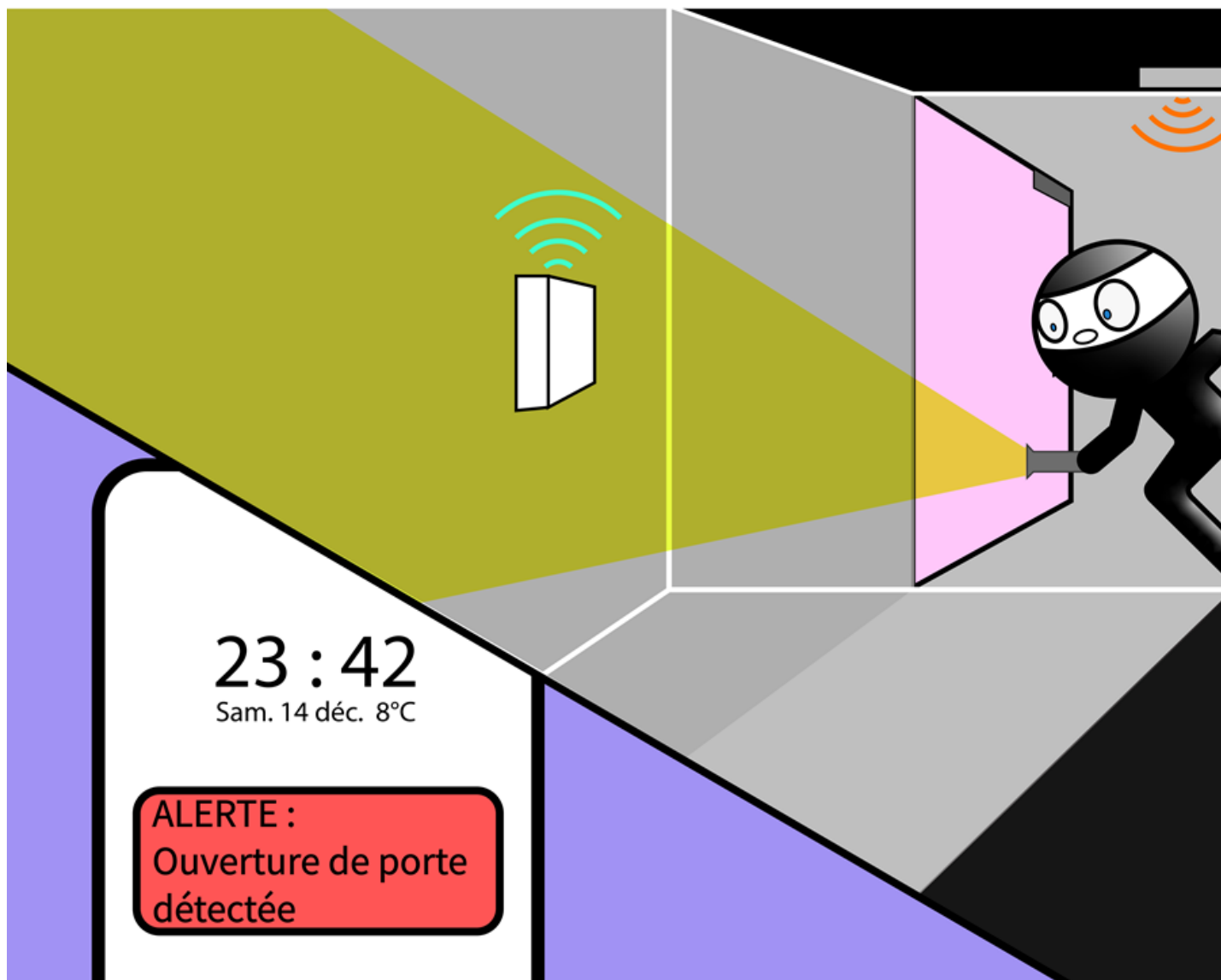


Rapport PST3 : Détection Autonome



Suiveur : Mr LEFAS

Membres : DEVOIR Charles-Philippe | DODARD Théo | GERNATH Adrien

Table des matières

Pourquoi ce projet ?	3
Quel est l'état de l'art ?	3
Que devons-nous produire (expression du besoin et spécifications) ?	3
Electronique :	4
Introduction :	4
Circuit électronique d'un capteur.....	4
Composant utilisé :	4
Le choix a-t-il été pertinent ?.....	5
Schéma :	5
Capteurs :	6
Magnétique :	6
De présence :	6
Code du capteur :	7
ESP32 :	8
Code de l'ESP32 :	8
Base de données :	9
Mobile :	10
Inscription/Connexion :	10
Espace utilisateur :	12
Paramètre du capteur :	15
Amélioration possible :	18
Miniaturiser le circuit :	18
Modéliser et imprimer un boîtier pour le circuit imprimé :	18

Pourquoi ce projet ?

Nous avons choisi ce projet car il comporte une partie électronique importante. Étant intéressés par la filière Systèmes Embarqués l'année prochaine, nous voulions que ce projet scientifique et technique allie la technique, à la vie de tous les jours. Le PST Détection Autonome est donc le compromis parfait puisqu'il est une définition de la domotique.

Quel est l'état de l'art ?

Notre projet se divise en deux domaines : Domotique et Informatique. La domotique rassemble les différentes techniques qui permettent de contrôler, programmer et d'automatiser une habitation. Les domaines de l'Électronique de l'Informatique sont regroupés dans le terme Domotique. Elle permet de programmer la plupart des appareils et dispositifs électriques d'une maison, de l'éclairage et du chauffage aux équipements audiovisuels et électroménagers en passant par l'ouverture des fenêtres. La domotique facilite également le contrôle de l'habitation en gérant les systèmes d'alarme, de préventions d'incendie mais encore de la température au sein des pièces.

Que devons-nous produire (expression du besoin et spécifications) ?

Nous devons produire deux modules mettant en avant chacun un capteur différent. Ces modules seront à placer directement sur une fenêtre pour le capteur magnétique et sur le plafond orienté vers la fenêtre pour le capteur de mouvement (infrarouge). Nous devrions également nous assurer que nos modules pourront être présentés lors du salon des PST, possiblement fabriquer un module permettant leur présentation.

Qu'avons-nous fait ?

Nous avons automatisé la détection d'infractions de natures différentes. En effet, l'objectif principal était de réaliser un livrable concernant le capteur magnétique. Or, le processus se ressemblant pour plusieurs autres capteurs, nous nous sommes penchés sur la possibilité de détecter différentes données de manière autonome. Nous avons utilisé un capteur infrarouge afin de détecter les mouvements ou la présence d'un individu au sein d'une pièce.

Nous avons codé sur Arduino les composants de manière à recevoir et envoyer une donnée lorsqu'un capteur détecte un événement. Ces données sont stockées dans une base de données distante.

Nous avons codé, en parallèle, une application Android dans le but d'avertir l'utilisateur en temps réel. Cette application récupère les informations du capteur actif et affiche, sur l'application, l'horaire de chaque détection.

Electronique :

Introduction :

L'électronique est le cœur de notre projet. Il est donc important qu'il soit à la fois efficace, simple d'entretien et peu coûteux.

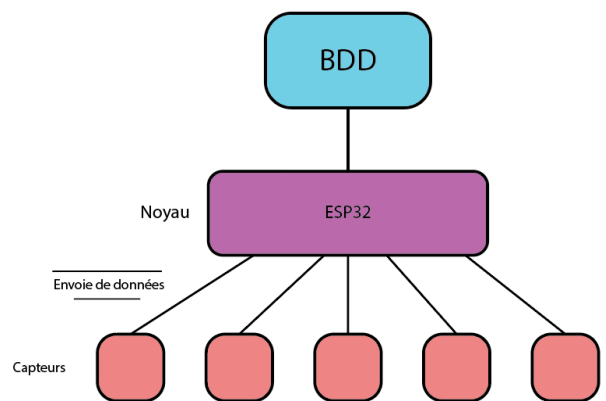
Pour répondre à ces critères, nous avons choisi de créer des petits capteurs capables de détecter un événement (exemple : ouverture d'une porte, présence d'un individu), et d'envoyer une donnée. Aucun calcul ne sera effectué par ce capteur. En effet, dans le but de gagner en rapidité, toutes les données seront traitées par « l'unité centrale ».

Pour simplifier la compréhension, on peut voir ce dispositif comme un arbre :

Lorsqu'un capteur détecte un événement, il va envoyer son numéro de série au noyau. Dans notre cas, cette information suffit, mais dans le cas d'un capteur d'humidité ou de température, il faudrait également envoyer la valeur relevée par le capteur.

Le noyau peut recevoir 18 données de capteur à la fois.

Le noyau récolte toutes les données, les traite puis les envoie à la base de données. Avant d'envoyer la donnée, nous nous assurons que le capteur n'avait pas déjà envoyé une information dans un délai d'une minute, pour éviter la surcharge dans la base de données.

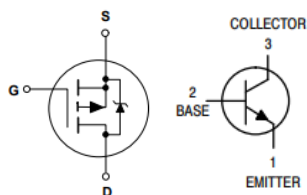


Circuit électronique d'un capteur :

Composant utilisé :

Nous avons cherché à ce que nos circuits soient les plus petits possible c'est pour cela que nous avons choisi un attiny85 comme microcontrôleur. Sa caractéristique principale étant sa petite taille et son faible coût, il est également possible de le programmer au travers d'un Arduino.

Deux transistors, NDP6020P et 2N3903.



Un nRF24L01 servant d'émetteur, mais à la particularité d'être à la fois un transmetteur.

Et une pile de 3.3V car l'attiny85 et le nRF24L01 ne fonctionnent qu'avec une alimentation de 3.3V.

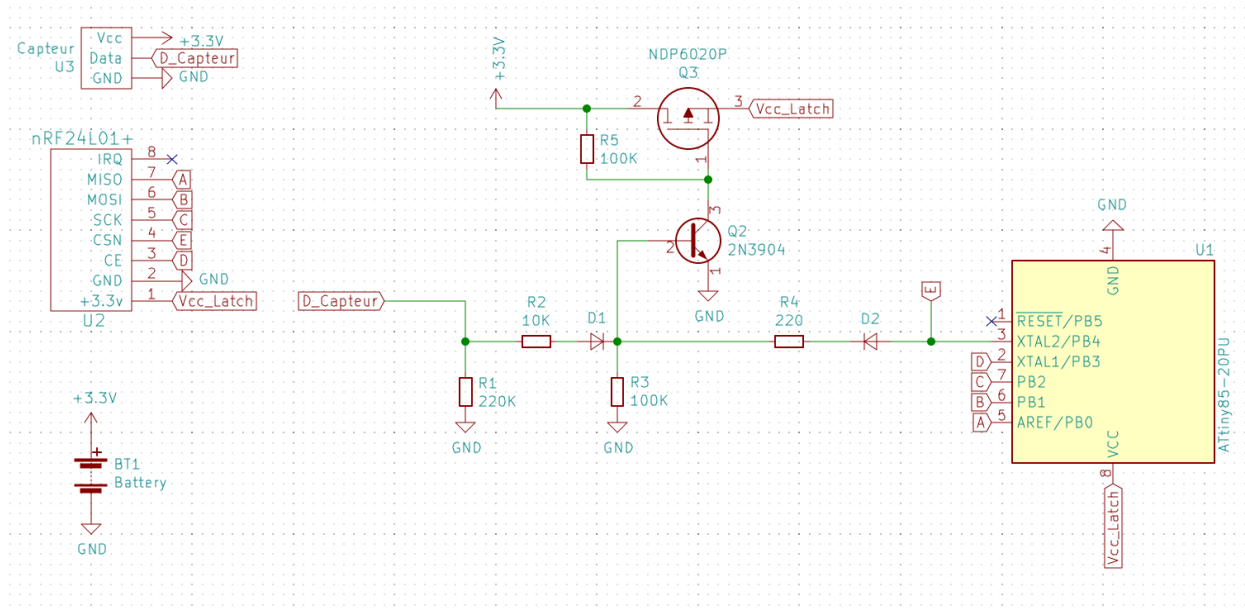
Le choix a-t-il été pertinent ?

Le choix pouvait nous sembler pertinent dans un premier temps car nous avons pour seul objectif d'implémenter des capteurs capables de se mettre dans l'état 0 ou 1.

Dans la possibilité d'une évolution du projet, l'attiny85 atteindrait très vite ses limites. En effet, avec des capteurs « binaire », le microcontrôleur n'a plus aucun pin de libre. L'implémentation de capteurs comme un ceux de température ou d'humidité serait donc impossible.

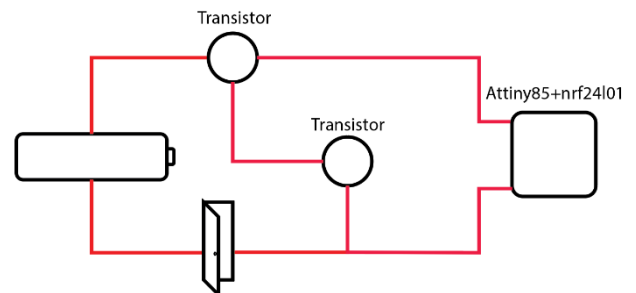
Il faudra donc prendre un microcontrôleur avec plus de pins.

Schéma :

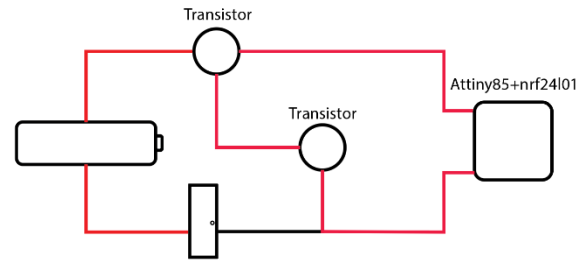


Notre circuit est basé sur un circuit latch, c'est-à-dire qu'il utilise du courant seulement lorsque le capteur détecte un évènement.

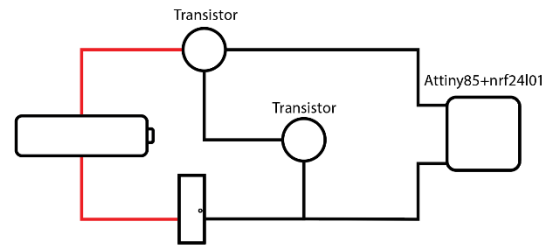
Circuit lorsque le capteur détecte un évènement :



Circuit lorsque le capteur ne détecte plus d'événement, mais que l'attiny85 est encore en train d'envoyer la donnée :



Circuit lorsqu'il est éteint :

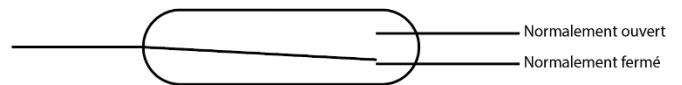


Capteurs :

Magnétique :

Pour notre capteur magnétique, nous avons choisi un interrupteur Reed à 3 broches :

L'avantage c'est qu'avec cet interrupteur nous pouvons avoir, un capteur envoyant des données uniquement lorsqu'il est en contact avec un aimant, et un autre envoyant des données lorsqu'il perd le contacte avec l'aimant.



De présence :

Pour notre capteur de présence, nous avons choisi un détecteur infrarouge Gravity SEN0018, principalement pour son prix. Lorsqu'il détecte une présence, il envoie un courant que l'on utilise pour allumer le circuit et envoyer l'information. Il est possible de régler la sensibilité du capteur afin qu'il détecte dans un espace plus ou moins grand.

Code du capteur :

```
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>

#define CE_PIN 3
#define CSN_PIN 4

const int powerLatch = 4;

RF24 radio(CE_PIN, CSN_PIN);

//Adresse sur le quel les deux modules radio communique
const byte address[6] = "node1";

void setup() {

    // Définition des pins
    pinMode(powerLatch, OUTPUT);

    // Garde le circuit allumé
    digitalWrite(powerLatch, HIGH);

    //Initialisation du module radio
    BegeningTransmission();

}

void BegeningTransmission(){
    radio.begin(); // Allume la radio

    radio.openWritingPipe(address); // Mode emetteur
}

void loop(void){
    bool send = false;
    const char text[] = "/*Numéro de série*/";
    while (send == false) //S'assure que le message c'est bien envoyé
    {
        if (!radio.write(&text , sizeof(text))) //Envoie le message
        {
            send = false;
        }
        else
            send = true;
    }
    digitalWrite(powerLatch, LOW); //Permet d'éteindre le circuit
    exit(0);
}
```

ESP32 :

Nous avons choisi l'ESP32 comme noyau car, en plus d'être peu coûteux, il dispose d'un module WIFI d'intégré.

Nous l'utilisons principalement pour traiter les données envoyées par les capteurs, mais également pour envoyer ces données dans la base une fois traitées.

Code de l'ESP32 :

```
#include <WiFi.h>
#include <HTTPClient.h>

#include "nRF24L01.h"
#include "RF24.h"

RF24 radio(4, 5); //Pin MESSO et MISSO
const byte address[6] = "node1";

const char* ssid = "/*Nom de la WIFI*/";
const char* password = "/*Mdp de la WIFI*/";
const char* host = "/*Adresse ip*/";

void setup()
{
    //Initialisation du module radio
    Serial.begin(115200);
    Serial.println("Starting...");
    radio.begin();
    radio.setPALevel(RF24_PA_MIN); //Mode de consommation faible
    radio.openReadingPipe(0, address); //Pin de reception 0
    radio.startListening(); //Mode transmetteur

    // Connexion à la WIFI
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }

    Serial.println("Wifi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void loop()
{
    //Si une donnée reçus par la radio alors
    if (radio.available())
    {
        char numSerie[32];
        while (radio.available())
        {
            //On lit la donnée reçus
            int len = radio.getDynamicPayloadSize();
            radio.read(&numSerie, len);
        }

        HTTPClient http;

        WiFiClient client;
        //On entre dans la BDD l'activation d'un capteur
        http.begin(String("https://nimble-lead-277612.ew.r.appspot.com/?id=0&numSerie=") + numSerie); //On prépare la connexion
        int httpCode = http.GET(); //On execute
        http.end(); //On ferme la connexion
    }
}
```


Base de données :

La base de données est le lien qui unie les parties électroniques et informatiques de notre projet.

On y enregistre chaque détection effectuée par n'importe quel capteur.

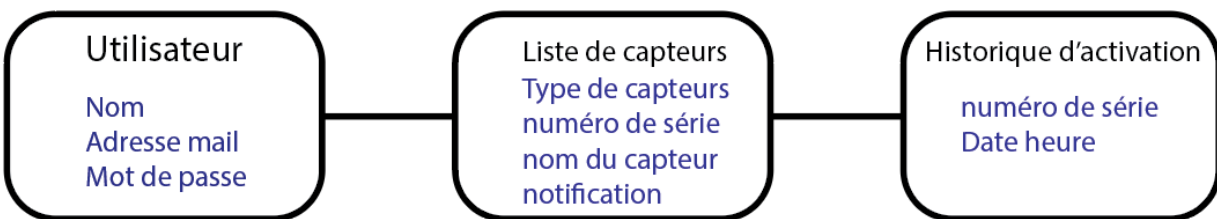
En revanche, il y a un délai d'une minute qui équivaut à une sécurité. En effet ce délai s'active lorsque le capteur détecte une infraction. Il empêche la base de données de se remplir inutilement si le capteur détecte plusieurs fois la même donnée pendant ce délai.

Lors d'une détection, le type du capteur qui a détecté un évènement, son numéro de série ainsi que la date de détection sont ajoutés dans la table. Une ligne s'ajoute dans cette table à chaque détection d'un capteur.

Dans le but de classer les informations qu'elle divulgue, les informations concernant les utilisateurs de l'application mobile sont aussi enregistrées. Cela permet de traiter les informations individuellement via l'interface mobile.

Nous avons été contraints d'utiliser une base de données en ligne. Effectivement, le module ESP32 ne peut se connecter correctement à une base de données locale via phpmyadmin comme vu en cours. Nous avons donc utilisé un serveur en ligne via google cloud.

Le schéma ci-dessous illustre synthétiquement notre utilisation de la base de données.



Mobile :

L'application mobile correspond à l'interface utilisateur de notre projet. Elle est simple et compréhensible pour tous types d'utilisateurs. Ses principales fonctionnalités sont les suivantes :

Inscription/Connexion :

Un formulaire de connexion et d'inscription permet évidemment à l'utilisateur de rester connecté s'il le souhaite mais il nous permet également de gérer sa session afin de le rediriger vers sa propre page d'accueil et que nous puissions identifier les capteurs de nos utilisateurs lorsqu'ils s'activent.

```
$add = '';
$username = $add.$_GET['username'].$add;
$email = $add.$_GET['email'].$add;
$password = $add.$_GET['password'].$add;

$trimemail = trim($email, ''); //Permet de supprimer les "" qui encadre notre variable

if ($req = $connect->query("SELECT * FROM users WHERE email = '$trimemail'"))
{
    if ($req->num_rows == 0)
    {
        $req->free();
        $register = $connect->prepare("INSERT INTO users(username, email, password) VALUES('$username', '$email', '$password')");
        $register->execute();
        echo "Successfully Registered !";
        $register->free();
    }
    else
        echo "Mail already used, please change it !";
}
else
    echo "Something went wrong, please retry !";
```

On retrouve l'inscription de l'utilisateur dans le code ci-dessus, ainsi que le formulaire correspond ci-contre.

The image shows a mobile application interface for a registration screen. At the top, there is a status bar with the time 19:03 and various icons. Below the status bar is a navigation bar with a back arrow and the text 'Retour'. The main content area has a title 'Register' in bold. Below the title are three input fields labeled 'Username', 'Email', and 'Password'. At the bottom of the form is a rounded rectangular button labeled 'Register'. The interface is displayed on a smartphone screen with a black home indicator bar at the very bottom.

En ce qui concerne le code ainsi que l'interface ci-dessous, ils coïncident avec la connexion de l'utilisateur à son compte personnel.

```
$email = $_GET['email'];
$password = $_GET['password'];

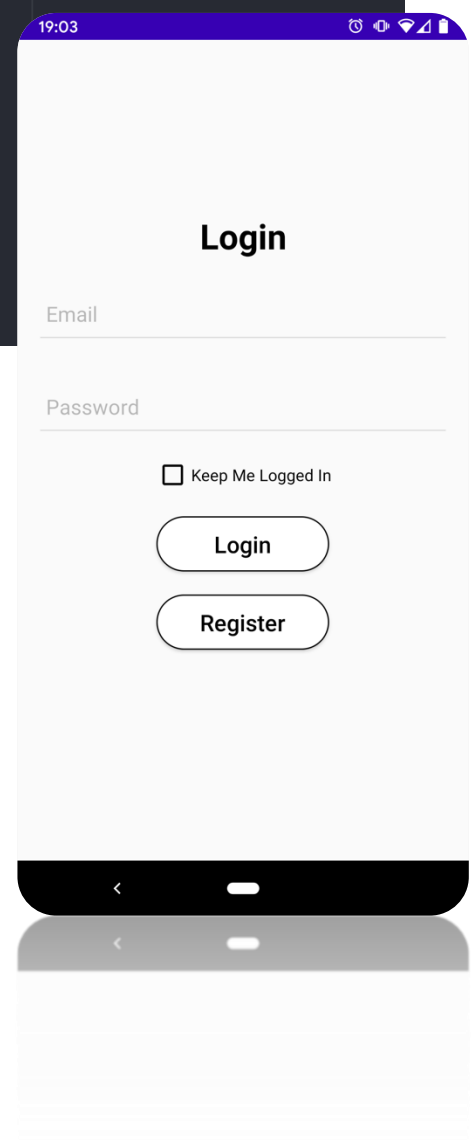
$req = $connect->query("SELECT * FROM users WHERE email = '".$email."'");

if($req->num_rows > 0)
{
    $reqLogin = $connect->query("SELECT * FROM users WHERE email = '".$email."' AND password = '".$password."'");

    if($reqLogin->num_rows > 0)
    {
        echo "Login Success";

        $userinfo = $reqLogin->fetch();
        //Insertion en BDD

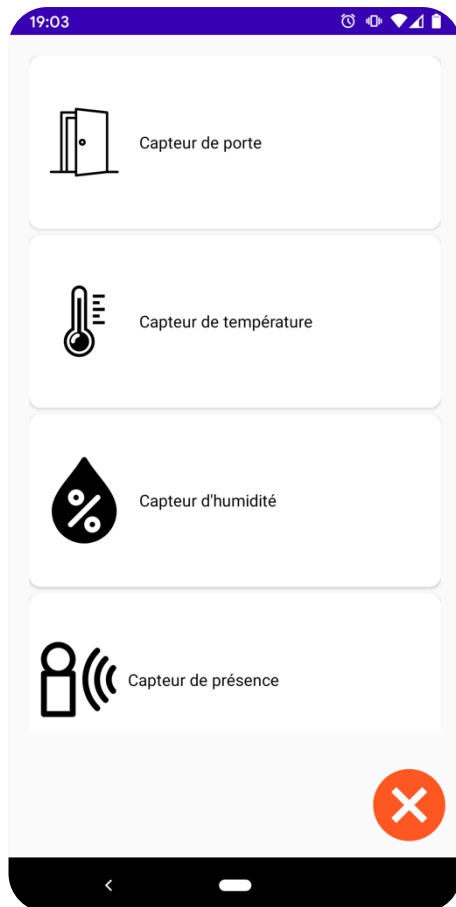
    }
    else
    {
        echo "Wrong Password";
    }
}
else
{
    echo "This email is not registered";
}
```

A mobile application interface for a login screen. The screen has a white background with a dark blue header bar at the top displaying the time 19:03 and various status icons. The title "Login" is centered in a bold, black font. Below the title are two input fields: "Email" and "Password", each with a light gray border and a light gray placeholder text. Below the "Password" field is a checkbox labeled "Keep Me Logged In". At the bottom of the form are two rounded rectangular buttons: "Login" and "Register", both with black text and a thin black border. The interface is shown on a smartphone with a black bezel and a white home indicator bar at the bottom.

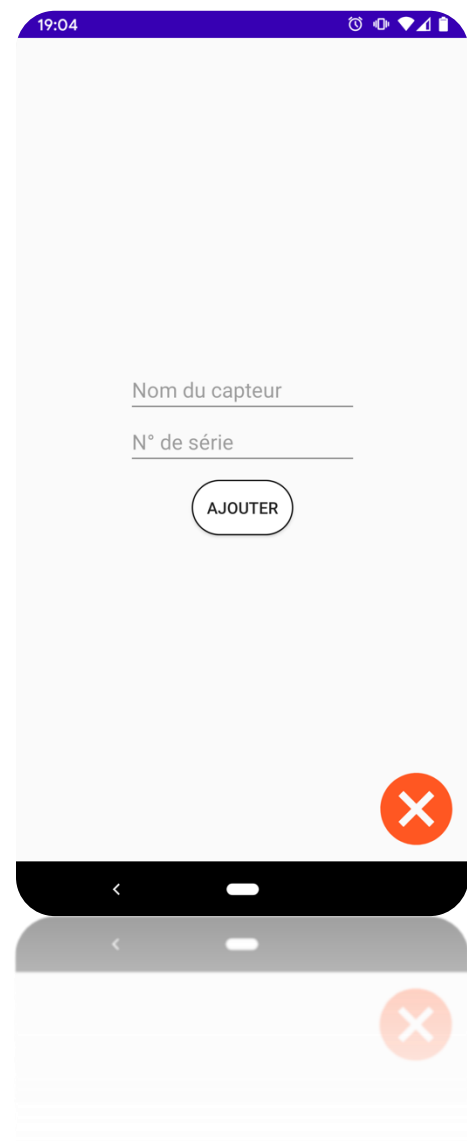
Espace utilisateur :

La page d'accueil permet à l'utilisateur de visualiser les capteurs qu'il possède déjà. On retrouve le type de capteur, son numéro de série mais également le nom que l'utilisateur lui a donné.

Sur cette même page le client peut ajouter un capteur à sa liste. On retrouve ci-dessous les deux étapes qui concernent l'ajout d'un capteur.



Dans un premier temps, le client choisit le type de capteur qu'il vient d'acheter et qu'il souhaite ajouter sur l'application.



Ensuite, il renseigne le nom du capteur selon son choix et le numéro de série de son capteur. Le capteur sera inséré sur la page d'accueil du client une fois la procédure complétée.

Le code ci-dessous permet l'ajout d'un nouveau capteur à la liste de l'utilisateur.

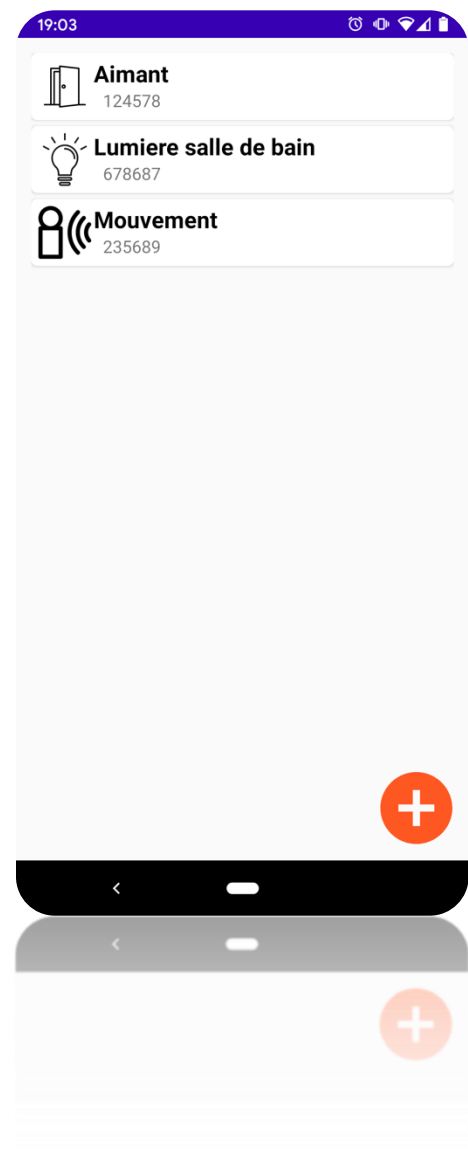
```
$add = '';
$numSerie = $_GET['numSerie'];
$numCapteur = $add.$_GET['nomCapteur'].$add;
$typeCapteur = $_GET['typeCapteur'];

$req = $connect->query("SELECT * FROM capteur_list WHERE numSerie = '".$numSerie."'");

if($req->num_rows == 0)
{
    $sql_register = $connect->prepare("INSERT INTO capteur_list(numCapteur, numSerie, typeCapteur) VALUES('".$numCapteur."', '".$numSerie."', '".$typeCapteur."')");
    $sql_register->execute();
    echo "Successfully added !";
}
else
{
    echo "Serial number already used !";
}
```

La page d'accueil utilisateur s'apparente à l'écran ci-contre :

Pour ajouter un capteur à cette liste, il a fallu cliquer sur le « + ».



La mise en forme des informations contenues dans la base de données s'effectue selon le code ci-dessous :

```
$req = "SELECT typeCapteur, numSerie, nomCapteur, notification FROM capteur_list";

$result = mysqli_query($connect, $req);

$json_array = array();

while ($row = mysqli_fetch_assoc($result))
{
    $json_array[] = $row;
}
echo '{"Liste":';
echo json_encode($json_array);
echo '"';
```

Code qui permet de disposer les informations sous format JSON comme ci-dessous :

```
{
  "Liste": [
    {
      "typeCapteur": "3",
      "numSerie": "235689",
      "nomCapteur": "Mouvement",
      "notification": "1"
    },
    {
      "typeCapteur": "0",
      "numSerie": "124578",
      "nomCapteur": "Aimant",
      "notification": "1"
    }
  ]
}
```

Paramètre du capteur :

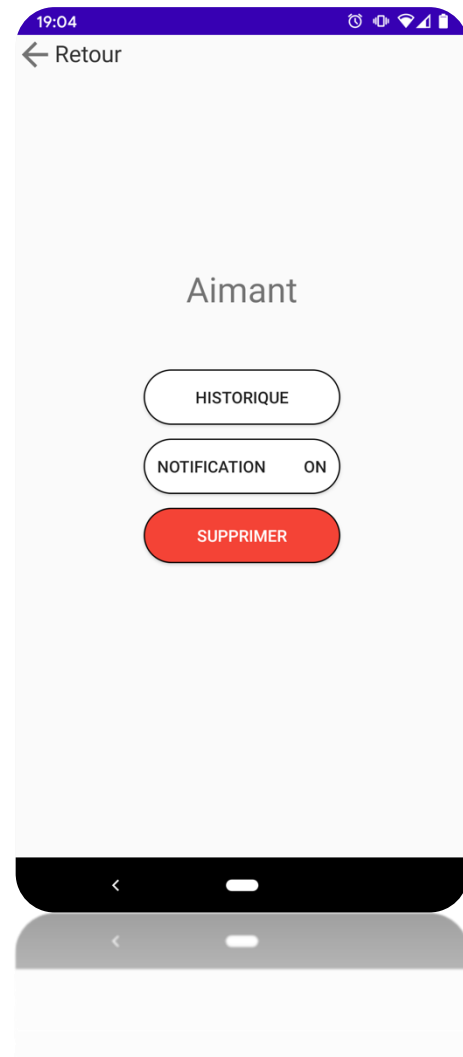
En cliquant sur un capteur, l'utilisateur peut avoir accès à de nouvelles informations sur son capteur.

Cela nous amène la page ci-contre :

On y retrouve le nom du capteur avec quelque paramètres sur celui-ci :

Activer les notifications permettra à l'utilisateur de les recevoir lorsque ce capteur aura détecter un évènement.

On retrouve le code correspondant ci-dessous :



```
$numSerie = $_GET['numSerie'];
$numSerie = trim($numSerie, '');
$numSerie = trim($numSerie, '"');

$notif = $connect->query("SELECT notification FROM capteur_list WHERE numSerie = '". $numSerie . "'");
$result = $notif->fetch_row();
$mode = $result[0];

if ($mode == "0")
{
    $reqON = $connect->prepare("UPDATE capteur_list SET notification = 1 WHERE numSerie = '". $numSerie . "'");
    $reqON->execute();
}
elseif ($mode == "1")
{
    $reqOFF = $connect->prepare("UPDATE capteur_list SET notification = 0 WHERE numSerie = '". $numSerie . "'");
    $reqOFF->execute();
}
else
    echo "Failure !";
```

Le bouton « Supprimer » engendrera la suppression du capteur de votre liste. En revanche, cela ne supprimera pas son historique de détection. Cela permet, en cas de suppression par inadvertance, de retrouver le même historique après rajout du même capteur, grâce à son numéro de série.

La suppression est régie par le code ci-dessous :

```
$numSerie = $_GET['numSerie'];
$numSerie = trim($numSerie, '');
$numSerie = trim($numSerie, '"');

$id = $connect->query("SELECT ID_list FROM capteur_list WHERE numSerie = '".$numSerie.'");

if($id->num_rows != 0)
{
    $req = $connect->prepare("DELETE FROM capteur_list WHERE numSerie = '".$numSerie.'" ");
    $req->execute();
    echo "Successfully deleted !";
}
else
    echo "Sensor already deleted !";
```


L'historique relate les détections présentes dans la base. Seules ces données mettent à jour cette activité. Le code ci-dessous arrange les données dans un tableau JSON.

```
$numSerie = $_GET['numSerie'];

$req = "SELECT time FROM capteur_info WHERE numSerie = '". $numSerie . "'";

$result = mysqli_query($connect, $req);

$json_array = array();

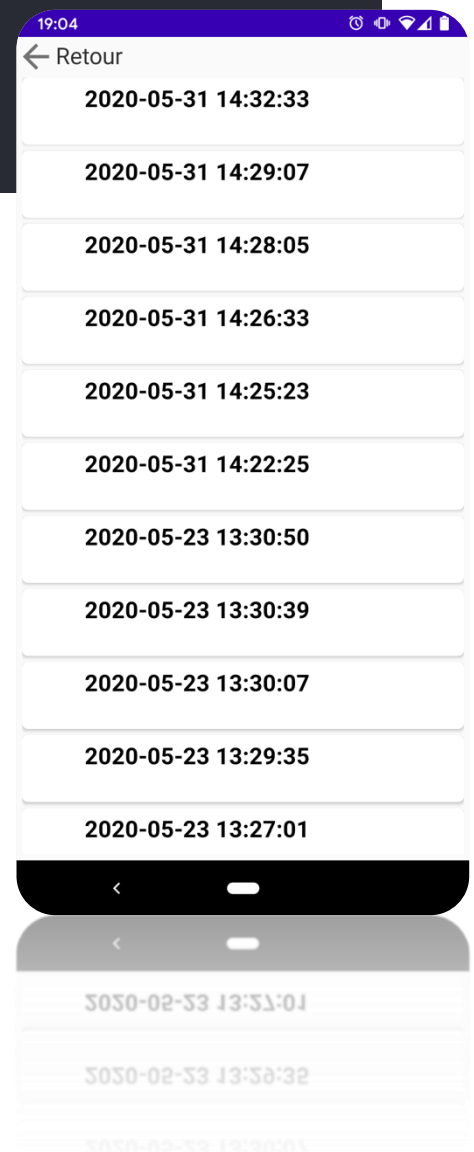
while ($row = mysqli_fetch_assoc($result))
{
    $json_array[] = $row;
}

echo '{"Liste":';
echo json_encode($json_array);
echo '"}";
```

Tableau de données JSON ci-dessous :

```
{
  "Liste": [
    {
      "time": "2020-05-23 12:54:35"
    },
    {
      "time": "2020-05-23 12:54:42"
    },
    {
      "time": "2020-05-23 12:55:02"
    }
  ]
}
```

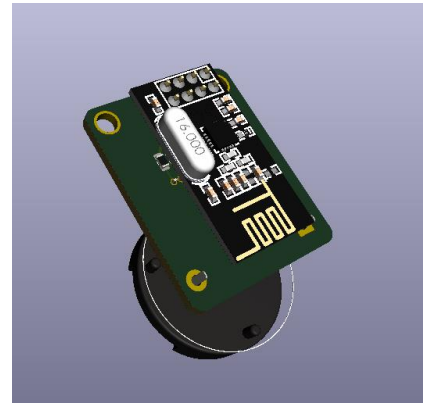
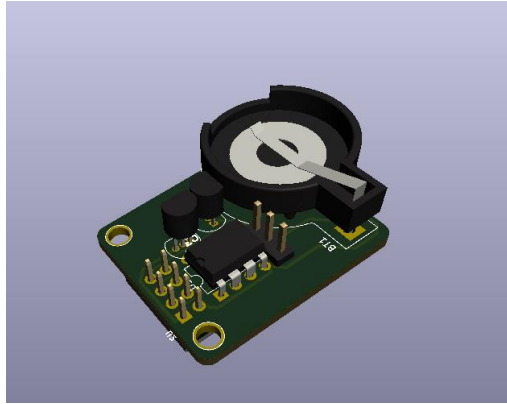
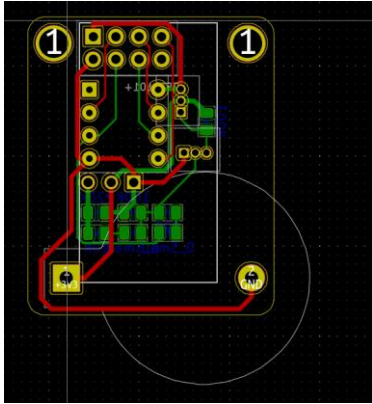
L'interface ci-contre permet d'avoir une idée très précise du moment où l'infraction a été détectée.



Amélioration possible :

Miniaturiser le circuit :

Pour avoir un produit fini faire un circuit imprimé serait une possibilité



Modéliser et imprimer un boîtier pour le circuit imprimé :

