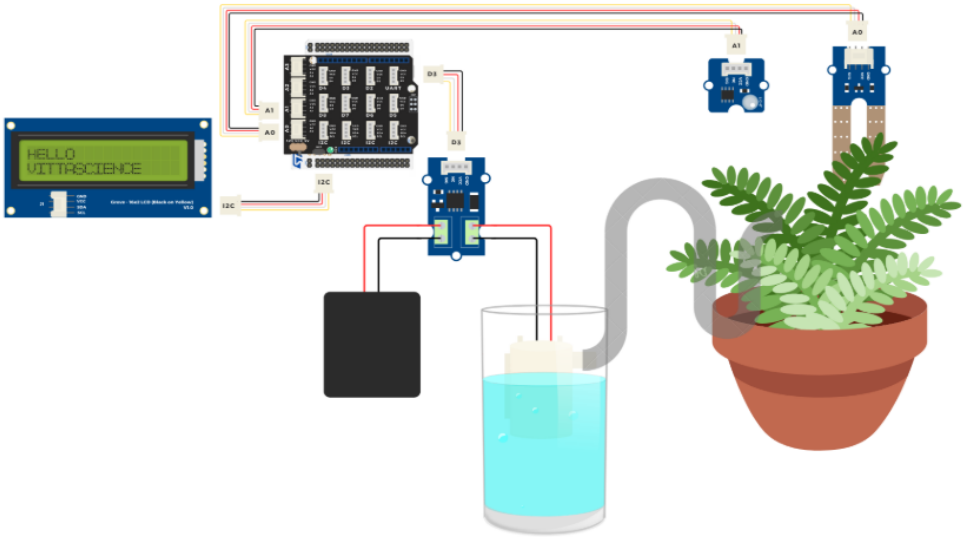
UNIVERSITÉ TOULOUSE III PAUL SABATIER

Université de Toulouse

Bureau d'études



Mini Projet - Plante connectée

Encadrant:
Thierry Perisse

Auteurs:
Skhiri Mohamed Wassim - Taibi Reda

Contents

Table de matière

1 Introduction

Dans le cadre de notre programme, nous avons eu l'opportunité de travailler sur un projet innovant : la conception et la réalisation d'une plante connectée. Cette initiative s'inscrit dans une démarche de découverte des cartes programmables et des différents composants électroniques. À travers trois ateliers distincts, nous avons exploré les fonctionnalités de ce kit "plante connectée" pour finalement aboutir à la création d'un système autonome capable de surveiller et de prendre soin des plantes de manière intelligente.

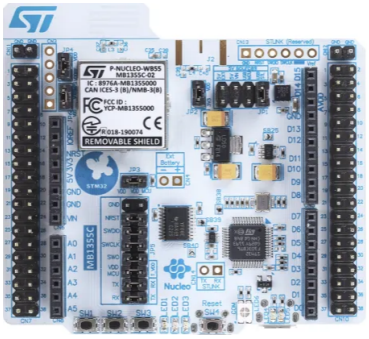
2 Objectif

L'objectif principal de ce projet était de concevoir un système capable de surveiller les besoins en eau et en lumière d'une plante, et d'agir en conséquence pour assurer son bien-être. Pour ce faire, nous avons divisé notre projet en plusieurs activités, chacune visant à explorer et à maîtriser un aspect spécifique du système global .

3 Materiels Utilisés

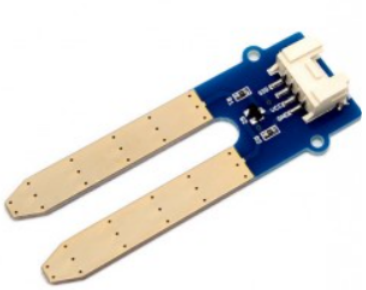
Carte ST NUCLEO-WB55RG

Cette carte est le cœur du système de la plante connectée. Elle intègre un microcontrôleur STM32WB55RG de STMicroelectronics, offrant une puissance de calcul suffisante pour gérer les différentes fonctionnalités du projet. La carte NUCLEO-WB55RG dispose de nombreuses broches d'entrée/sortie (GPIO), de ports de communication série et d'autres interfaces qui permettent de connecter et de contrôler les différents composants du système.



Capteur d'humidité du sol

Ce capteur détecte le niveau d'humidité dans le sol autour de la plante. Il fournit des données précises sur les besoins en eau de la plante, permettant au système de déclencher l'arrosage au moment opportun pour maintenir un niveau d'humidité optimal.



Capteur de luminosité

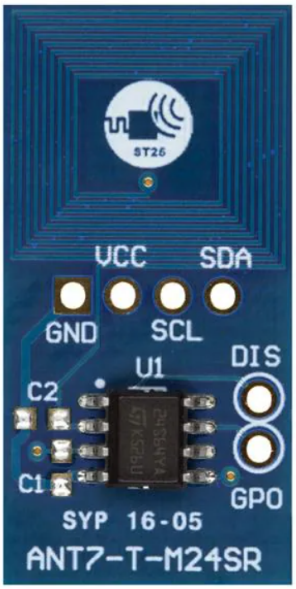
Ce capteur est essentiel pour mesurer la quantité de lumière reçue par la plante. Il convertit la luminosité ambiante en une valeur numérique que le microcontrôleur peut utiliser pour prendre des décisions en temps réel sur l'éclairage de la plante.

Afficheur LCD

L'afficheur LCD Grove 16x2 Black on Yellow est un type spécifique d'afficheur LCD conçu pour être compatible avec les modules Grove. Il dispose de 16 caractères sur 2 lignes et affiche du texte en noir sur un fond jaune, offrant une bonne lisibilité dans différentes conditions d'éclairage.

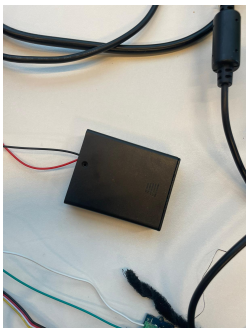
Module NFC ST M24SR64

Ce module permet d'établir une communication sans fil entre la plante connectée et d'autres appareils compatibles NFC. Il peut être utilisé pour partager des données ou configurer le système à distance, offrant ainsi une flexibilité accrue dans la gestion de la plante.



Bloc d'alimentation 6V

Ce bloc d'alimentation fournit une source stable et fiable de tension électrique au système. Il garantit que tous les composants fonctionnent correctement en recevant l'énergie nécessaire pour leurs opérations.



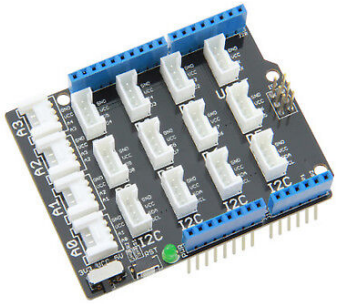
Câble USB

Le câble USB est utilisé pour la connexion entre la carte et l'ordinateur hôte. Il permet le transfert de données et parfois l'alimentation électrique de la carte.

Pompe à eau

La pompe à eau est responsable de l'arrosage automatique de la plante. Elle est activée en fonction des données recueillies par le capteur d'humidité du sol, assurant ainsi que la plante reçoive la quantité d'eau nécessaire à sa croissance.

Shield Grove Arduino



Ce shield offre une plateforme de prototypage rapide pour connecter facilement différents capteurs et modules à la carte Arduino. Il simplifie le câblage et permet une intégration rapide des composants dans le système global.

Module MOSFET

Ce module permet de contrôler la puissance fournie à la pompe à eau. Il agit comme un interrupteur électronique, activant ou désactivant le flux d'électricité vers la pompe en fonction des signaux du microcontrôleur.

4 Introduction à la carte NUCLEO et à STM32CubeIDE :

La carte NUCLEO est une plateforme de développement puissante basée sur les microcontrôleurs STM32 de STMicroelectronics. Elle offre un environnement idéal pour expérimenter, prototyper et développer des projets embarqués. La carte NUCLEO intègre un microcontrôleur STM32 ainsi que divers périphériques et interfaces qui facilitent le développement de solutions électroniques.

Pour programmer la carte NUCLEO et développer des applications embarquées, nous utiliserons STM32CubeIDE. STM32CubeIDE est un environnement de développement intégré (IDE) gratuit et complet développé par STMicroelectronics. Il est spécifiquement conçu pour simplifier le processus de développement sur les microcontrôleurs STM32 en offrant une suite d'outils puissants, notamment un éditeur de code, un compilateur, un débogueur et des fonctionnalités avancées de génération de code.

En combinant la carte NUCLEO avec STM32CubeIDE, nous bénéficierons d'un environnement de développement robuste et convivial pour concevoir et programmer notre projet de plante connectée.

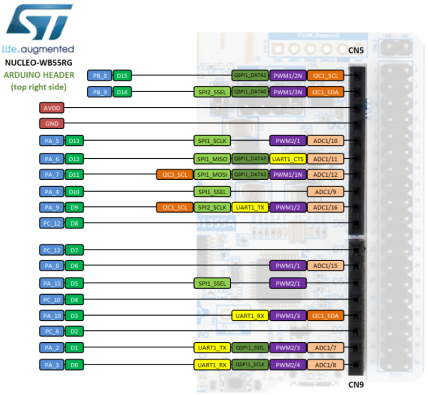


Figure 1: Nucleo-WB55RG

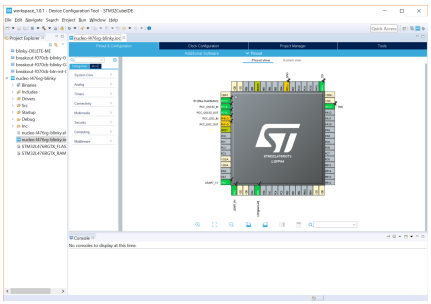


Figure 2: StmCubeIde

5 Activité A : Mesurer l'humidité du sol

5.1 Cahier de charge

Dans cette section, l'objectif est de mesurer l'humidité du sol à l'aide d'un capteur dédié et d'afficher les résultats sur un écran LCD.

5.2 Schéma de câblage

Pour cette partie, nous avons utilisé le logiciel Fritzing pour créer un schéma de câblage détaillé. Nous avons connecté le capteur d'humidité du sol au microcontrôleur STM32 Nucleo WB55RG en utilisant les broches appropriées. De plus, nous avons connecté l'écran LCD au microcontrôleur en suivant les spécifications du datasheet.

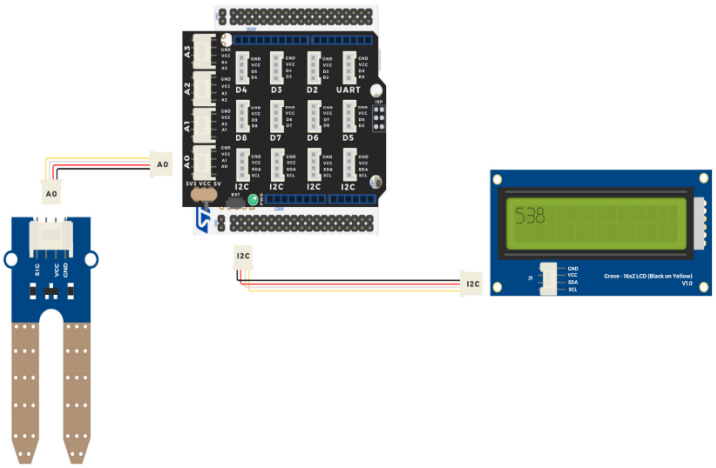


Figure 3: Schéma de câblage N°1

5.3 Programmation

Dans cette phase de programmation, nous avons commencé par configurer les broches du microcontrôleur STM32 Nucleo WB55RG pour permettre la lecture des données provenant du capteur d'humidité du sol. Cela implique de sélectionner les broches analogiques appropriées qui sont connectées à l'ADC intégré du microcontrôleur. Ensuite, nous avons programmé le microcontrôleur pour effectuer une conversion analogique-numérique (ADC) afin de convertir les signaux analogiques en données numériques que le microcontrôleur peut traiter. Pour cela, nous avons configuré les registres appropriés du microcontrôleur pour définir la broche analogique à lire, spécifier la résolution de la conversion et démarrer le processus de conversion.

Une fois les données analogiques converties en données numériques, nous avons mis en place le code nécessaire pour récupérer ces données émises par le capteur d'humidité du sol. Cela implique généralement de lire les valeurs converties à partir du registre de l'ADC et de les stocker dans une variable pour un traitement ultérieur.

Ensuite, nous avons développé un algorithme de conversion des données brutes en pourcentage d'humidité du sol. Cet algorithme utilise les données numériques captées du capteur d'humidité pour calculer le niveau d'humidité du sol en pourcentage. Cela peut impliquer l'application d'une formule de conversion spécifique basée sur les caractéristiques du capteur et les spécifications du fabricant.

Enfin, pour assurer la lisibilité des informations affichées, nous avons transmis les résultats de l'humidité du sol à l'écran LCD pour affichage(protocole I2C). Cela implique généralement l'utilisation des bibliothèques appropriées pour contrôler l'affichage sur l'écran LCD et l'envoi des données à afficher.

En combinant ces étapes de programmation, nous avons pu créer un système fonctionnel capable de mesurer l'humidité du sol de manière fiable et d'afficher les résultats de manière conviviale sur l'écran LCD, contribuant ainsi au bon fonctionnement du projet de plante connectée.

```
uint16_t Humidite; // D claration de la variable Humidite pour stocker
    la valeur de l'humidit  du sol

while (1) // Boucle infinie
{
    // Affichage de l'humidit  sur l' cran  LCD
    void affichage_humidite(uint16_t Humidite)
    {
        char txt[20] = {0};
        sprintf(txt, "Humi:%u", Humidite); // Formatage de la cha ne
            de caract res avec la valeur de l'humidit
        lcd_position(&hi2c1, 0, 0); // Positionnement du curseur sur la
```

```
            premi re ligne de l' cran  LCD
11          lcd_print(&hi2c1, txt); // Affichage de la cha ne de
               caract res sur l' cran  LCD
12          lcd_position(&hi2c1, 7, 0); // Positionnement du curseur sur la
               septi me colonne de la premi re ligne
13          lcd_print(&hi2c1, "%"); // Affichage du symbole '%' pour
               indiquer l'unit
14      }
15
16      MX_ADC1_Init(); // Initialisation du module ADC
17      HAL_ADC_Start(&hadc1); // D marrage de la conversion
           analogique-num rique
18      HAL_ADC_PollForConversion(&hadc1, 1); // Attente de la fin de la
           conversion
19      Humidite = HAL_ADC_GetValue(&hadc1); // R cup ration de la valeur
           num rique de l'humidit
20
21      // Conversion de la valeur num rique de l'humidit  en pourcentage
22      Humidite = (Humidite / 4095.0) * 100;
23  }
```

Ce code met en œuvre une boucle infinie pour mesurer en continu l'humidité du sol. Il utilise un convertisseur analogique-numérique pour obtenir la valeur de l'humidité, qu'il affiche ensuite sur un écran LCD. La fonction 'affichagehumidite' formate et affiche la valeur de l'humidité sur l'écran LCD avec le symbole '%' pour indiquer l'unité.

5.4 Résultat obtenu

Lors des tests, nous avons placé le capteur d'humidité du sol dans un verre rempli de terre pour simuler des conditions réelles. Ensuite, nous avons observé les valeurs d'humidité affichées en temps réel sur l'écran LCD. Nous avons vérifié la précision et la fiabilité des mesures en comparant les valeurs affichées avec les attentes en fonction de l'humidité réelle du sol.
Remarque : Il est essentiel de réaliser plusieurs tests dans différentes conditions pour valider la précision et la fiabilité du système de mesure d'humidité du sol.

6 Activité B : Mesurer la luminosité

6.1 Cahier de charge

Dans la suite, nous allons aborder la mesure de la luminosité.L'objectif de cette section est de mesurer la luminosité ambiante à l'aide d'un capteur de luminosité et d'afficher les

6.2 Schéma de câblage

Avant d'attaquer la programmation et la configuration, nous commençons par réaliser le schéma de câblage pour ce projet. Pour cela, nous avons utilisé Fritzing, et le schéma de câblage est le suivant :

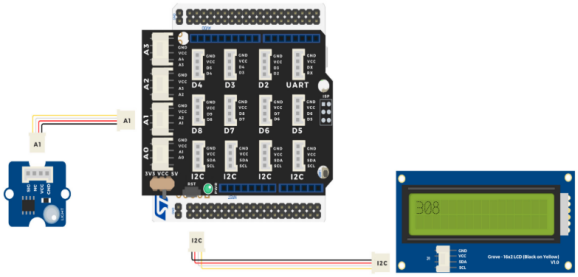


Figure 4: Schéma de câblage N°2

6.3 Programmation

Dans cette phase de programmation, nous avons débuté par la configuration des broches du microcontrôleur STM32 Nucleo WB55RG pour permettre la lecture des données provenant à la fois du capteur de luminosité et du capteur d'humidité du sol. Étant donné que ces deux capteurs utilisent le même protocole de communication, à savoir l'ADC (Convertisseur Analogique-Numérique), nous avons optimisé l'utilisation des ressources en utilisant un seul ADC pour les deux capteurs.

Ainsi, nous avons sélectionné les broches analogiques appropriées qui sont connectées à l'ADC intégré du microcontrôleur pour lire les données des capteurs. En configurant correctement ces broches, nous avons permis au microcontrôleur de commuter entre la lecture des données de luminosité et d'humidité du sol de manière efficace et cohérente.

En procédant de cette manière, nous avons pu simplifier le processus de programmation en évitant la configuration de plusieurs ADC et en optimisant l'utilisation des ressources matérielles disponibles sur la carte Nucleo. Cela a également permis une gestion plus efficace du temps de traitement et des interruptions, contribuant ainsi à améliorer les performances globales du système de mesure de luminosité et d'humidité du sol.

```
void affichage_lum(uint16_t Lum){

    char txt[20]={0}; // Initialise une cha ne de caract res pour
       stocker le texte    afficher
    sprintf(txt,"Lum:%d",Lum); // Formatte la valeur de luminosit
       dans la cha ne de caract res
    lcd_position(&hi2c1,0,1); // Positionne le curseur de l' cran  LCD
          la premi re colonne de la deuxi me ligne
    lcd_print(&hi2c1,txt); // Affiche le texte format  sur l' cran  LCD
    lcd_position(&hi2c1,7,1); // Positionne le curseur de l' cran  LCD
          la huiti me colonne de la deuxi me ligne
    lcd_print(&hi2c1,"%"); // Affiche le symbole de pourcentage sur
       l' cran  LCD
}


 while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    uint16_t Lum; // D claration de la variable Lum pour stocker la
       valeur de luminosit
MX_ADC1_Init2(); // Initialise l'ADC1 pour mesurer la luminosit
HAL_ADC_Start(&hadc1); // D marre la conversion ADC
HAL_ADC_PollForConversion(&hadc1, 1); // Attend la fin de la conversion
Lum=HAL_ADC_GetValue(&hadc1); // R cup re la valeur convertie de
   luminosit
```

```
Lum=(Lum/4059.0)*100; // Convertit la valeur de luminosit  en
    pourcentage
sConfig.Channel = ADC_CHANNEL_2;
  }
```

Ce code gère l'affichage de la luminosité sur un écran LCD. Tout d'abord, une fonction est définie pour afficher la luminosité sur l'écran LCD en pourcentage. Ensuite, une variable est déclarée pour stocker la valeur de luminosité mesurée. Enfin, l'ADC1 est initialisé pour mesurer la luminosité à l'aide du canal 2, puis la valeur convertie est récupérée et convertie en pourcentage.
La ligne de code "sConfig.Channel = ADC_CHANNEL_2;" configure le canal de l'ADC du microcontrôleur STM32 pour qu'il lise les données provenant de la source connectée au canal 2. Dans ce contexte, le canal 2 correspond à l'entrée analogique spécifique où le capteur de luminosité est connecté. En configurant le canal de cette manière, le microcontrôleur sait quelle source de données il doit lire lors de la conversion analogique-numérique.

6.4 Résultat Obtenu

Après avoir configuré et programmé le microcontrôleur pour lire les données des capteurs d'humidité du sol et de luminosité, nous avons réussi à afficher avec précision ces données sur l'écran LCD en temps réel. Les tests ont démontré que le système était capable de détecter et d'afficher efficacement les variations d'humidité et de luminosité, validant ainsi son fonctionnement pour le projet de plante connectée.

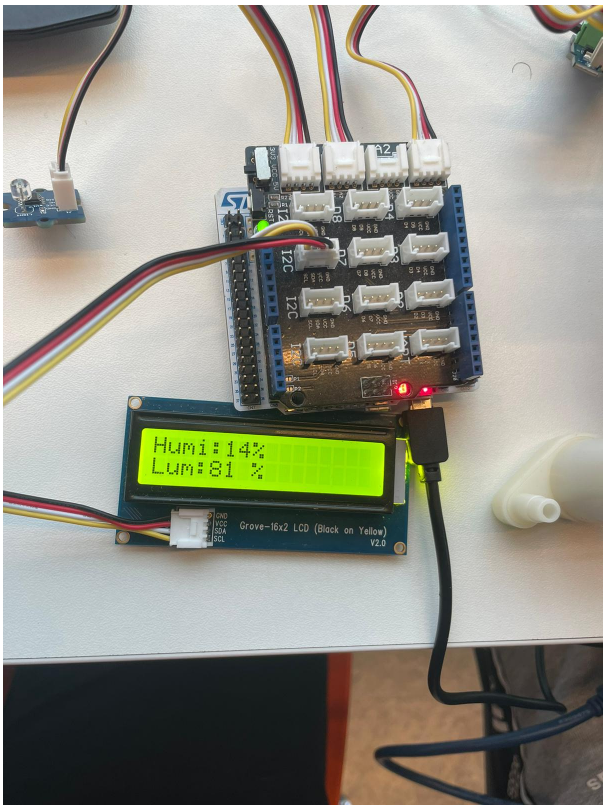


Figure 5: Affichage de la luminosté et l'humidité

7 Activité C : Activer une pompe pour l'arrosage

7.1 Cahier de charge

Dans cette phase, nous avons mis en œuvre le contrôle de l'arrosage en utilisant une pompe à eau et un module MOSFET agissant comme un interrupteur. L'objectif est de contrôler l'arrosage des plantes en activant la pompe pendant une durée prédéfinie à l'aide du module MOSFET. Nous devions utiliser la carte STM32 Nucleo WB55RG avec le shield Grove, un MOSFET, une pompe à eau et un boîtier d'alimentation 6V.

7.2 Schéma de câblage

Pour cette partie, nous avons élaboré un schéma de câblage détaillé à l'aide du logiciel Fritzing. Nous avons connecté le moteur de la pompe au microcontrôleur en utilisant les broches compatibles avec le signal PWM. De plus, nous avons prévu une alimentation adéquate pour le moteur afin de garantir un fonctionnement stable et sûr.

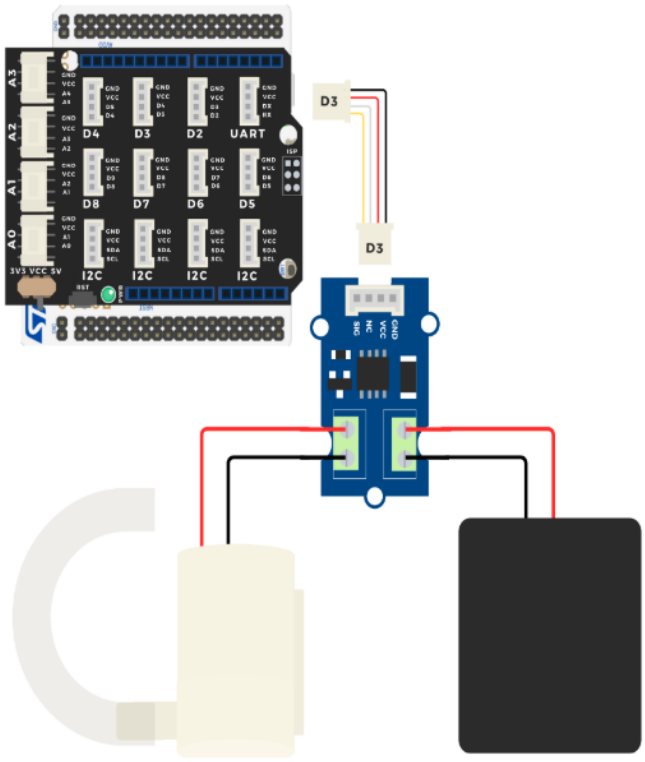


Figure 6: Schéma de câblage N°3

7.3 Programmation

Dans la phase de programmation, nous avons commencé par configurer les broches du microcontrôleur pour générer un signal PWM à la fréquence et à la durée appropriées pour contrôler la vitesse du moteur de la pompe. Ensuite, nous avons développé un code permettant de gérer le démarrage, l'arrêt et la vitesse du moteur en fonction des besoins d'arrosage des plantes.

```
1   while (1)
2   while (1)
3 {
4     // Activation de l'arrosage avec un rapport cyclique de 60% pendant
        5 secondes
```

```
    __HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_1, 60); // D finition du rapport cyclique
    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1); // D marrage du signal PWM
    HAL_Delay(5000); // Attente de 5 secondes
    HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_1); // Arr t du signal PWM
    HAL_Delay(5000); // Attente de 5 secondes avant la prochaine session d'arrosage

    // Lecture des valeurs d'humidit  du sol et de luminosit
    MX_ADC1_Init(); // Initialisation du canal ADC pour l'humidit  du sol
    HAL_ADC_Start(&hadc1); // D marrage de la conversion ADC
    HAL_ADC_PollForConversion(&hadc1, 1); // Attente de la fin de la conversion
    Humidite = HAL_ADC_GetValue(&hadc1); // R cup ration de la valeur d'humidit

    MX_ADC1_Init2(); // Initialisation du canal ADC pour la luminosit
    HAL_ADC_Start(&hadc1); // D marrage de la conversion ADC
    HAL_ADC_PollForConversion(&hadc1, 1); // Attente de la fin de la conversion
    Lum = HAL_ADC_GetValue(&hadc1); // R cup ration de la valeur de luminosit

    // Affichage des valeurs d'humidit  et de luminosit  sur un cran  LCD
    affichage_humidite(Humidite); // Affichage de l'humidit  du sol
    affichage_lum(Lum); // Affichage de la luminosit
    HAL_Delay(1000); // Pause d'une seconde avant la prochaine it ration
}
```

Ce code représente une boucle infinie où l'arrosage de la plante est activé avec un rapport cyclique de 60% pendant 5 secondes, suivi d'une période d'attente de 10 secondes avant la prochaine session d'arrosage. Ensuite, il lit périodiquement les valeurs d'humidité du sol et de luminosité à l'aide de deux canaux ADC différents. Enfin, il affiche ces valeurs sur un écran LCD et attend une seconde avant la prochaine itération.

7.4 Résultat Obtenu

Suite à la mise en œuvre du programme, nous avons réussi à contrôler efficacement le démarrage, l'arrêt et la vitesse du moteur de la pompe en fonction des commandes envoyées par le microcontrôleur via le signal PWM. Les tests ont démontré que le système était capable d'activer la pompe de manière fiable et de fournir un arrosage précis aux plantes, contribuant ainsi à la réussite du projet de plante connectée.

8 Activité D : la pompe selon un taux d'humidité

8.1 Cahier de charge

Dans cette étape, notre objectif était de mettre en place un système d'arrosage automatique qui se déclenche lorsque le niveau d'humidité du sol est trop bas. Pour cela, nous avons utilisé les données fournies par les capteurs d'humidité du sol, et nous avons programmé la pompe à eau en conséquence.

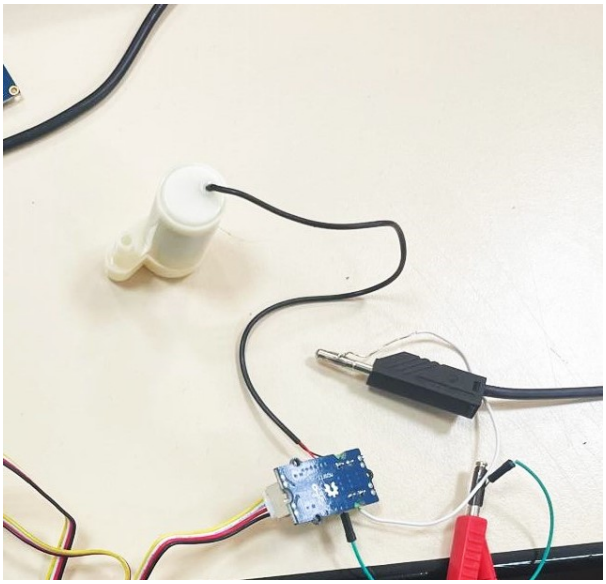


Figure 7: Fonctionnement du moteur

8.2 Schéma de câblage

Nous avons connecté les capteurs d'humidité du sol au microcontrôleur STM32 Nucleo WB55RG via les broches analogiques appropriées. Ensuite, nous avons relié la pompe à eau au module MOSFET pour le contrôle de l'alimentation. Le montage a été réalisé en respectant les spécifications électriques de chaque composant.

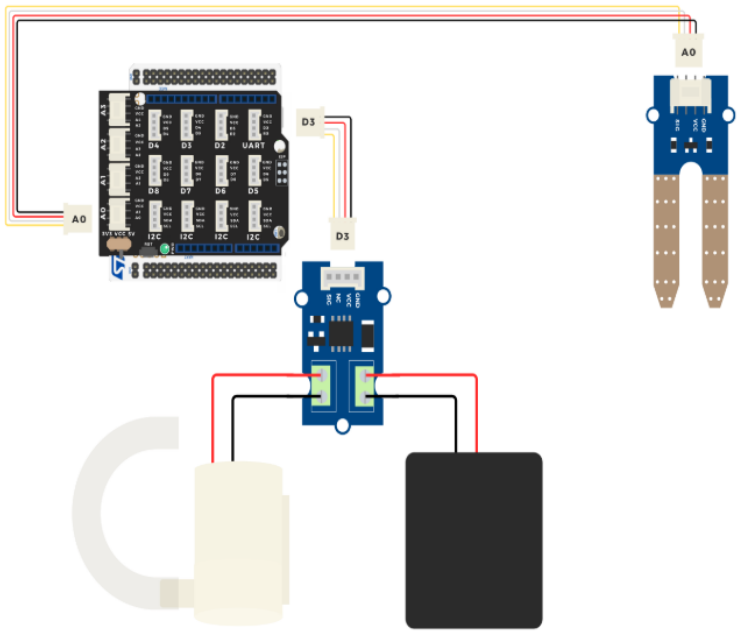


Figure 8: Schéma de câblage N°4

8.3 Programmation

Le programme élaboré a permis de surveiller en continu les niveaux d'humidité du sol . Lorsque le niveau d'humidité du sol est en dessous d'un seuil prédéfini et le système déclenche automatiquement l'arrosage en activant la pompe à eau pendant un certain temps. Cette logique a été implémentée en utilisant des instructions conditionnelles et des boucles pour surveiller en permanence les données des capteurs.

```c
static rgb_lcd lcddata;
uint16_t Humidite, Lum;

/* Private function prototypes
   -----------------------------------------------*/
void SystemClock_Config(void);
void PeriphCommonClock_Config(void);

/* Private user code
   ---------------------------------------------------------*/
void affichage_humidite(int Humidite) {
    char txt[20] = {0};
    sprintf(txt, "Humi:%d", Humidite);
    lcd_position(&hi2c1, 0, 0);
    lcd_print(&hi2c1, txt);
}

void affichage_lum(int Lum) {
    char txt[20] = {0};
    sprintf(txt, "Lum:%d", Lum);
    lcd_position(&hi2c1, 0, 1);
    lcd_print(&hi2c1, txt);
}

void MX_ADC1_Init2(void) {
    ADC_ChannelConfTypeDef sConfig = {0};

    hadc1.Instance = ADC1;
    hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
    hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
    hadc1.Init.LowPowerAutoWait = DISABLE;
    hadc1.Init.ContinuousConvMode = DISABLE;
    hadc1.Init.NbrOfConversion = 1;
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
    hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
    hadc1.Init.DMAContinuousRequests = DISABLE;
    hadc1.Init.Overrun = ADC_OVR_DATA_PRESERVED;
    hadc1.Init.OversamplingMode = DISABLE;

    if (HAL_ADC_Init(&hadc1) != HAL_OK) {
        Error_Handler();
    }

    sConfig.Channel = ADC_CHANNEL_2;
    sConfig.Rank = ADC_REGULAR_RANK_1;
    sConfig.SamplingTime = ADC_SAMPLETIME_2CYCLES_5;
    sConfig.SingleDiff = ADC_SINGLE_ENDED;
    sConfig.OffsetNumber = ADC_OFFSET_NONE;
    sConfig.Offset = 0;

    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
        Error_Handler();
    }
}

int main(void) {
```

```
    HAL_Init();
    SystemClock_Config();
    PeriphCommonClock_Config();
    MX_GPIO_Init();
    MX_USART1_UART_Init();
    MX_USB_PCD_Init();
    MX_TIM2_Init();
    MX_ADC1_Init();
    MX_I2C1_Init();

    __HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_1, 20);
    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
    HAL_ADCEx_Calibration_Start(&hadc1, ADC_SINGLE_ENDED);

    clearlcd();
    lcd_init(&hi2c1, &lcddata);

    while (1) {
        HAL_ADCEx_Calibration_Start(&hadc1, ADC_SINGLE_ENDED);
      clearlcd();
      lcd_init(&hi2c1,&lcddata);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

        MX_ADC1_Init();
        HAL_ADC_Start(&hadc1);
        HAL_ADC_PollForConversion(&hadc1, 1);
        Humidite = HAL_ADC_GetValue(&hadc1);
        Humidite =(Humidite/4095.0)*100;
        MX_ADC1_Init2();
        HAL_ADC_Start(&hadc1);
        HAL_ADC_PollForConversion(&hadc1, 1);
        Lum=HAL_ADC_GetValue(&hadc1);
        Lum=(Lum/5000.0)*100;
        affichage_humidite(Humidite);
        affichage_lum(Lum);
        if ( Humidite < 50) {
                    __HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_1, 30);
                    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
                    HAL_Delay(2000);
                    HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_1);
                    HAL_Delay(2000);
                }

                HAL_Delay(1);

  }
  /* USER CODE END 3 */
}


}
```

Ce code vise à contrôler un système de plante connectée.

Il utilise les périphériques intégrés du microcontrôleur, tels que les convertisseurs analogique-numérique (ADC) pour la lecture des capteurs, les timers pour générer des signaux PWM, et les interfaces de communication comme l'I2C pour la communication avec l'écran LCD.

Le programme configure d'abord les périphériques nécessaires, comme l'ADC et le timer, puis entre dans une boucle infinie où il lit périodiquement les valeurs des capteurs, les affiche sur l'écran LCD, et contrôle la pompe à eau en fonction des seuils prédéfinis.

En cas de conditions spécifiques détectées, telles qu'un niveau d'humidité bas, le programme active la pompe à eau pendant un certain temps, puis la désactive pour éviter l'excès d'arrosage.

8.4 Résultat Obtenu

Le système d'arrosage automatique a fonctionné avec succès, fournissant de l'eau aux plantes uniquement lorsque cela était nécessaire en fonction des conditions d'humidité du sol . Cette approche a permis d'optimiser l'utilisation de l'eau et de créer un environnement idéal pour la croissance des plantes.

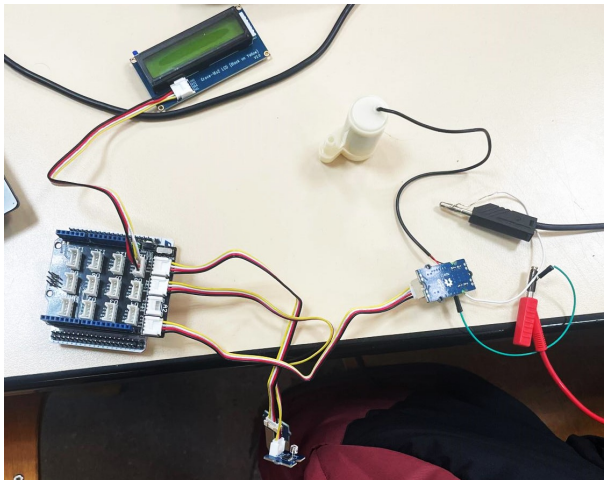


Figure 9: Fonctionnement global

9 Activité E : Récupérer des informations à l'aide d'un tag NFC et d'un smartphone

9.1 Cahier de charge

L'objectif de cette activité était de récupérer les informations des capteurs à l'aide d'un tag NFC et d'un smartphone équipé de la technologie NFC. Nous devions mettre en place un système permettant l'échange de données entre le microcontrôleur et le smartphone via la technologie NFC, en utilisant le module NFC ST M24SR64 inclus dans le kit.

9.2 Schéma de câblage

Avant de passer à la programmation, nous devons d'abord créer le schéma de câblage de ce projet sur Fritzing. Ceci nous permettra de mieux comprendre les étapes à suivre avant de passer à la pratique. Voici le schéma de câblage pour ce montage.

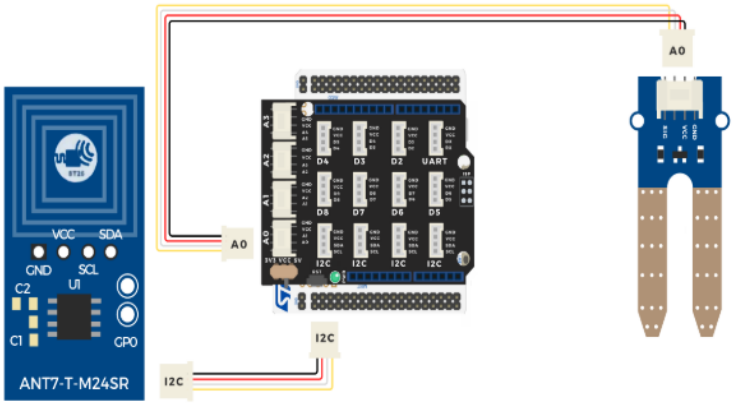


Figure 10: Schéma de câblage N°5

9.3 Programmation

Nous avons programmé le microcontrôleur pour interagir avec le module NFC et échanger des données avec un smartphone compatible NFC. Cela impliquait la mise en place des protocoles de communication NFC et la configuration des registres du microcontrôleur pour permettre la lecture et l'écriture de données sur le module NFC. Nous avons également développé une interface utilisateur sur le smartphone pour recevoir et afficher les informations transmises par le microcontrôleur.

9.4 Résultat Obtenu

Une fois la programmation terminée,malheuresement nous n'avons pu lire et écrire des données sur le module NFC à partir du smartphone.

10 Conclusion

Après avoir parcouru les différentes activités de ce projet de plante connectée, il est clair que la combinaison des technologies de capteurs, de microcontrôleurs et de communication sans fil offre un potentiel immense pour le domaine de l'agriculture intelligente. En exploitant les fonctionnalités avancées des microcontrôleurs STM32 et en intégrant une variété de capteurs, nous avons pu concevoir un système capable de surveiller de manière autonome les conditions environnementales importantes pour la croissance des plantes, telles que l'humidité du sol, la luminosité et d'autres paramètres.

La programmation et la configuration de chaque composant ont été des étapes cruciales pour assurer le bon fonctionnement du système. De la configuration des broches du microcontrôleur à la mise en œuvre d'algorithmes de conversion de données en passant par la gestion des interruptions et la communication sans fil via NFC, chaque partie du projet a nécessité une attention méticuleuse aux détails.

En intégrant des composants matériels tels que des capteurs, des écrans LCD, des modules NFC et des pompes à eau, nous avons créé un système polyvalent capable de répondre à une variété de besoins en matière de surveillance environnementale et d'irrigation des plantes.

En conclusion, ce projet a démontré le potentiel des technologies émergentes dans le domaine de l'agriculture intelligente. En combinant l'ingénierie matérielle et logicielle, nous avons créé un système complet et fonctionnel qui peut contribuer à améliorer la productivité et l'efficacité dans le secteur agricole tout en favorisant une gestion durable des ressources.

11 Bibliographie

1. ST-Nucleo-WB55RG Platform - Mbed
2. Datasheet ST M24SR64
3. Grove Light Sensor - Seeed Studio
4. Grove Light Sensor Datasheet - Mouser
5. STM32 In-Application Programming with NFC ST25 Dynamic Tag - ST
6. Getting Started with ADC - ST Wiki
7. Getting Started with I2C - ST Wiki