

Année 2016-2017

## **La sécurité 2.0**

Aurélien Burdot / Alexandre Andriesse

### **Project E-Vault**

Mantes la Ville, France



Professeur encadrant : M.Touchard



*« L'Homme et sa sécurité doivent constituer la première préoccupation de toute aventure technologique»*

Albert Einstein

# Table des matières

Introduction .....	5
1. Cahier des charges .....	6
2.Réalisation du châssis.....	6
2.1.Design du coffre.....	6
2.2.Machines et moyens utilisés.....	9
3.Choix des composants électroniques.....	12
3.1.Shield NFC .....	12
3.2.PinPad : le pavé numérique simpliste .....	15
3.3.Arduino Mega : la dev-Board par excellence .....	17
3.4.Servomoteur.....	20
3.5.Ecran LCD .....	21
4.Intelligence du système .....	23
5.Améliorations possibles : un grand avenir.....	27
6.Problèmes rencontrés.....	28
7.Conclusion .....	29
Index.....	30
8 Annexes.....	31
8.1.Projet_coffre.ino.....	31
8.2.Coffrefort.h.....	33
8.3.Coffrefort.cpp.....	34
8.4.Nfc.h.....	40
8.5.Nfc.cpp.....	41

## **Introduction**

Dans un monde où la criminalité augmente, où il est difficile de faire confiance à son propre voisin, les entreprises de sécurités grandissent. Tout le monde veut pouvoir cacher et sécuriser ses biens précieux, que ce soit bijoux, papier important ou affaires de grande valeur sentimentale. Bien qu'un bon système d'alarme de maison soit la meilleure solution, il reste cher et pas forcément facile d'installation. La solution facile reste le coffre-fort. D'un autre côté, lors d'un vol dans votre coffre il est difficile de savoir qui a pris quoi. Le E.Vault est la solution à ce problème, un concentré de technologie pour une meilleure sécurité, une facilité d'utilisation déconcertante.

Ce coffre-fort est un projet d'étude proposé par le CFA mécavenir et l'école d'ingénieur ISTY dans le cadre du cours « Arduino »

Ce document regroupe les différentes technologies et procédés de fabrication du E.Vault. Dans un premier temps, il vous sera présenté le cahier des charges et les différentes fonctionnalités du coffre. Par la suite, vous pourrez trouver les composantes technologiques du système ainsi que la réalisation mécanique par des procédés révolutionnaires. Pour finir vous trouverez une présentation du système d'intelligence du produit.

## **1. Cahier des charges**

Le but du projet est de réaliser un coffre intelligent sécurisé en NFC. Le projet est nommé E-Vault.

Le produit final aura les caractéristiques suivantes :

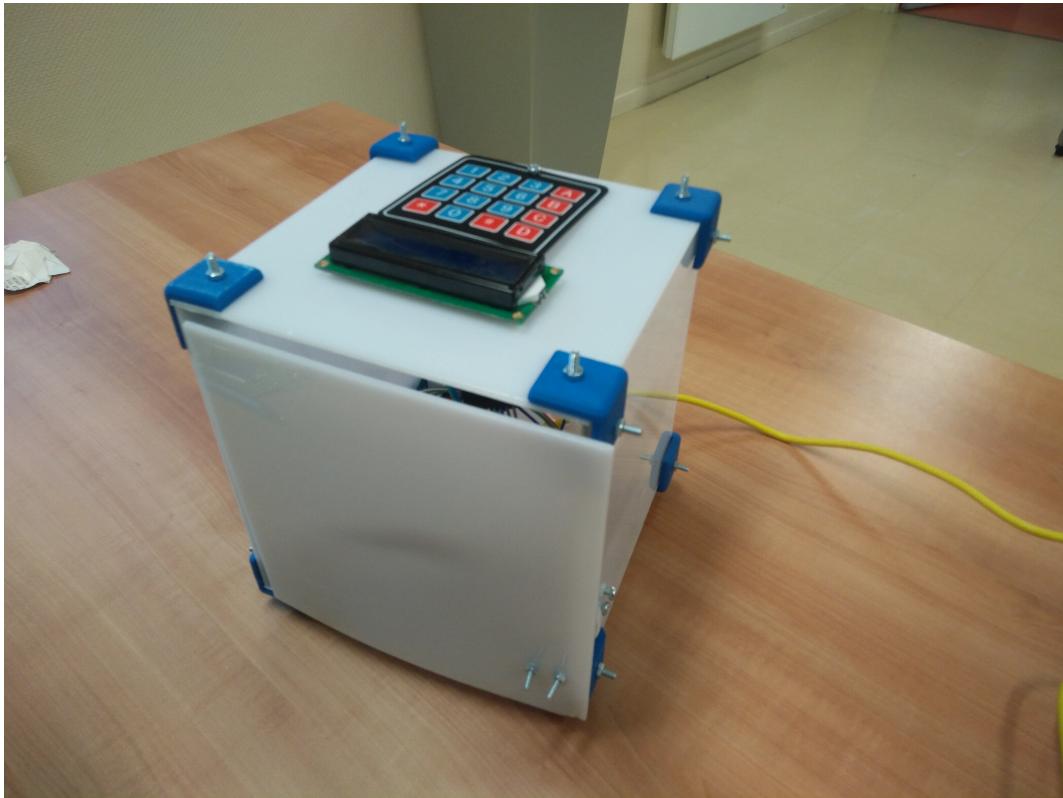
- Devra être équipé d'une sécurité NFC
- Devra pouvoir être utilisable par n'importe qui
- Devra supporter plusieurs utilisateurs et plusieurs badges
- Devra demander un mot de passe propre à chaque utilisateur
- Devra pouvoir enregistrer de nouveaux utilisateurs

## **2. Réalisation du châssis**

Notre projet consiste en la réalisation d'un prototype qui n'égalera en aucun cas les performances mécaniques d'un coffre-fort standard.

### **2.1. Design du coffre**

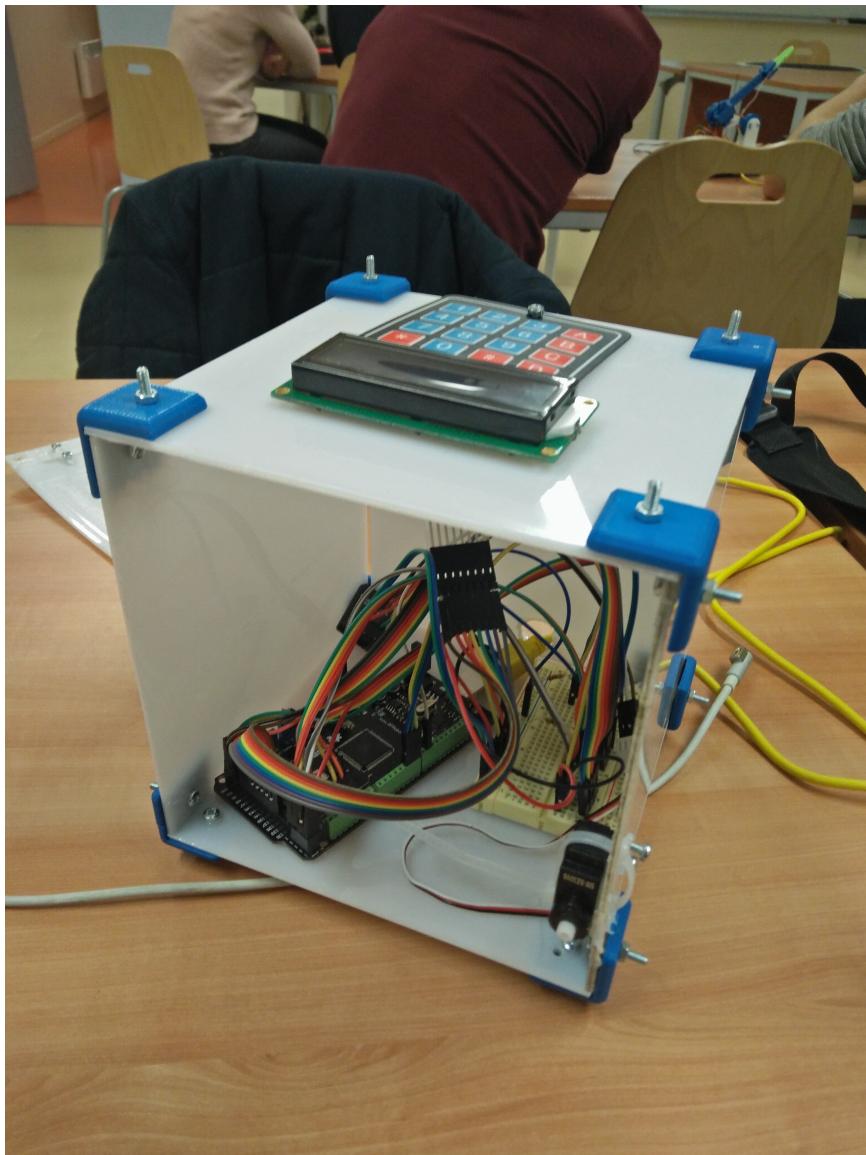
Le design de l'E-vault consiste en un cube de dimension 15x15x15cm. Il est composé d'une partie électronique et d'un espace utile. Le cube est réalisé en plaque d'acrylique opaline jointe entre elle par de petites pièces en ABS imprimées en 3D.



*Illustration 1: Prototype du E-Vault*

Ces pièces forment des coins extérieurs qui vont maintenir la structure à l'aide de boulons et d'écrous.

Le coffre s'ouvre par un de ses cotés. Une fois le coffre déverrouillé la plaque de devant, actionnée par un servomoteur, bascule vers la droite à 90° depuis son coin inférieur droit.



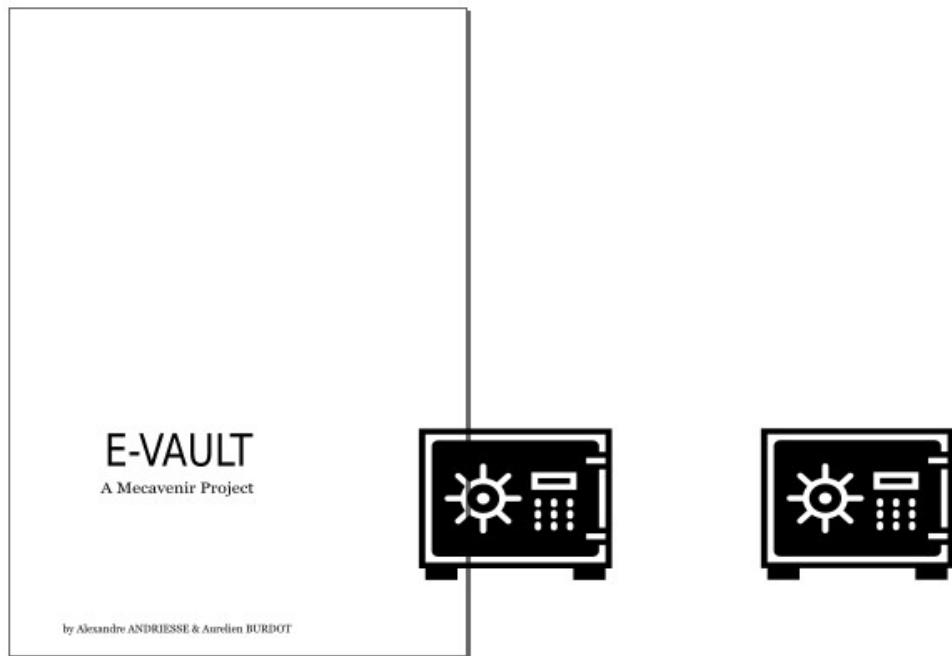
*Illustration 2: Intérieur du E-Vault*

Comme dit précédemment, E-Vault n'est qu'un prototype qui ne sert qu'à valider un modèle électronique et informatique. Les boulons apparents, les places d'acryliques, et la porte assez fragile ne sont pas des problèmes ici.

## **2.2. Machines et moyens utilisés**

Pour fabriquer le châssis du coffre, nous dûmes utiliser plusieurs machines de l'innolab. Mais avant d'utiliser tout appareil, il est obligatoire d'avoir un modèle informatique. Des fichiers en formats STL pour l'imprimante 3D et SVG (dessin vectoriel) pour la découpe laser.

Nous avons donc commencé à designer les différentes faces de l'E-vault à l'aide du logiciel open source Inkscape. Des contraintes de couleurs et d'épaisseur étaient à respecter afin que la découpe laser puisse correctement interpréter les consignes.



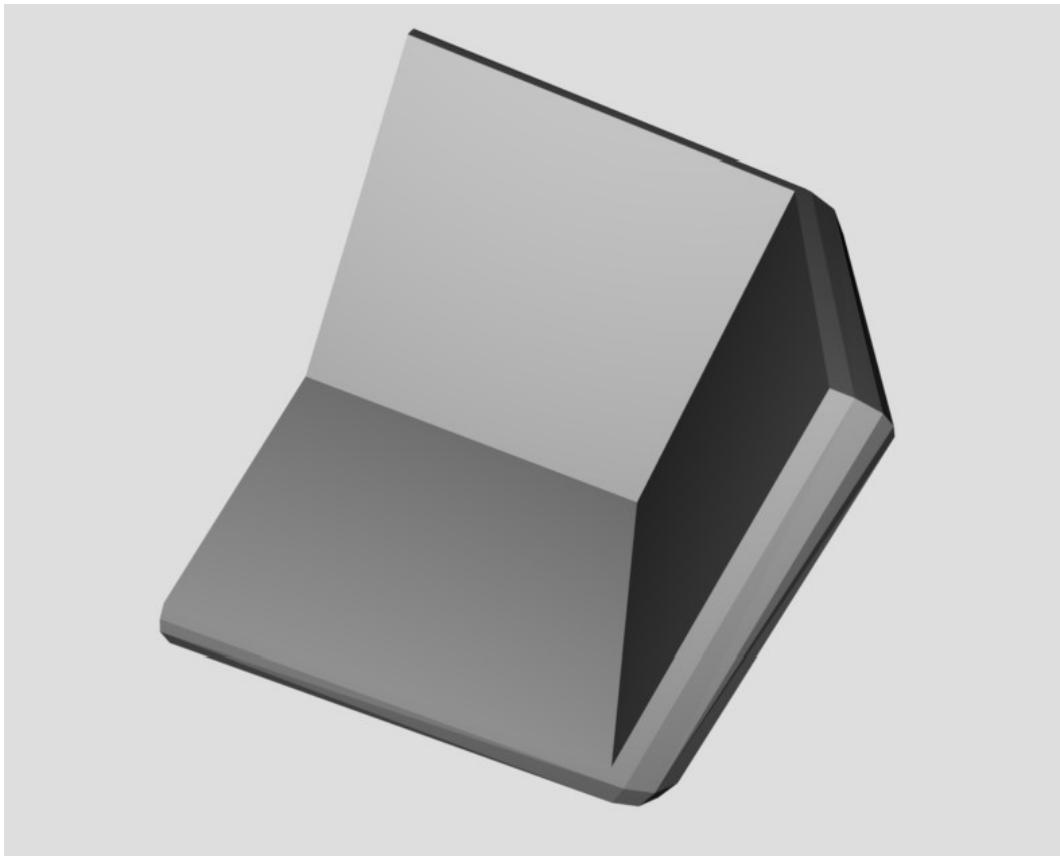
*Illustration 3: Dessin pour gravure sur Inkscape*

On distingue donc 2 couleurs, le noir pour la gravure et le rouge pour la découpe (on ne distingue pas le rouge car l'épaisseur du trait doit être de 0.01mm).

A l'aide monsieur Touchard, nous avons lancé une découpe sur une plaque de 50x50 cm d'acrylique opaline. Malgré quelques problèmes de paramétrage, nous réussîmes à obtenir 6 plaques blanches opaque.

En parallèle de la fabrication des plaques, nous réfléchissions à une manière simple mais propre de fixer les plaques et maintenir la structure. La première idée fut de fabriquer des coins intérieures mais cela faisais perdre de la place à l'intérieur du coffre. C'est donc tout naturellement que l'on a choisi de faire des coins extérieurs qui offraient deux avantages : Facile à installer et agis comme « support » de la boite.

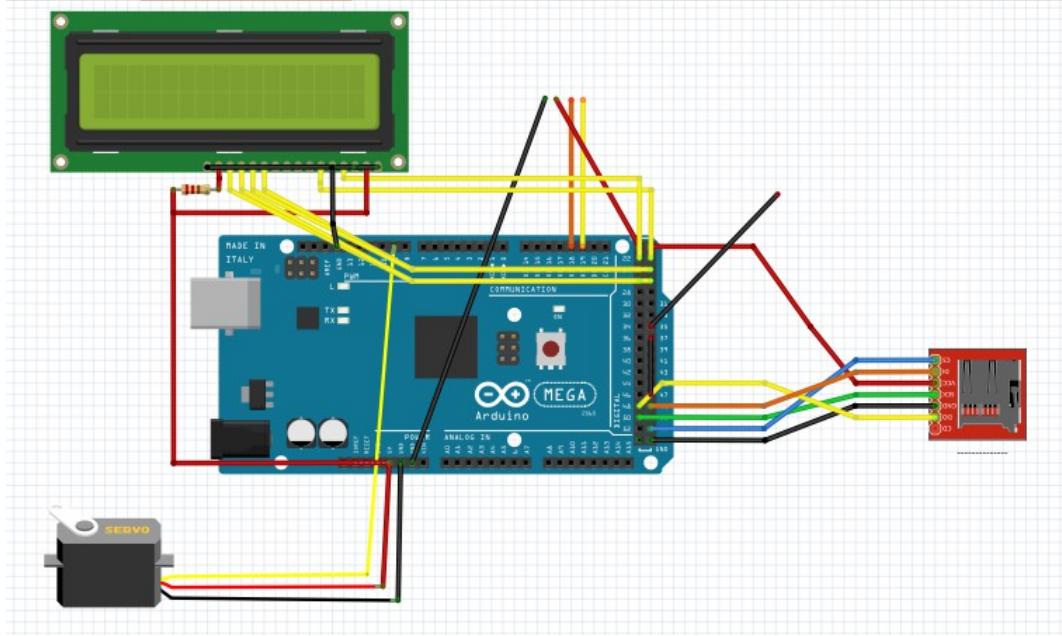
Apres une phase de mesure, nous commençâmes à dessiner les coins en trois dimensions sous le logiciel CATIA. Quand le design fut prêt et correspondant à nos exigences, nous l'avons converti en format exploitable par l'imprimante Zotrax.



*Illustration 4: Jointure en 3D sous catia*

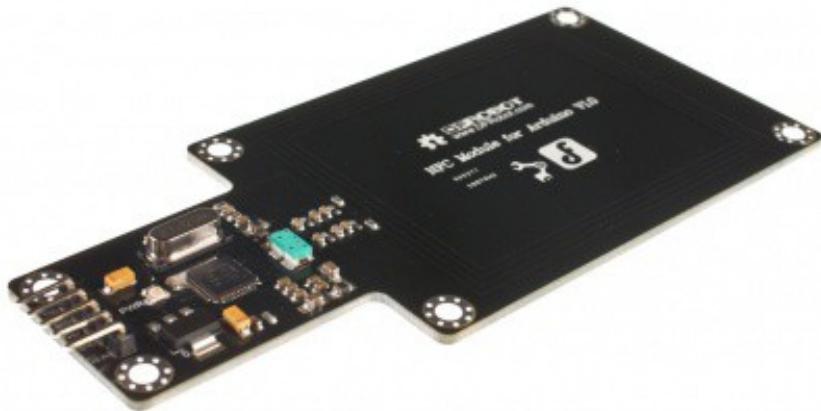
Il y a donc deux types de joint, les joints reliant 3 plaques et les joints en contact avec la porte. L'impression des 8 pièces priment 3h30 au total.

### **3. Choix des composants électroniques**



*Illustration 5: Schema de cablage*

### **3.1. *Shield NFC***



### *Illustration 6: Capteur NFC*

Le NFC, Near Field Communication, (ou communication en champ proche) est une technologie permettant d'échanger des données à moins de 10cm, entre deux appareils équipés de ce dispositif. Le NFC est intégré à la plupart de nos terminaux mobiles sous forme de puce, ainsi que sur certaines cartes de transport ou de paiement.



*Illustration 7: Tag NFC*

Le NFC a trois modes de fonctionnement différents :

### **Mode émulation de carte**

Le terminal mobile fonctionne comme une carte sans contact. La carte SIM du portable peut être utilisée pour stocker des informations chiffrées, et les sécuriser.

Exemples d'utilisations : paiement sans contact, gestion des coupons de réduction ou des points de fidélité dans un magasin.

## **Mode lecteur**

Le mobile équipé du NFC est capable de lire des « tags » (étiquettes électroniques), pour récolter des informations pratiques, ou pour lancer une action de manière automatique sur un smartphone.

Exemples d'utilisations : parcours dans un musée, automatisation d'une tâche (changer la sonnerie de son téléphone, ou lancer une application, à l'approche du tag NFC)

Un « tag NFC » est une étiquette électronique équipée de la technologie NFC. L'intérêt étant de pouvoir le programmer, de façon à envoyer une information aux appareils situés dans son champ d'action. On peut acheter des tags NFC, à programmer soi-même sur Internet. Pour programmer son tag NFC, il existe des applications mobiles comme « NFC Task Launcher », disponible sur le Google Play.

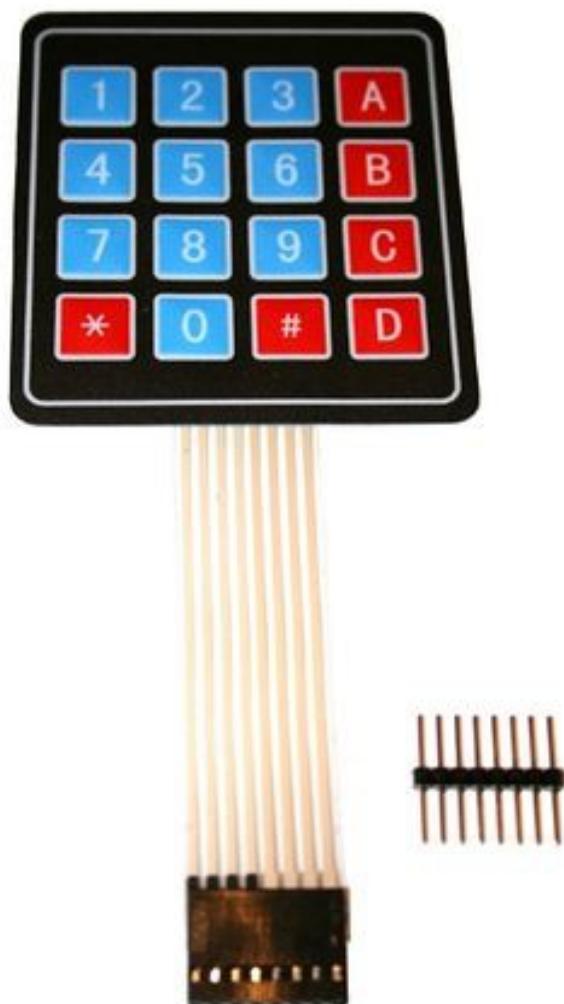
## **Mode peer to peer**

Ce mode de fonctionnement permet l'échange d'informations entre deux appareils équipés du NFC.

Exemple d'utilisation : un échange de photos entre une tablette et un Smartphone, l'ouverture des portières de sa voiture.

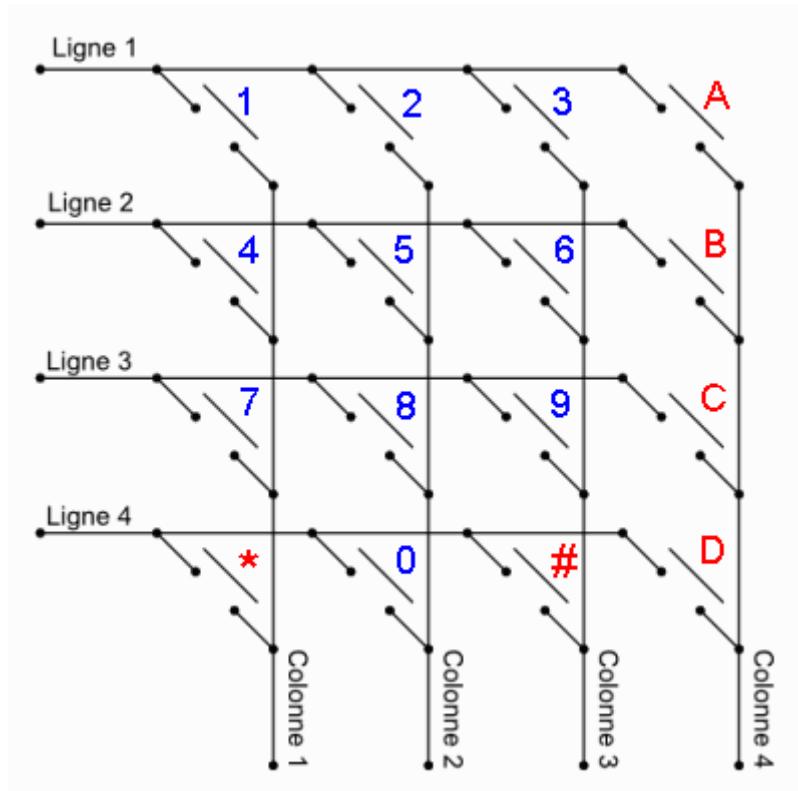
### **3.2. PinPad : le pavé numérique simpliste**

Il existe plusieurs technologies de pavé numérique. Nous avons choisi d'intégrer un pavé matriciel, non pas par sa beauté et son design, mais pour facilité d'intégration. En effet un petit clavier matriciel 4x4 répond intégralement à nos exigences et correspond au 3/4 des claviers de portes verrouillés par digicode.



*Illustration 8: Clavier Matriciel 4x4*

D'un point de vue esthétique, il s'agit d'un petit clavier de moins d'un millimètre d'épaisseur donc facilement collable aux parois du coffre. Il est composé de 4 lignes et 4 colonnes permettant d'avoir d'autres caractères que les chiffres classiques.



*Illustration 9: Schema interne d'un pavé matriciel*

Pour la partie technique de cette interface, nous retrouvons un système simple et ingénieux. Au lieu d'avoir tout simplement une sortie par touche, soit un total de 17 fils minimum, nous en avons que 8. Chaque lignes et colonnes sont connectées à un câble, donc ici 8 fils au total. D'un point de vue électronique, les touches sont des interrupteurs qui vont connecter un fil d'une colonne au fil d'une ligne. Le programme de pilotage envoie par balayage des signaux logiques sur les lignes et en parallèle lit la valeur des colonnes (ou inversement).

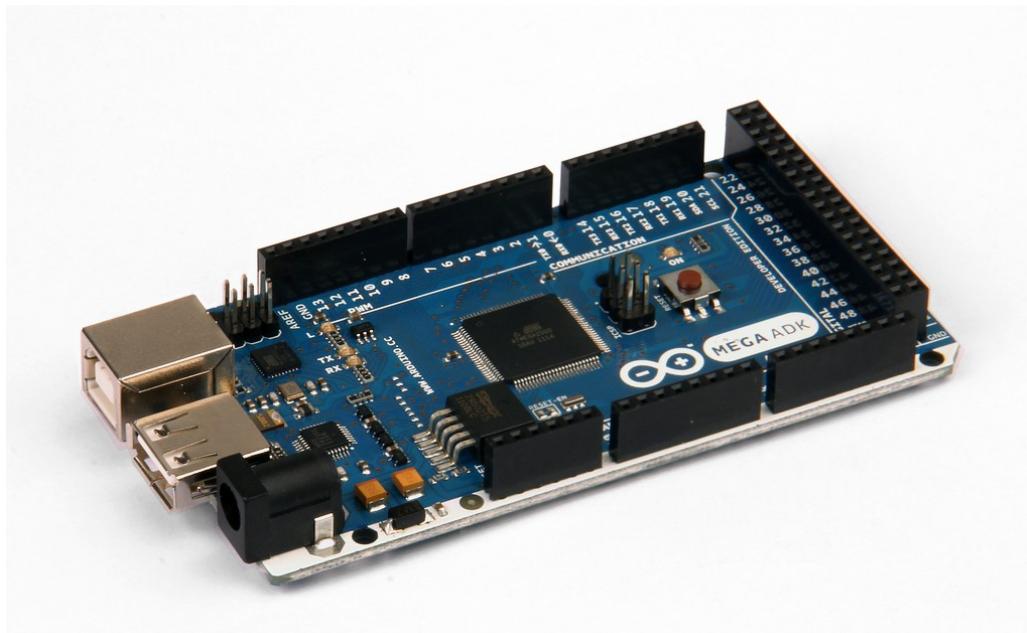
Le système est câblé en pull-down, de ce fait lors d'un appuie le microcontrôleur lira un '0' et en relâché un '1'. Le système connaissant à quelle ligne on a envoyé un signal pendant le balayage et grâce à la sortie activé on peut en déduire la touche.

Exemple : signal envoyé sur la ligne 2 et détection d'un niveau logique bas sur la colonne 3, correspond à un « 6 ».

### **3.3. Arduino Mega : la dev-Board par excellence**

Il existe différents types de microcontrôleurs, on peut facilement nommer Arduino, Mbed, Keil, Raspberry Pi. C'est avec ce premier que nous avons choisi de travailler. D'une simplicité déroutante et d'une accessibilité remarquable, l'Arduino est devenu l'outil indispensable pour le « Do it Yourself » en électronique.

Les Arduinos sont des microcontrôleurs de développement simple composés d'un processeur faible consommation et d'entrées/sorties en tout genre. La marque a su de faire connaître depuis des années et de nombreuses copies ont vu le jour grâce au libre accès que fichier de fabrication.



*Illustration 10: Arduino Mega*

Nous avons retenu l'Arduino Mega pour son nombre d'entrées/sorties. En effet elle dispose de 54 I/O numérique contre 14 pour l'Arduino Uno. Outre ces I/O numérique elle dispose de plusieurs liaisons Serie, qui sont indispensables pour l'utilisation du shield NFC. En plus, cette carte est dotée d'un panel de bus de communication tel que I2C ou le SPI mais qui ne sont pas utilisés dans notre système.

La programmation se fait via le logiciel Arduino IDE disponible gratuitement sur le site officiel. Elle se fait en C++ simplifié pour être accessible au plus grand nombre. Cependant, personnellement je trouve l'utilisation de ce logiciel très limite et pas du tout friendly et c'est pourquoi j'utilise Sublime Text avec un addon pour la compilation Arduino. Celui-ci permet d'avoir plus de couleur et l'écriture rapide.

```

Project_coffre.ino ×
18  static int NbID = 4;
19
20  static unsigned char knownID[NB_BADGE_KNOWN][5] = {{0x52, 0xA3, 0xAA, 0x56, 0x09},
21  //static String NameID[10]={"NewID", "Aurelien"};
22  //static String NameID[10]={"NewID", "Aurelien"};
23
24  static char KnownPassword[NB_BADGE_KNOWN][4]={{'1','1','1','1'},
25  {'1','2','3','4'}};
26
27
28
29
30
31
32 void setup(){
33   Serial.begin(9600); // open serial with PC
34   Serial1.begin(115200); //open serial1 with device
35
36
37 myservo.write(90);
38
39 lcd.begin(16, 2);
40
41 wake_card();
42 delay(100);
43 read_ACK(15);
44 delay(100);
45 display(15);
46 pinMode(LED, OUTPUT);
47 digitalWrite(LED,LOW);
48
49
50
51
52 }
53
54
55 void loop(){
56
57
58
59
60
61
62
63
64

```

**SaveTOSD(KnownPassword, knownID, NbID);**  
**//ReadSD(KnownPassword, knownID, NbID);**  
**test++;**  
**Serial.println();**  
**Serial.println("Press # to open the van+");**

Arduino 1.8.1, Arduino/Cermino Uno, on /dev/cu.Bluetooth-Incoming-Port, Line 1, Column 1

Spaces: 2      Arduino

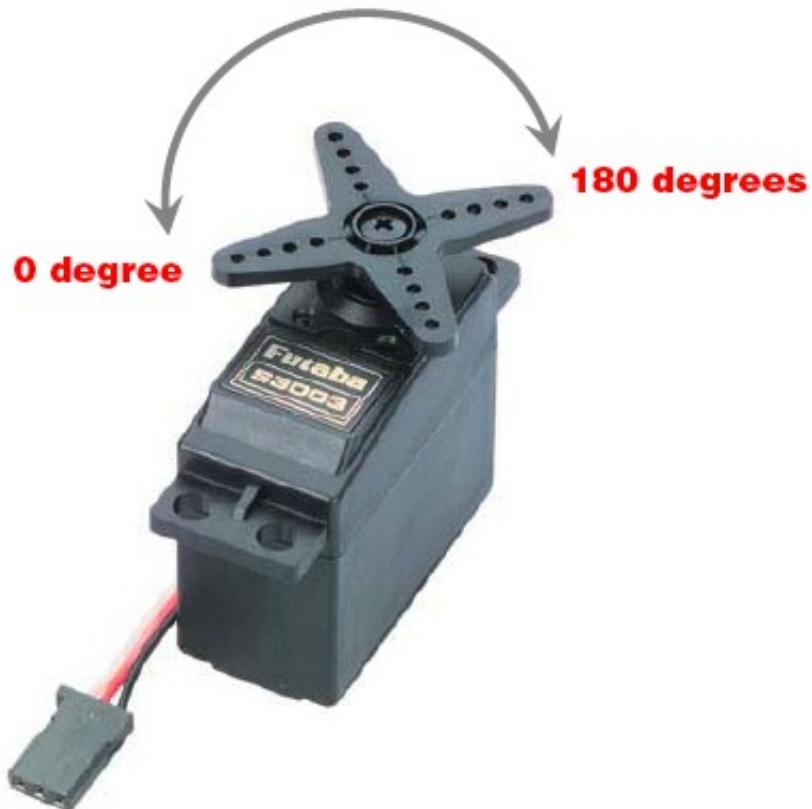
Illustration 11: Sublime Text

### **3.4. Servomoteur**

La façon la plus simple d'ouvrir la porte de notre boite était l'utilisation d'un servomoteur.

Très facile à utiliser et à alimenter. Il est fixé en bas à droite de l'ouverture.

Le servo possède 3 fils : la phase +5V, le neutre et le fils de commande. On utilise la librairie arduino servo.h qui permet la création d'objet spécifique à chaque servomoteur. On commande la position du servo grâce à un signal pwm (signal de tension carré).



*Illustration 12: Servomoteur classique*

### **3.5. Ecran LCD**

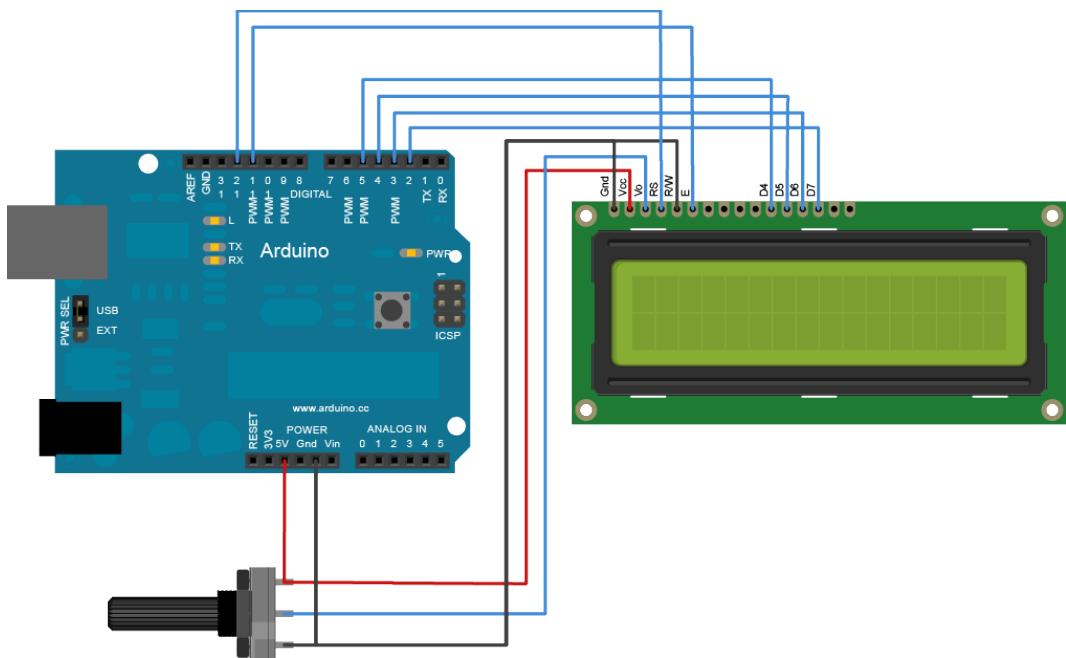
LCD signifie « Liquid Crystal Display ». Ces écrans se retrouvent absolument partout : dans les montres, les tableaux de bord de voiture, les calculatrices, etc. Cette utilisation intensive est due à leur faible consommation et coût. Les écrans LCD sont aussi sous des formes plus complexes telles que la plupart des écrans d'ordinateur ainsi que les téléviseurs à écran plat. C'est une technologie bien maîtrisée et dont le coût de production est assez bas.

L'utilisation de LCD dans notre cas est indispensable afin de communiquer avec l'utilisateur. Il transmet des messages simples tel que : « Enter password» ou encore « please badge ».

La programmation du LCD sur arduino est très simple car une librairie toute façon est fournie avec le composant. Il suffit juste de faire des « printf » dans l'objet du LCD correspondant pour afficher du texte.



### *Illustration 13: Ecran LCD*



*Illustration 14: Cable classique d'un LCD sur Arduino*

## **4. Intelligence du système**

A première vue on pourrait penser que la programmation est ultra-simpliste, la logique du code l'est, mais de nombreux problèmes l'ont complexifié.

Le principe est simple, un utilisateur peut ouvrir le coffre en utilisant son badge et son code personnel. Par la suite et grâce à un utilisateur connu on pourra ajouter des badges avec mot de passe associé.

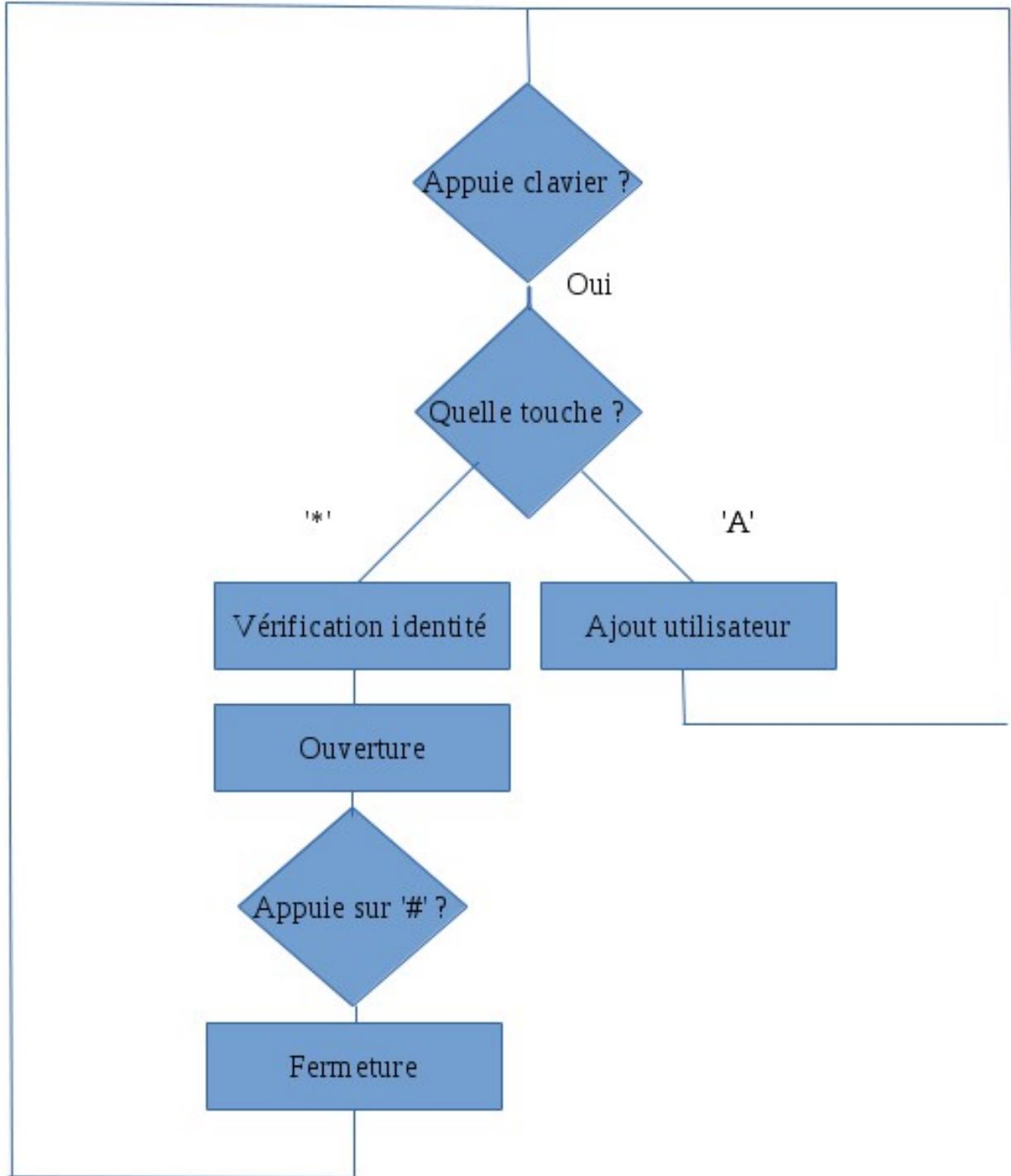


Illustration 15: Logigramme général du programme

Au menu principal l'utilisateur a deux possibilités : Appuyer sur '\*' pour passer dans le mode ouverture ou sur 'A' pour le mode ajout d'utilisateur. Pour une raison de sécurité le menu ne propose que l'ouverture, l'ajout d'utilisateur étant une fonctionnalité cachée.

Chaque fonctionnalité reprend les fonctionnalités de base qui comprend : la vérification d'un identifiant dans la base de donnée et le mot de passe associé à ce badge.

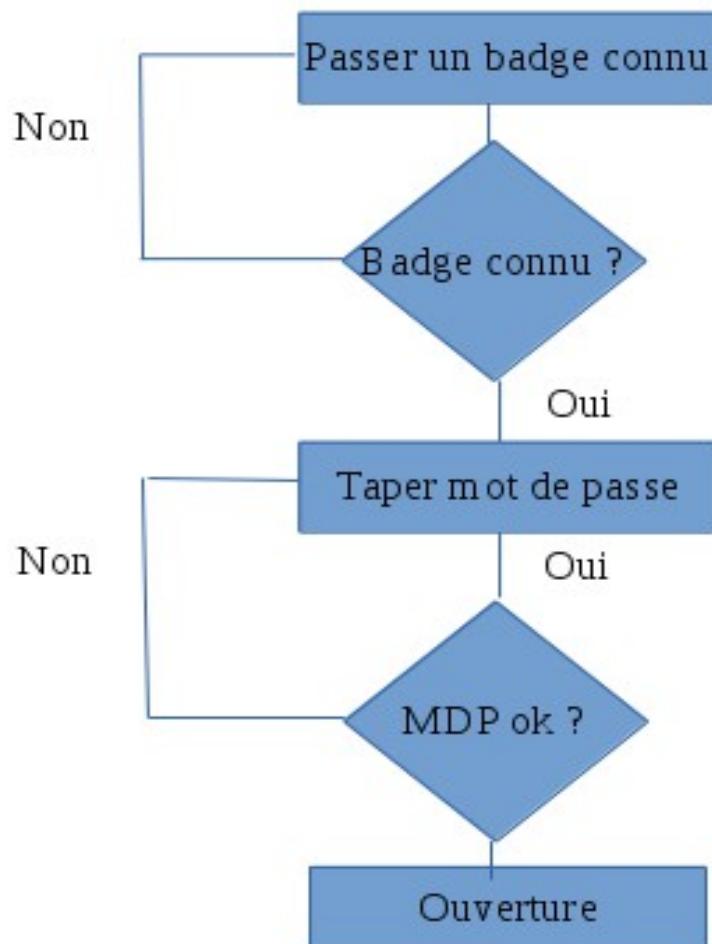


Illustration 16: Logigramme "gestion d'ouverture"

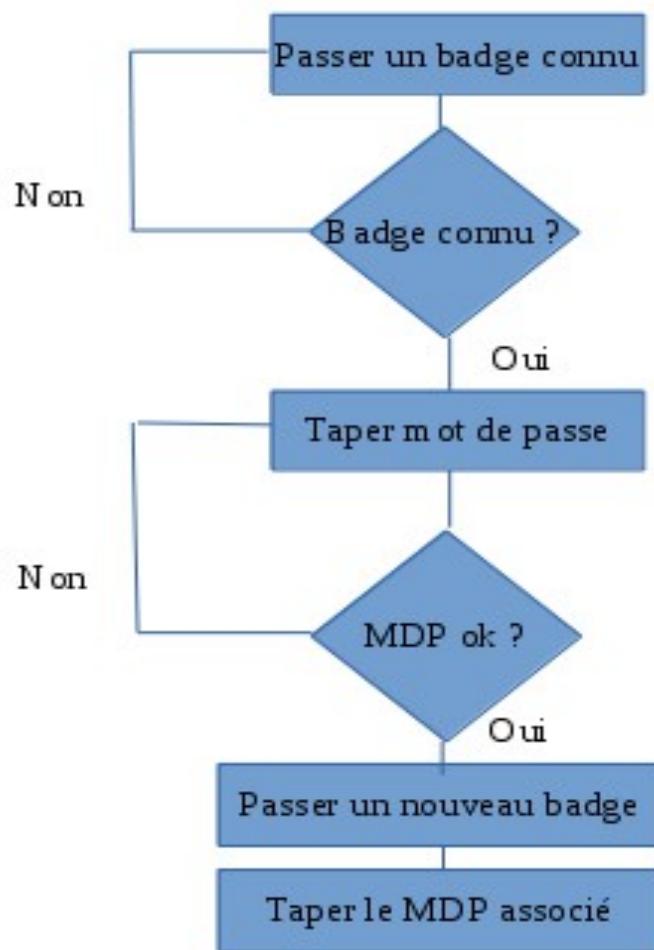


Illustration 17: Logigramme "ajout utilisateur"

## **5. Améliorations possibles : un grand avenir**

Le produit est fonctionnel, mais a besoin de pas mal d'amélioration. Dans un premier temps la partie cruciale est le design et la méca de la boite. Ce point est déjà en étude avec la possibilité de fabriquer une boite en aluminium pour un coté esthétique. Il est également possible d'envisager, dans le but d'avoir un produit robuste, un système de parois composites composés de fibre de carbone/verre, de carton alvéole et de plaque d'aluminium. Cependant, étant qu'un sujet d'étude et non un produit avec un but commercial la solution aluminium est la meilleure solution pour la démonstration.

Par la suite il serait envisageable d'effectuer une mise à jour du programme avec une meilleure gestion de la base de données et une refonte de son architecture.

Enfin, dans une deuxième version, il est possible d'imaginer l'ajout de la gestion des utilisateurs par serveurs WEB et le log des ouvertures sur ce même serveur.

## **6. Problèmes rencontrés**

Nous avons rencontré pas mal de problème quand à la conception mécanique du prototype. En effet, les machines de l'innolab requièrent des paramétrages et calibrations que nous ne maitrisons pas encore parfaitement. Sachant que chaque matériau possède ses propres paramètres, nous avons fait quelques tests avant de réaliser nos plaques définitives. Notre plus gros problème concernant le montage fut le suivant : La découpe laser de l'innolab enflammait systématiquement le plastique lors de la découpe. Nous avons dû baisser la puissance et la vitesse de la tête laser afin d'éviter le problème. Nous n'avons donc pu réaliser nos gravures.

Pour la partie programmation, le problème majeur rencontré était la sur-compilation des .h créant des conflits avec la « base de données » qui est créer en global. Ce problème a été résolu en passant en passant les tableaux en « static » et en utilisant des pointeurs dans chaque fonction.

## **7. Conclusion**

Ce projet fut extrêmement formateur, tant sur le plan électronique que conception mécanique. Nous avons pu apprendre à utiliser les différentes machines de l'innolab. Nous comprenons désormais très bien la technologie NFC. Nous ne comptons pas arrêter l'E-Vault au stade de prototype. Notre objectif est de le continuer et développer une version finale qui servira au BDE de l'ISTY.

Même si la réalisation de ce projet n'a pas été simple, les problèmes rencontrés nous ont permis d'apprendre à y faire face en trouvant de nouvelles solutions. Car c'est bien en faisant des erreurs que l'on apprend le mieux.

# **Index**

## **Index des figures**

Illustration 1: Prototype du E-Vault.....	7
Illustration 2: Intérieur du E-Vault.....	8
Illustration 3: Dessin pour gravure sur Inkscape.....	9
Illustration 4: Jointure en 3D sous catia.....	11
Illustration 5: Schema de cablage.....	12
Illustration 6: Capteur NFC.....	12
Illustration 7: Tag NFC.....	13
Illustration 8: Clavier Matriciel 4x4.....	15
Illustration 9: Schema interne d'un pavé matriciel.....	16
Illustration 10: Arduino Mega.....	18
Illustration 11: Sublime Text.....	19
Illustration 12: Servomoteur classique.....	20
Illustration 13: Ecran LCD.....	22
Illustration 14: Cable classique d'un LCD sur Arduino.....	22
Illustration 15: Logigramme général du programme.....	24
Illustration 16: Logigramme "gestion d'ouverture".....	25
Illustration 17: Logigramme "ajout utilisateur".....	26

## 8. Annexes

### 8.1. Projet\_coffre.ino

```
*****  
Author : Aurélien Burdot  
Project : E-Vault  
Projet_coffre.ino  
25/01/16  
*****  
  
#include "File/CoffreFort.h"  
#include "File/nfc.h"  
//#include "donees.h"  
  
#define LED 13  
#define NB_BADGE_KNOWN 10  
  
char add=0;  
  
int test=0;  
  
static int NbID = 2;  
  
static unsigned char knownId[NB_BADGE_KNOWN][5] = {{0x52, 0xA3, 0xAA, 0x56, 0xD9},  
..... {0x00, 0x33, 0x72, 0x28, 0x01}};  
//static String NameId[10]={"NewID","Aurelien"};  
  
static char KnownPassword[NB_BADGE_KNOWN][4]={{'1','1','1','1'},  
..... {'1','2','3','4'}};  
  
void setup(){  
    Serial.begin(9600); // open serial with PC  
    Serial1.begin(115200); //open serial1 with device  
  
    myservo.write(90);  
    lcd.begin(16, 2);  
  
    wake_card();  
    delay(100);  
    read_ACK(15);  
    delay(100);  
    display(15);  
    pinMode(LED, OUTPUT);  
    digitalWrite(LED,LOW);  
}  
*****
```

```

void loop(){

    if(!test){
        SaveToSD(KnownPassword,knownId,NbID);
        //ReadSD(KnownPassword,knownId,&NbID);
        test++;
        Serial.println();
        Serial.println("Press # to open the volt");

        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Press # to open");

    }
    //Serial.println(NbID);

    //PrintAllID(knownId,NbID);
    char key=0;

    key = keypad.getKey();

    if (key){
        if(key=='#') {
            if(WaitToUnlock(KnownPassword,knownId,NbID))
                while(!WaitTolock(KnownPassword,knownId,NbID));
            }else if (key == 'A'){ //Pour ajouter un nouvel utilisateur placer le badge et appuyer sur #
                addNewId(KnownPassword,knownId,&NbID);
                Serial.println(NbID);
            }
        }

    // WaitTolock();
}

```

## 8.2. Coffrefort.h

```
1  /*****  
2   Author : Aurélien Burdot  
3   Project : E-Vault  
4   CoffreFort.h  
5   25/01/16  
6  
7  *****/  
8  
9  #ifndef COFFREFORT_H  
10 #define COFFREFORT_H 1  
11  
12 #include "nfc.h"  
13 #include <Keypad.h>  
14 #include <SPI.h>  
15 #include <SD.h>  
16 #include <Servo.h>  
17 #include <LiquidCrystal.h>  
18  
19 #define PAD_ROWS 4 //four rows  
20 #define PAD_COLS 4 //three columns  
21  
22 /*typedef struct {  
23     char KnownPassword;  
24     unsigned char knownId[5];  
25 }Ids;*/  
26 static const char keys[PAD_ROWS][PAD_COLS] = {  
27     {"1",'2','3','A'},  
28     {"4",'5','6','B'},  
29     {"7",'8','9','C'},  
30     {"*','0','#','D'}  
31 };  
32 static byte rowPins[PAD_ROWS] = {31+2, 33+2, 35+2, 37+2}; //connect to the row pinouts of the keypad  
33 static byte colPins[PAD_COLS] = {39+2, 41+2, 43+2, 45+2}; //connect to the column pinouts of the keypad  
34 static Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, PAD_ROWS, PAD_COLS );  
35 static Servo myservo;  
36 static LiquidCrystal lcd(22, 23, 24, 25, 26, 27);  
37  
38 char PrintAllID(unsigned char knownId[][5],int* NbID);  
39 char GetPassword(int nbNumber, char* password);  
40 char ComparePassword(int nbNumber, char* password,char IdSelected, char KnownPassword[][4],int* NbID);  
41 char addNewId(char KnownPassword[][4],unsigned char knownId[][5],int* NbID);  
42 int SaveToSD(char KnownPassword[][4],unsigned char knownId[][5],int NbID);  
43 int ReadSD(char KnownPassword[][4],unsigned char knownId[][5],int* NbID);  
44 int WaitToLock(const char KnownPassword[][4],const unsigned char knownId[][5],const int NbID);  
45 int WaitToUnlock(const char KnownPassword[][4],const unsigned char knownId[][5],const int NbID);  
46 void OpenDoor();  
47 void CloseDoor();  
48  
49 #endif  
50
```

### 8.3. CoffreFort.cpp

```
1  /*****  
2   Author : Aurélien Burdot  
3   Project : E-Vault  
4   CoffreFort.cpp  
5   25/01/16  
6   *****/  
7  
8 #include "CoffreFort.h"  
9 #include "nfc.h"  
10  
11 char PrintAllID(unsigned char knownId[][5],int* NbID){  
12  
13  
14  
15     Serial.println("-----IDs-----");  
16     Serial.println();  
17     Serial.println(*NbID);  
18     delay(1000);  
19     for(int j=0;j<(*NbID);j++){  
20         Serial.print("NbID : ");  
21         Serial.println(*NbID);  
22         for(int i=0;i<5;i++)  
23             Serial.print(knownId[j][i],HEX);  
24         Serial.println();  
25     }  
26     Serial.println("-----");  
27     return 1;  
28 }  
29 char GetPassword(int nbNumber, char* password ){  
30  
31     for(int i=0;i<4;i++)password[i]='F';  
32     //Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, PAD_ROWS, PAD_COLS );  
33     char key=0;  
34  
35     int i=0;  
36     for(i=0;((i<nbNumber));i++){  
37         key = keypad.getKey();  
38         if(key=='#')break;  
39         if(key){  
40             password[i] = key;  
41             Serial.print('*');  
42         }  
43         else i--;  
44     }  
45     while((key!='#')){  
46         key = keypad.getKey();  
47         if(key){  
48             password[i] = key;  
49             Serial.print('*');  
50             Serial.println();  
51         }  
52     }  
53     return 1;  
54 }  
55 char ComparePassword(int nbNumber, char* password,char IdSelected, char KnownPassword[][4],int* NbID){  
56     int val=0;  
57     Serial.println(IdSelected);  
58     for(int i;i<nbNumber;i++){  
59         Serial.print(KnownPassword[IdSelected][i]);  
60         if(password[i]==KnownPassword[IdSelected][i])val++;  
61     }  
62     if(val>=nbNumber) return 1;  
63     else return 0;  
64 }
```

```

71  char addNewId(char KnownPassword[][4],unsigned char knownId[][5],int* NbID){
72
73      lcd.clear();
74      lcd.setCursor(0, 0);
75      Serial.println("Please badge a known badge to add new one");
76      lcd.print("Use a known badge");
77      //Serial.println("Please Badge with a know id");
78      char NewId=0;
79      unsigned char localid[5];
80      char idOK=0;
81      char password[4]={'F','F','F','F'};
82      char key='F';
83      int ok=0;
84      int* pNbID = NbID;
85
86
87      while(ok==0){
88          if(ReadBadge(localid)){ ///lecture badge connu
89              idOK=VerifId(localid,knownId,pNbID); //verificatiuon du badge
90              Serial.println(idOK);
91              for(int i=0;i<5;i++) localid[i];
92              if(idOK){
93                  lcd.clear();
94                  lcd.setCursor(0, 0);
95                  Serial.println("Please enter your passeword");
96                  lcd.print("Enter your pwd");
97                  if(GetPassword(4, password )){

98                      if(ComparePassword(4, password,idOK,KnownPassword,pNbID)){
99                          lcd.clear();
100                         lcd.setCursor(0, 0);
101                         Serial.println("Please new badge");
102                         lcd.print("Please new badge");
103                         while(!ReadBadge(localid)){
104                             /*for(int i=0;i<5;i++){
105                             knownId[(*NbID)-1][i]=localid[i];
106                             //Serial.print(localid[i],HEX);
107                             }*/
108                             //Serial.println("did it");
109                         }
110                         lcd.clear();
111                         lcd.setCursor(0, 0);
112                         Serial.println("Please the new password");
113                         lcd.print("Enter the new pwd");
114                         if(GetPassword(4, password )){
115                             lcd.clear();
116                             lcd.setCursor(0, 0);
117                             Serial.println("New ID added in the database !");
118                             Serial.print("New ID added in the database !");
119                             Serial.println(*NbID);
120                             (*NbID)=(*NbID)+1;
121                             Serial.println(*NbID);
122                             for(int i=0;i<5;i++){
123                                 KnownPassword[(*NbID)-1][i]=password[i];
124                             }
125                             Serial.print("knowId apres ajout : ");
126                             for(int i=0;i<5;i++){
127                                 knownId[(*NbID)-1][i]=localid[i];
128                                 Serial.print(knownId[(*NbID)-1][i],HEX);
129                             }
130                         }
131                     }
132                 }
133             }
134         }
135     }
136     Serial.println();
137     Serial.print("localid apres ajout : ");
138     for(int i=0;i<5;i++)
139         Serial.print(localid[i],HEX);
140     ok=1;
141     }else {
142         lcd.clear();
143         lcd.setCursor(0, 0);
144         Serial.println("Unknown ID");
145         lcd.print("Unknown ID");
146
147     }
148 }
149
150 }
151 lcd.clear();
152 lcd.setCursor(0, 0);
153 lcd.print("Press # to open");
154
155 return 1;
156 }
```

```

158 int SaveToSD(char KnownPassword[][4],unsigned char knownId[][5],int NbID){
159
160     File KnownPasswordFILE;
161     File KnownIdFILE;
162     File NbIDFILE;
163
164     Serial.print("Initializing SD card...");
165
166     if (!SD.begin(53)) {
167         Serial.println("initialization failed!");
168         return -1;
169     }
170     Serial.println("initialization done.");
171     SD.remove("password.txt");
172     SD.remove("ids.txt");
173     SD.remove("nbid.txt");
174
175 //-----password
176
177     KnownPasswordFILE = SD.open("password.txt", FILE_WRITE);
178
179     if (KnownPasswordFILE) {
180         Serial.print("Writing to password.txt...");
181         for(int j=0;j<NbID;j++){
182             for(int i=0;i<4;i++){
183                 KnownPasswordFILE.print(KnownPassword[j][i]);
184                 KnownPasswordFILE.print(",");
185             }
186             KnownPasswordFILE.println();
187         }
188
189         KnownPasswordFILE.close();
190         Serial.println("done.");
191     } else {
192         Serial.println("error opening password.txt");
193         return -1;
194     }
195
196 //-----IDS
197
198
199     KnownIdFILE = SD.open("ids.txt", FILE_WRITE);
200
201     if (KnownIdFILE) {
202         Serial.print("Writing to ids.txt...");
203         for(int j=0;j<NbID;j++){
204             for(int i=0;i<5;i++){
205                 KnownIdFILE.print(knownId[j][i]);
206                 //KnownIdFILE.print("*");
207                 KnownIdFILE.print(",");
208             }
209             KnownIdFILE.println();
210         }
211         KnownIdFILE.close();
212         Serial.println("done.");
213     } else {
214         Serial.println("error opening ids.txt");
215         return -1;
216     }
217
218 //-----NbIds
219
220     NbIDFILE = SD.open("nbid.txt", FILE_WRITE);
221
222     if (NbIDFILE) {
223         Serial.print("Writing to nbid.txt...");
224         NbIDFILE.println(NbID);
225         NbIDFILE.close();
226         Serial.println("done.");
227     } else {
228         Serial.println("error opening nbid.txt");
229         return -1;
230     }
231     return 1;
232 }
```

```

234 int ReadSD(char KnownPassword[] [4],unsigned char knownId[] [5],int* NbID) {
235
236     File KnownPasswordFILE;
237     File KnownIdFILE;
238     File NbIDFILE;
239     unsigned char localids[5][5];
240     int i=0,j=0;
241     unsigned char read=0;
242
243     Serial.print("Initializing SD card...");
244     /* if (!SD.begin(53)) {
245         Serial.println("initialization failed!");
246         return -1;
247     }*/
248     Serial.println("initialization done.");
249
250     KnownIdFILE = SD.open("ids.txt", FILE_READ);
251
252     if (KnownIdFILE) {
253
254         while (KnownIdFILE.available()) {
255             read=KnownIdFILE.read();
256             //Serial.write(read);
257             if(read==';') {
258                 i++;
259             }
260             else if (read==13){
261                 j++;
262                 i=0;
263             }
264             else localids[j][i]=read;
265             Serial.println();
266         }
267         KnownIdFILE.close();
268         Serial.println("done.");
269     } else {
270         Serial.println("error opening ids.txt");
271         return -1;
272     }
273
274     for(int i=0;i<5;i++){
275         Serial.print(localids[0][i]);
276     }
277     Serial.println();
278
279
280     return 1;
281 }
```

```

285 int WaitToLock(const char KnownPassword[][4],const unsigned char knownId[][5],const int NbID){
286     char key=0;
287     int val=0;
288     lcd.clear();
289     lcd.setCursor(0, 0);
290     Serial.println("Press # to lock agin");
291     lcd.print("Press # to lock");
292     while(!val){
293         key = keypad.getKey();
294         if (key){
295             if(key=='#') {
296                 CloseDoor();
297                 val++;
298             }
299         }
300     }
301     return val;
302 }
303
304 int WaitToUnlock(const char KnownPassword[][4],const unsigned char knownId[][5],const int NbID){
305     int newid=0;
306     int lock=0;
307     char password[4]={'F','F','F','F'};
308     unsigned char id[5];
309
310     lcd.clear();
311     lcd.setCursor(0, 0);
312     Serial.println("Please badge to open the vault");
313     lcd.print("Badge to open");
314     while(!lock){
315
316         if(ReadBadge(id)){
317             PrintId(id);
318             newid=VerifId(id,knownId,&NbID);
319             //Serial.println(newid);
320             if(newid){
321                 lcd.clear();
322                 lcd.setCursor(0, 0);
323                 Serial.println("Please enter your passeword");
324                 lcd.print("Enter your pwd");
325                 if(GetPassword(4, password )){
326                     if(ComparePassword(4, password,newid,KnownPassword,&NbID)){
327
328                         lock++; //unlock
329                         OpenDoor();
330
331                     } else {
332                         lcd.clear();
333                         lcd.setCursor(0, 0);
334                         Serial.println("Wrong Password ! Badge again");
335                         lcd.print("Wrong Password!");
336                         return 0;
337                     }
338                 }
339             }
340         }
341     }
342     lcd.clear();
343     lcd.setCursor(0, 0);
344     Serial.println("Unknown ID");
345     lcd.print("Unknown ID");
346     return 0;
347 }
348     for(int i=0;i<5;i++)
349         id[i]=0xFF;
350     }
351 }
352     return lock;
353 }
```

```
356 void OpenDoor(){
357     lcd.clear();
358     lcd.setCursor(0, 0);
359     lcd.print("Welcome");
360
361     myservo.attach(9);
362     Serial.print("Welcome ");
363     myservo.write(180);
364     lcd.setCursor(0, 1);
365     Serial.println(" Door is open");
366     lcd.print("Door is open");
367     lcd.setCursor(0, 0);
368 }
369 void CloseDoor(){
370
371     lcd.clear();
372     lcd.setCursor(0, 0);
373
374     myservo.attach(9);
375     lcd.print("The Door is locked");
376     Serial.println("The door is locked");
377     myservo.write(90);
378     lcd.setCursor(0, 1);
379     lcd.print("GoodBye");
380     Serial.print("GoodBye");
381     lcd.setCursor(0, 0);
382     lcd.clear();
383     lcd.setCursor(0, 0);
384     lcd.print("Press # to open");
385 }
```

## 8.4. Nfc.h

```
1  /*****  
2   Modified by Aurélien Burdot  
3   Project : E-Vault  
4   nbc.h  
5   25/01/16  
6  
7  *****/  
8  #ifndef NFC_H  
9  #define NFC_H  
10  
11 #if defined(ARDUINO) && ARDUINO >= 100  
12 #include "Arduino.h"  
13 #define print1Byte(args) Serial1.write(args)  
14 #define print1LnByte(args) Serial1.write(args),Serial1.println()  
15 #else  
16 #include "WProgram.h"  
17 #define print1Byte(args) Serial1.print(args,BYTE)  
18 #define print1LnByte(args) Serial1.println(args,BYTE)  
19 #endif  
20  
21 const unsigned char wake[24]={  
22 | 0x55, 0x55, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
23 | 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x03, 0xfd, 0xd4, 0x14, 0x01, 0x17, 0x00}//wake up NFC module  
24  
25 const unsigned char firmware[9]={  
26 | 0x00, 0x00, 0xFF, 0x02, 0xFE, 0xD4, 0x02, 0x2A, 0x00};//  
27  
28 const unsigned char tag[11]={  
29 | 0x00, 0x00, 0xFF, 0x04, 0xFC, 0xD4, 0x4A, 0x01, 0x00, 0xE1, 0x00}//detecting tag command  
30  
31 const unsigned char std_ACK[25] = {  
32 | 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x0C, \  
33 | 0xF4, 0xD5, 0x4B, 0x01, 0x01, 0x00, 0x04, 0x08, 0x04, 0x00, 0x00, 0x00, 0x4b, 0x00};  
34  
35 int VerifId(unsigned char *, unsigned char knownId[][5], int* NbID);  
36 void PrintId(unsigned char * );  
37 char ReadBadge(unsigned char * );  
38 void copy_id (void);  
39 char cmp_id (void);  
40 int test_ACK (void);  
41 void send_id (void);  
42 void UART1_Send_Byt(unsigned char command_data);  
43 void UART_Send_Byt(unsigned char command_data);  
44 void read_ACK(unsigned char temp);  
45 void wake_card(void);//send wake[] to device  
46 void firmware_version(void);//send fireware[] to device  
47 void send_tag(void);//send tag[] to device  
48 void display(unsigned char tem);//send receive_ACK[] to Pd  
49 #endif
```

## 8.5. Nfc.cpp

```
1  /*****************************************************************************  
2   Modified by Aurélien Burdot  
3   Project : E-Vault  
4   nfc.cpp  
5   25/01/16  
6  
7  ****  
8  #include "nfc.h"  
9  
10 unsigned char old_id[5];  
11 unsigned char receive_ACK[25];//Command receiving buffer  
12  
13 int VerifId(unsigned char * id, unsigned char knownId[][5],int* NbID){  
14  
15     int val =0;  
16     Serial.print("NBID:");  
17     Serial.println(*NbID);  
18  
19     for(int i=0;i<5;i++)  
20         Serial.print(id[i],HEX);  
21     Serial.println();  
22  
23     for(int j=0;j<(*NbID);j++){  
24         val=0;  
25         for (int i=0;i<5;i++) {  
26             if (id[i] == knownId[j][i])  
27                 val++;  
28             if(val==5) {  
29                 Serial.println(j);  
30                 return j;  
31             }  
32         }  
33     }  
34     return 0;  
35 }  
36  
37 void PrintId(unsigned char* id){  
38     for(int i=0;i<5;i++)  
39         UART_Send_Byt(id[i]);  
40     Serial.println();  
41 }  
42  
43 char ReadBadge(unsigned char* id){  
44     char newVal=0;  
45     send_tag();  
46     read_ACK(25);  
47     delay(100);  
48  
49     if (!cmp_id ()) {  
50         if (test_ACK ()) {  
51             display (25);  
52             newVal=1;  
53             delay (100);  
54         }  
55     }  
56     copy_id ();  
57     for(int i=0;i<5;i++)  
58         id[i]=old_id[i];  
59     return newVal;  
60 }  
61  
62 void copy_id (void) {//save old id  
63     int ai, oi;  
64     for (oi=0, ai=19; oi<5; oi++,ai++) {  
65         old_id[oi] = receive_ACK[ai];  
66     }  
67 }  
68  
69 char cmp_id (void){//return true if find id is old  
70     int ai, oi;  
71     for (oi=0,ai=19; oi<5; oi++,ai++) {  
72         if (old_id[oi] != receive_ACK[ai])  
73             return 0;  
74     }  
75     return 1;  
76 }  
77  
78 int test_ACK (void) {// return true if receive_ACK accord with std_ACK  
79     int i;  
80     for (i=0; i<19; i++) {  
81         if (receive_ACK[i] != std_ACK[i])  
82             return 0;  
83     }  
84     return 1;  
85 }
```

```

87 void send_id (void) { //send id to PC
88     int i;
89     Serial.print ("ID: ");
90     for (i=19; i<= 23; i++) {
91         Serial.print (receive_ACK[i], HEX);
92         Serial.print (" ");
93     }
94     Serial.println ();
95 }
96
97 void UART1_Send_Byte(unsigned char command_data){//send byte to device
98     print1Byte(command_data);
99 #if defined(ARDUINO) && ARDUINO >= 100
100     Serial1.flush(); // complete the transmission of outgoing serial data
101 #endif
102 }
103
104 void UART_Send_Byte(unsigned char command_data){//send byte to PC
105     Serial.print(command_data,HEX);
106     Serial.print(" ");
107 }
108
109 void read_ACK(unsigned char temp){//read ACK into receive_ACK[]
110     unsigned char i;
111     for(i=0;i<temp;i++) {
112         receive_ACK[i]= Serial1.read();
113     }
114 }
115
116 void wake_card(void){//send wake[] to device
117     unsigned char i;
118     for(i=0;i<24;i++) //send command
119         UART1_Send_Byte(wake[i]);
120 }
121
122 void firmware_version(void){//send fireware[] to device
123     unsigned char i;
124     for(i=0;i<9;i++) //send command
125         UART1_Send_Byte(firmware[i]);
126 }
127
128 void send_tag(void){//send tag[] to device
129     unsigned char i;
130     for(i=0;i<11;i++) //send command
131         UART1_Send_Byte(tag[i]);
132 }
133
134 void display(unsigned char tem){//send receive_ACK[] to PC
135     unsigned char i;
136     for(i=0;i<tem;i++) //send command
137         UART_Send_Byte(receive_ACK[i]);
138     Serial.println();
139 }
```









