de notre travail sur l'analyse des besoins et l'élaboration du cahier des charges peuvent être synthétisés dans le diagramme de cas d'utilisation suivant :

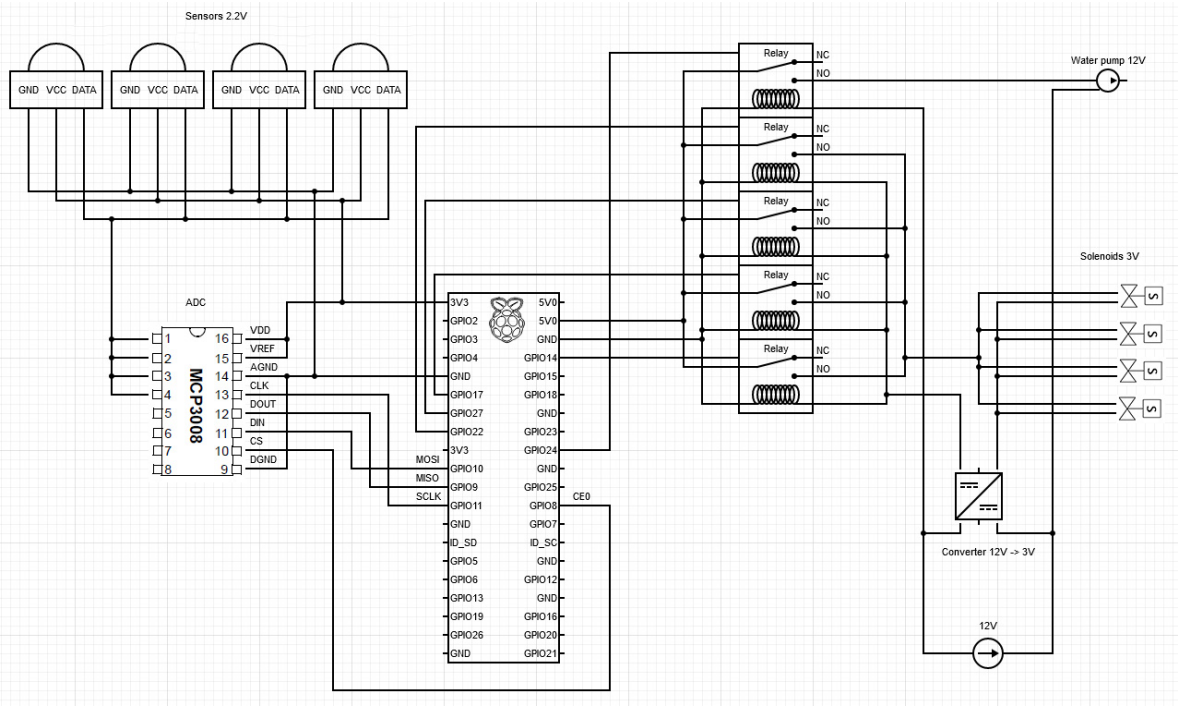


2. Conception de l'architecture du système (à l'exception de la partie liée à l'apprentissage).
   L'analyse des bibliothèques existantes pour communiquer avec l'ADC nous a montré qu'il existe des solutions prêtes à l'emploi qui fournissent les fonctions de drivers pour notre matériel, donc on n'a plus besoin de les implémenter. Nous avons choisi la librairie gpiozero de langage Python, comme étant la plus complète et adaptée à nos besoins. Ainsi, notre schéma initial des composants du système a été révisé :



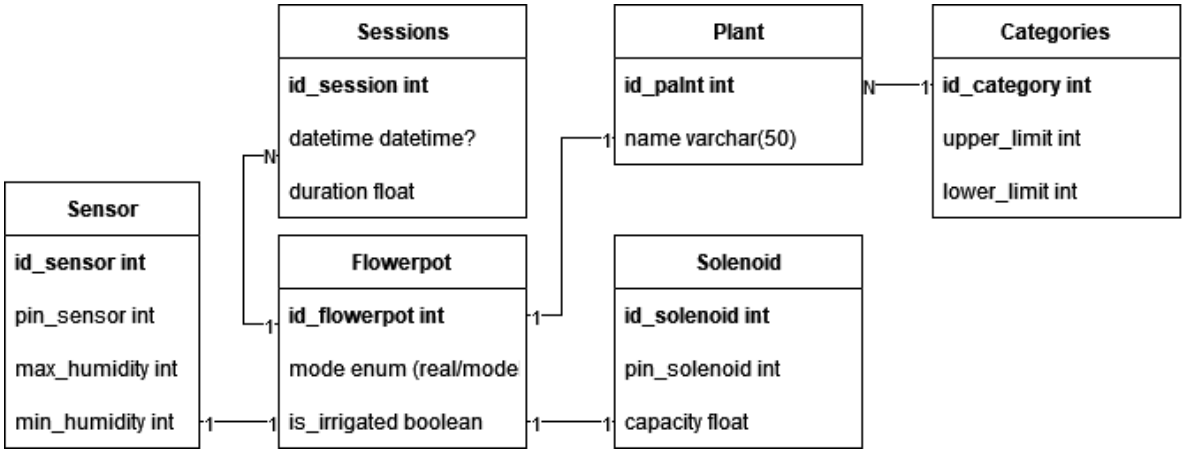
3. Étude de la documentation Raspberry Pi, choix d'ADC.
4. Étude des caractéristiques techniques de composants périphériques et leur sélection (relais, bloc d'alimentation supplémentaire, convertisseurs de tension, connecteurs, câbles).
5. Conception et calcul du circuit électrique.

6. Montage du circuit. Il a été monté pour quatre paires de solénoïde-capteur (l'ADC choisi permet d'ajouter encore trois paires si nécessaire). Schéma électrique du projet :



7. Conception et implémentation partielle de la base de données : schéma et tableaux.

   Nous avons choisi MariaDB comme système de gestion de base de données open-source ayant de bonnes performances. Nous avons créé un modèle de données relationnel basé sur les données à utiliser dans le système, effectué une normalisation et implémenté la base de données dans le système :



8. Conception de la maquette de GUI.

   Sur la base de la liste des actions que l'utilisateur peut effectuer, nous avons créé un modèle d'interface graphique du serveur qui comprend les pages suivantes :

   — page d'accès avec informations générales sur le système, liste des plantes et liste des derniers avertissements et erreurs ;

   — ajout et suppression de plantes ;

   — outil pour changer le mode de contrôle - passer en mode manuel ;

   — liste de tous les logs avec la possibilité de les filtrer.

9. Conception et implémentation partielle du serveur Web : affichage des logs, ajout et suppression de plantes.

On a choisi LAMP software bundle : Linux est le système installé par défaut sur le Raspberry Pi, Apache en tant que serveur, MariaDB pour gérer la base de données et le langage de script PHP pour générer des pages web dynamiques et communiquer avec le serveur MariaDB.

10. Les tests ont été effectués pour :
    — s'assurer du bon fonctionnement des périphériques ;
    — effectuer la normalisation des solénoïdes et des capteurs.

Problèmes rencontrées

Le fait le **plus influant** sur déroulement du projet est notre manque d'expérience avec le matériel. Pour cette raison, nous n'avons pas été en mesure d'évaluer correctement certaines étapes du projet, ce qui a évoqué les modifications des objectifs et du calendrier. Par exemple, le développement du serveur et de la base de données a été influencé par le fait qu'il y a des écarts des valeurs bruts arrivant à partir des capteurs selon la session et que les solénoïdes ont des débits variés. Pour s'arranger, on est obligé à fournir au utilisateur les moyens à mesurer le débit d'un solénoïde et de l'insérer dans la base de données à l'aide du site Web.

Les défis plus spécifiques auxquels nous avons été confrontés étaient :

1. **Sélection du matériel.** Au départ, on n'a eu que la vision générale du fonctionnement du système. Cependant, on ne savait pas quels matériels, en plus de Raspberry Pi, des capteurs et d'une pompe, devaient y être inclus. L'un des premiers défis a été le fait que les capteurs et la pompe sont les appareils électriques sans aucun protocole de réception des signaux à partir de Raspberry Pi. La recherche de solution nous a conduit à l'utilisation des relais qui, à son tour, sont à contrôlés par Raspberry Pi.

   Transformation de signaux analogues (issus des capteurs) aux signaux numériques (pouvant être traité par Raspberry Pi) n'a posé autant des problèmes grâce le cours de réseaux en M1.

2. **Élaboration d'un circuit électrique.** Le calcul de la consommation électrique requise a révélé qu'il était impossible d'alimenter tous les matériels à partir de la Raspberry Pi, car celle-ci ne fournit que 5 volts et 3 ampères, ce qui est insuffisant.

   La solution évidente a été d'ajouter une source d'énergie supplémentaire. Cependant, sa puissance requise ne pouvait être estimée qu'à partir d'élaboration du circuit dans son intégrité. Plusieurs itérations ont donc été nécessaires pour définir la puissance du bloc d'alimentation supplémentaire, ainsi que celle des convertisseurs.

   Il y avait aussi quelques difficultés avec la conception du circuit électrique lui-même. Les questions particulières comme « quels composants il faut connecter en parallèle et quels séquentiellement ? », « comment ça va influencer la tension et l'intensité du courant fournis ? » etc.

3. **Assemblage de périphériques.** Certains composants ne sont pas physiquement adaptés les uns aux autres : malgré la présence de large gamme du matériel, on ne peut pas toujours choisir ce qui est parfaitement compatible. Il a fallu un peu de bricolage (utilisation des convertisseurs et d'adaptateurs, pose de joints etc).

4. **Normalisation et utilisation de données des capteurs pour apprentissage de IA.** Les tests de solénoïdes ont révélé les écarts des valeurs de débit : 4,6 ml/s, 3,7 ml/s, 3,9 ml/s et 4,2 ml/s. Les tests de capteurs ont montré qu'un capteur dans le même état d'humidité affiche des valeurs variées selon une série de mesures. Par exemple, le capteur A à l'état sec, lors de sa première connexion affiche toutes les 5 secondes les valeurs suivantes : 791, 790, 790, 791 ; à la deuxième connexion : 789, 789, 789, 789

   Après rebut de Raspberry Pi à la première connexion : 990, 964, 959, 964 ; à la deuxième connexion : 948, 953, 950, 968

   Cela ajoute à la complexité d'utilisation de ces données dans l'apprentissage.

5. **La pression d'eau dans tubes varie en fonction du jeu des solénoïdes ouverts en moment donné.** Par conséquent, il est nécessaire de faire varier la durée d'arrosage de chaque pot, non seulement en tenant compte des données du capteur et des besoins des plantes, mais aussi en tenant compte de jeu des solénoïdes ouverts en un moment.

6. **Choix du serveur web.** Il existe de nombreux serveurs et le choix a été fait entre Apache et Nginx. Malgré les avantages du second en termes d'occupation de la mémoire et de consommation de ressources, on a choisi apache parce qu'on a déjà de l'expérience avec ce serveur – cela nous a permis de démarrer rapidement le projet.
7. **Sécurité.** On est obligé de limiter l'accès au serveur en raison de sécurité. Notre système ne fournit pas la gestion de comptes, car tous les utilisateurs du système auront la même interface et les mêmes fonctionnalités. Par conséquent, nous utilisons la configuration du contrôle d'accès avec l'outil interne d'Apache.

   Un autre problème est que pour se connecter à la base de données à partir d'un fichier php, il faut spécifier un identifiant et un mot de passe. Actuellement, ces données sont écrites directement dans le code php.

Parties du projet à réaliser

1. **Serveur web.** Terminer le serveur, ajouter des fonctions manquantes – mise en mode manuel, arrosage manuel des plantes, vérification des capteurs.

   Il est également nécessaire de changer le serveur Apache contre Nginx.
2. **Base de données.** Il faut ajouter des transactions lors de l'ajout et de la suppression d'une plante, vérifier si d'autres transactions et triggers sont nécessaires.
3. **Apprentissage.** Choisir un modèle, collecter des données et inclure dans les fonctionnalités existantes.
4. Résoudre le **problème de normalisation**. Étant donné que les tests de capteurs ont montré l'écart important des valeurs selon la session, il faudra élaborer l'algorithme de normalisation pour qu'on soit capable à utiliser les données arrivant de capteurs en apprentissage de IA
5. Implémenter les fonctions d'**interaction de Device Control et la base de données**, ainsi qu'assurer l'enregistrement et le **stockage des logs**.
6. En Device Control implémenter les programmes de manipulation en **mode manuel** pour assurer la fonctionnalité fournie par l'interface Web.