


<h1 style="color: red; text-align: center;">Systèmes d'Information et Numérique</h1>		TSTI2D
	Comment est structuré le produit ?	SEANCE 2
	Organisation logicielle d'une serrure Codée	Activité 1

Durée : 2 H 00

Objectif visé : O3 - Analyser l'organisation fonctionnelle et structurelle d'un produit
O6 – Préparer une simulation et exploiter les résultats pour prédire un fonctionnement,

Compétences : CO 3.3 CO 6.2 CO 6.4

Connaissance visée : SA 2.4.3. Codage et traitement de l'information
SA 2.4.5. Structure d'une application logicielle
SA 3.4.1. Nature et représentation de l'information

Matériel nécessaire : Poste informatique équipé de Proteus ISIS et Arduino



Objectifs de l'activité : L'objectif de cette activité est d'apprendre aux élèves à programmer une serrure codée. À partir du cahier des charges et de l'algorithme en pseudo code, l'élève doit être capable :

- D'analyser le concept de codage d'une information, et de s'appropriier les fonctions propres à la gestion d'un clavier matriciel
- De compléter une partie de l'algorithme en fonction du cahier des charges, et de programmer en langage C++ celui-ci
- De tester et valider la solution retenue par simulation.

1- PRÉSENTATION DU PROJET

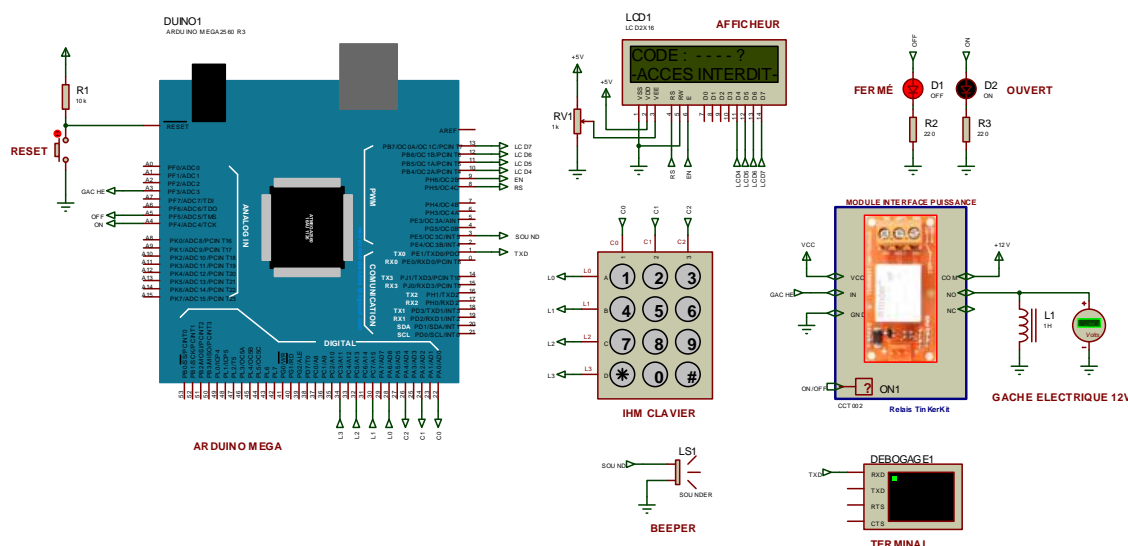
On désire réaliser un système permettant la gestion d'un accès par commande d'une gâche électrique.

Le principe consiste à utiliser une serrure codée, qui si le code entré est correct, permet l'ouverture de la porte.



1.1 - Le modèle numérique de simulation :

Le schéma de simulation Proteus ISIS vous est donné en ressource.



1.2 - Cahier des charges et EXIGENCES du PROJET :

La **gâche électrique** est commandée par la broche "A3" de la carte Arduino MEGA par l'intermédiaire d'une interface de puissance (Module RELAIS)

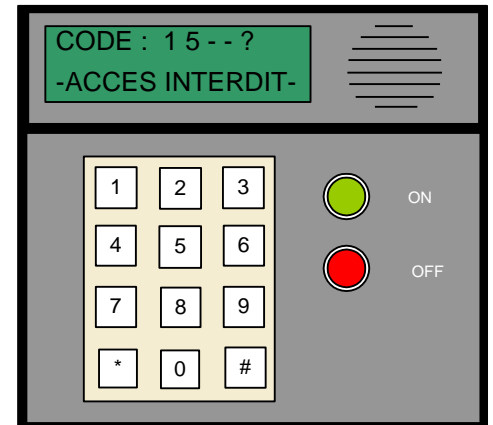
Un voyant **ROUGE** connecté sur la broche "A5" et un voyant **VERT** connecté sur "A4" permettront à l'utilisateur de connaître l'**état d'ouverture** de l'accès :

ROUGE : accès **interdit**, la gâche n'est pas alimentée
VERT : accès **autorisé**, la gâche est alimentée en tension

Si le code entré au clavier correspond au code préenregistré, alors la gâche s'ouvre pendant 5 sec avec le voyant vert allumée.

Au bout de **5 sec** la gâche électrique n'est plus alimentée, ce qui verrouille l'accès et le voyant d'accès repasse au **Rouge**.

- La touche "#" permet de **valider** le code entré au clavier
- La touche "★" permet **d'annuler** le code saisi



Un **son spécifique** doit être joué pour :

- l'appui d'une touche
- autorisation de l'accès au local (Code correct)
- code incorrect
- verrouillage de l'accès au local (annulation ou fin de l'accès ou code incorrect)

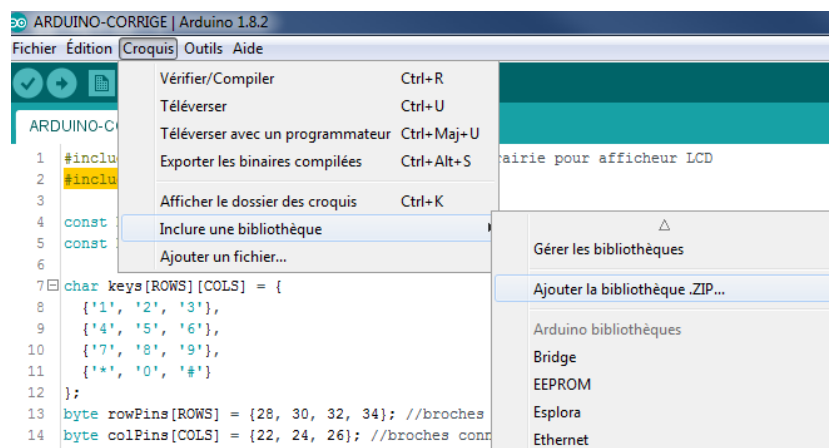
2 - UTILISATION DE LA LIBRAIRIE "KEYPAD.H" :

2.1 - Présentation :

Afin de faciliter l'utilisation du clavier, on utilisera une librairie dédiée à gestion des claviers matriciels : la librairie "**Keypad.h**"

Cette librairie peut être ajoutée à l'environnement de travail Arduino en effectuant l'import du fichier "**keypad.zip**" contenu dans le dossier ...**\ressources\librairie**.

Pour cela, sous Arduino cliquez sur "Croquis" -> "Inclure bibliothèque" -> "Ajouter .ZIP" :



2.2 - Liste des fonctions de la librairie Keypad

```
Keypad(makeKeymap(keys), row[], col[], rows, cols)
begin()
getKey()
getState()
setHoldTime(unsigned int time)
setDebounceTime(unsigned int time)
addEventListener(keypadEvent)
```



2.3 - Création d'un "objet clavier"

Pour créer un "objet clavier", il faut utiliser le constructeur suivant :

```
Keypad(makeKeymap(keys), row[], col[], rows, cols)
```

PARAMÈTRES :

keys : tableau à 2 dimensions définissant les symboles des touches
row[] : tableau correspondant aux **numéros des broches** utilisées pour les **lignes**
col[] : tableau correspondant aux **numéros des broches** utilisées pour les **colonnes**
rows : nombre de **lignes**
cols : nombre de **colonnes**

2.4 - EXEMPLE DE CRÉATION

```
const byte rows = 4; //4 lignes
const byte cols = 3; //3 colonnes

char keys[rows][cols] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};

byte rowPins[rows] = {5, 4, 3, 2}; //broches ligne L0->L3 connectés
byte colPins[cols] = {8, 7, 6};    //broches colonne C0->C2 connectés

Keypad clavier = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols );
```

Ce code crée un objet de type **Keypad**, appelé **clavier**, qui utilise les broches 5,4,3,2 pour les lignes et les broches 8,7,6 pour les colonnes. Le clavier a 4 lignes et 3 colonnes et a donc 12 touches.

La valeur qui sera renvoyée est contenue dans le tableau **clavier[rows][cols]**. C'est donc un caractère (**char**), correspondant à la touche appuyée.

2.5 - Description des fonctions

☞ `void BEGIN()`

Initialise toutes les variables. En fait le constructeur le fait déjà.

```
void begin(makeKeymap(userKeymap))
```

☞ `char GETKEY()`

Renvoie la touche (sous forme caractère ASCII) qui est appuyée, si une l'a été.

☞ `KEYPADSTATE GETSTATE()`

Renvoie l'état courant du clavier.

Les 4 états possibles sont IDLE, PRESSED, RELEASED et HOLD.

☞ `SETHOLDTIME(unsigned int TIME)`

Définit le nombre de millisecondes que l'utilisateur doit appuyer sur un bouton pour activer l'état HOLD.

☞ `SETDEBOUNCETIME(unsigned int TIME)`

Définit le nombre de millisecondes entre 2 appuis de touches. C'est une pause anti-rebond.

☞ `ADDEVENTLISTENER(KEYPADEVENT)`

Crée un évènement si le clavier est utilisé. Identique à la fonction d'interruption vue précédemment. Voir exemple de librairie EventSerialKeypad.

2.6 - EXEMPLE COMPLET

Voici l'exemple complet contenu dans la liste des exemples de la librairie Keypad.h :

```
#include <Keypad.h>
const byte ROWS = 4; //4 lignes
const byte COLS = 3; //3 colonnes
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};

byte rowPins[ROWS] = {5, 4, 3, 2}; //connexion des lignes à l'arduino
byte colPins[COLS] = {8, 7, 6}; // connexion des colonnes à l'arduino

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup() {
  Serial.begin(9600);
}

void loop() {
  char key = keypad.getKey(); //Récupère le caractère de la touche

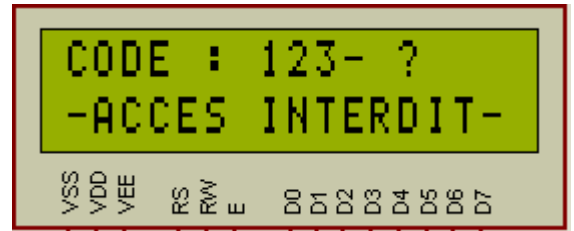
  if (key != NO_KEY) {
    Serial.println(key); //DEBOG la touche appuyée
  }
}
```

3 - ALGORITHME DE PROGRAMMATION :

Pour arriver à programmer ce projet, il vous faudra "**découper**" votre étude **en plusieurs parties** :

3.1 - Acquisition et affichage du code

Le code est composé de **4 caractères**, soit une combinaison de "0000" à "9999". Chaque appui de touche affiche le code sur l'afficheur, en **commençant** par le **chiffre des milliers** et en **terminant** par le **chiffre des unités**.



👉 Le **code** complet peut donc être **décomposé** comme étant une chaîne de caractères (String) contenant les 4 touches appuyées pour former le code.

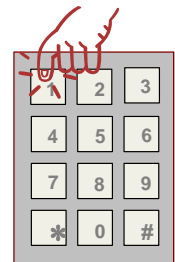
Exemple : la combinaison "**1234**" peut se décomposer comme étant "**1**" + "**2**" + "**3**" + "**4**"

Pour reconstituer ce **CODE**, il faut donc **concaténer** chaque caractère entré au clavier :

$$\text{CODE}_n = \text{CODE}_{n-1} + \text{touche_appuyée}$$

Et vérifier par une variable notée "n" que 4 touches composent le code :

n = 0 : **CODE** = "" + "1" → **CODE** = "1"
n = 1 : **CODE** = "1" + "2" → **CODE** = "12"
n = 2 : **CODE** = "12" + "3" → **CODE** = "123"
n = 3 : **CODE** = "123" + "4" → **CODE** = "1234"



Lorsque **n = 3** nous obtenons notre code entré au clavier : "**1234**" et l'on ne doit plus prendre en compte de nouvel appui sauf les touches "annulation" et "validation".

Il suffira en cas de validation de **comparer** la combinaison au code d'accès préenregistré en mémoire dans le programme.

Q1 : Quels sont les seuls caractères autorisés formant le code de 4 chiffres ?

Seuls les 10 caractères alphanumérique 0 à 9 doivent être pris en compte pour former le CODE

Q2 : Quelle touche permet de vérifier si le code entré au clavier permet le déverrouillage de l'accès ? :

La touche "#" permet de vérifier si le CODE entré au clavier autorise l'accès ou non au local

Q3 : Quelle touche permet d'annuler le code entré au clavier (On "vide" le CODE) ?

La touche "*"

Q4 : Quelle est la condition de test sur la variable "n" pour autoriser l'ajout d'un caractère dans le code ?

Tant que n < 4 le programme ajoute le caractère entré au clavier au CODE

Q5 : Complétez l'algorithme suivant en pseudo-code permettant de générer la combinaison entrée au clavier :

```

CODE est une chaine de caractères ← ""
key est un caractère ← ''
n est un entier ← 0

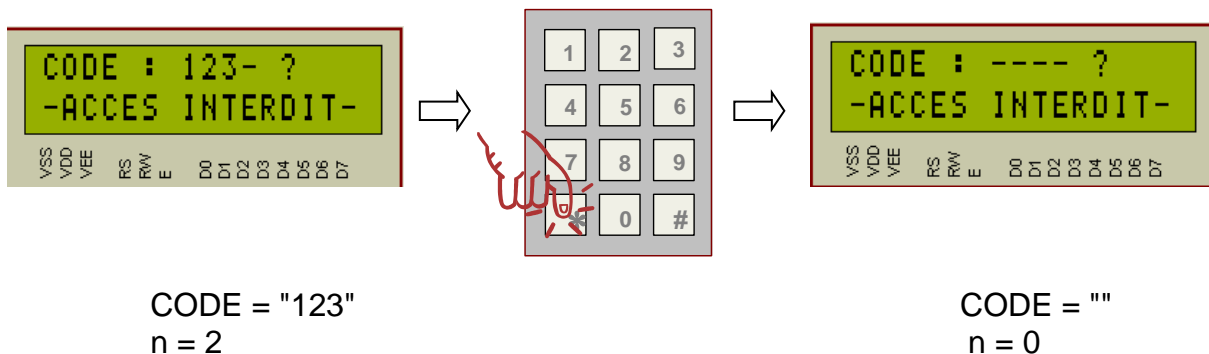
Procédure LOOP()
    key ← caractère entré au clavier
    SI key ≠ NO_KEY ALORS
        Génère SonTouche
        SI key ≠ '*' ET key ≠ '#' ALORS
            SI n < 4 ALORS
                CODE ← CODE + touche
                n ← n + 1
            FIN
        FIN
    FIN
FIN

```

3.2 - Gestion de la touche de contrôle '*'

- La touche '*' permet d'annuler le code entré. Lorsque cette touche est appuyée, le système revient dans son état initial, à savoir :

- CODE est une chaine vide ("")
- Affichage initial
- n est remis à 0
- On arrête d'alimenter la gâche
- La LED Verte est éteinte
- La LED Rouge est allumée
- Génération d'un son "annulation"



Pour simplifier la programmation, nous ferons appel à une fonction appelée `annulationCODE()` lorsque la touche "*" sera appuyée.

Q6 : Complétez l'algorithme suivant en pseudo-code permettant la gestion de la touche '*' :

CODE est une chaîne de caractères \leftarrow ""

key est un caractère \leftarrow ''

n est un entier \leftarrow 0

Procédure **LOOP**()

key \leftarrow caractère entré au clavier

SI **key** \neq **NO_KEY** **ALORS**

 Générer SonTouche

SI **key** = '*' **ALORS**

annulationCODE()

FIN

 etc...

FIN

FIN

Procédure **annulationCODE** ()

 Réinitialise l'afficheur

CODE \leftarrow ""

n \leftarrow 0

 Arrêter d'alimenter la gâche

 LED Verte \leftarrow 0

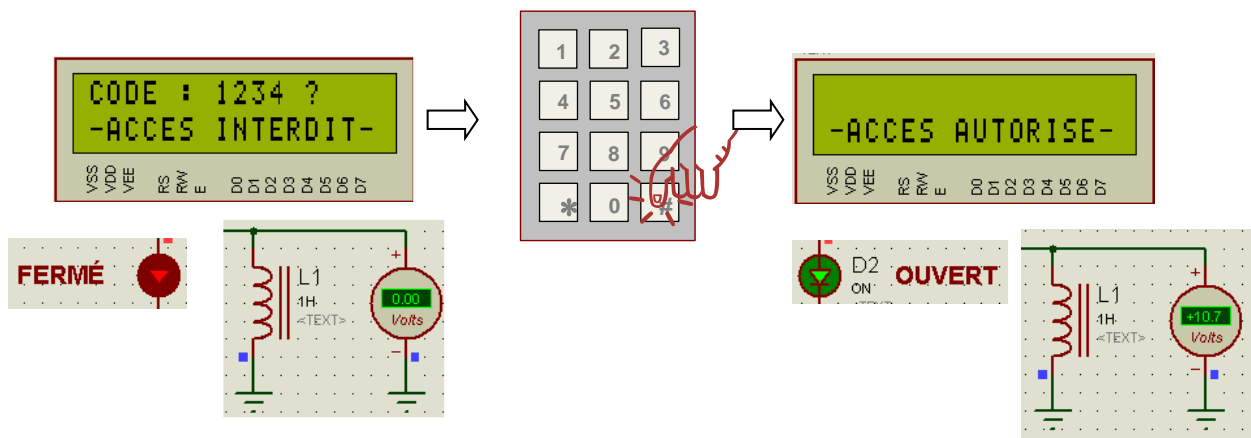
 LED Rouge \leftarrow 1

 Générer SonANNULE

FIN

3.3 - Gestion de la touche de contrôle '#'

- La touche '#' permet de valider le code entré au clavier. Lorsque cette touche est appuyée, le système va comparer la valeur de la variable **CODE** avec le code d'accès préenregistré. Si ces deux codes sont identiques, l'accès est alors autorisé :



Pour simplifier la programmation, nous ferons appel à une fonction appelée `compareCODE()` lorsque la touche "#" sera appuyée, suivi après la temporisation de l'appel à la fonction `annulationCODE()` pour remettre le système à son état initial.

Q7 : D'après le cahier des charges étudié en 2I2D, combien de temps est-il nécessaire de laisser l'accès ouvert (activation de la gâche électrique) ?

L'accès au local est autorisé pendant 5 sec.....

Q8 : Complétez l'algorithme suivant en pseudo-code permettant la gestion de la touche '#' :

```
CODE est une chaine de caractères ← ""
key est un caractère ← ''
n est un entier ← 0
CODE_OK est une chaine ← "1234"
```

Procédure `LOOP()`

```
    key ← caractère entré au clavier
    SI key ≠ NO_KEY ALORS
        Générer SonTouche
        SI key = '#' ALORS
            compareCODE()
        FIN
        etc...
    FIN
FIN
```

Procédure `compareCODE()`

```
    SI CODE = CODE_OK ALORS
        Sortie gâche ← 1
        LED Rouge ← 0
        LED Verte ← 1
        Générer SonOK
        Attendre 5 sec
    SINON
        Générer SonERREUR
    FIN
    annulationCODE()
FIN
```


3 – RÉALISATION ET SIMULATION DU PROGRAMME :

3.1 - Acquisition et affichage du code

À partir de l'étude, réalisez sous Arduino le programme permettant de gérer l'accès par clavier. Testez votre solution sur le simulateur Proteus ISIS. Mettez au point le programme si besoin.

IMPORTANT : Travaillez par étape. Ne cherchez pas à programmer l'ensemble en une fois. Gérez au fur et à mesure chaque partie :

- Initialisation de la matrice clavier et des variables de programmation
- Construction du CODE entré au clavier
- Gestion de la touche "*"
- Gestion de la touche "#"
- Gestion de l'affichage

...et surtout déboguez votre programme !

3.2 – Pour les plus rapides : Remplacement des chiffres par des étoiles ' * '

Une fois le programme au point, pensez également à **remplacer les chiffres tapés** au clavier par des **étoiles** :

...Sinon n'importe qui présent peut connaître le code d'accès !

