# Table des matières

# Avant propos :

Depuis tous les temps l'homme se pose des questions portant sur comment exploiter la nature dans laquelle il vie face à des situations de plus en plus complexes. I essaye d'améliorer du jour au lendemain ses conditions de vie, de travail…Etant acteur de la science, il tente de rendre automatique ses outils et essaye de limiter leur maniement à un nombre strict de ses semblables. C'est-à-dire ceux à qui l'accès est autorisé.

Ainsi après de longue année d'étude supérieure, en électronique et une brillante observation sur ce qui se passe dans nos différents édifices public et privées, nous nous sommes trouvés comme tâche de sécuriser les banques, les coffres forts, les bureaux, bref tout endroit dont nous souhaiterons limiter l'accès.

Pour atteindre cet objectif, nous avons choisi comme thème de stage de fin d'étude supérieure : « **la conception et la réalisation d'une serrure électronique codée** ». Ce thème qui clôt notre cursus se propose de concevoir un système qui pourra limiter l'accès dans nos différents édifices ciblés.

Nous sommes conscients que cette tâche n'est pas banale et que nul n'est parfait mais nous irons à bout de nos efforts ; n'empêche que cela soit un jour beaucoup plus améliorer car la science évolue.

Malgré l'ampleur du sujet, et la variété des notions à exposer, le volume du thème est raisonnable :

- ✓ Par le choix des notions fondamentales à traiter ;
- ✓ Par l'utilisation des circuits programmables pour rendre plus explicite notre rapport.

Nous espérons que ce manuel sera un outil de référence, efficace et agréable. Nous souhaitons vivement rentrer en contact avec les utilisateurs et nous remercions à l'avance ceux d'entre eux qui voudront bien nous faire part de leurs observations et leurs suggestions.

# I. INTRODUCTION.

## I.1. Cahier de charge.

Ce travail, qui s'inscrit dans le cadre de notre thème de stage de fin d'étude supérieure, porte sur la réalisation d'une serrure électronique codée permettant de commander l'ouverture électrique.

La porte reste fermé jusqu'après l'introduction d'un code à travers le clavier numérique qui équipe la serrure. Après validation du code par le bouton OK, le système compare le code inséré au mot de passe stocké dans sa mémoire.

Si le code introduit est égal au mot de passe, le voyant (LED) verte reste allumer, un message de bienvenue s'affiche et la porte s'ouvre puis se referme automatiquement après un court instant, le temps pour l'utilisateur d'entrer dans la salle. Une fois la porte fermée, elle se verrouille à nouveau.

Le mot de passe doit donc être connu à l' avance par l'utilisateur.

Si le code inséré n'est pas correcte, l'utilisateur est informé de la non validité du code suivi d'un message d'avertissement. Le nombre de tentative est limité à 4 tentatives successives d'un mauvais code et à chaque insertion il affiche un message d'avertissement et finira par déclencher une alarme et une LED rouge s'allume.

Ce système sera installé par soudure sur la porte de telle sorte qu'on ne pourra l'ouvrir par n'importe quel matériel que ce soit.

## I.2.Elaboration du schéma synoptique :

Le dispositif doit comporter, comme indiqué sur la figure ci-dessous :

- ✓ Un clavier qui reçoit le code saisi par l'utilisateur ;
- ✓ Un afficheur qui affiche des messages en rapport avec le code saisi ;
- ✓ Une unité centrale qui traite les informations issues de l'utilisateur et commande les différents périphériques de sortie (voyants, afficheur, porte) en fonction de l'information reçue.

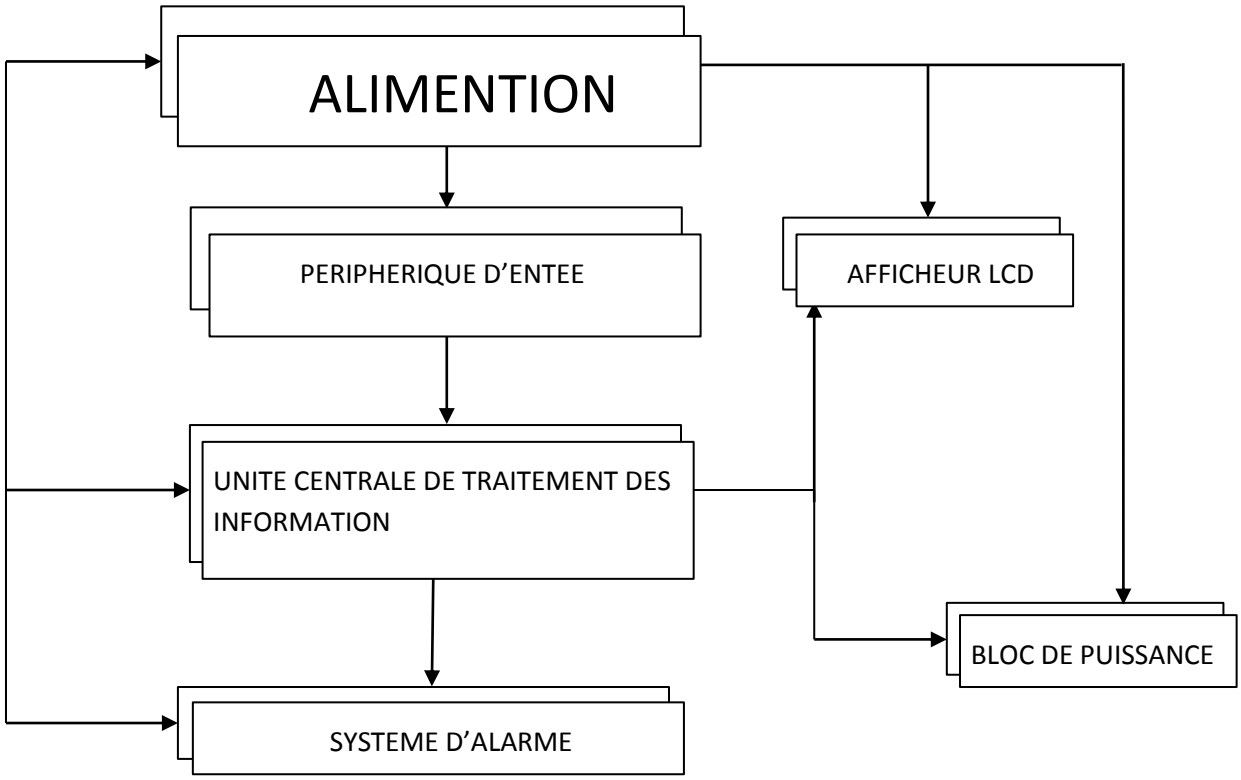



**Fig1 :** Schéma sagittal de la serrure électronique codée

# II- ETUDE DU PIC16F876A.

## II.1.Description générale.

### II.1.1. Identification de PIC 16F876A.

La dénomination PIC est sous copyright de Micro Chip, les autres fabricants sont dans l'impossibilité d'utiliser ce terme.

Les deux premiers chiffres indiquent la catégorie du PIC : ici 16 indique un PIC de la famille Mid Range (milieu de gamme) qui utilise des mots de 14 bits pour coder une instruction.

Ensuite on peut trouvez la lettre « L» qui indique que le PIC peut fonctionner avec une plage de tension beaucoup plus tolérante.

Ensuite vous trouverez les lettres suivantes :

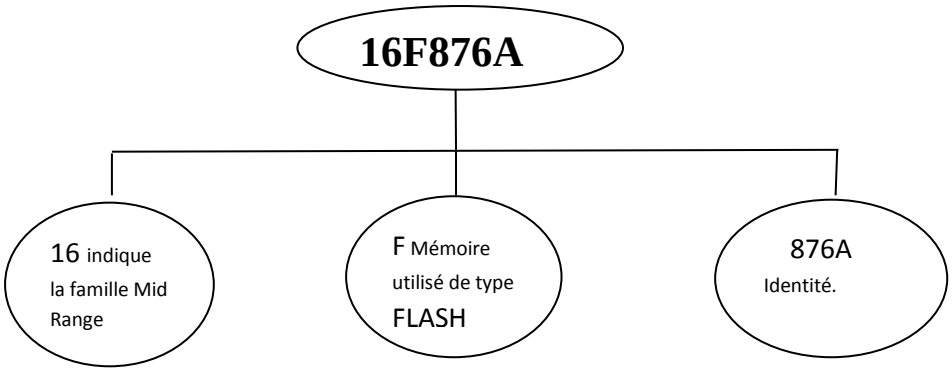
« C» : la mémoire programme est une EPROM ou plus rarement une EEPROM,

« CR» : la mémoire programme est de type ROM,

«F» : la mémoire programme est de type FLASH.

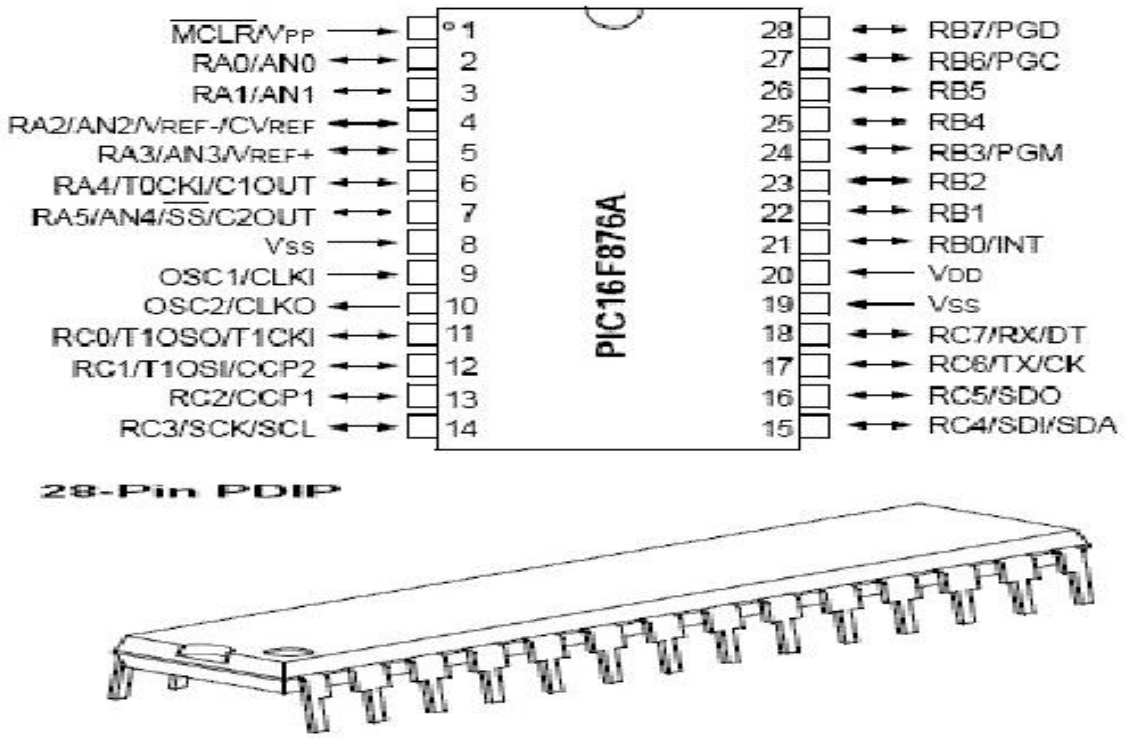
876 : représente le numéro du circuit en question.

Le PIC 16F876A est un circuit intégré de type CMOS. Il est de la famille MID-RANGE(16) et la mémoire programme est de type FLASH (F). Il est capable de fonctionner à des fréquences d'horloge allant de 0 a 20MHz.




### II.1.2. Brochage.

Le 16F876A est un circuit intégré de 28 broches, que l'on peut trouver dans un boîtier DIL (Dual In Line) de 2x14pattes comme indiqué à la figure ci-dessous. A chacune de ses broches, il est associé une ou plusieurs fonctions. Chaque broche peut donc jouer plusieurs rôles selon les configurations effectuées lors de la programmation du PIC.



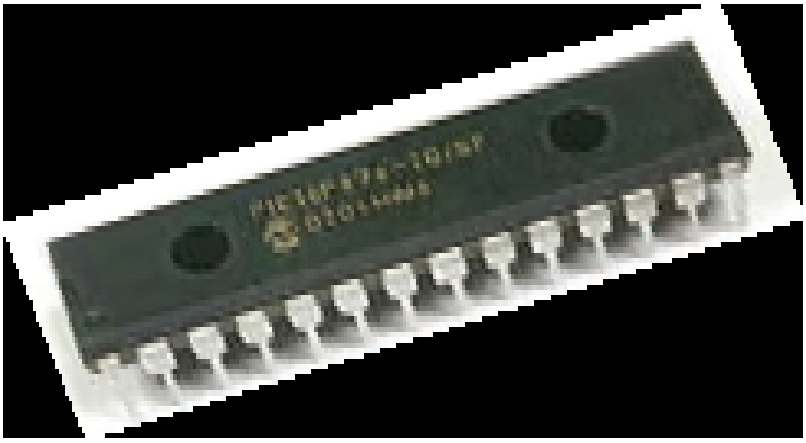



Figure.2.1.circuit de Brochage de PIC 16F876A.

## II.1.3. Les caractéristiques du PIC 16F876A.

- Une mémoire morte de type FLASH de 8K mots (1mot = 14 bits), elle est réinscriptible à volonté.
- Une Fréquence de fonctionnement qui va de 0 à 20 MHz.
- 3 Temporisateurs : TIMER0 (8 bits avec pré diviseur), TIMER1 (16 bits avec pré diviseur avec possibilité d'utiliser une horloge externe réseau RC ou QUARTZ et TIMER2 (8 bits avec pré diviseur et post diviseur).
- 13 sources d'interruption.
- 3 ports de communication.

# II.2. Etude des ports d'entrée sortie.

Le 16F876A possède jusqu'à 22 entrées/sorties :

- ✓ 6 sur le port A (RA0 à RA5) ;
- ✓ 8 sur le port B (RB0 à RB7) ;
- ✓ 8 sur le port C (RC0 à RC7) ;

Notez qu'il y a deux broches de masse (broches 8 et 19).

Certaines broches sont multiplexées à d'autres fonctions comme indiqué ci-dessous :

## II.2.1. Le PORTA.

Les abréviations et configurations relatives au port A sont données ci-dessous.

- ***RA0/AN0 :***
  - ✓ RA0 : Entrée Sortie numérique.
  - ✓ AN0 : Entrée analogique.
- ***RA1/AN1 :***
  - ✓ RA1 : Entrée Sortie numérique.
  - ✓ AN1 : Entrée analogique.
- ***RA2/AN2 :***
  - ✓ RA2 : Entrée Sortie numérique.
  - ✓ AN2 : Entrée analogique.
- ***RA3/AN3/$V_{REF}$ :***
  - ✓ RA3 : Entrée Sortie numérique.
  - ✓ AN3 : Entrée analogique.
  - ✓ $V_{REF}$ : Tension de référence.
- ***RA4/T0CKI*** :
  - ✓ RA4 : Entrée Sortie numérique.
  - ✓ T0CKI : Entrée d'horloge du TMR0.
- ***RA5/$\overline{SS}$/AN4*** :
  - ✓ RA5 : Entrée Sortie numérique.
  - ✓ $\overline{SS}$ : Entrée de sélection esclave pour le port série synchrone.
  - ✓ AN4 : Entrée analogique.

## II.2.2. Le PORTB.

Les abréviations et configurations relatives au port B sont données ci-dessous.

- ***RB0/INT :***
  - ✓ RB0 : Entrée Sortie numérique.
  - ✓ INT : Broche d'interruption externe.
- ***RB1 :***
  - ✓ RB1 : Entrée Sortie numérique.
- ***RB2 :***
  - ✓ RB2 : Entrée Sortie numérique.
- ***RB3/PGM*** :
  - ✓ RB3 : Entrée Sortie numérique.
  - ✓ PGM : Entrée de la tension de programmation basse tension.
- ***RB4*** :

- ✓ RB4 : Entrée Sortie numérique.
- ***RB5*** :
  - ✓ RB5 : Entrée Sortie numérique.
- ***RB6/PGC*** :
  - ✓ RB6 : Entrée Sortie numérique.
  - ✓ PGC : Entrée d'horloge en mode programmation.
- ***RB7/PGD*** :
  - ✓ RB7 : Entrée Sortie numérique.
  - ✓ PGD : Entrée de donnée en mode programmation.

## II.2.3. Le PORTC.

Les abréviations et configurations relatives au port C sont données ci-dessous.

- ***RC0/T1OS0/T1CKI :***
  - ✓ RC0 : Entrée Sortie numérique.
  - ✓ T1OSO : Sortie d'oscillateur du TMR1.
  - ✓ T1CKI : Entrée d'horloge du TMR1.
- ***RC1/T1OSI/CCP2 :***
  - ✓ RC0 : Entrée Sortie numérique.
  - ✓ T1OSI : Entré de l'oscillateur du TMR1.
  - ✓ CCP2 : Entrée/Sortie du module CCP2.
- ***RC2/CCP1 :***
  - ✓ RC2 : Entrée Sortie numérique.
  - ✓ CCP1 : Entrée/Sortie du module CCP1.
- ***RC3/SCK/SCL*** :
  - ✓ RC3 : Entrée Sortie numérique.
  - ✓ SCK : Entrée d'horloge en mode SPI.
  - ✓ SCL : Entrée d'horloge en mode $I^2C$.
- ***RC4/SDI/SDA*** :
  - ✓ RC4 : Entrée Sortie numérique.
  - ✓ SDI : Entrée de données en mode SPI.
  - ✓ SDA : Entrée/Sortie de données en mode $I^2C$.
- ***RC5/SDO*** :
  - ✓ RC5 : Entrée Sortie numérique.
  - ✓ SDO : Sortie de données en mode SSP.
- ***RC6/TX/CK*** :
  - ✓ RC6 : Entrée Sortie numérique.
  - ✓ TX : Broche de transmission en mode USART Asynchrone.
  - ✓ CK : Entrée d'horloge en mode USART synchrone.
- ***RC7/RX/DT*** :
  - ✓ RC7 : Entrée Sortie numérique.
  - ✓ RX : Broche de réception en mode USART Asynchrone.
  - ✓ DT : Entrée/Sortie en mode USART synchrone.

**MCLR** : master clear. Broche de réinitialisation.

**$V_{PP}$** : Tension de programmation≈13V.

# II.3. Les registres spéciaux.

## II.3.1. Les registres TRISA, TRISB, TRISC.

Ces registres permettent de configurer les broches soit en entrée soit en sortie. Exemple : ***TRISC.2 = 1*** signifie que RC2 est en entrée. Lorsqu'un bit est à 1, la broche correspondante est en entrée, si c'est 0 la broche est en sortie.

## II.3.2. Le registre OPTION_REG.

Il permet la configuration les paramètres du TMR0, du chien de garde et des résistances de tirage comme indiqué ci-dessous.

***OPTION_REG.7*** = $\overline{\boldsymbol{RBU}}$ : Permet d'activer les résistances de tirage (0).

***OPTION_REG.6 = INTEDG*** : Permet de sélectionner le front, montant (1) ou descendant, du signal d'interruption de la broche RB0.

***OPTION-REG.5. = T0CS*** : Permet de sélectionner la source d'horloge, interne (0) ou externe (1), du TMR0.

***OPTION-REG.4. = T0SE*** : Permet de sélectionner le front d'horloge, montant (0) ou descendant (1), du TMR0.

***OPTION-REG.3. = PSA*** : Permet d'appliquer le pré-diviseur soit sur le TMR0 (0), soit sur le chien de garde (1).

***OPTION-REG.2 : OPTION-REG.0. = PS2 :PS0*** : permettent de fixer le rapport de la pré-division selon le tableau ci-dessous.

| **Valeur binaire** | **Rapport de la pré-division** | |
|---|---|---|
| | **TMR0** | **Chien de garde** |
| 000 | 1/2 | 1/1 |
| 001 | 1/4 | 1/2 |
| 010 | 1/8 | 1/4 |
| 011 | 1/16 | 1/8 |
| 100 | 1/32 | 1/16 |
| 101 | 1/64 | 1/32 |
| 110 | 1/128 | 1/64 |
| 111 | 1/256 | 1/128 |

### II.3.3. Le registre INTCON.

Ce registre permet d'effectuer les interruptions et de stocker les indicateurs d'interruptions comme indiquer ci-dessous.

***INTCON.7 = GIE*** : Permet d'activer ou de désactiver (0) les interruptions de façon globale.

***INTCON.6 = PEIE*** : Permet d'activer ou de désactive (0) toutes les interruptions externes.

***INTCON.5 = T0IE*** : Permet d'activer ou de désactive (0) l'interruption liée au TMR0.

***INTCON.4 = INTE*** : Permet d'activer ou de désactive (0) l'interruption liée à la broche RB0.

***INTCON.3 = RBIE*** : Permet d'activer ou de désactive (0) l'interruption liée au changement d'état sur l'une des broches de RB7 à RB4.

***INTCON.2 = T0IF*** : C'est l'indicateur (1) d'interruption du TR0.

***INTCON.1 = INTF*** : C'est l'indicateur (1) d'interruption sur la broche RB0.

***INTCON.0 = RBIF*** : C'est l'indicateur (1) d'interruption sur l'une des broches de RB7 à RB4.

### II.3.1. Le registre TMR0.

C'est un compteur/temporisateur dont les caractéristiques sont les suivants :

- Compteur sur 8 bits.
- Pré-diviseur 8 bits programmable.
- Choix de l'horloge : interne (mode temporisateur) ou externe (mode compteur).
- Interruption au débordement (passage de FF à 00).
- Choix du front de l'horloge en mode horloge externe.

Tous les bits de configuration du TMR0 sont dans le registre OPTION.

## II.4. Etude des Interruptions.

Interruption par définition est un programme qui provoque une rupture d'un autre programme de sorte que ce programme puisse reprendre la ou il était après exécution du premier.

Le fonctionnement par interruptions sur un microcontrôleur permet à celui-ci d'exécuter une tâche (programme principal) qui peut être interrompue par un événement. Le processeur doit alors exécuter une tâche (sous-programme) associée à cette source d'interruption. Quand la tâche est exécutée, le processeur revient à sa tâche principale.

On peut faire une analogie avec quelqu'un qui fait la cuisine tranquillement: il peut être interrompu par plusieurs sources: la sonnette de la maison, le téléphone, la minuterie de ses appareils de cuisson, le détecteur de fumée. Il doit traiter l'événement avant de revenir à sa

tâche principale. Le traitement de ces évènements peut lui-même être interrompu par un événement jugé plus important.

De la même façon, avec un système ordonné, on doit souvent définir des priorités dans les interruptions. Ainsi le traitement d'une interruption peut être interrompu par une interruption qui lui est prioritaire.

Le PIC16F876A comporte 13 sources d'interruption dont entre autres :

1. Débordement du TMR0.
2. Transition sur RB0.
3. Changement d'état sur l'une des broches de RB7 à RB4.

# III. CONCEPTION ET RÉALISATION DE LA SERRURE ELECTRONIQUE CODÉE

## III.1. Elaboration du schéma.

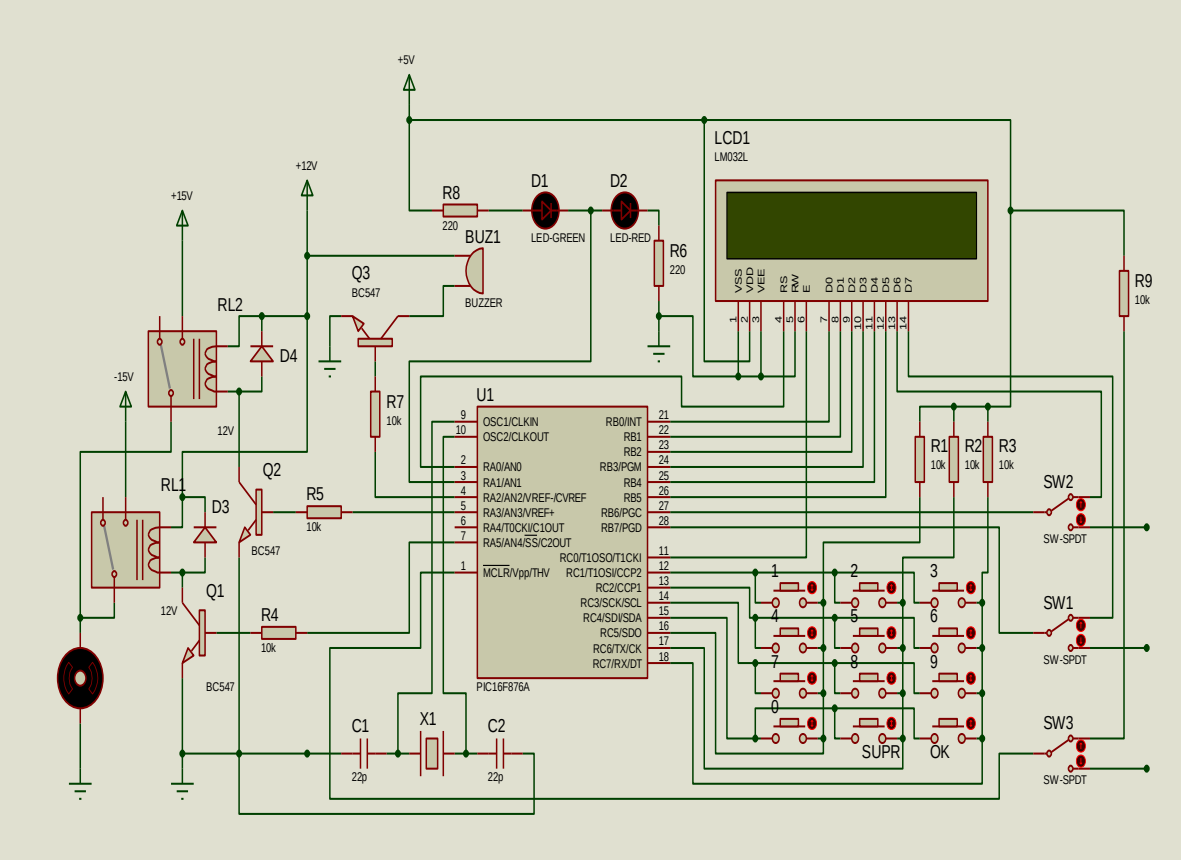
Le système comporte deux parties essentielles.

- La carte mère qui comporte tous les éléments liés au fonctionnement du système.
- La carte d'alimentation qui permet d'adapter la forme d'énergie à celle nécessaire pour le fonctionnement de l'ensemble des composants du système.

### III.1.1. Schéma électrique de la carte mère.

La figure ci-dessous donne ainsi le schéma électrique de la carte mère de la serrure codée. Elle comporte essentiellement :

- D'un microcontrôleur qui est le cerveau du système. Il traite les informations reçues à partir du clavier ; affiche les résultats ; commande les voyants, le buzzer ainsi l'ouverture ou la fermeture de la porte.
- D'un afficheur LCD.
- D'un clavier.



## III.1.2. Schéma électrique de l'alimentation.

L'alimentation est chargée de fournir l'énergie nécessaire à tout le système pour son fonctionnement. Les tensions de sortie désirées sont :

- +5V pour alimenter le microcontrôleur.
- +12V pour faire tourner le moteur dans un sens.
- -12V pour faire tourne le moteur dans l'autre sens.

Le schéma de principe est ainsi donné à la figure ci-dessous. Il comporte le programmateur qui permet de programmer le circuit sur site.

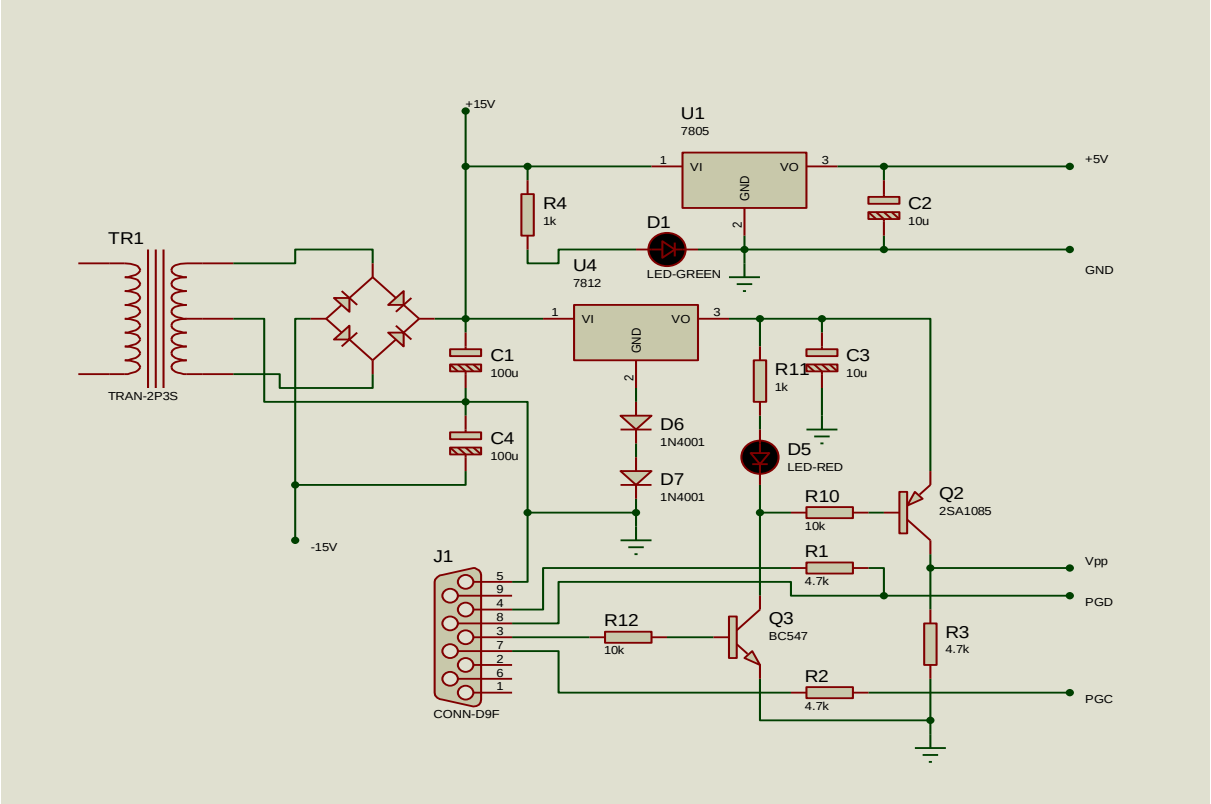



*Schéma électrique de l'alimentation.*

Il s'agit d'une alimentation symétrique linéaire à régulateurs intégrés. Le transformateur T1 abaisse la tension du secteur. Cette tension abaissée est transformée, par le pont de diode BR1, en une tension continue pulsée qui est ensuite lissée par le condensateur C1. C4 éliminent les fréquences parasites pouvant provenir du secteur en les court-circuitant à la masse. La tension pratiquement lisse est transmise aux régulateurs de tension intégré U1 et U2 qui se chargent de fournir à leur sortie respectivement une tension de 12V et 5V bien stable. C5 et C6 améliorent le temps de réponse des régulateurs.

# III.2. Choix des composants.

## III.2.1. Choix des régulateurs.

L'alimentation doit fournir la puissance nécessaire au bon fonctionnement du système. Le bilan de puissance (alimentation des relais, du buzzer, l'afficheur LCD, le clavier, le microcontrôleur) nous fait comprendre que nous devons avoir une alimentation capable de fournir une tension de 5V et 12V/500mA. Le choix du LM7812 et LM7805 (+5 et 12V/1A) s'avère donc normal.

## III.2.2. Choix du pont de diode.

Le pont doit être choisi de sorte à pouvoir supporter la tension maximale ($12\sqrt{2} \approx 17$V) et le courant d'alimentation des différents circuits (soit 1A). On choisit donc le 2W005G (50V/2A).

## III.2.3. Choix du transformateur.

En tenant compte de la tension différentielle du régulateur, de la chute de tension aux bornes des diodes du pont nous pouvons choisir un transformateur de 15V/1.5A à point milieu.

## III.2.4. Choix du fusible.

Le fusible doit être choisi de sorte que le courant au secondaire du transformateur ne dépasse pas 1,5A. En supposant le transformateur parfait nous avons la relation suivante :

$$U_p \times I_p = U_s \times I_s$$

D'où $I_p = \frac{U_s}{U_p} \times I_s = \frac{15}{220} \times 1{,}5 = 102{,}27\text{mA}$ On choisit un fusible de 250V/150mA.

# III.3. Etude de quelques composants essentiels du système.

## III.3.1. Le clavier

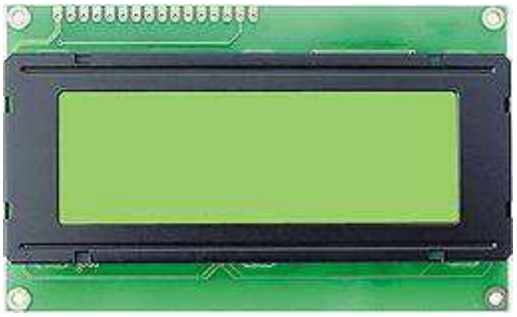
Le type de clavier utilisé est un clavier matriciel. Lorsqu'une touche est appuyée, elle est identifiée à partir de son numéro de colonne et son numéro de ligne. Le programme ainsi écrit à ce sujet permet de décoder cette information relative à l'adresse de la touche.

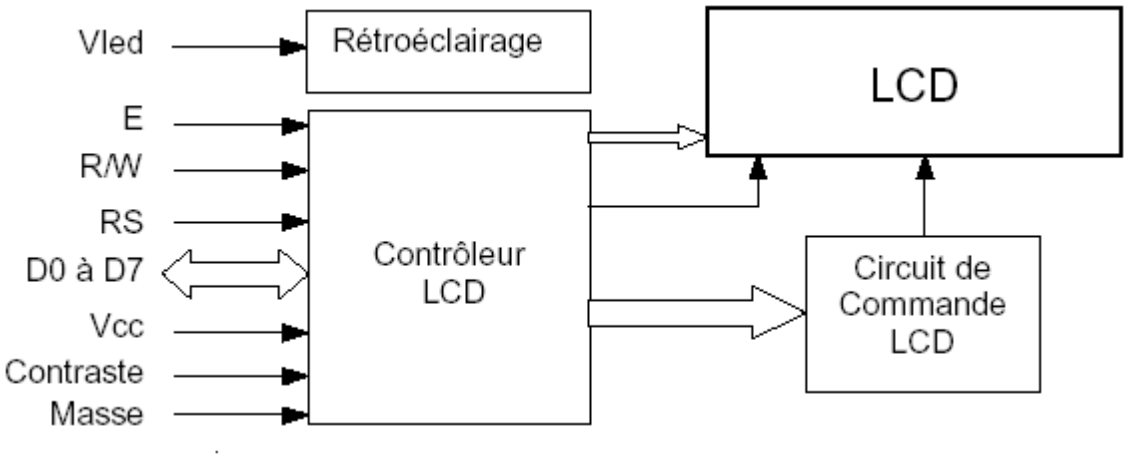
## III.3.2. L'afficheur LCD (Liquid Cristal Display).

Les afficheurs LCD sont devenus indispensables dans les systèmes techniques qui nécessitent l'affichage de paramètres de fonctionnement. Grâce à la commande par un microcontrôleur ces afficheurs permettent de réaliser un affichage de messages aisés. Ils permettent également de créer ses propres caractères.

**a) Schéma fonction et description des broches.**

Le schéma fonctionnel ainsi qu'une photo d'un d'afficheur sont donnés à la figure ci-dessous.






La description des différentes broches est donnée comme suit.

- **$V_{CC}$** : alimentation de l'afficheur LCD (5V).
- **GND** : la masse.
- **Contraste** : entrée permettant de régler le contraste de l'afficheur LCD. Il faut appliquer une tension continue réglable (entre 0V et 5V) à l'aide d'un potentiomètre.
- **$V_{led}$** : différence de potentiel permettant de commander le rétro éclairage afin d'assurer la visibilité dans l'obscurité.
- **E** : entrée de validation (ENABLE), elle permet de valider les données sur un front descendant. Lorsque E=0 alors le bus de données est à l'état haute impédance.
- **RS** : (Register Select) cette entrée permet d'indiquer à l'afficheur si l'on souhaite réaliser une commande (RS=0) par des instructions spécifiques ou écrire une donnée (envoie du code du caractère à afficher) sur le bus (RS=1).
- **R/W** : entrée de lecture (R/W=1) et d'écriture (R/W=0). Lorsqu'on commande l'afficheur LCD il faut se placer en écriture.
- **D7…D0** : bus de données bidirectionnel, il permet de transférer les instructions ou les données à l'afficheur LCD.

**b) Mise en œuvre d'un afficheur LCD.**

Un afficheur LCD sera commandé par un microcontrôleur. Il faut donc penser aux mises en œuvre :

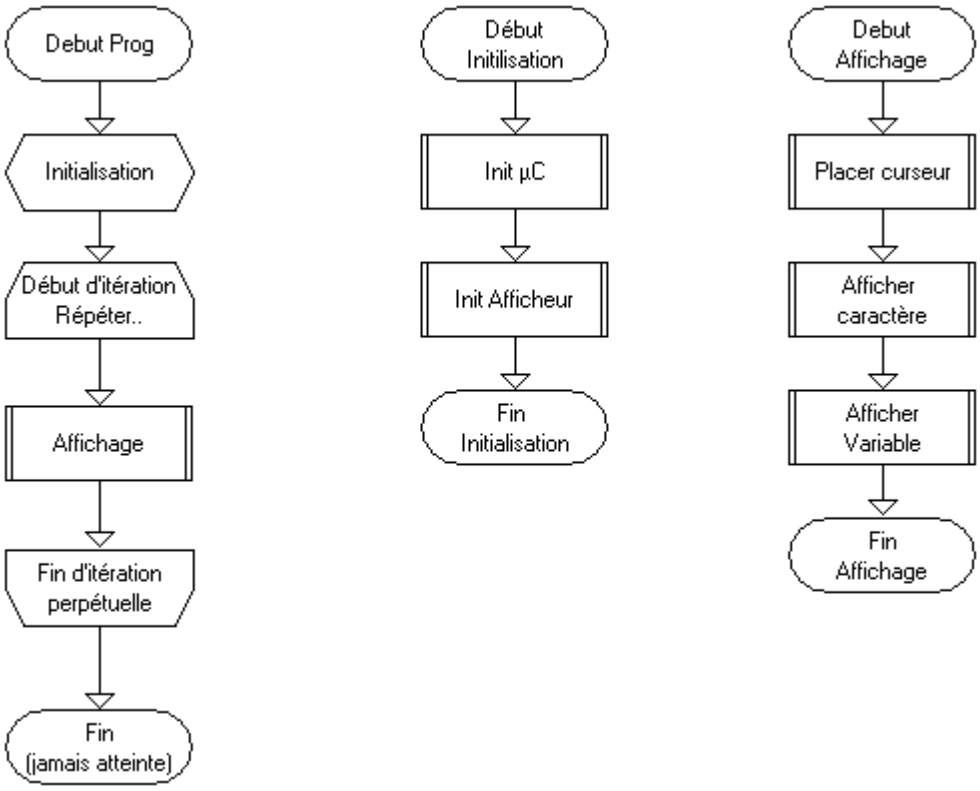
- ✓ matérielle : connexion des broches du microcontrôleur à l'afficheur LCD,
- ✓ logicielle : utilisation de sous programmes permettant de commander l'afficheur (initialisation, effacement, affichage d'un caractère, affichage d'une variable,…).

En fonction du mode de commande choisis de l'afficheur, il existe deux modes de commande d'un afficheur LCD :

- ✓ commande en 4 bits,
- ✓ commande en 8 bits.

En mode de commande 8 bits on utilise plus de broches du microcontrôleur. Il faut utiliser 11 broches des ports d'entrées/sorties du microcontrôleur (configurées en sorties) de manière à commander l'afficheur.

La mise en œuvre logicielle est décrite suivant les algorigramme ci-dessous.




L'initialisation du microcontrôleur doit permettre de configurer les broches des ports d'entrées/sorties en sorties.

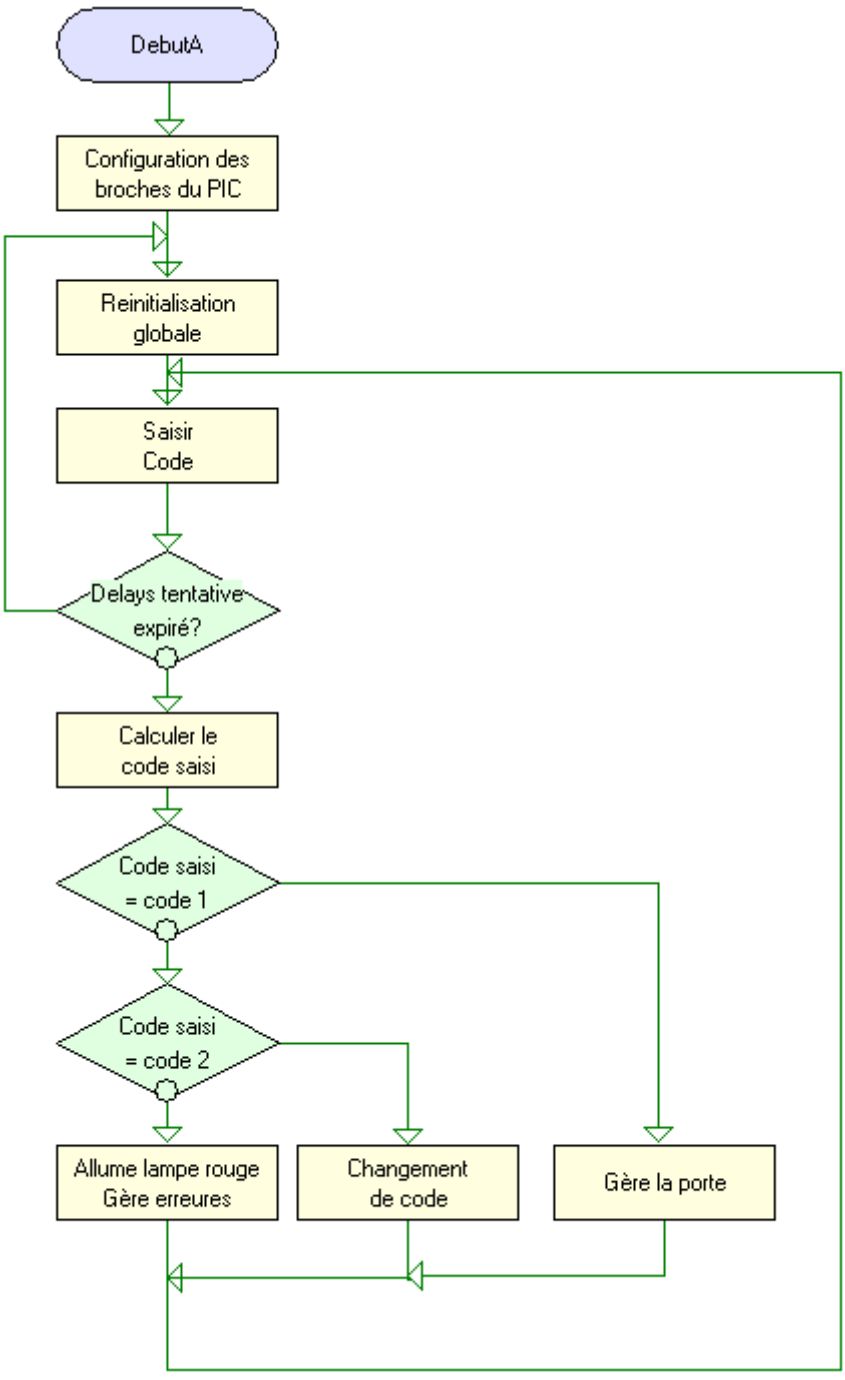
L'initialisation de l'afficheur permet :

- ✓ d'attendre la mise sous tension de l'afficheur, il faut réaliser une temporisation donnée par la documentation constructeur (10 à 20ms),
- ✓ de préparer l'initialisation en envoyant des instructions,
- ✓ de configurer le mode de commande de l'afficheur (8 bits ou 4 bits),
- ✓ de sélectionner le nombre de lignes et de configurer le format du caractère (5x7 ou 5x10 pixels),
- ✓ d'allumer l'afficheur,
- ✓ de configurer le curseur (déplacement à gauche ou à droite après affichage, clignotement),
- ✓ de réaliser le reset de l'affichage.

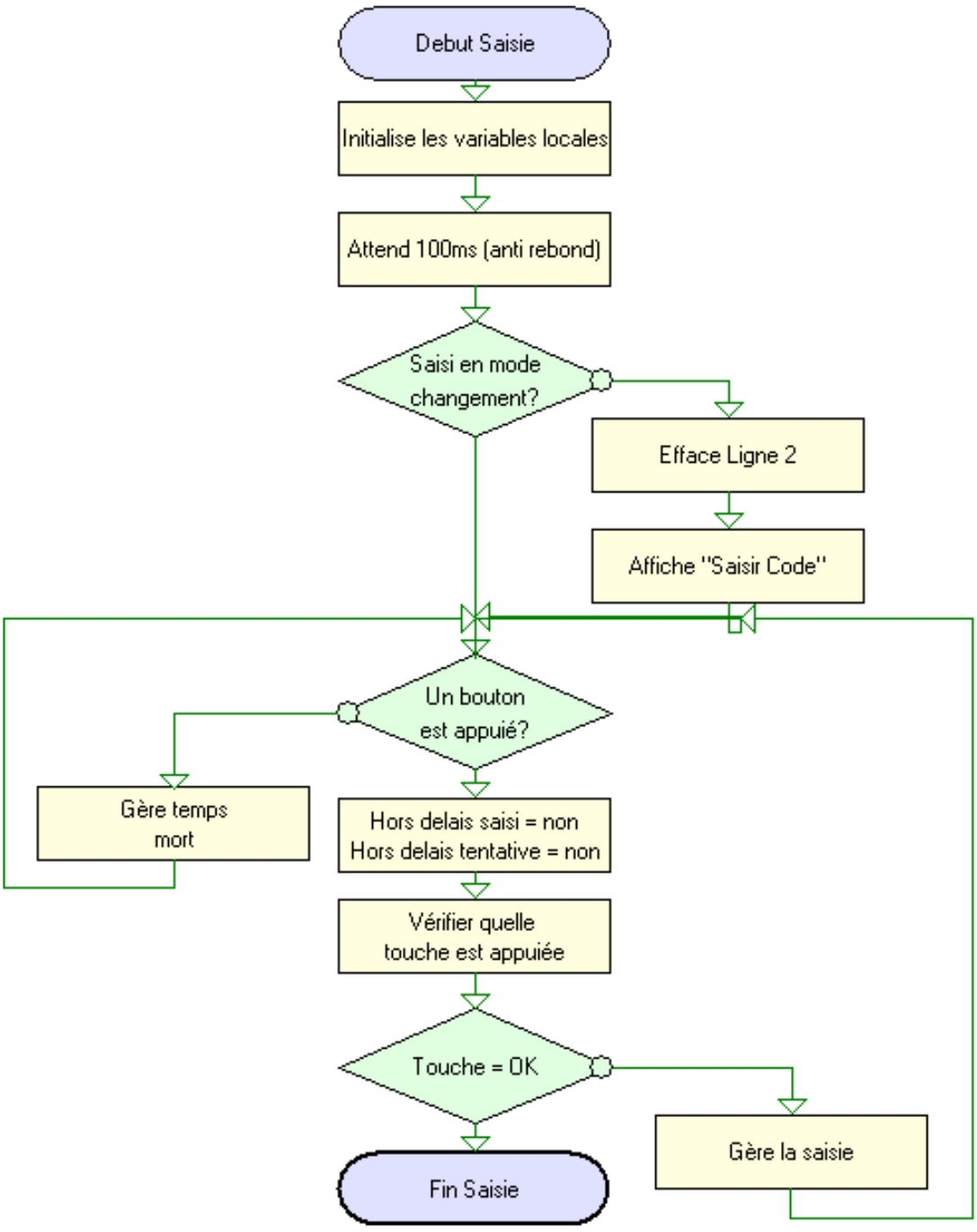
## III.2. Elaboration de l'algorigramme.

Les différents algorigrammes relatifs au fonctionnement du système sont donnés sur les figures ci-dessous.

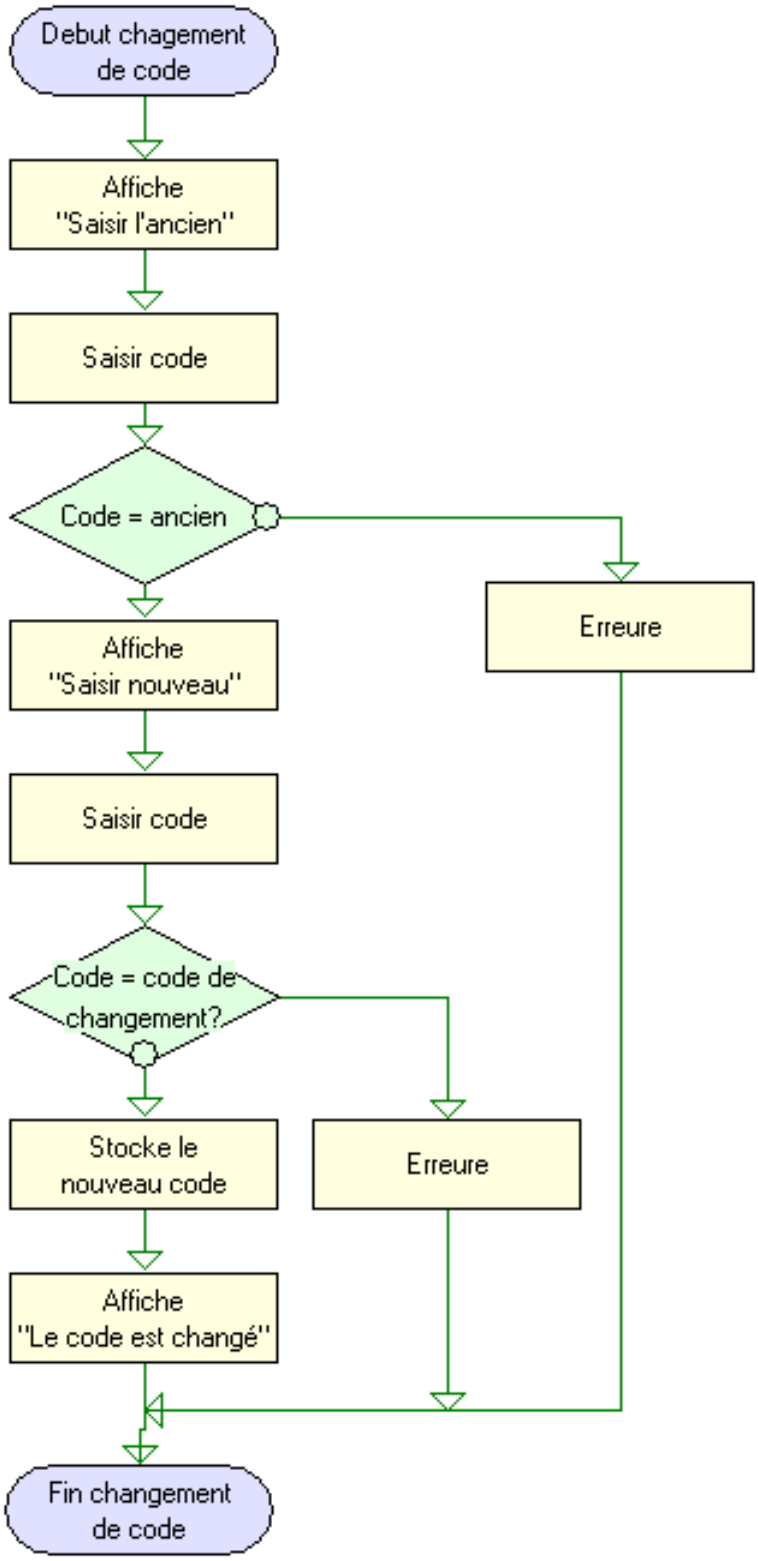
### a) Fonction principale.



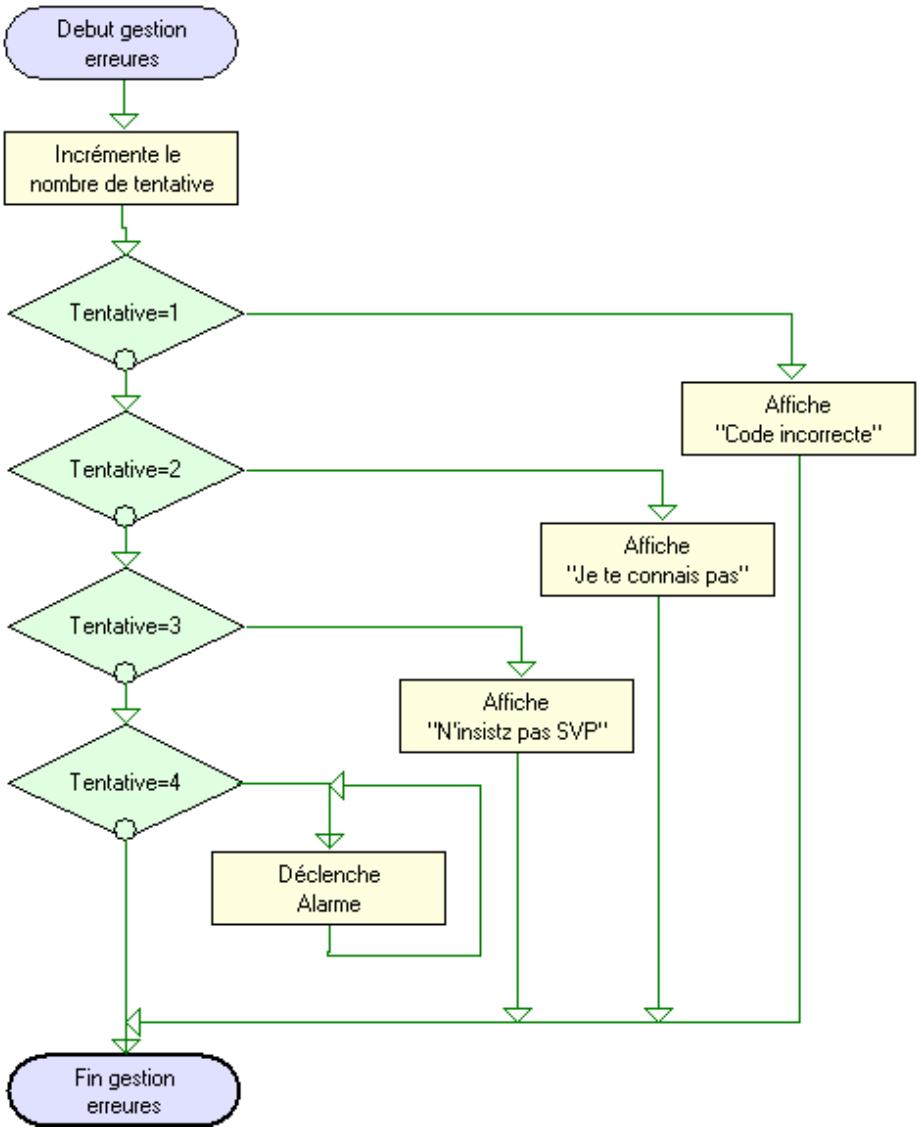
**b) Fonction de saisie de code.**



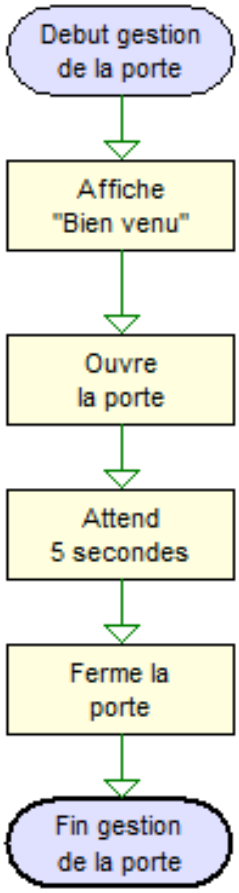
**c) Fonction de changement de code.**



## d) Fonction de gestion des erreures.



### e) Fonction de gestion de la porte.




## III.3. Ecriture du programme en C.

Le système a été programmé en langage C dont le code source est donnée ci-dessous.

```
#define Mde8b2Ligne8Pt 0b00111011
#define AffVisCurVisFx 0b00001110
#define EffEcran 0b00000001
#define Home 0b00000010

#define Rouge 1
#define Vert 0
#define Non_Appuie 0b11100000
#define Code_1 1983
#define Code_2 2003
#define Non_Changement 0
#define Changement 1
#define Tentative 2
#define Simple 0
#define Identique 2
```

```
#define OUI 1
#define NON 0
#define OK 0b01101110
#define SUPRIME 0b10101110

//déclaration des fonction secondaires
void Mazoughou_Saisi(bit Type_Code);
char Calculer_Code(char Type_Calcul);
void Gestion_Porte(void);
void Changement_Code(void);
void Gestion_Erreurs(char Type_Erreur);
void Attend_ms(unsigned long int duree);
void LCD_Initilisation_us (unsigned long int DureeInit);
void Mazoughou_Affiche(const unsigned char* PtrCarMsg);
void EcritCar (unsigned char Caractere);
void Gestion_Temps(char Type, char Duree);
void Stocke_Nouveau_Code(void);
void Ecrit_Commande(char Valeur);
void Efface_Ligne(char Ligne);
void Position_Cpteur(char Colonne, char Ligne);
void Mazoughou_Aff_Cord(char Caractere, char Colonne, char Ligne);
void Attend_us(unsigned long int duree);

//déclaration des variables globales
unsigned long int Code, Temps_us, Temps_ms;

char Numero_Chiffre_Saisi, Numero_Chiffre_Code, Temps_s, Nombre_Tentative,
        Controle_Changement, table, Mode_Changement;

char Tableau_Code_Saisi[10];
char Tableau_Code_1[10];
char Tableau_Intermediaire_Code_1[10];
char Tableau_Code_2[10];

char Port_Donnee @PORTB;
char Sens_Port_Donnee @TRISB;
char Port_Clavier @PORTC;
char Sens_Port_Clavier @TRISC;

bit Lampe @PORTA.1;
bit Moteur_Sens_Direct @PORTA.3;
bit Moteur_Sens_Inverse @PORTA.5;
bit Alarme @PORTA.2;
bit RS @PORTA.0;
bit E @PORTC.4;
bit Out_Date_Saisi, Out_Date_Tentative, Changement_Reussi;

const char Ligne_Colonne[] =
                                                {
                                                        0b11001101,    //1
                                                        0b10101101,    //2
                                                        0b01101101,    //3
                                                        0b11001011,    //4
                                                        0b10101011,    //5
                                                        0b01101011,    //6
```

```
                                                0b11000111,    //7
                                                0b10100111,    //8
                                                0b01100111,    //9
                                                0b11001110,    //0
                                                0b10101110,    //DEL
                                                0b01101110     //OK
                                        };
const Caracteres[] = "1234567890*#";

void main(void)
{
        Tableau_Code_1[0] = '1';
        Tableau_Code_1[1] = '9';
        Tableau_Code_1[2] = '8';
        Tableau_Code_1[3] = '3';

        Tableau_Code_2[0] = '2';
        Tableau_Code_2[1] = '0';
        Tableau_Code_2[2] = '0';
        Tableau_Code_2[3] = '3';
        Numero_Chiffre_Code = 4;

        ADCON1 = 7;
        TRISA = 0;
        Sens_Port_Donnee = 0;
        Sens_Port_Clavier = 0b11100000;
        Out_Date_Tentative = OUI;
        Out_Date_Saisi = OUI;
        Changement_Reussi = NON;

        Reinitialise: nop();
        PORTA = 0;
        Port_Donnee = 0;
        Port_Clavier = 0xE0;

        // Initialisation de l'afficheur
        LCD_Initilisation_us (500);
        Mazoughou_Affiche(" Serure Codee IMP7 ");
        for(;;)
        {
                Mazoughou_Saisi(Non_Changement);
                if(Out_Date_Tentative == OUI)
                        goto Reinitialise;
                Calculer_Code(Non_Changement);
                switch(Code)
                {
                        case Code_1:
                                Gestion_Porte();
                        break;

                        case Code_2:
                                Changement_Code();
                        break;

                        default:
```

```
                        {
                                Lampe = Rouge;
                                Gestion_Erreurs(Simple);
                        }
                }
        }
}

//fonction de saisie du code
void Mazoughou_Saisi(bit Type_Code)
{
        char Chiffre_Saisi, j, a, b;
        bit Sortir;

        Numero_Chiffre_Saisi = 0;
        Controle_Changement = 0;
        Sortir = NON;
        Chiffre_Saisi = 0;
        Attend_ms(100);
        if(Mode_Changement == NON)
        {
                Efface_Ligne(2);
                Mazoughou_Affiche("Saisir code");
        }

        do
        {
                while(Port_Clavier == Non_Appuie)
                {
                        if(Out_Date_Saisi == NON)
                        {
                                if(Numero_Chiffre_Saisi != 0)
                                        Gestion_Temps(Non_Changement, 10);
                                else
                                {
                                        if(Controle_Changement != 0)
                                        {
                                                Gestion_Temps(Changement, 10);

                                        }
                                }
                                if(Out_Date_Saisi == OUI)
                                        goto Fin;
                        }
                        else
                        {
                                if(Out_Date_Tentative == NON)
                                {
                                        if(Nombre_Tentative != 0)
                                        {
                                                Gestion_Temps(Tentative, 10);
                                                if(Out_Date_Tentative == OUI)
                                                        goto Fin;
                                        }
                                }
```

```
		}
	}
	Out_Date_Saisi = NON;
	Out_Date_Tentative = NON;
	for(j=0; j<12; j++)
	{
		Port_Clavier = Ligne_Colonne[j];
		a = Ligne_Colonne[j];
		if(Port_Clavier == a)
		{
			Chiffre_Saisi = Caracteres[j];
			b = Ligne_Colonne[j];
			if(b == 0b01101110)
				Sortir = OUI;
			break;
		}
	}
	if(b != OK)
	{
		if((Numero_Chiffre_Saisi == 0)&&(Controle_Changement == 0))
			Efface_Ligne(2);

		if(Type_Code == Changement)
		{
			if(b == SUPRIME)
			{
				if(Controle_Changement > 0)
				{
					Controle_Changement--;
					Mazoughou_Aff_Cord(' ', Controle_Changement, 2);
				}
				else
					Mazoughou_Affiche("Bouton invalide");
			}
			else
			{
				Tableau_Intermediaire_Code_1[Controle_Changement]	=
Chiffre_Saisi;
				Mazoughou_Aff_Cord('*', Controle_Changement, 2);
				Controle_Changement++;
			}
		}
		else	//Si on est pas en mode changement de code
		{		//chaque chiffre saisi est stocké dans le tableau de saisi
			if(b == SUPRIME)
			{
				if(Numero_Chiffre_Saisi > 0)
				{
					Numero_Chiffre_Saisi--;
					Mazoughou_Aff_Cord(' ', Numero_Chiffre_Saisi, 2);
				}
				else
					Mazoughou_Affiche("Bouton invalide");
			}
			else
```

```
                        {
                                Tableau_Code_Saisi[Numero_Chiffre_Saisi] = Chiffre_Saisi;
                                Mazoughou_Aff_Cord('*', Numero_Chiffre_Saisi, 2);
                                Numero_Chiffre_Saisi++;
                        }
                }
                Sortir = NON;
                Port_Clavier = Non_Appuie;
        }
        Attend_ms(20); //On attend un instant, le temps que le bouton appuié se stabilise
        Temps_us = 0;  //On initialise le temps qui controle l'écart entre deux saisies
        Temps_ms = 0;
        Temps_s = 0;
    }
    while(Sortir == NON);
    if(Type_Code == Changement)
        Changement_Reussi = OUI;
    Fin: nop();
}

void Stocke_Nouveau_Code(void)
{
    if(Controle_Changement != 0)
    {
        for(Numero_Chiffre_Code=0;      Numero_Chiffre_Code<Controle_Changement;
Numero_Chiffre_Code++)
        {
            table = Tableau_Intermediaire_Code_1[Numero_Chiffre_Code];
            Tableau_Code_1[Numero_Chiffre_Code] = table;
        }
        Controle_Changement = 0;
    }
}

void Gestion_Temps(char Type, char Duree)
{
    char i;
    Temps_us++;
    if(Temps_us > 100)
    {
        Temps_us = 0;
        Temps_ms++;
        if(Temps_ms > 100)
        {
            Temps_ms = 0;
            Temps_s++;
            if(Temps_s > Duree)
            {
                Temps_s = 0;
                switch(Type)
                {
                    case Non_Changement:
                    {
                        Numero_Chiffre_Saisi = 0;
                        Out_Date_Saisi = OUI;
```

```
                                        }
                                        break;

                                        case Changement:
                                                Out_Date_Saisi = OUI;
                                        break;

                                        case Tentative:
                                        {
                                                Nombre_Tentative = 0;
                                                Out_Date_Tentative = OUI;
                                        }
                                        break;
                                }
                                Efface_Ligne(2);
                                Mazoughou_Affiche("Hors Delais");
                                Attend_ms(100);

                        }
                }
        }
}

//Fonction de calcule du code
char Calculer_Code(char Type_Calcul)
{
        char i, table;
        Code = 0;
        if(Type_Calcul == Non_Changement)
        {
                if(Numero_Chiffre_Saisi == Numero_Chiffre_Code)
                {
                        for(i = 0; i<Numero_Chiffre_Saisi; i++)
                        {
                                table = Tableau_Code_1[i];
                                if(Tableau_Code_Saisi[i] != table)
                                        break;
                        }
                        if(i == Numero_Chiffre_Saisi)
                                Code = Code_1;
                        else
                        {
                                for(i = 0; i<Numero_Chiffre_Saisi; i++)
                                {
                                        table = Tableau_Code_2[i];
                                        if(Tableau_Code_Saisi[i] != table)
                                                break;
                                }
                                if(i == Numero_Chiffre_Saisi)
                                        Code = Code_2;
                        }
                }
        }
        else
        {
```

```
                for(i = 0; i<Controle_Changement; i++)
                {
                        table = Tableau_Code_2[i];
                        if(Tableau_Intermediaire_Code_1[i] != table)
                                break;
                }
                if(i == Controle_Changement)
                        Code = Code_2;
        }
        return Code;
}

//Fonction de gestion des erreurs
void Gestion_Erreurs(char Type_Erreur)
{
        Nombre_Tentative++;
        Efface_Ligne(2);
        switch(Nombre_Tentative)
        {
                case 1:
                        if(Type_Erreur ==Identique)
                                Mazoughou_Affiche("Code invalide");
                        else
                                Mazoughou_Affiche("Code non correcte");
                break;

                case 2:
                        Mazoughou_Affiche("Je vous connais pas");
                break;

                case 3:
                        Mazoughou_Affiche("N'insistez pas SVP");
                break;

                case 4:
                {
                        Mazoughou_Affiche("SOS:  Voleur");
                        for(;;)
                        {
                                Alarme = !Alarme;
                                Attend_ms(10);
                        }
                }
                break;
        }
}

//fonction de gestion de la porte
void Gestion_Porte(void)
{
        Lampe = Vert;
        Efface_Ligne(2);
        Mazoughou_Affiche("Bien Venu");

        Moteur_Sens_Direct = OUI;
```

```
	Attend_ms(300);
	Moteur_Sens_Direct = NON;
	Attend_ms(100);

	Moteur_Sens_Inverse = OUI;
	Attend_ms(300);
	Moteur_Sens_Inverse = NON;
}

//fonction attend en ms pour l'antirebond et la temporisation
void Attend_ms(unsigned long int duree)
{
	unsigned long int micro_seconde, milli_seconde;

	micro_seconde = 0;
	milli_seconde = 0;
	do
	{
		while(micro_seconde < 1000)
			micro_seconde++;
		micro_seconde = 0;
		milli_seconde++;
	}
	while(milli_seconde < duree);
	milli_seconde = 0;
}

void Attend_us(unsigned long int duree)
{
	unsigned long int micro_seconde, milli_seconde;

	micro_seconde = 0;
	while(micro_seconde < duree)
		micro_seconde++;
	micro_seconde = 0;
}

//Fonction de changement du code
void Changement_Code(void)
{
	Efface_Ligne(2);
	Mazoughou_Affiche("Saisir l'ancien");
	Mode_Changement = OUI;
	Mazoughou_Saisi(Non_Changement);
	Calculer_Code(Non_Changement);
	if(Code == Code_1)
	{
		Efface_Ligne(2);
		Mazoughou_Affiche("Saisir le nouveau");
		Mazoughou_Saisi(Changement);
		if(Changement_Reussi == OUI)
		{
			Calculer_Code(Changement);
			if(Code == Code_2)
				Gestion_Erreurs(Identique);
```

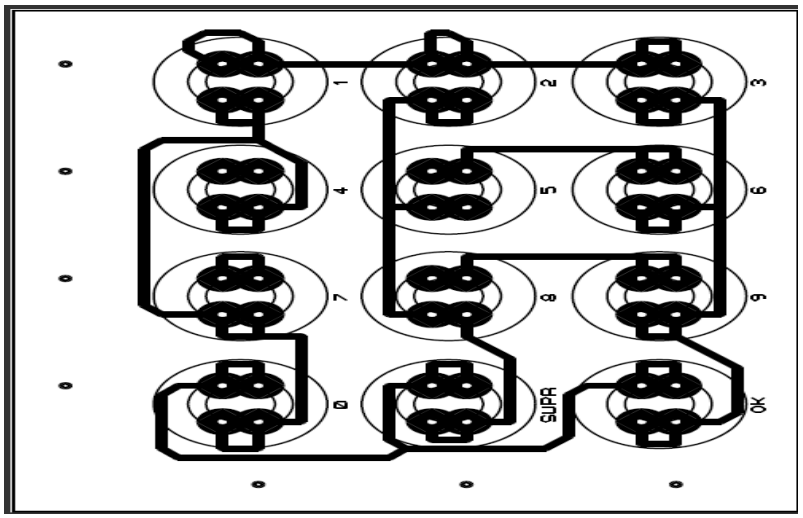
```
                        else
                        {
                                Stocke_Nouveau_Code();
                                Efface_Ligne(2);
                                Mazoughou_Affiche("Le code est changé");
                                Mode_Changement = NON;
                                Changement_Reussi = NON;
                        }
                }
        }
        else
                Gestion_Erreurs(Changement);
}

void Mazoughou_Affiche(const unsigned char* PtrCarMsg)
{
        while (*PtrCarMsg != 0)
        {
                EcritCar(*PtrCarMsg);
                PtrCarMsg ++;
        }
}

void Efface_Ligne(char Ligne)
{
        char Nbr_Caract = 0;
        Position_Cpteur(0, Ligne);
        while(Nbr_Caract != 20)
        {
                EcritCar(0x20);
                Nbr_Caract++;
        }
        Position_Cpteur(0, Ligne);
}

void Position_Cpteur(char Colonne, char Ligne)
{
        if(Ligne == 1)
                Ecrit_Commande(0x80 + Colonne);
        if(Ligne == 2)
                Ecrit_Commande(0xC0 + Colonne);
}

void Ecrit_Commande(char Valeur)
{
        Port_Donnee = Valeur;
        E = 1;
        nop();
        E = 0;
        Attend_us(500);
}

void EcritCar (unsigned char Caractere)
{
        Port_Donnee = Caractere;
```

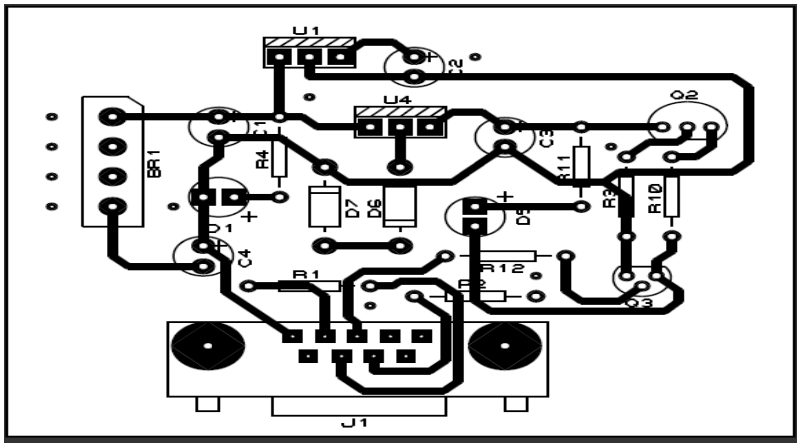
```
	RS = 1;
	nop();
	E = 1;
	nop(); // pour permettre de respecter la durée à l'état haut de E avec un µC rapide
	E = 0;
	nop();
	RS = 0 ;
	Attend_us(500); // peut être remplacé par
}

void Mazoughou_Aff_Cord(char Caractere, char Colonne, char Ligne)
{
	Position_Cpteur(Colonne, Ligne);
	EcritCar (Caractere);
}

/*******		Fonction d'initialisation du LCD		**************************/
void LCD_Initilisation_us (unsigned long int DureeInit)
{
	RS = 0;			//Envoyer une commande
	Attend_us(DureeInit);
	E = 1;

	//affichage en mode 8 bit sur 4 lignes avec 8 point;
	Attend_us(DureeInit);
	Port_Donnee = Mde8b2Ligne8Pt;
	Attend_us(DureeInit);
	E = 0;

	//Afficheur visible; curseur visible et clignotant
	Attend_us(DureeInit);
	E = 1;
	Port_Donnee = AffVisCurVisFx;
	Attend_us(DureeInit);
	E = 0;

	//Efface l'écran
	Attend_us(DureeInit);
	E = 1;
	Port_Donnee = EffEcran;
	Attend_us(DureeInit);
	E = 0;
}
/*****************************************************************************/
```

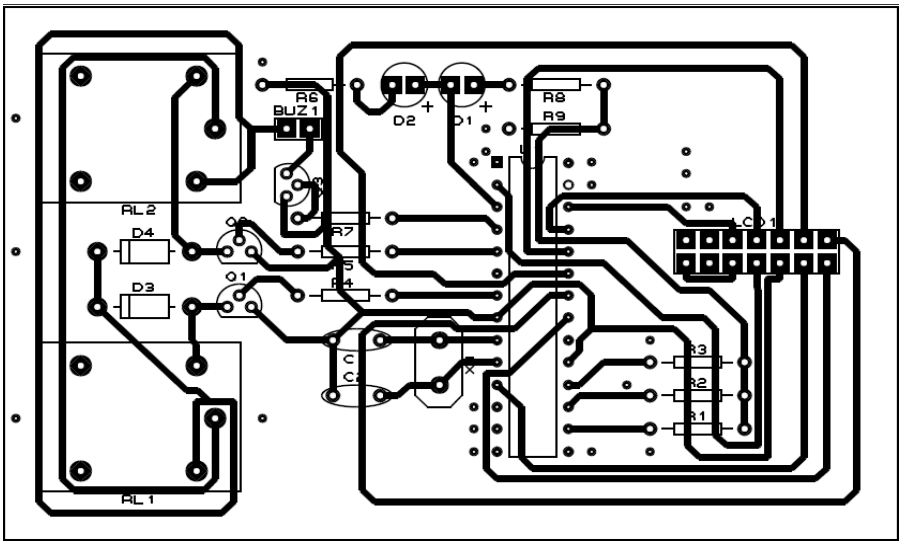
# III.4. Réalisation de la serrure électronique codée.

## a) Simulation ISIS.

**b) Tracé de circuit imprimé et image en 3D dans ARES.**



## c) Montage et expérimentions.

# IV. Conclusion.

Au terme de notre travail, il convient de retenir que la réalisation de ce projet a permis d'avoir une notion approfondie dans la conception matérielle et logicielle des systèmes. Ainsi, ce projet permettra à tout utilisateur de comprendre avec aisance la conception et la programmation des systèmes électroniques.

Ce projet dont la fin est satisfaisante pour nous s'est déroulé avec peu de difficultés rencontrées, nos solutions ont été largement inspirées des conseils avisés de notre professeur suiveur qui a su nous guider. Aussi, faut-il noter que le bon résultat d'une réalisation repose essentiellement sur les recherches, les documentations appropriées, le courage, la patience et surtout les hommes et les rapports qu'ils sont aptes à entretenir entre eux.

La réussite de ce travail nous le devons a nos encadreurs Mr Béavogui Orabaou nous saisissons cette occasion pour remercier tout d'abord le tout puissant Allah de nous avoir donné la vie le courage la santé la force nécessaire pour aboutir au terme de objectif. Nous remercions en général notre chère guinée et en particulier l'institut supérieur de technologie de Mamou a travers son directeur général Dr Kanté El hadj cellou et ses adjoints Dr Sâa Poindo Tonguino chargé des Etudes et Dr Barry Mamadou Foula chargé de la recherche scientifique sans oublier la secrétaire générale Dr Hadja Mafori Bangoura qui n'ont ménagé aucun effort pour la réussite de notre formation, au pool financier, au chargé de la bibliothèque pour sa disponibilité et sa bonne compréhension également a tout le corps professoral de l'institut supérieur de technologie de Mamou en général et plus particulièrement ceux du département instrumentation et Mesures physiques . Nos remerciements vont également à l' endroit de nos chers parents pour leur soutient moral et matériel durant notre cursus universitaire. A la population de Mamou pour leur hospitalité durant notre cycle, a nos tuteurs, ami(e)s et toute la septième promotion de l'institut supérieur de technologie de Mamou pour leur assistance. L'homme n'étant pas parfaits nous demandons pardon a tous ceux dont nous avions offensé par nos actes, faits ou parole de façon direct ou indirect pendant notre cursus en fin nous rendons un vibrant hommage a notre illustre disparu (Mr Bérété) que son âme repose en paix amen