POLYTECH NICE-SOPHIA

Wissal Bellahcen

Rapport de projet : Programmateur d'arrosage
Séance 1+2

Introduction

Ce projet vise à réaliser un système d'arrosage automatique pour quatre circuits indépendants, piloté par un **ESP32 WIFI**. Le système utilise un écran LED et des boutons pour l'interface utilisateur, ainsi que des capteurs de température, hygrométrie et pluviométrie pour adapter l'irrigation aux conditions réelles.

Chaque vanne peut être activée individuellement et fonctionne successivement pour maintenir un débit optimal, avec une durée d'arrosage réglable par circuit. La connectivité WIFI permettra à terme le contrôle à distance et le transfert des données vers un serveur.

Première séance : Choix des composants et rôle dans le projet

Lors de la première séance, notre groupe a travaillé sur l'élaboration du cahier des charges, définissant les besoins fonctionnels et techniques du système. Ma contribution s'est concentrée sur le choix et la description des composants électroniques principaux :

ESP32 WROOM 32D

L'ESP32 WROOM 32D a été sélectionné comme microcontrôleur central du système. Ce module combine Wi-Fi et Bluetooth et dispose de suffisamment de puissance et de mémoire pour gérer simultanément les électrovannes, les capteurs et l'affichage sur l'écran LED. Dans notre projet, l'ESP32 est responsable de :

- La lecture des capteurs (humidité, température, pluviométrie) pour adapter l'arrosage aux conditions réelles.
- La commande des quatre électrovannes, en assurant que chaque circuit fonctionne de manière autonome mais coordonnée.
- La gestion de la consommation d'énergie, grâce à son processeur ULP qui permet de maintenir certains capteurs actifs en veille.

Cette approche permet de centraliser toutes les fonctions du système sur un seul module puissant et flexible.

Module horloge RTC DS3231

Pour la gestion du temps, nous avons choisi le module RTC DS3231, capable de conserver l'heure même lorsque le système est hors tension grâce à sa pile intégrée. Dans notre projet, la RTC :

- Fournit une référence horaire précise pour planifier les cycles d'arrosage.
- Permet l'affichage de l'heure et de la date sur l'écran LED.
- Facilite l'adaptation de l'arrosage aux besoins spécifiques des différentes zones grâce à la programmation horaire.

En résumé, ces deux composants sont au cœur du fonctionnement du système : l'ESP32 assure le contrôle intelligent et la communication, tandis que la RTC garantit une gestion précise du temps et des cycles d'irrigation.

Deuxième séance : Programmation des électrovannes

Lors de la deuxième séance, j'ai développé le code Arduino permettant le contrôle de la fermeture et l'ouverture des quatre électrovannes via l'ESP32.

3.1 Objectifs du code

- Initialiser les sorties correspondant aux électrovannes
- Définir les fonctions pour ouvrir et fermer chaque vanne
- Fermer toutes les vannes au démarrage pour sécurité
- Tester le fonctionnement via le moniteur série

3.2 Fonctionnalités principales

- Ouverture et fermeture d'une vanne spécifique via les fonctions `ouvrirVanne(n)` et `fermerVanne(n)`
- Fermeture de toutes les vannes avec `fermerLesVannes()`
- Boucle principale (`loop`) testant l'ouverture et la fermeture de la vanne 1 avec un intervalle de 2 secondes

```
#define Vanne1 23
#define Vanne2 22
#define Vanne3 21
#define Vanne4 19

#define ETAT_ACTIF HIGH
#define ETAT_INACTIF LOW

void ouvrirVanne(int n);
void fermerVanne(int n);

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

  pinMode(Vanne1, OUTPUT);
  pinMode(Vanne2, OUTPUT);
  pinMode(Vanne3, OUTPUT);
  pinMode(Vanne4, OUTPUT);

  fermerLesVannes();


  Serial.println("Système d'arrosage prêt");
}
```

```
void ouvrirVanne(int n) {
  switch (n) {
    case 1: digitalWrite(Vanne1, ETAT_ACTIF); break;
    case 2: digitalWrite(Vanne2, ETAT_ACTIF); break;
    case 3: digitalWrite(Vanne3, ETAT_ACTIF); break;
    case 4: digitalWrite(Vanne4, ETAT_ACTIF); break;
    default: return;
  }
  Serial.print("Vanne");
  Serial.print(n);
  Serial.println("ouverte");
}

void fermerVanne(int n) {
  switch (n) {
    case 1: digitalWrite(Vanne1, ETAT_INACTIF); break;
    case 2: digitalWrite(Vanne2, ETAT_INACTIF); break;
    case 3: digitalWrite(Vanne3, ETAT_INACTIF); break;
    case 4: digitalWrite(Vanne4, ETAT_INACTIF); break;
    default: return;
  }
  Serial.print("Vanne ");
  Serial.print(n);
  Serial.println(" fermée");
}
```

```
50    Serial.print(n);
51    Serial.println(" fermée");
52  }
53
54  void fermerLesVannes() {
55    digitalWrite(Vanne1, ETAT_INACTIF);
56    digitalWrite(Vanne2, ETAT_INACTIF);
57    digitalWrite(Vanne3, ETAT_INACTIF);
58    digitalWrite(Vanne4, ETAT_INACTIF);
59    Serial.println("Toutes les vannes sont fermées");
60  }
61
62  void loop() {
63    ouvrirVanne(1);
64    delay(2000);
65    fermerVanne(1);
66    delay(2000);
67
68  }
69
```

Avantages : Ce code constitue la base de la gestion des électrovannes, permettant d'ajouter facilement la lecture des capteurs et la planification automatique des cycles d'arrosage.

Conclusion

Ces deux premières séances ont permis de définir les composants clés du système et de mettre en place le contrôle logiciel de base des électrovannes. Mon travail a porté sur la documentation technique de l'ESP32 et du module RTC, ainsi que sur le développement du code de pilotage des vannes. Ces bases serviront pour les prochaines étapes : intégration des capteurs et mise en place d'un planning d'arrosage automatisé.