Livrable de projet G: **Prototype II et rétroaction des clients**

Présenté par:
Camille Kulinda-Olibrice
Asish Amudra
Alexandre Beaudette
Juan Roque Sanchez
Anthony Abou Fayssal

GNG 1503 – Génie de la conception
Présenté à: Emmanuel Bouendeu

Faculté de génie – Université d'Ottawa
Le 12 Novembre 2020

Introduction

Grâce au premier prototype, on a pu apporter des modifications nécessaires pour améliorer notre projet. Ceci nous a permis de concevoir notre deuxième prototype. Notre premier prototype était une estimation de la taille est le placement des sous-systèmes tandis que le deuxième prototype est concentré sur la conception du code Arduino nécessaire pour faire arroser la plante automatiquement. La conception du code commencera lors de ce prototypage. Nous prendrons les rétroactions du client en considération pour améliorer nos concepts. Ceci nous rapprocherons de notre projet final.

Plan d'essai pour prototypage

Pourquoi est-ce qu'on fait cet essai?

Les essais que nous avons réalisés nous ont permis de démontrer notre solution ainsi que de découvrir les avantages et désavantages de notre sous-système. Il nous permettront aussi de voir si notre concept répond aux besoins du client de façon efficace. Ces essais nous permettent aussi de déterminer si notre solution est fonctionnelle ou si elle n'est pas faisable. Pour compléter ces essais, il faudrait que notre code Arduino soit capable d'arroser les plantes automatique.

Description des objectifs de l'essai?

Quels sont les objectifs spécifiques de l'essai?

L'objectif de ces essais est de créer le sous-système le plus critique, le système d'arrosage automatique. Nous décrivons si les composantes qu'on a choisies sont compatibles avec l'Arduino ainsi que si notre plan d'action est possible avec l'utilisation de l'Arduino. Tu peux trouver notre code au bas de ce livrable.

Qu'est-ce qu'on peut apprendre ou communiquer exactement avec ce prototype?

Ce prototype sert à développer un code fonctionnel pour le système d'arrosage. Ainsi que de tester les limites des composantes choisies pour notre projet (capteurs, pompe, etc).

Quels sont les types de résultats possibles?

Grâce à ces essais nous obtiendrons des résultats expérimentaux. Malgré qu'on manque notre planteur (pot) pour concevoir notre produit final, nous développeront le code pour le système d'arrosage grâce à nos capteurs, notre pompe, l'Arduino et plusieurs autres composantes. Les données des capteurs fourniront l'information nécessaire pour arroser la plante seulement quand c'est nécessaire.

Comment est-ce que ces résultats vont aider à prendre des décisions ou choisir des concepts?

Les résultats nous permettront de déterminer si notre concept fonctionnera ou s' il devrait être modifié. On déterminera aussi si la pompe est assez puissante pour arroser les plantes et si nos capteurs sont assez sensibles pour fournir l'information sur les plantes. Sinon, il faudra modifier notre liste du coût du projet et échanger nos composantes pour celles que nous permettrons de répondre aux besoins de notre clients.

Quels sont les critères de succès ou d'échec de l'essai?

Le critère essentiel de succès serait que le fonctionnement de notre sous-système d'arrosage automatique ait un fonctionnement indépendant sans aide externe.

Qu'est-ce qu'on va faire et comment?

Décrivez le type de prototype (p. Ex cible ou compréhensif) et la raison de votre choix de ce type de prototype.

Ce prototype est compréhensif puisqu'il inclut les composantes nécessaires pour notre prototype final. La raison qu'on a choisi ce type de prototype est que le sous-système d'arrosage automatique est le sous-système le plus important donc son fonctionnement est clé au produit final et devrait être commencé le plus tôt que possible.

Décrivez le processus d'essai avec assez de détails pour permettre à quelqu'un d'autre que vous de construire et d'essayer le prototype.

La programmation sera créée à l'aide de recherches en ligne. Pour notre essai on suivra les étapes suivantes :

1. Connecte le capteur d'humidité de sol à l'aide de fils à l'Arduino et au plaque de prototypage et place le dans un contenant de terre sèche.
2. Connecter la pompe à l'aide de fils à l'aide de fils à l'Arduino et au relay et place le dans un contenant rempli d'eau.
3. Connecter la source de pouvoir (chargeur) au relay et à l'Arduino (VIN) et branche la source de pouvoir au mur.
4. Ouvrir l'Arduino IDE.
5. Téléverser le code fourni sur l'Arduino.
6. Ouvrir le moniteur en série pour observer les données fourni par le capteur
7. Après avoir observé les résultats, transférer le capteur d'humidité dans un contenant de terre humide pour voir les résultats.

Qu'est-ce qui sera mesuré?

L'efficacité et la performance du sous-système sera testé. De plus, la fonctionnalité et la mobilité des composantes de notre produit.

Qu'est ce qui sera observé et comment est-ce que ce sera documenté?

Tout d'abord nous observerons le fonctionnement des composantes de notre prototype (pompe, capteurs, etc). Ensuite nous observerons la compatibilité des composantes avec l'Arduino et leur efficacité. Pour la documentation, on a pris des captures écrans de nos codes Arduino ainsi qu'une photo de l'installation de toutes les composantes en fonctionnement.

Quels matériaux sont requis et quelle est l'estimation de leurs coûts approximatifs?

Deux capteurs (12.88CAD), pompe (14.28CAD), plaque de prototypage (gratuit), chargeur d'ordinateur portable (gratuit), relay (gratuit) et lot de fils (11.95CAD). Total environ (39.11CAD).

Quels travaux (P. Ex logiciel d'essai ou travail de construction ou de modélisation ou de recherche doit être fait?

La majorité du travail à faire est informatique donc il est nécessaire de faire de la recherche sur les bibliothèques nécessaire pour l'Arduino IDE ainsi que les composantes nécessaires pour rendre la pompe et les capteurs compatibles avec l'Arduino. De plus, il faut faire de la recherche sur le fonctionnement des capteurs d'humidité et de la pompe.

Comment est-ce que cela va se passer?

Combien de temps est-ce que l'essaie va prendre et quelles sont les dépendances (c-a-d qu'est-ce qui doit arriver avant de pouvoir faire l'essai)?

La période de test prend moins d'une journée à finir. Nos tests dépendent de la comptabilité de nos composantes avec l'Arduino. Ainsi que leur efficacité et leur fonctionnement.

Quand est-ce que les résultats sont requis? Et qu'est-ce qui dépend des résultats de cet essai dans le plan du projet?

Le plus tard que les résultats sont requis est le 12 novembre puisqu'il faut commencer la conception de notre troisième et dernier prototype de notre produit ainsi que la complétion de notre livrable. Les modifications appliquées seront clé à notre troisième prototype ainsi que notre produit final.

Rétroaction de l'essai

Aspect positif:

- Le code compile bien
- Les capteurs, le float et la pompe sont compatibles avec l'Arduino
- Les composantes du circuit fonctionnent efficacement
- Le tube est très long et nous donne beaucoup à travailler avec.

Aspect négatif:

- Longueur des fils
- Plusieurs fils partout, nettoyer le système (meilleur aspect esthétique)
- Assurer connection des fils

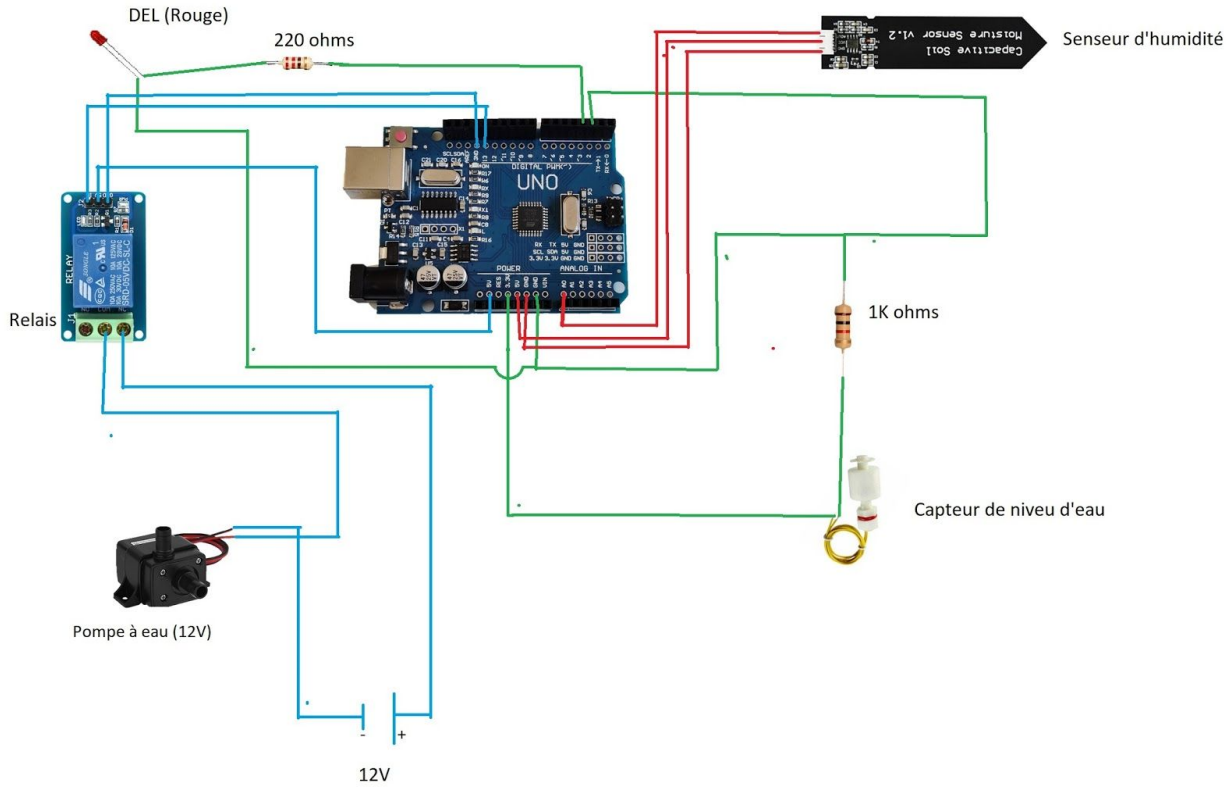
Numéro de tests	**Objectif du test (pourquoi)**	**Description du Prototype Utilisé et de la Méthode de Test de Base (Quoi)**	**Description des Résultats à Documenter et Comment ces Résultats seront Utilisés (Comment)**	**Durée Estimée du Test et Date Prévue du Début du Test (Quand)**
1	Tester la pompe	On a brancher la pompe et la relay à une source de pouvoir externe ainsi qu'à l'Arduino et on a écrit un code pour l'alimentation de la pompe	La pompe à eau est compatible avec l'Arduino. De plus, son fonctionnement est très efficace.	30 minutes
2	Tester le capteur	On a branché le capteur à l'Arduino puis à l'aide de deux échantillons de terre (1 sec et 1 mouiller) on teste son fonctionnement.	Le capteur à une bonne gamme entre humide et sec. On a pu déterminer les chiffres nécessaire pour déterminer quand la plante est assez humide et la plante est assez sec	30 minutes
3	Tester la pompe avec le capteur	On utilise l'installation des deux essais précédents pour combiner les deux code Arduino en un. Pour déterminer comment contrôler la pompe avec les données reçu du capteur	Le capteur et la pompe fonctionnent très bien ensemble. On a pu alimenter la pompe grâce à l'information obtenue par le capteur. Il est très efficace	1 heures

<u>Conclusion</u>

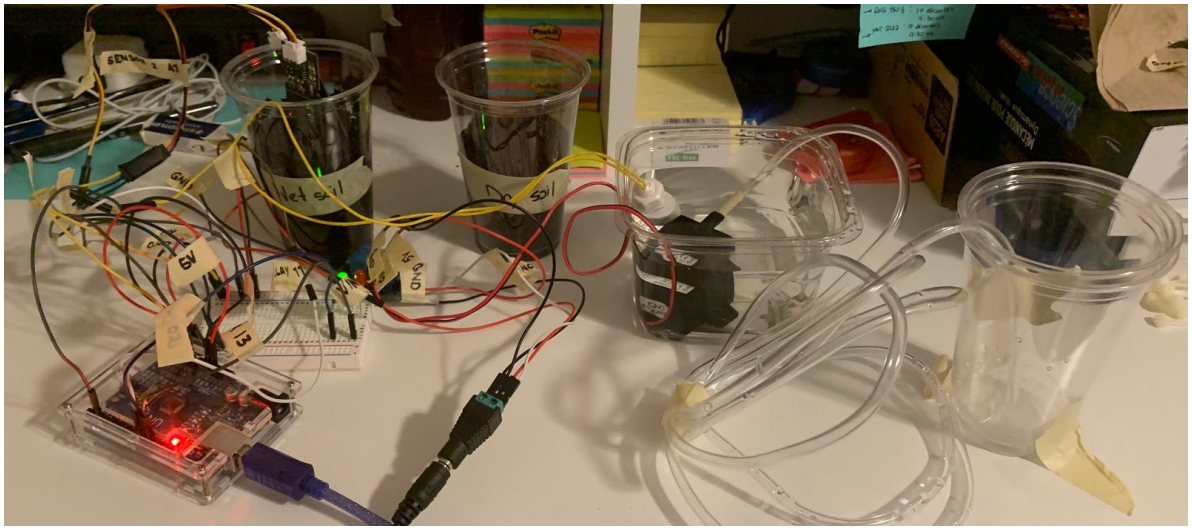
Grâce à ce prototype, on a pu déterminer que notre concept peut effectivement répondre aux besoins du client. De plus, il nous a permis de déterminer que nos composantes (pompe, capteur) sont parfaites pour notre concept et qu'elles sont compatibles avec l'Arduino. Nos tests nous ont aussi permis de créer le code pour notre sous-systèmes d'arrosage automatique. Au

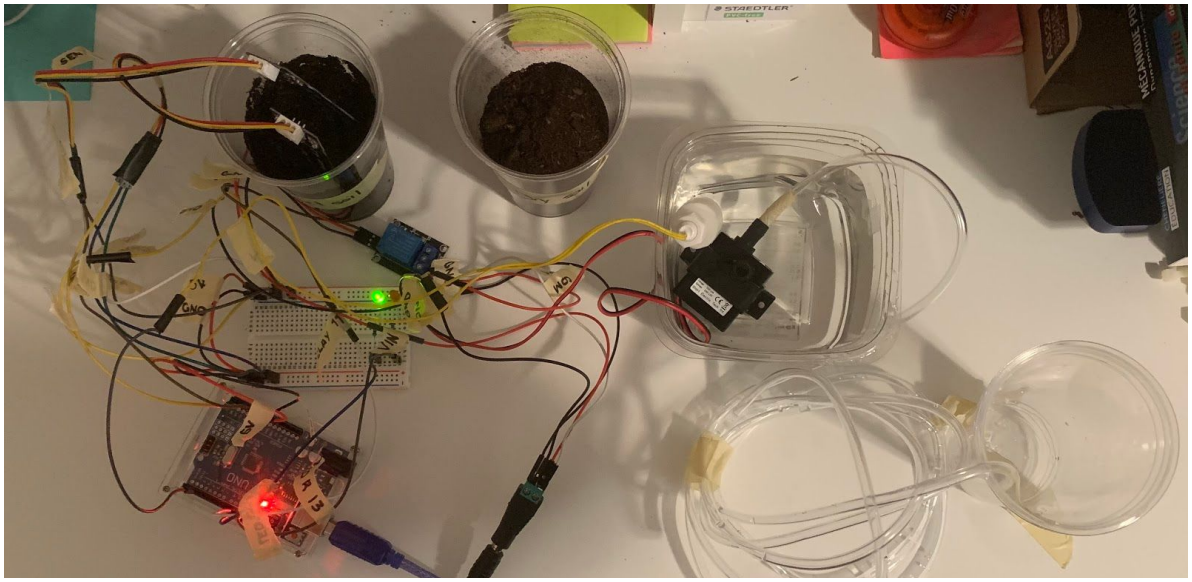
cours de la prochaine étape, on concevra notre troisième et dernier prototype. Avec toutes les modifications nécessaires pour améliorer les fonctionnement et l'efficacité on obtiendra un produit final impressionnant.

Plan du circuit:



Prototype :





Moniteur en série :

Sol sec

```
11:08:56.716 -> Sensor 1 : 348
11:08:56.716 -> Sensor 2 : 354
11:08:59.918 -> Sensor 1 : 349
11:08:59.918 -> Sensor 2 : 354
11:09:03.132 -> Sensor 1 : 348
11:09:03.132 -> Sensor 2 : 354
11:09:06.316 -> Sensor 1 : 348
11:09:06.316 -> Sensor 2 : 354
11:09:09.502 -> Sensor 1 : 347
11:09:09.502 -> Sensor 2 : 354
11:09:12.718 -> Sensor 1 : 347
11:09:12.718 -> Sensor 2 : 355
11:09:15.935 -> Sensor 1 : 347
11:09:15.935 -> Sensor 2 : 354
```

Sol humide

```
11:12:29.601 -> Sensor 1 : 195
11:12:29.601 -> Sensor 2 : 200
11:12:29.789 -> Sensor 1 : 195
11:12:29.789 -> Sensor 2 : 200
11:12:29.977 -> Sensor 1 : 196
11:12:29.977 -> Sensor 2 : 200
11:12:30.212 -> Sensor 1 : 195
11:12:30.212 -> Sensor 2 : 200
11:12:30.398 -> Sensor 1 : 195
11:12:30.398 -> Sensor 2 : 200
11:12:30.586 -> Sensor 1 : 195
11:12:30.586 -> Sensor 2 : 200
11:12:30.774 -> Sensor 1 : 195
11:12:30.774 -> Sensor 2 : 200
```

Code pour le sous-système d'arrosage automatique :

soil_sensor_water_pump_led_water_level §

```
#define Soil_Moisture_Sensor_1 A0       //Soil sensor 1 is connected to A0
#define Soil_Moisture_Sensor_2 A1       //Soil sensor 2 is connected to A1
#define Water_Level_Sensor 2            //Vertial float is connected to pin 2
int relay = 11;                         //Relay is connneced to pin 11
int soilMoisture1 = 0;                  //Variable that stores value of sensor 1
int soilMoisture2 = 0;                  //Variable that stores value of sensor 2
bool watered = false;                   //Variable that contains state of the soil
int redLED = 13;                        //Red LED is connneced to pin 13
int greenLED = 12;                      //Green LED is connneced to pin 12
int waterPumpCounter = 0;               //Counter for the # of pumps it takes to empty reservoir
int waterPumpMax = 0;                   //Stores the # of pumps it takes to empty the reservoir for future use

void setup()
{
  pinMode(redLED, OUTPUT);                          //Initialize red LED pin as output
  pinMode(greenLED, OUTPUT);                        //Initialize green LED pin as output
  pinMode(relay, OUTPUT);                           //Initialize relay pin as output
  digitalWrite(relay, LOW);                         //Set the relay to OFF
  pinMode(Soil_Moisture_Sensor_1, INPUT);           //Initialize sensor 1 as input
  pinMode(Soil_Moisture_Sensor_2, INPUT);           //Initialize sensor 2 as input
  pinMode(Water_Level_Sensor, INPUT_PULLUP);        //Initialize vertical float as input pullup (acts like a button)
  Serial.begin(9600);                               //Initialize serial monitor
}

void loop() {
  ledStatus();                                     //Runs void below names ledStatus to adjust light status
  if (digitalRead(Water_Level_Sensor) == LOW) {    //This code will run if float sensor is LOW (when reservoir is EMPTY)
```

```
    waterPumpMax = waterPumpCounter;              //Stores # of pumps required to empty reservoir for future use
    waterPumpCounter = 0;                         //Resets value of waterPumpCounter to 0 for the next watering
    ledStatus();                                  //Runs void below names ledStatus to adjust light status
  }
  ledStatus();                                    //Runs void below named ledStatus to adjust light status
  Serial.print("Sensor 1 : "); Serial.println(analogRead(Soil_Moisture_Sensor_1)); //Prints value of sensor 1 to the serial monitor
  Serial.print("Sensor 2 : "); Serial.println(analogRead(Soil_Moisture_Sensor_2)); //Prints value of sensor 2 to the serial monitor
  delay(100);                                                                      //Delay

  if (soilMoisture1 > 300 and soilMoisture2 > 300 and watered == false and digitalRead(Water_Level_Sensor) == HIGH and waterPumpCounter <= waterPumpMax) {
  //This code will run if value of sensor 1 & 2 are above 300, watered is false (plant not watered), if the water reservois is not empty and if the number of
  //pumps dose not exceed the amount used previously to prevent from running the pump without water and damaging it
    ledStatus();                                  //Runs void below named ledStatus to adjust light status
    //pumpON();                                   //Runs void below named pumpOn to turn on the water pump
    delay(3000);                                  //Runs void below named ledStatus to adjust light status
    pumpOFF();                                    //Runs void below named pumpOff to turn off the water pump
    ledStatus();                                  //Runs void below named ledStatus to adjust light status
  } else {                                        //This code runs if one or more than one of the conditions above are false
    ledStatus();                                  //Runs void below named ledStatus to adjust light status
    pumpOFF();                                    //Runs void below named pumpOff to turn off the water pump
    ledStatus();                                  //Runs void below named ledStatus to adjust light status
  }
  if (watered == true) {                          //This code runs if the plant has just been watered
    ledStatus();                                  //Runs void below named ledStatus to adjust light status
    delay(10000);                                 //10 secondes delay (to be adjusted) as a waiting period before checking if the plant has been watered enough
    ledStatus();                                  //Runs void below named ledStatus to adjust light status
    soilMoisture1 = analogRead(Soil_Moisture_Sensor_1); //Verifies the value of sensor 1
```

```
    soilMoisture2 = analogRead(Soil_Moisture_Sensor_2); //Verifies the value of sensor 2
    delay(100);                                 //Delay
    watered = false;                            //Set value of watered to false to allow plant to be watered again when needed
    ledStatus();                                //Runs void below named ledStatus to adjust light status
  } else {                                      //This code runs if the plant has not been watered
    ledStatus();                                //Runs void below named ledStatus to adjust light status
    soilMoisture1 = analogRead(Soil_Moisture_Sensor_1); //Verifies the value of sensor 1
    soilMoisture2 = analogRead(Soil_Moisture_Sensor_2); //Verifies the value of sensor 2
    delay(100);                                 //Delay
    ledStatus();                                //Runs void below named ledStatus to adjust light status
  }
}

void ledStatus () {
  if (digitalRead(Water_Level_Sensor) == LOW) { //This code runs if the reservoir is empty
    digitalWrite(redLED, HIGH);                 //Turns red LED ON
    digitalWrite(greenLED, LOW);                //Turns green LED OFF
  } else {                                      //This code runs if the reservois is not empty
    digitalWrite(greenLED, HIGH);               //Turns greeen LED ON
    digitalWrite(redLED, LOW);                  //Turns red LED OFF
  }
}

void pumpON() {                                 //This code is in charge of turning the pump ON
  ledStatus();                                  //Runs void below named ledStatus to adjust light status
  digitalWrite(relay, HIGH);                    //Turns relay ON
  watered = true;                               //Set value of watered to true to prevent plant from being watered again
```

soil_sensor_water_pump_led_water_level §

```
    ledStatus();                                //Runs void below named ledStatus to adjust light status
  }
}

void ledStatus () {
  if (digitalRead(Water_Level_Sensor) == LOW) { //This code runs if the reservoir is empty
    digitalWrite(redLED, HIGH);                 //Turns red LED ON
    digitalWrite(greenLED, LOW);                //Turns green LED OFF
  } else {                                      //This code runs if the reservois is not empty
    digitalWrite(greenLED, HIGH);               //Turns greeen LED ON
    digitalWrite(redLED, LOW);                  //Turns red LED OFF
  }
}

void pumpON() {                                 //This code is in charge of turning the pump ON
  ledStatus();                                  //Runs void below named ledStatus to adjust light status
  digitalWrite(relay, HIGH);                    //Turns relay ON
  watered = true;                               //Set value of watered to true to prevent plant from being watered again
  waterPumpCounter = waterPumpCounter + 1;      //Everytime the pump is turned the variable +1 and stores the value
  ledStatus();                                  //Runs void below named ledStatus to adjust light status
}

void pumpOFF() {                                //This code is in charge of turning the pump OFF
  ledStatus();                                  //Runs void below named ledStatus to adjust light status
  digitalWrite(relay, LOW);                     //Turns relay OFF
  ledStatus();                                  //Runs void below named ledStatus to adjust light status
}
```