

COMPTE RENDU

SERRURE CONNECTÉE

TABLE DES MATIÈRES

I - Cahier des charges.....	4
2 - Objectifs principaux.....	5
3 - Objectifs secondaires.....	6
II - Gestion de Projet.....	7
1 - Planning.....	7
2 - Répartition du travail.....	8
3 - Difficultés rencontrées.....	10
III - Réalisation du Projet.....	11
1 - Liste des composants.....	11
2 - Schéma.....	13
IV - Conception du projet.....	15
1 - Programme capteur d'empreinte.....	15
2 - PROGRAMME esp32(WIFI).....	18
3 - PROGRAMME servomoteur.....	20
4 - PROGRAMME esp32(blueetooth).....	20
5 - Application bluetooth.....	20

INTRODUCTION

Dans le monde d'aujourd'hui, l'ère du connectée notamment l'objet connecté donne un nouvel élan à notre société numérique telle que nous la connaissons, tous ces objets connectés utilisent des technologies qui permettent de se connecter sans fil et à distance.

Les communications sans fil sont donc un élément important dans le domaine des objets connectés, nous avons eu l'opportunité de réaliser un projet dans ce domaine.

Le projet est une serrure connectée, s'ouvrant par empreinte digitale, il est possible de la déverrouiller aussi de son smartphone grâce à une application.



Figure 1: Serrure Connectée

I – CAHIER DES CHARGES

1 – PRÉSENTATION DU PROJET

Notre projet est un verrou électrique de porte à empreinte digitale s'ouvrant par capteur d'empreinte digitale.

La serrure connectée sera capable de communiquer par mail ou par message pour l'ouverture, de plus il est possible de déverrouiller la serrure directement par smartphone.

Il sera possible de communiquer avec la serrure par plusieurs moyens de communication sans fils notamment par Lora, wifi ou Bluetooth.

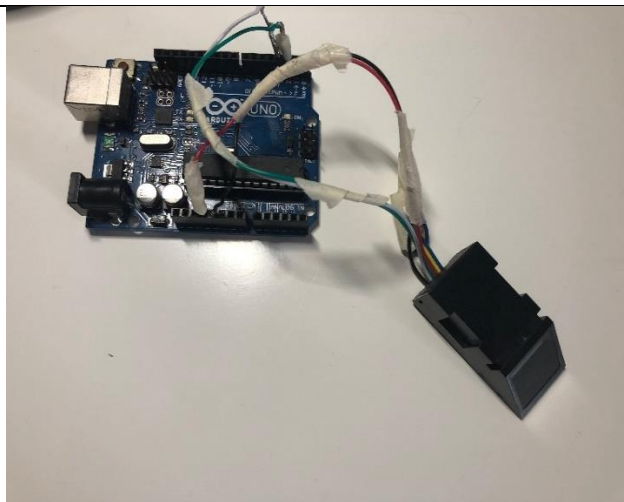


Figure 2: capteur d'empreinte

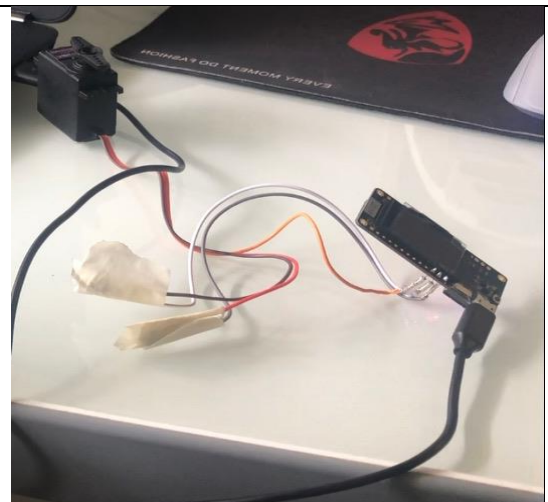


Figure 3: ESP32

2 - OBJECTIFS PRINCIPAUX

Les objectifs principaux ont été adopter en fonction des priorités du projet et des objectifs donnés par les professeurs (réaliser des systèmes complexes et communicants).

Le principal objectif étant de réussir à communiquer sans fils, le but était d'alerter si la serrure s'ouvrait en envoyant par exemple un message d'alerte.

Ensuite un objectif qui est relié au premier, utiliser le module wifi pour communiquer avec son téléphone à notre projet. Pour cela le module wifi sera l'ESP32, qui permettra la connexion Wifi.

Et pour terminer, il fallait aussi impérativement faire marche le capteur d'empreinte digital, le système de verrouillage marche grâce au capteur, il est donc essentiel pour le projet de réussir à le programmé correctement.

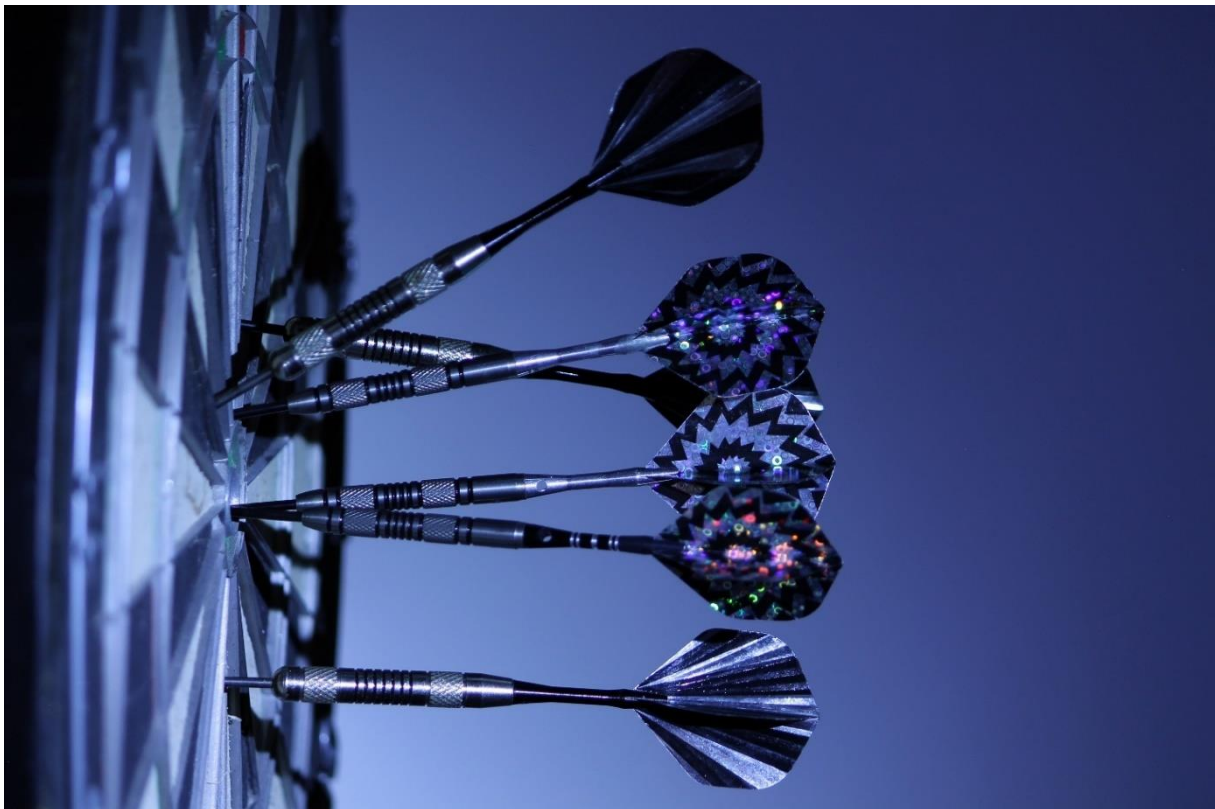


Figure 4: Objectifs Principaux

3 - OBJECTIFS SECONDAIRES

Contrairement aux objectifs principaux, les objectifs secondaires ont été choisis en fonction du projet et aux choix des composants car les capteurs peuvent avoir des options supplémentaires qui permettront d'enrichir le projet.

Premièrement, L'ESP32 a plusieurs communications sans fil autre que le Wifi, alors pour le projet, il est envisageable d'utiliser le Bluetooth pour déverrouiller la serrure à partir de son téléphone via une application.

Dans un second temps, en fonction de l'avancement, de faire une maquette qui illustre parfaitement le projet. La maquette sera imprimée en 3d via les machines du Fablab.

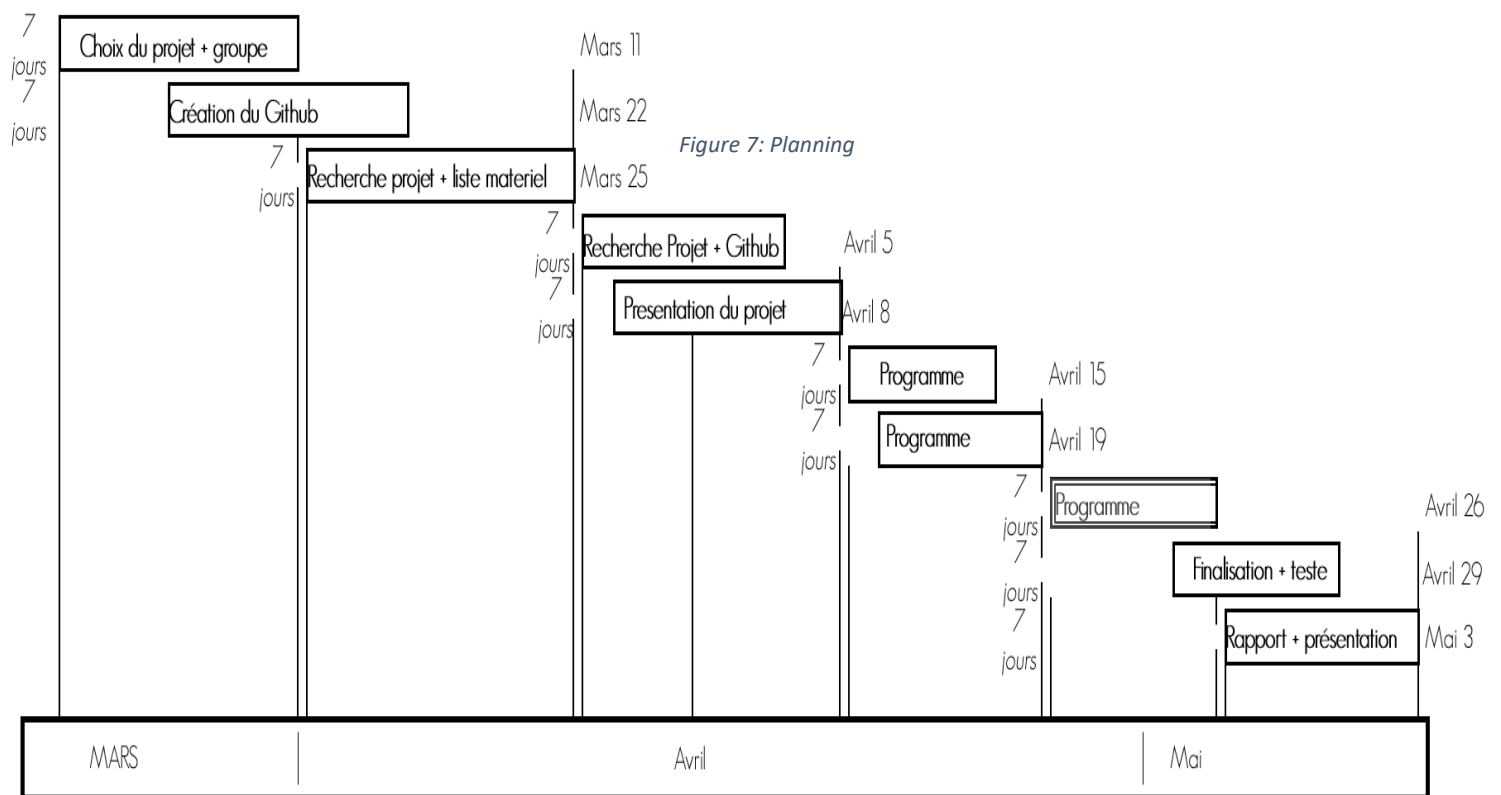
Et pour finir, l'objectif secondaire le moins important, utiliser la technologie NFC pour déverrouiller la serrure ou utiliser le capteur d'empreinte digitale d'un smartphone pour ouvrir ou fermer le verrou.



Figure 5: Objectifs secondaires

II – GESTION DE PROJET

1 - PLANNING



2 - RÉPARTITION DU TRAVAIL

Le groupe du projet est composé de deux personnes, la répartition des tâches a été facilement réalisée. Les tâches se sont partagées en fonction des qualités et goûts des étudiants.

Personnellement (Ayoub BENALLAL), ma tâche a été de programmer le capteur d'empreinte digitale. Il a fallu une longue recherche sur le capteur car il y a peu de documentation sur le capteur que ce soit en français ou en anglais.

Il a fallu ensuite le programme sur Arduino, il y a eu aussi beaucoup de problèmes que l'on énoncera dans les difficultés rencontrées.

Je me suis aussi occupé de la partie communication sans fil pour l'ESP32, particulièrement la partie Wifi, pour le projet, il est mieux de d'utiliser le Wifi pour envoyer un message et notamment discord, la partie n'a pas été aussi complexe que pour le capteur d'empreinte digital.



Figure 8: Répartition du Travail

Ensuite, comme verrou, l'utilisation d'un servomoteur est plus adaptée car il est facilement programmable sur Arduino, Il faut juste le faire tourner d'un angle dès que le capteur d'empreinte reconnaitra la bonne empreinte.

Et pour terminer, après avoir programmer séparément tous les capteurs, il a fallu les réunir en un seul programme, la difficulté était de synchroniser tous les capteurs afin que tous soient actifs au bon moment, par exemple le servo doit être tourné au bon moment ou l'alerte au moment de l'ouverture de la serrure.



Figure 9: Répartition du travail

3 - DIFFICULTÉS RENCONTRÉES

Les difficultés ont été nombreuses pour ce projet, tout d'abord le problème majeur a été que malheureusement le capteur n'est pas compatible avec l'esp32, il a donc fallu séparer le projet en deux parties.

La carte Arduino Uno est donc connectée au capteur d'empreinte et à un servo moteur qui joue le rôle de verrou

Pour continuer sur le capteur d'empreinte digitale, le principal problème vient au niveau du programme car il est très complexe de coder ce capteur avec peu de ressources numériques

Ensuite, il a fallu réunir tous les programmes en un seul, l'un des problèmes est que quand le capteur et le servomoteur était branché sur la carte Arduino en même temps, le moteur ne tournait pas sûrement à cause d'une puissance trop importante pour la carte.

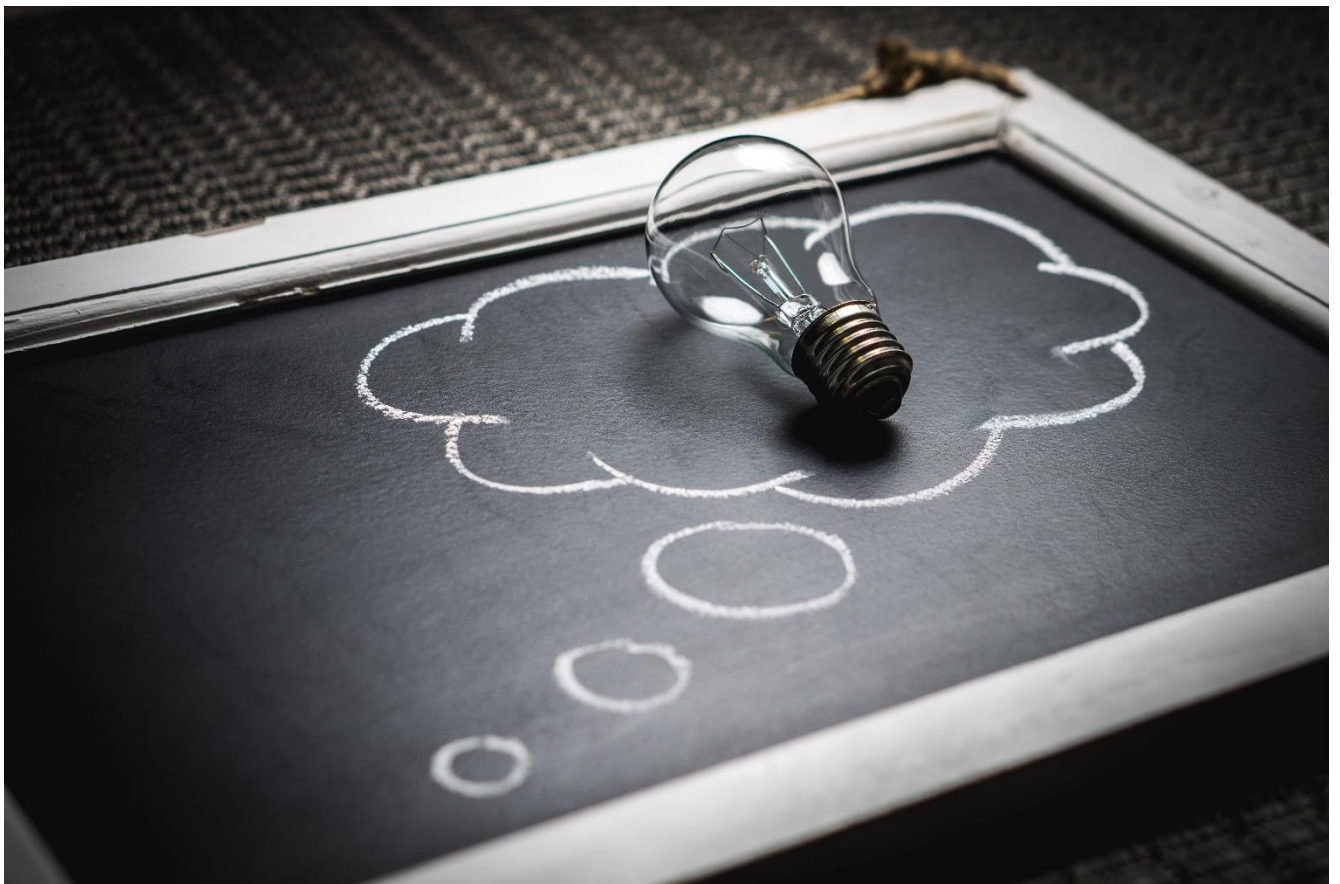


Figure 10: Difficultés rencontrées

III - RÉALISATION DU PROJET

1 - LISTE DES COMPOSANTS



ESP32 TTGO

L'ESP32 est une puce combinée Wi-Fi et Bluetooth unique à 2,4 GHz conçue avec le TSMC ultra-basse consommation.

C'est une carte électronique équipée d'un microcontrôleur, l'esp32 peut très bien jouer le rôle d'une carte Arduino.

Ce microcontrôleur dispose d'interfaces Wifi, Bluetooth et Lora.



Carte Arduino Uno

C'est une carte électronique équipée d'un microcontrôleur. Le microcontrôleur permet à partir d'e

	<h3>Module d'identification du lecteur d'empreintes digitales</h3> <p>Permet la collecte des empreintes digitales, l'enregistrement des empreintes digitales, la comparaison des empreintes digitales et la recherche d'empreintes digitales.</p>
	<h3>Servo-moteur</h3> <p>Un servomoteur est un système qui a pour but de produire un mouvement précis en réponse à une commande externe.</p>
	<h3>Pile 9v</h3> <p>Permet d'alimenter la carte Arduino.</p>

2 - SCHÉMA

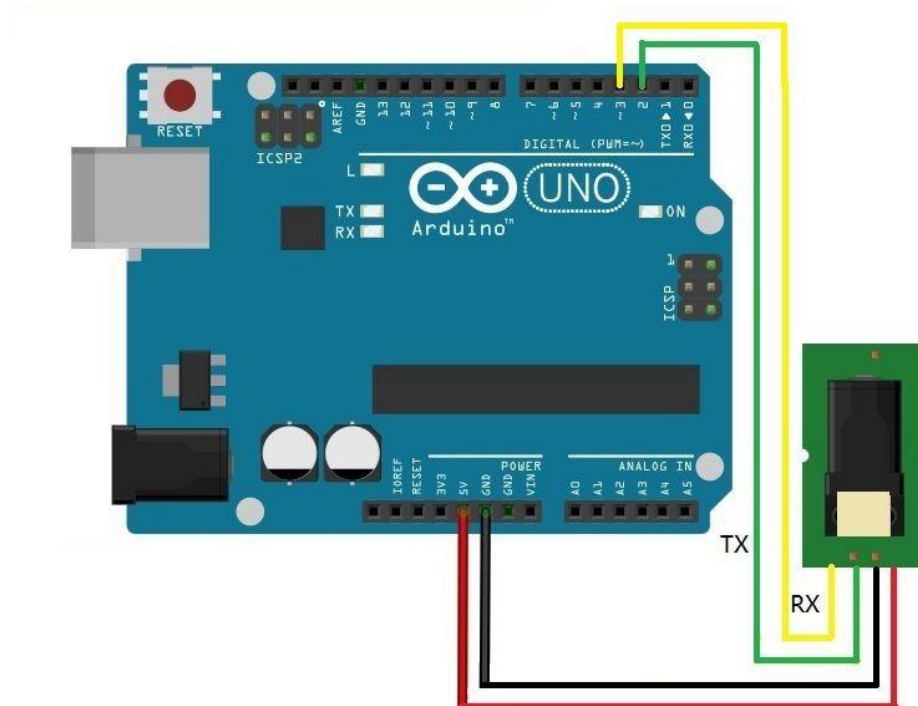


Figure 11: Schéma Lecteur d'empreinte

Fil vert sur le numérique 2

Fil blanc sur le numérique 3

Fil rouge 5v

Fil noir GND

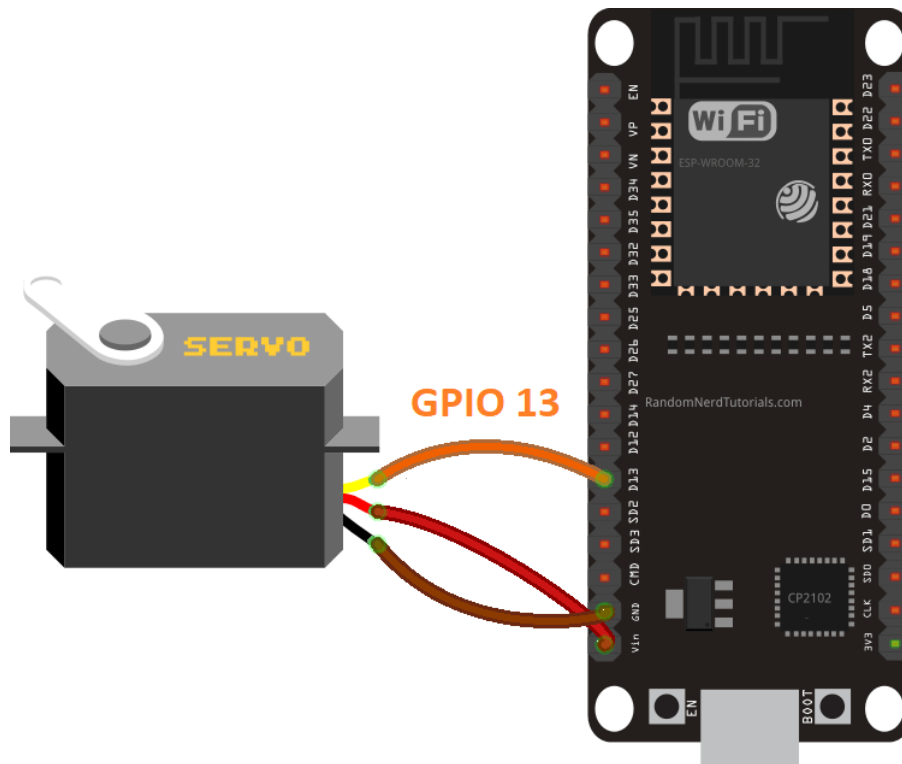


Figure 12: Schéma ESP32

Fil orange sur le numérique 13

FIL rouge 5v

Fil noir GND

IV - CONCEPTION DU PROJET

1 - PROGRAMME CAPTEUR D'EMPREINTE

Avant de programmer sur Arduino, le capteur d'empreinte digitale permet d'enregistrer des empreintes, un logiciel permet d'enregistrer des empreintes, il se prénomme « SFG Demo ». Ce logiciel permet aussi de scanner et de détecter les empreintes déjà enregistrer.

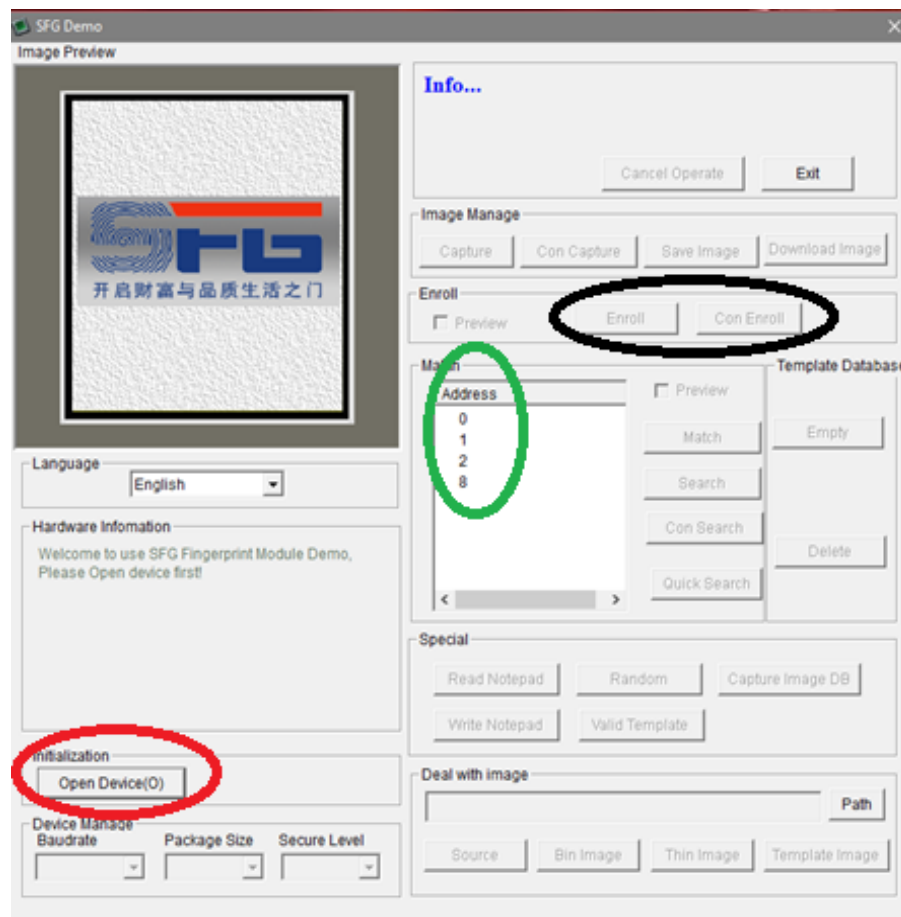


Figure 13: Logiciel SFG Demo

Cercle rouge : Connection avec le capteur.

Cercle noir : Permet d'enregistrer des nouvelles empreintes.

Cercle Vert : Affiche les empreintes numérotées.

Ensuite après avoir utiliser le logiciel, le programme peut commencer, tout d'abord il faudra installer des librairies, la première est la librairie du capteur d'empreinte digitale et la deuxième « softwareserial », les deux bibliothèques sont absolument essentielles pour le fonctionnement du capteur.

```

fingerprint $
#include <SoftwareSerial.h>

#include <Adafruit_Fingerprint.h>

#include <Servo.h>

#include <SoftwareSerial.h>
  
```

Figure 14: librairie Capteur d'Empreinte

Après avoir installer les librairies, on peut voir sur l'image ci-dessous, le void setup, la première fonction « finger.verifyPassword », qui détecte si le capteur est bien connecté, il affiche un message sur le moniteur série si Arduino le détecte bien.

La fonction « finger.getTemplateCount() », affiche le message sur le moniteur en série « attente d'une bonne empreinte ».

```

void setup()
{
  Serial.begin(9600);
  serrure.attach(5);
  while (!Serial); // For Yun/Leo/Micro/Zero...
  delay(100);
  Serial.println("\n\nAdafruit finger detect test");

  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor : (");
    while (1) { delay(1); }
  }

  finger.getTemplateCount();
  Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println(" templates");
  Serial.println("Waiting for valid finger...");
}
  
```

Figure 15: Void Setup Capteur d'Empreinte

Pour le void loop, la fonction principale qui servira tous au long du programme est « `finger.fingerFastSearch()` », ici, on utilise la fonction `switch`, qui avec les instructions `case`, va permettre d'afficher tous les conditions pour sa réalisation, par exemple « `case FINGERPRINT_OK` », affiche le message « pas d'empreinte détecté », cela veut dire que si le capteur ne détecte pas d'empreinte il affiche ce message.

```
void loop()                                // run over and over again
{
  getFingerprintIDez();
}

uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println("No finger detected");
      return p;
    case FINGERPRINT_PACKETRECEIVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }
}
```

Figure 16: Void loop(1)

Ensuite, toujours Pour le void loop, on utilise toujours la fonction principale qui servira tous au long du programme qui est « `finger.fingerFastSearch()` », ici, cette fonction affiche sur le moniteur en série des message d'erreur si il ne trouve pas la bonne empreinte ou si il en reçoit pas les packets, puis si il trouve la bonne empreinte il affiche « Found Id » et juste après le numéro de l'empreinte.

```
p = finger.fingerFastSearch();
if (p == FINGERPRINT_OK) {
  key = 1;
  Serial.println("Found a print match!");
} else if (p == FINGERPRINT_PACKETRECEIVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_NOTFOUND) {
  Serial.println("Did not find a match");
  return p;
} else {
  Serial.println("Unknown error");
  return p;
}

// found a match!
Serial.print("Found ID #"); Serial.print(finger.fingerID);
Serial.print(" with confidence of "); Serial.println(finger.confidence);

return finger.fingerID;
```

Figure 17: Void loop(2)

2 - PROGRAMME ESP32(WIFI)

Tout d'abord, avant de programmer il faut créer sur discord qui servira au projet de communication sans fil, c'est-à-dire que c'est sur la plateforme discord que l'utilisateur recevra les messages venant du projet. Pour discord, il faut créer un « webhook » sur la plateforme. Un webhook est un robot qui envoie des messages automatisés directement dans le salon textuel de votre serveur.

Après avoir créé le webhook, le programme peut commencer, le programme est divisé en trois parties,

La première partie est très légère car elle contient juste le débit de donné, la fonction « connectWifi » qui permet de se connecte au Wifi et la fonction « sendDiscord » qui enverra le message sur la plateforme Discord. Cette partie inclus les autres parties du programme ou sont détaillé les fonctions « connectWifi » et « sendDiscord ».



```

discord_test_esp32  arduino_secrets.h  discord.h
#include "arduino_secrets.h"
/*
  Discord WebHook Example for ESP32
*/
#include "discord.h"

void setup() {
  Serial.begin(9600);
  connectWIFI();
  sendDiscord("Hello World!");
}

void loop() {
}
  
```

Figure 18: Programme ESP32(1)

Ensuite la deuxième partie sont toutes les identifications qui permettent de se connecter au Wifi et à Discord comme sur le programme ci-dessous.

```
discord_test_esp32  arduino_secrets.h  discord.h
#define SECRET_SSID "iPhone de Ayoub"
#define SECRET_PASS "ayoub063"
#define SECRET_WEBHOOK "https://discordapp.com/api/webhooks/570236438096904202/1TGtg9Lm7y40csa_BdcQnmWmEGub8Hv23LJQVB-CTo8Z__4PcOb4TjJXQ3sfItARpZCs"
#define SECRET_TTS "false"
```

Figure 19: Programme ESP32(2)

Il suffit juste d'indiquer le nom du Wifi et son mot de passe pour la connexion, il faut aussi indiquer l'url du discord que l'on obtient en allant dans les paramètres du salon discord comme sur l'image ci-dessous.

MODIFIER LE WEBHOOK

Captain Hook #général

ICÔNE DU WEBHOOK
Nous recommandons une image d'au moins 256×256

Envoyer une image

Taille minimum : 128x128

URL DU WEBHOOK
https://discordapp.com/api/webhooks/570236... Copy

Annuler Enregistrer

Figure 20: Webhook

3 - PROGRAMME SERVOMOTEUR

Et pour terminer, le programme du servomoteur, ici, il suffit juste de programme en le faisant tourner d'un angle, le servomoteur joue le rôle d'un verrou, changé l'angle permet de passer de l'ouverture ou la fermeture de la serrure.

Le programme commence avec la librairie servo, qui est obligatoire pour faire marcher le programme.

Ensuite on indique deux positions, la position fermé et ouverte que l'on met dans deux variables.

Après cela, on indique le numéro de la pin auquel le servo moteur est branché.

Et pour finir le void loop, avec le changement de position, de fermé a ouverte avec la fonction openserrure().

```
#include <Servo.h>

Servo serrure; // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 180;    // variable to store the servo position
int key = 1;
void setup() {
  serrure.attach(5); // attaches the servo on pin 13 to the servo object
}

void loop() {
  if (key==1) {
    openserrure();
    key = 0;
  }
}
```

Figure 21: Servo moteur (1)

Le programme se termine avec la fonction « openserrure » qui permet le changement de position, en changeant d'angle, cela va permettre au servo de jouer le rôle d'un verrou.

```
void openserrure() {
  serrure.write(pos);
  delay(2000);
  pos = 90;
  serrure.write(pos);
  delay(2000);
}

// for (pos = 0; pos <= 180; pos += 1) {
//   serrure.write(pos);
//   delay(15);
// }
```

Figure 22: Servo moteur (2)

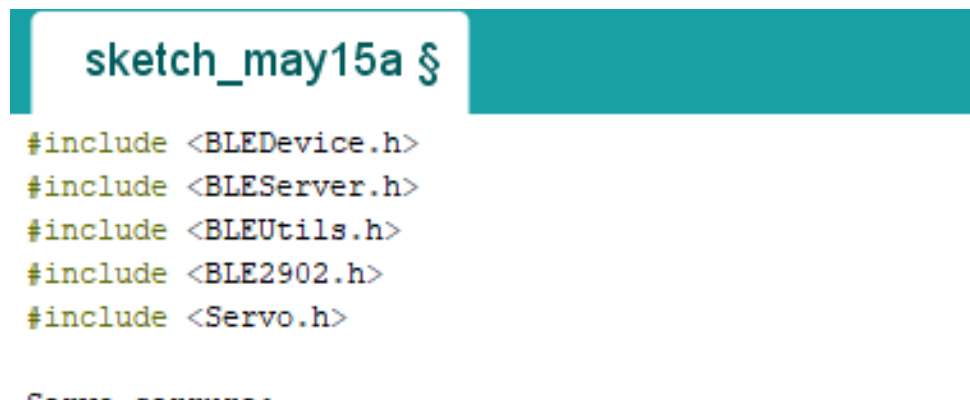
3 - PROGRAMME ESP32(BLUETOOTH) :

Le projet est séparé en deux parties, une partie expliquée juste avant, la première partie est la serrure qui s'ouvre et se ferme grâce au capteur d'empreinte digitale, un message s'envoie par discord pour alerter l'utilisateur.

Ici, le projet a une application sur smartphone permettant de débloquent la serrure par Bluetooth

Pour le programme Bluetooth, les ressources numériques nous ont fortement aidé.

Le début du programme commence par l'insertion des librairies sur l'image ci-dessous.



```
sketch_may15a §
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <Servo.h>
```

Figure 23: Librairie Bluetooth

Après avoir utilisé les librairies, il faut connecté par Bluetooth l'ESP32 à l'application,

Après cela il faut programmer l'application, quand on appuie sur le bouton de l'application, le moteur tourne, la serrure est ouverte, il y a un bouton pour ouvrir et un bouton pour fermer le verrou.

La programmation est complexe notamment au niveau de la connexion au Bluetooth, on voit sur le programme ci-dessous, où l'on aperçoit comment le moteur se tourne pour s'ouvrir et se ferme.

```

if (rxValue.length() > 0) {
  Serial.println("*****");
  Serial.print("Received Value: ");

  for (int i = 0; i < rxValue.length(); i++) {
    Serial.print(rxValue[i]);
  }

  Serial.println();

  // Do stuff based on the command received from the app
  if (rxValue.find("A") != -1) {
    Serial.print("ouverture de la porte!");
    serrure.write(180);
    delay(2000);
  }
  else if (rxValue.find("B") != -1) {
    Serial.print("");
    serrure.write(90);
    delay(2000);
  }

  Serial.println();
  Serial.println("*****");
}
}

```

Figure 24: Programme Bluetooth (1)

Ensuite, on voit comment se déroule la connexion Bluetooth, le message sur le serial moniteur s'affiche, s'il y'a une connexion Bluetooth, pour se connecter entre l'esp32 et un appareil Bluetooth, il suffit juste de se rendre dans les paramètres de son smartphone par exemple, et de se connecter à l'appareil qui se nomme « ESP32 », et à ce moment l'utilisateur peut lancer l'application afin d'ouvrir ou ferme le verrou.

```

class MyServerCallbacks: public BLEServerCallbacks {
  void onConnect(BLEServer* pServer) {
    deviceConnected = true;
  };

  void onDisconnect(BLEServer* pServer) {
    deviceConnected = false;
  }
};

class MyCallbacks: public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic *pCharacteristic) {
    std::string rxValue = pCharacteristic->getValue();

    if (rxValue.length() > 0) {
      Serial.println("*****");
      Serial.print("Received Value: ");

      for (int i = 0; i < rxValue.length(); i++) {
        Serial.print(rxValue[i]);
      }

      Serial.println();
    }
  }
};

```

Figure 25: Programme Bluetooth (2)

3 – APPLICATION BLUETOOTH :

Pour l'application Bluetooth, nous avons créé directement une application, la création est un avantage car elle permet de la créer comme l'utilisateur le veut.

La création se fait sur Thunkable. C'est un site qui permet de créer les applications sur smartphone et notamment Android.

Ici, on peut apercevoir la création de l'application, on voit les boutons « connecter », « déconnecter », « ouvert » et « fermé ».

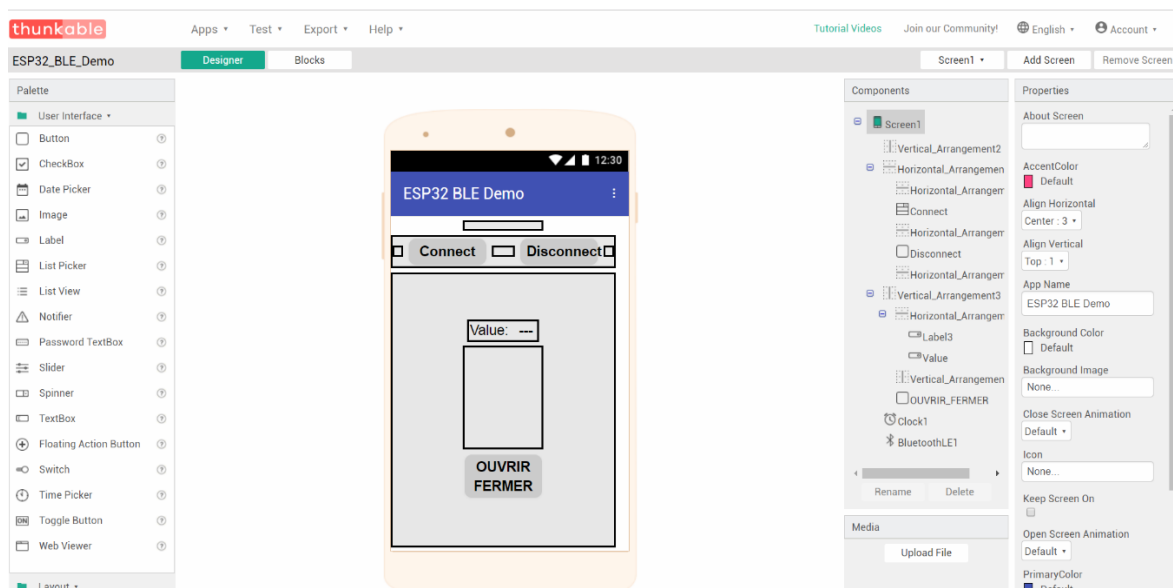


Figure 26: Création de l'Application Bluetooth

On voit ci-dessous, l'application Bluetooth qui a servi le groupe pour le projet, quatre boutons suffit amplement pour le projet, donc il faut lancer l'application se connecter, ensuite appuyer sur le bouton ouvrir ou fermé comme le veut l'utilisateur.

Une vidéo démonstration sur le GitHub montre bien l'utilisation de l'application avec le projet.



Value: ---



Figure 27: Application Bluetooth

III - AMÉLIORATIONS

Malheureusement, nous n'avons pas pu remplir tous nos objectifs que ce soient principaux ou secondaires, le projet est qu'une première version et ne peut qu'être améliorer dans le temps.

Une des premières améliorations à faire est de créer une maquette pour illustrer parfaitement le projet, la maquette permettra aussi de cacher les câbles et les capteurs. La maquette embellira le projet.

Dans un second temps, il serait vraiment très important de mettre tous les capteurs sur la carte Arduino Uno, cela permettrait une parfaite synchronisation de tous les capteurs, il suffira aussi d'une seule alimentation que pour la carte Arduino.

Et pour terminer, une des améliorations serait d'utiliser le capteur de son téléphone pour déverrouiller la serrure, le capteur du smartphone est beaucoup plus fiable que le celui du capteur d'empreinte digitale, cela permettra aussi un gain de temps important



Figure 28: Améliorations

CONCLUSION

Ce rapport a traité de toutes les notions étudiées lors l'UE communications sans fil, il a surtout parlé du projet final.

Ce projet a permis au membre du groupe de créer un objet connecté reliant tous les objectifs demandés par les professeurs (communication sans fil).

Lors de ce projet cela a aussi permis au membre du groupe d'apprendre le travail d'équipe sur le long terme, il a aussi été une expérience enrichissante car il nous a permis de mettre en pratique les notions étudiées en cours.

La liberté du choix de projet a ouvert au groupe une totale autonomie ainsi qu'une meilleure organisation.

Nous remercions les professeurs de nous avoir donner la chance de participer a des projets qui nous ont aidé beaucoup dans les années venir notamment grâce a l'expérience engrangée.



Figure 29: Conclusion

ANNEXE

RESSOURCES NUMÉRIQUES :

<https://randomnerdtutorials.com/esp32-servo-motor-web-server-arduino-ide/>

<https://learn.adafruit.com/adafruit-optical-fingerprint-sensor?view=all>

<https://circuitdigest.com/microcontroller-projects/using-classic-bluetooth-in-esp32-and-toogle-an-led>

<https://www.pexels.com/>

<https://github.com/Ayoub-ben/Serrure-Connectee>

LÉGENDES :

Figure 1 : Serrure Connectée.....	1
Figure 2 : Capteur d'Empreinte.....	2
Figure 3 : ESP32.....	3
Figure 4 : Objectifs Principaux.....	4
Figure 5 : Objectifs secondaires.....	5
Figure 6 : Planning.....	6
Figure 7 : Répartition du Travail (1).....	7
Figure 8 : Répartition du Travail (2).....	8
Figure 9 : Difficultés rencontrées.....	9
Figure 10 : Schéma Capteur d'Empreinte.....	10
Figure 11 : Schéma ESP32	11
Figure 12 : SFG Demo	12
Figure 13 : Librairie Capteur d'Empreinte.....	13
Figure 14 : Void Setup Capteur d'Empreinte	14
Figure 15 : Void loop(1) Capteur d'Empreinte	15
Figure 16 : Void loop(2) Capteur d'Empreinte	16
Figure 17 : Programme ESP32(1).....	17
Figure 18 : Programme ESP32(2).....	18
Figure 19 : Webhook	19
Figure 20 : Programme ESP32(3).....	20
Figure 21 : Programme Servo (1).....	21
Figure 22 : Programme Servo (2).....	22
Figure 23 : Librairie Bluetooth.....	23
Figure 24 : Programmation Bluetooth (1).....	23
Figure 25 : Programme Bluetooth (2).....	24
Figure 26 : Création de l'Application Bluetooth.....	20
Figure 27 : Application Bluetooth.....	21
Figure 28 : Améliorations).....	22

