Fiche de Suivi d'un projet :

Titre du projet : **Contrôle d'accès**

Équipe : Karandeep, Nathan, Sarangan, Fabio

Résumé des objectifs du projet :

Dans ce projet, nous devons paramétrer un digicode à l'aide de la carte Arduino en langage Python. On doit programmer le fonctionnement pour les deux types de personnes : l'utilisateur ayant le code d'accès à 4 chiffres et l'installateur qui pourra configurer plusieurs paramètres du système de verrouillage (activation de l'alarme, temporisation d'ouverture), pour accéder à ces paramètres, un code à 6 chiffres est mis en place. Tout ces paramètres seront expliqués dans un organigramme.

Les tâches du projet :

Numéro	Désignation
1	Comprendre le fonctionnement de Arduino et le câblage
2	Réfléchir sur le problème majeur et sur la conception
3	Reproduire des modèles simples et tester
4	Création d'un organigramme
5	Création du programme du coté utilisateur pour simuler le système d'accès
6	Création du programme du côté installateur pour simuler
7	Travail sur les sorties
8	Travail sur les entrées
9	Unir les travaux distincts sur un seul script
10	Travail sur l'esthétique
11	Vérification finale

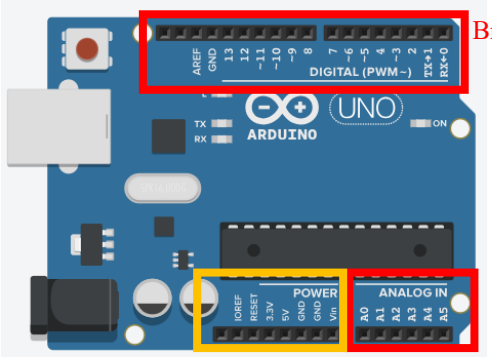
I/ Comprendre le fonctionnement de Arduino et le câblage

Tâche réalisé par : Karandeep, Nathan, Sarangan, Fabio.

Au tout début, nous avons commencer par voir comment fonctionne la carte Arduino. C'est-à-dire que qu'on a connu les capacités et limites de cette carte.

Nous avons constaté que l'Arduino Uno peut configurer des entrées et des sorties. Ces entrées et sorties sont reliés via des broches de deux types : Digitale et Analogique.
Les broches digitales sont des ports sur lesquels, la carte peut émettre et recevoir des signaux binaires donc 0 (LOW) ou 1 (HIGH). Ces ports sont ainsi les plus utilisé par les capteurs.
Les broches analogiques sont des ports sur lesquels, la carte peut émettre et recevoir des signaux électriques de tension différentes. Pour recevoir cette variation de tension dans la carte, on peut utiliser plusieurs différentes résistances pour varier la tension à recevoir.
Dans notre projet, nous avons décidé d'utiliser seulement les broches digitales, car nos composants sont installés plus convenablement.

Cette carte est programmable avec l'éditeur de texte nommé « Arduino IDE » téléchargeable sur ordinateur. Le programme doit être écrit en C++, mais on doit utilisé majoritairement le langage étudié au lycée qui est le Python.



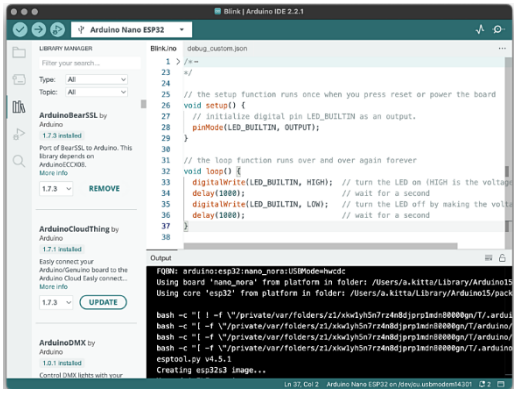
Broches de type digital

Arduino Uno
Source : TINKERCAD

Broches pour alimentation des composants connectés

Broches de type analogue

Logiciel Arduino IDE
Source : Documentation Arduino



Pour obtenir les diverses information sur la carte Arduino, nous avons étudier la documentation de cette carte sur le site officiel https://docs.arduino.cc .

II/ Réfléchir sur le problème majeur et sur la conception

Tâche réalisé par : Karandeep, Nathan, Sarangan, Fabio.

Après avoir étudier le fonctionnement de Arduino, notre groupe commence donc à réfléchir sur le projet. Mais non seulement on devait penser à la conception mais aussi à la condition imposé par les professeurs qui était de minimiser l'utilisation de C++ et de coder majoritairement en Python, ce qui était pour nous un obstacle à première vue.

Ce qui concerne la conception, on a pensé à mettre obligatoirement un digicode (clavier à code) pour permettre de aux personnes de communiquer avec la carte, des ampoules LED pour permettre à l'utilisateur et l'installateur de savoir l'état de sa demande si elle est validée ou refusée, un buzzer qui a pour rôle de d'alarme et enfin si possible un écran LCD, cela complique notre tâche car la librairie qui gère l'affichage est exécutable seulement en C++.

Afin de résoudre de franchir l'obstacle, qui est de minimiser l'utilisation de C++, nous avons commencer à chercher sur internet sur plusieurs plateformes. On a cherché sur des forums (https://forum.arduino.cc, https://stackoverflow.com), vidéos (https://www.youtube.com) et des site de tutoriels (https://realpython.com/arduino-python)/(https://www.instructables.com).

Après avoir effectué des recherches, on a obtenu deux réponse à notre problématique. Pour minimiser le C++, il existe deux librairies différentes pour manipuler la carte Arduino en Python dont on a essayé de comprendre leurs utilisation dans le code. Ces deux librairies sont « PySerial » et « PyFirmata », une librairie peut fonctionner sur la carte à la fois.

La librairie « PySerial » permet seulement de communiquer avec la carte et l'ordinateur en échanger des messages. Cela veut dire qu'avec cet outil je pourrai envoyer une donné à la carte et puis elle vérifiera, si dans son programme en C++, à la réception de cette donnée il doit faire appel à une fonction ou pas. On peut aussi le faire dans le sens inverse, c'est-à-dire que Arduino peut envoyer un message à mon ordinateur qui lui vérifiera si dans son programme Python, ce message correspond à une demande ou pas. L'avantage de cette librairie est que l'ordinateur pour recevoir les données de certains capteur dont ils ne sont pas compatibles avec Python mais avec C++. L'inconvénient de cela est que la majorité des configuration du systèmes (des entrées et sorties, les fonctions etc…) serait à écrire en C++, et Python serait là juste pour les appeler en utilisant seulement l'algorithme des conditions.

La librairie « PyFirmata » permet de configurer et manipuler tout les entrées et sorties de type digital ou analogue à partir de Python. Pour utiliser cette librairie en Python, il faut téléverser le programme en C++ fournie avec cette librairie dans la carte, une seul fois. Ensuite, tout peut se faire par le l'éditeur de texte de Python c'est-à-dire Pyscripter ou EduPython, en important la librairie et en définissant le pour de la carte. Cette librairie répond bien à nos besoins car grâce à celle-ci on peut code majoritairement en Python, ce qui est aussi son avantage. Mais l'inconvénient de cette librairie est qu'elle n'arrive pas à configurer toute sorte d'entrées et de sorties. Dans notre cas, elle est incapable les informations émis par le digicode tant que entrée, alors que la librairie « PySerial » y arrive.

Par la suite d'avoir trouver les solutions à notre problématique, nos problèmes ne sont encore pas tout à fait résolus car une librairie sur les deux ne fait que la moitié de nos attentes.

Nous avons donc remis en question notre conception car on a décidé que notre système ne serait pas basé sur une carte mais deux carte Arduino Uno. L'utilisation de deux cartes Arduino nous a permis d'exploiter les capacités des deux librairies en Python.

Nous avons donc désigné une carte que gérera que les sorties et l'autre qui aura pour but gérer les entrées.

III/ Reproduire des modèles simples et tester

Tâche réalisé par : Nathan, Sarangan, Fabio, Karandeep.

Comme on a trouvé les librairies pour pouvoir utiliser la Carte Arduino en Python, nous allons maintenant prendre certains exemple qui sont importants pour la création du projet depuis le site TINKERCARD. Les modèles crées sur TINKERCAD sont bien schématisés pour procéder au câblage mais le code est écrit en C++ d'où l'intérêt de les reproduire et de les recoder dans notre langage nécessitant. Certes les langages de programmation sont différents mais on remarque que la logique du code est la même c'est-à-dire qu'il faut configurer chaque port, mettre instruction dans une boucle infini etc…

Comme dit lors de la conception, on aura besoin des fonctionnement d'ampoule LED, un écran LCD et digicode.

Production N°1 : Ampoule LED

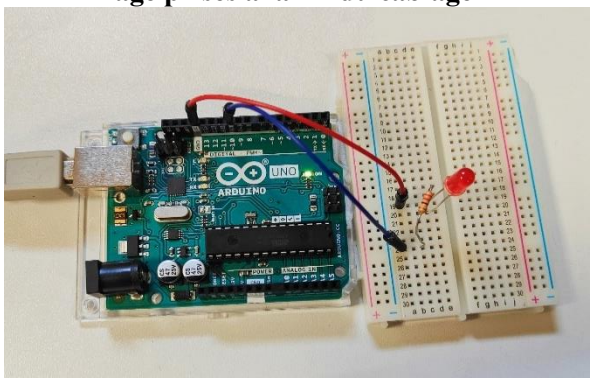
Dans ce modèle nous allons crée une ampoule qui se clignote tout les secondes infiniment. Pour ce modèle nous avons crée le programme en utilisant la librairie « PyFirmata », car avec cette librairie, on va gérer tout les sorties tels que cette ampoule. Dans cette carte Arduino, nous avons déjà téléversé en C++ le programme nécessaire pour exécuter librairie en Python.

Exemple de code écrit en C++ (pas besoin pour le fonctionnement)**:**

```
void setup() {  //fonction pour configurer tout les ports
  pinMode(13, OUTPUT); //définir le port 13 tant que sortie
}

void loop() {  //fonction créant une boucle infini
  digitalWrite(13, LOW); //écrit 0 au port 13 "LED éteint"
  delay(1000);  //attendre 1000ms = 1s
  digitalWrite(13, HIGH); //écrit 1 au port 13 "LED allume"
}
```

Image prises à la fin du câblage



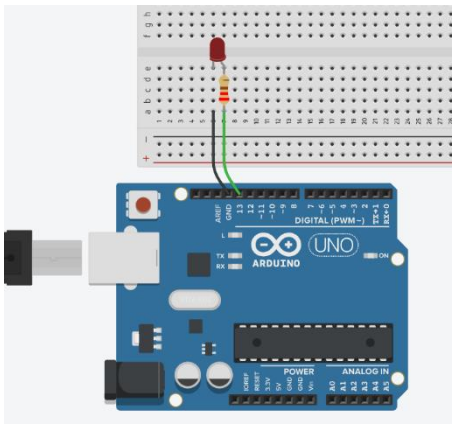
Code écrit en Python :

```
from pyfirmata import Arduino  #importation de libraire pour la carte
import time   #importation de librairie pour gérer le temps

board = Arduino('COM7')  #définir le port USB de la carte
led = board.get_pin('d:13:o')   #définir le port 13 tant que (sortie)

while True:   #Création de la boucle infini
    led.write(1)  #écrire 1 pour HIGH au port 13 pour allumer le LED
    time.sleep(1)  #prendre une pause de 1s
    led.write(0) #écrire 0 pour LOW au port 13 pour éteindre le LED
```

Schéma capturé par TINKERCAD



Production N°2 : Crée un clavier à bouton poussoir

Nous allons créer notre propre clavier à code parce que pour l'instant nous n'avons pas le digicode. A sa place nous avons mis 5 boutons poussoirs dont on a attribué un chiffre lorsqu'il est appuyé.

Pour débuter, nous avons commencer par coder un seul bouton poussoir.

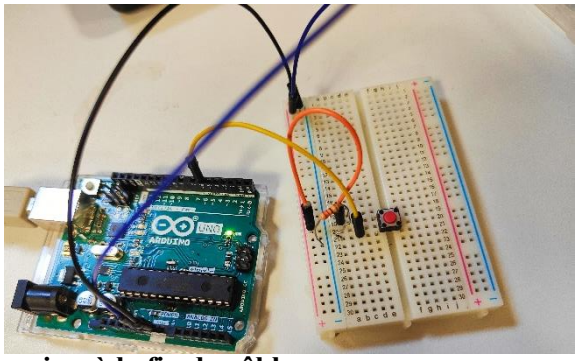
Dans ce modèle, nous avons besoin d'utiliser les langages, C++ pour savoir si l'entrée est appuyé et Python pour recevoir le statut de l'entrée par C++ et afficher « bouton appuyé » dans console.

Code écrit en C++ :

```
void setup() {
  Serial.begin(9600); //commencer la communication entre PC
  pinMode(7, INPUT_PULLUP); //définir le bouton poussoir tant
                            //que INPUT_PULLUP au port 7
}

void loop() {
  byte buttonState = digitalRead(7); //lecture du port 7
  if (buttonState == HIGH) {  //Il faut que bouton soit appuyé
      Serial.println("1"); //Il envoie 1 vers l'ordinateur
  }
}
```

Image



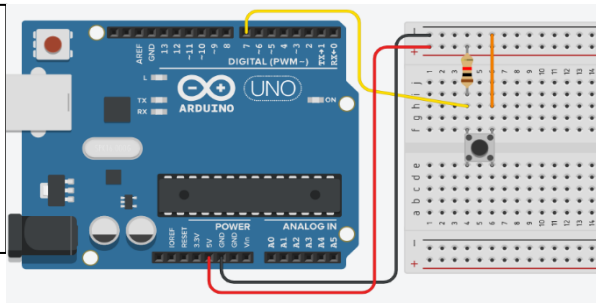
prises à la fin du câblage

Code écrit en Python :

```
import serial  #Importer la libraire PySerial
ser=serial.Serial('COM8', 9600)#Définir la carte à utiliser

response = ser.readline().decode('ascii').rstrip()
           #Définir la donnée à recevoir en la décodant
print(response) #afficher la donnée reçu
```

Schéma crée sur TINKERCAD

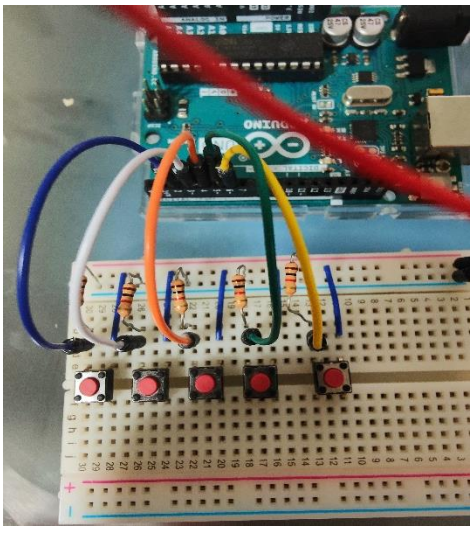


On peut voir dans le code de C++ que le port 7 est défini comme INPUT_PULLUP, ce qui permet de configurer une entrée de bouton dans la carte, ce type d'entrée n'existe pas dans la librairie « PyFirmata », voilà pourquoi on ne peut pas la utiliser pour le digicode.

Code C++ finale :

```
void setup() {
  Serial.begin(9600);
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
  pinMode(5, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);
  pinMode(7, INPUT_PULLUP);
}
void loop() {
  if (digitalRead(3)== HIGH) {Serial.println("1");}
  if (digitalRead(4)== HIGH) {Serial.println("2");}
  if (digitalRead(5)== HIGH) {Serial.println("3");}
  if (digitalRead(6)== HIGH) {Serial.println("4");}
  if (digitalRead(7)== HIGH) {Serial.println("#");}
}
```

Image du modèle finale:



Production N°3 : Affichage sur l'écran LCD

Ce qui concerne l'affichage sur l'écran LCD, sur tout exemple et modèles trouvés sur internet étaient en C++ seulement, parce que la librairie <LiquidCrystal.h> qui permet d'afficher le contenu sur l'écran est seulement en C++.

On a donc pensé vu ce que afficher est une sorte de sortie de la carte. Nous avons donc commencé à penser comment on peut intégrer le code l'affichage simple en C++ dans la librairie PyFirmata téléversé dans la carte en C++. Par la suite de quelques essaies, nous avons bien réussi à crée une fonction dans la librairie elle-même.

Vous trouverez le code de librairie avant et après la modification en partant sur le lien dans la Documentation.

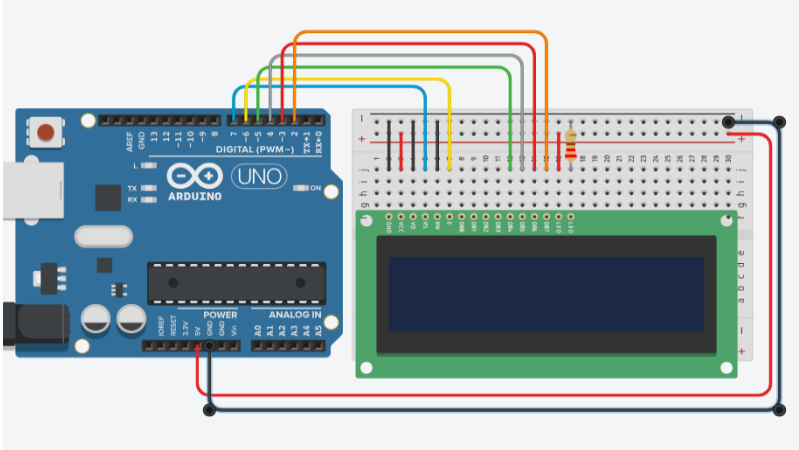
Code C++ basique pour affichage :

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

void setup() {
 lcd.begin(16, 2);
}

void loop() {
    lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Hello world");
}
```

Schéma crée sur TINKERCAD :



Les codes ajoutés en plus dans la librairie PyFirmata :

```
#include<LiquidCrystal.h> //importation de la librairie pour LCD
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);//configurer la LCD
//reste des déclarations des variables......

void stringcall(char *stringData){  //Fonction avec un paramètre
    lcd.clear();    //Effacer les caractères présents sur l'écran
    lcd.setCursor(0,0); //Remettre à la position 0
    lcd.print(stringData);  //Afficher le contenu de stringData
}
//reste des fonctions….

void setup(){
lcd.begin(16, 2);//Déclaration d'un écran 16*2
Firmata.attach(STRING_DATA,stringcall);//Attacher la fonction stringcall() à l'appel via Python
  // STRING_DATA est une déclaration d'un élément à envoyer par Python pour retrouver la fonction
//reste….
}
```

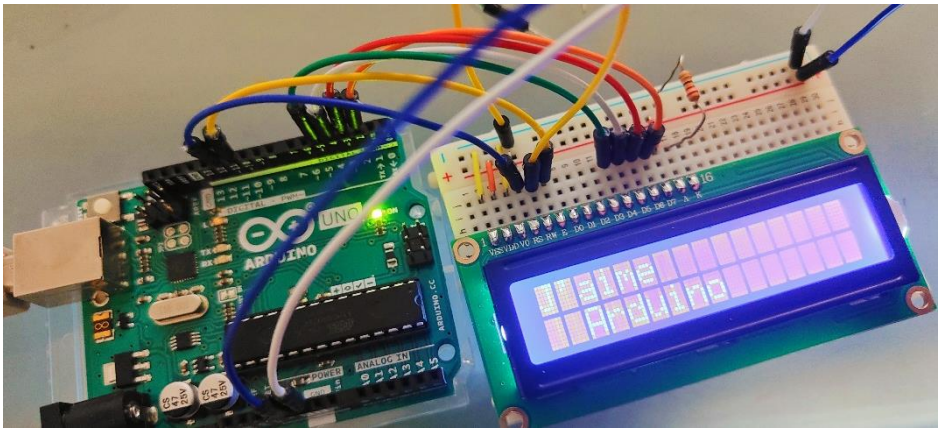
Code Python pour envoyer le contenu à afficher :

```
from pyfirmata import Arduino, util, STRING_DATA
import time

port = 'COM7'
board = Arduino(port)

response="J'aime Arduino"
board.send_sysex(STRING_DATA,util.str_to_two_byte_iter(listToString(response)))
#send_sysex est une fonction qui permet d'envoyer une donnée
#on envoie STRING_DATA, pour permettre à la librairie de trouver la fonction dont
cela correspond
```

Image après l'exécution du programme Python



Certes on utilise du C++ pour afficher, mais cela fera maintenant partie de la librairie PyFirmata. Cela veut dire que pour gérer les différentes sorties possibles on a seulement besoin de Python.

Et pour gérer l'entrée on aura belle et bien besoin du C++ et Python, pour configurer le digicode dès qu'on le reçoit.

IV/ Création d’un organigramme

Tâche réalisé par : Nathan, Fabio.

Comme nous avons bien compris les fonctions des différents composants ainsi que leurs manières de fonctionner. Maintenant il faut aussi qu’on comprenne aussi les différentes instructions et configurations qui pourront être activer, désactiver et modifier par l’installateur de ce système de contrôle d’accès.

Schéma crée avec Canva.

Les possibilités de modifications de ce système sont tous inscrites dans le cahier de charge sous forme de texte. Pour mieux comprendre les étapes, nous allons le transmettre sous forme de diagramme d’algorithme.

Il y a certaines configurations que nous ne prendrons pas en compte pare que nous n’avons pas le matériel nécessaire notamment le lecteur NFC et les badges.

V/ Création du programme du coté utilisateur en simulation

Tâche réalisé par : Nathan, Fabio.

Vu que dans notre système, il y a un coté utilisateur et un coté installateur, nous allons simuler en Python le programme pour le coté utilisateur. Il sera mis en place pour vérifier si l'utilisateur met le bon code et donnera l'accès ou refusera l'accès si le code est mauvais.

Comme c'est une simulation, nous n'allons pas utiliser la carte et les composant d'Arduino, nous allons seulement utiliser EduPython avec sa console et la fonction input. On utilise la console python pour afficher l'état de l'accès si il est ouvert ou qu'il est refusé et au lieu d'utiliser le digicode, nous allons utiliser input() dans une boucle infini pour que l'utilisateur puisse insérer un par un les caractères de son code, nous allons aussi créer une fonction qui nous servira comme un chronomètre pour donner un temps limité à l'ouverture ce qui est par défaut de 5 secondes.

Le code vous trouverez dans la lien inscrit dans la partie Documentation.

VI/ Simulation des différents paramétrages de l'installateur

Tâche réalisé par : Sarangan, Karandeep.

Nous allons maintenant créé le programme pour le côté installateur. Il sera mis en place pour modifier certaines options tels que la temporisation et gère les différents mot de passe attribué à chaque utilisateur.
Voici une liste d'options modifiables par l'installeur :

- Temporisation de verrouillage
- Alarme anti-sabotage
- Ajout de l'utilisateur
- Suppression de l'utilisateur
- Changer le code public
- Changer le code installateur.

Au lieu de faire plusieurs fonctions pour configurer, c'est-à-dire activer ou désactiver ou encore modifier certains paramètres, nous avons directement utilisé des booléens qui sont True ou False, afin de déterminer si c'est activé ou désactivé. Et pour modifier un valeur, nous avons directement insérer une boucle While afin d remplir les commandes demandés prioritairement à la demande général du code. Certes le code est seulement basé sur les conditions mais il reste assez bien compréhensible.

Pour voir le code en entier veuillez aller dans la partie Documentation parce que le code fait plus de 200 lignes.

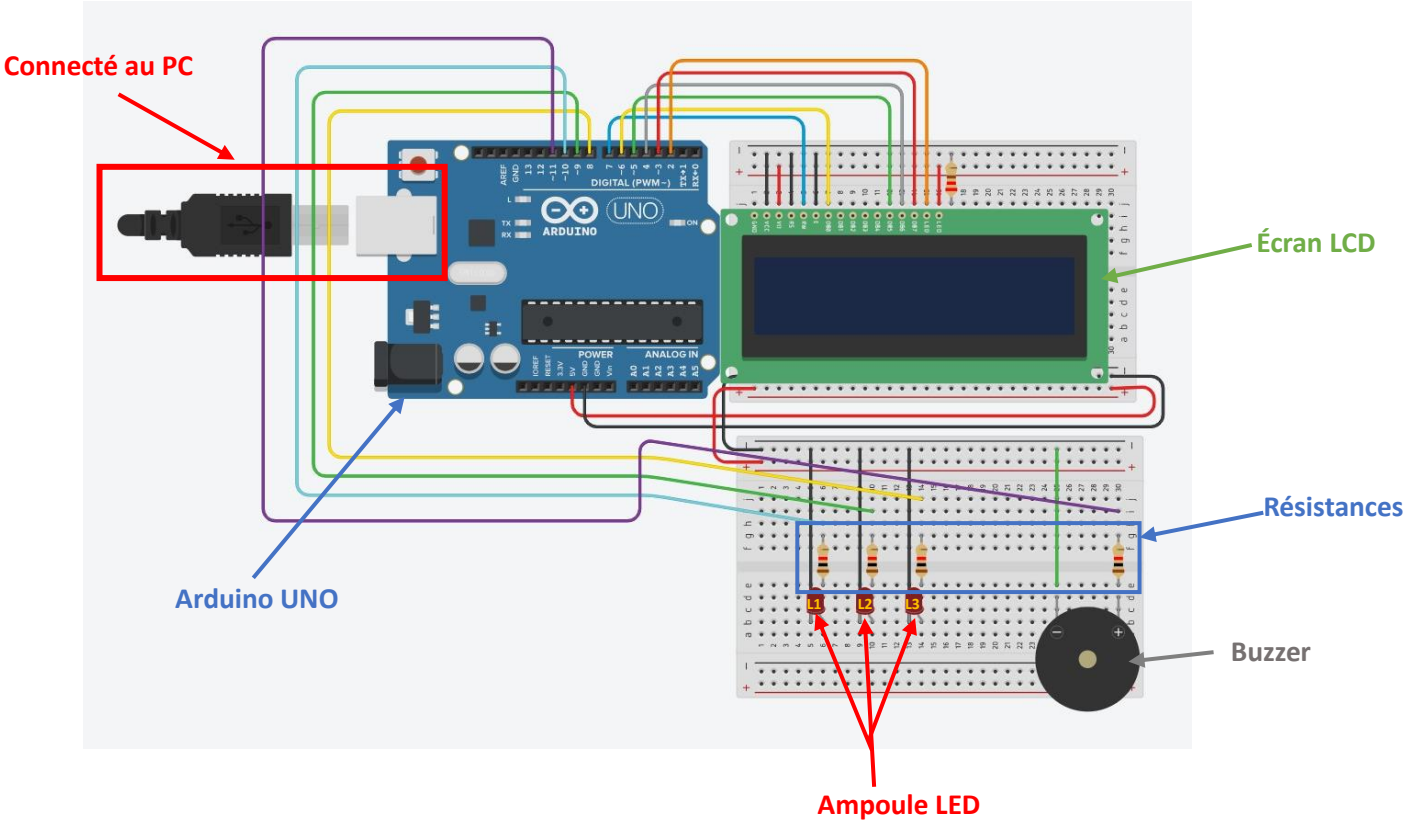
Maintenant comme nous connaissons bien les équipements et les composants nécessaires pour réaliser le système du contrôle d'accès. Nous allons maintenant commencer par assembler les composants à la carte et les faire fonctionner pour obtenir le système demandé. Pour cela, nous sommes divisés en deux groupes et puis un groupe travail sur les sorties et l'autre sur les entrées.

VII/ Travail dans le domaines des sorties

Tâche réalisé par : Nathan, Karandeep.

Notre but est connecté à la carte Arduino tout les composants de sorties cité lors de la conception. Nous allons brancher trois ampoules LED, un Buzzer et un écran LCD. Pour garder en mémoire, les branchements qu'on aurait effectués on va produire dans un premier temps un schéma de circuit sur TINKERCAD.

Voici le circuit de branchement :



Nous avons mis des résistances avant que chaque composant reçoive du courant parce que le courant émis par la carte sous forme de signal digitale est beaucoup plus que leur capacité à résister en elle-même, si nous ne mettons pas ces résistances, il y a des chance qu'un composant grille.

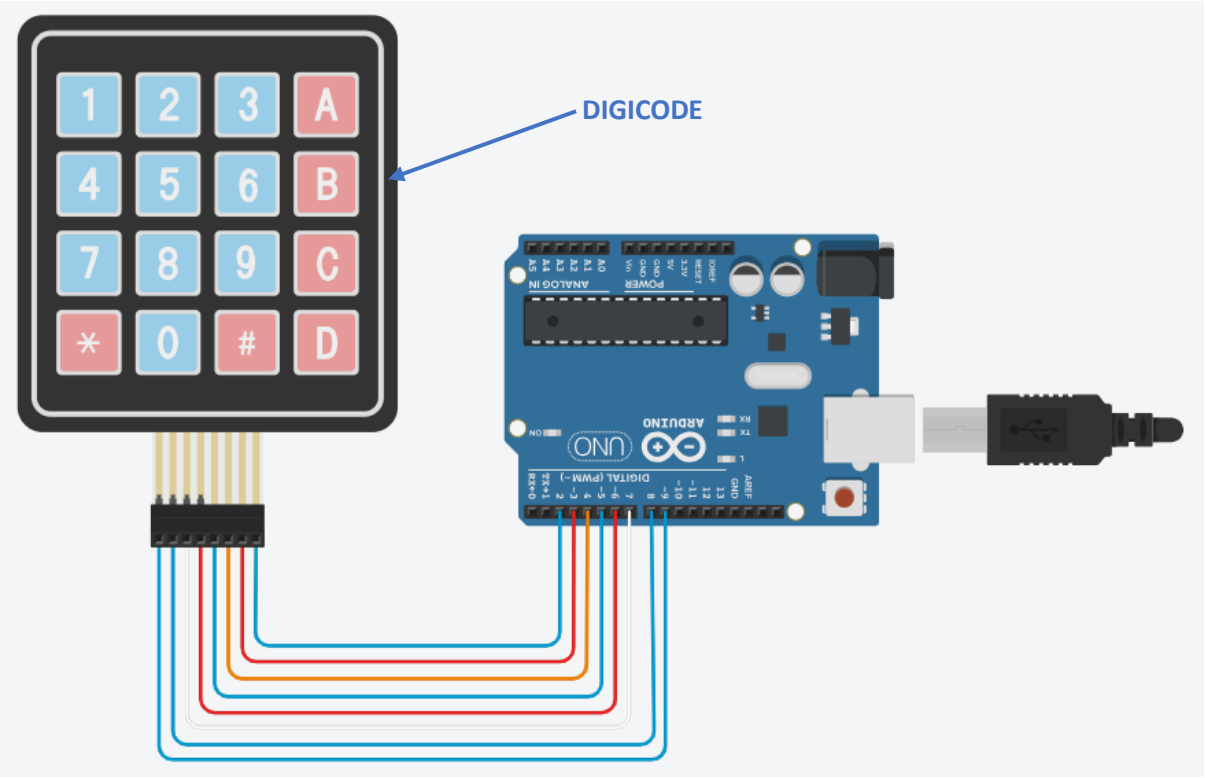
Le buzzer dans le schéma est autrement appelé Piezo, mais ce qu'on a pour ce projet est un buzzer qui fait une taille petite par rapport au Piezo.

Nous avons codé que l'écran affiche bonjour, les 3 LEDs doivent être allumés, et que le buzzer fait un son 'bip' tout les 2 secondes.

VIII/ Travail sur les entrées

Tâche réalisé par Sarangan, Fabio.

Lors de cette tâche, nous allons implanté le digicode (clavier à code) avec la carte Arduino. En plus d'un composants physique, nous allons intégré une horloge dans le programme tant qu'une entrée. On doit intégrer l'horloge parce que dans notre système y a un code public pour lequel tout utilisateur peut y utiliser, mais pendant une certaine période de temps.



Le fonctionnement du digicode :

Le Digicode est relié à Arduino avec 8 câbles. Dans le digicode les boutons sont rangés par colonnes et par lignes, celui-ci c'est une version 4x4. Donc chaque câbles représentent une colonne et une lignes. Chaque boutons lorsqu'il est appuyé crée un passage du courant comme un interrupteur.

Exemple :

Si on appuie, le bouton '2' :

Le bouton '2' se situe à la colonne 2 et au rang 1.

Donc il envoie un signal HIGH (1) dans les câbles de cette colonne et rangé à la carte Arduino. Lorsqu'elle reçoit des signaux, pour ce sont des coordonnées (x ; y). Donc il cherche à quelle valeur ces coordonnées appartiennent, après trouvé il nous renvoi le caractère appuie. Pour cela, les câblages doivent être bien définis dans le code pour éviter une mauvaise coordination.

Nous avons codé que le bouton appuyé soit affiché dans la console Python. Code est dans Documentation.

IX/ Unir les travaux distincts en un seul projet

Tâche réalisé par : Karandeep.

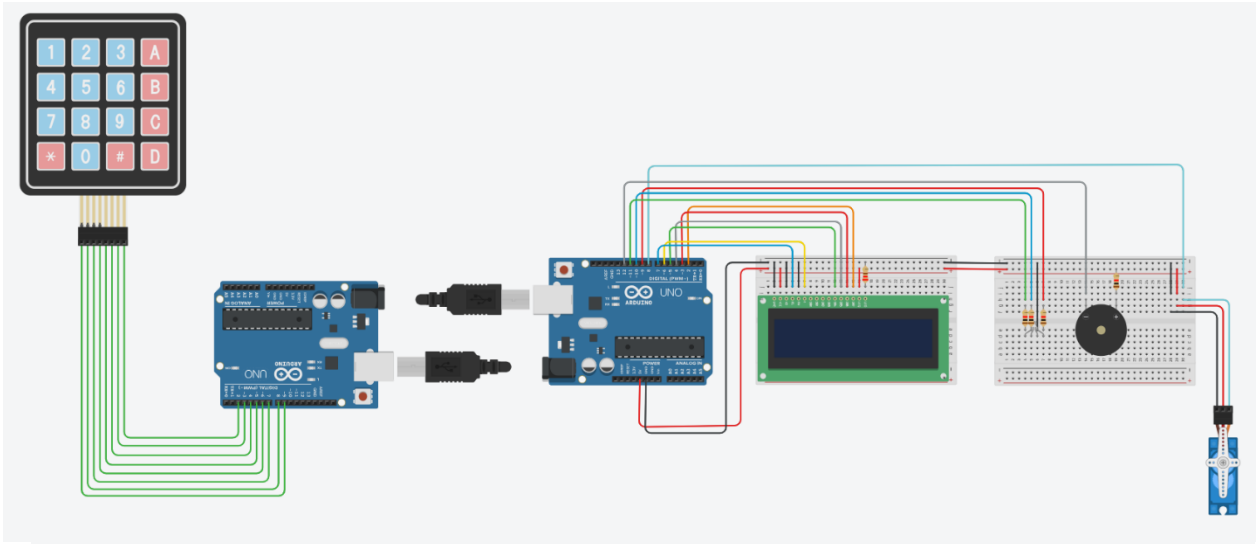
Après avoir configuré les 2 cartes Arduino en Python et C++, nous allons donc faire en sorte que ces deux cartes soient contrôlables grâce à un seul script Python qui marchera toutefois sur l'ordinateur. Lors de cette on a qu'à recrée un script Python en mettant en commun les 2 scripts des différentes cartes. En même temps que cette fusion de travaux, on va aussi se focaliser sur l'esthétique du produit.
Le code fusionné se trouve dans la partie Documentation.

X/ Travail sur l'esthétique

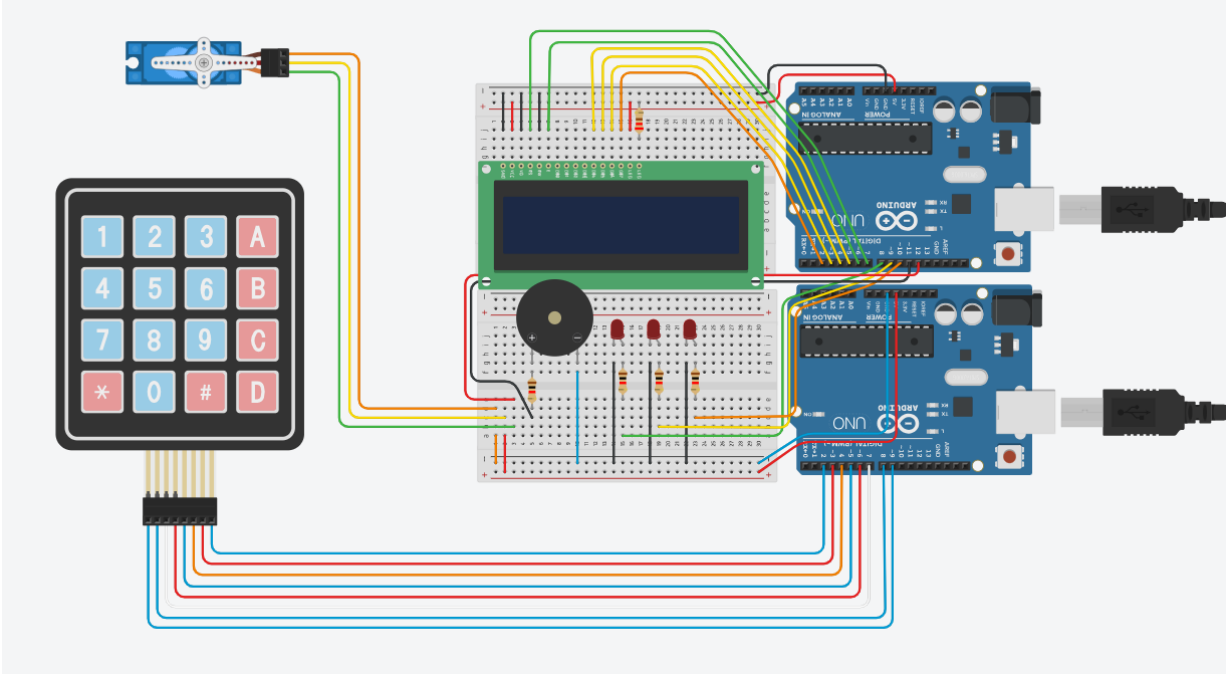
Tâche réalisé par : Nathan, Fabio, Karandeep

Lors de cette tâche, nous avons plusieurs sous-tâches. Nous devons rendre ce système d'accès assez compacte en termes de câblage. Et puis nous allons crée un coffret dans lequel ce système serait plus facile à transporter et sera plus rigide. Et fin nous allons faire un système de porte coulissante qui sera contrôler par ces cartes pour un prototype.

Câblage :
Nous sommes passées de cela :

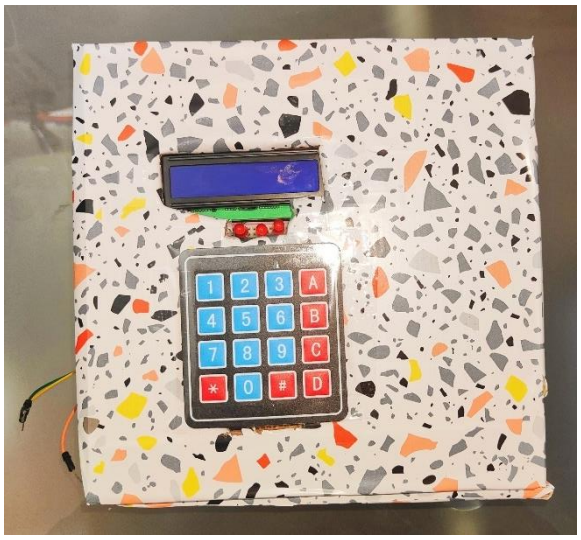


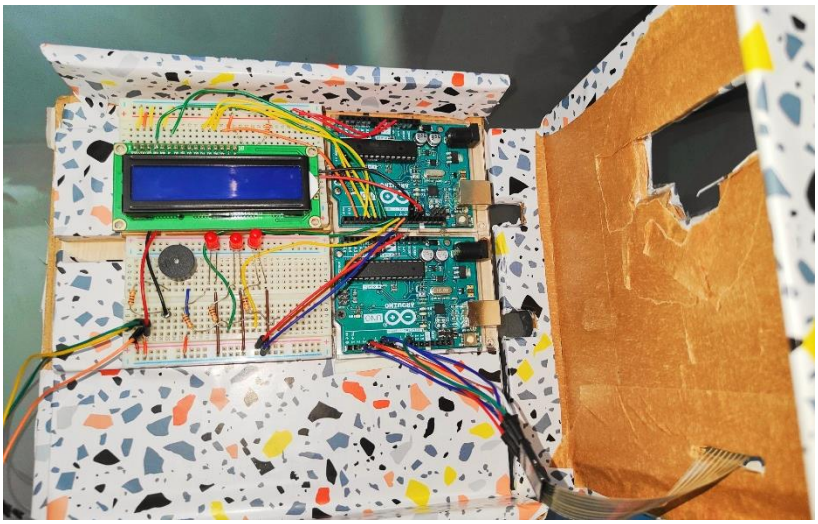
À :



Construction du coffret :

A l'aide des pièces de bois, nous avons créé un coffret en bois. Dans lequel nous avons fixée les 2 cartes après avoir bien câbler. On a mis des trous pour insérer les câbles pour brancher à l'ordinateur, puis un petit espace pour faire passer le câble du digicode et un espace carré pour l'écran LCD et les 3 ampoules LED. Enfin pour cacher tout les mauvais coupure du bois et les marquages de brouillon, on l'a recouvert par un papier autocollant.

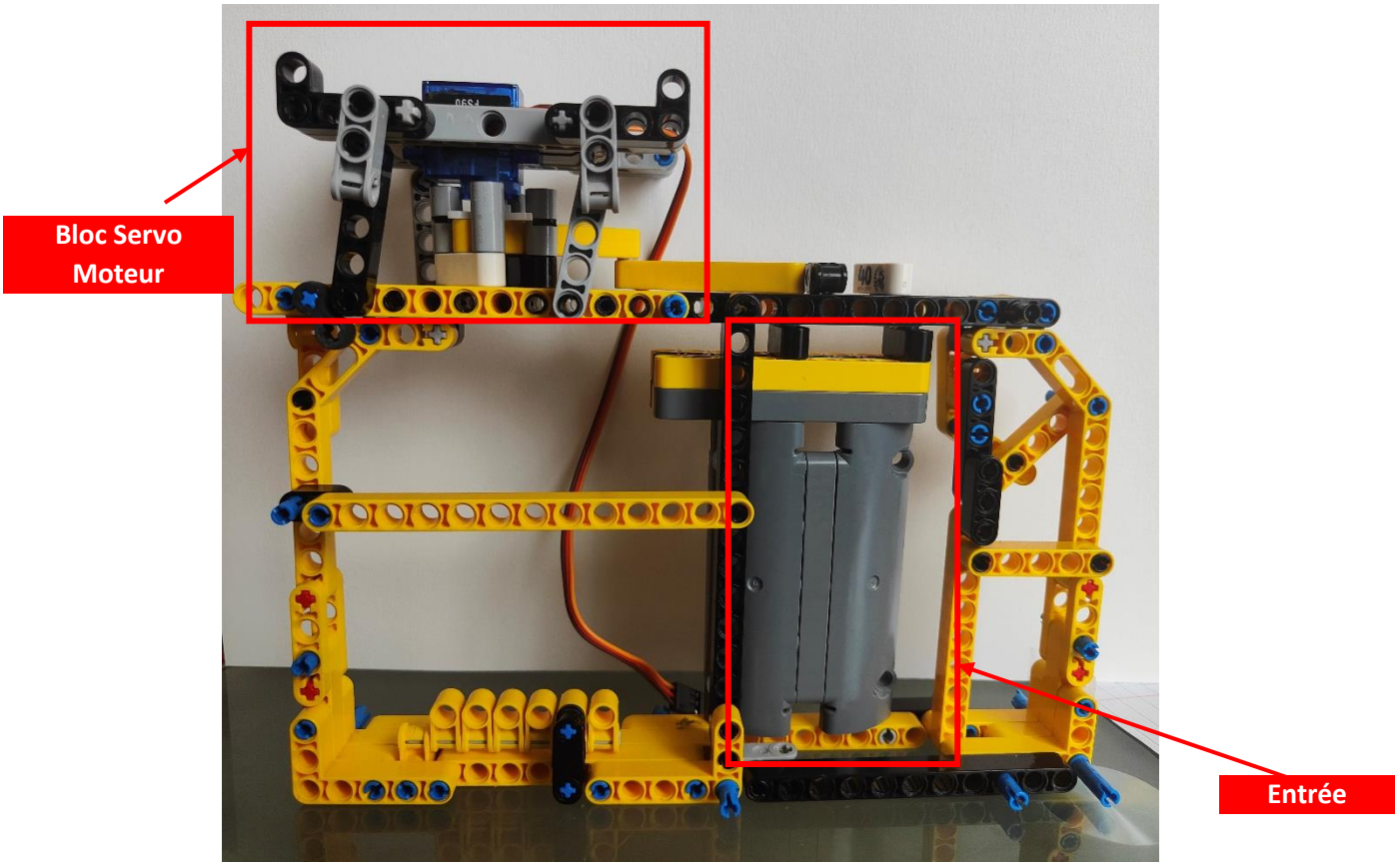




Système de porte coulissante électronique :

Après tout ces retouches sur le système de contrôle d'accès, nous allons maintenant faire un modèle de porte coulissante avec des Lego Technic.

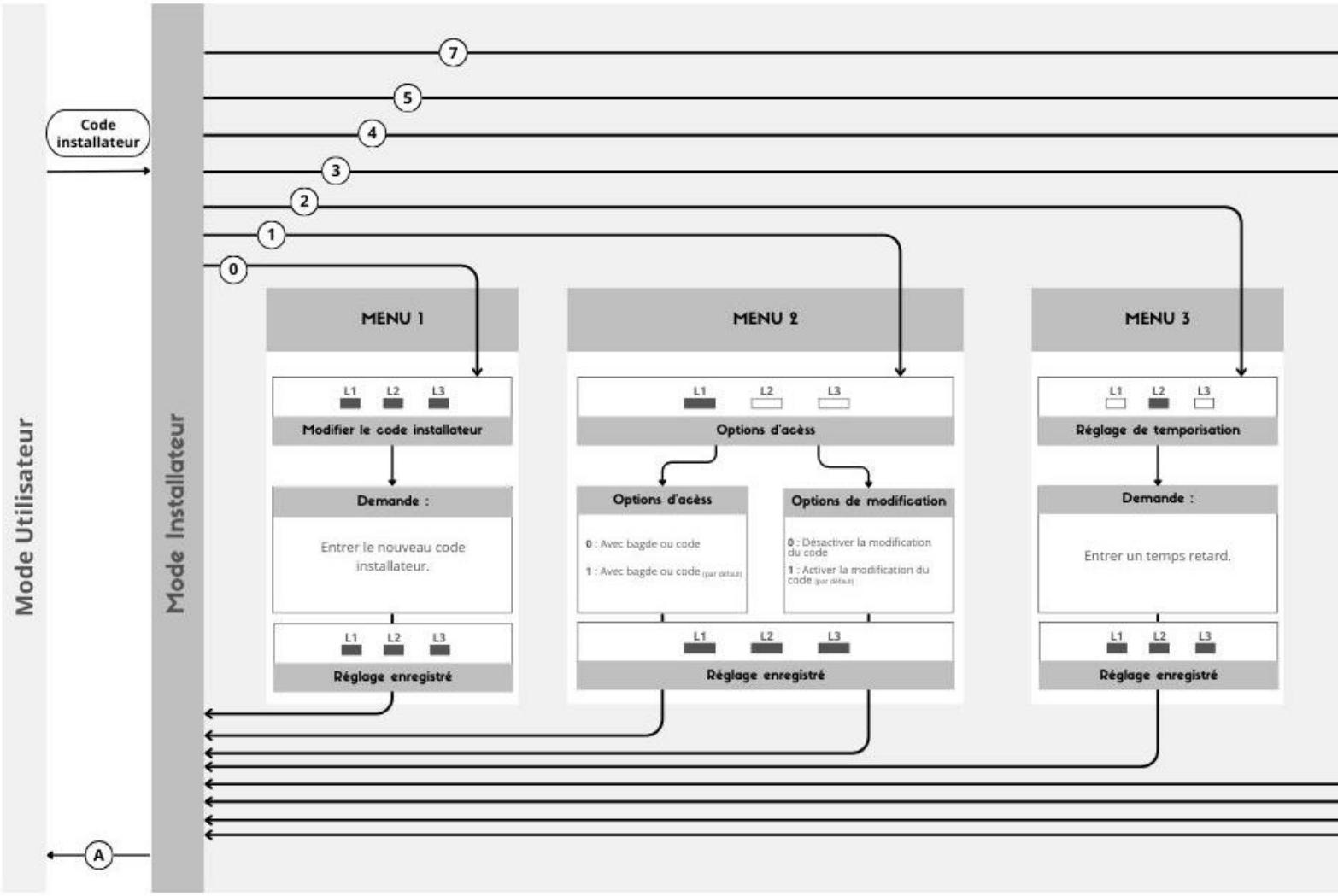
Pour créer ceci, nous sommes basés sur le glissement de la porte qui est suspendu sur une rail. Après avoir suspendu la porte sur la rail, nous allons pensé comment on peut bouger les roues ou fixations glissantes qui permettent de suspendre la porte sur la rail. Après de longue expérimentation, nous avons enfin trouver un genre de mécanisme qui permettre de tirer et pousser la porte, ce mécanisme fonctionnera à l'aide du servo moteur.



XI/ Mise en place d'un manuel pour l'installateur

Travail réalisé par Sarangan, Karandeep.

Nous avons créé cette affiche pour le but qu'elle sert à l'installateur comme un guide de configuration.



Nous l'avons créé sur le logiciel Canva.

XII/ Documentation

Tâche réalisé par Karandeep.

Tout les codes que vous souhaitez voir sont disponible sur ce lien. Ce lien vous redirigera vers GitHub, c'est un site sur lequel les développeur mettent en commun leur projet pour pouvoir travailler en équipe avec d'autres personne.

Le lien de ma répertoire est le suivant:
https://github.com/kstar937/Controle_acces

Ou scanner le code QR :

Dans cette répertoire vous trouver tout les librairies ainsi que leur code. Et aussi les images du projet et les codes que nous avons utilisés, ceux en C++ et en Python avec les commentaires pour comprendre et expliquer le code en même temps que lecture du code.