RAPPORT PROJET DE PROJET DE CONCEPTION EN INGÉNIERIE – PROJET DU COURS

INF1643-01-Architecture des ordinateurs II

Département d'informatique et d'ingénierie
Baccalauréat en génie informatique 7643
Université du Québec en Outaouais

Travail présenté par :
Waberi Wilsan WABW84320500
Séné Samuela SENS88330408
Youkna Kamyap William Dikapa YOUW15279807

Travail présenté à :
Dr. Rahmani Naim Mohamed

Date d'échéance : Lundi 7 avril 2025

Table des matières

1.Introduction

1.1 Présentation du projet

Dans le cadre du cours d'Architecture des ordinateurs II, notre projet final est la construction d'une canne intelligente pour malvoyants. Ce système permettra aux malvoyants, de se promener en toute sécurité, de même pour les piétons qui circulent autour. Ce projet inclus aussi la construction d'un système de détection dans le lieu de travail du malvoyant. Le but principal, est d'éviter toutes collisions et cela grâce à des capteurs, un système visuel et un système sonore. Pour ce faire, deux microprocesseurs, seront les pièces mère de ce projet. Deux codes Arduino et les composantes installées permettrons de détecter des obstacles sur le chemin et de détecter la présence du malvoyant sur son lieu de travail. Ce projet permet de mettre en application toute la connaissance qui nous a été donné depuis le début du semestre. Effectivement, on codera en utilisant le langage de programmation Arduino, l'application Arduino IDE avec ses fonctionnalités et l'utilisation d'un microprocesseur.

1.2 Objectifs

L'objectif de ce projet est de créer un système de navigation qui permet d'aider à la circulation d'un malvoyant. Il doit de plus, assurer la sécurité de tous les piétons, la gestion d'un système d'identification et des signaux lumineux/sonores afin de prévenir les accidents du malvoyant.

1.3 Contexte et justification

Tout d'abord, il est important de comprendre que sans ce type de système construit de la bonne manière, des accidents souvent causés par des collisions ou des systèmes obsolètes, peuvent entraîner des pertes humaines et matérielles importantes.

Ce projet permettra donc, d'améliorer la sécurité pour les piétons et le malvoyant à l'aide des signaux lumineux et sonores. Il permettra aussi d'optimiser l'efficacité en réduisant les délais non

nécessaires et les difficultés dans la circulation d'un malvoyant. Puis, il permettra de réduire la dépendance humaine et ainsi réduire les erreurs humaines.

2.Description détaillée du projet

2.1 Spécifications du système

Cette section va être séparée en 3 différentes sous sections. Nous allons commencer en abordant les fonctionnalités, puis les exigences et finalement, les contraintes.

Pour les fonctionnalités:

- Détecter des obstacles ;
- Système sonore qui informe le malvoyant (le son doit être différent selon la proximité de l'obstacle) ;
- Système visuel pour informer si la canne est allumée ou pas Détecteur de la canne (pour la porte) ;
- Afficher des messages;
- Système qui permet de lire une carte;
- Portail qui s'ouvre et qui se ferme;
- Émettre un son pour signaler au malvoyant qu'il peut passer lorsque la barrière est ouverte;

Maintenant que les fonctionnalités ont été établis, passons aux exigences du système. Notre professeur nous a donner une liste des exigences matérielle, voici les options que nous avons retenus:

- Carte de développement à microcontrôleur (Arduino);
- Capteurs ultrasoniques et infrarouges;
- Bouton-poussoir;
- Écran à cristaux liquides (LCD);
- Carte à puce;
- Buzzer;

- LED ;
- Servomoteur;

Les exigences logicielles qu'on a retenues sont les suivantes:

- Programmation en Arduino

Puis après les exigences, nous pouvons voir les différentes contraintes que nous avions à suivre pour s'assurer d'avoir un bon système.

Notre première contrainte est le temps, en effet, il faut que les obstacles sur le chemin du malvoyant soit détecté à temps, il faut aussi que les capteurs soit capable de transmettre l'information dans un délai spécifique. Il faut que tout soit rapides et précis selon les événements. Puis le système doit pouvoir afficher son message d'entrée au bon moment, ainsi que la barrière d'entrée qui se lève et qui se baisse à temps.

La prochaine contrainte est la fiabilité. Effectivement, nous devons être sûr que si par exemple un des capteurs arrête de fonctionner, on est un système de redondance permettant à un autre capteur de détecter indépendamment. Par la suite, il était important que notre système assure une certaine sécurité pour les usagers de la canne et les piétons. Cette section donne une vision claire et complète des attentes et des spécifications du système conçus.

2.2 Description fonctionnelle

Dans le cadre de la conception d'une canne intelligente destinée à améliorer la mobilité et la sécurité des malvoyants, il est essentiel de détailler les mécanismes fonctionnels du système. Cette section met en lumière les différentes interactions entre les composants matériels et logiciels qui rendent le dispositif efficace et fiable. En plus d'offrir un soutien en temps réel aux utilisateurs malvoyants grâce à des capteurs, un système sonore et lumineux, la canne intelligente intègre également des fonctionnalités permettant de sécuriser l'accès à leur lieu de travail via un système de contrôle d'entrée. Cette section présente de manière complète et structurée le fonctionnement

de ces systèmes, en expliquant comment chaque élément joue un rôle crucial pour garantir la sécurité, l'autonomie et la fluidité des déplacements.

Lorsqu'un obstacle est détecté, le capteur infrarouge intégré dans la canne intelligente émet un signal simultanément vers les systèmes lumineux et sonores pour alerter le malvoyant. L'utilisateur peut activer ou désactiver la canne en appuyant sur un bouton dédié : lorsque la canne est en marche, une LED verte s'allume, tandis que lorsque la canne est désactivée, la LED verte s'éteint et une LED rouge s'active pour indiquer son état. La détection d'un obstacle par le capteur infrarouge déclenche également l'activation du BUZZER, avec une intensité sonore proportionnelle à la distance de l'obstacle identifié.

En parallèle, le système de contrôle de la porte repose sur des interactions précises entre différents composants : lorsque le lecteur de carte détecte un signal valide, le BUZZER émet un son pour confirmer la correspondance. Si le capteur de la porte identifie la proximité de la canne, le système devient opérationnel et un message informatif est généré. Une fois l'identification validée, un servomoteur est activé pour ouvrir la porte, permettant au malvoyant de passer en toute sécurité. Ce fonctionnement garantit un système fiable et réactif aux différents besoins des utilisateurs.

2.3 Digramme de blocs fonctionnels

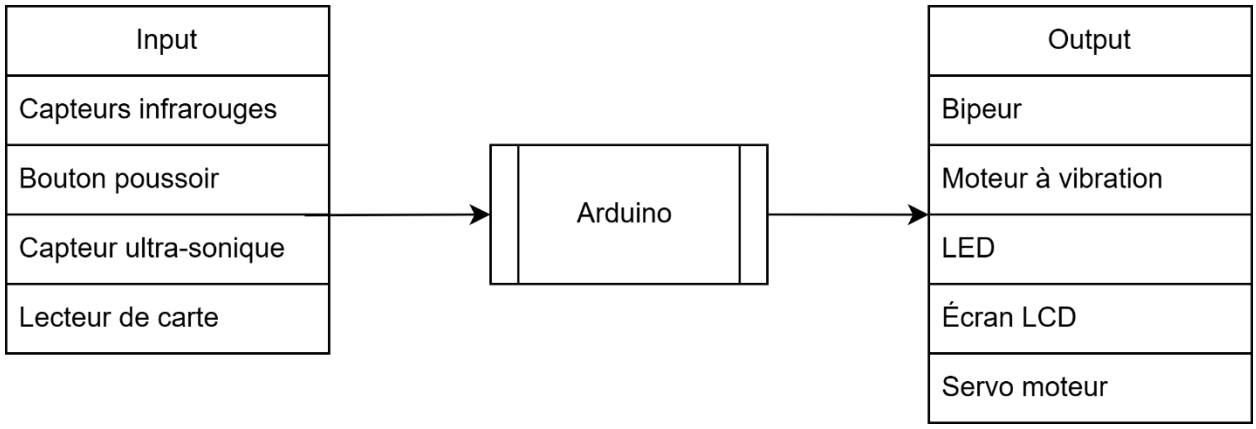


Figure 1: diagramme de blocs fonctionnels

Le diagramme de blocs fonctionnels illustre les interactions entre les différents composants, en mettant en évidence leur rôle dans le système global. Les capteurs, tels que le capteur infrarouge, le capteur ultrasonique et le lecteur de carte, agissent comme des dispositifs d'entrée en transmettant des informations essentielles au microcontrôleur Arduino. Celui-ci, en tant qu'unité centrale, traite ces données et commande les différents actuateurs, notamment les LEDs, le buzzer, l'écran LCD, et le servomoteur. Ces composants produisent des signaux lumineux et sonores ou activent des mécanismes spécifiques, tels que l'ouverture de la porte, pour garantir un fonctionnement fluide et sécuritaire. Ce diagramme met en lumière la manière dont chaque composant contribue à la coordination et à l'efficacité du système, en assurant une réactivité optimale face aux besoins des utilisateurs malvoyants. En définissant clairement les flux d'information et les interactions fonctionnelles, il constitue un outil indispensable pour la conception et l'analyse du dispositif.

2.4 Machine d’état algorithmique

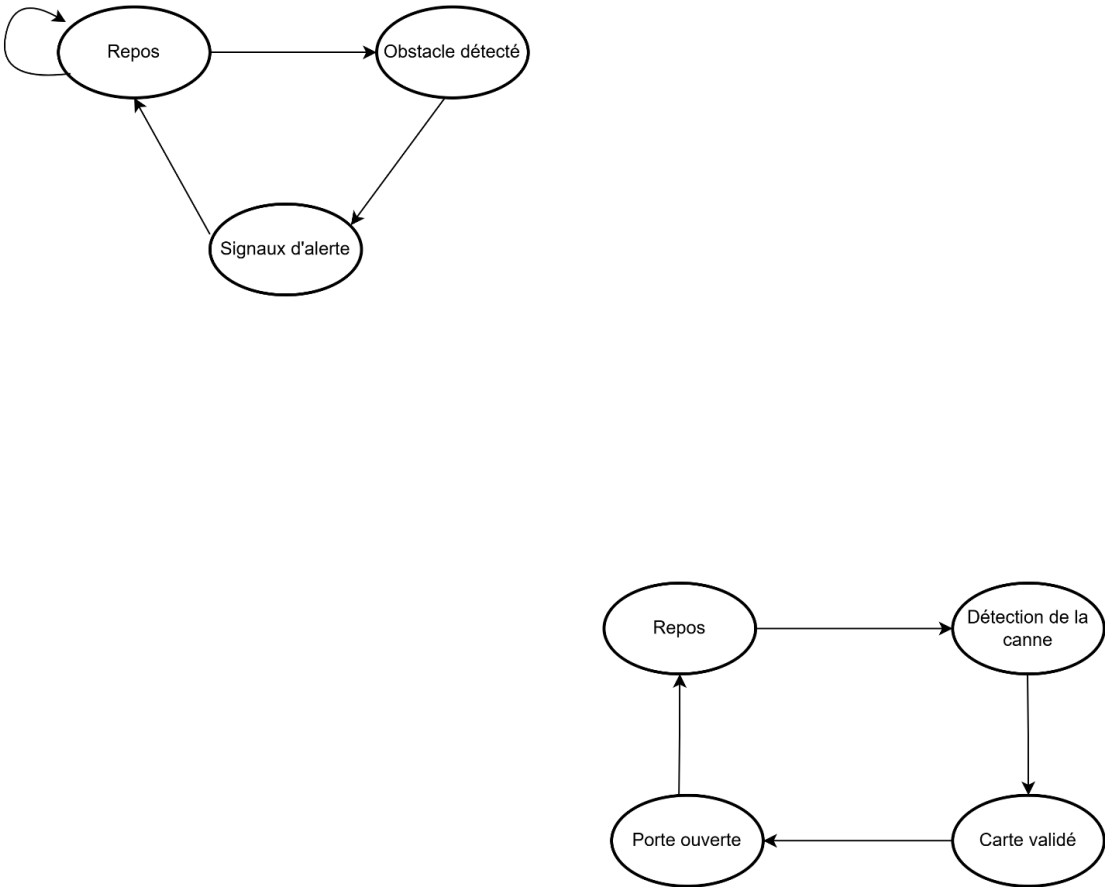


Figure 2: Machines à états finis pour la porte et la canne

La machine à états finis (Finite State Machine) définit les différents états du système, les conditions qui permettent de passer d’un état à un autre, ainsi que les interactions entre ces états pour garantir une coordination fluide et efficace.

FMS pour la gestion des obstacles :

Dans ce module, trois états principaux ont été identifiés :

1. Repos : Le système reste inactif lorsqu'aucun obstacle n'est détecté, économisant ainsi de l'énergie.

2. Obstacle détecté : Lorsque le capteur infrarouge identifie un obstacle sur le chemin, le système passe immédiatement à cet état.
3. Signaux d'alerte : Cet état se déclenche pour alerter le malvoyant grâce à des signaux lumineux et sonores adaptés (intensité variable selon la distance de l'obstacle).

Les transitions entre ces états assurent une réactivité en temps réel. Par exemple, le passage de l'état "Repos" à "Obstacle détecté" se fait dès qu'un obstacle est détecté par le capteur, puis l'état "Signaux d'alerte" est activé pour prévenir l'utilisateur. Une fois l'alerte traitée et l'obstacle contourné, le système revient à l'état "Repos".

FMS pour le contrôle d'accès via la porte :

Pour le contrôle d'accès sécurisé, quatre états principaux sont définis :

1. Repos : Dans cet état, le système de détection est inactif.
2. Détection de la canne : Lorsqu'un capteur détecte la présence de la canne à proximité de la porte, le système s'active.
3. Carte validée : Une fois que le lecteur de carte reconnaît une correspondance valide, le système passe à cet état pour valider l'accès.
4. Porte ouverte : Après validation, le servomoteur est activé, ouvrant la porte pour permettre à l'utilisateur de passer en toute sécurité.

Les liens entre ces états permettent une transition fluide et synchronisée. Par exemple, après détection de la canne, une vérification de la carte est effectuée pour confirmer l'identification. Si celle-ci est validée, un message sonore est émis, et la porte s'ouvre en activant le servomoteur. Une fois l'utilisateur passé, le système retourne à l'état "Repos".

2.5 Diagramme de flux

Le diagramme de flux est un outil fondamental pour comprendre et représenter les processus impliqués dans le fonctionnement du système de la canne intelligente et de son mécanisme de contrôle d'accès. Il illustre les séquences d'actions et les décisions nécessaires pour garantir une utilisation efficace et sécurisée. Cette section se divise en deux parties : la première est consacrée

au flux de fonctionnement pour la gestion de la porte, et la seconde détaille le processus lié à la navigation avec la canne.

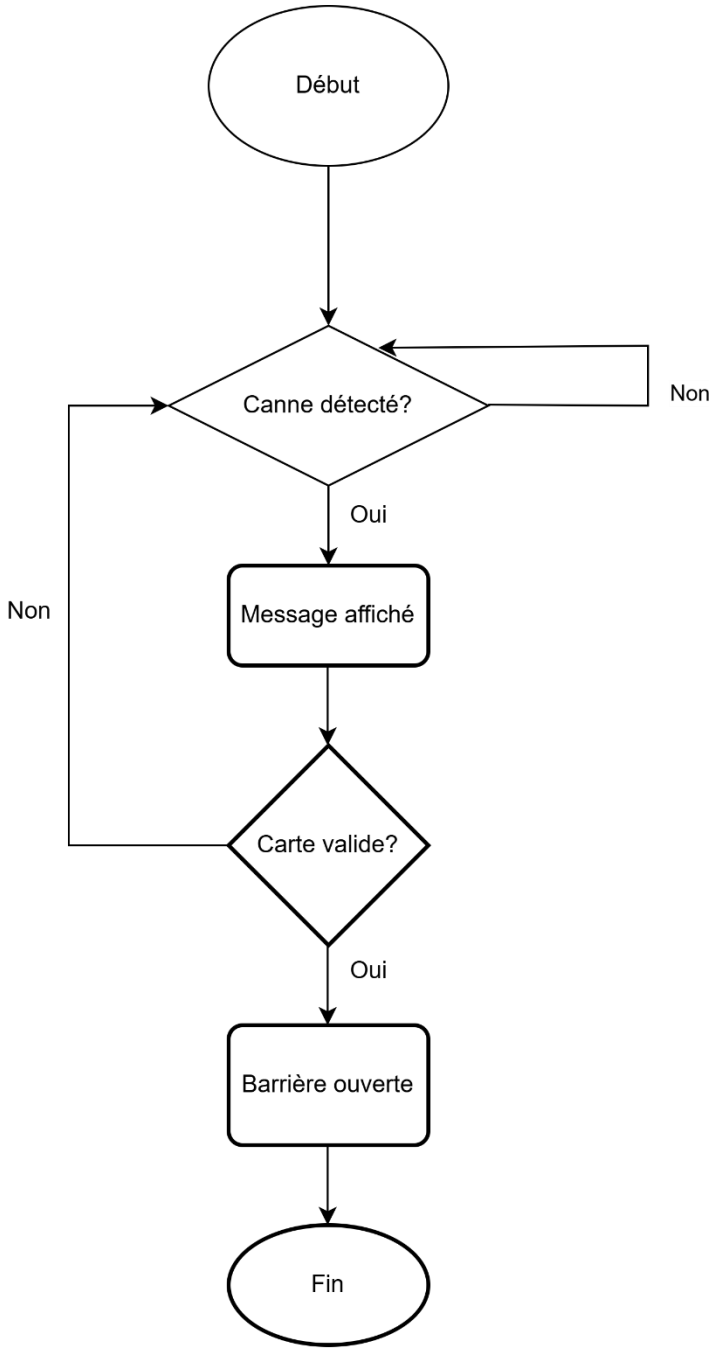


Figure 3 Diagramme de flux pour la porte

Ce premier diagramme de flux se concentre sur le système de contrôle d'accès, qui vise à sécuriser et faciliter l'entrée des malvoyants dans des zones spécifiques, dans ce cas, à son lieu de travail. Le processus débute avec la détection de la canne par un capteur à proximité de la porte. Une fois la présence confirmée, le système affiche un message informatif pour signaler son activation. Ensuite, un lecteur de carte effectue une vérification de l'identité via une carte d'accès. Si la correspondance est validée, un signal sonore est émis pour notifier l'utilisateur, et le servomoteur est activé pour ouvrir la porte. Une fois l'utilisateur passé, le système retourne à l'état initial de veille. En cas d'échec de la validation de la carte ou d'absence de la canne, aucune action supplémentaire n'est effectuée, et le système conserve son état inactif.

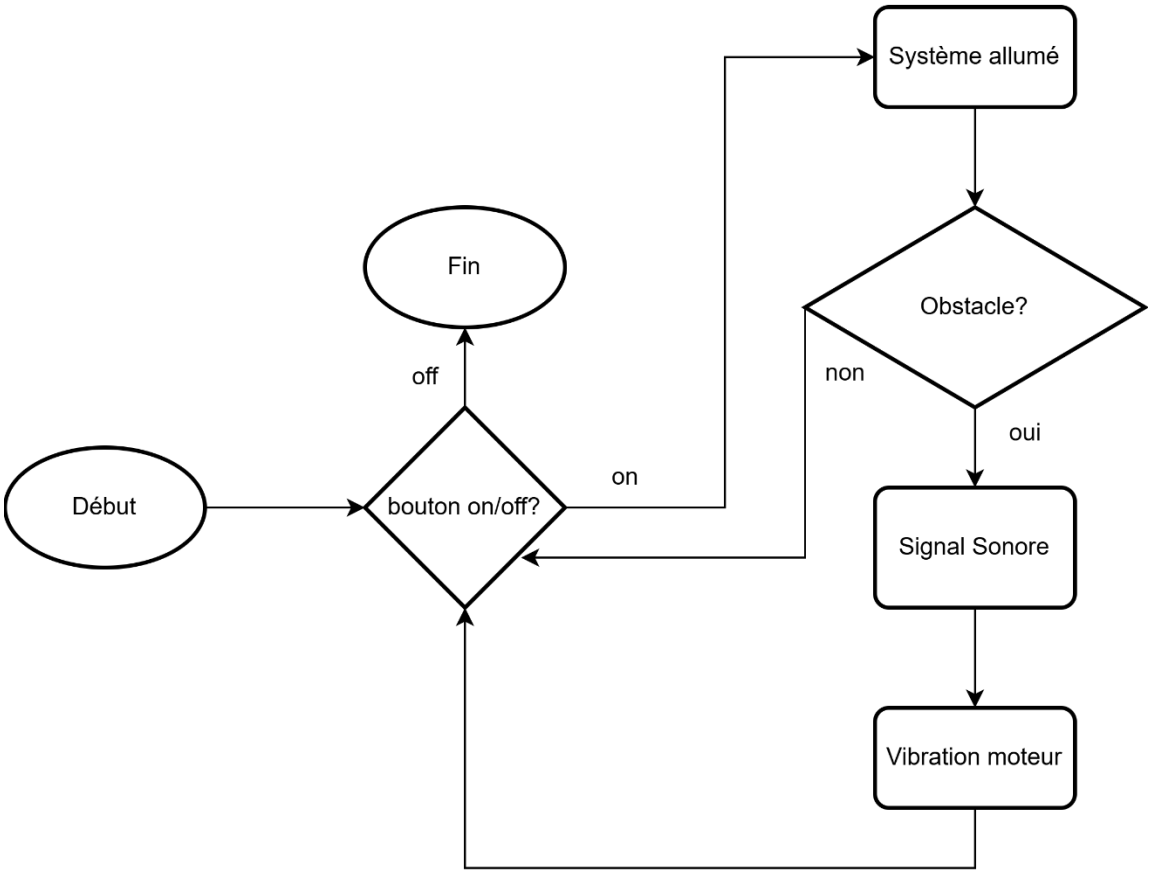


Figure 4:Diagramme de flux pour la canne

Ce second diagramme de flux détaille le processus lié à la navigation autonome de l'utilisateur grâce à la canne intelligente. Lorsque l'utilisateur active la canne via un bouton dédié, le système se met en mode surveillance et commence à détecter l'environnement en temps réel. Dès qu'un

obstacle est identifié par les capteurs, un signal sonore est généré par le buzzer et complété par une vibration du moteur, assurant une alerte adaptée à l'utilisateur. L'intensité sonore varie en fonction de la proximité de l'obstacle, offrant un retour précis et immédiat. Une fois l'alerte donnée et l'obstacle contourné, le système revient à une étape de vérification pour continuer sa surveillance active. Ce processus garantit une navigation fluide et sécurisée dans des environnements variés.

Ces diagrammes de flux, distincts mais complémentaires, offrent une vision claire des interactions et des processus clés du système. Ils jouent un rôle essentiel dans l'analyse et la validation des fonctionnalités, en mettant l'accent sur la fiabilité et la sécurité pour l'utilisateur final.

3.Méthodologie

3.1 Schémas

Maintenant nous allons pouvoir regarder les schémas physique, électriques et connexions du système.

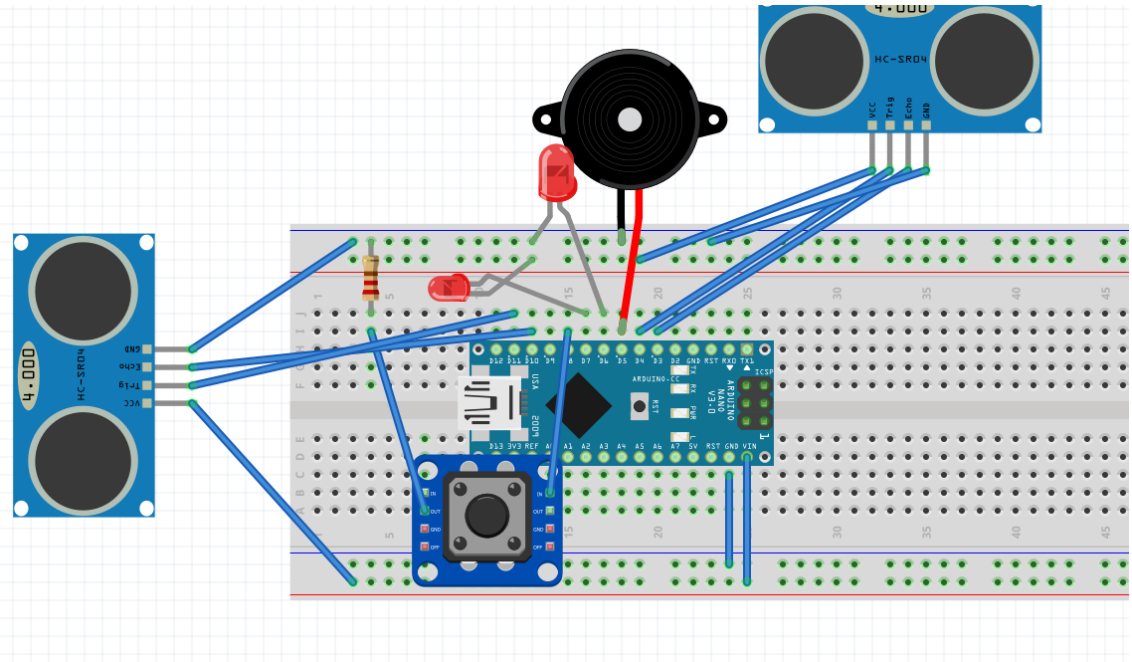


Figure 5: Schéma physique de la canne

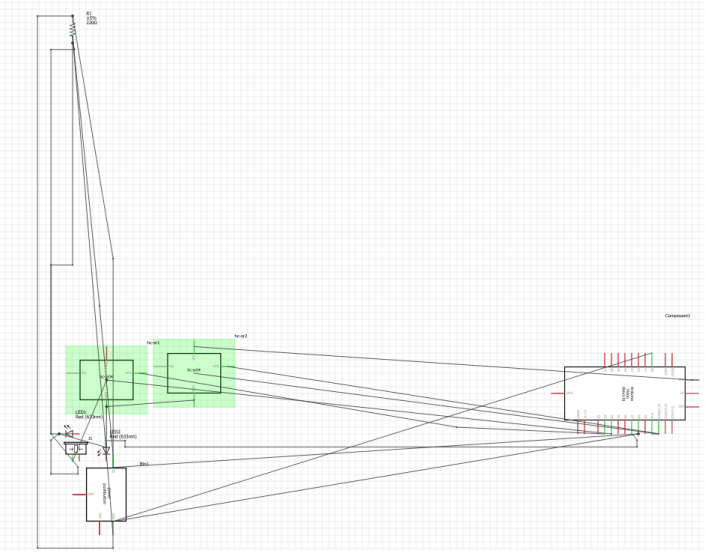


Figure 6: Schéma électrique de la canne

La première analyse est celle de la canne. Les capteurs qui sont connectées aux entrées digitales du microcontrôleur. On a deux capteurs capteur ultrasonique HC-SR04.

Pour ce qui est des signalisation lumineuse et sonore. Les LEDS et le buzzer sont connectés a des entrées digitales de l'Arduino.

Il y a quelques résistor et d'autre composants passifs, qui permet principalement de limiter le courant pour les LED et le buzzer. Ils servent aussi de protection des composants plus sensible.

Puis le bouton qui est aussi connecter à un port digital de l'Arduino.

Toutes les composantes sont alimentées via une source centralisées connectée a l'alimentation de 5V du microcontrôleur de même pour la mise à terre, toutes les composantes sont reliées aux ground de l'Arduino. Le microcontrôleur quant à lui, est alimenter par la tension transmise par l'ordinateur.

La deuxième analyse est celle de la porte.

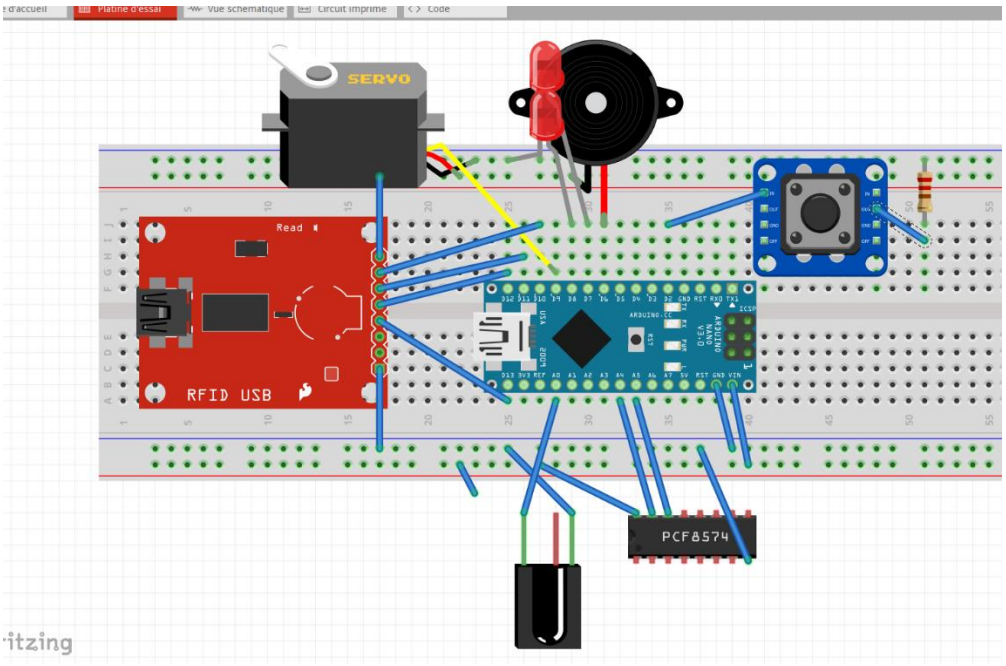


Figure 7: Schéma physique de la porte

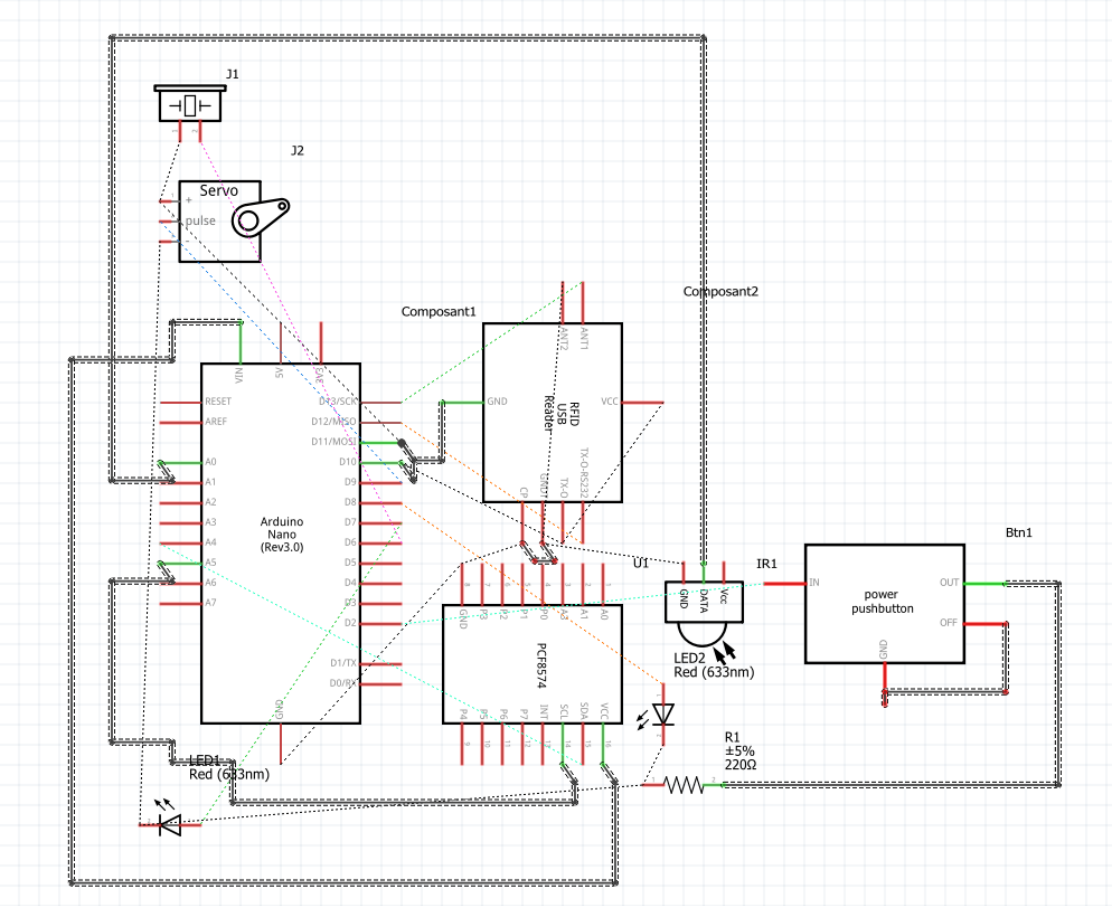


Figure 8: Circuit électrique porte

Tels que mentionner pour la canne, les composantes sont alimentées de la même manière par la source de 5V et le ground est connecté de la même manière.

Le capteur, qui est un capteur infrarouge, est à une entrée analogue du microcontrôleur.

Pour ce qui est des signalisation lumineuse et sonore. Les LEDS, le servomoteur et le buzzer sont connectés a des entrées digitales de l'Arduino.

Il y a aussi quelques résistor et d'autre composants passifs, qui permet principalement de limiter le courant pour les LED et le buzzer. Ils servent aussi de protection des composants plus sensible.

Notre lecteur de carte qui est un RFID-RC522, est connecter aux entrées digitales du microcontrôleur.

Pour ce qui est de notre écran LCD, il est relié à un I2C PCF8574T. Tout ceci est connecter aux entrées analogues de l'Arduino.

Puis le bouton qui est aussi connecter à un port digital de l'Arduino.

Pour plus de détails sur les connexions spécifiques de chacun des composantes, le code Arduino de la section 4, aura des détails plus élaborés. Chaque entrée utiliser par chaque composante, se trouve au début de chacun des codes.

3.2 Justification des choix techniques

Le design du produit final s'imposait en partie de part les spécifications du projet. Compte tenu qu'il s'agissait d'une canne pour aveugle, le modèle se devait de s'apparenter le plus possible de l'outil dont il est nommé. Ainsi, considérant cela, il fallait simplement déterminer où installer les différentes composantes électroniques.

Déjà, un deuxième senseur ultrasonique avait été ajouté par mesure de sécurité et de d'autocorrection. Ceci a été changé pour couvrir une plus grande diversité de scénario. En effet, les senseurs sont placés à différentes hauteurs sur la canne pour détecter les obstacles de différentes tailles. Comme mentionné précédemment, le buzzer signale dès la réception d'un signal. Aussi, les senseurs sont montés sur des supports qui ajustent leur alignement pour être parallèle au sol; ceci afin d'éviter de déclencher des alarmes à cause du sol.

Ensuite, le circuit électronique a été placé dans une boîte, logé à trente centimètres de l'extrémité du manche. Cette boîte sert à protéger le circuit et le microcontrôleur. De même, les différents éléments de contrôle d'utilisateur y sont insérés; notamment, les diodes d'émission lumineuses, le buzzer, et le bouton poussoir. Ces éléments ont été placés ainsi pour faciliter l'accès

à l'utilisateur. Seul les câbles des senseurs ultrasoniques s'échappent de ce boîtier pour rejoindre leur module respectif.

Finalement, en guise de porte du lieu de travail, un modèle simpliste a été implémenté. Il ressemble une barrière de stationnement. Tous les modules sont fixés à même un bloc de support. Une conception pratique a été priorisé puisqu'il s'agissait principalement de s'assurer de l'efficacité du montage.

3.3 Pseudocodes

Avant de concevoir nos code Arduino, il était important de créer un pseudocode qui représentait le squelette de notre projet. Voici le pseudocode utilisé pour la construction de notre canne:

Canne

Pseudocode si la canne est allumée ou éteint avec les LED:

```
Début
    Lire état du bouton-poussoir
        Si le bouton-poussoir appuyée (high) :
            Musique qui signale que la canne est allumée.
            Éteindre LED rouge.
            Allumée LED verte.
    Sinon
        Musique qui signal que la canne est éteinte.
        Éteindre LED verte.
        Faire clignoter la LED rouge pendant 10 secondes.
        Éteindre LED rouge.
        Fin
```

Ce code vérifie l'état du bouton-poussoir. Si le bouton a été cliquer, cela signifie que la canne est activée et ainsi, la lumière verte est allumée et une petite mélodie du début indiquant a l'aveugle que la canne est belle et bien allumée. Si le bouton est de nouveau cliqué, cela signifie

que la canne est éteinte et ainsi, la lumière rouge clignote pendant 10 secondes et une musique de fin indiquant a l'aveugle que la canne est éteinte.

Pseudocode pour la détection des obstacles et signal sonore :

```
Début

    Lire capteur1 & capteur 2. Prendre la distance la plus petite.
        Si distance la plus petite < distance max
            Fréquence buzzer = constante / distance max
            Active buzzer
        Sinon :
            Désactivée buzzer
Fin
```

Ce code prend la distance la plus courte mesurée par les capteurs, et active le son du buzzer avec une fréquence relative à la distance reçue.

Porte :

Pseudocode pour la détection d'une présence :

```
Début

    Lecture du capteur infra-rouge

        Si présence détecter :

            LCD affiche : ` "insere ta carte`
        Sinon :
            LCD affiche : `aucune presence`

Fin
```

Dans se code, si le capteur infrarouge détecte une présence, l'écran LCD affiche a l'utilisateur d'insérer sa carte. Lorsque aucune présence n'est détectée, l'écran LCD affiche que aucune présence n'est détectée.

Pseudocode pour la lecture de la carte :

Début

Lecture de la carte insérer

Si la bonne carte est insérée :

LCD affiche : `Acces autoriser`

Ouvrir la porte (faire pivoter le servomoteur à 90°)

Buzzer allumée 5.5 secondes

Arrêter buzzer.

Fermer la porte (ramener le servomoteur à 0°)

Sinon

LCD affiche : `Acces refuser`

Buzzer bruit 3 secondes.

Buzzer éteint

Fin

Puis, ce code permet à l'utilisateur d'insérer sa carte. Lorsque la bonne carte est insérée, le servomoteur tourne a 90 degrée et reste dans cette position pendant 5.5 secondes. Le buzzer commence a beeper 1.5 secondes avant que la barrière ne se referme.

4.Code Arduino

Afin de structurer et d'organiser notre code de manière claire, nous avons séparer nos codes en 2 codes d'instinct chacun de ses codes implémente des méthodes, des objets et des librairies déjà existantes. Cela nous a permis de rendre le code le plus compact et clair possible.

Code pour la canne :

```
#include "pitches.h" // Notes de musique pour le buzzer


// Définition des broches
```

```
const int trigPin1 = 4; //capteur 1
const int echoPin1 = 3; //capteur 1
const int trigPin2 = 10; //capteur 2
const int echoPin2 = 11; //capteur 2
const int BUZZER = 9;  // Buzzer
const int Led_Rouge = 7; // Lumière rouge
const int Led_Verte = 6; // Lumière verte
const int BUTTON = 8; // Bouton-poussoir

// Variables pour le bouton poussoir
bool buttonState = false;  // État du système
bool lastButtonState = LOW; // Dernier état du bouton
bool playedOpeningMelody = false;
bool playedClosingMelody = false;

// Variables pour les capteurs
float distance1, distance2, SmallerDistance;
int frequence_buzzer;

// Variables pour la gestion du clignotement de la LED rouge après OFF
unsigned long ledRougeStartTime = 0;
unsigned long ledRougeLastBlink = 0;
bool ledRougeClignote = false;
bool ledRougeState = false;

void setup() {
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);

  pinMode(Led_Rouge, OUTPUT);
  pinMode(Led_Verte, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);  // Active la résistance pull-up interne
```

```
  Serial.begin(9600);
}

void loop() {
  bool currentButtonState = digitalRead(BUTTON);

  // Détection du front montant (appui unique)
  if (currentButtonState == HIGH && lastButtonState == LOW) {
    buttonState = !buttonState; // Inverse l'état du système
    Serial.println(buttonState ? "Mode Capteurs ON" : "Mode Capteurs OFF");
    delay(200);  // Anti-rebond

    if (!buttonState) {
     // Lorsque le bouton OFF est cliqué, démarrer le clignotement pour 10 secondes
      ledRougeStartTime = millis();
      ledRougeLastBlink = millis();
      ledRougeClignote = true;
      ledRougeState = false;
    }
  }

  lastButtonState = currentButtonState;  // Mise à jour de l'état précédent

  if (buttonState) { // Mode ON
    if (!playedOpeningMelody) { // Jouer la mélodie d’ouverture une seule fois
      playOpeningMelody();
      playedOpeningMelody = true;
      playedClosingMelody = false; // Réinitialiser la fermeture
    }

    digitalWrite(Led_Verte, HIGH);
    digitalWrite(Led_Rouge, LOW);
    Capteurs();
```

```
  } else { // Mode OFF
    if (!playedClosingMelody) { // Jouer la mélodie de fermeture une seule fois
      playClosingMelody();
      playedClosingMelody = true;
      playedOpeningMelody = false; // Réinitialiser l'ouverture
    }

    digitalWrite(Led_Verte, LOW);

    // Lorsque le bouton off est cliqué je veux que la lumière rouge attende 10 secondes et ensuite s’éteint
    if (ledRougeClignote) {
      unsigned long currentTime = millis();

      // Clignoter toutes les 500 ms
      if (currentTime - ledRougeLastBlink >= 500) {
        ledRougeLastBlink = currentTime;
        ledRougeState = !ledRougeState;
        digitalWrite(Led_Rouge, ledRougeState);
      }

      // Arrêter après 10 secondes
      if (currentTime - ledRougeStartTime >= 10000) {
        ledRougeClignote = false;
        digitalWrite(Led_Rouge, LOW);
      }
    } else {
      digitalWrite(Led_Rouge, LOW);
    }
  }
}

void Capteurs() {
  distance1 = getDistance(trigPin1, echoPin1);
  distance2 = getDistance(trigPin2, echoPin2);
```

```
  if (isValid(distance1) && isValid(distance2)) {
    SmallerDistance = min(distance1, distance2);
  } else if (isValid(distance1)) {
    SmallerDistance = distance1;
  } else if (isValid(distance2)) {
    SmallerDistance = distance2;
  } else {
    SmallerDistance = -1; // Aucune distance valide
  }

  Serial.print("Capteur 1: "); Serial.print(distance1); Serial.print(" cm, ");
  Serial.print("Capteur 2: "); Serial.print(distance2); Serial.print(" cm, ");
  Serial.print("Distance plus petite: "); Serial.println(SmallerDistance);

  if (SmallerDistance > 0) {
    frequence_buzzer = constrain(-22.1 * SmallerDistance + 2610, 400, 2500);
    tone(BUZZER, frequence_buzzer);
  } else {
    noTone(BUZZER);
  }
}

float getDistance(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  float duration = pulseIn(echoPin, HIGH);
  return (duration * 0.0343) / 2;
}

bool isValid(float distance) {
```

```
  return (distance > 2 && distance < 200);
}

void playMelody(int melody[], int duration[], int size) {
    for (int i = 0; i < size; i++) {
        tone(BUZZER, melody[i], duration[i]);
        delay(duration[i] * 1.3);
    }
    noTone(BUZZER);
}

void playOpeningMelody() {
    int melody[] = {NOTE_C4, NOTE_D4, NOTE_E4};
    int duration[] = {300, 300, 300};
    playMelody(melody, duration, 3);
}

void playClosingMelody() {
    int melody[] = {NOTE_E4, NOTE_D4, NOTE_C4};
    int duration[] = {300, 300, 300};
    playMelody(melody, duration, 3);
}
```

Code porte :

```
#include "pitches.h" // Notes de musique pour le buzzer

// Définition des broches
const int trigPin1 = 4; //capteur 1
const int echoPin1 = 3; //capteur 1
const int trigPin2 = 10; //capteur 2
const int echoPin2 = 11; //capteur 2
const int BUZZER = 9;  // Buzzer
```

```
const int Led_Rouge = 7; // Lumière rouge
const int Led_Verte = 6; // Lumière verte
const int BUTTON = 8; // Bouton-poussoir

// Variables pour le bouton poussoir
bool buttonState = false;  // État du système
bool lastButtonState = LOW; // Dernier état du bouton
bool playedOpeningMelody = false;
bool playedClosingMelody = false;

// Variables pour les capteurs
float distance1, distance2, SmallerDistance;
int frequence_buzzer;

// Variables pour la gestion du clignotement de la LED rouge après OFF
unsigned long ledRougeStartTime = 0;
unsigned long ledRougeLastBlink = 0;
bool ledRougeClignote = false;
bool ledRougeState = false;

void setup() {
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);

  pinMode(Led_Rouge, OUTPUT);
  pinMode(Led_Verte, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);  // Active la résistance pull-up interne

  Serial.begin(9600);
}

void loop() {
```

```
bool currentButtonState = digitalRead(BUTTON);

// Détection du front montant (appui unique)
if (currentButtonState == HIGH && lastButtonState == LOW) {
  buttonState = !buttonState; // Inverse l'état du système
  Serial.println(buttonState ? "Mode Capteurs ON" : "Mode Capteurs OFF");
  delay(200);  // Anti-rebond

  if (!buttonState) {
   // Lorsque le bouton OFF est cliqué, démarrer le clignotement pour 10 secondes
    ledRougeStartTime = millis();
    ledRougeLastBlink = millis();
    ledRougeClignote = true;
    ledRougeState = false;
  }
}

lastButtonState = currentButtonState;  // Mise à jour de l'état précédent

if (buttonState) { // Mode ON
  if (!playedOpeningMelody) { // Jouer la mélodie d’ouverture une seule fois
    playOpeningMelody();
    playedOpeningMelody = true;
    playedClosingMelody = false; // Réinitialiser la fermeture
  }

  digitalWrite(Led_Verte, HIGH);
  digitalWrite(Led_Rouge, LOW);
  Capteurs();

} else { // Mode OFF
  if (!playedClosingMelody) { // Jouer la mélodie de fermeture une seule fois
    playClosingMelody();
    playedClosingMelody = true;
    playedOpeningMelody = false; // Réinitialiser l'ouverture
```

```
    }

    digitalWrite(Led_Verte, LOW);

    // Lorsque le bouton off est cliqué je veux que la lumière rouge attende 10 secondes et ensuite s’éteint
    if (ledRougeClignote) {
      unsigned long currentTime = millis();

      // Clignoter toutes les 500 ms
      if (currentTime - ledRougeLastBlink >= 500) {
        ledRougeLastBlink = currentTime;
        ledRougeState = !ledRougeState;
        digitalWrite(Led_Rouge, ledRougeState);
      }

      // Arrêter après 10 secondes
      if (currentTime - ledRougeStartTime >= 10000) {
        ledRougeClignote = false;
        digitalWrite(Led_Rouge, LOW);
      }
    } else {
      digitalWrite(Led_Rouge, LOW);
    }
  }
}

void Capteurs() {
  distance1 = getDistance(trigPin1, echoPin1);
  distance2 = getDistance(trigPin2, echoPin2);

  if (isValid(distance1) && isValid(distance2)) {
    SmallerDistance = min(distance1, distance2);
  } else if (isValid(distance1)) {
    SmallerDistance = distance1;
```

```
  } else if (isValid(distance2)) {
    SmallerDistance = distance2;
  } else {
    SmallerDistance = -1; // Aucune distance valide
  }

  Serial.print("Capteur 1: "); Serial.print(distance1); Serial.print(" cm, ");
  Serial.print("Capteur 2: "); Serial.print(distance2); Serial.print(" cm, ");
  Serial.print("Distance plus petite: "); Serial.println(SmallerDistance);

  if (SmallerDistance > 0) {
    frequence_buzzer = constrain(-22.1 * SmallerDistance + 2610, 400, 2500);
    tone(BUZZER, frequence_buzzer);
  } else {
    noTone(BUZZER);
  }
}

float getDistance(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  float duration = pulseIn(echoPin, HIGH);
  return (duration * 0.0343) / 2;
}

bool isValid(float distance) {
  return (distance > 2 && distance < 200);
}

void playMelody(int melody[], int duration[], int size) {
    for (int i = 0; i < size; i++) {
```

```
        tone(BUZZER, melody[i], duration[i]);
        delay(duration[i] * 1.3);
    }
    noTone(BUZZER);
}

void playOpeningMelody() {
    int melody[] = {NOTE_C4, NOTE_D4, NOTE_E4};
    int duration[] = {300, 300, 300};
    playMelody(melody, duration, 3);
}

void playClosingMelody() {
    int melody[] = {NOTE_E4, NOTE_D4, NOTE_C4};
    int duration[] = {300, 300, 300};
    playMelody(melody, duration, 3);
}
```

5.Test de validation

Une fois le code opérationnel et le prototype monté, des tests ont été effectué pour s'assurer de la qualité de la solution. Il n'y avait qu'un cas d'utilisation pour la porte, celui-ci présentant cependant un scénario de mauvaise utilisation. Le cas en question est détaillé dans le tableau 1 qui suit. Ceci différait grandement en ce qui concerne la canne. Non seulement, il fallait détecter les obstacles statiques, mais aussi les obstacles mobiles, et s'assurer du respect des comportements programmés. Les résultats des tests se trouvent dans le tableau 2.

Table 1 : Tests de validation de la porte

Test	Attente	Résultats	Validation
Ne détecte personne	Message LCD: aucune présence	Message LCD : aucune présence	☑
Détecte une présence	Message LCD: insère ta carte	Message LCD: insère ta carte	☑
Insère une carte erronée	Barriere reste fermer	Barriere reste fermer	☑
Insère la bonne carte	Barriere se lève, buzzer sonne et ensuite se ferme	Barriere se lève, buzzer sonne et ensuite se ferme	☑

Table 2 : Tests de validation de la canne

Test	Attentes	Résultats	Validation
On	• Lumière verte s'allume • Joue un timbre ascendant	• Lumière verte s'allume • Joue un timbre ascendant	☑
Off	• Lumière rouge s'allume 10 secs. • Joue un timbre descendent	• Lumière rouge s'allume 10 secs. • Joue un timbre descendent	☑
Capteur détecte un objet proche	• Buzzer sonne selon la distance	• Buzzer sonne selon la distance	☑
Capteur détecte un objet loin	• Buzzer sonne selon la distance	• Buzzer sonne selon la distance	☑
Capteur détecte un objet se rapprochant	• Buzzer sonne avec une fréquence ascendante	• Buzzer sonne avec une fréquence ascendante	☑
Capteur détecte un objet s'éloignant	• Buzzer sonne avec une fréquence descendante	• Buzzer sonne avec une fréquence descendante	☑

6.Analyse de sécurité

Comme dans toute construction, la canne et la porte présentes des failles de sécurité. En ce qui concerne la canne, l'un d'entre eux est le manque de détection d'un obstacle par les deux capteurs.

En effet, par leur disposition, si un objet plus petit que le plus bas détecteur, et commençant plus haut que le plus court détecteur, l'obstacle en question ne serait pas détecté. Il en est de même lorsqu'un obstacle se trouve nettement dans l'espace entre les deux détecteurs. Le niveau de la batterie n'étant pas surveillé par le programme, un manque d'alimentation à cause de pile faible empêcherait le dispositif de fonctionner. Un dernier aspect serait la possibilité de dommages sur les câbles reliant le microcontrôleur et les senseurs. Ceux-ci sont exposé et pourraient s'abîmer lors de la manipulation de canne; interrompant ainsi son fonctionnement.

Autre que le même souci de contrôle du niveau de batterie, le système d'ouverture de la porte ne comporte pas de mécanisme d'interruption si la barrière venait à rencontrer une résistance. Cette dernière pourrait être causé par un passant pris sous la barrière.

7.Résolution de problèmes et débogage

7.1 Défis et solutions

Lors de la conception du projet, plusieurs défis ont fait surface. Bien qu'il soit difficile d'énumérer tous les problèmes rencontrés, un résumé de quelques-uns des principaux défis et de leurs solutions est présenté ci-dessous.

Planification et organisation

- **Recherche de matériaux** :
    - *Défi* : Trouver le matériel approprié pour fabriquer une canne longue et qui permet de déposer notre Arduino ainsi que nos composantes. Il y a aussi certaine composante que nous voulions inclure telle qu'un module de choque et un GPS, qui nous a été impossible de trouver a temps. De plus, es fils assez long pour relier nos capteurs a notre Arduino.
    - *Solution* : L'équipe a finalement trouvé opter pour un manche a balais et du carton pour fabriquer notre produit. Nos essaie de l'imprimante 3D se sont avéré infructueuses. Nous avons dû abandonner notre projet du module choque et du GPS. En effet, nous nous étions trompés et avons pris un capteur de choque au lieu du détecteur puis faute de temps, il nous était impossible de trouver un module

choque. Pour le GPS, un membre de notre équipe pensait posséder le module, cependant, cette assomption c'est avérer fausse et faute de temps, il nous était impossible d'en commander un. Pour les fils des capteurs, à cause d'une mauvaise communication, seulement 4 longs fils furent acquis au lieu de 8. Ce qui nous a demander de la créativité et de la soudure de multiple fils de longueur plus courte.

- **Visualisation et compréhension du projet** :
    - *Défi* : Des incompréhensions entre les membres de l'équipe ont compliqué la modélisation du problème technique.
    - *Solution* : Des croquis et schémas ont été réalisés pour clarifier les idées et visualiser le système, facilitant la collaboration et la résolution des désaccords.
- **Répartition des tâches et gestion du temps** :
    - *Défi* : La répartition des tâches et la gestion de l'horaire n'étaient pas faciles, causant des retards et des frustrations. Ayant tous des horaires chargées et différents avec des engagements divers, s'accorder semblait presque impossible. Des rencontres de 30 minutes chaque jeudi était effectué, mais n'était pas assez pour bien avancer.
    - *Solution* : Un calendrier de tâches a été créé à l'aide de Word, permettant de mieux organiser le travail. Un groupe WhatsApp a également été utilisé pour assurer une communication régulière et coordonnée.

Conception technique

- **Mauvais fonctionnement de la barrière** :
    - *Défi* : La barrière fonctionnait à l'inverse, se levant lorsqu'elle devait se baisser et vice-versa.
    - *Solution* : L'équipe a corrigé cette erreur en ajustant la logique du programme et en vérifiant les signaux de commande.
- **Défaut de la LED** :
    - *Défi* : Les LED et capteurs ne fonctionnait pas correctement en raison d'un mauvais branchement.
    - *Solution* : En regardant a multiple reprise le schéma des pins, les erreurs on été capter et corrigées.

7.2 Outils de débogage

Il est essentiel de mentionner les outils de débogage qui nous ont principalement aidés.

1. **Dessins techniques et schémas** :
    - Utilisés pour visualiser les concepts et résoudre les désaccords techniques au sein de l'équipe.
2. **Microsoft Excel** :
    - Servi à créer un calendrier des tâches, facilitant une meilleure répartition et suivi des responsabilités.
3. **WhatsApp** :
    - Utilisé pour maintenir une communication constante et coordonner les efforts de l'équipe en temps réel.
4. **Fritzing :**
    - Utilisé pour le design électrique et physique du projet.
5. **Arduino IDE :**
    - Utilisé pour le codage du projet.
6. **GitHub :**
    - Utilisé pour déposer les codes du projet.

8.Discussion et conclusion

8.1 Discussion

8.2 Amélioration

Il est important d'aborder les différentes améliorations possibles, qui permettrai d'augmenter les fonctionnalité et l'efficacité du système. Une liste d'amélioration a envisagé dans le futur sera démontré ci-dessous.

Tout d'abord, l'ajout d'une phase d'initialisation ainsi qu'un détecteur de mal fonction de l'équipement dans le code, serai un élément à prendre en considération lors de la reproduction de ce projet. En effet, il sera primordial de s'assurer que si une composante de la canne ou du système de porte deviens défaillant ou se brise, le système puisse alerter l'utilisateur.

Ensuite, un ajout de capteur permettrait d'avoir un champ de détection d'obstacle plus vaste et permettrai à l'aveugle de réduire sont risque de collisions.

Par la suite, puisque les enjeux sur l'environnement sont de plus en plus importants, il serait bien d'intégrer un mode de consommation d'énergie. En effet, les capteurs pourraient avoir un mode veille, permettant de réduire la consommation énergétique des composants inactifs.

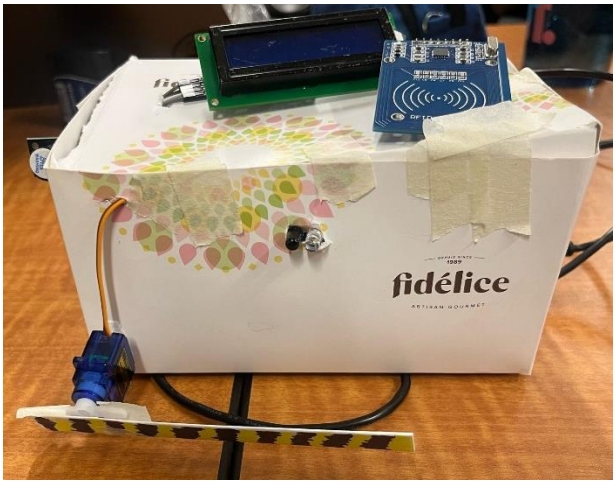
Pour continuer, on pourrait avoir un capteur sur la barrière qui mène au lieu de travaille ainsi, si un objet ou une personne est détecter, la barrière aurai le temps de s'arrêter sans heurter ce qui se trouve sous celle-ci.

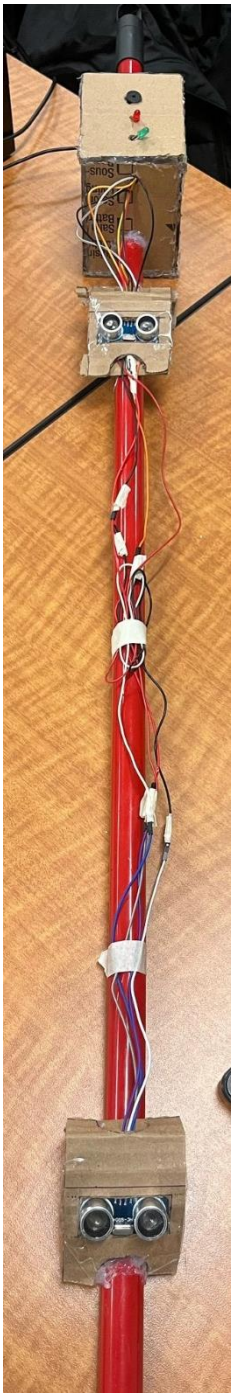
L'ajout de plusieurs autre composantes pourrait aussi être utile au projet. Tel que le module GPS sur la canne qui permettrait de suivre en temps réel la localisation de la canne. Un module de vibration qui permettrait à l'utilisateur non seulement d'entendre lorsqu'il se rapproche d'un objet, mais de aussi ressentir une vibration de la canne. En plus, un module Wifi pourrais être un excellent ajout, permettant de contrôler la canne et de voir le mouvement du système de porte a distance.

8.3 Conclusion

9.Annexes

Photos du prototype.





10. Reference

ArduinoModules.info. (s.d.). *Module interrupteur à vibration KY-002*. Lien URL : https://arduinomodules.info/ky-002-vibration-switch-module/ (Consulté le 1er avril 2025).

Components101. (s.d.). *Bouton poussoir – Interrupteur tactile*. Lien URL : https://components101.com/switches/push-button (Consulté le 18 mars 2025).

HandsOn Tech. (s.d.). *Fiche technique du module RFID RC522*. Lien URL : https://www.handsontec.com/dataspecs/RC522.pdf (Consulté le 20 mars 2025).

Instructables. (s.d.). *Clavier 4x4 avec Arduino*. Lien URL : https://cdn.instructables.com/ORIG/FW9/SBS0/J3EPQTB8/FW9SBS0J3EPQTB8.pdf (Consulté le 24 mars 2025).

Murata Manufacturing Co., Ltd. (2006). *Composants sonores piézoélectriques : PKM13EPYH4000-A0*. Lien URL : https://www.farnell.com/datasheets/1498852.pdf (Consulté le 16 mars 2025).

ParaArduino. (s.d.). *Module buzzer actif KY-012*. Lien URL : https://paraarduino.com/kits/kit-de-37-sensores/modulo-buzzer-activo-ky-012/ (Consulté le 28 mars 2025).

Sayad, O., & Ouled Lahocine, A. (2024). *Conception et réalisation d'une canne intelligente pour malvoyants* [Mémoire d'ingéniorat, Université Badji Mokhtar Annaba]. Lien URL : https://biblio.univ-annaba.dz/ingeniorat/wp-content/uploads/2024/11/memoire-final-oussama-SAYAD-OUSSAMA-OULED-LAHOCINE-Amer.pdf (Consulté le 30 mars 2025).

SparkFun Electronics. (s.d.). *Fiche technique du capteur à ultrasons HC-SR04*. Lien URL : https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf (Consulté le 15 mars 2025).

Texas Instruments. (2004). *PCF8574 – Extension d'E/S 8 bits pour bus I²C*. Lien URL : https://www.ti.com/lit/ds/symlink/pcf8574.pdf (Consulté le 22 mars 2025).

TowerPro. (s.d.). *Fiche technique du micro servo SG90*. Lien URL : http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf (Consulté le 26 mars 2025).