

SAE "transmissions 100 fils"

Cecchini Thomas - Ocana Kévin - Ben Oiel Hugo - Soragna Quentin

Introduction :

Dans le cadre de cette SAÉ nous allons mettre en œuvre plusieurs briques afin de réaliser un escape game présenté aux 1ères années.

Pour se faire notre escape game sera décomposé en 2 Briques :

Briques 1 : Serveur de jeu et logs

Briques 2 : Systèmes de Transmissions (HF, IR et radio)

Ces briques seront accompagnées de petites énigmes faisant office de liaison pour rendre notre escape game plus optimisé.

Dans le cadre de ce rapport chaque étape sera détaillée joint de photos/vidéos avec des explications des programmes pour que notre projet soit bien compris par tout type de personne et facilement reproductible.

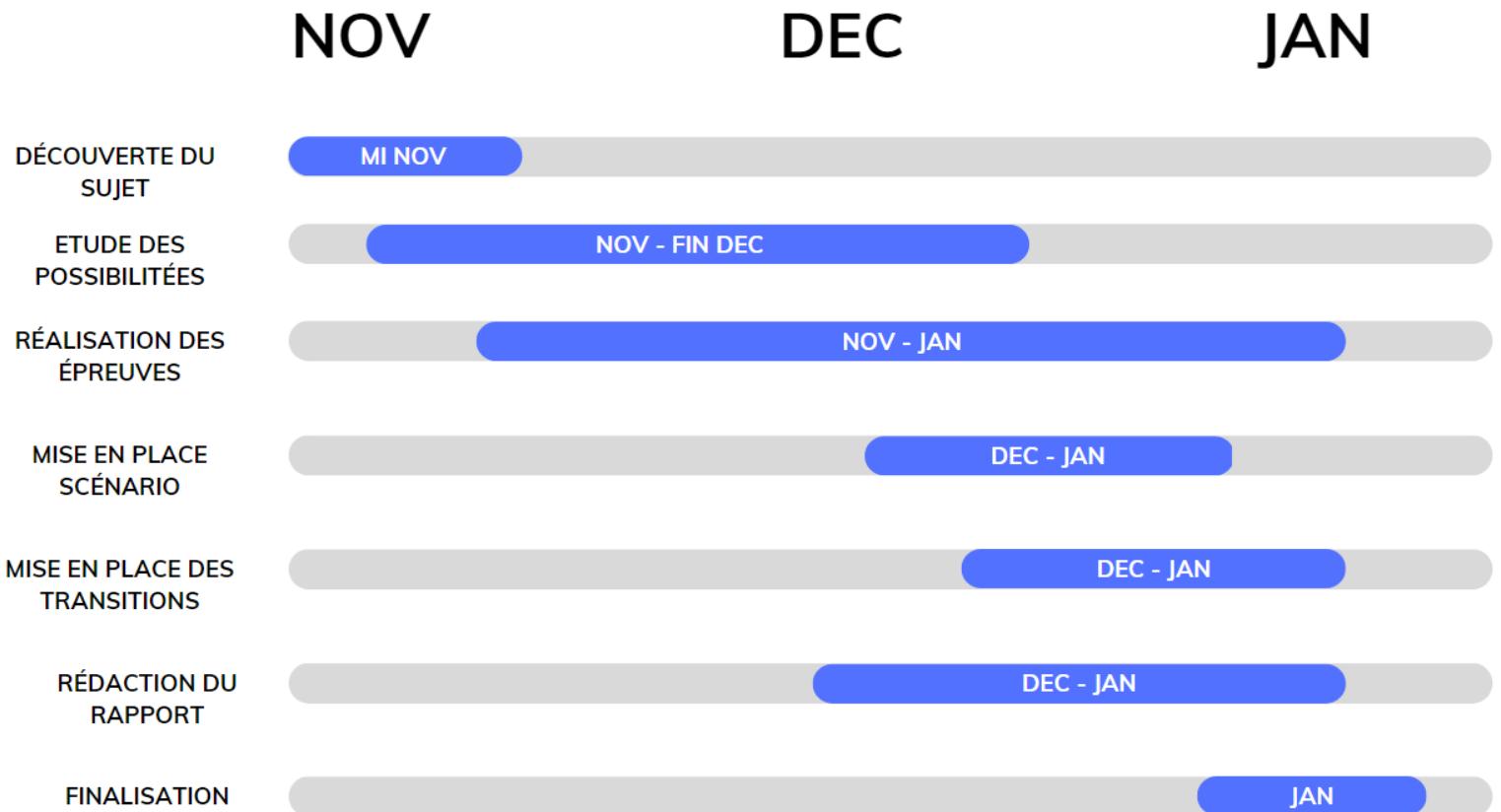
Sommaire :

<i>I - Gestion du projet</i>	4
<i>II - Présentation du scénario</i>	5
<i>III - Site Web PHP sur apache avec BDD</i>	6
<i>IV - Serveur de jeu et logs</i>	7
A - Comment fonctionne le programme	7
B - Comment consulter les logs ?	9
C - Comment accéder au programme et au logs sur le réseau Thorin ?	10
<i>V - Les différents systèmes de transmissions</i>	11
A - Première étape : Système de transmission IR (infra-rouge)	11
1) A quoi sert un capteur infra-rouge ?.....	11
2) Prérequis pour utiliser le capteur IR.....	11
3) Notre utilisation du capteur IR.....	13
B - Deuxième étape : Système de transmission 433 Mhz	15
1) Arduino avec modules émetteur/récepteur.....	15
2) Pièces requises.....	15
3) Installation de la bibliothèque RadioHead.....	15
4) Circuit émetteur.....	16
5) Code émetteur	17
6) Comment fonctionne l'esquisse de l'émetteur.....	18
7) Circuit récepteur.....	19
8) Comment fonctionne l'esquisse du récepteur.....	20

C - Troisième étape : Système de transmission Radio.....	23
1) Introduction à la radio avec raspberry.....	23
2) Prérequis.....	23
3) Notre utilisation du raspberry pour la radio.....	24
VI - eMail Escape Game.....	26
VII - Les problèmes rencontrés.....	27
VIII - Conclusion.....	28
IX - Code Source	29
1) Infrarouge.....	29
2) Récepteur 433 MHz.....	36
X - Annexes.....	45

I - Gestion du projet

DIAGRAMME GANTT



II - Présentation du scénario

L'équipe des challengers devra résoudre des énigmes afin de retrouver un objet précieux perdu depuis fort longtemps... La légende raconte que beaucoup de personnes ont essayé de le retrouver mais personne n'a jamais réussi à mettre la main dessus. Le propriétaire de cet objet a volontairement laissé des pistes pour que quelqu'un puisse faire cette découverte extraordinaire qui ferait une avancée fulgurante sur la technologie dans notre société. Les concurrents seront placés devant un ordinateur, sur son écran, un site où les participants devront s'identifier afin d'en savoir un peu plus... Une fois l'utilisateur et le mot de passe entré, les étapes pour lancer le script python (base du jeu) seront énumérées. Une fois le script lancé, la première énigme sera lancée et le jeu peut enfin commencer... Devant les participants, une télécommande avec une série de leds prêtes à être allumées. Le but est de retrouver un code couleur produit avec l'interaction de la télécommande qui donnera un drapeau. Une fois le drapeau entré, un bout de code s'affiche et nous pourrons enfin passer à la seconde étape. La 2ème étape est un blind test sur 3 musiques. Une fois les 3 musiques rentrées, des coordonnées s'affichent et ainsi qu'un autre bout de code. Les coordonnées devront être rentrées sur internet et afficher la localisation Singapour qui est la salle à côté de Vladivostok. Dans cette salle se trouvera la 3eme énigme qui est une charade avec le système FM radio. Un téléphone verrouillé avec une enceinte sera sur une table et les élèves devront déverrouiller avec un dessin le téléphone et devront se rendre dans l'application FM radio. Ils devront se mettre sur la bonne fréquence afin de capter la charade. Ils devront ensuite entrer la commande qui va lancer l'audio et ensuite devront résoudre la charade en question. La réponse de la charade donnera une adresse mail et un autre morceau du code. Les 3 morceaux du code forment le mot de passe de l'adresse mail. Un seul message sera sur cette boîte mail : le lieu où se trouve le trésor caché. Les élèves verront une boîte à ce lieu, une fois ouverte, émerveillés face à ce qu'ils voient devant eux, découvriront juste des twix.

III - Site Web PHP sur apache avec BDD

Pour le début de l'escape game, nous avons décidé d'utiliser plusieurs pages php pour tout d'abord expliquer les règles, le contexte de cette escape game. La site php sera accessible sur tous les pc de vladivostok car il sera relié au switch et donc présent dans le même réseau et hébergé sur un serveur apache personnel ainsi qu'un serveur mysql personnel. Ce site permet aussi notamment de mettre en place quelques énigmes bonus notamment trouvé dans le code source des informations, la possibilité de faire une injection sql aussi, même si aucun étudiants n'y a penser, il permet donc de faire de petites énigmes avant la vraie partie escape game qui utilise les trois systèmes de transmissions, ou pour donner des indices utiles dans les prochaines parties comme par exemple un code de téléphone ou même encore, pour amener les étudiants vers le programme python qui servira de serveur de jeux dans lequel on vous expliquera le fonctionnement par la suite.

Bien sûr, pour que cela marche correctement, j'ai dû mettre en place un serveur apache sur lequel est hébergé la page php ainsi qu'un serveur MySQL sur lequel est rentré l'username et le mdp avec laquelle les étudiants doivent commencer sur le site web à part s'il décide de faire une injection sql qui a été pensé de notre côté. De plus le site étant hébergé sur mon serveur apache alors il génère des minis logs avec la bd lorsque l'utilisateur clique sur une réponse bouton ou même lorsqu'il change de page, etc...

Vous aurez à la toute fin de ce compte-rendu, quelques screenshots de la page php et de ses différents points de vue. Puis dans un fichier zip tout le code si vous désirez reproduire ce système, il faudra juste l'héberger à un serveur apache et vous n'êtes pas obligé de le lier à une base de données pour qu'il fonctionne.

IV - Serveur de jeu et logs

Afin de superviser le déroulement de notre escape game, nous avons pensé à créer une programme python pour entrer les réponses des différentes épreuves pour cela, le joueur sera guidé jusqu'au programme python qu'il devra lancer dans un terminal. Une fois le programme lancé, le programme annonce au joueur qu'il devra réussir trois épreuves. Pour démarrer le jeu, le joueur doit appuyer sur entrée.

```
-----  
Debut de l'Escape Game, vous devez réussir 3 épreuves pour finir l'Escape Game,  
Bonne chance à vous !  
-----  
Appuyer sur entrée pour commencer
```

A) Comment fonctionne le programme ?

Pour chaque épreuve un petit algorithme est écrit.

Le programme contient également un système de vie afin de dynamiser l'escape game. Le joueur démarre la partie avec 5 vies et il perd une vie à chaque fois qu'il rentre une mauvaise réponse mais si il n'a plus de vie le programme continue quand même et on lui rajoute 5 vies pour éviter que le système de vie pénalise trop le joueur.

```
Entrez la réponse de l'enigme (La réponse est qu'un seul mot) : test  
Mauvaise réponse, Il vous reste plus que 2 vie(s)  
Entrez la réponse de l'enigme (La réponse est qu'un seul mot) : toto  
Mauvaise réponse, Il vous reste plus que 1 vie(s)  
Entrez la réponse de l'enigme (La réponse est qu'un seul mot) : reponse  
Mauvaise réponse, Il vous reste plus que 0 vie(s)  
Game Over  
5 vies vous ont été rajoutées, bonne chance  
Entrez la réponse de l'enigme (La réponse est qu'un seul mot) : █
```

Pour l'épreuve des LEDs RGB et l'épreuve de la charade, ce n'est qu'une boucle while qui a comme condition de sortie que la réponse demandée par un input soit bien 'france' ou 'interdire'.

```
● ● ●  
win=0  
vie=5  
while win!=1:  
    reponse=input("Entrez la réponse de l'enigme (La réponse est qu'un seul mot) : ")  
    if reponse=='France' or reponse=='france' or reponse=='FRANCE':  
        print("Bien joué tu as réussi la première épreuve")  
        win=1  
        logging.info("Le joueur a réussi l'enigme")  
    else:  
        logging.info("Le joueur a entre une mauvaise réponse à l'enigme et a perdu une vie")  
        vie-=1  
        print("Mauvaise réponse, Il vous reste plus que ",vie,"vie(s)")  
    if vie==0:  
        logging.info("Le joueur n'a plus de vie")  
        print("Game Over")  
        print("5 vies vous ont été rajoutées, bonne chance")  
        vie=5
```

```

● ● ●

win=0
vie=5
while win!=1:
    reponse=input("Entrez la reponse de la charade : ")
    if reponse=='Interdire' or reponse=='interdire' or reponse=='INTERDIRE':
        print("Bien joue tu as reussi la derniere epreuve")
        win=1
        logging.info("Le joueur a reussi la charade")

    else:
        logging.info("Le joueur a entre une mauvaise reponse a la charade et a perdu une vie")
        vie-=1
        print("Mauvaise reponse, Il vous reste plus que ",vie,"vie(s)")

    if vie==0:
        logging.info("Le joueur n'a plus de vie")
        print("Game Over")
        print("5 vies vous ont ete rajoute, bonne chance")
        vie=5

```

Le programme du blind test est un peu différent mais reste similaire aux autres. On effectue un test pour chaque réponse des 3 chansons. Sauf que le système de vie est plus important que pour les autres épreuves. Dans cette épreuve, il faut réussir à trouver les 3 titres des chansons pour réussir celle-ci avec 5 vies. Dans le cas contraire, le joueur devra recommencer le blind test depuis le début et récupérera 5 vies.

Lorsque le joueur a réussi le blind test, une adresse mail est affichée et on lui indique que le mot de passe est devant ces yeux. En effet, à chaque fois que le joueur réussissait une épreuve, une chaîne de caractère apparaissait.

```

Bravo vous avez reussi la deuxieme epreuve
0aW
Entrez quelque chose pour commencer la derniere epreuve

```

```

Bien joue tu as reussi la derniere epreuve
8=

```

```

Bien joue tu as reussi la premiere epreuve
cmF
Entrez quelque chose pour commencer la deuxieme epreuve

```

Les 3 chaînes de caractères réunies forment le mot de passe d'une adresse mail qui nous servira pour la suite de l'énigme...

B) Comment consulter les logs ?

Le serveur de jeu contient également la capacité de créer des logs dès que le joueur commencera la partie, rentrera une mauvaise ou bonne réponse, finira une épreuve...

```
2023-01-20 00:24:40 INFO Debut de l'escape game
2023-01-20 00:24:42 INFO Debut de la premiere epreuve : LED RGB
2023-01-20 00:24:44 INFO Le joueur a entre une mauvaise reponse a l'enigme et a perdu une vie
2023-01-20 00:24:45 INFO Le joueur a entre une mauvaise reponse a l'enigme et a perdu une vie
2023-01-20 00:24:47 INFO Le joueur a reussi l'enigme
2023-01-20 00:24:49 INFO Debut de la deuxieme epreuve : Blind Test
2023-01-20 00:24:53 INFO Le joueur a reussi la premiere question
2023-01-20 00:24:55 INFO Le joueur a entre une mauvaise reponse a la deuxieme question et a perdu une vie
2023-01-20 00:25:00 INFO Le joueur a reussi la deuxieme question et a reussi la premiere epreuve
2023-01-20 00:25:01 INFO Le joueur a entre une mauvaise reponse a la derniere question et a perdu une vie
2023-01-20 00:25:02 INFO Le joueur a entre une mauvaise reponse a la derniere question et a perdu une vie
2023-01-20 00:25:04 INFO Le joueur a entre une mauvaise reponse a la derniere question et a perdu une vie
2023-01-20 00:25:08 INFO Le joueur a reussi la derniere question
2023-01-20 00:25:10 INFO Debut de la derniere epreuve
2023-01-20 00:25:12 INFO Le joueur a entre une mauvaise reponse a la charade et a perdu une vie
2023-01-20 00:25:13 INFO Le joueur a entre une mauvaise reponse a la charade et a perdu une vie
2023-01-20 00:25:17 INFO Le joueur a reussi la charade
2023-01-20 00:25:17 INFO Fin de l'escape game
```

Tout ceci est crée grâce à un module python qui est :



```
import logging
```

Le module consiste à entrer des paramètres dans la commande suivante afin de paramétrier les logs :



```
logging.basicConfig(filename='log.txt', format='%(asctime)s %(levelname)-8s %(message)s', level=logging.INFO, datefmt='%Y-%m-%d %H:%M:%S')
```

on entre en paramètre le fichier dans lequel on veut que les logs soit écrit, le format dans lequel les logs doivent être écrites et si on veut que la date et l'heure apparaissent et dans quel format.

Après ça, on a juste à mettre des **logging.info** où l'on veut que les messages avec le message que le log doit contenir.

```
logging.info("Debut de l'escape game")
```

```
logging.info("Le joueur a reussi l'enigme")
```

```
logging.info("Debut de la premiere epreuve : LED RGB")
```

C) Comment accéder au programme et aux logs sur le réseau Thorin ?

Pour rajouter un peu de difficultés et de connaissances aux 1ères années, nous avons mis en place la solution de façon cachée. Durant cette escape game nous allons utiliser le répertoire thorin d'un ancien 1ère année à qui nous avons demandé l'autorisation car celui si est vide est donc plus propre pour l'escape game.

Tout d'abord nous avons créer un dossier sur thorin pour chaque pc disponible en salle vladivostok et donc chaque groupe d'élèves possibles comme le montre le screen suivant:

```
markassuza@thorin:~$ ls
PC-A  PC-B  PC-C  PC-D  PC-E  PC-F  PC-G  PC-H
markassuza@thorin:~$
```

Maintenant les élèves peuvent commencer l'escape game correspondant à leur PC. Une fois dans leur répertoire il vont sûrement réaliser la commande "ls" pour en voir le contenu mais il vont ne rien voir car le seul contenu qu'il y a est une image cachée car elle possède un "." devant son nom pour la voir nous devons réaliser un "ls -a"

```
markassuza@thorin:~/PC-A$ ls
markassuza@thorin:~/PC-A$ ls -a
.  ..  .debian.png
```

Nous voyons bien une image caché maintenant avec l'indice sur le site web php qui nous dit "N'oubliez pas de sortir couvert avec vitre veste zip" , ce que les élèves sont censé comprendre c'est de unzip l'image, ce qui n'est pas instinctif c'est vrai. Une fois la commande unzip .debian.png rentré nous allons voir que des fichiers sont sortis de l'image on peut y réaliser à nouveau un "ls -a" pour les voir car ils sont aussi cachés.

```
markassuza@thorin:~/PC-W$ unzip .debian.png
Archive:  .debian.png
warning [.debian.png]: 32752 extra bytes at beginning or within zipfile
(attempting to process anyway)
  inflating: .gameserv.py
  inflating: log.txt
markassuza@thorin:~/PC-W$ ls -a
.  ..  .debian.png  .gameserv.py  log.txt
```

Nous voyons alors un fichier python ainsi que un fichier txt apparaître le fichier python comporte le python expliqué juste au dessus qui pourront lancer à l'aide de la commande python3 nom_du_fichier.py et le fichier log.txt permet à nous de superviser l'escape game et de voir quand les élève se trompe pour ainsi consulter leur nombre de vie sur chaque PC.

V - Les différents systèmes de transmissions

A) Première étape : Système de transmission IR (infra-rouge)

L'un des systèmes de transmission de notre escape est un capteur infra-rouge comme celui de l'image ci dessous :



1) A quoi sert un capteur infra-rouge ?

Le capteur infrarouge est l'un des principaux outils de la surveillance volumétrique qui consiste à détecter un phénomène suspect dans un volume donné, pour être plus concret, il nous permet à l'aide d'une télécommande comme on peut voir ci-contre, de capter des signaux dont la longueur d'onde correspond aux infra-rouges, et peut être utiliser pour déclencher une action lors de la réception de ce signal, par exemple: capteur de température.

Dans ce projet nous avons dû utiliser ce dispositif pour transmettre de l'information avec un arduino.



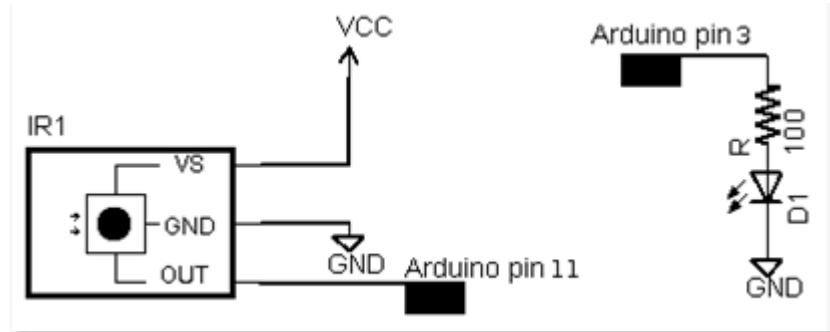
2) Prérequis pour utiliser le capteur IR

La première chose à faire avant d'utiliser le capteur IR, c'est de savoir comment le brancher avec un arduino, et de télécharger la bibliothèque qui permet d'émettre et de recevoir via l'infrarouge; elle permet de gérer les protocoles NEC, Sony, Philips RC5/RC6 et le traitement de données brutes. Nous allons voir comment la mettre en œuvre.

La librairie peut être téléchargée à l'adresse

<https://github.com/shirriff/Arduino-IRremote>

Pour tester l'utilisation de celle-ci, il faut utiliser un récepteur infrarouge 38 KHz, une résistance de 100 Ohms et une LED émettrice infrarouge. Le câblage sera le suivant sur l'Arduino :



Le code suivant nous est fourni :

```
#include <IRremote.h>

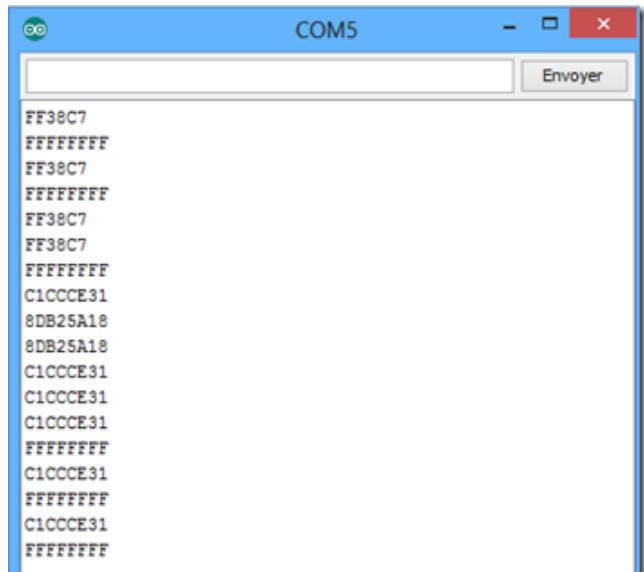
int broche_reception = 11; // broche 11 utilisée
IRrecv reception_ir(broche_reception); // crée une
decode_results decode_ir; // stockage données reçues

void setup()
{
    Serial.begin(9600);
    reception_ir.enableIRIn(); // démarre la réception
}

void loop()
{
    if (reception_ir.decode(&decode_ir))
    {
        Serial.println(decode_ir.value, HEX);
        reception_ir.resume(); // reçoit le prochain code
    }
}
```

nous intéresser (pour notre projet).

Il s'agit du modèle de réception IR, et également de la partie qui va le plus



Comme on peut le voir il n'y a pas encore d'interaction très visuelle, mise à part à la fin du code ou l'on envoie à la console les valeurs brute du signal qui ressemble à cela :

3) Notre utilisation du capteur IR

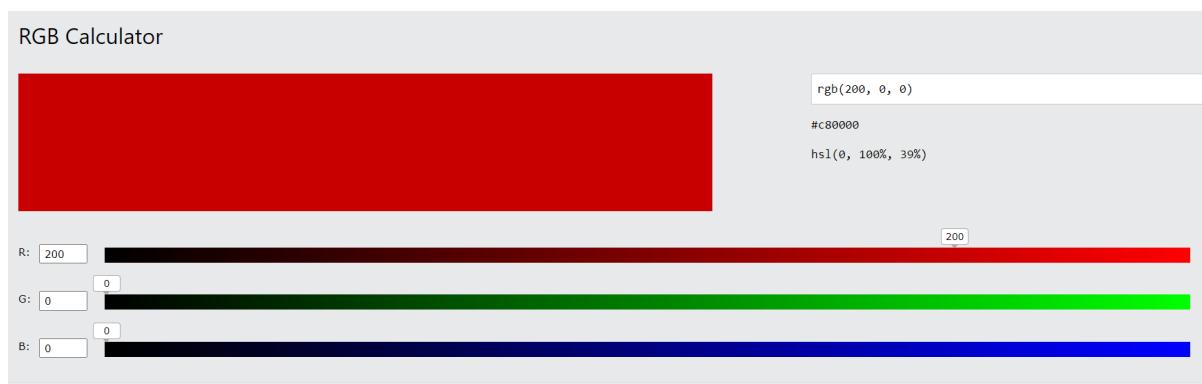
Nous allons maintenant voir ce que nous avons mis en place à l'aide des prérequis.

Tout d'abord nous voulions que lorsque que la télécommande communique avec le capteur, peu importe les valeurs du signal, que cela déclenche une action en rapport avec des LEDS. Notre idée fut donc de réussir à allumer des LEDs lors de la réception de données. Cependant on voulais un peu plus que juste allumer une LEDS, car il est difficile de créer un jeu ou épreuve juste avec juste une led qui s'allume, nous nous sommes alors dit que nous pourrions utiliser trois LEDs, rouge, vert et bleu qui représente les couleurs en RGB, et qui selon leurs clignotements, donnerais des informations sur la couleurs que l'on veut transmettre.

Exemple : en faisant clignoter la LED rouge 3 fois, et en disant à l'élève de multiplier le nombre de clignotement par un nombre choisi par nous même, on obtient une couleur avec le système RGB :

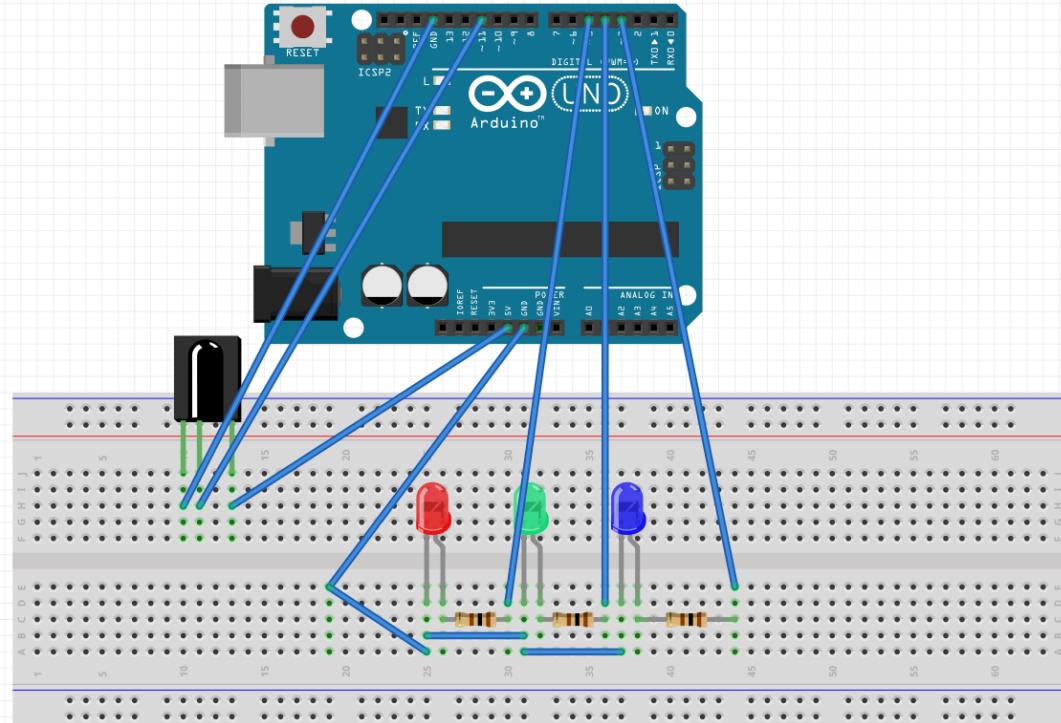


La LED rouge clignote 2 fois et la notice dit de de multiplier par 100, ce qui donne 200, on obtient sur un site :



Ainsi en enchaînant des suites de clignotement nous pouvons “transmettre des couleurs”, ce qui devient tout de suite plus intéressant.

Nous avons alors réaliser le branchement suivant en respectant les consigne précédentes :



Et ainsi à l'aide d'un enchaînement de code comme celui ci-contre, avec moins de delay et d'autre PIN, nous pouvons alors faire clignoter nos LEDs. Le code ci-contre associé à un PIN permet des méthodes “digitalWrite” qui permettent d'allumer ou éteindre la LED, le “delay” est du délai et permet le clignotement.

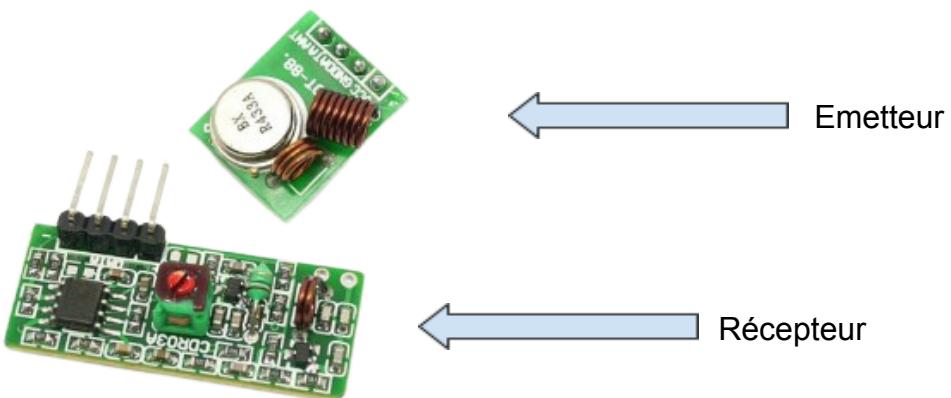
Il ne reste plus qu'à transmettre aux cobayes, les indices pour savoir par combien multiplier chaque clignotement pour que cela affiche la couleur que l'on veut.

```
void setup() {  
    pinMode(13, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(13, HIGH);  
    delay(1000);  
    digitalWrite(13, LOW);  
    delay(1000);  
}
```

A la fin de notre séquence de clignotements et avec le bon raisonnement en s'aidant de la notice, 3 couleurs sont ainsi deviner, le bleu, le blanc et le rouge, qui représente une drapeau, le nom du pays en question est le mot "code" qui sert à débloquer une partie du mot de passe et passer à l'étape suivante.

B) Deuxième étape : Système de transmission 433 Mhz

Pour la seconde partie concernant les systèmes de transmissions, nous nous sommes penchés vers le système de transmission émetteur/Récepteur.



1) Arduino avec modules émetteur/récepteur :

Le but de cette partie est d'envoyer un message d'un Arduino à une autre carte Arduino en utilisant 433 MHz. Une carte sera connectée à un émetteur 433 MHz et enverra n'importe quel message, l'autre carte Arduino sera connectée à un récepteur 433 MHz pour recevoir les messages.

2) Pièces requises :

Pour réaliser ce système, nous avons besoin des composants suivants :

- 2x Arduino
- Récepteur/émetteur RF 433 MHz
- Planche à pain
- Fils de liaison

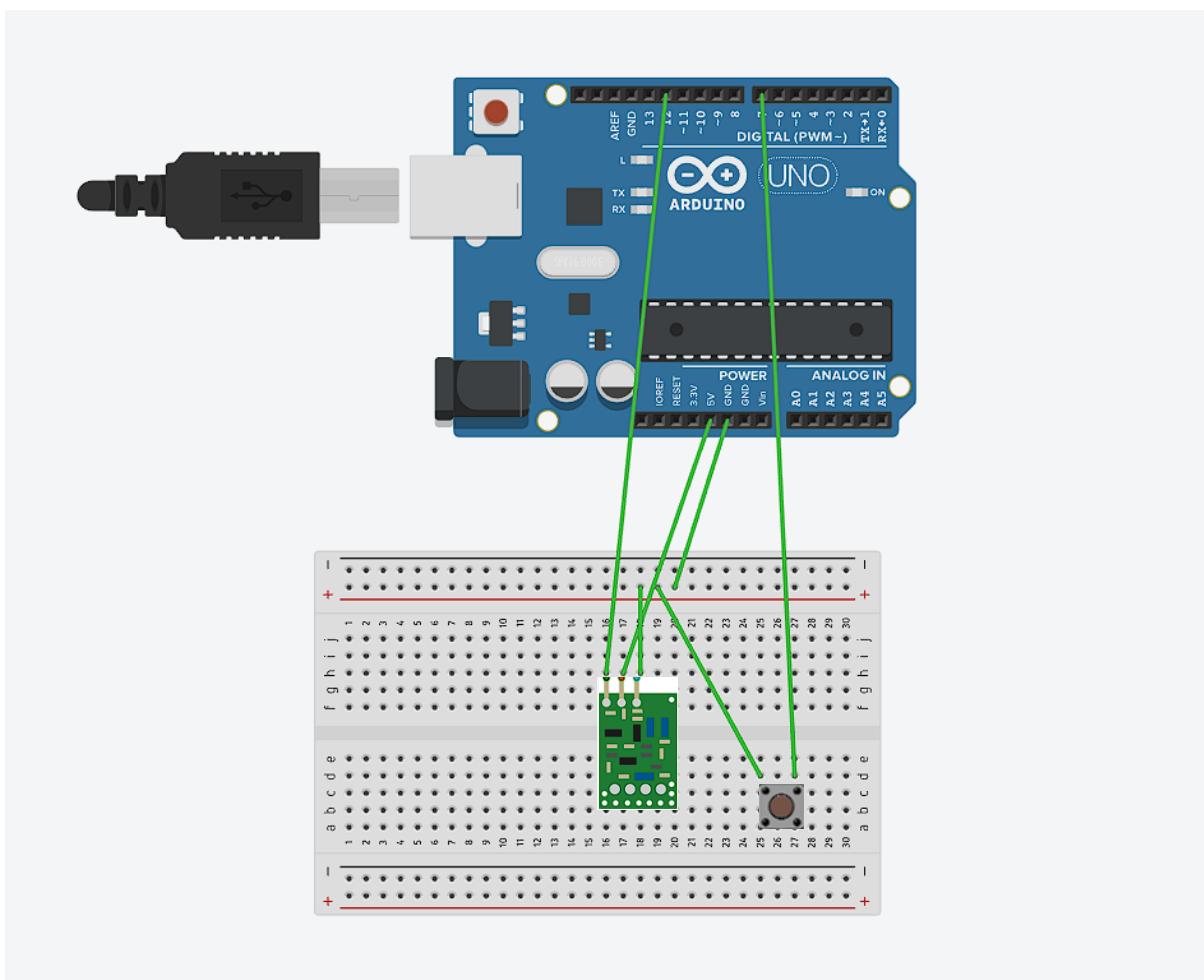
3) Installation de la bibliothèque RadioHead

La bibliothèque [RadioHead](#) offre un moyen simple de travailler avec l'émetteur/récepteur 433 MHz avec l'Arduino. Le moyen de l'avoir est de le télécharger en .zip, décompresser la bibliothèque RadioHead, déplacer le dossier de

la bibliothèque vers le dossier des bibliothèques d'installation de L'IDE Arduino et redémarrer l'IDE. La bibliothèque RadioHead est excellente et fonctionne avec presque tous les modules RF du marché.

4) Circuit émetteur

Câbler le module émetteur à l'Arduino en suivant le schéma suivant :



5) Code émetteur :

```
#include <RH_ASK.h>
#include <SPI.h> // Not actually used but needed to compile

RH_ASK driver;
const int BUTTON_PIN = 7;

// Variables will change:
int lastState = HIGH;
int currentState;

void setup() {
    Serial.begin(9600);    // Debugging only
    if (!driver.init())
        Serial.println("init failed");
    pinMode(BUTTON_PIN, INPUT_PULLUP);
}

void loop() {
    currentState = digitalRead(BUTTON_PIN);
    const char *msg = "Musique";
    if(currentState == LOW){
        Serial.println("OK");
        driver.send((uint8_t *)msg, strlen(msg));
        driver.waitPacketSent();
        delay(1000);
    }
}
```

6) Comment fonctionne l'esquisse de l'émetteur :

Tout d'abord, incluez la bibliothèque RadioHead ASK.

```
#include <RH_ASK.h>
```

Cette bibliothèque a besoin de la bibliothèque SPI pour fonctionner. Donc, vous devez également inclure la bibliothèque SPI.

```
#include <SPI.h>
```

Après cela, créez un objet RH_ASK appelé driver.

Dans setup(), initialisez l'objet RH_ASK en utilisant la méthode init().

```
Serial.begin(9600); // Debugging only
if (!driver.init())
    Serial.println("init failed");
```

Dans la boucle(), nous écrivons et envoyons notre message. Le message est enregistré dans la variable msg. Veuillez noter que **le message doit être de type char**.

```
const char *msg = "Musique";
```

Enfin, nous envoyons notre message comme suit :

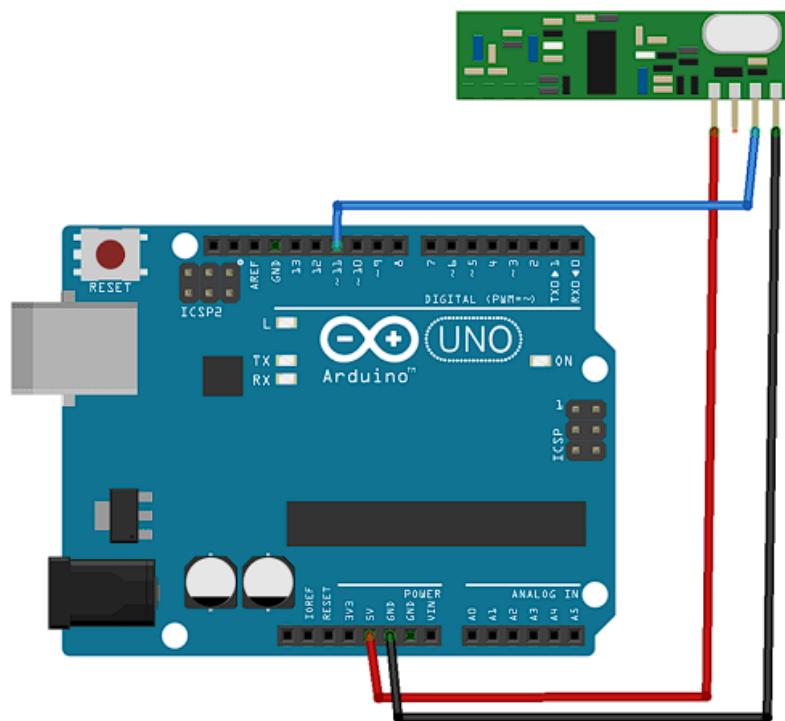
```
driver.send((uint8_t *)msg, strlen(msg));
driver.waitPacketSent();
```

Le message est envoyé toutes les secondes, mais vous pouvez régler ce délai.

```
delay(1000);
```

7) Circuit Récepteur :

Câbler le module récepteur à l'Arduino en suivant le schéma suivant :



8) Comment fonctionne l'esquisse du récepteur :

Le programme arduino du récepteur fait 360 lignes de code.

Il est séparé en différentes parties. Tout d'abord l'ajout des bibliothèques

Le récepteur utilise également les bibliothèques RadioHead ASK et SPI.

```
#include <RH_ASK.h>
#include <SPI.h>
```

On crée l'objet RH_ASK appelé driver



```
RH_ASK driver;
```

Ensuite, on définit les musiques. on définit en premier les notes de la musique. Voici un extrait de celle-ci.

```
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
```

```
int mario_melody[] = {
    NOTE_E7, NOTE_E7, 0, NOTE_E7,
    0, NOTE_C7, NOTE_E7, 0,
    NOTE_G7, 0, 0, 0,
    NOTE_G6, 0, 0, 0,

    NOTE_C7, 0, 0, NOTE_G6,
    0, 0, NOTE_E6, 0,
    0, NOTE_A6, 0, NOTE_B6,
    0, NOTE_AS6, NOTE_A6, 0,

    NOTE_G6, NOTE_E7, NOTE_G7,
    NOTE_A7, 0, NOTE_F7, NOTE_G7,
    0, NOTE_E7, 0, NOTE_C7,
    NOTE_D7, NOTE_B6, 0, 0,

    NOTE_C7, 0, 0, NOTE_G6,
    0, 0, NOTE_E6, 0,
    0, NOTE_A6, 0, NOTE_B6,
    0, NOTE_AS6, NOTE_A6, 0,

    NOTE_G6, NOTE_E7, NOTE_G7,
    NOTE_A7, 0, NOTE_F7, NOTE_G7,
    0, NOTE_E7, 0, NOTE_C7,
    NOTE_D7, NOTE_B6, 0, 0
};
```

Avec ces notes, on définit maintenant les mélodies des musiques.

Les mélodies sont uniquement une suite de notes

Les '0' correspondent à des moments vides.

Après cela, nous devons définir le tempo des mélodies. Le tempo des mélodies constitue le temps de chaque note.

```
int mario_tempo[] = {  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
  
    9, 9, 9,  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
  
    9, 9, 9,  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
    12, 12, 12, 12,  
};
```

Maintenant qu'on a fait cela, on passe à la fonction **void setup()**

```
void setup()  
{  
    Serial.begin(9600);  
    if (!driver.init())  
    {  
        Serial.println("init failed");  
    }  
  
    pinMode(3, OUTPUT); //buzzer  
    delay(1000);  
    Serial.println("start");  
}
```

- On ouvre le port série et fixe le débit de communications à 9600 bauds
- On initialise le driver
- Enfin on définit le pin où se trouve le buzzer sur l'arduino

On crée la fonction **void loop()**

La fonction void loop() sert à faire en sorte que quand le récepteur recevra un signal il affichera "Lancement" puis les chansons se lanceront (sing 1,2 et 3)

```
void loop()
{
    uint8_t buf[7];
    uint8_t buflen = sizeof(buf);
    if (driver.recv(buf, &buflen))
    {
        int i;
        Serial.print("Lancement ");
        Serial.println((char*)buf);
        {
            sing(1);
            sing(2);
            sing(3);
        }
    }
}
```

On crée la fonction **void sing()** qui contient les fonctions de lancement des chansons

```
int song = 0;
void sing(int s) {
    song = s;
}
if (song == 1) {
    Serial.println(" Musique 1 :");
    int size = sizeof(mario_melody) / sizeof(int);
    for (int thisNote = 0; thisNote < size; thisNote++) {
        int noteDuration = 1000 / mario_tempo[thisNote];
        buzz(melodyPin, mario_melody[thisNote], noteDuration);
        int pauseBetweenNotes = noteDuration * 1.30;
        delay(pauseBetweenNotes);
        buzz(melodyPin, 0, noteDuration);
    }
}
```

Enfin, on crée la fonction **void buzz()** qui configure le buzzer

```
void buzz(int targetPin, long frequency, long length) {
    digitalWrite(13, HIGH);
    long delayValue = 1000000 / frequency / 2;
    long numCycles = frequency * length / 1000;
    for (long i = 0; i < numCycles; i++) {
        digitalWrite(targetPin, HIGH);
        delayMicroseconds(delayValue);
        digitalWrite(targetPin, LOW);
        delayMicroseconds(delayValue);
    }
    digitalWrite(13, LOW);
}
```

C- Troisième étape Système de transmission Radio:

1) Introduction à la radio avec raspberry

La radio avec Raspberry Pi est un projet qui permet de construire un émetteur FM ou une station de radio en utilisant un ordinateur de la taille d'une carte de crédit. Le Raspberry Pi est un ordinateur à faible coût et à faible consommation d'énergie qui peut être utilisé pour diffuser de la musique, des émissions de radio ou des podcasts en utilisant un logiciel de diffusion en ligne. Ce projet est populaire auprès des amateurs de technologie et des passionnés de radio car il permet de créer une station de radio personnelle à un coût relativement faible.

2) Prérequis :

Pour utiliser un raspberry, il faut pouvoir l'alimenter et afficher ce qu'il renvoie avec un câble hdmi. Nous n'avons pas besoin de le configurer étant donné que ceux présents en classe le sont déjà, il ne reste que la partie pour utiliser la radio. Pour cela, il y a une bibliothèque à télécharger sur le raspberry, et au lien suivant :

<https://github.com/ChristopheJacquet/PiFmRds>

Cette librairie permet Pi-FM-RDS, qui dépend de la sndfile bibliothèque. Pour installer cette bibliothèque sur des distributions de type Debian, par exemple Raspbian, exécutez sudo apt-get install libsndfile1-dev.

Pi-FM-RDS dépend également du rpi-mailbox pilote Linux, vous avez donc besoin d'un noyau Linux récent. Les versions de Raspbian ont ceci à partir d'août 2015. Important. Les binaires compilés pour le Raspberry Pi 1 ne sont pas compatibles avec le Raspberry Pi 2/3, et inversement. Recompilez toujours lorsque vous changez de modèle, alors ne sautez pas la make clean étape dans les instructions ci-dessous !

Clonez le dépôt source et exécutez make-le dans le src répertoire :

```
git clone https://github.com/ChristopheJacquet/PiFmRds.git
cd PiFmRds/src
make clean
make
```

Si make signale une erreur, aucun pi_fm_rds fichier exécutable n'est généré (et vice versa). Toute erreur doit donc être corrigée avant de pouvoir passer aux étapes suivantes. make peut échouer si une bibliothèque requise est manquante (voir ci-dessus), ou il peut s'agir d'un bogue sur une distribution spécifique/plus récente.

Il faut également pouvoir écouter notre son. Pour cela nous avons un téléphone qui peut se brancher en prise jack et qui peut écouter des fréquences.

3) Notre utilisation du raspberry pour la radio

Pour ce système nous pouvions aller plus en profondeur mais nous nous sommes arrêtés à la diffusion d'une charade en continu surtout par manque de temps.

Après avoir câblés tout ce qu'on pouver sur le raspberry :

On arrive ainsi sur un écran dans lequel on peut lancer un terminal.



```

Fichier Édition Onglets Aide
pi@Boss: ~PiFmRds/src
pi@Boss: ~PiFmRds/src$ gcc -o pi_fm_rds pi_fm_rds.c
pi@Boss: ~PiFmRds/src$ ./pi_fm_rds -audio vache.wav -ps RT-FM -rt
RT: RT
Starting to transmit on 107.9 MHz.
**Terminating: cleanly deactivated the DMA engine and killed the carrier.
pi@Boss: ~PiFmRds/src$ ip link
1: lo: <loopback, no queueing discipline> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.0.0.1 scope host lo
        valid_lft forever preferred_lft forever
inet6 ::1/128 brd :: scope host

```

Dans le terminal après avoir télécharger la bibliothèque, on accède dans le dossier PiFmRds/src à une banque de son que l'on peut lancer avec l'une des deux commande ci-dessous



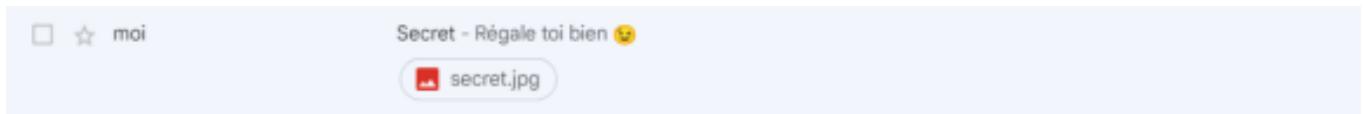
```
pi_fm_rds [-freq freq] [-audio file] [-ppm ppm_error] [-pi pi_code] [-ps ps_text] [-rt rt_text]
sudo ./pi_fm_rds -audio sound.wav
```

Une fois lancé, par défaut il faut aller sur le téléphone à la fréquence 107.9 et avec des écouteurs en prise on écoute le son.

Le son à écouter est une charade qui fait deviner le mot "interdire". Après avoir deviné, l'élève est en possession de tous les indices pour terminer cette escape game.

VI - eMail Escape Game

Dans cette partie, avec les résultats de nos systèmes de transmissions, des codes se sont affichés à des endroits et moments précis dans l'escape game. Une fois que les challengers ont réussi à retrouver et rassembler le code qui forme le mot de passe secret de l'adresse mail, ils ont juste à se connecter sur celle-ci, un seul mail apparaîtra :



Une image est disponible et quand nous cliquons dessus, ceci apparaît :



L'endroit où notre trésor est caché se situe sur la photo. C'est ainsi que se termine l'escape game, avec nos participants partants les poches pleines de twix !

VII - Les problèmes rencontrés

Au niveau Leds, RGB, nous sommes passés par plusieurs étapes de conceptions et de tests avant d'arriver à ce système de couleurs. En effet nous avons lors de nos premières tentatives, vu que selon le bouton appuyer avec la télécommande, les valeurs brutes des données ont des correspondances. En effet, si l'on appuie sur le bouton 1 de la télécommande, une certaine suite de valeurs s'affichera. Nous avons eu une idée qui nous semble plutôt bien et qui était de devoir rentrer un code précis avec la télécommande pour passer l'épreuve. Cependant nous avons rencontré plusieurs problèmes à ce sujet. Le premiers problème, fut un problème d'équipement. En effet les télécommandes disponibles en classe, n'étaient pas toutes très performantes ou ne fonctionnaient tout simplement pas. De ce fait, nous avons décidé d'abandonner.

Nous avons également eu des difficultés par rapport à l'écriture du programme, car en plus d'être long, il faut être rigoureux pour ne pas faire d'erreur.

Pour la partie émetteur/récepteur, nous avons eu un problème de mémoire de l'arduino car les musiques que nous mettions prenaient énormément d'espace. En effet, les notes, la tempo étaient assez lourdes pour le programme. Du coup, nous n'avons mis que trois musiques. Le second problème était que notre buzzer sortait des sons totalement non reconnaissables. Certaines musiques ne devaient pas être compatibles avec l'Arduino. Enfin, notre dernier problème a été un problème de shield. En effet, le seul moyen pour refaire marcher le programme et la connexion émetteur/récepteur a été d'enlever les shields des arduinos qui bloquaient la transmission d'informations.

Pour la partie Radio FM, le seul problème que nous avons eu était les grésillements et le son qui sortait qui était inaudible. Même avec une bonne fréquence, nous devions orienter la prise jack de manière précise pour que le son sorte de manière propre.

VIII - Conclusion

Pour conclure, ce projet a été très bénéfique pour nous. Nous avons pu découvrir et apprendre de nouveaux systèmes de transmission en les configurant à notre manière. Le fait de présenter notre projet en direct à d'autres élèves nous a permis d'avoir plus de recul sur le matériel que nous avons pratiqué. La compréhension des avantages et inconvénients de chaque système de transmission ainsi que les situations dans lesquelles ils sont le plus appropriés a apporté un grand plus dans l'amélioration de nos compétences dans ce domaine. La réalisation de cet escape game était l'une des plus grandes par rapport à tous nos projets que nous avons produits. Cela nous a donc permis de renforcer nos connaissances dans ce domaine.

IX - CODE Source :

1) Infrarouge

Le code ci-dessous ne sera pas réellement commenté, car en fait, il est très simple. En réalité, nous allumons juste et éteignons juste les leds sur les pins 3,4,5 correspondant à RGB (Red, Green, Blue) et en fonction du nombre de fois qu'elle s'allume les élèves ont une feuille devant eux leur indiquant un facteur X a multiplié à chacune des leds qui s'allument x fois avec bien évidemment un délai entre chaque changement d'état pour pouvoir apercevoir la led clignoter à une vitesse agréable pour que les 1ères années aient le temps de réfléchir.

Exemple:

Pour faire la couleur bleue, on fait clignoter 2 fois, la led rouge 4 fois la led verte et 3 fois la led bleu et les élèves doit multiplier par 2^5 ; 4^5 ; 3^60 = ce qui donne 10 ; 20 ; 180 et ceci en rgb donne la couleur Bleu

On fait de même pour la couleur blanche :

Ou on fait $8^10\pi$; 5^50 ; 10^25 . Ce qui donne 250 ; 250 ; 250 soient la couleur blanche en rgb

On termine par la couleur rouge :

Ou on fait 2^{125} ; 3^4 ; 2^{10}

Ce qui nous donne le code suivant :

```
/*
  Reception infrarouge
*/

#include <IRremote.h>

int broche_reception = 11; // broche 11 utilisée
IRrecv reception_ir(broche_reception); // crée une instance
decode_results decode_ir; // stockage données reçues

#define RECV_PIN 11

IRrecv irrecv(RECV_PIN);
decode_results results;

int a = 0xFF6897;

void setup()
{
  Serial.begin(9600);
  reception_ir.enableIRIn(); // démarre la réception
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
}

void loop()
{
//  if (reception_ir.decode(&decode_ir)){
//    Serial.println(decode_ir.value, HEX);
//    reception_ir.resume(); // reçoit le prochain code
//
//    digitalWrite(3,HIGH);
//    delay(250);
//    digitalWrite(3,LOW);
//  if ((decode_ir.value, HEX)== 0xFF6897) {
//    Serial.print("Vous avez appuyer sur 0");
if (irrecv.decode(&results)) {
  Serial.println(results.value, HEX);

  //Début

  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  delay(150);
  digitalWrite(3,LOW);
```

```

digitalWrite(4,LOW);
digitalWrite(5,LOW);
delay(150);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);
delay(150);
digitalWrite(3,LOW);
digitalWrite(4,LOW);
digitalWrite(5,LOW);
delay(150);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);
delay(150);
digitalWrite(3,LOW);
digitalWrite(4,LOW);
digitalWrite(5,LOW);
delay(1000);

//Première couleur (bleu)

digitalWrite(3,HIGH);
delay(250);
digitalWrite(3,LOW);
delay(250);
digitalWrite(3,HIGH);
delay(250);
digitalWrite(3,LOW);
delay(250);
delay(1000);

digitalWrite(4,HIGH);
delay(250);
digitalWrite(4,LOW);
delay(250);
delay(1000);

digitalWrite(5,HIGH);

```

```

delay(250);
digitalWrite(5,LOW);
delay(250);
digitalWrite(5,HIGH);
delay(250);
digitalWrite(5,LOW);
delay(250);
digitalWrite(5,HIGH);
delay(250);
digitalWrite(5,LOW);
delay(250);

delay(250);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);
delay(150);
digitalWrite(3,LOW);
digitalWrite(4,LOW);
digitalWrite(5,LOW);

//Deuxième couleur (blanc)

delay(1500);

digitalWrite(3,HIGH);
delay(250);
digitalWrite(3,LOW);
delay(250);
digitalWrite(3,HIGH);
delay(250);

```

```
digitalWrite(3,LOW);
delay(250);
digitalWrite(3,HIGH);
delay(250);
digitalWrite(3,LOW);
delay(250);
delay(1000);

digitalWrite(4,HIGH);
delay(250);
digitalWrite(4,LOW);
delay(250);
delay(1000);

digitalWrite(5,HIGH);
delay(250);
digitalWrite(5,LOW);
delay(250);
digitalWrite(5,HIGH);
```

```

delay(250);
digitalWrite(5,LOW);
delay(250);
digitalWrite(5,HIGH);
delay(250);
digitalWrite(5,LOW);
delay(250);
digitalWrite(5,HIGH);
delay(250);
digitalWrite(5,LOW);
delay(250);
digitalWrite(5,HIGH);
delay(250);
digitalWrite(5,LOW);
delay(250);
digitalWrite(5,HIGH);
delay(250);
digitalWrite(5,LOW);
delay(250);

delay(250);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);
delay(150);
digitalWrite(3,LOW);
digitalWrite(4,LOW);
digitalWrite(5,LOW);

//Troisième couleur (rouge)

delay(2000);

digitalWrite(3,HIGH);
delay(250);
digitalWrite(3,LOW);
delay(250);
digitalWrite(3,HIGH);
delay(250);
digitalWrite(3,LOW);
delay(250);
delay(1000);

digitalWrite(4,HIGH);
delay(250);
digitalWrite(4,LOW);
delay(250);
digitalWrite(4,HIGH);
delay(250);
digitalWrite(4,LOW);
delay(250);

```

```

    digitalWrite(4,HIGH) ;
    delay(250) ;
    digitalWrite(4,LOW) ;
    delay(250) ;
    delay(1000) ;

    digitalWrite(5,HIGH) ;
    delay(250) ;
    digitalWrite(5,LOW) ;
    delay(250) ;
    digitalWrite(5,HIGH) ;
    delay(250) ;
    digitalWrite(5,LOW) ;
    delay(250) ;
    delay(1500) ;

//FIN

digitalWrite(3,HIGH) ;
digitalWrite(4,HIGH) ;
digitalWrite(5,HIGH) ;
delay(150) ;
digitalWrite(3,LOW) ;
digitalWrite(4,LOW) ;
digitalWrite(5,LOW) ;
delay(150) ;
digitalWrite(3,HIGH) ;
digitalWrite(4,HIGH) ;
digitalWrite(5,HIGH) ;
delay(150) ;
digitalWrite(3,LOW) ;
digitalWrite(4,LOW) ;
digitalWrite(5,LOW) ;
delay(150) ;
digitalWrite(3,HIGH) ;
digitalWrite(4,HIGH) ;
digitalWrite(5,HIGH) ;
delay(150) ;
digitalWrite(3,LOW) ;
digitalWrite(4,LOW) ;
digitalWrite(5,LOW) ;

if (results.value == 0xFF6897) Serial.println ("ok") ;
irrecv.resume() ; // Receive the next value

}

}

```

2) RECEPTEUR 433MHz :

```
#include <RH_ASK.h>
#include <SPI.h>

RH_ASK driver;

const int songspeed = 1.5;
const int buzzer = 3;

#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
```

```
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
```

```

#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
#define REST 0

#define melodyPin 3
//Mario melody
int mario_melody[] = {
    NOTE_E7, NOTE_E7, 0, NOTE_E7,
    0, NOTE_C7, NOTE_E7, 0,
    NOTE_G7, 0, 0, 0,
    NOTE_G6, 0, 0, 0,

    NOTE_C7, 0, 0, NOTE_G6,
    0, 0, NOTE_E6, 0,
    0, NOTE_A6, 0, NOTE_B6,
    0, NOTE_AS6, NOTE_A6, 0,

    NOTE_G6, NOTE_E7, NOTE_G7,
    NOTE_A7, 0, NOTE_F7, NOTE_G7,
    0, NOTE_E7, 0, NOTE_C7,
    NOTE_D7, NOTE_B6, 0, 0,

    NOTE_C7, 0, 0, NOTE_G6,
    0, 0, NOTE_E6, 0,
    0, NOTE_A6, 0, NOTE_B6,
    0, NOTE_AS6, NOTE_A6, 0,

    NOTE_G6, NOTE_E7, NOTE_G7,
    NOTE_A7, 0, NOTE_F7, NOTE_G7,
    0, NOTE_E7, 0, NOTE_C7,

```

```

    NOTE_D7, NOTE_B6, 0, 0
};

//Mario tempo
int mario_tempo[] = {
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    9, 9, 9,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    9, 9, 9,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,
};

//Take On Me melody
int TOM_melody[] = {
    NOTE_FS5, NOTE_FS5, NOTE_D5, NOTE_B4, NOTE_B4, NOTE_E5,
    NOTE_E5, NOTE_E5, NOTE_GS5, NOTE_GS5, NOTE_A5, NOTE_B5,
    NOTE_A5, NOTE_A5, NOTE_A5, NOTE_E5, NOTE_D5, NOTE_FS5,
    NOTE_FS5, NOTE_FS5, NOTE_E5, NOTE_E5, NOTE_FS5, NOTE_E5
};

//Take on me tempo
int TOM_tempo[] = {
    12, 12, 12, 8, 8, 8,
    8, 10, 12, 12, 12, 12,
}

```

```

12, 12, 12, 8, 8, 8,
8, 10, 12, 12, 12, 12
};

#define NOTE_C4 262
#define NOTE_D4 294
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_G4 392
#define NOTE_A4 440
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_D5 587
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_G5 784
#define NOTE_A5 880
#define NOTE_B5 988

//Harry Potter melody
int h_melody[] = {
    0, NOTE_D4,
    NOTE_G4, NOTE_AS4, NOTE_A4,
    NOTE_G4, NOTE_D5,
    NOTE_C5,
    NOTE_A4,
    NOTE_G4, NOTE_AS4, NOTE_A4,
    NOTE_F4, NOTE_GS4,
    NOTE_D4,
    NOTE_D4,

    NOTE_G4, NOTE_AS4, NOTE_A4,
    NOTE_G4, NOTE_D5,
    NOTE_F5, NOTE_E5,
    NOTE_DS5, NOTE_B4,
    NOTE_DS5, NOTE_D5, NOTE_CS5,
    NOTE_CS4, NOTE_B4,
    NOTE_G4,
    NOTE_AS4,

    NOTE_D5, NOTE_AS4,
    NOTE_D5, NOTE_AS4,
    NOTE_DS5, NOTE_D5,

```

```

NOTE_CS5, NOTE_A4,
NOTE_AS4, NOTE_D5, NOTE_CS5,
NOTE_CS4, NOTE_D4,
NOTE_D5,
0, NOTE_AS4,

NOTE_D5, NOTE_AS4,
NOTE_D5, NOTE_AS4,
NOTE_F5, NOTE_E5,
NOTE_DS5, NOTE_B4,
NOTE_DS5, NOTE_D5, NOTE_CS5,
NOTE_CS4, NOTE_AS4,
NOTE_G4
};

//Harry Potter tempo
int h_tempo[] = {
    2, 4,
    4, 8, 4,
    2, 4,
    2,
    2,
    4, 8, 4,
    2, 4,
    1,
    4,

    4, 8, 4,
    2, 4,
    2, 4,
    2, 4,
    4, 8, 4,
    2, 4,
    1,
    4,

    2, 4,
    2, 4,
    2, 4,
    2, 4,
    4, 8, 4,
    2, 4,
    1,
}

```

```

4, 4,
2, 4,
2, 4,
2, 4,
2, 4,
4, 8, 4,
2, 4,
1
};

void setup()
{
    Serial.begin(9600);
    if (!driver.init())
    {
        Serial.println("init failed");
    }

    pinMode(3, OUTPUT); //buzzer
    pinMode(4, OUTPUT);
    delay(1000);
    Serial.println("start");

}

void loop()
{
    uint8_t buf[7];
    uint8_t buflen = sizeof(buf);
    if (driver.recv(buf, &buflen))
    {
        int i;
        // Message lorsque le recepteur capte un signal
        Serial.print("Lancement ");
        Serial.println((char*)buf);
        {
            //musique à chanter
            sing(1);
            sing(2);
            sing(4);
        }
    }
}

```

```

        }

    }

int song = 0;

void sing(int s) {
    song = s;
    if (song == 2) {
        Serial.println(" Musique 2 :");
        int size = sizeof(TOM_melody) / sizeof(int);
        for (int thisNote = 0; thisNote < size; thisNote++) {
            int noteDuration = 1000 / TOM_tempo[thisNote];
            buzz(melodyPin, TOM_melody[thisNote], noteDuration);
            int pauseBetweenNotes = noteDuration * 1.30;
            delay(pauseBetweenNotes);
            buzz(melodyPin, 0, noteDuration);
        }
    }
    if (song == 1) {

        Serial.println(" Musique 1 :");
        int size = sizeof(mario_melody) / sizeof(int);
        for (int thisNote = 0; thisNote < size; thisNote++) {
            int noteDuration = 1000 / mario_tempo[thisNote];

            buzz(melodyPin, mario_melody[thisNote], noteDuration);
            int pauseBetweenNotes = noteDuration * 1.30;
            delay(pauseBetweenNotes);
            buzz(melodyPin, 0, noteDuration);
        }
    }
    if (song == 3) {
    }

    if (song == 4) {
        int size = sizeof(h_tempo) / sizeof(int);

        Serial.println(" Musique 3 : ");
        for (int note = 0; note < size; note++) {
            int duration = 1000 / h_tempo[note];
            tone(3, h_melody[note], duration);
            int pauseBetweenNotes = duration * 1.30;
    }
}

```

```
    delay(pauseBetweenNotes);
    noTone(3);
}
}
}
}

void buzz(int targetPin, long frequency, long length) {
    digitalWrite(13, HIGH);
    long delayValue = 1000000 / frequency / 2;
    long numCycles = frequency * length / 1000;
    for (long i = 0; i < numCycles; i++) {
        digitalWrite(targetPin, HIGH);
        delayMicroseconds(delayValue);
        digitalWrite(targetPin, LOW);
        delayMicroseconds(delayValue);
    }
    digitalWrite(13, LOW);
}
```

X - Annexes

À savoir que les noms des pages ne veulent rien dire pour que les étudiants ne puissent pas prédire le nom des prochaines pages avec un peu de logiques. Il y a seulement les images des pages les plus importantes sinon on en finit plus. Évidemment que certaines informations sont cachées dans le code source de la page.

index.php

The screenshot shows a simple login interface titled "LOGIN". It contains two input fields labeled "User Name" and a single "Password" field. Below the password field is a "Login" button.

Q1-v2.php

The screenshot displays a page titled "Explication des règles". It includes a yellow disclaimer: "Il est formellement interdit d'utiliser les commandes pour afficher le contenu d'un fichier nano, cat, vim". Below it is another yellow message: "Nous voyons toute la moindre suspicion de triches et vous avez perdu ! Si vous êtes bloqués pendant trop longtemps appellés nous on vous fournira un indice". At the bottom are two buttons: "J'accepte" and "Je refuse".

Q16.php

The screenshot shows the first step of a game server setup titled "Etape 1: Serveur de Jeu". It contains several pieces of text: "Pour commencer vous allez lancer le serveur de jeu qui est un programme python", "Que vous devez exécuter sans en lire le contenu avec nano, cat, vim ...", "Il se trouve sur le thorin de "markassuza"" (highlighted in yellow), "Le mot de passe est super sécurisé", "Choisissez ensuite le répertoire attribué à votre PC puis fouille bien le contenu", "Mais surtout n'oubliez pas de sortir couvert avec une veste zip", "En ce moment il fait froid non ?", "J'ai trouvé le programme python", and "De quoi tu me parles encore ?".

requin.php

The screenshot shows the first step of a game server setup titled "Etape 1: Serveur de Jeu". It includes a yellow warning: "Lance ce programme et c'est ici que tu vas rentrer toutes les réponses". Below it is a yellow note: "Rappel pour exécuter : python3 nom_du_prog.py". A yellow warning at the bottom states: "Attention tu as un nombre de vies limité". At the bottom are two buttons: "Première étape" and "Retour".

led.php

Trouve le montage suivant

J'ai trouvé le montage

Je trouve pas

indice1.php

Lance le avec la télécommande et bonne chance

Utilise la feuille devant toi !

Le résultats de cette énigme forme le drapeau de ...

N'oublie pas de rentrer les résultats dans le python exécutable

Oui Chef !

Non Chef !



Tu en as finis avec ce site cherche par toi-même maintenant

Et suit le programme python

Lien vidéo 433MHz :

<https://youtube.com/shorts/0MuFq-VJynw>

Lien vidéo Radio FM:

<https://www.youtube.com/watch?v=DHnpaGvTETw>

Lien vidéo Infrarouge:

<https://youtu.be/78LOLV6yJzc>