

SFL6

Escape Game Téléthon :

Rapport de conception



Projet d'étude de BTS Systèmes Numériques I.R.
2020

Étudiant 1 : MONVOISIN Guillaume

Table des matières

I.	Introduction :	3
II.	Réalisation du projet :	4
a)	Rappel de la tâche de l'étudiant.....	4
b)	Communication entre les différents éléments.....	5
c)	Communication	6
d)	Chronogrammes	6
e)	Système socket	7
f)	Code du client.....	7
g)	Code du serveur.....	8
III.	Réalisation de l'application :	8
h)	Récupération de l'UID	8
i)	Allumage des LEDs et déclenchement du module relais.....	9

I. Introduction :

Rappel du cahier des charges :

Afin que la partie se déroule correctement un système pour les « médailles » est à fournir.

Ce système sera au cœur même de la salle de l'escape game et devra prendre en charges les demandes suivantes :

- ❖ Le lecteur RFID doit pouvoir lire un UID (*User Identifier*) à travers une plaque de contreplaqué de 3mm.
- ❖ Utiliser le bus SPI (*Serial Peripheral Interface*) de la Raspberry pour le contrôle des lecteurs RFID.
- ❖ Le superviseur doit pouvoir récupérer UID (*User Identifier*) afin de le changer facilement pour le programme principal dans le cas d'une perte d'un des médailles.
- ❖ Le joueur doit pouvoir constater l'allumage ou non d'une LED dans la seconde en fonction de l'état d'un lecteur RFID (si un badge ou « médaille » valide est posé dessus ou non).
- ❖ Dans le cas où le joueur place correctement quatre badges la gâche électrique doit s'activer permettant l'ouverture de la porte. Un ordre de fin de partie doit également être envoyé via un système de client socket.
- ❖ L'application de supervisions doit pouvoir détecter l'ordre de fin de partie via un système serveur socket.

II. Réalisation du projet :

a) Rappel de la tâche de l'étudiant

Au sein de ce projet d'escape game, ma partie consiste à créer un système permettant :

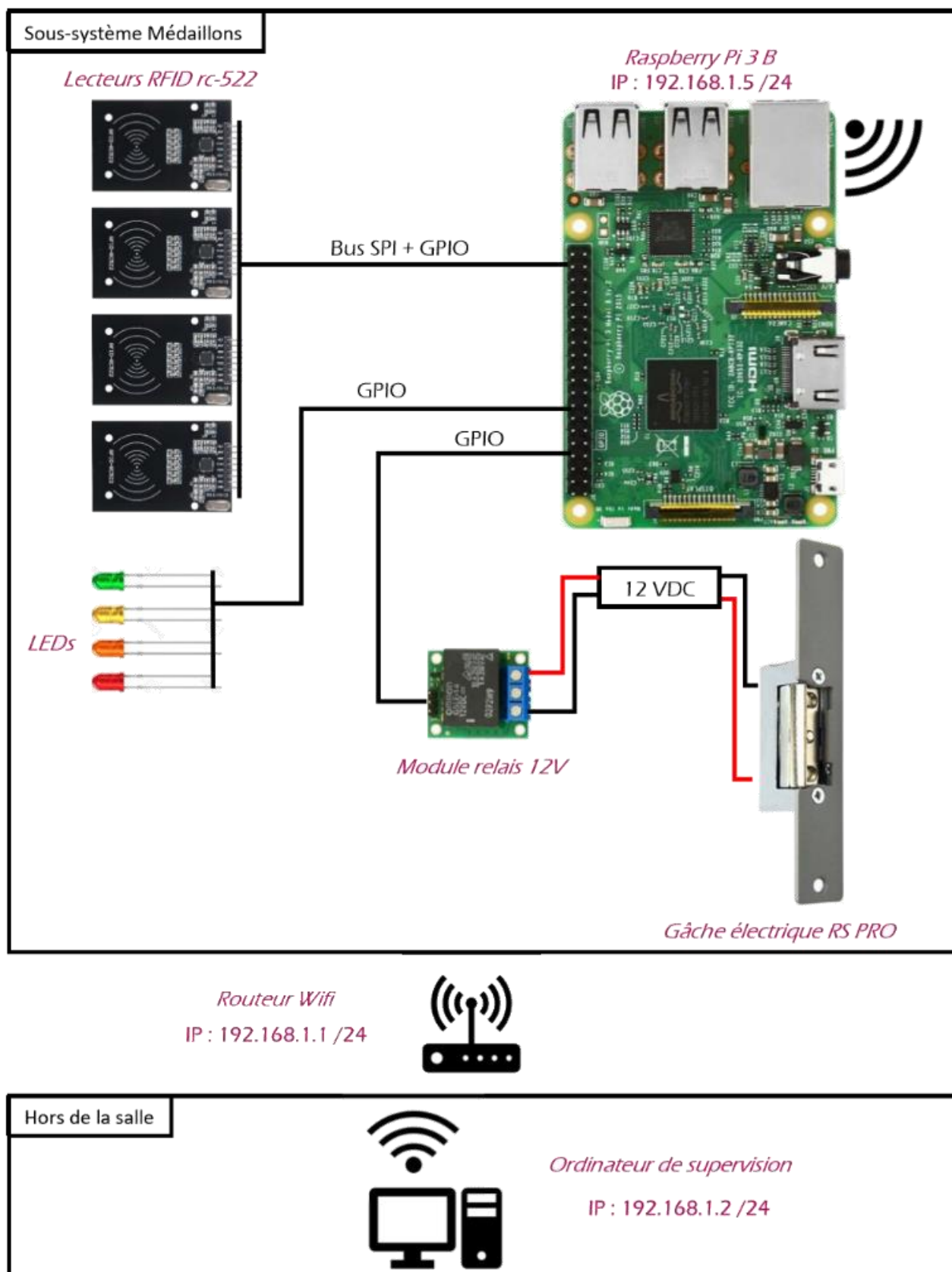
- a. La détections des « médailles ».
- b. L'allumage des différentes LED.
- c. L'ouverture de la porte.
- d. La communication entre système « médailles » et l'application de supervision

Le langage python est utilisé avec différentes librairies, comme « pi-RF522 » qui permet la gestion des lecteurs RFID, « socket » qui permet l'utilisation des services socket. Comme IDE j'ai choisi « Thonny » car très simple d'utilisation, ce choix m'était libre.

Le système doit être développé sur une carte Raspberry sous Raspbian. Le contrôle du superviseur de cette dernière sera possible via l'application « Anydesk » qui est compatible avec les deux environnements, Raspbian et Windows. Elle permet le contrôle à distance via un protocole TCP.

b) Communication entre les différents éléments

Synoptique



L'ordinateur de supervision, le routeur wifi ainsi que Raspberry sont adressés en IP statiques.

c) Communication

Le bus SPI (Serial Peripheral Interface) est un bus de données série synchrone en full-duplex. Il fonctionne sous un schéma maître-esclave. Dans ce cas le maître est la Raspberry et les esclaves sont les lecteurs RFID rc-522.

Ce modèle possède 8 interfaces dont 4 qui sont dédiés au bus SPI :

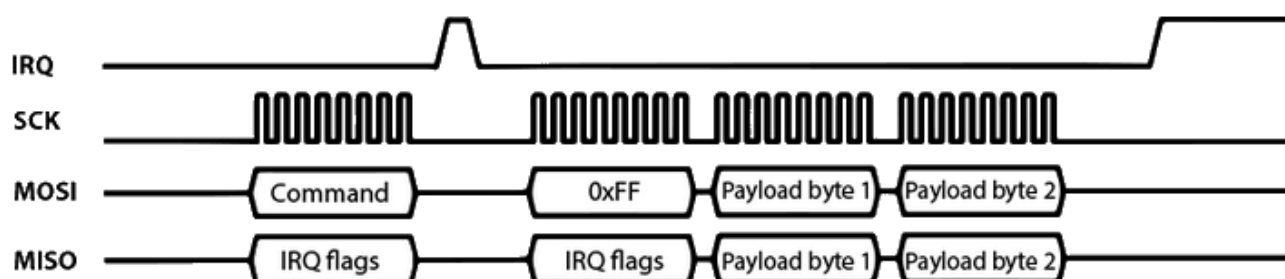
- SCK – Consacrer à l'horloge et généré par le maître.
- MOSI – Master Output, Slave Input (généré par le maître).
- MISO – Master Input, Slave Output (généré par l'esclave).
- NSS – La sélection de l'esclave (généré par l'esclave).

Les 4 interfaces restante sont :

- VCC – Alimente le module avec un voltage de 3,3 Volt.
- GND – La terre.
- RST – Permet la réinitialisation et la mise hors tension du module.
- IRQ – Permet une interruption, cela alerte le module quand un tag RFID se trouve à proximité

d) Chronogrammes

Dans le premier chronogramme on voit que quand un badge passe à proximité donc quand l'IRQ signal le passage d'un tag RFID. La première transaction qui se passe avant le signal de l'IRQ, c'est l'octet (8 bites) de commande. Dans la second transaction 24 bits son transmis, les 16 derniers bits transmis sont l'UID (User Identifier) qui est propre à chaque taf RFID.



Chronogramme 1

Dans le second chronogramme on voit comment se passe la transaction d'un octet, le bite de poids fort (MSB) est transmis en premier.



Chronogramme 2

e) Système socket

Pour envoyer l'ordre de fin partie un système de socket a été choisi avec comme « client » l'application du sous-système médailles et comme serveur l'application de supervision. Ainsi le serveur sera en « écoute » en attendant que le client envoie un message qui confirmera la réussite de l'équipe qui participera à la session de jeu de l'escape game.

Pour ce faire les informations suivantes sont requises :

Pour le client :

- Adresse IP : **192.168.1.2** (adresse IP l'ordinateur de supervision)
- Port : **8888**

Pour le serveur :

- Adresse IP : **192.168.1.5** (adresse IP de la Raspberry)
- Port : **8888**

f) Code du client

Afin que l'application puisse envoyer l'ordre de fin de partie qui un simple message « END », il faut utiliser le code suivant en python :

```
import socket

hote = '192.168.1.2'
port = 8888

socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket.connect((hote, port))
print "Connection on {}".format(port)

socket.send("END")

socket.close()
```

La méthode `import` d'accéder à différents modules, dans ce cas on importe le module socket.

Par la suite on définit l'hôte et le port de communication. On crée ensuite le socket avec la méthode `socket(socket.AF_INET, socket.SOCK_STREAM)` puis on envoie une requête de connexion au serveur avec la méthode `socket.connect()`. Si la connexion n'est pas rejetée on envoie le message de fin avec la méthode `socket.send()` puis on ferme le socket avec `socket.close()`.

g) Code du serveur

Encore à étudier

III. Réalisation de l'application :

Afin que la partie se déroule, il est nécessaire d'avoir une application qui permettra de détecter les médailles, allumer les LEDs, déclencher le module relais et enfin envoyer l'ordre de fin partie comme si les médailles sont bien placées.

h) Récupération de l'UID

En premier lieu il faut pour le superviseur pouvoir récupérer l'UID d'un badge RFID dans le cas d'une perte et/ou vol.

Pour ce faire le code suivant est utilisé.

```
import RPi.GPIO as GPIO
from piRC522 import RFID
import time

GPIO.setwarnings(False)

rc522 = RFID()

while True:
    rc522.wait_for_tag()
    (error, tag_type) = rc522.request()
    if not error:
        (error, uid) = rc522.anticoll()
        if not error:
            print("UID:" + str(uid))
            time.sleep(1)
```

Dans un premier temps importe le module RFID depuis la librairie « pi-rc522 » et on instancie la librairie avec `rc522 = RFID()`.

Ensuite on crée une boucle infinie pour lire en boucle. Dans cette boucle on utilise la fonction `wait_for_tag()` pour attendre qu'un badge RFID passe à portée de lecture. Quand un badge passe à proximité on récupère son UID avec `request()` et on nettoie les possibles collisions avec `anticoll()`, ça arrive si plusieurs badges passent à portée en même temps. Puis on affiche l'UID unique du badge avec `str()`. Pour la fin on attend 1 pour ne pas lire le badge des centaines de fois en quelques milli-secondes avec `sleep(1)`, 1 étant 1 seconde.

i) Allumage des LEDs et déclenchement du module relais

L'allumage de la LED doit pouvoir se faire dans la seconde en fonction de l'état du capteur donc si un badge RFID est proximité ou non. Pour cela le code suivant est utilisé.

```
import RPi.GPIO as GPIO
from pirc522 import RFID
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

LED = 7
GPIO.setup(LED, GPIO.OUT)
UID = [116, 96, 204, 131, 91]

rc522 = RFID()
GPIO.output(LED, GPIO.LOW)

while True:
    rc522.wait_for_tag()
    (error,tag_type) = rc522.request()
    if not error:
        (error,uid) = rc522.anticoll()
        if not error :
            if UID == uid :
                print('UID valide'.format(uid))
                GPIO.output(LED, GPIO.HIGH)
                time.sleep(1)
                GPIO.output(LED, GPIO.LOW)
            else :
                print('UID non valide'.format(uid))
                GPIO.output(LED, GPIO.LOW)
```

Une fois qu'un UID a été récupéré on peut l'utiliser pour faire un comparatif avec l'UID récupéré par le lecteur RFID et ainsi allumer la LED. Pour ce faire on définit le numéro du port GPIO qui va contrôler la LED, ici port n°7 et on active le contrôle du GPIO avec `setup(LED,GPIO.OUT)` puis on initialise ce port à l'état bas (LED éteinte) avec `output(LED,GPIO.LOW)`. Ensuite on reprend la boucle d'écoute décrite précédemment et on fait la comparaison des deux UID. Si l'UID est valide on allume la LED avec `output(LED,GPIO.HIGH)` puis on attend 1 seconde et on éteint la LED. Si le badge reste à portée du lecteur RFID la LED restera allumée car la boucle est infinie.

Pour le déclenchement du module relais on utilise sensiblement la même démarche.

Étudiant 2 : DOHIN Cyril

Table des matières

I.	Introduction.....	11
A.	Rappel du cahier des charges.....	11
II.	Réalisation du projet	12
B.	Tache de l'étudiant.....	12
C.	Langages et matériels.....	13
III.	Réalisation Application web	14
A.	Informations de connexion au serveur BDD :	14
B.	Code de sélection de la base de données :	14
C.	Formulaires de connexion :	14
D.	Gérer les comptes superviseur et les créneaux	16
E.	Réservation de créneaux	17
F.	Consulter, modifier, annuler sa réservation :	19
IV.	Réalisation système énigme musique	20
A.	Câblages des différents composants.....	20
B.	Les différentes bibliothèques utilisées	20
C.	Code de connexion	21
D.	Programme pour jouer la musique et afficher le message grâce au mot de passe	23

I. Introduction

A. Rappel du cahier des charges

Deux mécanismes :

- Détection de médailles :

Lorsque le bon code est saisi, l'application devra afficher un message sur un écran LCD et jouer la chanson « Au clair de la lune » par un buzzer.

- Enigme musique :
 - L'administrateur peut ajouter, supprimer et modifier des comptes superviseur.
 - Le superviseur peut ajouter des créneaux de jeu d'une durée d'une heure trente.
 - Tout visiteur pourra réserver un créneau libre en précisant son nom, son numéro de téléphone ainsi que le nombre de joueurs (entre trois et cinq).
 - Lors de la validation d'une réservation, le joueur est invité à se présenter quinze minutes avant l'heure de début du créneau.
 - En option qu'un SMS soit envoyé la veille du créneau au joueur et au superviseur.

II. Réalisation du projet

B. Tache de l'étudiant

Dans le système Escape Game mes taches se découpe en deux parties distinctes :

- **L'application de réservation**
- **Le Mécanisme énigme musique**

L'application de réservation consiste à créer une application web permettant aux membres de l'association de pouvoir paramétrer les jours et heures d'ouverture. Les visiteurs pourront réserver un créneau et l'administrateur devra pouvoir gérer les comptes des membres de l'association.

Liste des taches :

- Se connecter
- Gérer les comptes superviseur
- Gérer les créneaux
- Réserver un créneau
- Créer la base de données

Le mécanisme énigme musique consiste à créer une application Arduino qui va donner deux indices pour trouver un mot mystère. Il y aura deux indices : un message à afficher sur écran LCD et une musique à jouer. Le déclenchement de ses indices doit s'opérer lors de la détection sur un pavé numérique d'un code prédéfini.

Liste des taches :

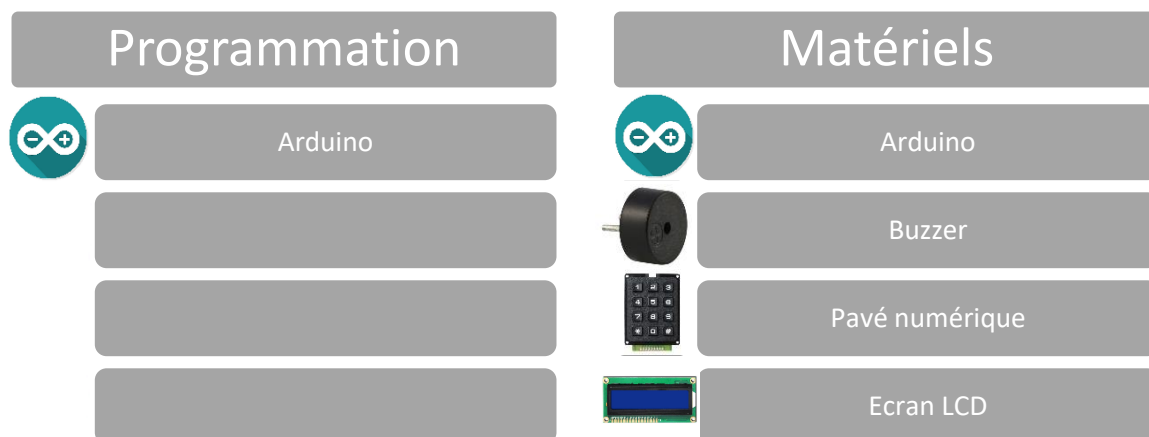
- Détecter le code numérique
- Afficher l'énigme
- Jouer la musique

C. Langages et matériels

Application de réservation :



Mécanisme énigme musique :



III. Réalisation Application web

A. Informations de connexion au serveur BDD :

Pour se connecter au serveur de base de données hébergé en local sur l'ordinateur, les informations suivantes sont requises :

- Base de données : **projet_telethon**
- Utilisateur : **root**
- Mot de passe : « »

B. Code de sélection de la base de données :

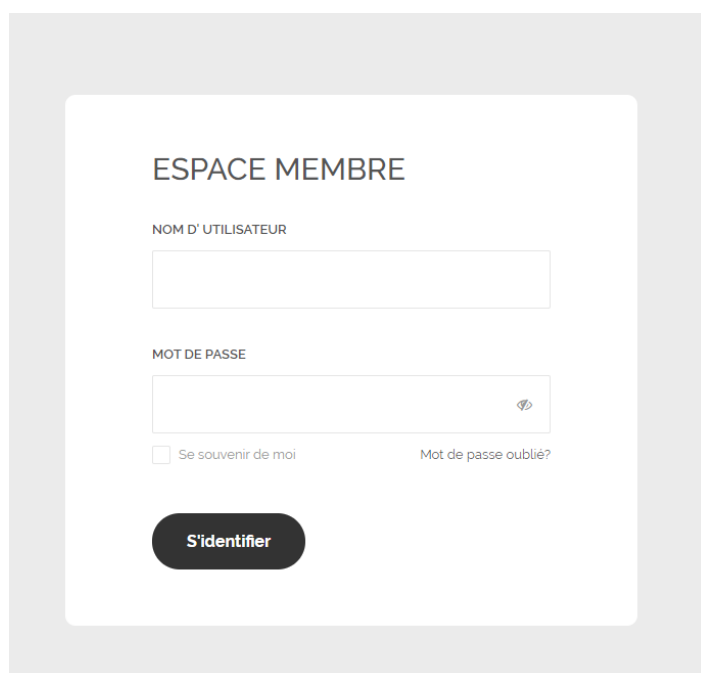
Afin que l'application puisse communiquer avec la base de données, il est nécessaire d'employer le code suivant :

```
<?php
//Permet de se connecter à la BDD
$pdo = new PDO("mysql:dbname=projet_telethon;host=localhost", "root", "");

// Requete SQL permettant de sélectionner la table "users"
$stmt = $pdo->prepare("SELECT * FROM users ORDER BY id");
$stmt->execute(); // Permet d'exécuter la requête
// Enregistre les informations de la BDD dans la variables "types"
$types = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

C. Formulaires de connexion :

Pour permettre à l'administrateur et au superviseur d'accéder à leurs page sans que les joueurs y est accès mais aussi pour permettre aux joueurs de consulter leurs créneaux deux pages d'authentification doit être mise en œuvre.



Cette page d'authentification doit pouvoir rediriger l'administrateur et le superviseur vers leurs pages dédiées, une fois leurs noms d'utilisateurs et leurs mots de passe renseignés.

Pour cela l'administrateur et le superviseur doivent être attribuée à un rôle définit :

- Rôle administrateur permettant d'accéder à la page pour gérer les comptes superviseurs.
- Rôle superviseur permettant d'accéder à la page pour gérer les créneaux.

Pour cela j'ai créé deux fonctions, une qui permet de créer une session et l'autre de la détruire :

```
function init_php_session() : bool // Permet d'initialiser une session
{
    if (!session_id()) // Récupère ou définit l'identifiant de session pour la session courante.
    {
        session_start(); // Démarre une nouvelle session ou reprend une session existante
        session_regenerate_id(); // Remplace l'identifiant de session courant par un nouveau
        return true; // Retourner vrai
    }
    return false; // Sinon retourner faux
}

function clean_php_session() : void // Permet de détruire une session
{
    session_destroy();
}
```

Ensuite j'ai créé une fonction qui permet de savoir si une session est en cours

```
function is_logged() : bool // Permet de savoir si il y a une session en cours
{
    if(isset($_SESSION['login'])) // Si il y a une session d'ouverte
    {
        return true; // Retourner vrai
    }
    return false; // Sinon retourner faux
}
```

et deux autres fonctions permettant d'identifier si l'utilisateur est un administrateur ou un superviseur :

```
function is_admin() : bool // Permet de savoir si l'utilisateur connecté est un admin
{
    if(is_logged()) // Si il y a une session en cours
    {
        if(isset($_SESSION['roles_id']) && $_SESSION['roles_id'] == 1){
            // Si une variable est déclarée et est différente de NULL ET si "roles_id" = 1
            return true; // Retourner vrai
        }
    }
    return false; // Sinon retourner faux
}

function is_superviseur() : bool // Permet de savoir si l'utilisateur connecté est un superviseur
{
    if(is_logged()) // Si il y a une session en cours
    {
        if(isset($_SESSION['roles_id']) && $_SESSION['roles_id'] == 2){
            // Si une variable est déclarée et est différente de NULL ET si "roles_id" = 2
            return true; // Retourner vrai
        }
    }
    return false; // Sinon retourner faux
}
```

Pour finir grâce aux fonctions `is_admin()` et `is_superviseur()` je peux rediriger convenablement. si l'administrateur est connecté je le redirige vers la page pour modifier les comptes superviseurs, si le superviseur est connecté je le redirige vers la page pour modifier les créneaux.

```
if (is_admin()){ // Appel de la fonction is_admin (si l'admin est connecté)
    header('Location: modification_superviseurs.php'); // rediriger vers la page

} elseif (is_superviseur()){ // Appel de la fonction is_superviseur (si le superviseur est connecté)
    header('Location: modification_creneau2.php'); // rediriger vers la page
}
```

D. Gérer les comptes superviseur et les créneaux

L'administrateur doit pouvoir modifier, supprimer et ajouter des comptes superviseurs.

Liste des superviseurs

Nom utilisateur	Mot de passe	Modifier	Supprimer	Ajouter compte superviseur
superviseur	superviseur	Modifier	Supprimer	
admin	admin	Modifier	Supprimer	

Liste des creneaux

Statut	Date	Heure	Modifier	Supprimer	Ajouter créneaux
Réservé	2020-04-09	10H00	Modifier	Supprimer	
Libre	2020-04-04	11H00	Modifier	Supprimer	

Permet de supprimer grâce au paramètre « supprimer » passé par l'url. Lors de la requête SQL le serveur peut capter l'identifiant du compte superviseur à supprimer.

```
if (isset($_GET['supprimer'])){ //Si la variable "supprimer" passé par l'url est déclarée et est différente de NULL

    // Requete SQL permettant de supprimer la table "users"
    $stmt = $pdo->prepare("DELETE FROM users WHERE id = :monid");
    //Associe la valeur $_GET['supprimer'] à un paramètre
    $stmt->bindValue(":monid", $_GET['supprimer'], PDO::PARAM_INT);
    $stmt->execute(); // Permet d'executer la requête
}
```


La modification suit le même principe que la suppression. La requête permet de modifier le libellé pour un « id » donné. La fonction « bindValue() » s'occupe de faire l'association des éléments récupérés par le serveur avec les variables SQL.

```
if (isset($_POST['modifier'])) { //Si la variable "modifier" passé par le formulaire est déclarée et est différente de NULL

    // Requete SQL permettant de mettre à jour la table "users" en associant les colonnes de la BDD à des paramètres
    $stmt = $pdo->prepare("UPDATE users SET login = :monNom, password = :mdp WHERE id = :monid");
    //Associe la valeur "$_POST" (informations rentrées dans le formulaire) au paramètre correspondant
    $stmt->bindValue(":monid", $_POST['modifier'], PDO::PARAM_INT);
    $stmt->bindValue(":monNom", $_POST['Nom'], PDO::PARAM_STR);
    $stmt->bindValue(":mdp", $_POST['Mdp'], PDO::PARAM_STR);
    $stmt->execute(); // Permet d'exécuter la requête
}
```

Pour ajouter il suffit de se connecter à la BDD, de récupérer le login et le mot de passe passé dans le formulaire et de faire une requête d'insertion.

```
//Requête d'insertion
$sql_query = "INSERT INTO users(id, login, password, roles_id) VALUES (0, :login,:password, 2)";
```

Ajouter superviseur

Ajouter

Ajouter un créneau

Ajouter

E. Réservation de créneaux

Les joueurs doivent pouvoir réserver un créneaux. Voici un calendrier avec les créneaux disponibles. Le joueur doit cliquer sur l'heure qu'il désire.

Avril 2020

Les heures en verts sont les créneaux disponibles

Lundi 30	Mardi 31	Mercredi 01	Jeudi 02	Vendredi 03	Samedi 04 11H00 - 12H30	Dimanche 05
06	07	08 07H00 - 08H30	09	10	11	12
13	14	15	16 19H00 - 20H30 15H01 - 16H31	17 12H00 - 13H30	18	19 10H00 - 11H30
20	21 09H00 - 10H30	22	23	24	25	26
27	28	29	30	01	02	03

Fonctions du calendrier :

Voici le constructeur du calendrier :

```
public function __construct(?int $month = null, ?int $year = null) { // Constructeur du Mois

    if ($month === null || $month < 1 || $month > 12) { // Si le mois est null OU inférieur à 1 OU supérieur à 12
        $month = intval(date('m')); // "month" prend le mois en chaine de caractère
    }

    if ($year === null) { // Si l'année est null
        $year = intval(date('Y')); // "année" prend l'année en chaine de caractère
    }

    $this->month = $month; // Accès à la propriété "mois" qui prend la valeur passé au constructeur
    $this->year = $year; // Accès à la propriété "année" qui prend la valeur passé au constructeur
}
```

La fonction getWeeks permet de retourner le nombre de semaines dans le mois :

```
public function getWeeks(): int { // Retourne le nombre de semaines dans le mois

    $start = $this->getStartingDay(); // Premier jour du mois
    $end = $start->modify('+1 month -1 day'); // Dernier jour du mois
    $startWeek = intval($start->format('W')); // Première semaine du mois
    $endWeek = intval($end->format('W')); // Dernière semaine du mois

    if ($endWeek === 1) { // Si la dernière semaine du mois est égale à 1
        $endWeek = intval($end->modify('-7 days')->format('W')) + 1;
    }

    $weeks = $endWeek - $startWeek + 1; // "weeks" prend le nombre de semaines
    if ($weeks < 0) { // Si le nombre de semaines est inférieur à 0
        $weeks = intval($end->format('W')); // Le nombre de semaines égale au numéro de semaines de la date de fin
    }

    return $weeks;
}
```

Ces deux fonctions permettent de naviguer entre les mois :

```
public function nextMonth(): Month { // Permet de naviger vers les mois suivants
    $month = $this->month + 1; // Le mois prend +1
    $year = $this->year;

    if ($month > 12) { // Si le mois est supérieur à 12
        $month = 1; // Le mois est égale à 12
        $year += 1; // Incrémentation de l'année
    }

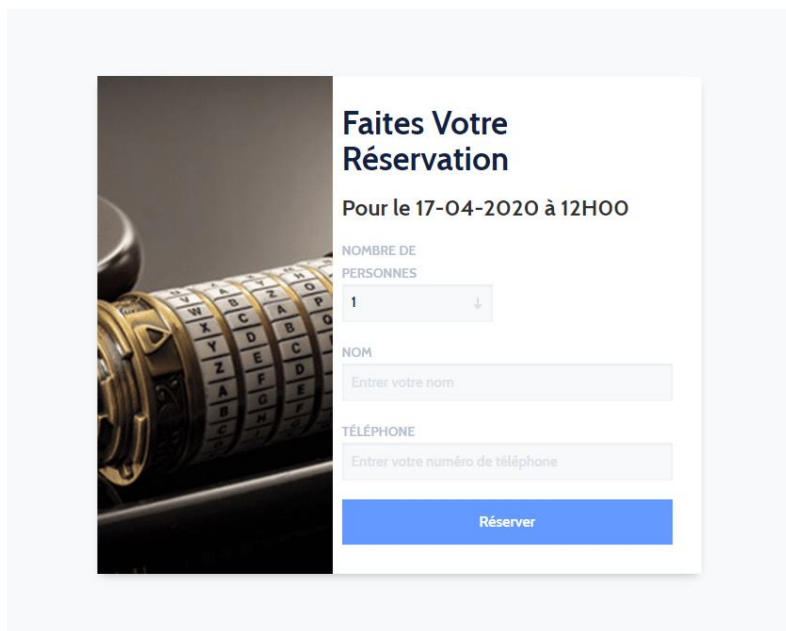
    return new Month($month, $year);
}

public function previousMonth(): Month { // Permet de naviger vers les mois précédents
    $month = $this->month - 1; // Le mois prend -1
    $year = $this->year;

    if ($month < 1) { // Si le mois est inférieur à 1
        $month = 12; // Le mois est égale à 12
        $year -= 1; // Décrémentement est égale à 12
    }

    return new Month($month, $year);
}
```

Une fois la date et l'heure choisie, le joueur est redirigé vers un formulaire où il doit renseigner le nombre de personnes, son nom et son numéro de téléphone.



Une fois la réservation effectuée, un message de remerciement est affiché avec un mot de passe permettant au joueur d'accéder à la page « Ma réservation » pour consulter, modifier ou annuler sa réservation.

TÉLÉTHON ST COLOMBAN 

MENU RÉSERVER MA RÉSERVATION ESPACE MEMBRE CONTACT SE DÉCONNECTER

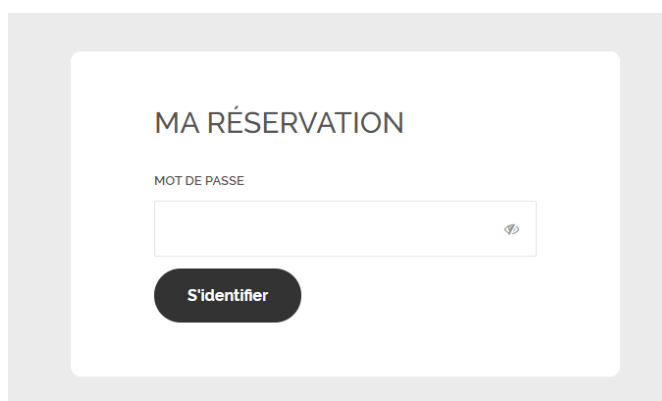
Merci!

Merci pour votre inscription du 04-04-2020 à 11H00, Vous pouvez consulter, modifier ou annuler votre réservation en accédant à "Ma réservation" grâce à ce mot de passe : 5fT94dV1xz


F. Consulter, modifier, annuler sa réservation :

Pour consulter, modifier ou annuler sa réservation le joueur pourra accéder à sa réservation grâce au mot de passe fourni à la fin de sa réservation.

Connexion :



Ma réservation :



Ma Réservation

Pour le 16-04-2020 à 15H01

NOMBRE DE PERSONNES
1

NOM
jo

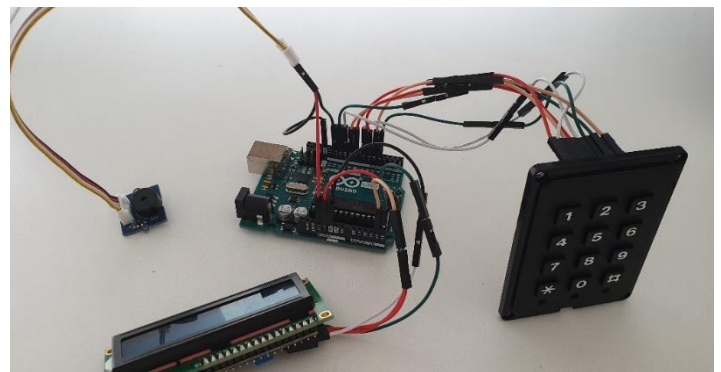
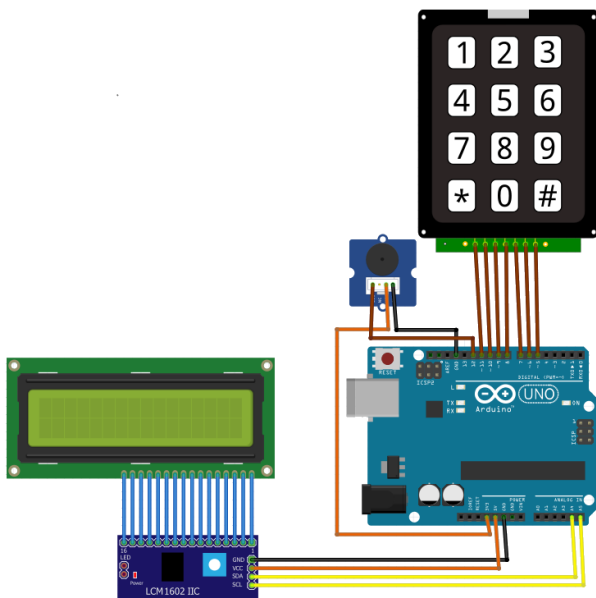
TÉLÉPHONE
0670449615

Modifier

Annuler réservation

IV. Réalisation système énigme musicale

A. Câblages des différents composants



B. Les différentes bibliothèques utilisées

`#include <Keypad.h>` : Aide à l'abstraction matérielle, et à l'amélioration de la lisibilité du code. Automatise les instructions `pinMode` et `digitalRead` appelées par l'utilisateur.

`#include <Wire.h>` : Permet la gestion du bus I2C.

`#include <LiquidCrystal I2C.h>` : Permet de contrôler les écrans LiquidCrystal (LCD) basés sur le chipset Hitachi HD44780, que l'on trouve sur la plupart des LCD textuels .

`#include <Password.h>` : Simplifie la gestion des mots de passe .

C. Code de connexion

- **Pavé numérique :**

Il faut tout d'abord déclarer la bibliothèque « Keypad.h », puis déclarer les lignes et les colonnes, ensuite il faut connecter les broches aux lignes ou colonnes correspondant.

```
#include <Keypad.h>

const byte ROWS = 4; // 4 lignes
const byte COLS = 4; // 3 colonnes

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

byte rowPins[ROWS] = {11, 10, 9, 8}; // Connection aux broches des lignes du clavier
byte colPins[COLS] = {4, 7, 6, 5}; // Connection aux broches des colonnes du clavier
```

La bibliothèque « keypad » inclue un constructeur qui crée un « objet clavier », il faut ensuite ouvrir le port série à 9600 bps.

```
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS ); // Constructeur qui crée un "objet clavier"
// userKeymap : 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

void setup(){
  Serial.begin(9600); // Ouvre le port série à 9600 bps
}
```

Pour finir « keypad.getKey() » renvoie la touche appuyée et « Serial.println » l'écrit sur le port série.

```
void loop(){
  char key = keypad.getKey(); // Renvoie la touche qui est appuyée sous forme caractère ASCII

  if (key){ // Si "key" est activé
    Serial.println(key); // Imprime les données de "key" sur le port série sous forme de
                        // texte ASCII lisible par l'homme suivi d'un caractère de retour chariot
  }
}
```

- **Ecran LCD :**

Il faut tout d'abord déclarer les bibliothèques « Wire.h » et « LiquidCrystal_I2C.h », puis définir la taille de l'écran (ici 16 par 2).

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x20,16,2); // Définition de la taille de l'écran
```

Il faut ensuite initialiser l'écran et la technologie Backlight. Pour finir il reste à écrire le message.

```
void setup()
{
  lcd.init(); // Initialisation de l'écran

  lcd.backlight(); // Initialisation de la technologie de rétroéclairage Backlight déployée dans certains écrans LED

  lcd.print("Bonjour les amis"); // Ecriture du message
}

void loop()
{
}
```

- **Buzzer :**

```
// les constantes des fréquences de base
const char DON = 65;
const char DOD = 69;
const char REN = 74;
const char RED = 78;
const char MIN = 83;
const char FAN = 87;
const char FAD = 93;
const char SON = 98;
const char SOD = 104;
const char LAN = 110;
const char LAD = 117;
const char SIN = 123;
//Le tableau pour la mélodie
char auClair[11][3]={
  DON, 2, 2,
  DON, 2, 2,
  DON, 2, 2,
  REN, 2, 2,
  MIN, 4, 2,
  REN, 4, 2,
  DON, 2, 2,
  MIN, 2, 2,
  REN, 2, 2,
  REN, 2, 2,
  DON, 8, 2
};
```

```

int dureeBase=500; //on fixe la durée de basse à 500 millisecondes
unsigned long tempsDep; // variable pour le temps de départ
unsigned long tempsAct; // variable pour le temps actuel
int duree; //variable pour la durée d'attente de la note en cours
int n=0; // position dans le tableau de mélodie

void setup(){
  pinMode(12,OUTPUT); //on met le pin 3 en mode OUTPUT
  tempsDep=millis(); // on initialise le temps de départ au temps Arduino
  duree=0; //on initialise l'attente à 0
}

void loop(){
  tempsAct=millis(); // on récupère le temps Arduino
  if (tempsAct-tempsDep>=duree){ // on regarde si le temps est écoulé
    noTone(12); // on stoppe le son
    delay(10); // délai pour l'attaque du son
    joueNote(auClair[n][0],auClair[n][2]); // on appelle la fonction qui joue la bonne note
    duree=dureeBase*auClair[n][1]; // on fixe la duree d'attente
    tempsDep=tempsAct; //on initialise le temps de départ
    n++; // on incrémente la position dans le tableau
    if (n>10) // on teste si on dépasse la fin du tableau
      n=0; // on revient au début du tableau
  }
  // on peut placer ici du code à exécuter en attendant
  // il faut bien-sûr ne pas utiliser la fonction delay() ;)
}

// fonction de calcul de la fréquence en fonction de l'octave
void joueNote(int nt,int oc){
  tone(12,nt*pow(2,oc)); //on joue la note à la bonne fréquence
}

```

D. Programme pour jouer la musique et afficher le message grâce au mot de passe

Définition des touches pour valider et réinitialiser

```

void kpdEvent (KeypadEvent Key)
{
  switch (kpd.getState())
  {
    case PRESSED :
      switch (Key)
      {
        // appui sur '*' -> vérification de la saisie en cours
        case '*' : checkPassword(); break;
        // appui sur '#' -> réinitialisation de la saisie en cours
        case '#' : pwd.reset(); lcd.clear(); noTone(12);break;
        // sinon on ajoute le chiffre à la combinaison
        default : pwd.append(Key); break;
      }
      default : break;
  }
}

```

Vérification du mot de passe une fois que la touche pour valider est appuyée. Si le mot de passe est bon, on affiche « bravo » et on joue la musique.

```
void checkPassword(void)
{

// si le mot de passe est juste...
if (pwd.evaluate())
{
    jouer();// Fonction pour jouer la musique
    lcd.backlight();
    lcd.print("Bravo !!!!");
    pwd.reset(); // on remet à zéro la saisie
}
// si le mot de passe est faux...
else
{
    lcd.backlight();
    lcd.print("erreur, réessayez");
    pwd.reset(); // on remet à zéro la saisie
}
// on remet à zéro systématiquement après avoir vérifié pour ne pas avoir d'erreur
pwd.reset();
```


Étudiant 3 : GUIGAND Nathan

Table des matières

I.	Introduction :	26
II.	Réalisation du projet :	27
a)	Tâches étudiant :	27
b)	Interconnexion du réseau :	27
c)	Communication :	27
d)	Informations de connexion au serveur BDD :	13
e)	Code de connexion à la base de données :	14
III.	Réalisation de l'application de supervision :	30
a)	Lancement :	30
b)	Superviser :	31

I. Introduction :

Rappel du cahier des charges :

Afin de superviser la partie durant le déroulement de celle-ci, une application de supervision est à fournir.

L'installation de celle-ci sera sur un ordinateur situé en dehors de la salle de l'escape game.

L'application de supervision doit respecter le cahier des charges suivant :

- ✓ Création d'une partie : Le superviseur saisit le nom de l'équipe et lance manuellement la partie. Un chronomètre d'une durée d'une heure doit ensuite se lancer automatiquement.
- ✓ Visualiser la partie : Depuis l'application, le superviseur doit avoir accès aux images provenant de la caméra de surveillance installée au sein de la salle du jeu.
- ✓ Interagir avec les joueurs : Le superviseur a la possibilité d'envoyer un message écrit à l'afficheur situé dans la salle du jeu. Ce message peut être prédéfini ou être nouvellement saisi.
- ✓ Connaître les résultats : A la fin de la partie, qu'elle soit gagnée ou perdue, le superviseur peut connaître le score de l'équipe actuelle. De plus, cette équipe doit pouvoir être comparée avec d'autres équipes. Un taux de réussite en pourcentage sera calculé et affiché.

La communication entre les différents équipements locaux nécessaires à l'escape game se fera grâce à une intercommunication locale basée sur le protocole IP et gérée par un routeur Wi-Fi.

II. Réalisation du projet :

j) Tâches de l'étudiant :

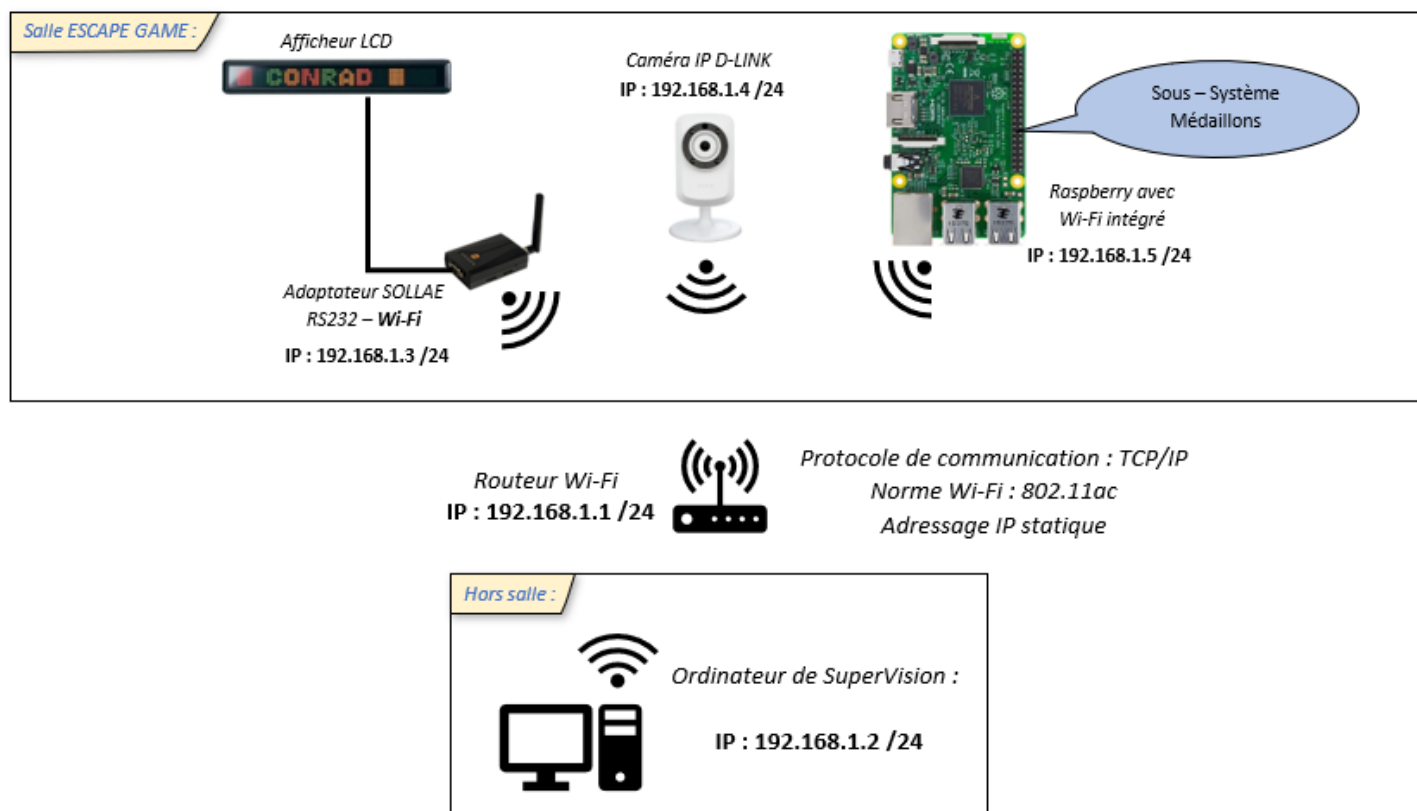
Au sein du projet, les parties me concernant sont :

- Le développement de l'application de supervision en respectant le cahier des charges.
- L'établissement et la mise en place de l'intercommunication réseaux entre les différents matériels.

Le développement de l'application doit être pour un ordinateur sous Windows. J'ai ainsi choisi le langage C# et l'IDE Visual Studio comme environnement de développement. Le choix de ces deux derniers m'était libre.

La communication réseau doit être en Wi-Fi. L'afficheur LCD ne comportant pas de carte Wi-Fi, je vais ajouter un adaptateur RS232 / Wi-Fi de type SOLLAE CSW-H85K présent en nombre suffisant au sein de la section.

k) Interconnexion du réseau :



L'intégralité des matériels ci-dessus doivent être adressés en IP statiques.

l) Communication :

La communication réseau entre les différents matériels se fera par le Wi-Fi en utilisant le protocole TCP/IP.

La communication entre l'adaptateur SOLLAE RS232/Wi-Fi et l'afficheur LCD se fera grâce au protocole RS232.

Enfin, la communication entre la base de données hébergeant les indices et les équipes avec l'application de supervision se fera par l'intermédiaire de localhost (127.0.0.1). En effet, la base de données MySQL sera installée localement sur le même poste que l'application de supervision.

m) Informations de connexion au serveur BDD :

Pour se connecter au serveur de base de données hébergé en local sur l'ordinateur, les informations suivantes sont requises :

- Adresse IP : **127.0.0.1**
- Base de données : **dbsupervision**
- Utilisateur : **superviseur**
- Mot de passe : **Nantes44**

n) Code de connexion à la base de données :

Afin que l'application puisse communiquer avec la base de données, il est nécessaire d'employer le code suivant (C#) :

```
private void InitConnexion()  
{  
    // Création de la chaîne de connexion  
    string connectionString = "SERVER=127.0.0.1; DATABASE=dbsupervision; UID=superviseur; PASSWORD=Nantes44";  
    this.connection = new MySqlConnection(connectionString);  
}
```

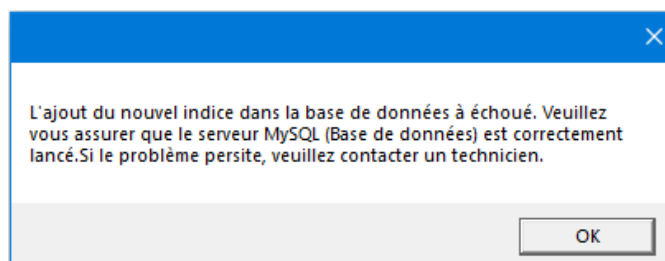
La méthode d'initialisation de connexion ci-dessus permet de renseigner l'adresse du serveur MySQL, l'utilisateur et le mot de passe nécessaire pour la connexion avec la base de données. Je définie ensuite l'élément *connection* étant un type *MySqlConnection()* (déclarer au préalable).

La méthode suivante permet quant à elle l'exécution de la requête SQL depuis l'application. Cette méthode est identique qu'il s'agisse de l'envoi de données ou de la récupération.

L'exemple suivant permet l'ajout d'un indice dans la base de données, donc un envoi depuis l'application.

L'utilisation des objets passés en paramètres varie en fonction des paramètres renseignés dans la requête SQL.

En cas de problème, il est judicieux de mettre en place un message d'information en cas de mauvais envoi. Celui-ci s'effectue donc avec la condition « *try* » / « *catch* ». On essaye ainsi l'exécution de la requête dans le « *try* ». Si cette action est impossible, on entre dans le « *catch* ». L'élément « *catch* » contient un élément de type *MessageBox.Show* qui permet d'afficher, dans une nouvelle vignette, du texte à l'utilisateur.



```
public void AddIndice(Indice indice)
{
    try
    {
        // Ouverture de la connexion SQL
        this.connection.Open();

        // Création d'une commande SQL en fonction de l'objet connection
        MySqlCommand cmd = this.connection.CreateCommand();

        // Requête SQL
        cmd.CommandText = "INSERT INTO tbindice (id, text) VALUES (@id, @text)";

        // Utilisation de l'objet contact passé en paramètre
        cmd.Parameters.AddWithValue("@id", indice.Id);
        cmd.Parameters.AddWithValue("@text", indice.Text);

        // Exécution de la commande SQL
        cmd.ExecuteNonQuery();

        // Fermeture de la connexion
        this.connection.Close();
    }

    catch
    {
        MessageBox.Show("L'ajout du nouvel indice dans la base de données à échoué. " +
            "Veuillez vous assurer que le serveur MySQL (Base de données) est correctement lancé.");
    }
}
```

L'ouverture de la connexion SQL par la méthode *Open()* exécute la chaîne de connexion.

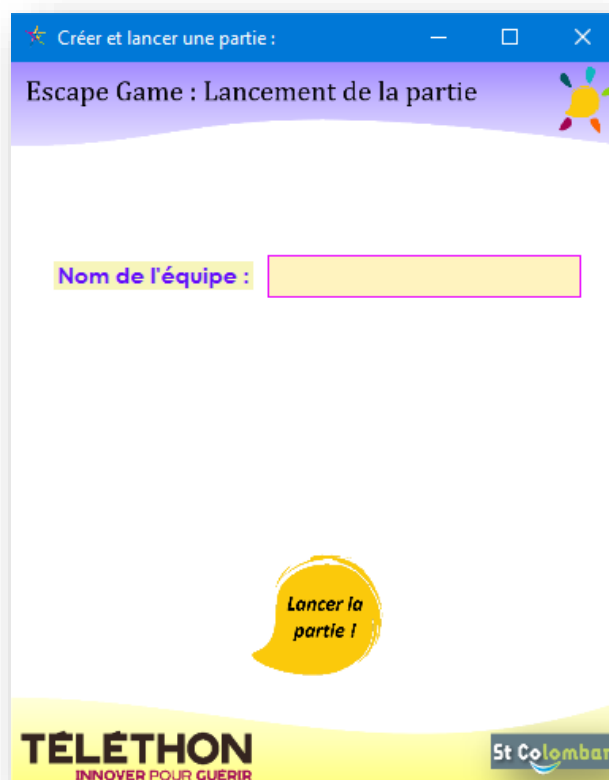
A l'inverse, ma méthode *Close()* ferme la connexion établie avec la base de données.

III. Réalisation de l'application de supervision :

Afin que le superviseur de la partie puisse visualiser et aiguiller l'équipe qui joue, il est nécessaire d'avoir une application dédiée qui communique entre les éléments d'interactions utilisateur et superviseur.

a) Lancement :

Dès le lancement de l'application par double-clic sur l'icône, cette page s'affiche :



Le superviseur est invité à saisir le nom choisi par l'équipe dans le champ correspondant. Ensuite, le clic sur « **Lancer la partie !** » permet d'enregistrer l'équipe dans la base de données et d'appeler la fenêtre principale de l'application. Pour ce faire, j'utilise le code suivant :

```
private void btnWindow_superviser_Click(object sender, EventArgs e)
{
    // Créer une équipe à ajouter
    Equipe equipe = new Equipe();
    equipe.Nom = TxtBoxEquipe.Text;
    equipe.Score = "0000";
    equipe.Heure_fin = "00:00:00";

    // Création de l'objet Bdd pour l'interaction avec la base de donnée MySQL
    Bdd bdd = new Bdd();
    bdd.AddEquipe(equipe);

    MainWindow FenSuperviser = new MainWindow();
    FenSuperviser.ShowDialog();

    this.Close();
}
```

b) Superviser :

La fenêtre principale de l'application est celle de supervision. Depuis celle-ci, on peut visualiser l'image en direct provenant de la caméra IP (située dans la salle de l'escape game), appeler la fenêtre d'envoi d'indice afin d'envoyer un indice à l'afficheur. Le superviseur a également la possibilité de surveiller le chronomètre.



Premièrement, j'utilise l'élément `WebBrowser` pour récupérer l'image de la caméra IP :

L'image de la caméra D-Link est fournie par l'intermédiaire de son adresse IP. Afin de pouvoir l'afficher dans l'application, l'utilisation de l'élément `WebBrowser` est la plus adaptée. Il s'agit en effet d'un navigateur web simplifié et allégé fourni par Microsoft et facilement intégrable dans un code en XAML.

```
<WebBrowser x:Name="wbDLink" Source="https://192.168.1.4/" />
```

L'image fournie étant fixe, il est nécessaire d'utiliser une fonction de rafraîchissement pour que celle-ci soit fluide :

```
private DispatcherTimer timercam; //Minuterie pour la caméra IP.

// Fonction de rafraîchissement de l'image de la caméra IP :
timercam = new DispatcherTimer();
timercam.Interval = TimeSpan.FromSeconds(5);
timercam.Tick += timercam_Tick;
timercam.Start();
```

Dans l'exemple précédent, le rafraîchissement de l'image s'effectuait toutes les cinq secondes. Pour une utilisation optimale, je programme celui-ci toute les secondes.

En second lieu, le chronomètre utilise également la minuterie *DispatcherTimer*. Cette minuterie fiable permet son intégration dans de nombreux cas. Basée sur l'intervalle, je l'utilise à deux répétitions au sein de l'application de supervision. Voici le code qui permet de faire fonctionner le chronomètre de la partie et qui y met fin lorsque celui-ci arrive à son terme :

```
private DispatcherTimer timerchrono; //Minuterie pour le chronometre.

private int time = 3600; //1h = 3600s

//Fonction du chronometre :
timerchrono = new DispatcherTimer();
timerchrono.Interval = new TimeSpan(0, 0, 1);
timerchrono.Tick += timerchrono_Tick;
timerchrono.Start();

void timerchrono_Tick(object sender, EventArgs e)
{
    if (time > 0)
    {
        time--;
        TBCountDown.Text = string.Format("00:{00}:{1}", time / 60, time % 60);
    }
    else
    {
        timerchrono.Stop();
        Window_fin Fen = new Window_fin();
        Fen.ShowDialog();
    }
}
```

Ensuite, le bouton « **Envoyer un indice** » appelle une nouvelle fenêtre qui elle permet de sélectionner un indice prédéfini ou d'en saisir un nouveau. Se référer à la partie « Envoi d'indice ».

Ci-dessous le code qui est appelé lors du clic sur ce bouton :

```
// Action du clic sur l'option d'envoyer un indice :
1 référence
private void btnWindow_indice_Click(object sender, EventArgs e)
{
    Window_indice Fen = new Window_indice();
    Fen.ShowDialog();
}
```

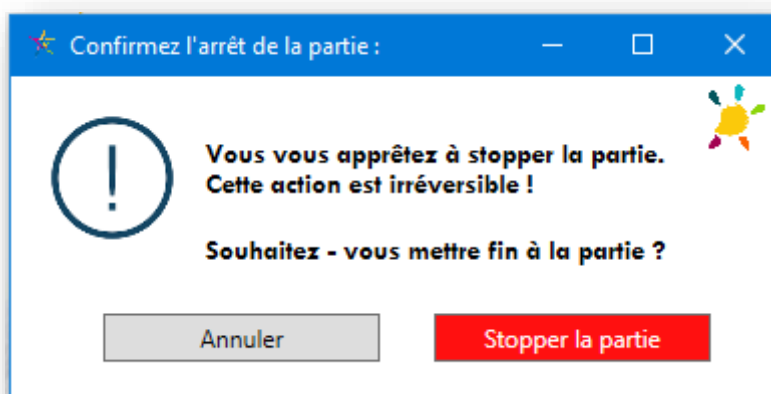
Sous ce bouton, se trouve une zone qui permet d'afficher le dernier indice envoyé à l'afficheur. Le code est présent ci-dessous :

* CODE RECUPERATION DE L'INDICE ENVOYE *

De même, le nom de l'équipe est récupéré en haut de la fenêtre. Pour ce faire, j'utilise la communication avec la base de données suivante :

```
string connectionString = "SERVER=127.0.0.1; DATABASE=dbsupervision;" +  
    " UID=superviseur; PASSWORD=Nantes44";  
string myConnection = connectionString;  
MySQLConnection myConn = new MySqlConnection(myConnection);  
myConn.Open();  
  
string sql2 = "SELECT nom FROM tbequipe WHERE date = (SELECT MAX(date)) " +  
    "ORDER BY date DESC LIMIT 1 ";  
MySQLCommand cmd2 = new MySqlCommand(sql2, myConn);  
TxtEquipe.Text = cmd2.ExecuteScalar().ToString();
```

Enfin, si le superviseur souhaite, pour quelconques raisons, mettre fin prématurément à la partie, il suffit de cliquer sur l'option « **Stopper la partie** ». Le clic sur celle-ci ouvre la fenêtre suivante :



L'option « **Annuler** » ferme la fenêtre ci-dessus et reviens à la fenêtre de supervision sans avoir le moindre effet sur la partie en cours. L'option « **Stopper la partie** » met quant à elle fin à la partie en appelant la fenêtre de fin de partie.