Lycée TOUCHARD WASHINGTON

Programmation Orientée Objet

TP de Contrôle

SNir 1
2021-2022

Le coffre de l'hôtel

Afin de stocker en toute sécurité, passeports et objet précieux de leurs clients, certains hôtels mettent à disposition un coffre-fort dans les chambres.

Celui dont vous devez faire l'étude dispose d'un clavier 12 touches et d'un écran OLED pour informer le client.

Afin d'assurer la confidentialité entre les clients et le personnel de l'hôtel, son fonctionnement est le suivant :

- Le coffre est ouvert lorsque le client prend la chambre.
- Pour fermer le coffre, il saisit une combinaison d'au moins 4 chiffres et appui sur la touche dièse et ferme la porte. La serrure du coffre se verrouille.
- Pour ouvrir le coffre, il saisit cette même combinaison et appuie sur la touche dièse. La serrure du coffre se déverrouille et la porte s'ouvre.

Le clavier possède également une touche étoile qui à tout moment, lorsqu'elle est utilisée, purge le code en cours de saisie. Le client est invité à commencer son code par la touche étoile afin de s'assurer qu'il n'y a pas de chiffre accumulé dans la combinaison.

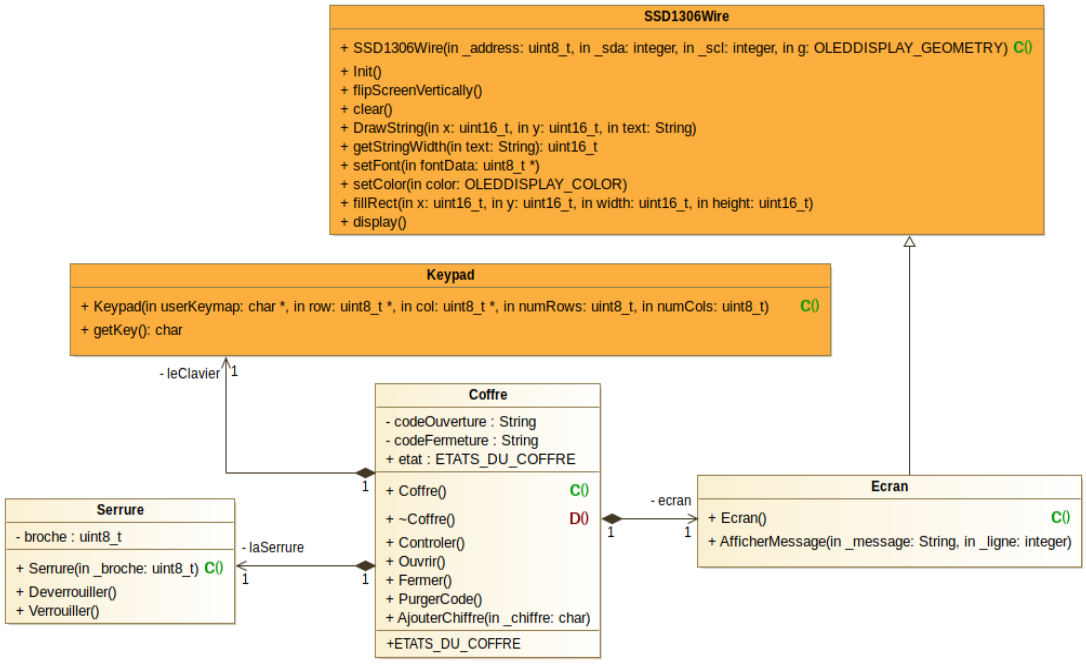
Exemple de code : *1234#

Vous disposez d'une maquette opérationnelle, vérifiez son fonctionnement avant de téléverser un quelconque programme dans l'ESP32.

Attention : il existe plusieurs types de maquettes pour cet exercice, elles ne sont pas toutes câblées de manière identique.

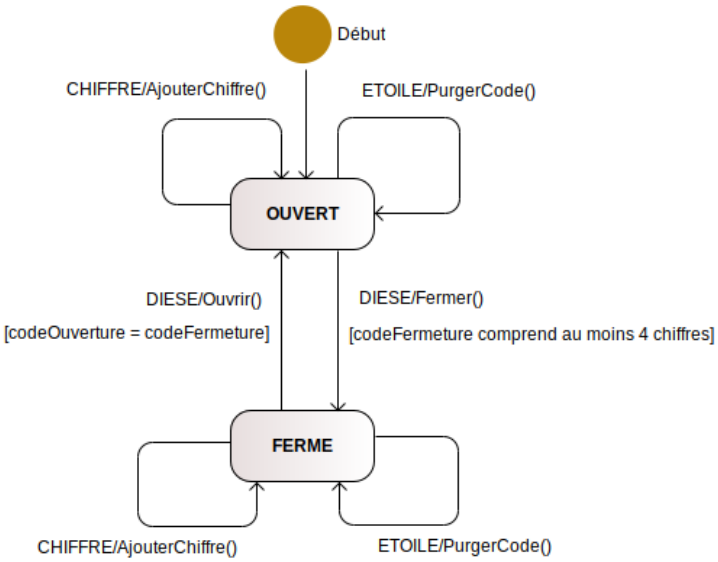
1- Analyse du projet

L'analyse du projet fait apparaître le diagramme de classe suivant :



Les classes en clair sont celles à réaliser. Les classes en orange font partie des librairies à disposition pour l'ESP32 avec les principales méthodes utilisées dans cette application.

Le comportement de la classe **Coffre**, la classe principale de l'application, est représentée par le diagramme états-transitions suivant :



Ce diagramme correspond à la méthode **controler()** de la classe **Coffre**, elle sera étudiée en détail ultérieurement.

Le diagramme fait apparaître 2 états pour notre système, soit le coffre est OUVERT soit il est FERMÉ.

Lors de la mise sous tension, le coffre se trouve dans l'état OUVERT.

2- Création du projet

Sous **LibreOffice Writer** créez un fichier **CtrlCoffre_<Votre nom>.odt**, il servira pour faire le compte-rendu de cet exercice. Pour chaque étape, vous rappellerez le titre du paragraphe.

À l'aide de **PlatformIO,** réalisez un projet nommé **CoffreHotel** pour votre environnement de développement. Ouvrez ce projet avec votre environnement et terminez sa configuration pour qu'il s'exécute sur votre cible ESP32. Ajoutez un fichier **main.cpp** dans le dossier **src** de votre projet, il contiendra les deux fonctions, **setup()** et **loop()**, nécessaires à la programmation compatible **Arduino**.

Dans votre compte-rendu, indiquez les différentes lignes de commandes réaliser avec le terminal pour créer le projet. Faites une copie d'écran de la configuration du projet.

3- Développement de la classe Ecran

Dans votre compte-rendu, insérerez l'image « ***Diagramme de classe Ecran.png*** » et indiquez le nom de la relation qui lie ces deux classes. Expliquez brièvement l'avantage que procure cette relation.

Créez maintenant, toujours dans le répertoire **src**, la classe **Ecran** dans votre projet conformément au diagramme de classes et codez le constructeur à partir des instructions suivantes :

Le constructeur transmet les paramètres au constructeur de la classe **SSD1306Wire**.

Adresse de l'écran sur le bus I2C		0x3C
Bus I2C	Broche sda	valeur de la constante SDA
	Broche scl	valeur de la constante SCL
Géométrie de l'écran		128 x 64

Il initialise l'afficheur OLED, éventuellement retourne le sens d'affichage pour qu'il soit dans le même sens de lecture que le clavier sur votre maquette et fixe la police d'écriture du type **Dialog_plain_16**. Si vous ne disposez pas de cette police vous pouvez la créer à partir du site : https://oleddisplay.squix.ch/

Dans votre compte-rendu, faites une copie d'écran du site et expliquez brièvement la démarche pour créer une nouvelle police pour votre écran OLED.

Pour l'affichage, on considère que notre écran OLED est divisé en deux zones de 128 x 32. La première représente la ligne du haut, la ligne du bas.

La méthode **AfficherMessage()** reçoit 2 paramètres. Le premier représente le message à afficher sous la forme d'une **String**, le second indique le numéro de ligne sur laquelle on souhaite faire l'affichage. Sa valeur par défaut est _ligne = 0 pour la ligne du haut. Ainsi pour écrire dans la partie haute de l'écran, il ne sera pas nécessaire de préciser cette valeur. Pour la ligne du bas, _ligne doit prendre la valeur 1.

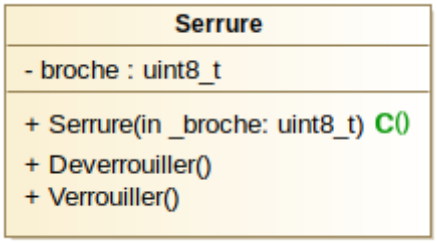
On souhaite que les messages soient parfaitement centrés horizontalement, même nombre de pixels à droite et à gauche du texte affiché.

Avant d'afficher un message, il peut être nécessaire d'effacer la zone d'affichage. La solution consiste à remplir un rectangle avec la couleur de fond ici **BLACK** et d'écrire le nouveau message dans la couleur normale **WHITE**.

Dans la fonction **setup()** de votre fichier main.cpp instanciez dynamiquement votre classe Ecran et réalisez l'affichage du message « Porte ouverte » dans la partie supérieure de l'écran. Dans la partie inférieure, vous afficherez « Code : »

Dans votre compte-rendu, reproduisez la déclaration de l'instance de la classe **Ecran** ainsi que la fonction **setup()**.

4- Développement de la classe Serrure



Ajoutez à votre projet la classe **Serrure** conformément à sa représentation UML ci-contre.

Le constructeur initialise l'attribut de la classe et configure la broche chargée de piloter la serrure en sortie. Par défaut, cette broche est la broche **2**. Cette broche correspond à la LED bleue présente sur votre carte ESP32.

Ne disposant pas pour l'instant de servomoteur les méthodes **Verrouiller()** et **Deverrouiller()** se contenteront uniquement d'allumer et d'éteindre cette LED.

Après avoir retiré les éléments concernant le test de la classe **Ecran**. Instancier de manière globale la classe **Serrure** sans prévoir d'allocation dynamique. Dans la fonction **loop()** de votre fichier **main.cpp** appelez successivement les méthodes **Verrouiller()** et **Deverrouiller()** avec entre chaque appel un délai de 500 ms. La LED bleue doit clignoter.

Dans votre compte-rendu, reproduisez la déclaration de l'instance de la classe **Serrure** ainsi que la fonction **loop()**.

5- Ajout d'une nouvelle librairie

Installez, de manière globale à tous les projets, la librairie chargée de piloter le clavier 12 touches, elle porte l'identifiant **165** dans PlatformIO.

Dans votre compte-rendu, reproduisez la ligne de commande permettant de réaliser cette installation et faites une copie d'écran montrant son installation dans le répertoire.

6- Développement de la classe Coffre

Dans votre compte-rendu, précisez la nature des relations entre la classe **Coffre** et les classes **Ecran**, **Serrure** puis **Keypad**. Que représentent les noms à l'extrémité des flèches ?

Ajoutez la classe **Coffre** à votre projet en respectant le diagramme de classes présenté lors de l'analyse. **ETATS_DU_COFFRE** représente une constante énumérée de la classe **Coffre** désignant les états que peut prendre le coffre conformément au diagramme états-transitions.

La classe **Ecran** sera instanciée de manière dynamique dans le constructeur. Les classes **Keypad** et **Serrure** seront instanciées de manière automatique et seront donc de simples attributs de la classe **Coffre**.

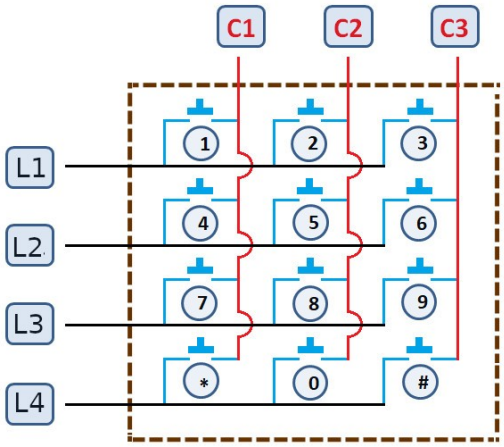
Le constructeur initialisera les constructeurs des classes **Keypad** et **Serrure** en utilisant la liste d'initialisation.

Afin de réaliser l'initialisation de la classe **Keypad** il est nécessaire d'ajouter les attributs suivants à la classe Coffre :

```
static const uint8_t LIGNES = 4;
static const uint8_t COLONNES = 3;
char touches[LIGNES * COLONNES] = {
'1', '2', '3',
'4', '5', '6',
'7', '8', '9',
'*', '0', '#'
};
//affectation des E/S GPIO aux L1,L2,L3,L4 du clavier
uint8_t brochesEnLigne[LIGNES] = {.. , .., .., ..};
//affectation des E/S GPIO aux C1,C2,C3 du clavier
uint8_t brochesEnColonne[COLONNES] = {.., .., ..};
```

Les deux tableaux **brochesEnLigne** et **brochesEnColonne** sont à compléter en fonction de votre maquette, vous utiliserez la colonne A ou la colonne B.

	A	B
L1	33	23
L2	25	4
L3	26	3
L4	27	17
C1	14	19
C2	12	18
C3	13	5



Pour vérifier le fonctionnement de votre clavier, complétez la méthode **Controler()** de la classe coffre avec le code suivant :

```
void Coffre::Controler()
{
    char touche = leClavier.getKey();
    if(touche != NO_KEY)
    {
        Serial.println(touche);
    }
}
```

Le fichier main.cpp devient alors :

```
#include <Arduino.h>
#include "coffre.h"
Coffre *leCoffre;
void setup()
{
    Serial.begin(115200);
    leCoffre = new Coffre;
}
void loop()
{
    leCoffre->Controler();
}
```

En utilisant **putty**, vous devez voir le nom de la touche enfoncée, vérifiez le bon fonctionnement de chacune des touches.

Vous réaliserez une copie d'écran de l'exécution de **putty** avec l'ensemble des touches pour compléter votre compte-rendu.

Codez le destructeur de la classe **Coffre**.

Codez la méthode **PurgerCode()**. Elle vide l'une des chaines de caractères **codeFermeture** ou **codeOuverture** en fonction de l'état dans lequel se trouve le coffre et affiche le message « Code : » sur la deuxième partie de l'écran.

Codez la méthode **AjouterChiffre()**. Le caractère **_chiffre** passé en paramètre à la fonction est ajouté à l'une des deux chaines en fonction de l'état dans lequel se trouve le coffre. L'affichage du chiffre sera remplacé par une étoile sur l'écran. Pour cela, il faut connaître la longueur du code et ajouter autant de fois que nécessaire le caractère « * » à la chaine « Code : ». Ensuite, le message ainsi constitué est écrit sur la deuxième partie de l'écran.

Réalisez le codage des méthodes **Fermer()** et **Ouvrir()** en utilisant les diagrammes de séquence suivant :

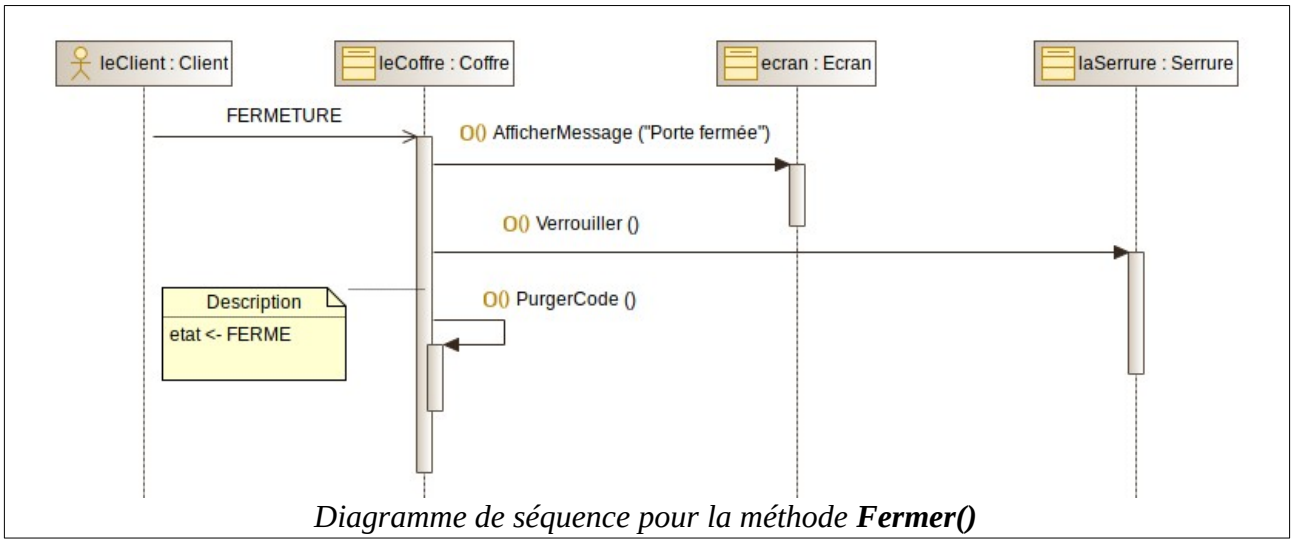


*Diagramme de séquence pour la méthode **Fermer()***

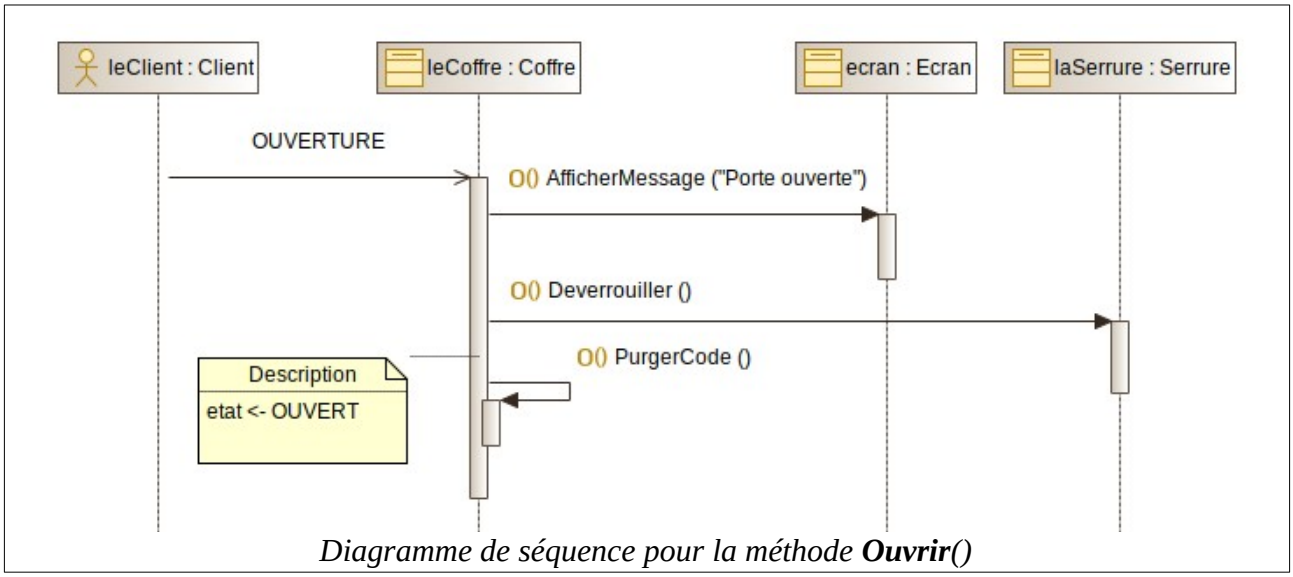


*Diagramme de séquence pour la méthode **Ouvrir()***

Vous pouvez à présent compléter le codage de la méthode **Controler()** à partir du diagramme états-transitions proposé dans l'analyse.

Vérifiez le fonctionnement de votre coffre pour la chambre d'hôtel avec une combinaison à 4 chiffres puis avec une combinaison comportant plus de chiffres.

Dans votre compte-rendu, indiquez combien de chiffres vous pouvez saisir avant que l'affichage déborde de l'écran.