ESISAR

– Électronique–

PX112 Électronique: Cahier des charges d'une serre intelligente à base de microcontrôleur Arduino®

Jamyson Vetter - Julien Falquès
Juin 2025

Table des matières

1 Introduction **1**
1.1 **Motivation** . . . 1
1.2 **Problématique** . . . 1
1.3 **Objectifs** . . . 1
1.4 **Contexte** . . . 1

2 Besoins et contraintes **3**
2.1 Environnement d'utilisation . . . 3

3 Délais et budgets **4**
3.1 Délai . . . 4
3.2 Budgets prévisionnels . . . 4

4 Identification des briques **5**
4.1 Brique Matérielles . . . 5
4.2 Briques logicielles . . . 7

5 Tests unitaires **9**
5.1 Capteur DTH11 . . . 9
5.2 Capteur de sécherresse du sol . . . 10
5.3 Capteur de distance . . . 11
5.4 Test Relay . . . 13
5.5 Ecran `LCD` . . . 14
5.6 Alimentaion du circuit . . . 15

6 Conception **16**
6.1 Conception matérielle . . . 16
6.2 Conception logicielle . . . 17
6.3 Tests Globaux . . . 17

7 Conclusion **19**

8 Annexes **21**

1 Introduction

1.1 Motivation

Dans le cadre du Projet de **PX112-Electronique**, nous avons orienté notre travail vers le développement de systèmes automatisés pour l'agriculture urbaine et domestique, en particulier à travers la conception de serres intelligentes et autonomes. Ce domaine suscite un intérêt croissant, notamment en raison des avantages offerts par l'automatisation, qui permet de pallier les contraintes liées à la disponibilité des utilisateurs, en assurant une surveillance continue des paramètres environnementaux essentiels à la croissance des plantes. Ce type de solution s'avère particulièrement pertinent dans un contexte d'optimisation des ressources et de simplification d'accès à une agriculture domestique. Notre projet s'inscrit dans cette logique : concevoir un système de régulation automatisée des conditions de culture, adapté à une mini-serre à usage domestique. À défaut de client réel, l'étude repose sur l'hypothèse d'une entreprise fictive intéressée par la commercialisation d'une solution complète, accessible techniquement et économiquement.

1.2 Problématique

Les systèmes de culture en environnement contrôlé exigent une attention constante pour garantir des conditions optimales de développement. L'enjeu principal est de permettre à un particulier d'exploiter une serre performante et autonome, notamment en cas d'absence prolongée, sans nécessiter de compétences avancées en électronique ou en informatique.

Notre problématique est donc la suivante : Comment développer un système automatisé, accessible et économe, permettant à un utilisateur non expert de cultiver efficacement des plantes en intérieur ou dans un espace restreint, sans nécessiter une surveillance permanente ?

Pour répondre à cette problématique, il est nécessaire d'intégrer des capteurs et des actionneurs capables de surveiller et d'ajuster en temps réel les paramètres environnementaux essentiels à la croissance des plantes. Le défi consiste à garantir une autonomie fonctionnelle tout en gardant une simplicité d'usage, avec une interface utilisateur minimale.

1.3 Objectifs

L'objectif principal est la réalisation d'une serre connectée et autonome, pilotée par une carte **Arduino® Uno R3**[1], et capable de :

- Mesurer l'humidité du sol à l'aide d'un capteur **YL-69**[2] pour déclencher un arrosage automatique lorsque le seuil de sécheresse est atteint.
- Contrôler l'environnement thermique via un capteur **DHT11**[3] qui mesure la température et l'humidité ambiantes, et activer un système de ventilation si nécessaire.
- Réguler la température, l'humidité via le capteur **DTH11**, à l'aide d'un ventilateur et d'une résistance chauffante.
- Visualiser l'ensoleillement de notre serre grâce à un capteur **Difital Light Sensor**[4]
- Afficher en temps réel les données collectées et les actions entreprises sur un **écran LCD**[5] et 3 LEDs rouge jeune verte pour le niveau du réservoir.

Le tout sera intégré dans une structure physique en plexiglas, abritant les composants électroniques, les plantes et le réservoir d'eau.

1.4 Contexte

L'automatisation des serres s'inscrit dans une tendance de fond liée à l'agriculture connectée. Plusieurs solutions commerciales existent, mais elles restent généralement coûteuses et peu adaptées aux environnements domestiques. À l'inverse, de nombreux projets open-source (issus de plateformes telles que *Instructables*[6] ou *Arduino Project Hub*[7]) proposent des montages à bas coût, mais souvent peu documentés ou difficilement reproductibles.

Ce projet se fonde sur l'analyse critique de ces projets existants afin de proposer une approche optimisée, combinant des composants simples, une logique de fonctionnement robuste et une mise en œuvre accessible dans le cadre d'un projet éducatif : budget limité, complexité raisonnable, documentation claire, et objectifs fonctionnels concrets.

Afin d'aider à la compréhension de potentiels termes techniques lors de ce compte rendu, voici ci-dessous une visualisation de la carte **Arduino® Uno R3**[1],

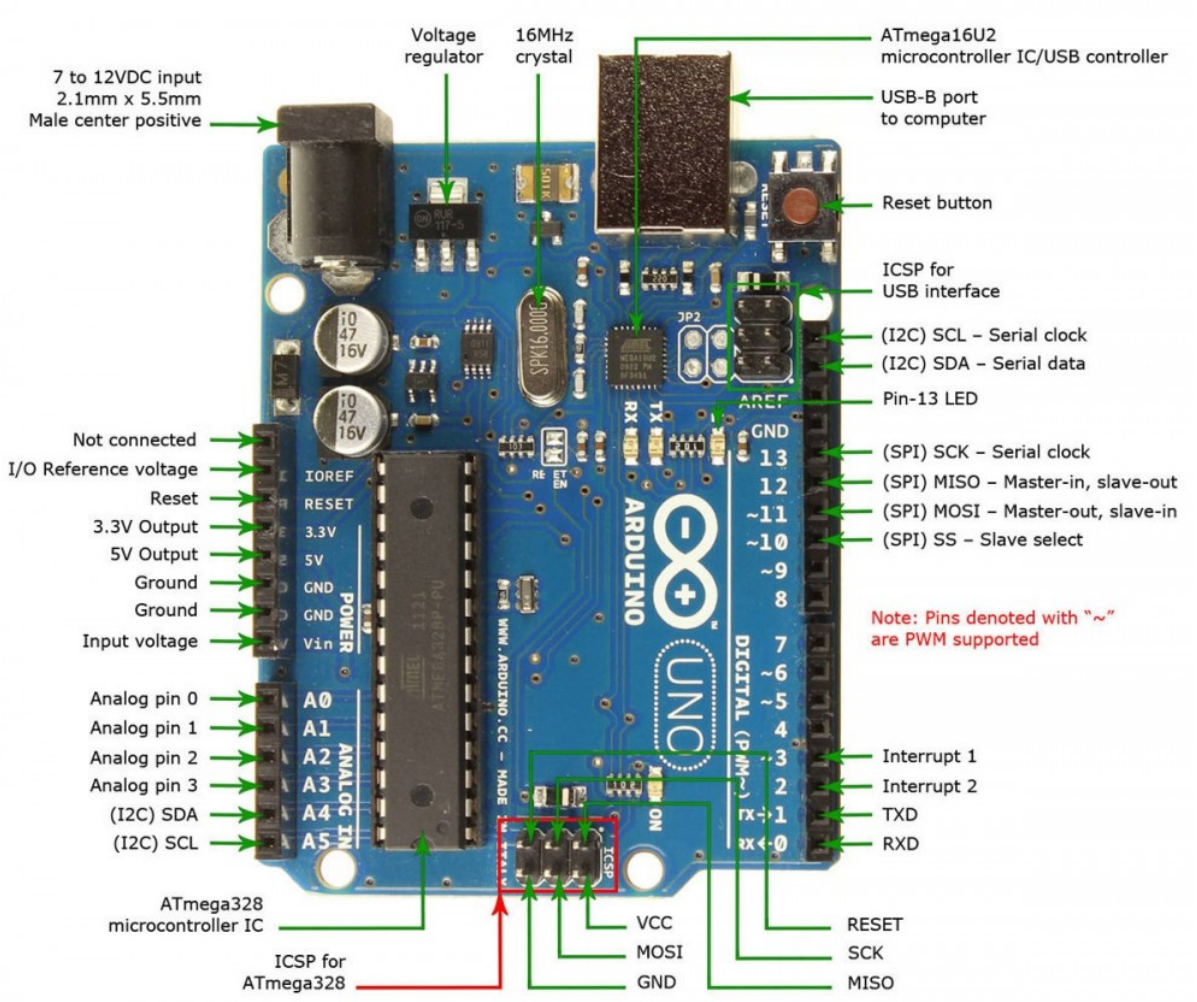


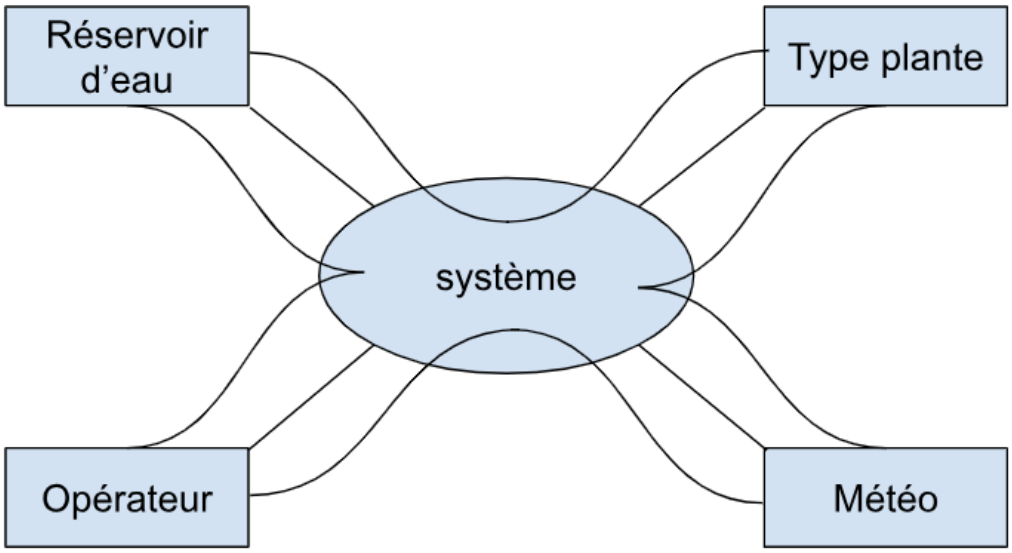
FIGURE 1 – *Carte* ***Arduino®*** Uno R3

Les pins dits "Digitaux" et "Analogiques" sont destinés à la récupération de `DATA` et à l'envoi de signaux vers nos différents modules. Les termes "`VCC`" et "`GND`" représentent respectivement une sortie de tension ou d'un courant ainsi que la terre.

2 Besoins et contraintes

2.1 Environnement d'utilisation

Le diagramme ci-dessous présente une vue macroscopique du système de serre intelligente. Il met en évidence les interactions entre le système automatisé central (piloté par une carte **Arduino®**) et son environnement fonctionnel.



Le système constitue l'unité centrale de traitement. Il reçoit des informations provenant de plusieurs éléments externes :

- **Le réservoir d'eau**, dont le niveau est surveillé afin d'assurer un arrosage autonome sans risque de fonctionnement à sec.
- **L'opérateur**, c'est-à-dire l'utilisateur du système, qui peut intervenir ponctuellement pour configurer, entretenir ou surveiller la serre.
- **Le type de plante**, qui détermine les paramètres environnementaux optimaux à respecter (luminosité, fréquence d'arrosage, etc.).
- **Les conditions météorologiques**, influant sur les décisions de ventilation ou d'activation de la résistance.

Les multiples connexions entre le système et ces entités traduisent des échanges d'informations et de commandes, assurant ainsi une gestion optimale des ressources et du climat de la serre. De plus, des liens entre certaines entités (par exemple entre le réservoir d'eau et le type de plante) reflètent la complexité et l'interdépendance des paramètres à prendre en compte.

Ce diagramme sert de référence pour la définition des fonctions détaillées et la conception du système dans les parties suivantes du rapport.

3 Délais et budgets

3.1 Délai

Le projet s'étend sur une période de 24 heures réparties en 8 séances de 3 heures chacune. Cette organisation permet de progresser étape par étape, en alliant phases de conception, de montage et de tests fonctionnels. Chaque séance est planifiée afin de respecter le calendrier global du projet tout en garantissant un travail approfondi à chaque étape.

3.2 Budgets prévisionnels

Les coûts liés à la réalisation du projet sont estimés comme suit :

Poste	Coût estimé (€)	Remarques
Capteurs (DHT11, YL-69, relay)	20	Achat de capteurs divers
Capteur niveau d'eau (flotteur ou ultrason)	10	Selon type de capteur choisi
Carte Arduino Uno	25	Module microcontrôleur
Écran LCD 16x2	10	Affichage des données
Structure en plexiglas	80	Matériaux pour la serre
Terre et terreau	10	Support de culture
Graines	5	Plantes sélectionnées
Total estimé	**165**	

4 Identification des briques

4.1 Brique Matérielles

Les briques matérielles constituent l'ensemble des composants physiques intervenant dans le système de régulation automatisée de la serre. Elles reposent principalement sur une carte **Arduino®** qui collecte les données environnementales à partir des capteurs, analyse ces informations et commande, via des relais, les actionneurs destinés à réguler les conditions de culture.

Les principaux éléments matériels sont :

— Les capteurs, qui fournissent des mesures précises de l'environnement :
  - Capteur d'humidité du sol (**YL-69**), pour détecter la nécessité d'arrosage.
  - Capteur de température et d'humidité de l'air (**DHT11**, pour contrôler la ventilation.
  - Capteur de niveau d'eau (capteur à ultrasons **Ultrasonic ranger v3**[**8**], pour surveiller la réserve d'eau.
  - capteur de luminosité (**Digital light senvor**)

— La carte **Arduino®**, qui sert d'unité centrale de traitement. Elle reçoit les signaux des capteurs et déclenche, en fonction de seuils programmés, les actionneurs via des modules relais.

— Les relais, qui jouent le rôle d'interrupteurs commandés par l'**Arduino®**[**1**], permettant de piloter des charges électriques telles que :
  - La pompe à eau pour l'irrigation.
  - Le ventilateur pour la ventilation.
  - Le système de résistance chauffante pour la température.

— L'alimentation électrique se fera à l'aide d'une alimentation à découpage d'ordinateur comprenant une sortie `5V` pour les composants électroniques et une sortie `12V` pour les actionneurs plus puissants. Une attention particulière est portée à la sécurité électrique, car cette alimentation convertit directement du `230V`.

L'architecture matérielle est conçue de manière modulaire et évolutive, ce qui permet de simplifier la maintenance du système et d'ajouter de nouveaux composants si nécessaire. Nous avons fait attention à la robustesse du câblage et à la qualité des connexions, conditions indispensables pour garantir la fiabilité et le bon fonctionnement de l'installation sur le long terme.

Ce schéma représentant le câblage physique réel est également présenté ci-dessous. Il précise les broches utilisées pour chaque composant ainsi que les interconnexions nécessaires au bon fonctionnement de l'ensemble.

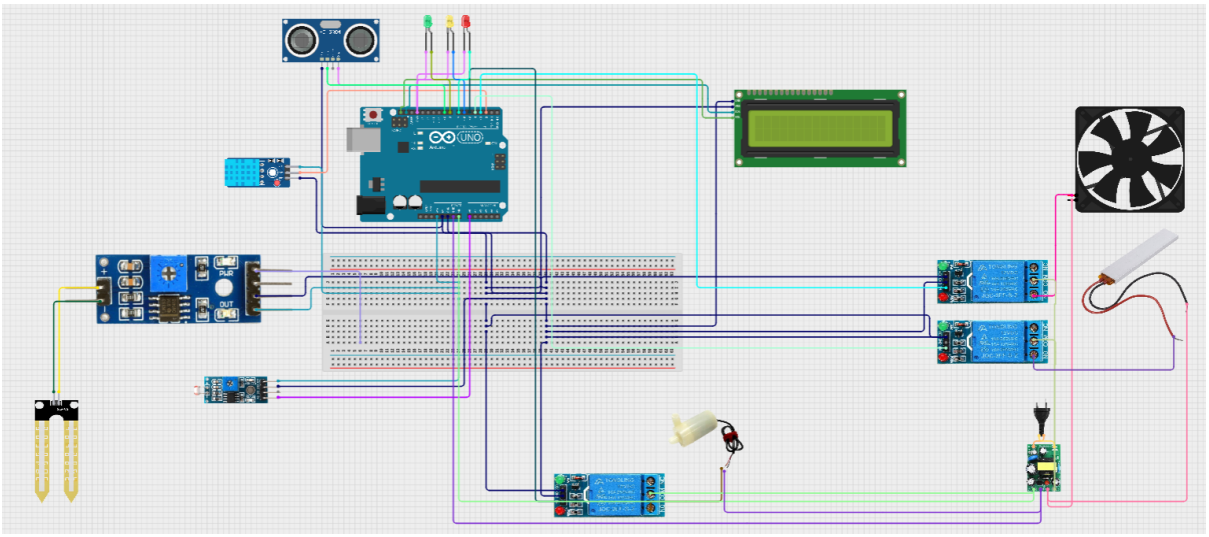


FIGURE 2 – Structure globale des câblages du projet

Remarque *: Le shéma ci-dessus est uniquement a but informatif car le logiciel utilisé ne dispose pas de tout les capteurs que nous utilisons.*

Remarque *: Le capteur à ultrasons utilisé dans notre projet ne dispose que de trois broches, contrairement à la version standard représentée sur le schéma, qui en comporte habituellement quatre (`VCC`, `GND`, `Trig`, `Echo`). Dans notre cas, le module intègre un circuit de traitement combinant les broches de signal, ce qui justifie cette différence de câblage. Cette particularité a été prise en compte lors du montage sans compromettre le fonctionnement du système. Il en est de même pour l'écran `LCD` qui a un cablage différents(car nous n'avons pas utilisé de module `I2C` et pour le capteur de lumière qui utilise les ports `SDA` et `SDL`*

4.2 Briques logicielles

Les briques logicielles correspondent aux fonctions programmées dans la carte **Arduino®** permettant de piloter les actionneurs, de traiter les données issues des capteurs et d'afficher les informations sur l'écran. Elles sont décomposées en fonctions principales clairement identifiées, selon une logique inspirée de la fiche fonctionnelle du **Cycle de Développement Fonctionnel** (CDF).

La structuration du programme se fait autour de deux grandes familles de fonctions :

A. Contrôle du matériel (Action)

Cette partie du programme regroupe les fonctions qui permettent à l'**Arduino®** d'agir sur l'environnement en pilotant directement les actionneurs du système.

- **A.1 Contrôle du ventilateur** : Activation/désactivation du ventilateur en fonction de la température et de l'humidité ambiantes mesurées par le capteur DHT11. Des seuils de déclenchement sont définis pour maintenir un climat optimal.
- **A.2 Contrôle de la Résistance Chauffante** : Activation/désactivation de la résistance chauffante en fonction de la température et de l'humidité ambiantes mesurées par le capteur DHT11. Des seuils de déclenchement sont définis pour maintenir un climat optimal.
- **A.3 Affichage des données** : mise à jour de l'écran `LCD 16x2` avec les principales valeurs mesurées (T°C, humidité, état de l'arrosage, etc.) et clui des LEDs rouge jaune et verte servant a représenter plus visuellement le niveau du réservoir. Cela permet à l'utilisateur de suivre en temps réel l'état de la serre.
- **A.4 Contrôle de l'arrosage** : activation de la pompe à eau en cas de sol sec, selon les mesures du capteur YL-69. Elle tient également compte du niveau d'eau pour éviter le fonctionnement à sec.

B. Récupération des données (Acquisition)

Ces fonctions assurent la lecture des valeurs issues des capteurs et leur mise à disposition pour le contrôle du système.

- **B.1 Lecture de la température et de l'humidité de l'air** : via le capteur DHT11, les données sont utilisées pour piloter le ventilateur.
- **B.2 Lecture de la luminosité ambiante** : le capteur de lumière fournit une valeur analogique ou numérique permettant d'évaluer l'intensité lumineuse.
- **B.3 Lecture de l'humidité du sol et du niveau d'eau** : ces valeurs proviennent respectivement du capteur d'humidité du sol (YL-69) et d'un capteur de niveau ultrasons et conditionnent l'activation de l'arrosage.

Cette architecture logicielle permet une meilleure lisibilité du code, une séparation claire des responsabilités, et facilite la mise au point ainsi que les tests individuels des fonctionnalités.

Fonction	Niveau exigé	Critère d'évaluation
F1 – Maintenir l'humidité du sol	$> 40\%$	Mesure par le capteur YL-69
F2 – Maintenir la température ambiante	18°C à 30°C	Mesure par le capteur DHT11
F3 – Afficher le niveau d'eau avec 3 LED	Actualisation < 20 s	Données visibles et lisibles en temps réel
F4 – Afficher les données sur l'écran LCD	Actualisation < 20 s	Données visibles et lisibles en temps réel
F5 – Gérer le niveau d'eau du réservoir	Niveau suffisant avant activation	Détection via capteur à ultrasons

5 Tests unitaires

5.1 Capteur DTH11

Ce capteur a pour but d'envoyer les données du niveau d'humidité et de la température interne à la serre en temps réel.
Pour cela, nous avons réalisé le montage suivant :

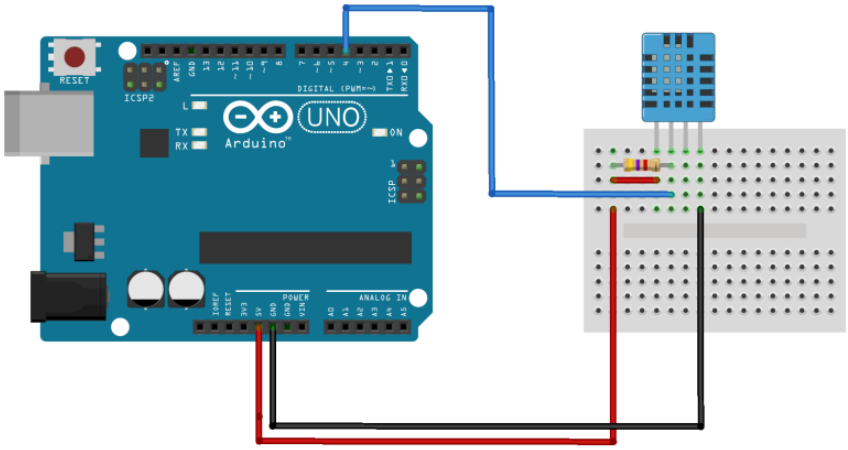


FIGURE 3 – Circuit de branchement du DHT11

Nous avons branché le capteur de type DTH11 au pin `4` de l'**Arduino®**, son `VCC` et `GND` au pin `5V` et `GND` de l'**Arduino®**. Puis, à l'aide du code suivant, lancer la récupération des données. Pour tester le capteur **DHT11**, il convient tout d'abord d'ajouter la bibliothèque correspondante dans l'environnement de développement **Arduino®**, via le gestionnaire de bibliothèques. Une fois la bibliothèque installée, le code d'exemple peut être téléversé sur la carte **Arduino®**. Après le téléversement, il faut ouvrir le moniteur série en veillant à régler le débit en bauds à 9600. Les valeurs de température et d'humidité relevées par le capteur seront alors affichées en temps réel dans le moniteur série, ce qui permet de vérifier le bon fonctionnement du dispositif.

```
#include <dht11.h>
#define DHT11PIN 4

dht11 DHT11;

void  setup()
{
  Serial.begin(9600);

}

void loop()
{
  Serial.println();

  int chk = DHT11.read(DHT11PIN);

  Serial.print("Humidity (%): ");
  Serial.println((float)DHT11.humidity, 2);

  Serial.print("Temperature  (C): ");
  Serial.println((float)DHT11.temperature, 2);

  delay(2000);

}
```

5.2 Capteur de sécherresse du sol

Ce capteur a pour but d'envoyer les données du niveau d'humidité du sol de la serre en temps réel. Pour cela, nous avons réalisé le montage suivant :

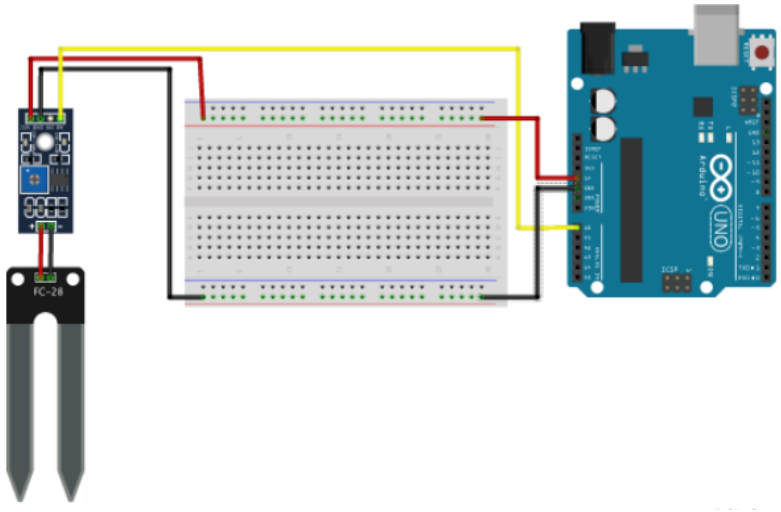


FIGURE 4 – Circuit de branchement du YL-69 ([9])

Le capteur de type YL-69 est branché au pin `A0` de l'**Arduino®**, son `VCC` et `GND` au pin `5V` et `GND` de l'**Arduino®**. Puis, à l'aide du code suivant, lancer la récupération des données. Une fois les connexions établies, le code de test peut être chargé dans l'environnement **Arduino® IDE**. Celui-ci contient l'initialisation du capteur, la lecture périodique des valeurs et leur affichage via le port série. Après avoir sélectionné le bon type de carte et le port de communication dans l'IDE, le programme est téléversé vers la carte **Arduino®**. Lorsque le téléversement est terminé, l'ouverture du moniteur série, avec une vitesse de transmission fixée à 9600 bauds, permet de visualiser en temps réel les mesures de température et d'humidité relevées par le capteur. Cette procédure permet de valider le bon fonctionnement du système avant intégration dans une application plus complète.

```
int rainPin = A0;

// you can adjust the threshold value
int  thresholdValue = 1000;

void setup(){
  pinMode(rainPin, INPUT);
  Serial.begin(9600);  // Streaming 9600 bits of Sensor Data per Second
}

void loop() {
  //  read the input on analog pin 0:
  int sensorValue = analogRead(rainPin);
  Serial.println("Soil-humidity sensor value: ");
  Serial.print(sensorValue);
  if(sensorValue < thresholdValue){
    Serial.println(" - Plant doesn't need  watering now...");

  }
  else {
    Serial.println(" - Time to  water your plant now...");

  }
  delay(3000);
}
```

Cela nous a permis, à travers des tests dans différents environnements (terre sèche, mouillée et juste arrosée) d'établir les paliers d'arrosage pour une plante méditerranéenne lambda suivant :

Paramètre	Intervalle recommandé
Température ambiante	18°C à 30°C
Humidité du sol	entre 350 et 950
Humidité de l'air	20% à 40%

5.3 Capteur de distance

Ce capteur a pour but d'envoyer les données en temps réel du niveau d'eau du réservoir.
Pour cela, nous avons réalisé le montage suivant (**schéma indicatif** car le composant utilisé n'est pas disponible sur l'application) :

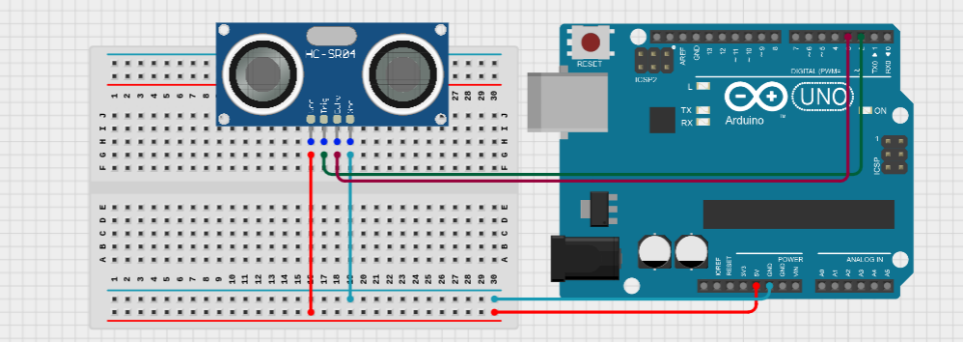


FIGURE 5 – Circuit de branchement indicatif du capteur de distance

Nous avons branché le capteur Ultrasonic Ranger V3 au pin `3` de l'**Arduino®**, son `VCC` et `GND` au pin `5V` et `GND`. Puis, à l'aide du code suivant, lancer la récupération des données.

```
#include "Arduino.h"
class Ultrasonic
{
    public:
    Ultrasonic(int pin);
    void DistanceMeasure(void);
    double microsecondsToCentimeters(void);
    double microsecondsToInches(void);
    private:
    int this_pin;//pin number of Arduino that is connected with SIG pin of
        Ultrasonic Ranger.
    long duration;// the Pulse time received;
};
Ultrasonic::Ultrasonic(int pin)
{
    this_pin = pin;
}
/*Begin the detection and get the pulse back signal*/
void Ultrasonic::DistanceMeasure(void)
{
    pinMode(this_pin, OUTPUT);
    digitalWrite(this_pin, LOW);
    delayMicroseconds(2);
    digitalWrite(this_pin, HIGH);
    delayMicroseconds(5);
    digitalWrite(this_pin,LOW);
    pinMode(this_pin,INPUT);
    duration = pulseIn(this_pin,HIGH);
}
/*The measured distance from the range 0 to 400 Centimeters*/
double Ultrasonic::microsecondsToCentimeters(void)
{
    return duration/29.0/2.0;
}
/*The measured distance from the range 0 to 157 Inches*/
double Ultrasonic::microsecondsToInches(void)
{
    return duration/74.0/2.0;
}

Ultrasonic ultrasonic(2);
void setup()
{
    Serial.begin(9600);
}
void loop()
{
    double RangeInInches;
    double RangeInCentimeters;
    ultrasonic.DistanceMeasure();// get the current signal time;
    RangeInInches = ultrasonic.microsecondsToInches();//convert the time to
        inches;
    RangeInCentimeters = ultrasonic.microsecondsToCentimeters();//convert the
        time to centimeters
    Serial.println("The distance to obstacles in front is: ");
    Serial.print(RangeInInches);//0~157 inches
    Serial.println(" inch");
    Serial.print(RangeInCentimeters);//0~400cm
    Serial.println(" cm");
    delay(1000);
}
```

Etant donné que notre réservoir a une profondeur de 25 cm, nous avons défini les paliers de remplissage (en supposant que notre capteur est légèrement surélevé du réservoir) suivants :

Paramètre	Intervalle recommandé
niveau haut	< 17.5
niveau moyen	$25 <$ niveau < 17.5
niveau bas	$25 <$

5.4 Test Relay

Ce test a pour but de vérifier le fonctionnement d'un Relay, un relay fonctionne de la même manière qu'un interrupteur à la différence près qu'il ne s'actionne pas à la main mais à l'aide d'un signal qu'enverra notre **Arduino®**. Pour cela nous avons utilisé le montage suivant :

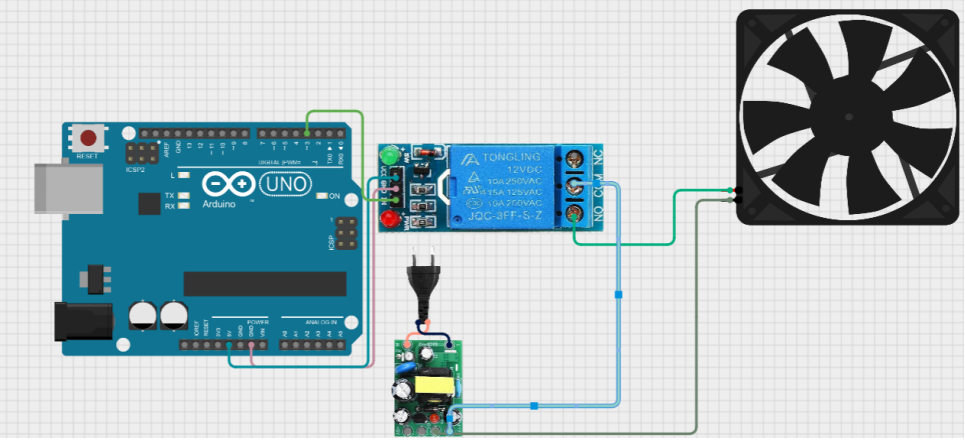


FIGURE 6 – Circuit de branchement du relay

Pour effectuer ce test, nous avons utilisé notre ventilateur qui fonctionne en `12V` et donc qui ne peut être alimenté à l'aide de l'**Arduino®**. Nous avons branché le relay au pin `3` de l'**Arduino®**, son `VCC` et `GND` au pin `5V` et `GND` de la carte. Le ventilateur lui a été connecté au pin `IN` du relay pour son `VCC` et la terre sur la sortie normalement ouverte afin qu'il ne se déclenche que lorsqu'un signal est envoyé au relay. Puis nous avons testé à l'aide du code suivant :

```
int fan_relay_pin = 3;
int cycle = 0;
void setup() {
  pinMode(fan_relay_pin, OUTPUT);          // initialise le pin du relay du
      ventilo en OUTPUT
}
void setup() {
    if (cycle ==0)digitalWrite(fan_relay_pin,HIGH);
    else if (cycle ==1)digitalWrite(fan_relay_pin,LOW);
    i++
    if (cycle == 2) cycle=0;
    delay(10000);
}
```

Cela nous a permis de vérifier le bon fonctionnement du relay et du ventilateur dans le cadre voulu.

5.5 Ecran LCD

Ce Test a pour but de vérifier le fonctionnement de notre écran LCD, Pour cela nous avons utilisé le montage suivant :

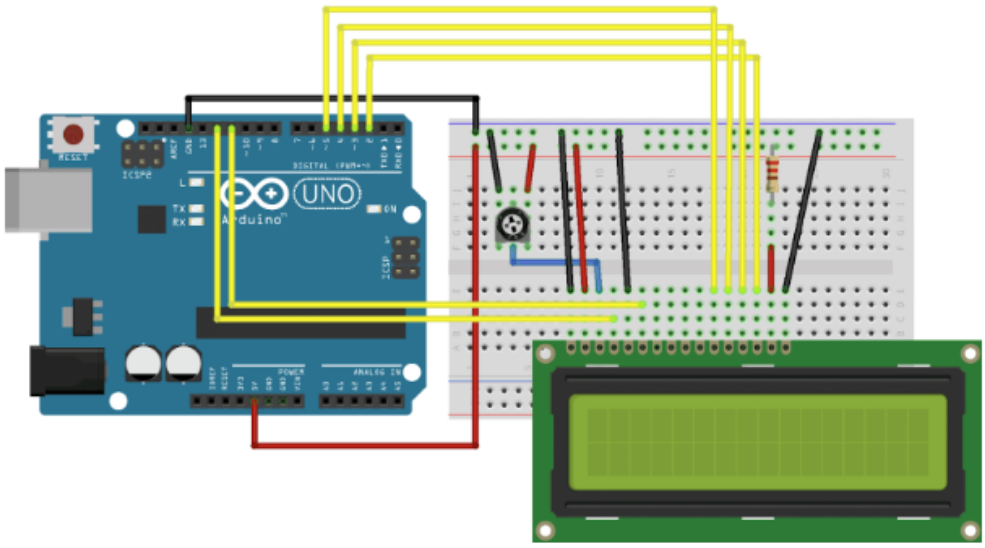


FIGURE 7 – *Branchement de l'écran LCD à l'**Arduino UNO®***

Pour effectuer ce test, nous avons utilisé le code suivant :

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

5.6 Alimentaion du circuit

Pour alimenter le circuit, nous avons décidé d'utiliser une alimentation de PC dû à sa grande capacité de sortie, comme dit ci-dessus. Pour cela, il a fallu vérifier son bon fonctionnement lors de la mise sous tension et que les sorties avaient la bonne valeur. Il nous a fallu aussi "*shunter*" l'alimentation. Ce procédé vise à relier directement le câble de vérification de l'alimentation à la terre. Ce câble vérifie que l'alimentation est bien reliée à une carte mère lorsqu'elle est allumée, or ce n'est pas notre cas. Si ce n'est pas fait, l'alimentation ne se mettra jamais en marche.

Remarque** : Pour effectuer l'entièreté de nos test nous avons utilisé les codes et exemples disponibles sur les datasheet disponible sur le site de **Groove®** ou celui d'**Arduino®

6 Conception

La présente section vise à détailler la démarche de conception du système, en s'appuyant sur les blocs matériels et logiciels identifiés précédemment. Alors que la partie précédente se limitait à la description des composants isolés — capteurs, actionneurs, modules de commande ou fonctions logicielles —, il s'agit ici de justifier les choix effectués, de décrire la manière dont les éléments ont été interconnectés et coordonnés, et de présenter l'architecture fonctionnelle complète du système.

La conception matérielle (section 6.1) s'attarde sur l'assemblage physique des composants, le schéma de câblage réel, ainsi que les contraintes techniques prises en compte pour garantir un fonctionnement fiable et sécurisé. Cette sous-partie met en évidence la logique d'intégration entre les capteurs, les relais et les actionneurs, en interaction avec la carte **Arduino®**.

La conception logicielle (section 6.2) expose l'organisation du programme embarqué. Elle décrit la structure du code, les interactions entre les différentes fonctions de lecture, de traitement et d'action, ainsi que les seuils de déclenchement retenus. Un soin particulier est apporté à la clarté du programme et à la modularité des fonctions, facilitant l'adaptation du système à d'autres configurations.

Enfin, les tests globaux (section 6.3) permettent de vérifier le bon fonctionnement du système dans son ensemble. Cette phase rassemble l'ensemble des éléments précédemment testés individuellement afin d'évaluer leur cohérence en situation réelle. Elle vise à valider les performances du système automatisé dans des conditions représentatives d'un usage domestique.

6.1 Conception matérielle

La conception matérielle de la serre intelligente repose sur l'interconnexion des capteurs et des actionneurs autour d'un microcontrôleur **Arduino®** Uno R3. Cette architecture physique permet de surveiller et d'ajuster en temps réel les conditions de culture à l'intérieur d'une structure en plexiglas conçue sur mesure.

L'ensemble des capteurs sélectionnés répond à des critères de simplicité d'intégration, de coût réduit et de fiabilité. Le capteur YL-69 permet de mesurer l'humidité du sol, tandis que le DHT11 collecte les données de température et d'humidité ambiantes. Un capteur de niveau d'eau à ultrasons (Ultrasonic Ranger v3) surveille le niveau du réservoir afin d'éviter le fonctionnement à sec de la pompe.

Ces informations sont traitées par la carte **Arduino®**, qui pilote, via des modules relais, les différents actionneurs : une pompe pour l'irrigation, un ventilateur pour la ventilation, et une résistance chauffante pour le maintien thermique. L'alimentation électrique est assurée par une alimentation à découpage issue d'un ancien PC, délivrant du `5V` pour la carte et du `12V` pour les éléments de puissance. La sécurité électrique a été particulièrement prise en compte lors du câblage.

L'ensemble est complété par un écran `LCD 16x2` sans `I2C`, utilisé pour l'affichage des mesures et des états du système. Chaque composant a été raccordé manuellement, en veillant à une identification claire des broches et à une organisation soignée du câblage pour limiter les interférences et faciliter la maintenance.

6.2 Conception logicielle

La conception logicielle repose sur un programme développé dans l'environnement **Arduino® IDE** Le code s'organise selon deux grandes familles de tâches : l'acquisition des données et le contrôle des actionneurs.

Les fonctions d'acquisition regroupent la lecture des valeurs issues des capteurs : humidité du sol (YL-69), température et humidité de l'air (DHT11), luminosité ambiante (capteur digital), et niveau d'eau (capteur à ultrasons). Ces données sont analysées en temps réel par la carte afin d'évaluer l'état de la serre.

En parallèle, les fonctions de contrôle déclenchent les actionneurs en fonction de seuils prédéfinis. Si le sol est trop sec et que le réservoir contient suffisamment d'eau, la pompe est activée pour arroser les plantes. Si la température dépasse un seuil, le ventilateur est mis en marche, et en cas de température trop basse, une résistance chauffante est activée. Les conditions de déclenchement sont codées avec une logique conditionnelle simple (`if/else`) pour garantir un comportement fiable et prévisible.

Un affichage temps réel des mesures et des actions en cours est assuré par l'écran `LCD`. Le programme actualise en continu les lignes de l'écran afin de permettre à l'utilisateur de visualiser l'état de la serre sans interaction supplémentaire.

Une attention particulière a été portée à la lisibilité du code, à la séparation des fonctions, et à l'usage de constantes pour faciliter les ajustements de seuils ou la reproductibilité du système grâce à la facilté du changement des pins des différents modules. Cette structure permet d'envisager facilement des extensions futures, telles que l'intégration d'un module `RTC` (**R**eal **T**ime **C**lock) soit une horloge en temps réel, ou de communication sans fil.

6.3 Tests Globaux

Les tests globaux ont été réalisés après l'intégration complète des briques matérielles et logicielles afin de valider le fonctionnement du système dans des conditions proches d'une utilisation réelle. L'objectif était de vérifier la cohérence des échanges entre capteurs, microcontrôleur et actionneurs, ainsi que la fiabilité des réactions du système face à des variations de l'environnement.

Chaque fonction a été testée dans des scénarios contrôlés. L'humidification automatique a été déclenchée en simulant un sol sec, et le bon fonctionnement de la pompe a été confirmé, avec vérification du niveau d'eau pour éviter tout fonctionnement à sec. De même, des variations de température ont été reproduites pour observer l'activation automatique du ventilateur et de la résistance chauffante selon les seuils programmés.

L'écran `LCD` a également été surveillé tout au long des tests afin de s'assurer de la mise à jour correcte des informations affichées. Une attention particulière a été portée à la stabilité de l'affichage, notamment lors de la commutation entre les différents états du système.

Enfin, le bon fonctionnement général du système a été validé sur une période prolongée, avec une alimentation continue et des cycles complets de régulation. Aucun dysfonctionnement critique n'a été observé, ce qui témoigne de la robustesse de l'ensemble.

Cependant, nous avons répertorié certains problèmes lors de l'assemblage :

— Le premier était celui du cache conçu en 3D pour accueillir les composants. Ce dernier étant trop petit, il ne pouvait pas accueillir l'ensemble des composants. De plus, il était de couleur noire, ce qui pourrait entraîner une surchauffe des composants au soleil. Et enfin un toit dépassant permettrait de protéger entièrement les composants de la pluie.

— Le capteur de type DHT11 est un capteur lent conçu pour suivre principalement les variations de température dans une pièce, donc lors de notre test global avec une variation rapide de température, celui-ci a mis du temps a réagir ce qui nous a fait croire à un dysfonctionnement.

— L'utilisation de bornes **Wago®** pour relier les câbles de l'alimentation à des câbles utilisables sur l'**Arduino**. Ces derniers sont pratiques pour des tests mais ne sont pas conformes à la norme NF C 15-100. Si ce projet se veut d'être conforme, il faudrait alors utiliser des sucres et les recouvrir de résine pour éviter tout court circuit.

— Si l'on veut encore développer les systèmes avec par exemple une lampe UV ou le module RTC précédemment cité, cela nécessiterait l'utilisation de bus pour augmenter le nombre de pins disponible sur l'**Arduino®**

— Lors de la première expérimentation, nous n'avons eu aucun courant et aucune tension de sortie de l'alimentation de PC. Nous avons d'abord crû que, dû a son âge, elle était dysfonctionnelle. Cependant, nous avons découvert le principe de *"shunter"* une alimentation après avoir tester avec une seconde alimentation.

— Enfin lors des premiers montages avec l'alimentation *"shuntée"* nous avons eu des problèmes en tentant de combiner l'alimentation de l'**Arduino®** et celle de l'alimentation pour nos composants. Nous avons donc dû effectuer nos tests suivants en alimentant l'ensemble des composants et l'**Arduino®** (port `Vin`) avec l'alimentation ce qui nous a permis de régler notre problème. Pour débuguer nous avons utilisé l'écran `LCD` ce qui nous a avancé sur l'affichage final obtenu avec ce dernier.

7 Conclusion

Le projet de serre intelligente développé dans le cadre du module PX112 a permis de concevoir et de tester une solution fonctionnelle d'automatisation de l'environnement de culture en intérieur, reposant sur une architecture simple, modulaire et accessible. En associant des capteurs essentiels (température, humidité, luminosité, niveau d'eau) à un microcontrôleur Arduino® et à des actionneurs commandés de manière conditionnelle, le système parvient à maintenir des conditions favorables au développement de plantes, tout en minimisant les interventions de l'utilisateur.

Les différents tests unitaires et globaux ont confirmé la fiabilité des modules intégrés, ainsi que la cohérence de l'ensemble logiciel dans la gestion des priorités et l'affichage en temps réel des informations utiles. L'approche retenue a mis l'accent sur la clarté du code, la lisibilité matérielle du montage, et la sécurité électrique, garantissant une base solide pour de futurs déploiements ou améliorations.

Des perspectives d'évolution intéressantes sont envisagées pour renforcer l'autonomie énergétique et la durabilité du système. À terme, l'alimentation pourrait être assurée par un panneau solaire similaire à ceux utilisés dans les parcs électriques pour animaux qui sont autonomes, avec un système de régulation de la charge. Par ailleurs, le réservoir pourrait être remplacé ou complété par un dispositif de récupération d'eau de pluie, muni d'un trop-plein pour éviter les débordements. De tels ajouts rendraient la serre presque totalement autonome, ne dépendant que des apports naturels en énergie solaire et en eau.

Ce projet pose ainsi les bases d'une solution d'agriculture domestique connectée, adaptable et respectueuse des contraintes environnementales, tout en restant économiquement viable et techniquement reproductible.

Références

[1] Arduino. *Arduino Uno Rev3*. 2025. URL : https://docs.arduino.cc/hardware/uno-rev3/.

[2] Seeed Studio. *Grove - Moisture Sensor*. 2025. URL : https://wiki.seeedstudio.com/Grove-Moisture_Sensor/.

[3] Arcaege Cengiz. *Using DHT11*. 2018. URL : https://projecthub.arduino.cc/arcaegecengiz/using-dht11-12f621.

[4] Seeed Studio. *Grove - Digital Light Sensor*. 2022. URL : https://wiki.seeedstudio.com/Grove-Digital_Light_Sensor/.

[5] Arduino. *Liquid Crystal Displays (LCD) with Arduino*. 2025. URL : https://docs.arduino.cc/learn/electronics/lcd-displays/.

[6] Instructables Community. *Smart Greenhouse Monitoring System*. 2025. URL : https://www.instructables.com.

[7] Arduino Project Hub. *Automated Greenhouse*. 2025. URL : https://create.arduino.cc/projecthub/projects/automated-greenhouse.

[8] Seeed Studio. *Grove - Ultrasonic Ranger*. 2025. URL : https://wiki.seeedstudio.com/Grove-Ultrasonic_Ranger/.

[9] Sheekar Banerjee. *YL69 Soil Moisture Sensor with Arduino*. 2022. URL : https://projecthub.arduino.cc/sheekar/yl69-soil-moisture-sensor-with-arduino-sheekar-banerjee-e2d9ca.

8 Annexes

Glossaire

Terme	Définition
Arduino®	Plateforme de prototypage open-source basée sur un microcontrôleur, permettant de lire des capteurs et contrôler des actionneurs.
Microcontrôleur	Circuit intégré programmable exécutant des instructions logiques pour interagir avec le monde physique.
DHT11	Capteur numérique mesurant la température et l'humidité relative de l'air.
YL-69	Capteur d'humidité du sol basé sur la conductivité électrique entre deux électrodes.
HC-SR04	Capteur à ultrasons mesurant la distance par réflexion d'onde sonore, utilisé pour détecter un niveau.
Grove / Seeed®	Modules standards préconnectés conçus pour une intégration rapide avec les cartes Arduino.
LCD 16x2	Écran à cristaux liquides à deux lignes de 16 caractères, utilisé pour afficher les mesures.
Relais	Interrupteur électromécanique permettant à l'Arduino de commander des charges plus puissantes.
Actionneur	Élément commandé par le système pour effectuer une action physique (pompe, LED, ventilateur).
Capteur	Dispositif mesurant une grandeur physique (humidité, lumière, température...).
Brochage	Description de la fonction des broches d'un composant (ex : VCC, GND, signal).
Seuil de déclenchement	Valeur programmée servant de limite pour activer ou désactiver un actionneur automatiquement.
Shunter (une alimentation)	Relier le câble de contrôle (généralement le fil vert) à la masse (fil noir) pour forcer son démarrage sans carte mère.