



Compte rendu SAE

Développement d'un système de contrôle d'accès RFID avec arduino

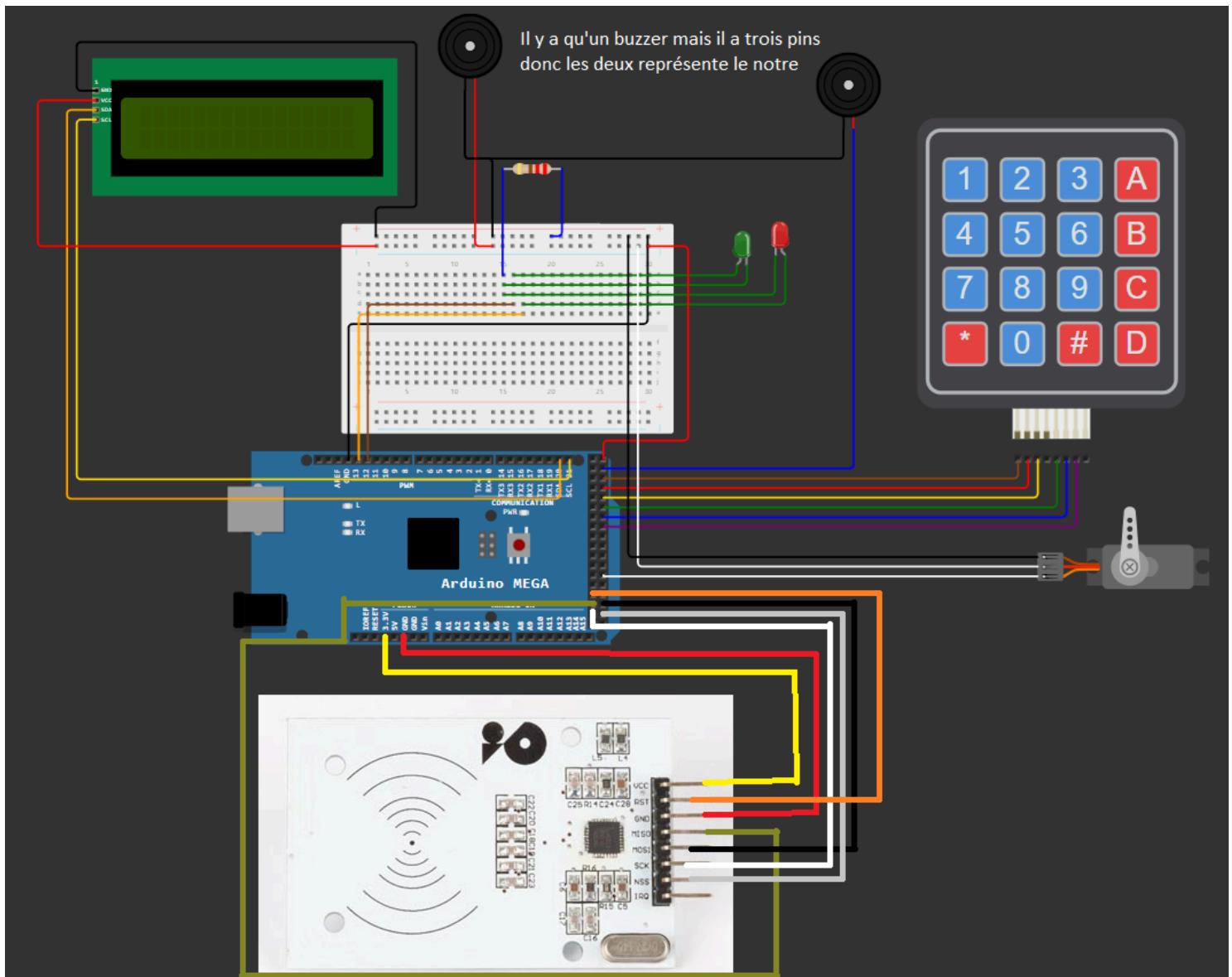
Elèves : BOROWEC - DHUIEGE - PATERNOTTE

Tuteur enseignant : HABIS Nicolas

Période du 14/10/2024 au 19/11/2024

Sommaire

Objectif 1: Lecture de l'UID d'un badge RFID	2
Objectif 2: Ajout d'une LED et d'un buzzer	3
Objectif 3: Écriture et affichage d'un prénom sur LCD	4
Objectif 4: Validation par donnée écrite et interaction avec LED/buzzer	5
Objectif 5: Ajout d'un actionneur	6
Objectif plus: Ajout d'un mot de passe	7
Fonctionnement des composants	10
Conclusion	17
Archives	18



Objectif 1: Lecture de l'UID d'un badge RFID

L'objectif est d'utiliser le module **RFID WPI405** pour lire l'**UID (Identifiant Unique)** d'un badge RFID, puis d'afficher cette information sur un **écran LCD 16x2 I²C**.



Pourquoi lire l'UID ?

Le WPI405 est un module performant utilisant la fréquence standard de 13,56 MHz, compatible avec la norme ISO/IEC 14443. Il permet :



- D'identifier rapidement un badge RFID grâce à son UID unique.
- De servir de point de départ pour des applications comme l'authentification, le contrôle d'accès, ou la traçabilité.

Programmation :

Pour afficher les données, nous utiliserons un écran LCD 16x2 I²C, programmé avec la bibliothèque LiquidCrystal_I2C. Cette bibliothèque simplifie la gestion des affichages en exploitant l'interface I²C du LCD.

Pour manipuler le module RFID WPI405, nous utiliserons la bibliothèque MFRC522, qui fournit les fonctions nécessaires pour lire l'UID et interagir avec le capteur. La communication entre l'arduino et le module RFID se fera via le protocole SPI, pris en charge par la bibliothèque SPI.

Branchements :

Nous allons brancher le capteur RFID ainsi :

Le LCD est branché ainsi :

RFID WPI405	Arduino
VCC	3.3V
RST	48
GND	GND
MISO	50
MOSI	51
SCK	52
NSS	53
IRQ	Non connecté

LCD	Arduino
GND	GND
Vcc	5V
SDA	20
SCL	21

Objectif 2: Ajout d'une LED et d'un buzzer

L'objectif est d'ajouter des **LED** et un **buzzer** au montage afin de donner un retour visuel et sonore lors de la lecture d'un badge RFID.

Pourquoi ajouter une LED et un buzzer ?

- **LED rouge** : Cette LED s'allume lorsque l'UID du badge RFID n'est pas reconnu ou est inconnu, indiquant ainsi une erreur.
- **LED verte** : Cette LED s'allume lorsque l'UID du badge est reconnu, indiquant que le badge est enregistré et autorisé.
- **Buzzer** : Le buzzer retentit lorsque l'UID est inconnu, pour alerter l'utilisateur de l'erreur.

Ces ajouts permettent une gestion améliorée de l'accès et offrent un retour immédiat sur l'état du badge (enregistré ou inconnu), en complément de l'affichage sur l'écran LCD.



Programmation

Les composants seront contrôlés de la manière suivante :

- La LED rouge sera activée si l'UID du badge est inconnu.
- La LED verte s'allumera si le badge est enregistré.
- Le buzzer émettra un son en cas de badge inconnu.

Nous utiliserons les mêmes bibliothèques et protocoles que pour l'objectif précédent (LiquidCrystal_I2C pour l'écran LCD et MFRC522 pour le module RFID). De plus, les LED et le buzzer seront contrôlés via des pins numériques sur l'Arduino.

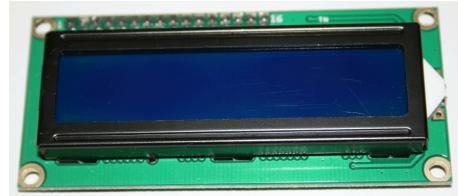
Branchement :

Les LEDs sont branchées avec une résistance avant le ground (GND).

- La LED verte est branchée sur le pin 13 de l'Arduino pour l'anode et la cathode sur le GND, avec une résistance de 220Ω en série.
- La LED rouge est branchée sur le pin 12 de l'Arduino pour l'anode et la cathode sur le GND, avec une résistance de 220Ω en série.
- Le signal du buzzer est branché sur le pin 25, avec le Vcc connecté à 5V et le GND connecté à GND.

Objectif 3: Écriture et affichage sur LCD

L'objectif est d'écrire le mot de passe de la carte sur une partie spécifique du badge RFID, puis de lire et afficher ce mot de passe sur un écran LCD 16x2 I²C lors du passage du badge sur le lecteur RFID.



Pourquoi afficher des informations sur un LCD, comme le prénom du propriétaire?

- Permet d'associer un prénom à un UID de badge RFID.
- Facilite l'identification des utilisateurs ou des objets associés à chaque badge.
- Apporte un retour visuel immédiat, simplifiant les opérations de contrôle ou d'authentification.

Programmation :

La liste des cartes autorisées est stockée dans l'Arduino. Lorsqu'une carte autorisée est scannée, le mot de passe unique sera lu depuis la carte. Le mot de passe sera ensuite affiché sur l'écran LCD 16x2 à l'aide de la bibliothèque LiquidCrystal_I2C.

La lecture de l'UID se fait avec la bibliothèque MFRC522, et l'affichage sur le LCD est assuré par la bibliothèque LiquidCrystal_I2C. Le mot de passe sera affiché sur la 2e ligne de l'écran LCD pour une visualisation claire.

Objectif 4: Validation par donnée écrite et interaction avec LED/buzzer

L'objectif est de fusionner les deux projets précédents : la lecture de l'**UID** du badge RFID et la lecture et la demande du mot de passe. L'IUD sera comparé à une liste d'IUD autorisés, et selon que l'UID soit connu ou non, l'Arduino demandera le mot de passe et interagira avec la **LED** et le **buzzer**.

Pourquoi valider par mot de passe ?

La validation par mot de passe permet d'ajouter une couche de sécurité supplémentaire à l'identification par badge RFID. En fonction du mot de passe associé à l'UID, cela permet de confirmer si le badge est scanné par son propriétaire.

- Si l'UID est **connu**, le mot de passe est demandé. En cas de code correct, la **LED verte** s'allume, indiquant que l'accès est autorisé. Sinon, la **LED rouge** s'allume et le **buzzer** retentit pour signaler un accès refusé
- Si l'UID est **inconnu**, le mot de passe n'est pas demandé : la **LED rouge** s'allume tout de suite et le **buzzer** retentit pour signaler un accès refusé.

Ce processus permet de fournir une confirmation visuelle et sonore de la validité du badge et de l'utilisateur, en fonction des informations lues par le lecteur RFID.

Programmation :

La programmation consistera à lire l'UID du badge RFID et à le comparer avec les UID associés dans la mémoire de l'Arduino. Si l'UID correspond à un de ceux enregistrés, le mot de passe sera lu puis demandé. Si celui-ci est correctement tapé, la LED verte s'allumera, sans que le buzzer soit activé. Si l'UID est inconnu ou si le mot de passe est incorrect, la LED rouge s'allumera et le buzzer retentira.

Nous utiliserons une structure de données pour stocker les UID. Une fois que l'UID est lu, le mot de passe sera récupéré dans la carte. La réponse (LED verte ou rouge, buzzer) sera alors donnée en fonction de la bonne écriture du mot de passe.

Objectif 5: Ajout d'un actionneur

L'objectif est d'ajouter un **actionneur**, tel qu'un **relais avec électroaimant** (utilisé pour une porte ou un coffre) ou un **servomoteur**, qui sera activé lorsque le badge RFID est reconnu comme valide. Cet actionneur permettra de réaliser une action physique en réponse à la lecture et à la validation d'un badge.



Pourquoi ajouter un actionneur ?

L'ajout d'un actionneur permet de rendre l'application interactive et fonctionnelle. Par exemple, un **relais avec électroaimant** peut être utilisé pour déverrouiller une porte ou un coffre lorsque le badge RFID valide est scanné. Un **servomoteur** pourrait être utilisé pour ouvrir une porte ou un mécanisme automatisé. Cette action physique est une extension logique d'un système de contrôle d'accès ou d'identification basé sur un badge RFID.

- Lorsque le badge est **valide** et le mot de passe correct, l'actionneur (relais ou servomoteur) s'active pour réaliser l'action physique (ouvrir une porte, activer un mécanisme, etc.) et la LED verte est allumée.
- Si le badge est **invalidé** ou si le mot de passe est incorrect, la LED rouge et le buzzer sont activés.

Programmation :

Dans cette partie, nous avons utilisé la bibliothèque Servo pour contrôler le servomoteur. Le principe est de lire l'UID et le prénom du badge RFID, et si l'UID est associé à un prénom valide, le servomoteur sera activé pour effectuer l'action physique (comme ouvrir une porte).

La bibliothèque Servo permet de contrôler le servomoteur avec précision en envoyant un signal PWM (modulation de largeur d'impulsion). Par exemple, en envoyant un angle spécifique au servomoteur, il effectuera une rotation pour réaliser l'action souhaitée.

Branchements :

Le servomoteur est connecté à la pin 45 de l'Arduino, avec le Vcc connecté à 5V et le GND connecté au ground (GND). Cette configuration permet de contrôler le servomoteur en envoyant des signaux PWM via la pin 45 pour activer son mouvement (par exemple, pour ouvrir une porte).

Objectif plus: Ajout d'un mot de passe

L'objectif de ce bonus est d'ajouter un **mot de passe personnalisé** pour chaque badge RFID à l'aide d'un **keypad à 12 touches**. Chaque badge RFID est associé à un **mot de passe unique**, et le **keypad** permet à l'utilisateur de saisir ce mot de passe pour valider son accès en plus de la lecture de l'**UID** du badge.

Pourquoi ajouter un mot de passe personnalisé ?

L'ajout d'un mot de passe personnalisé pour chaque badge RFID améliore la sécurité du système en permettant un double facteur d'authentification :



1. Le badge RFID, qui sert d'identifiant unique.
2. Le mot de passe personnalisé, qui ajoute une couche de sécurité supplémentaire.

Cela permet de garantir que l'utilisateur possède à la fois le badge physique et connaît le mot de passe associé, rendant ainsi le système plus sûr contre les accès non autorisés.

Programmation :

Pour cet objectif, nous avons utilisé la bibliothèque Keypad pour gérer l'entrée du mot de passe via le keypad à 12 touches. La bibliothèque MFRC522 est utilisée pour lire l'UID des badges RFID, et la bibliothèque Servo pour contrôler l'actionneur, qui dans ce cas est un servomoteur.

1. Lecture du badge RFID :

Le programme commence par lire l'UID du badge RFID à l'aide du module WPI405 et de la bibliothèque MFRC522. Lorsqu'un badge valide est détecté, son UID est comparé à une liste d'UID enregistrés.

2. Saisie du mot de passe via le keypad :

Après la lecture du badge, l'utilisateur doit entrer un mot de passe personnalisé en utilisant le keypad à 12 touches. Grâce à la bibliothèque Keypad, chaque touche appuyée est enregistrée et formée en une chaîne de caractères (le mot de passe). La touche # permet de valider le mot de passe et la touche * permet d'effacer les numéros rentrée.

3. Vérification du mot de passe :

Une fois le mot de passe saisi, le programme vérifie si le mot de passe correspond à celui stocké pour le badge RFID scanné.

- Si le mot de passe est correct, l'accès est validé et le programme allume la LED verte, et active l'actionneur (servomoteur).

- Si le mot de passe est incorrect, la LED rouge s'allume et le buzzer retentit pour signaler un accès refusé.
4. **Activation de l'actionneur :**
Si le badge et le mot de passe sont valides, l'actionneur (par exemple, un servomoteur) est activé pour effectuer une action physique (comme ouvrir une porte). Cela est contrôlé par la bibliothèque Servo.
 5. **Gestion de l'accès :**
Si le badge ou le mot de passe est incorrect, l'accès est refusé, et aucune action physique n'est effectuée (pas de mouvement du servomoteur).

Problème rencontré :

Lors de l'ajout du mot de passe personnalisé pour chaque badge RFID, une difficulté est survenue lors de la lecture des cartes. En effet, après avoir inscrit un mot de passe sur chaque badge, le module RFID a rencontré des problèmes pour lire l'UID des cartes.

Le principal problème semble être lié à des interférences entre la lecture de l'UID et les données supplémentaires (mot de passe) inscrites sur les cartes RFID. Les badges RFID sont conçus principalement pour contenir un UID unique et peuvent avoir des capacités limitées pour stocker d'autres types de données. Lorsque le mot de passe est écrit sur la carte, cela pourrait perturber la lecture de l'UID, rendant l'accès difficile ou impossible.

Fonctionnement des différents composants

Principe de fonctionnement du RFID WPI405

Le **RFID WPI405** fonctionne selon les principes de base de la technologie d'identification par radiofréquence (RFID). Voici, de manière simple, comment il opère :



1. Émission d'un signal radio

Le module WPI405 émet un champ électromagnétique via son antenne. Ce signal est utilisé pour détecter et communiquer avec les tags RFID à proximité.

2. Activation des tags passifs

Si le système interagit avec des tags passifs (qui n'ont pas de source d'énergie propre), ce champ électromagnétique sert à leur fournir l'énergie nécessaire pour répondre.

3. Échange d'informations

Une fois activé, le tag transmet ses données, généralement un identifiant unique ou des informations stockées dans sa mémoire. Le lecteur capte ce signal et le convertit en données exploitables.

4. Communication avec un système externe

Les données collectées par le WPI405 sont envoyées vers un ordinateur ou un autre système via une interface (comme USB, UART, ou RS232). Si le module est également capable d'écrire sur les tags, il peut envoyer des commandes pour modifier ou ajouter des informations.

5. Portée et fréquence

Le fonctionnement du module dépend de la fréquence utilisée :

- **LF (125-134 kHz)** : Portée courte, jusqu'à 10 cm.
- **HF (13,56 MHz)** : Portée jusqu'à environ 1 mètre.
- **UHF (860-960 MHz)** : Plus longue portée, pouvant atteindre plusieurs mètres.

La portée réelle dépend de la puissance du module, de l'antenne et du type de tag utilisé.

Exemples d'utilisation

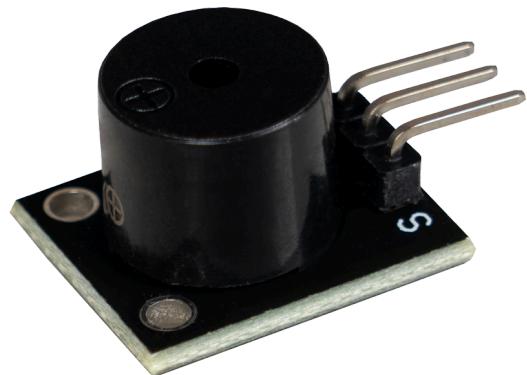
Le WPI405 peut être utilisé dans divers domaines :

- Gestion d'accès (portiques, cartes d'identification).
- Suivi d'objets (logistique, inventaire).
- Traçabilité des produits.

Principe de fonctionnement d'un buzzer

Un **buzzer** est un dispositif électromécanique ou électronique utilisé pour produire un son, généralement sous forme de bip ou d'alarme.

Voici comment il fonctionne :



1. Électromécanique ou Piézoélectrique ?

Il existe deux types principaux de buzzers :

- **Buzzer électromécanique** : Utilise un système à base d'électro aimant et de membrane vibrante.
- **Buzzer piézoélectrique** : Fonctionne grâce à un disque en matériau piézoélectrique qui vibre lorsqu'un courant alternatif est appliqué.

Le choix du type dépend de l'application, mais leur principe général est similaire.

2. Conversion de l'énergie électrique en son

- **Électromécanique** : Un courant électrique alimente un électroaimant, créant un champ magnétique. Ce champ attire ou repousse une membrane métallique, qui vibre et génère des ondes sonores.
- **Piézoélectrique** : Lorsqu'un courant alternatif traverse le matériau piézoélectrique, il se dilate et se contracte rapidement, provoquant des vibrations. Ces vibrations se propagent dans l'air sous forme de son.

3. Contrôle du son

- La **fréquence** du courant appliqué détermine le son émis (aigu ou grave).
- Certains buzzers sont "actifs" (émettent un son fixe dès qu'ils sont alimentés), tandis que les "passifs" nécessitent un signal de commande externe pour produire un son spécifique.

4. Applications

Les buzzers sont couramment utilisés dans :

- Alarmes (incendie, sécurité).
- Notifications (montres, minuteurs, équipements médicaux).
- Signaux sonores dans des circuits électroniques.

Exemple simplifié de fonctionnement

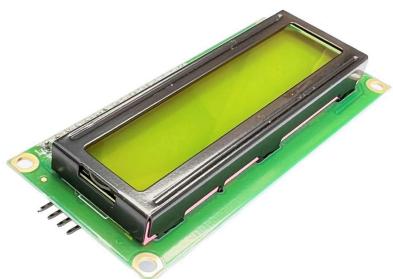
Lorsque le buzzer est alimenté :

1. Un courant électrique est appliqué.
2. Cela provoque la vibration d'une membrane ou d'un disque piézoélectrique.
3. Ces vibrations génèrent des ondes sonores audibles, produisant un "bip" ou un son continu.

Principe de fonctionnement d'un écran LCD I2C

Un écran LCD (**Liquid Crystal Display**) I2C est une interface d'affichage utilisée pour présenter des informations alphanumériques ou graphiques, tout en exploitant une connexion simplifiée grâce au protocole **I2C (Inter-Integrated Circuit)**.

Voici une explication claire de son fonctionnement :



1. Composition et Structure

L'écran LCD I2C est constitué de deux parties principales :

- **Le module LCD** : Contient les cristaux liquides et la matrice permettant d'afficher les caractères ou graphiques.
- **Le module I2C (adaptateur)** : Convertit les signaux du microcontrôleur (ou autre système) en commandes compatibles avec le LCD.

Le module LCD est souvent basé sur un contrôleur, qui gère l'affichage des caractères

2. Principe de fonctionnement de l'écran LCD

Un écran LCD utilise des cristaux liquides placés entre deux couches polarisantes. Lorsqu'une tension électrique est appliquée :

1. Les cristaux liquides modifient leur orientation.
2. Cela contrôle la quantité de lumière traversant les segments.
3. Ces segments forment des caractères ou images visibles à l'écran grâce au rétroéclairage.

3. Rôle de l'interface I2C

La communication I2C simplifie les connexions avec l'écran :

- Au lieu d'avoir 12-16 fils comme pour un écran LCD standard, le module I2C n'utilise que **4 fils** :
 - VCC : Alimentation
 - GND : Masse
 - SDA : Données série
 - SCL : Horloge série
- Le protocole I2C fonctionne avec un maître (comme un microcontrôleur) qui envoie des données et un esclave (le module LCD) qui les interprète pour actualiser l'affichage

4. Echange des données

1. Le microcontrôleur envoie des instructions ou des données (par ex., du texte à afficher) via les broches **SDA** et **SCL**.
2. Le module I2C les convertit en signaux compréhensibles pour le contrôleur LCD
3. L'écran met à jour l'affichage en fonction des instructions reçues.

5. Programmation

- Un écran LCD I2C est souvent contrôlé avec des bibliothèques préexistantes, comme la bibliothèque **LiquidCrystal_I2C** en Arduino.
- La bibliothèque simplifie l'envoi des commandes (comme afficher du texte, déplacer le curseur, ou effacer l'écran).

6. Applications

Les écrans LCD I2C sont largement utilisés dans :

- Projets Arduino et Raspberry Pi.
- Afficheurs pour instruments de mesure.
- Interfaces utilisateur pour équipements électroniques.

Tests des programmes

Le premier programme consiste à la lecture de toutes les cartes trouvées. Les données lues et affichées sur le terminal sont l'UID et La ligne 26, celle où le mot de passe est caché.

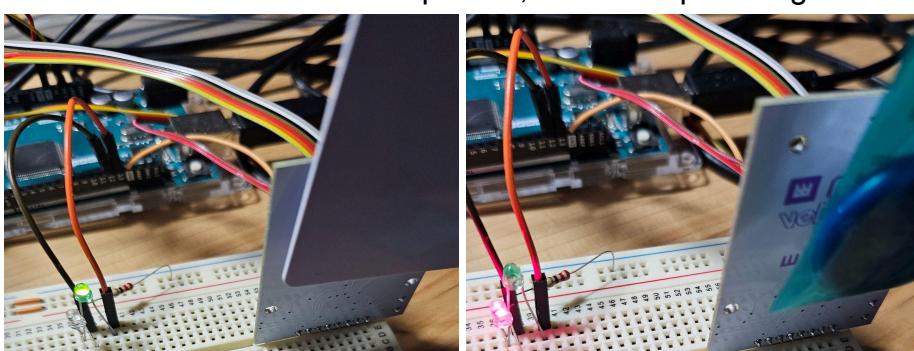
Après deux cartes et un badge de lus, le terminal nous renvoie :

```
20:38:15.775 -> **Carte trouvée : **  
20:38:15.775 -> Card UID: 70 68 DB 3C  
20:38:15.822 -> Card SAK: 08  
20:38:15.822 -> PICC type: MIFARE 1KB  
20:38:15.869 -> Ligne où se cache le code: 6750028806669102  
20:38:15.916 -> **Lecture terminée**  
20:38:15.916 ->  
20:38:19.822 -> **Carte trouvée : **  
20:38:19.822 -> Card UID: 19 B4 DB 3C  
20:38:19.869 -> Card SAK: 08  
20:38:19.869 -> PICC type: MIFARE 1KB  
20:38:19.916 -> Ligne où se cache le code: 2409022007770193  
20:38:19.963 -> **Lecture terminée**  
20:38:19.963 ->  
20:38:27.401 -> **Carte trouvée : **  
20:38:27.401 -> Card UID: 39 69 5F 7C  
20:38:27.448 -> Card SAK: 08  
20:38:27.448 -> PICC type: MIFARE 1KB  
20:38:27.495 -> Ligne où se cache le code:
```

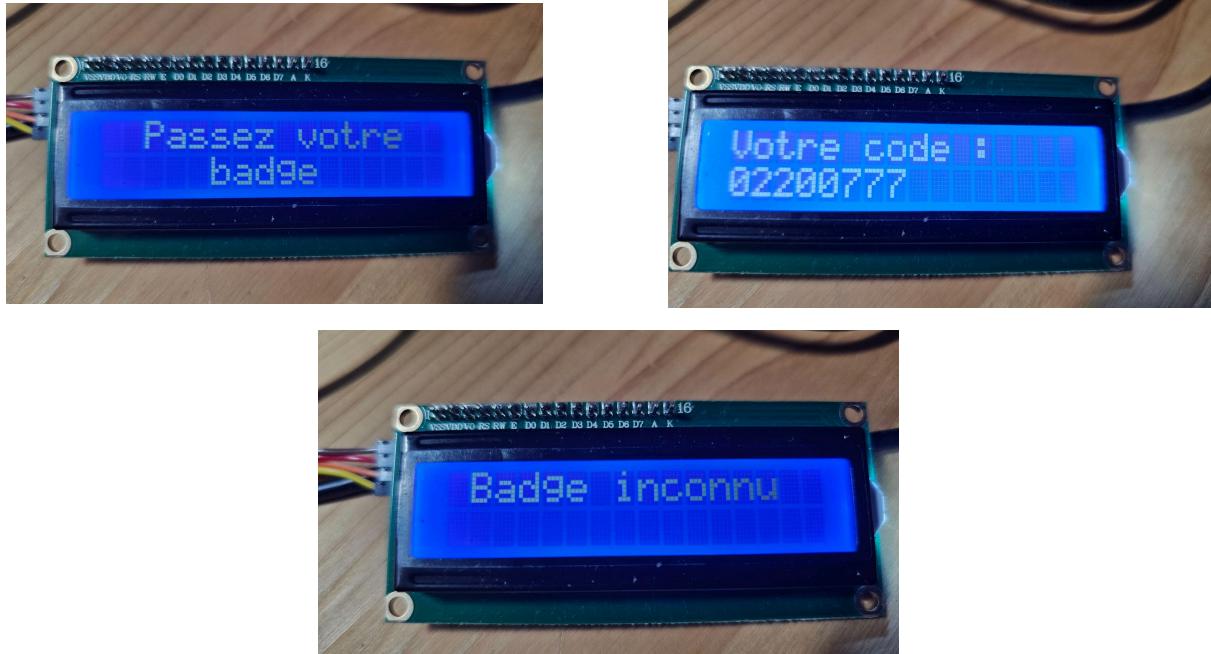
Nous remarquons qu'avec le badge, la ligne du code est vide. En effet, le badge ne fait pas partie des UID autorisés, donc n'a pas forcément le mot de passe caché dans d'autres chiffres. Pour le SAE, seules les deux cartes sont autorisées.

Le deuxième programme est chargé de l'ajout des LEDs et du buzzer.

Lorsqu'une carte connue est approchée, la LED verte s'allume. Sinon, la LED rouge s'allume et le buzzer fait un bip court, suivi d'un plus long.



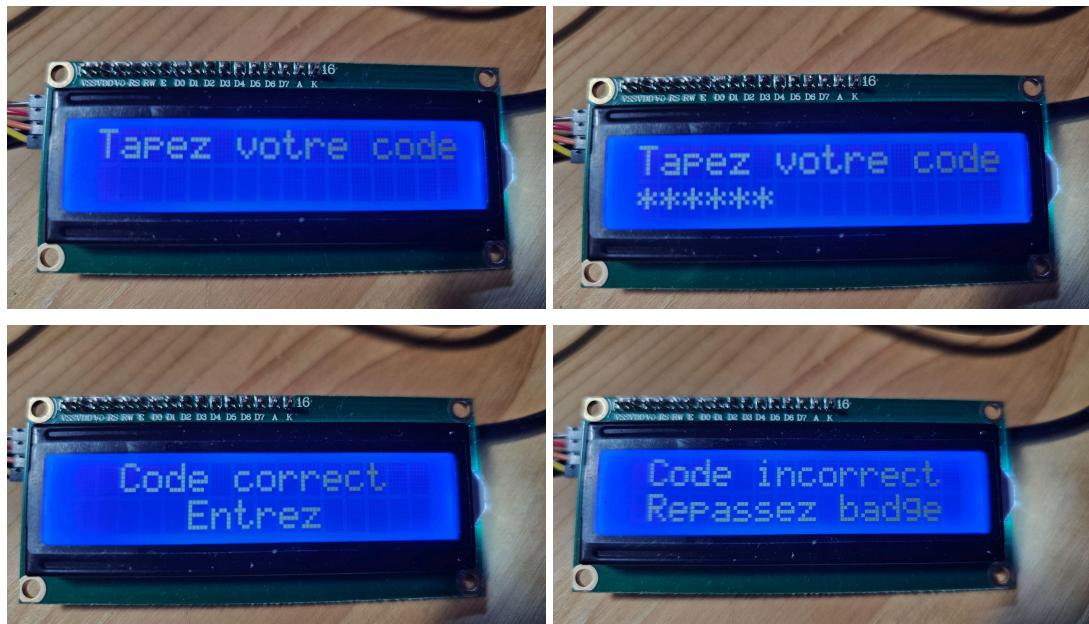
Le troisième programme met en place l'écran. Pour cela, l'écran affiche en permanence "Passez votre badge". Lorsqu'une carte ou un badge est lu, si l'UID est correct, l'écran affiche le code personnel de la carte. Sinon, l'écran affiche une phrase d'erreur.



Le quatrième programme est presque final. En effet, ce programme comporte tout ce que l'on a fait jusqu'ici : la lecture du mot de passe contenu dans la carte, l'affichage à l'écran, les LEDs et le buzzer. Cependant, nous allons ajouter l'obligation de taper le mot de passe avant de donner l'accès.

Lorsqu'un UID correct est détectée, le mot de passe est lu depuis la carte et stocké temporairement dans l'Arduino. Ensuite, l'utilisateur pourra taper son mot de passe au clavier numérique en utilisant la touche '*' pour recommencer la saisie et la touche '#' pour confirmer l'écriture. Au fur et à mesure que l'utilisateur écrit, des '*' s'ajoutent à chaque caractère entré. Si le mot de passe entré est correct, l'accès est ouvert temporairement.

Si le mot de passe entré est incorrect, le système obligera l'utilisateur à repasser son badge pour recommencer la saisie de son mot de passe.



Le dernier programme est l'ajout du servomoteur.

Au démarrage du programme, le servo se place en position initiale.

Lorsque l'accès est ouvert, le servo s'ouvre à 180° durant la durée d'accès, puis revient en position initiale.



A gauche, la position initiale, donc fermée.

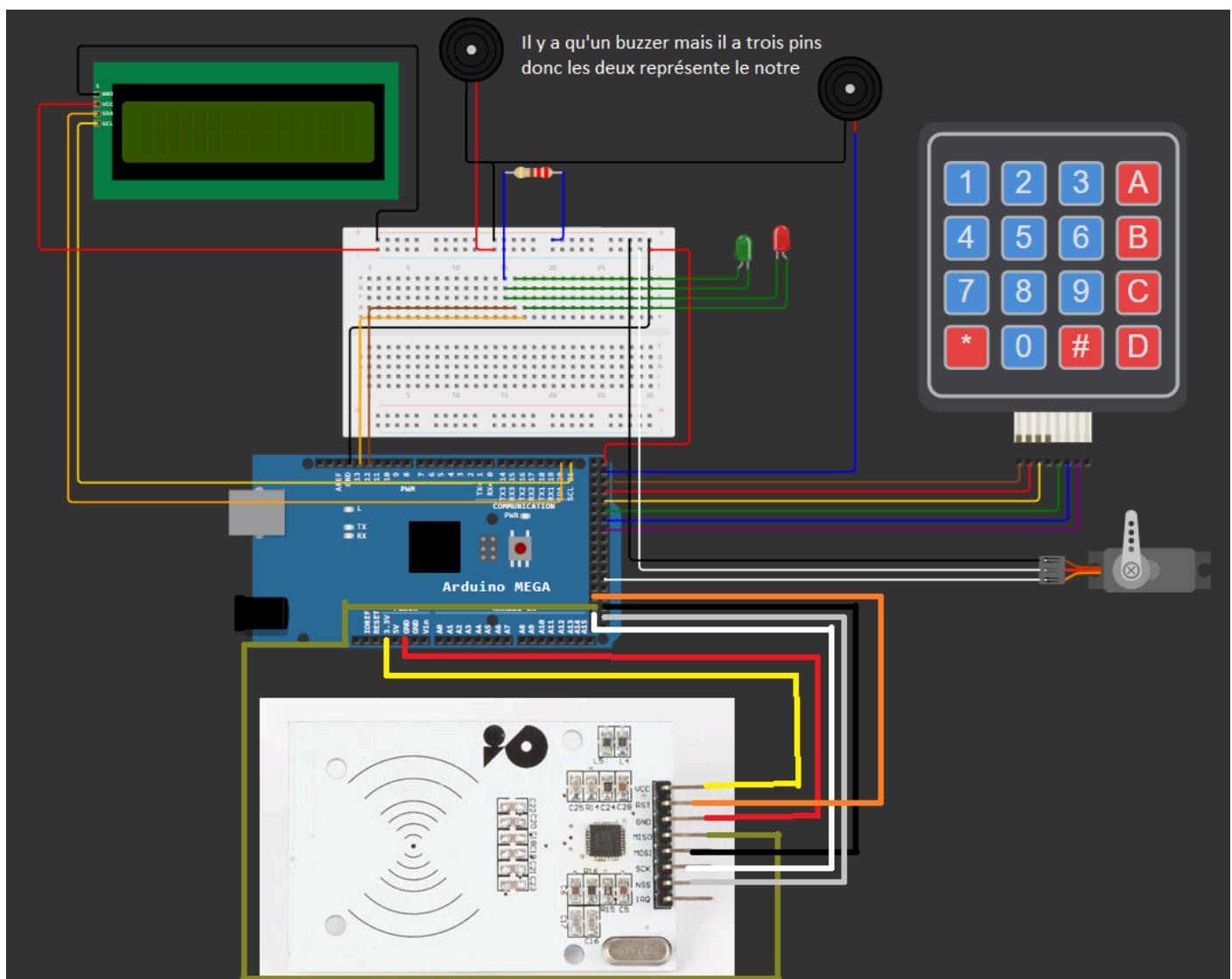
A droite, la position ouverte.

Conclusion :

Ce projet nous a permis d'explorer et de maîtriser plusieurs technologies et composants électroniques en intégrant des fonctionnalités avancées autour du module RFID WPI405, d'un écran LCD 16x2 I²C, de LED, d'un buzzer, d'un servomoteur, et d'un keypad.

Ce projet nous a permis de développer des compétences en électronique, en programmation Arduino, et en communication entre plusieurs composants. Il constitue une base solide pour des applications concrètes comme les systèmes de contrôle d'accès, tout en laissant la possibilité de l'améliorer pour des projets futurs, comme l'ajout d'une base de données externe ou la communication sans fil avec d'autres appareils.

En somme, ce projet a été un véritable apprentissage pratique et une expérience enrichissante dans le domaine des systèmes embarqués et de la sécurité électronique.



Archives

Programme partie 1 :

```
/* SAE Borowec Dhuiege Paternotte, octobre-novembre 2024, lecture données cartes
 * Gestion de cartes RFID
 * Lecture des cartes avec affichage des informations sur le terminal
 */

#include <SPI.h> //Librairie pour communiquer avec le lecteur de carte
#include <MFRC522.h> //Librairie du lecteur de carte
#define SS_PIN 53 //Pins du lecteur de cartes
#define RST_PIN 48

MFRC522 mfrc522(SS_PIN, RST_PIN); //Déclaration du lecteur de cartes

MFRC522::MIFARE_Key key;

byte uidPICC[4]; //Dans ce tableau sera stocké l'ID de la carte lue

void setup() { //Fonction setup
    Serial.begin(9600); //Commencer la liaison série
    SPI.begin(); //Et la liaison SPI
    mfrc522.PCD_Init(); //Initialiser le lecteur de cartes

    for (byte i = 0; i < 6; i++) {
        key.keyByte[i] = 0xFF;
    }
}

void loop() {

    MFRC522::MIFARE_Key key;
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

    //variables nécessaires
    byte block; //Ligne de la donnée lue
    byte len; //Longueur de la donnée lue
    MFRC522::StatusCode status;

    if ( ! mfrc522.PICC_IsNewCardPresent()) //Ne rien faire tant qu'il n'y a pas une carte sur
    le lecteur
        return; //Pour cela, le programme principal est relancé en boucle du début s'il n'y a pas
    de carte

    if ( ! mfrc522.PICC_ReadCardSerial()) //Vérifie si l'ID de la carte a bien été lue
        return; //Pour cela, la même technique est utilisée qu'au dessus

    Serial.println(F("**Carte trouvée : **")); //Afficher qu'une carte a bien été trouvée

    mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid)); //Affichage de quelques infos sur
```

```

la carte
//mfrc522.PICC_DumpToSerial(&(mfrc522.uid)); //uncomment this to see all blocks in
hex

Serial.print(F("Ligne où se cache le code: "));

byte buffer1[18]; //Buffer dans lequel sera stocké la ligne lue
block = 26; //La ligne lue sera la 26
len = 18; //On lie 16 caractères (2 en plus pour un bon fonctionnement)

status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 26,
&key, &(mfrc522.uid)); //Vérifier si la ligne 26 est disponible à la lecture
if (status != MFRC522::STATUS_OK) { //Si c'est pas le cas
    Serial.print(F("Authentication failed: ")); //Alors l'afficher sur le terminal
    Serial.println(mfrc522.GetStatusCodeName(status));
    return; //Et recommencer le programme principal
}

status = mfrc522.MIFARE_Read(block, buffer1, &len); //Lire la ligne
if (status != MFRC522::STATUS_OK) { //Si une erreur de lecture a eu lieu
    Serial.print(F("Reading failed: ")); //Alors l'afficher sur le terminal
    Serial.println(mfrc522.GetStatusCodeName(status));
    return; //Et recommencer le programme principal
}

//Pour chacun des 16 caractères de la ligne
for (uint8_t i = 0; i < 16; i++)
{
    if (buffer1[i] != 32) //Si le caractère n'est pas un espace
    {
        Serial.write(buffer1[i]); //ALors écrire le caractère sur le terminal
    }
}

Serial.println(F("\n**Lecture terminée**\n"));

delay(1000); //Petite attente avant nouvelle boucle du programme principal

mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto1();
}

```

Programme partie 2 :

```
/* SAE Borowec Dhuiege Paternotte, octobre-novembre 2024, programme complet
 * Gestion de cartes RFID
 * Lecture de leur ID.
 * Info ID connu ou pas via 2 LEDs et un buzzer
 */

#include <SPI.h> //Librairie pour communiquer avec le lecteur de carte
#include <MFRC522.h> //Librairie du lecteur de carte

const byte BUZZ = 23, LEDR = 12, LEDV = 13; //Déclaration des broches
MFRC522::StatusCode status;

#define SS_PIN 53 //Pins du lecteur de cartes
#define RST_PIN 48

byte CartesValides[10][4] = {{0x70,0x68,0xDB,0x3C},{0x19,0xB4,0xDB,0x3C}}; //ID des 2
cartes autorisées
bool FlagTrouve = 0; //Flag pour le programme principal
MFRC522 rfid(SS_PIN, RST_PIN); //Déclaration du lecteur de carte
byte uidPICC[4]; //Dans ce tableau sera stocké l'ID de la carte lue

void LEDRouge() { //Fonction pour la gestion de la LED rouge et du buzzer
    digitalWrite(LEDR,1); //Allumer la LED et faire sonner le buzzer
    digitalWrite(BUZZ,1);
    delay(50);
    digitalWrite(BUZZ,0); //Après un petit temps, stopper le buzzer pour faire un bip court
    delay(50);
    digitalWrite(BUZZ,1); //Puis relancer le buzzer pour un bip long
    delay(400);
    digitalWrite(BUZZ,0); //Arrêter le buzzer
    delay(1500);
    digitalWrite(LEDR,0); //Après un long temps, éteindre la LED
}

void setup() { //Fonction setup
    pinMode(LEDV,OUTPUT); //Mise en place des sorties
    pinMode(LEDR,OUTPUT); //Mise en place des sorties
    pinMode(BUZZ,OUTPUT); //Mise en place des sorties
    rfid.PCD_Init(); //Initialiser le lecteur de cartes
    Serial.begin(9600); //Commencer la liaison série
    SPI.begin(); //Et la liaison SPI
}

void loop() { //Programme principal
    MFRC522::MIFARE_Key key;
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

    if ( ! rfid.PICC_IsNewCardPresent()) //Ne rien faire tant qu'il n'y a pas une carte sur le
lecteur
        return; //Pour cela, le programme principal est relancé en boucle du début s'il n'y a pas
de carte
```

```

if ( ! rfid.PICC_ReadCardSerial() ) //Vérifie si l'ID de la carte a bien été lue
    return; //Pour cela, la même technique est utilisée qu'au dessus

Serial.print(F("PICC type: ")); //Affichage du type de carte lue
MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
Serial.println(rfid.PICC_GetTypeName(piccType));

//Vérification du type de la carte
if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI && piccType !=
MFRC522::PICC_TYPE_MIFARE_1K && piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("Your tag is not of type MIFARE Classic.")); //Si le type n'est pas le bon,
    ceci est précisé sur le terminal
    return; //Recommencer le programme principal
}
FlagTrouve = 0; //Reset du flag trouvé
for (int i=1;i<=10;i++) { //Pour chacune des 10 cartes autorisés (actuellement, il n'y en a que 3)
    if (rfid.uid.uidByte[0] == CartesValides[i-1][0] && rfid.uid.uidByte[1] ==
    CartesValides[i-1][1] && rfid.uid.uidByte[2] == CartesValides[i-1][2] && rfid.uid.uidByte[3] ==
    CartesValides[i-1][3]) { //Si l'ID entier correspond à une carte autorisée
        FlagTrouve = 1; //Mettre le flag trouvé à 1
    }
}
if (FlagTrouve==0) { //Si le flag est à 0 (aucune carte autorisée reconnue)
    LEDRouge(); //Lancer la séquence de la LED et du buzzer
}
else { //Si le flag est à 1 (une carte autorisée reconnue)
    MFRC522::MIFARE_Key key;
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
    digitalWrite(LEDV1,1); //Allumer la LED verte
    delay(3000); //Attendre 3s
    digitalWrite(LEDV0,0); //Éteindre la LED verte
}
rfid.PICC_HaltA(); // Halt PICC
rfid.PCD_StopCrypto1(); // Stop encryption on PCD
}

void printHex(byte *buffer, byte bufferSize) {
    for (byte i = 0; i < bufferSize; i++) {
        Serial.print(buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(buffer[i], HEX);
    }
}

```

Programme partie 3 :

```
/* SAE Borowec Dhuiege Paternotte, octobre-novembre 2024, partie 3
 * Gestion de cartes RFID
 * Lecture de leur ID et de leur contenu, affichage du mot de passe à l'écran
 * Le mot de passe est stocké dans la carte, ce qui permet un mot de passe unique par
carte
 * Code des cartes
 * 1 : 02880666
 * 2 : 02200777
 */

#include <SPI.h> //Librairie pour communiquer avec le lecteur de carte
#include <MFRC522.h> //Librairie du lecteur de carte
#include <LiquidCrystal_I2C.h> //Librairie pour un écran LCD I²C

const byte LEDR = 12, LEDV = 13, BUZZ = 23; //Déclaration des broches des composants
byte mdp[18], len = 18;
String code = "";

LiquidCrystal_I2C lcd(0x27,16,2); //Déclaration de l'écran, adresse 0x27, taille 16
colonnes par 2 lignes
MFRC522::StatusCode status;

#define SS_PIN 53 //Pins du lecteur de cartes
#define RST_PIN 48

byte CartesValides[10][4] = {{0x70,0x68,0xDB,0x3C},{0x19,0xB4,0xDB,0x3C}}; //ID des 2
cartes autorisées
bool FlagTrouve = 0; //Flag pour le programme principal

MFRC522 rfid(SS_PIN, RST_PIN); //Déclaration du lecteur de cartes

byte uidPICC[4]; //Dans ce tableau sera stocké l'ID de la carte lue

void LEDRouge() { //Fonction pour la gestion de la LED rouge et du buzzer
  digitalWrite(LEDR,1); //Allumer la LED et faire sonner le buzzer
  digitalWrite(BUZZ,1);
  delay(50);
  digitalWrite(BUZZ,0); //Après un petit temps, stopper le buzzer pour faire un bip court
  delay(50);
  digitalWrite(BUZZ,1); //Puis relancer le buzzer pour un bip long
  delay(400);
  digitalWrite(BUZZ,0); //Arrêter le buzzer
  delay(1500);
  digitalWrite(LEDR,0); //Après un long temps, éteindre la LED
}

void EcranComplet (String Ligne1, String Ligne2) { //Fonction pour actualiser le texte à
l'écran
  lcd.clear(); //Effacer l'écran
  lcd.print(Ligne1); //Afficher le contenu de la variable ligne1 sur cette ligne
  lcd.setCursor(0,1); //Se placer sur la ligne en dessous
  lcd.print(Ligne2); //Afficher le contenu de la variable ligne2 sur cette ligne
```

```

}

void setup() { //Fonction setup
    pinMode(LED_R,OUTPUT); //Mise en place des sorties
    pinMode(LED_V,OUTPUT);
    pinMode(BUZZ,OUTPUT);
    rfid.PCD_Init(); //Initialiser le lecteur de cartes
    lcd.init(); //Initialiser l'écran
    lcd.backlight(); //Allumer le rétroéclairage de l'écran
    Serial.begin(9600); //Commencer la liaison série
    SPI.begin(); //Et la liaison SPI
    EcranComplet(" Passez votre", " badge"); //Demander le badge à l'écran
}

void loop() { //Programme principal
    MFRC522::MIFARE_Key key;
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

    if ( ! rfid.PICC_IsNewCardPresent()) //Ne rien faire tant qu'il n'y a pas une carte sur le
    lecteur
        return; //Pour cela, le programme principal est relancé en boucle du début s'il n'y a pas
    de carte

    if ( ! rfid.PICC_ReadCardSerial()) //Vérifie si l'ID de la carte a bien été lue
        return; //Pour cela, la même technique est utilisée qu'au dessus

    Serial.print(F("PICC type: ")); //Affichage du type de carte lue
    MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
    Serial.println(rfid.PICC_GetTypeName(piccType));

    //Vérification du type de la carte
    if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI && piccType !=
    MFRC522::PICC_TYPE_MIFARE_1K && piccType !=
    MFRC522::PICC_TYPE_MIFARE_4K) {
        Serial.println(F("Your tag is not of type MIFARE Classic.")); //Si le type n'est pas le bon,
        ceci est précisé sur le terminal
        return; //Recommencer le programme principal
    }
    FlagTrouve = 0; //Reset du flag trouvé
    for (int i=1;i<=10;i++) { //Pour chacune des 10 cartes autorisés (actuellement, il n'y en a
    que 3)
        if (rfid.uid.uidByte[0] == CartesValides[i-1][0] && rfid.uid.uidByte[1] ==
        CartesValides[i-1][1] && rfid.uid.uidByte[2] == CartesValides[i-1][2] && rfid.uid.uidByte[3] ==
        CartesValides[i-1][3]) { //Si l'ID entier correspond à une carte autorisée
            FlagTrouve = 1; //Mettre le flag trouvé à 1
        }
    }
    if (FlagTrouve==0) { //Si le flag est à 0 (aucune carte autorisée reconnue)
        EcranComplet(" Badge inconnu","", ""); //Affichage de l'erreur
        LEDRouge(); //Lancer la séquence de la LED et du buzzer
        EcranComplet(" Passez votre", " badge"); //Puis, redemander un badge
    }
    else { //Si le flag est à 1 (une carte autorisée reconnue)
        MFRC522::MIFARE_Key key;

```

```

for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
status = rfid.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 26, &key,
&(rfid.uid)); //Confirmer que la ligne 26 de la carte est disponible à la lecture (ligne dans
laquelle le code est stocké entre des nombres aléatoires)
if (status != MFRC522::STATUS_OK) { //Si la ligne n'est pas lisible
Serial.print(F("Authentication failed: ")); //Informer l'erreur sur le terminal
Serial.println(rfid.GetStatusCodeName(status));
EcranComplet(" Erreur lecture", " Repassez badge"); //Et à l'écran
return; //Recommencer le programme principal
}
status = rfid.MIFARE_Read(26, mdp, &len); //lire la ligne 26 de la carte
if (status != MFRC522::STATUS_OK) { //Si la lecture a entraîné une erreur
Serial.print(F("Reading failed: ")); //Informer l'erreur sur le terminal
Serial.println(rfid.GetStatusCodeName(status));
EcranComplet(" Erreur lecture", " Repassez badge"); //Et à l'écran
return; //Recommencer le programme principal
}
code = ""; //Réinitialiser la variable code
for (uint8_t i = 4; i < 12; i++) //Pour les caractères 4 à 11 de la ligne
{
if (mdp[i] != 32) //Si le caractère n'est pas un espace
{
code = code + (char)mdp[i]; //Mettre le caractère dans le code
} //Au final, sur une ligne remplie de 16 chiffres, nous récupérons les chiffres de la 4e à
la 11e place, ce qui formera un mot de passe à 8 chiffres
} //Les autres chiffres ne composent pas le mot de passe et servent de diversion pour
faire croire que le mot de passe fait 16 chiffres
EcranComplet("Votre code :",code); //L'écran affiche le code qui a été récupéré dans la
carte
delay(3000);
EcranComplet(" Passez votre", " badge"); //Redemander de passer le badge
}

rfid.PICC_HaltA(); // Halt PICC
rfid.PCD_StopCrypto1(); // Stop encryption on PCD
}

void printHex(byte *buffer, byte bufferSize) {
for (byte i = 0; i < bufferSize; i++) {
Serial.print(buffer[i] < 0x10 ? " 0" : " ");
Serial.print(buffer[i], HEX);
}
}

```

Programme partie 4 :

```
/* SAE Borowec Dhuiege Paternotte, octobre-novembre 2024, Partie 4
 * Programme quasi complet : sans le servomoteur
 * Gestion de cartes RFID
 * Lecture de leur ID et de leur contenu, demande de mot de passe stocké dans la carte
 * Code des cartes
 * 1 : 02880666
 * 2 : 02200777
 */

#include <SPI.h> //Librairie pour communiquer avec le lecteur de carte
#include <MFRC522.h> //Librairie du lecteur de carte
#include <LiquidCrystal_I2C.h> //Librairie pour un écran LCD I²C
#include <Keypad.h> //Librairie pour clavier numérique

LiquidCrystal_I2C lcd(0x27,16,2); //Déclaration de l'écran, adresse 0x27, taille 16
colonnes par 2 lignes

const byte LEDR = 12, LEDV = 13, BUZZ = 23; //Déclaration des broches des composants

char cara;
String code = "",ecrit="";
byte mdp[18], len = 18;
MFRC522::StatusCode status;

#define ROWS 4 //Le clavier numérique contient 4 lignes et 3 colonnes
#define COLS 3
const char kp4x3Keys[ROWS][COLS] = {{'1', '2', '3'}, {'4', '5', '6'}, {'7', '8', '9'}, {'*', '0', '#'}};
/* Structure du clavier :
 * 7 8 9
 * 4 5 6
 * 1 2 3
 * * 0 # */

byte rowKp4x3Pin [4] = {37, 35, 33, 31}, colKp4x3Pin [3] = {29, 27, 25}; //Broches du
clavier
Keypad kp4x3 = Keypad(makeKeymap(kp4x3Keys), rowKp4x3Pin, colKp4x3Pin, ROWS,
COLS); //Déclaration du clavier

#define SS_PIN 53 //Pins du lecteur de cartes
#define RST_PIN 48

byte CartesValides[10][4] = {{0x70,0x68,0xDB,0x3C},{0x19,0xB4,0xDB,0x3C}}; //ID des 2
cartes autorisées
bool FlagTrouve = 0, sortie = 0; //Flags pour le programme principal

MFRC522 rfid(SS_PIN, RST_PIN); //Déclaration du lecteur de cartes

byte uidPICC[4]; //Dans ce tableau sera stocké l'ID de la carte lue

// Fonction pour l'acquisition d'une touche du clavier
char readKp4x3() {
    char customKey = kp4x3.getKey(); //Récupérer la touche appuyée
    if (customKey) { //Si UNE touche est appuyée
```

```

        return(customKey); //Renvoyer la touche appuyée
    }
    else return(0x00); //Sinon, envoyer le caractère 'null'
}

void LEDRouge() { //Fonction pour la gestion de la LED rouge et du buzzer
    digitalWrite(LEDR,1); //Allumer la LED et faire sonner le buzzer
    digitalWrite(BUZZ,1);
    delay(50);
    digitalWrite(BUZZ,0); //Après un petit temps, stopper le buzzer pour faire un bip court
    delay(50);
    digitalWrite(BUZZ,1); //Puis relancer le buzzer pour un bip long
    delay(400);
    digitalWrite(BUZZ,0); //Arrêter le buzzer
    delay(1500);
    digitalWrite(LEDR,0); //Après un long temps, éteindre la LED
}

void EcranComplet (String Ligne1, String Ligne2) { //Fonction pour actualiser le texte à
l'écran
    lcd.clear();      //Effacer l'écran
    lcd.print(Ligne1); //Afficher le contenu de la variable ligne1 sur cette ligne
    lcd.setCursor(0,1); //Se placer sur la ligne en dessous
    lcd.print(Ligne2); //Afficher le contenu du la variable ligne2 sur cette ligne
}

void setup() { //Fonction setup
    pinMode(LEDR,OUTPUT); //Mise en place des sorties
    pinMode(LEDV,OUTPUT);
    pinMode(BUZZ,OUTPUT);
    rfid.PCD_Init(); //Initialiser le lecteur de cartes
    lcd.init(); //Initialiser l'écran
    lcd.backlight(); //Allumer le rétroéclairage de l'écran
    Serial.begin(9600); //Commencer la liaison série
    SPI.begin(); //Et la liaison SPI
    EcranComplet(" Passez votre"," badge"); //Demander le badge à l'écran
}

void loop() { //Programme principal
    MFRC522::MIFARE_Key key;
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

    if ( ! rfid.PICC_IsNewCardPresent()) //Ne rien faire tant qu'il n'y a pas une carte sur le
lecteur
        return; //Pour cela, le programme principal est relancé en boucle du début s'il n'y a pas
de carte

    if ( ! rfid.PICC_ReadCardSerial()) //Vérifie si l'ID de la carte a bien été lue
        return; //Pour cela, la même technique est utilisée qu'au dessus

    Serial.print(F("PICC type: ")); //Affichage du type de carte lue
    MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
    Serial.println(rfid.PICC_GetTypeName(piccType));
}

```

```

//Vérification du type de la carte
if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI && piccType !=
MFRC522::PICC_TYPE_MIFARE_1K && piccType !=
MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("Your tag is not of type MIFARE Classic.")); //Si le type n'est pas le bon,
    ceci est précisé sur le terminal
    return; //Recommencer le programme principal
}
FlagTrouve = 0; //Reset du flag trouvé
for (int i=1;i<=10;i++) { //Pour chacune des 10 cartes autorisés (actuellement, il n'y en a
que 3)
    if (rfid.uid.uidByte[0] == CartesValides[i-1][0] && rfid.uid.uidByte[1] ==
CartesValides[i-1][1] && rfid.uid.uidByte[2] == CartesValides[i-1][2] && rfid.uid.uidByte[3] ==
CartesValides[i-1][3]) { //Si l'ID entier correspond à une carte autorisée
        FlagTrouve = 1; //Mettre le flag trouvé à 1
    }
}
if (FlagTrouve==0) { //Si le flag est à 0 (aucune carte autorisée reconnue)
    EcranComplet(" Badge inconnu","");
    LEDRouge(); //Lancer la séquence de la LED et du buzzer
    EcranComplet(" Passez votre badge"); //Puis, redemander un badge
}
else { //Si le flag est à 1 (une carte autorisée reconnue)
    code=""; //Réinitialisation de la variable code
    MFRC522::MIFARE_Key key;
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
    status = rfid.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 26, &key,
&(rfid.uid)); //Confirmer que la ligne 26 de la carte est disponible à la lecture (ligne dans
laquelle le code est stocké entre des nombres aléatoires)
    if (status != MFRC522::STATUS_OK) { //Si la ligne n'est pas lisible
        Serial.print(F("Authentication failed: ")); //Informer l'erreur sur le terminal
        Serial.println(rfid.GetStatusCodeName(status));
        EcranComplet(" Erreur lecture"," Repassez badge"); //Et à l'écran
        return; //Recommencer le programme principal
    }
    status = rfid.MIFARE_Read(26, mdp, &len); //lire la ligne 26 de la carte
    if (status != MFRC522::STATUS_OK) { //Si la lecture a entraîné une erreur
        Serial.print(F("Reading failed: ")); //Informer l'erreur sur le terminal
        Serial.println(rfid.GetStatusCodeName(status));
        EcranComplet(" Erreur lecture"," Repassez badge"); //Et à l'écran
        return; //Recommencer le programme principal
    }
    for (uint8_t i = 4; i < 12; i++) //Pour les caractères 4 à 11 de la ligne
    {
        if (mdp[i] != 32) //Si le caractère n'est pas un espace
        {
            code = code + (char)mdp[i]; //Mettre le caractère dans le code
        } //Au final, sur une ligne remplie de 16 chiffres, nous récupérons les chiffres de la 4e à
la 11e place, ce qui formera un mot de passe à 8 chiffres
    } //Les autres chiffres ne composent pas le mot de passe et servent de diversion pour
faire croire que le mot de passe fait 16 chiffres
    EcranComplet("Tapez votre code","");
    Lcd.setCursor(0,1); //Placer le curseur sur la 2e ligne
}

```

```

while (sortie==0) { //Tant que le flag sortie reste à 0
    cara = readKp4x3(); //Attendre que l'utilisateur appuie sur une touche du clavier
    if (cara=='*') { //Si la touche * est appuyée
        ecrit=""; //Effacer la variable écrit, dans laquelle le code en cours d'écriture est stocké
        EcranComplet("Tapez votre code","");
        EcranComplet("Tapez votre code","");
        complet
    }
    else if (cara=='#') { //Si la touche # est appuyée
        if (ecrit == code) { //Et que écrit (le code tapé par l'utilisateur) et le code de la carte
            sont identiques
                digitalWrite(LEDV1); //Allumer la LED verte
                EcranComplet(" Code correct"," Entrez"); //Informer que la porte est ouverte
                delay(3000); //Attendre 3s
                digitalWrite(LEDV0); //Eteindre la LED verte
                sortie = 1; //Mettre le flag sortie à 1 pour revenir au début du programme principal
        }
        else { //Cependant, si le code n'est pas correct
            EcranComplet(" Code incorrect"," Repassez badge"); //Informer que le code est
            incorrect et que l'utilisateur doit tout recommencer
            LEDRouge(); //Lancer la séquence de la LED et du buzzer
        }
        écrit=""; //Effacer la variable écrit
        sortie = 1; //Mettre le flag de sortie à 1
    }
    else if (cara!=0x00) { //Si un chiffre est appuyé
        lcd.print("*"); //Afficher sur l'écran un * supplémentaire
        écrit+=cara; //Ajouter le chiffre appuyé à écrit
    }
}
sortie = 0; //Une fois la boucle tant que sortie, remettre le flag à 0
EcranComplet(" Passez votre"," badge"); //Redemander de passer le badge
}

rfid.PICC_HaltA(); // Halt PICC
rfid.PCD_StopCrypto1(); // Stop encryption on PCD
}

void printHex(byte *buffer, byte bufferSize) {
    for (byte i = 0; i < bufferSize; i++) {
        Serial.print(buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(buffer[i], HEX);
    }
}

```

Programme partie 5 :

```
/* SAE Borowec Dhuiege Paternotte, octobre-novembre 2024, programme complet
 * Gestion de cartes RFID
 * Lecture de leur ID et de leur contenu, demande de mot de passe stocké dans la carte
 * Si mot de passe correct, ouverture de la porte (représentée par un servomoteur)
 * Code des cartes
 * 1 : 02880666
 * 2 : 02200777
 */

#include <Servo.h> //Librairie du servomoteur
#include <SPI.h> //Librairie pour communiquer avec le lecteur de carte
#include <MFRC522.h> //Librairie du lecteur de carte
#include <LiquidCrystal_I2C.h> //Librairie pour un écran LCD I²C
#include <Keypad.h> //Librairie pour clavier numérique

LiquidCrystal_I2C lcd(0x27,16,2); //Déclaration de l'écran, adresse 0x27, taille 16
colonnes par 2 lignes
Servo myservo; //Librairie renommée myservo

const byte LEDR = 12, LEDV = 13, BUZZ = 23, SERVO = 45; //Déclaration des broches
des composants

char cara;
String code = "",ecrit="";
byte mdp[18], len = 18;
MFRC522::StatusCode status;

#define ROWS 4 //Le clavier numérique contient 4 lignes et 3 colonnes
#define COLS 3
const char kp4x3Keys[ROWS][COLS] = {{'1', '2', '3'}, {'4', '5', '6'}, {'7', '8', '9'}, {'*', '0', '#'}};
/* Structure du clavier :
 * 7 8 9
 * 4 5 6
 * 1 2 3
 * * 0 # */

byte rowKp4x3Pin [4] = {37, 35, 33, 31}, colKp4x3Pin [3] = {29, 27, 25}; //Broches du
clavier
Keypad kp4x3 = Keypad(makeKeymap(kp4x3Keys), rowKp4x3Pin, colKp4x3Pin, ROWS,
COLS); //Déclaration du clavier

#define SS_PIN 53 //Pins du lecteur de cartes
#define RST_PIN 48

byte CartesValides[10][4] = {{0x70,0x68,0xDB,0x3C},{0x19,0xB4,0xDB,0x3C}}; //ID des 2
cartes autorisées
bool FlagTrouve = 0, sortie = 0; //Flags pour le programme principal

MFRC522 rfid(SS_PIN, RST_PIN); //Déclaration du lecteur de cartes

byte uidPICC[4]; //Dans ce tableau sera stocké l'ID de la carte lue

// Fonction pour l'acquisition d'une touche du clavier
```

```

char readKp4x3() {
    char customKey = kp4x3.getKey(); //Récupérer la touche appuyée
    if (customKey) { //Si UNE touche est appuyée
        return(customKey); //Renvoyer la touche appuyée
    }
    else return(0x00); //Sinon, envoyer le caractère 'null'
}

void LEDRouge() { //Fonction pour la gestion de la LED rouge et du buzzer
    digitalWrite(LED_R,1); //Allumer la LED et faire sonner le buzzer
    digitalWrite(BUZZ,1);
    delay(50);
    digitalWrite(BUZZ,0); //Après un petit temps, stopper le buzzer pour faire un bip court
    delay(50);
    digitalWrite(BUZZ,1); //Puis relancer le buzzer pour un bip long
    delay(400);
    digitalWrite(BUZZ,0); //Arrêter le buzzer
    delay(1500);
    digitalWrite(LED_R,0); //Après un long temps, éteindre la LED
}

void EcranComplet (String Ligne1, String Ligne2) { //Fonction pour actualiser le texte à l'écran
    lcd.clear(); //Effacer l'écran
    lcd.print(Ligne1); //Afficher le contenu de la variable ligne1 sur cette ligne
    lcd.setCursor(0,1); //Se placer sur la ligne en dessous
    lcd.print(Ligne2); //Afficher le contenu du la variable ligne2 sur cette ligne
}

void setup() { //Fonction setup
    pinMode(LED_R,OUTPUT); //Mise en place des sorties
    pinMode(LED_V,OUTPUT);
    pinMode(BUZZ,OUTPUT);
    pinMode(SERVO,OUTPUT);
    myservo.attach(SERVO); //Mise en place du servomoteur
    myservo.write(0); //Remettre le servomoteur sur sa position d'origine (porte fermée)
    rfid.PCD_Init(); //Initialiser le lecteur de cartes
    lcd.init(); //Initialiser l'écran
    lcd.backlight(); //Allumer le rétroéclairage de l'écran
    Serial.begin(9600); //Commencer la liaison série
    SPI.begin(); //Et la liaison SPI
    EcranComplet(" Passez votre"," badge"); //Demander le badge à l'écran
}

void loop() { //Programme principal
    MFRC522::MIFARE_Key key;
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

    if ( ! rfid.PICC_IsNewCardPresent()) //Ne rien faire tant qu'il n'y a pas une carte sur le lecteur
        return; //Pour cela, le programme principal est relancé en boucle du début s'il n'y a pas de carte

    if ( ! rfid.PICC_ReadCardSerial()) //Vérifie si l'ID de la carte a bien été lue

```

```

return; //Pour cela, la même technique est utilisée qu'au dessus

Serial.print(F("PICC type: ")); //Affichage du type de carte lue
MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
Serial.println(rfid.PICC_GetTypeName(piccType));

//Vérification du type de la carte
if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI && piccType != MFRC522::PICC_TYPE_MIFARE_1K && piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("Your tag is not of type MIFARE Classic.")); //Si le type n'est pas le bon,
    ceci est précisé sur le terminal
    return; //Recommencer le programme principal
}
FlagTrouve = 0; //Reset du flag trouvé
for (int i=1;i<=10;i++) { //Pour chacune des 10 cartes autorisés (actuellement, il n'y en a que 3)
    if (rfid.uid.uidByte[0] == CartesValides[i-1][0] && rfid.uid.uidByte[1] == CartesValides[i-1][1] && rfid.uid.uidByte[2] == CartesValides[i-1][2] && rfid.uid.uidByte[3] == CartesValides[i-1][3]) { //Si l'ID entier correspond à une carte autorisée
        FlagTrouve = 1; //Mettre le flag trouvé à 1
    }
}
if (FlagTrouve==0) { //Si le flag est à 0 (aucune carte autorisée reconnue)
    EcranComplet(" Badge inconnu","");
    LEDRouge(); //Lancer la séquence de la LED et du buzzer
    EcranComplet(" Passez votre","");
    badge"); //Puis, redemander un badge
}
else { //Si le flag est à 1 (une carte autorisée reconnue)
    code=""; //Réinitialisation de la variable code
    MFRC522::MIFARE_Key key;
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
    status = rfid.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 26, &key,
&(rfid.uid)); //Confirmer que la ligne 26 de la carte est disponible à la lecture (ligne dans laquelle le code est stocké entre des nombres aléatoires)
    if (status != MFRC522::STATUS_OK) { //Si la ligne n'est pas lisible
        Serial.print(F("Authentication failed: "));
        //Informer l'erreur sur le terminal
        Serial.println(rfid.GetStatusCodeName(status));
        EcranComplet(" Erreur lecture","");
        Repassez badge");
        //Et à l'écran
        return; //Recommencer le programme principal
    }
    status = rfid.MIFARE_Read(26, mdp, &len); //lire la ligne 26 de la carte
    if (status != MFRC522::STATUS_OK) { //Si la lecture a entraîné une erreur
        Serial.print(F("Reading failed: "));
        //Informer l'erreur sur le terminal
        Serial.println(rfid.GetStatusCodeName(status));
        EcranComplet(" Erreur lecture","");
        Repassez badge");
        //Et à l'écran
        return; //Recommencer le programme principal
    }
    for (uint8_t i = 4; i < 12; i++) //Pour les caractères 4 à 11 de la ligne
    {
        if (mdp[i] != 32) //Si le caractère n'est pas un espace
        {
            code = code + (char)mdp[i]; //Mettre le caractère dans le code
        }
    }
    //Au final, sur une ligne remplie de 16 chiffres, nous récupérons les chiffres de la 4e à
}

```

```

la 11e place, ce qui formera un mot de passe à 8 chiffres
} //Les autres chiffres ne composent pas le mot de passe et servent de diversion pour
faire croire que le mot de passe fait 16 chiffres
EcranComplet("Tapez votre code","");
//L'écran demande à taper le code stocké dans la
carte
lcd.setCursor(0,1); //Placer le curseur sur la 2e ligne
while (sortie==0) { //Tant que le flag sortie reste à 0
cara = readKp4x3(); //Attendre que l'utilisateur appuie sur une touche du clavier
if (cara=='*') { //Si la touche * est appuyée
ecrit=""; //Effacer la variable écrit, dans laquelle le code en cours d'écriture est stocké
EcranComplet("Tapez votre code","");
//Effacer la 2e ligne pour retaper le code au
complet
}
else if (cara=='#') { //Si la touche # est appuyée
if (ecrit == code) { //Et que écrit (le code tapé par l'utilisateur) et le code de la carte
sont identiques
digitalWrite(LEDV,1); //Allumer la LED verte
myservo.write(180); //Et ouvrir la porte représentée par le servomoteur
EcranComplet(" Code correct"," Entrez"); //Informer que la porte est ouverte
delay(3000); //Attendre 3s
digitalWrite(LEDV,0); //Eteindre la LED verte
myservo.write(0); //Et fermer la porte
sortie = 1; //Mettre le flag sortie à 1 pour revenir au début du programme principal
}
else { //Cependant, si le code n'est pas correct
EcranComplet(" Code incorrect"," Repassez badge"); //Informer que le code est
incorrect et que l'utilisateur doit tout recommencer
LEDrouge(); //Lancer la séquence de la LED et du buzzer
}
ecrit=""; //Effacer la variable écrit
sortie = 1; //Mettre le flag de sortie à 1
}
else if (cara!=0x00) { //Si un chiffre est appuyé
lcd.print("*"); //Afficher sur l'écran un * supplémentaire
ecrit+=cara; //Ajouter le chiffre appuyé à écrit
}
}
sortie = 0; //Une fois la boucle tant que sortie, remettre le flag à 0
EcranComplet(" Passez votre"," badge"); //Redemander de passer le badge
}

rfid.PICC_HaltA(); // Halt PICC
rfid.PCD_StopCrypto1(); // Stop encryption on PCD
}

void printHex(byte *buffer, byte bufferSize) {
for (byte i = 0; i < bufferSize; i++) {
Serial.print(buffer[i] < 0x10 ? " 0" : " ");
Serial.print(buffer[i], HEX);
}
}

```