LIVRABLE F: Prototype I et rétroaction du client

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

Le 5 Novembre 2020

Introduction:

Un prototype peut être utilisé pour apprendre davantage ou mieux comprendre un problème, obtenir la rétroaction des utilisateurs, réduire le risque associé à un aspect particulier d'un concept et mesurer la performance, souvent de la fonctionnalité globale.Un bon prototype est fidèle, comporte le coût total de la solution, le temps du cycle itératif et le rapport signal-bruit. Le prototype suivant est le premier conçu pour notre projet, qui comportera les composantes suivantes: Arduino Uno, capteur de mouvement et un écran d'affichage. On a donc préféré l'effectuer sur le logiciel Tinkercad.

Rétroaction du client:

Après avoir présenté le concept global au client, on a pu avoir des remarques constructives qui nous ont permi d'améliorer plusieurs points dans notre solution.
Globalement, le client a été satisfait de ce qu'on lui a présenté, mais il a quand même insisté sur certains points.
Tout d'abord, le client a exigé que l'écran d'affichage puisse être assez grand pour un affichage clair et visible pour tous. Ensuite, il a déclaré avoir une préférence pour une alimentation par électricité plutôt que par pile car cela serait plus pratique et éviterait de les changer à chaque fois. Puis enfin , il a insisté sur une bonne esthétique de la solution mais cela, sera bien évidemment à développer plus tard.

PROTOTYPE 1 : Arduino

Pour le premier prototype du projet, l'équipe de conception a décidé d'utiliser une simulation virtuelle de certaines composantes du dispositif dans le logiciel Tinkercad. Ceci permet de valider le principe logique derrière le programme de base qui sert à détecter l'entrée ou la sortie des utilisateurs. Ce prototype ne comporte pas la méthode d'affichage finale ni même une partie significative code final puisque le but de cette étape est d'abord de cibler la partie essentielle du produit (la capacité de compter les personnes passante) de façon analytique. La méthode d'affichage comporte beaucoup moins d'incertitudes et si jamais une erreur demande le remplacement ou l'annulation de ce sous-projet, le résultat final maintient tout de même sa base fonctionnelle. Les prochains prototypes devront donc inclure plus d'information sur l'affichage à l'utilisateur et sur l'emballage esthétique du produit.

Voici un lien pour accéder au prototype:

https://www.tinkercad.com/things/awUTmh1eScl-copy-of-pir-motion-sensor-with-arduino-blocks/editel?sharecode=n4N10byeP66HOtCU-9KAfjQAo8GwMgCUnJX9yRMq5aA

Plan d'essai:

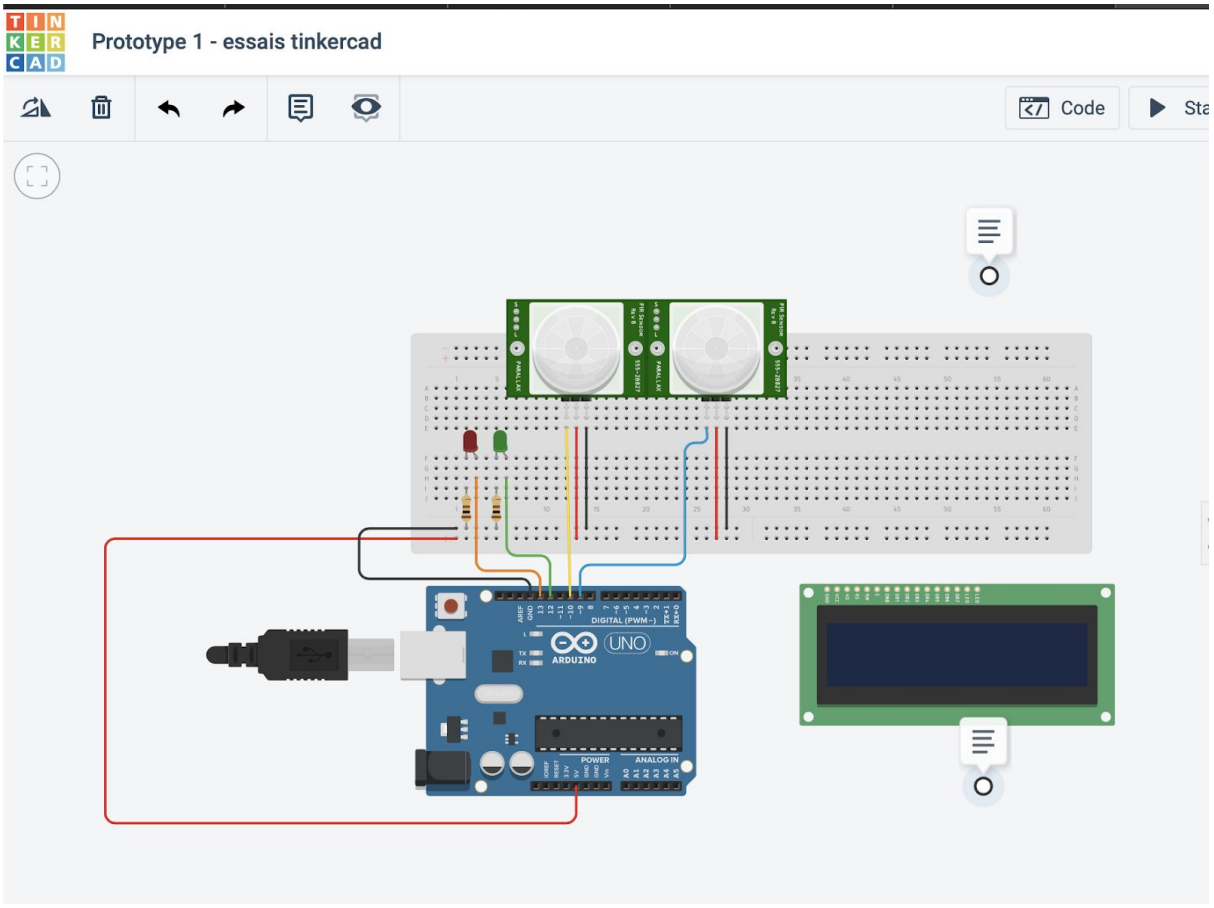
Pour ce prototype, une grande portion des essais sont en lien avec le code du arduino. Le tout n'étant qu'une conception simulé, il est difficile de réellement inspecter l'efficacité des

senseurs pour détecter l'entrée d'une personne réelle. Il est toutefois possible de déterminer les erreurs dans le code et les erreurs dans le circuit de base pour les senseurs. Ceci permet aussi de valider le concept central du produit final.

Pour l'essai, la première étape était simplement de mettre le code en marche pour s'assurer qu'il n'y a aucune erreur. Les erreurs simples corrigées, il faut mettre le code en marche à nouveau et cette fois-ci vérifier si le tout se comporte comme prévu. Le premier sous système à vérifier est le senseur. En passant la souris devant, est-ce qu'il semblent s'activer? Si non, il faut mettre fin à la simulation, tenter de le corriger puis recommencer. Par la suite il faut vérifier l'affichage DEL. Est-ce que les DELs s'allument quand elles devraient? Est-ce qu'elles s'éteignent lorsqu'elles devraient? Si non, il faut corriger cela puis recommencer à nouveau. Est-ce que l'affichage du *serial monitor* a du sens? Si non, réviser le code et réessayer de nouveau.

Quelques erreurs spécifiques qu'il faut absolument éviter sont que le système est bel et bien capable de différencier si l'activation des senseurs correspond à une entrée (+1) ou une sortie(-1) , que le compte de personnes est le bon et que les paramètres de temps peuvent être ajustés sans ruiner le reste du fonctionnement. Puisque le prototype est une simulation, la fiabilité du tout devrait être virtuellement 100%.

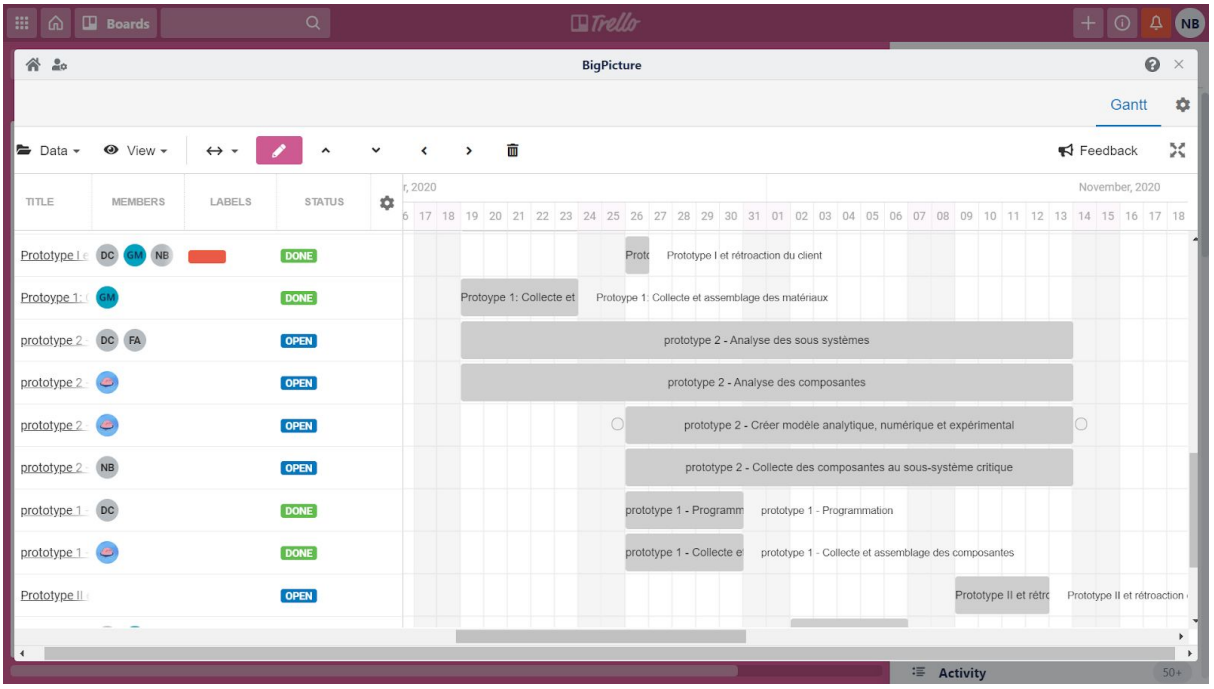
Tous ces objectif ont été atteints suite à quelques essais et erreurs.



Rétroaction du prototype:

En partageant le prototype à certains collègues étudiant, les commentaires étaient tous dans la même ligne d'idée. "C'est super, ça fonctionne, mais ça ressemble pas à grand chose d'utile" ont communiqué plusieurs utilisateurs potentiels. Ceci est considéré comme une victoire par l'équipe de conception, qui a visé justement pour cela. Le manque de détails compréhensif est possiblement le point faible, mais puisque c'est en lien avec l'objectif, ce n'est pas une rétroaction qui affecte le développement du projet.

Diagramme de Gantt:



ANNEXE:

1- Copie du code arduino du prototype 1

```
//TIMEOUT can be tweaked easily for better real world performance
#define TIMEOUT 2000

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

void setup()
{
  pinMode(9, INPUT);
  pinMode(10, INPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  Serial.begin(9600);

}

void loop()
{

  //two if statement to make sure only one LED is on at a time
  if (digitalRead(13) == HIGH) {
    digitalWrite(12, LOW);

  }
  if (digitalRead(12) == HIGH) {
    digitalWrite(13, LOW);

  }
  //counts the time at which sensor is activated
  if (digitalRead(9) == HIGH){
    sensor1TIME=millis();
    timeout= sensor1TIME + TIMEOUT;
  }
  //counts the time at which sensor is activated
  if (digitalRead(10) == HIGH){
    sensor2TIME = millis();
    timeout= sensor2TIME + TIMEOUT;
  }
  //determines which sensor went off first based on the timing and prints count
  if (sensor1TIME > sensor2TIME && sensor2TIME>0) {
```

```
    digitalWrite(12, HIGH);
    Serial.println("Sensor activated! +1");
    count=count+1;
    Serial.print(count);
    sensor1TIME=0;
    sensor2TIME=0;

          }
  if (sensor2TIME > sensor1TIME && sensor1TIME>0) {
    digitalWrite(13, HIGH);
    Serial.println("Sensor activated! -1");
    count=count-1;
    Serial.print(count);
    sensor1TIME=0;
    sensor2TIME=0;

          }
  //if sensors are not activated, LEDs turn off
  if (millis()>timeout && (sensor1TIME ||sensor2TIME)){
      digitalWrite(13, LOW);
      digitalWrite(12, LOW);
      sensor1TIME=0;
      sensor2TIME=0;
    }
  delay(100); // Delay to improve simulation performance
}
```