LIVRABLE H: Prototype III et rétroaction du client

Abdelli, Mohamed Fadhel
Beaudoin, Nicolas
Clarke, Daniel
El Bitar, Rania
Mckay, Gabrielle

Le 26 Novembre 2020

Table des matières

Introduction:

Suite à la réalisation du prototype II, il restait encore plusieurs éléments à ajouter dont l'écran LCD et le haut-parleur avant d'avoir un produit complètement fonctionnel. Il fallait aussi corriger certaines erreurs logiques dans le code du Arduino. C'est pourquoi l'équipe de conception s'est mise à l'œuvre sur le prototype III en ayant les buts d'implémenter les composantes manquantes et de modifier le code afin de rendre le fonctionnement plus fiable et performant. Des essais ont été réalisés afin de comparer la nouvelle performance à celle précédente et pour vérifier que les nouvelles composantes fonctionnent adéquatement. Les rétroactions d'élèves de l'université d'Ottawa ont aussi été prises en compte par l'équipe de conception afin de qualifier la performance du système.

Prototype III:

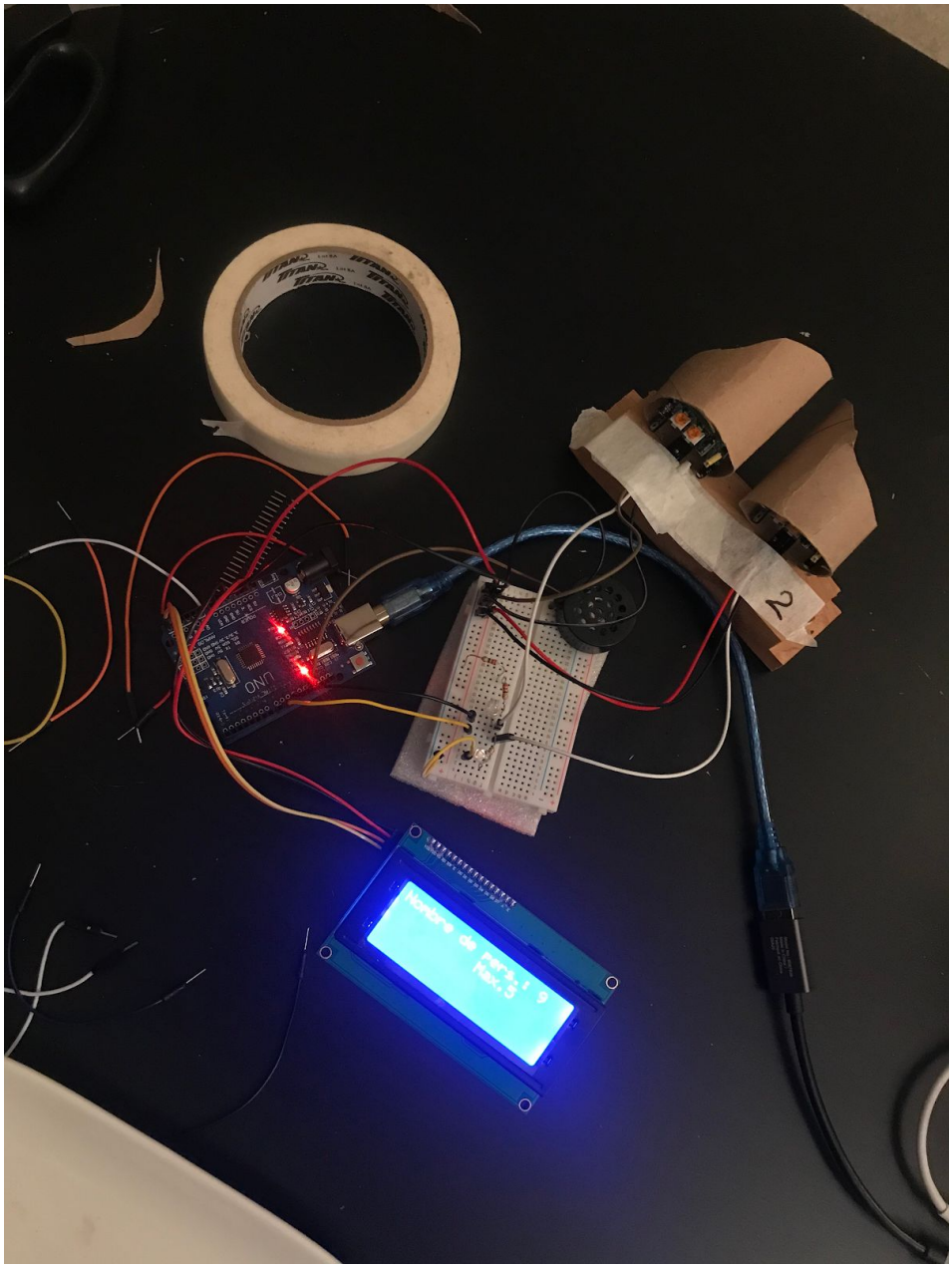
Le prototype III consiste de deux sous-tâches. La première est la partie de programmation du arduino. Celle-ci nécessite une révision en profondeur du code du prototype II afin de corriger les erreurs qui se produisaient lorsque deux objets (personnes) passaient l'un après l'autre trop rapidement. La solution trouvée était de réécrire complètement le code en utilisant une logique qui élimine le besoin d'une variable pour le temps. Cette dernière est ce que nous soupçonnons être le problème. La seconde partie est celle des composantes physiques. Il restait encore à implémenter l'écran LCD au arduino ainsi qu'un haut-parleur. Cette étape consistait uniquement à recevoir les pièces puis de les joindre au système.

Ce prototype devrait permettre à l'équipe de conception de valider le fonctionnement global du système final avant de se concentrer sur les autres besoins du client.

Lien pour simulation numérique du prototype avec écran LCD :
https://www.tinkercad.com/things/awUTmh1eScl-prototype-1-essais-tinkercad/editel?sharecode=k-5vzj44rHPOFDaORun2-qIoKYiulBnXcPZd4U4825A

Photo du prototype:

L’écran montre une présence de 9 personnes, vis à vis un max d’essais de 5 personnes. Les capteurs sont dans un 'boîtier' fait de tubes en carton, déviés pour que leurs champs ne se croisent pas. Dans ce cas, le haut-parleur sonnait l’alarme, puisque le nombre maximal de personnes était dépassé.



Essais du prototype:

N° de Test	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)	Résultat
1	Affichage de l'écran - En se servant de quelques essais de code Arduino, nous avons pu déterminer comment utiliser l'écran et son adaptateur I2C. Ce dernier nous permet d'avoir moins de fils, et donc un système plus simple et plus élégant.	Nous avons connecté l'adaptateur I2C au arduino est l'écran.	Ceci nous permet d'utiliser un écran d'affichage pour communiquer avec l'utilisateur. Il nous permet aussi d'avoir moins de fils électriques.	20 minutes, jeudi le 26 novembre 2020	Succès - L'écran s'est allumé est nous avons écrit dessus. Le tout fonctionne parfaitement.
2	Sensibilité à la lumière du capteur - Pour tester la sensibilité du capteur à la lumière. Ceci nous permet de savoir comment contrôler l'éclairage disponible aux capteurs.	Nous avons obscuré les capteurs de différentes manières afin de limiter leur champ de vue, ajusté leur sensibilité de distance et de durée d'activation, et modifié leur positions autour d'un obstacle mimiquant une porte.	Nous avons noté la performance du capteur à chaque niveau de sensibilité, de largeur de champ détection, et de position relative à l'un l'autre.	40 minutes, jeudi le 26 novembre 2020	Succès - Nous avons pu déterminer que les capteurs sont peu influencés par l'éclairage normal d'un bâtiment.

3	Test de logique des capteurs - plutôt au niveau de la programmation de ceux-ci, il s'agit de modifier le code afin de peaufiner la logique nécessaire pour que le logiciel fasse distinction entre entrée et sortie d'un individu	Nous avons essayé plusieurs différents types de logique - appel à la distinction entre le temps d'activation des capteurs, appel à la distinction entre types de combinaisons d'activation des capteurs, etc.	Nous avons utilisé ces tests pour déterminer quelle utilisation de la logique des différents cas d'activation (ex: que faire si capteur 1 est activé à la suite de capteur 2, mais que capteur 2 est à ce moment inactif) sera la méthode la plus efficace pour faire la tabulation du nombre de personnes.	90 minutes, 26 novembre 2020	Succès - nous avons pu déterminer que la logique du second programme est celle la mieux adaptée à notre prototype.
4	Test pour déterminer la largeur de la plage de détection de mouvement du senseur. - Ceci nous permet d'apprendre comment les senseurs vont détecté le mouvement à plusieurs endroits dans sa plage de détection.	L'équipe a mis une main à plusieurs endroits dans la plage de détection du capteur pour tester ces capacités.	L'équipe a conçu un schéma qui montre les positions où il peut placer une main. La performance du capteur pour chaque position a été notée.		Succès - L'équipe a pu apprendre que la plage de détection du capteur est assez large, et donc doit être limitée par un certain boîtier, mais cela ne devrait pas influencer le compte puisque les deux senseurs doivent êtres déclenchés.

Rétroaction:

En présentant le troisième prototype à des collègues de leur université, l'équipe de conception a pris en note des rétroactions constructives de la part de ces utilisateurs potentiels.

La première est en lien avec l'apparence du prototype. Après la mention qu'une apparence professionnelle est un critère important, les étudiants partageaient unanimement l'opinion que le prototype comme il est a aucunement une apparence finie. Ceci serait donc à corriger avant de finaliser le produit.

La deuxième est en lien avec l'affichage. Les étudiants ont majoritairement remarqué que l'écran LCD est assez petit et relativement difficile à lire par rapport aux autres méthodes d'affichages à travers l'université. Ceci était une crainte de la part de l'équipe de conception, mais étant donné le budget de 100$ du client, l'option d'un écran plus grand n'était pas disponible.

Finalement, ces premiers utilisateurs ont tous communiqué une certaine appréciation pour le fonctionnement automatique et fiable du système, ce qui consiste d'une amélioration par rapport au prototype II, largement dû au nouveau code implémenté dans ce prototype III.

Conclusion:

Pour conclure, le troisième prototype du processus de conception a permis de valider le fonctionnement du produit final. Des essais sur le fonctionnement des senseurs, de l'affichage et du code, accompagnés de rétroaction pertinentes d'utilisateurs potentiels, ont permis de déterminer que le prototype final est adéquat.
Il ne reste plus qu'à réorganiser le système dans un format plus esthétique afin d'obtenir le produit final.

ANNEXE:

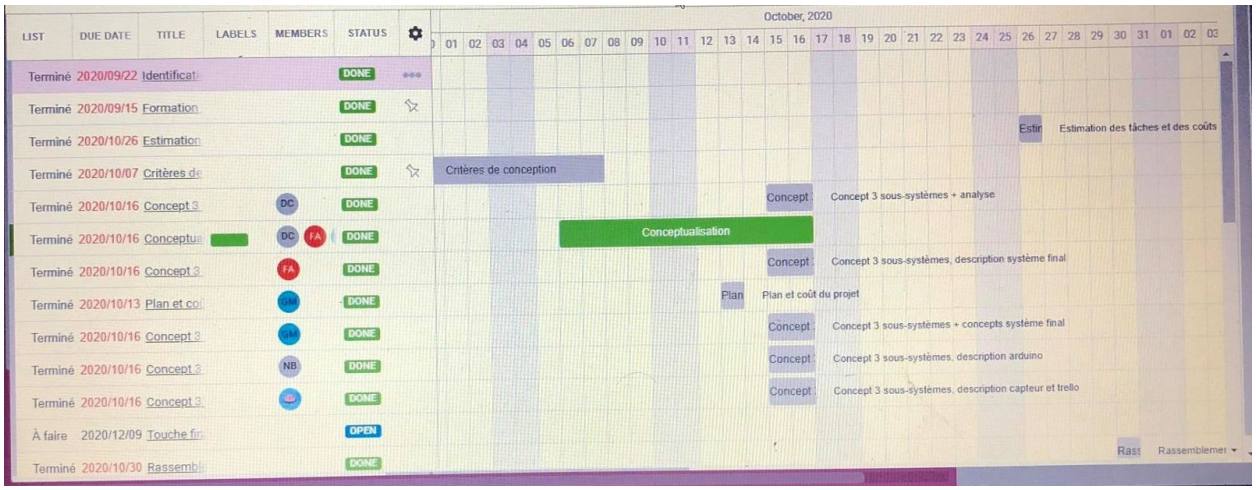
1- Copie du code arduino du prototype 3

```
//LCD setup
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4);

#define STATE_1 1
#define STATE_2 2
#define STATE_3 3
#define STATE_4 4

int currentState=0;
int lastState=0;
int count=0;
boolean sensor_1_on;
boolean sensor_2_on;
int MAX=20;          //Maximum number of people allowed in a room


void setup()
{

  Serial.begin(9600);

 //LCD screen setup

 lcd.init();
 lcd.backlight();
 lcd.print("Nombre de pers.:");
 lcd.setCursor(10,1);
 lcd.print("Max." );
 lcd.print(MAX);

 pinMode(9,INPUT);    //sensor 2
 pinMode(10,INPUT);   //sensor 1
 pinMode(13,OUTPUT);  //buzzer speaker


}
```

```
void loop()
{
 sensor_1_on = digitalRead(10) == HIGH;
 sensor_2_on = digitalRead(9) == HIGH;

 //determine current state
 if (!sensor_1_on && !sensor_2_on){
  currentState=STATE_1;
 }

 if (sensor_1_on && !sensor_2_on){
  currentState=STATE_2;
 }

 if (sensor_1_on && sensor_2_on){
  currentState=STATE_3;
 }

 if (!sensor_1_on && sensor_2_on){
  currentState=STATE_4;
 }

 //check for transition
 if(lastState==currentState){
  //do nothing
 }

 if (currentState==STATE_1 || lastState==STATE_1){
  //do nothing
 }

 if (currentState==STATE_3){
  //do nothing
 }

 if (lastState==STATE_3 && currentState==STATE_4){
  //entering
  count++;

  //Prints count on LCD and serial Monitor
  lcd.setCursor(0,1);
  lcd.print("     ");
```

```
  lcd.setCursor(0,1);
  lcd.print(count);
  Serial.println(count);
 }

 if (lastState==STATE_3 && currentState==STATE_2){
  //exiting
  count--;

  //Prints count on LCD and serial Monitor
  lcd.setCursor(0,1);
  lcd.print("     ");
  lcd.setCursor(0,1);
  lcd.print(count);
  Serial.println(count);
 }



 //Turns on alarm if MAX in exceeded
 if (count>MAX){
  tone(13, 1000);
 }

 if (count<=MAX) {
  noTone(13);
 }

 lastState=currentState;

 delay(100);

}
```

2-Diagramme de Gantt



3-Copy du code du prototype II

```
//TIMEOUT can be tweaked easily for better real world performance
//changing timeout leads to some weird stuff
#define TIMEOUT 300

//include lcd library
#include <LiquidCrystal.h>

LiquidCrystal lcd(7,6,5,4,3,2);

 //identifying variables
unsigned long sensor1TIME=0;
unsigned long sensor2TIME=0;
unsigned long timeout;
int count=0;

int MAX=20; //nombre maximum de personnes permis, à ajuster pour chaque salle
      //NOT fully implemented yet, adjust str below
void setup()
{
 pinMode(A0, OUTPUT);       //LCD contrast
 pinMode(9, INPUT_PULLUP);   //sensor 1
 pinMode(10, INPUT_PULLUP);  //sensor 2
 pinMode(8,OUTPUT);      //sensor 1 power
 pinMode(11,OUTPUT);     //sensro 2 power
 pinMode(13,OUTPUT);     //Buzzer speaker

 Serial.begin(9600);

 //LCD screen setup
 analogWrite(A0, 100); //Contrast setting for LCD
```

```
 lcd.begin(12,2);
 lcd.print("Nombre de pers.");
 lcd.setCursor(9,1);
 lcd.print("Max. 20" );
}


void loop()
{
 digitalWrite(8,LOW);
 digitalWrite(11,LOW);
 delay(10);
 digitalWrite(8,HIGH);
 digitalWrite(11,HIGH);
 lcd.setCursor(0,1);

 //counts the time at which sensor is activated
 if (digitalRead(9) == HIGH){
  sensor1TIME=millis();
  timeout= sensor1TIME + TIMEOUT;
  digitalWrite(9,LOW);
 }

 //counts the time at which sensor is activated
 if (digitalRead(10) == HIGH){
  sensor2TIME = millis();
  timeout= sensor2TIME + TIMEOUT;
  digitalWrite(10,LOW);
 }

 //determines which sensor went off first based on the timing and prints count
 if (sensor2TIME>0 && sensor1TIME > sensor2TIME ) {

  digitalWrite(9, LOW);
  digitalWrite(10,LOW);
  sensor1TIME=0;
  sensor2TIME=0;
  Serial.println("Sensor activated!+1");
  count=count+1;
  Serial.print(count);
  lcd.print("    ");
  lcd.setCursor(0,1);
  lcd.print(count);
  }

 if (sensor1TIME>0 && sensor2TIME > sensor1TIME) {

  digitalWrite(9, LOW);
  digitalWrite(10,LOW);
  sensor1TIME=0;
  sensor2TIME=0;
  Serial.println("Sensor activated!-1");
  count=count-1;
  Serial.print(count);
```

```
    lcd.print("    ");
    lcd.setCursor(0,1);
    lcd.print(count);
    }

  //if both sensors are not activated, reset time variables to 0
  if ((millis()>timeout) && (sensor1TIME ||sensor2TIME)){

      sensor1TIME=0;
      sensor2TIME=0;
      digitalWrite(9, LOW);
      digitalWrite(10,LOW);
    }

  //speaker turns on if MAX is surpassed
  if (count>MAX){
    tone(13, 1000);
  }

  if (count<=MAX) {
    noTone(13);
  }

  delay(100);//somewhat related to problem???
        //Also still not sure why a delay is needed
     // I guess for sensor speed???
}
```