Rapport de projet :

Préambule

Le projet

Faire un digicode à 2 input, communiquant à distance.
Le code d'accès sera modifiable à distance via l'application. Plusieurs codes d'accès seront disponible.

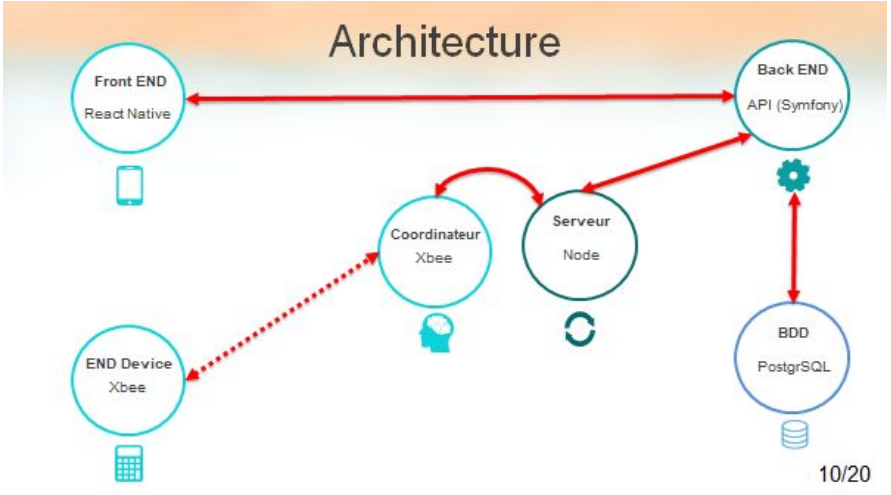
Contexte d'utilisation

-Porte d'entrée
-Escape Game
-Code éphémères pour un livreur de colis
-Air BnB

Pré-requis

-Logiciel XCTU
-Ordinateur (sous Windows or Mac)
-PhpStorm (+ PHP & Node.JS)
-Docker (+ docker-compose cli)
-NPM
-YARN
-Xbee (+ différent matériel électronique: LED, boutons poussoirs, fils, breadboard ..)

Architecture



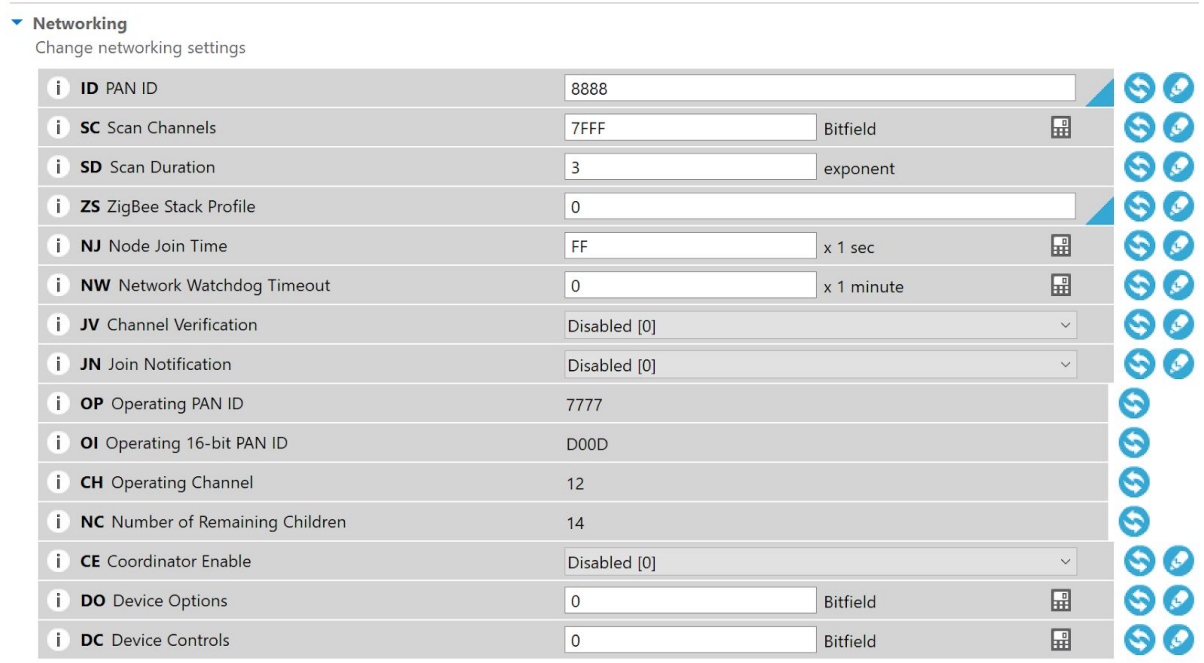
1 - Conception et installation du matériel

La conception de ce projet ai fais en 2 parties, une partie qui permet de recevoir les informations et un autre permettant de les envoyer.

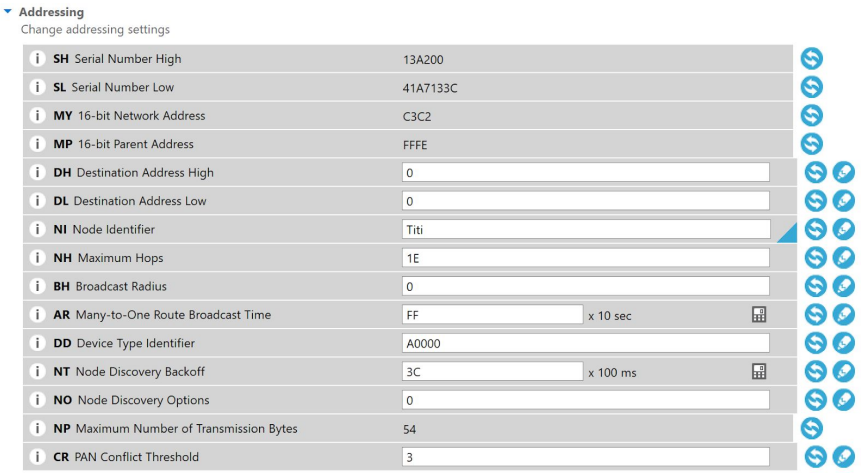
Partie récepteur

Commandes à faire :

AT ID 8888
AT CE 0



AT DH 0
AT DL 0



AT AP 0

Serial Interfacing

Change modem interfacing options

BD Baud Rate	9600 [3]	
NB Parity	No Parity [0]	
SB Stop Bits	One stop bit [0]	
RO Packetization Timeout	3	x character times
D6 Pin 16 - DIO6/nRTS Configuration	Disable [0]	
D7 Pin 12 - DIO7/nCTS Configuration	nCTS flow control [1]	
AP API Enable	Transparent mode [0]	
AO API Output Mode	Native [0]	

AT D0 5
AT D1 5
AT D2 5
AT D3 5

I/O Settings

Modify DIO and ADC options

D0 Pin 20 - DIO0/AD0/CB Configuration	Digital Out, High [5]	
D1 Pin 19 - DIO1/AD1/nSPI_ATTN Configuration	Digital Out, High [5]	
D2 Pin 18 - DIO2/AD2/SPI_SCLK Configuration	Digital Out, High [5]	
D3 Pin 17 - DIO3/AD3/nSPI_SSEL Configuration	Digital Out, High [5]	
D4 Pin 11 - DIO4/SPI_MOSI Configuration	Disabled [0]	
D5 Pin 15 - DIO5/Associated Configuration	Associated indicator [1]	
D8 Pin 9 - DIO8/nDTR/Sleep_Rq Configuration	Sleep_Rq [1]	
D9 Pin 13 - DIO9/nOn_Sleep Configuration	Awake/Asleep indicator [1]	
P0 Pin 6 - DIO10/RSSI PWM/PWM0 Configuration	RSSI PWM Output [1]	
P1 Pin 7 - DIO11/PWM1 Configuration	Disabled [0]	
P2 Pin 4 - DIO12/SPI_MISO Configuration	Disabled [0]	
P3 Pin 2 - DIO13/DOUT Configuration	DOUT [1]	
P4 Pin 3 - DIO14/DIN/nConfig Configuration	DIN [1]	
PR Pull-up Resistor Enable	1FBF	
PD Pull-up/down Direction	1FFF	
LT Associate LED Blink Time	0	x10 ms
RP RSSI PWM Timer	28	x 100 ms

AT IR 0
AT IC FFFF

I/O Sampling

Configure IO sampling parameters

IR IO Sampling Rate	0	x 1 ms
IC Digital IO Change Detection	FFFF	
V+ Supply Voltage High Threshold	0	

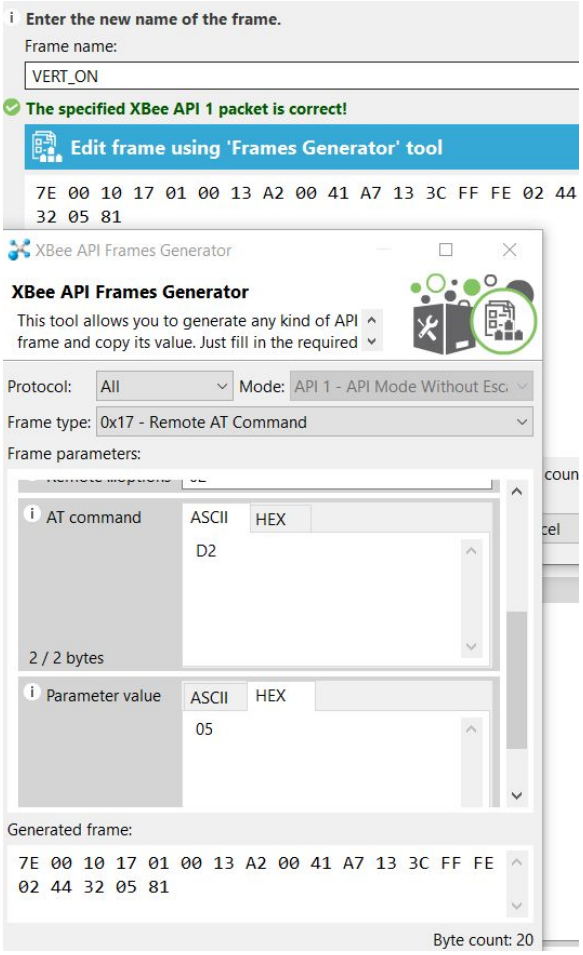
Partie Coordinateur

Commandes à faire :

AT ID 8888
AT CE 1
AT DH 13A200
AT DL 41A7133C
AT AP 1

Frames envoyés par le Coordinateurs depuis XCTU :

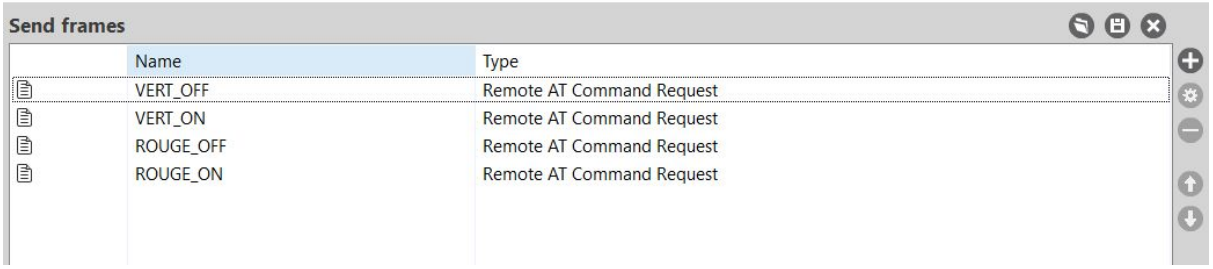
Allumer Led Vert :

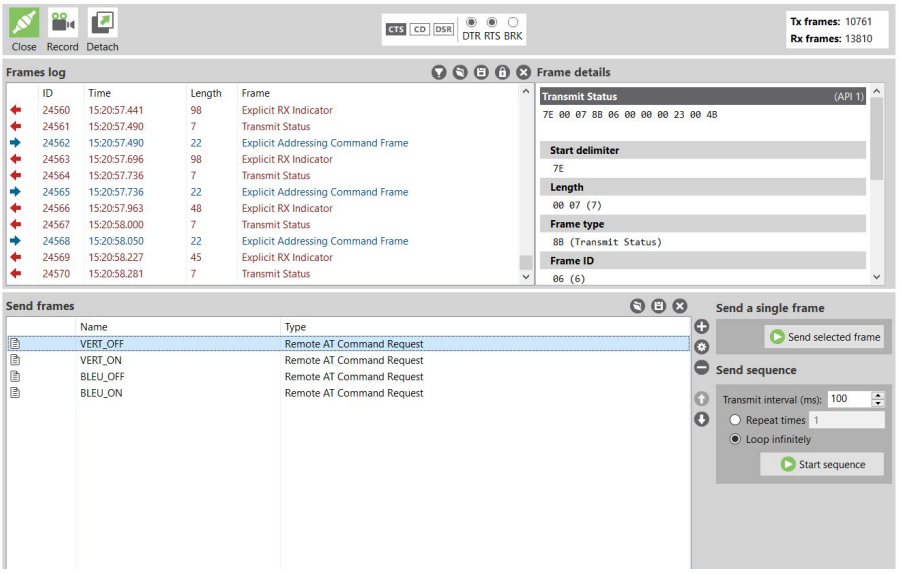


Pour éteindre je passe en parametre 00 (disable)

Je peux créer en suivante cette logique une séquence, dans l'exemple ci-dessous, Je fait clignoter la led vert puis la led bleu autant de fois que je le souhaite (je peux aussi paramétrer la vitesse d'exécution de la séquence dans le champ "Transmit interval").

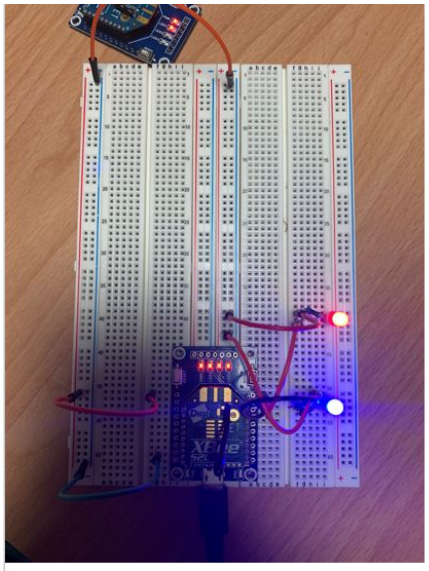
Ici la liste des frames que j’utilise

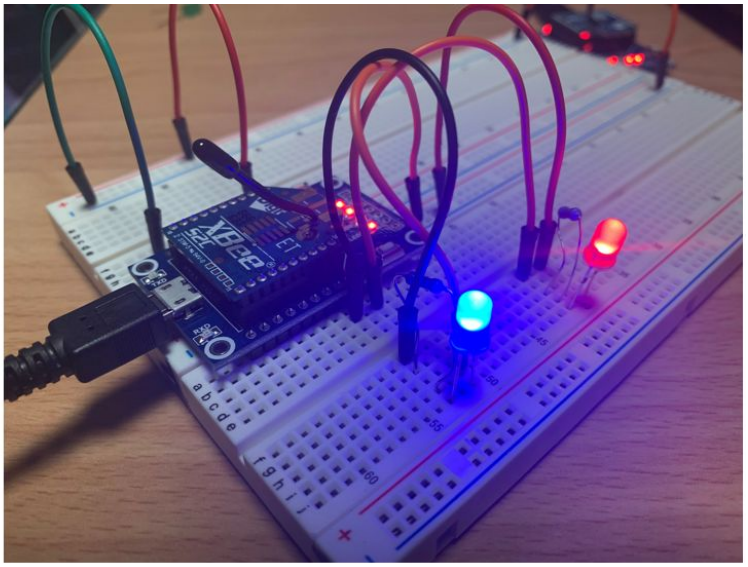


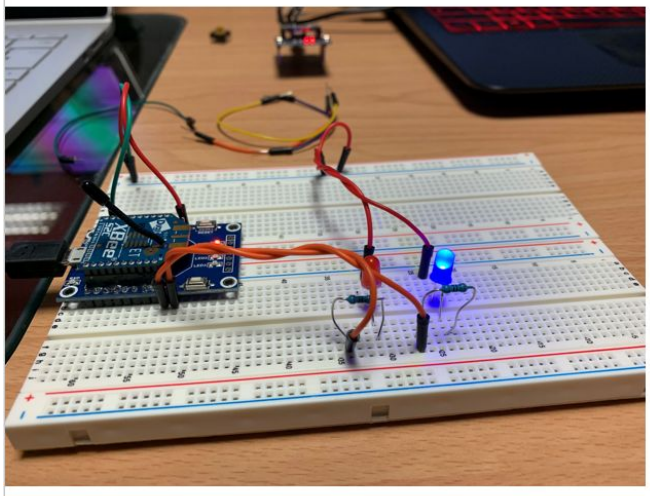


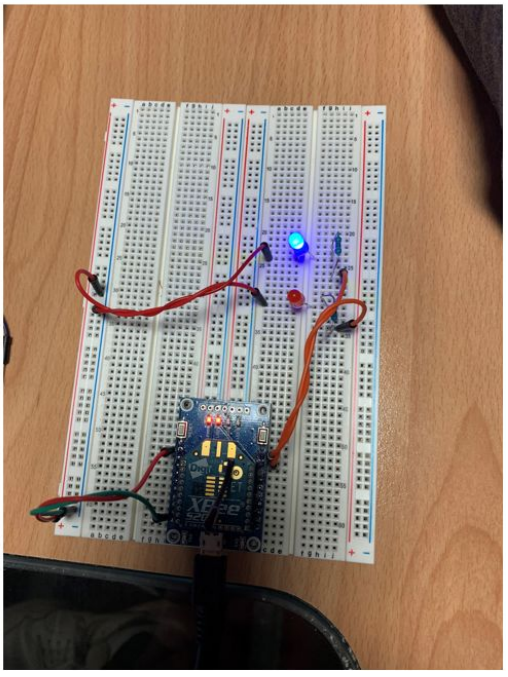
2 - Evolution du Montage

Montage sans bouton

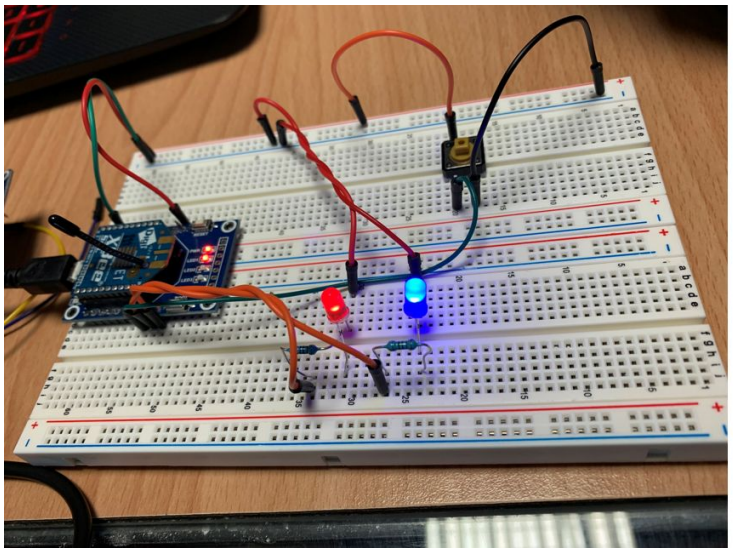


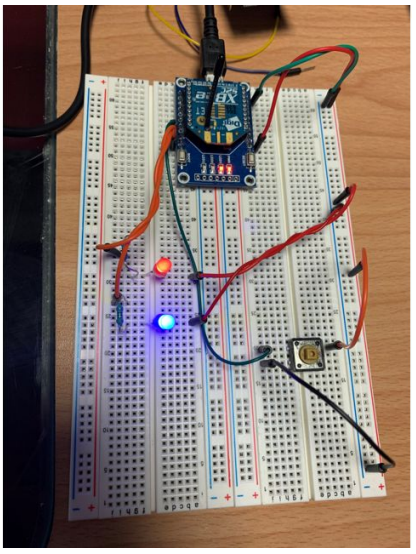




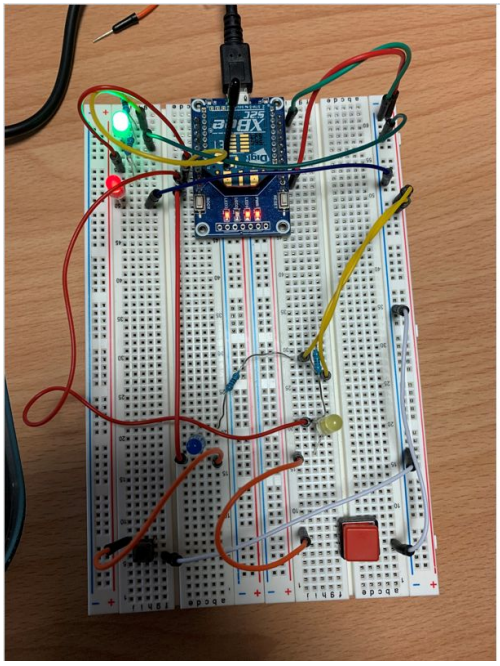


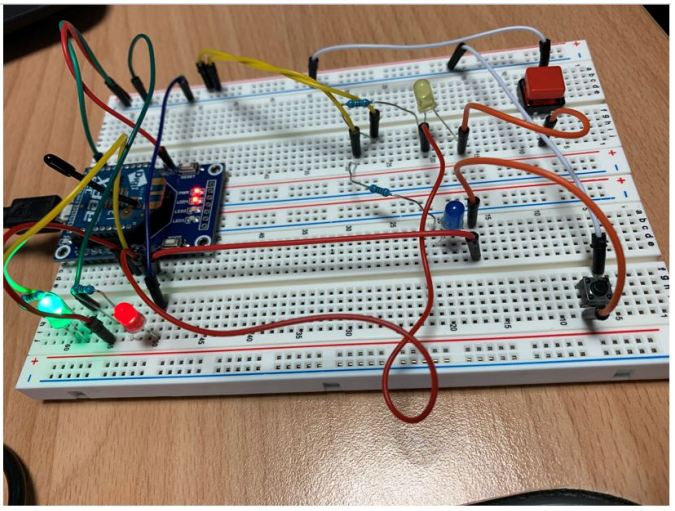
Montage avec bouton



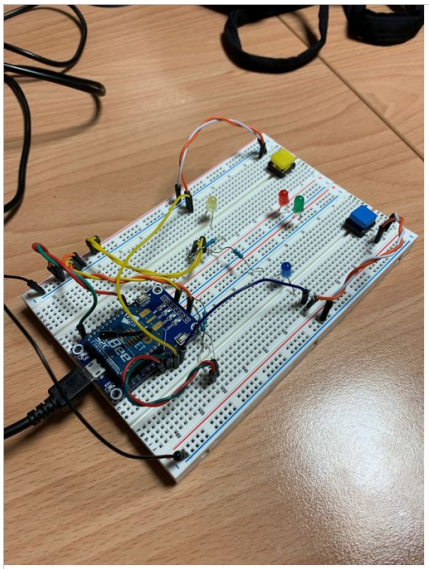


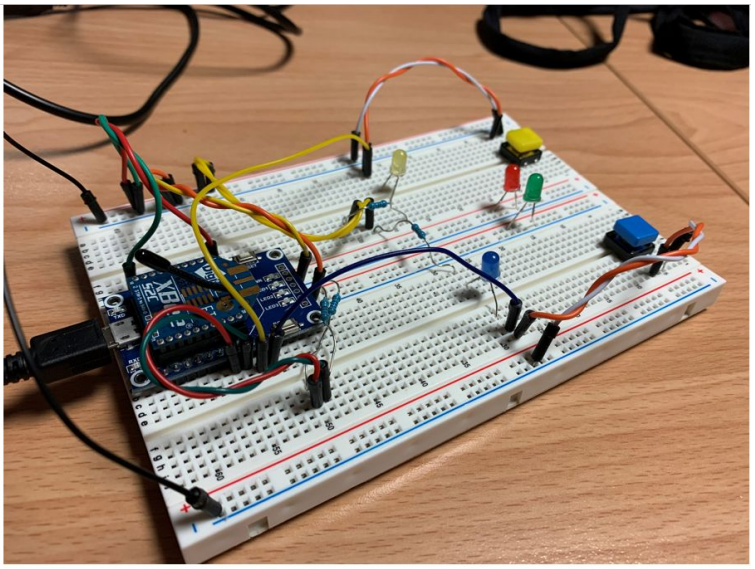
Montage 2 boutons & 4 leds

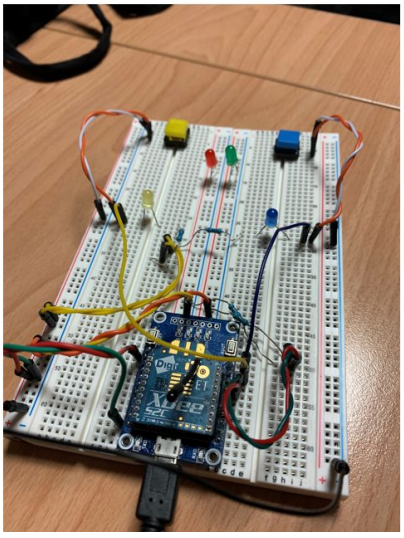




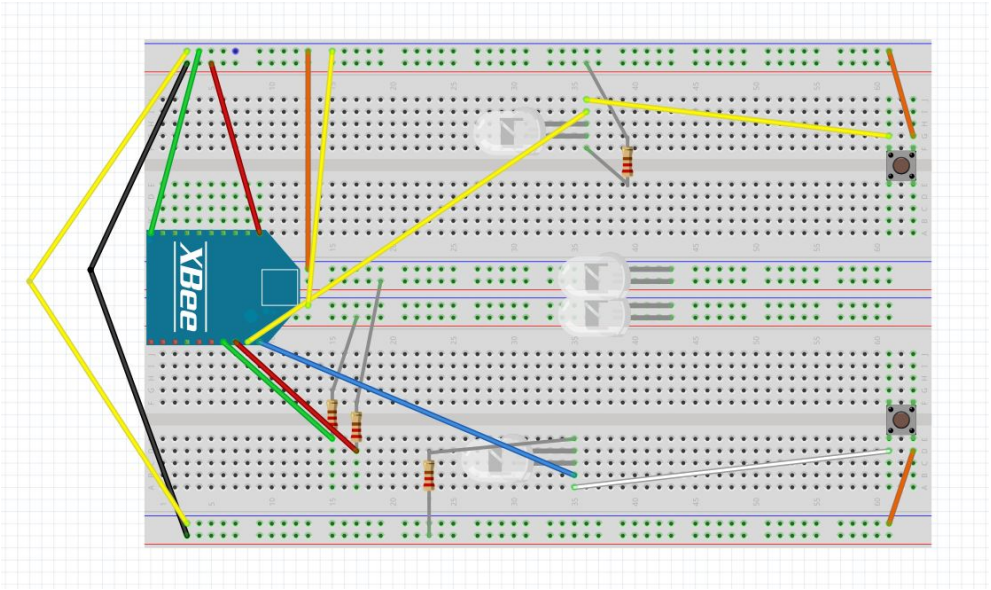
Montage final optimisé







Schéma

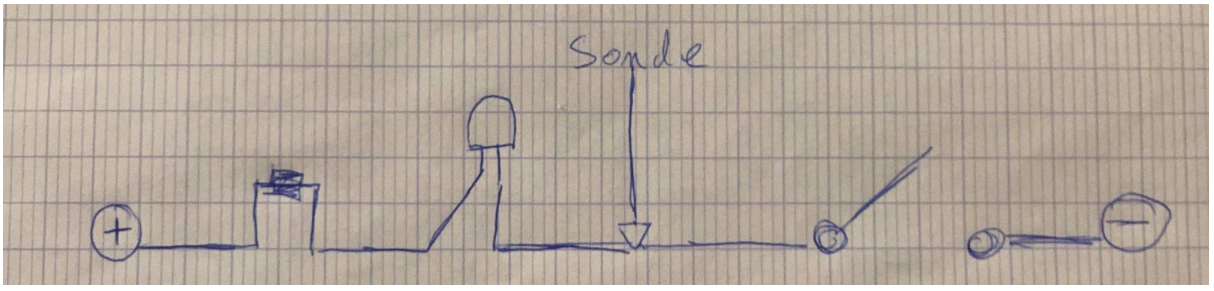


3 - Fonctionnement en détail

Pour envoyer les informations au Coordinateur :

IC FFFF veux dire que à chaque détection de changement sur une parmis toutes les PIN du Xbee on vas envoyer notre Sample au coordinateur.
Dans notre cas à chaque fois qu'on appuie sur un bouton une trame est envoyé mais aussi lorsqu'on le relâche, il a fallu donc côté code trier l'information reçu en doublon.

Schéma simplifié du fonctionnement du bouton :



Le bouton agit comme un coup circuit dans notre maquette, tant qu'il n'est pas actionné le circuit n'est pas connecté à la masse, de ce fait la led reste off.
Cependant à partir du moment où l'on presse le bouton, le circuit est relié à la masse et donc la led s'allume, la sonde détecte donc un changement dans notre circuit et donc grâce à la configuration vu précédemment le xbee envoie un Sample contenant l'état de tous nos PIN.

4 - Développement serveur Node.JS

Documentation

Première étape afin de bien prendre en main le "squelette" de code fourni dans le projet fourni par l'enseignant, c'est la documentation sur le Xbee-api, la librairie utilisée par la partie "/socket" du projet.

Tests

Ensuite, nous avons testé de voir ce que donne le code en lisant les frames reçues sur le port série branché au Xbee "coordinateur"

Explications de quelques lignes de code :

Ces lignes de code permettent de récupérer les mots de passe qui sont enregistré en base de données et de les stocker dans une variable.
Elles permettent également de vérifier si le bouton pressé est bleu ou jaune et tri l'information afin de savoir si le bouton est pressé ou relâché.

Nous faisons une boucle afin de vérifier si la combinaison entrée actuellement correspond à un des mots de passe enregistré en base.
Enfin si un la combinaison possède une correspondance, une led verte s'allume pendant 3000 ms puis s'éteint afin de laisser la led rouge s'allumer.

```
xbeeAPI.parser.on("data", function (frame) {
  //on new device is joined, register it
  RecoverPassword(function(data){
    passwords=data;
    console.log(passwords);
  });
  if(frame.digitalSamples !== undefined)
  {
    if(frame.digitalSamples['DIO0'] == 0){
      combinationEnteredString += "B";
    }
    if(frame.digitalSamples['DIO1'] == 0){
      combinationEnteredString += "J";
    }
    for(let i = 0; i < passwords.length; i++){
      let password =  passwords[i];
      console.log(password);

      if(!truePass && combinationEnteredString.match(password)){
        truePass = true;

        xbeeAPI.builder.write(enableGreenLight);
        xbeeAPI.builder.write(disableRedLight);

        console.log(combinationEnteredString + " : Combinaison correct !");

        setTimeout(function(){
          xbeeAPI.builder.write(disableGreenLight);
          xbeeAPI.builder.write(enableRedLight);
          combinationEnteredString = "";
          truePass = false;
        },3000);
      }
      else{
        console.log(combinationEnteredString);
      }
    }
    // Si la combinaison entré est la bonne, on allume la led verte pendant 10sec
  }
```

Dans cette partie, la fonction est focalisée afin de faire un appel en base pour récupérer les mots de passe qui y sont stockés.
Les informations récupérées sont alors transcrites en JSON puis un parcours de cette transcription est faites pour récupérer les informations contenues dans "**hydra:member**".

```
function RecoverPassword(callback){
  var options = {
    url: 'https://localhost:8443/access_passes',
    strictSSL: false,
    secureProtocol: 'TLSv1_method'
  }

  request.get(options, function(error, response, body) {
    if (!error) {
      let jsonData = JSON.parse(body);
      let passwordArray = [];
      for(let i = 0; i< jsonData["hydra:member"].length; i++){
        passwordArray.push(jsonData["hydra:member"][i]["pass"]);
      }
      callback(passwordArray);
    }
    else {
      console.log(error);
    }
  });
}

```

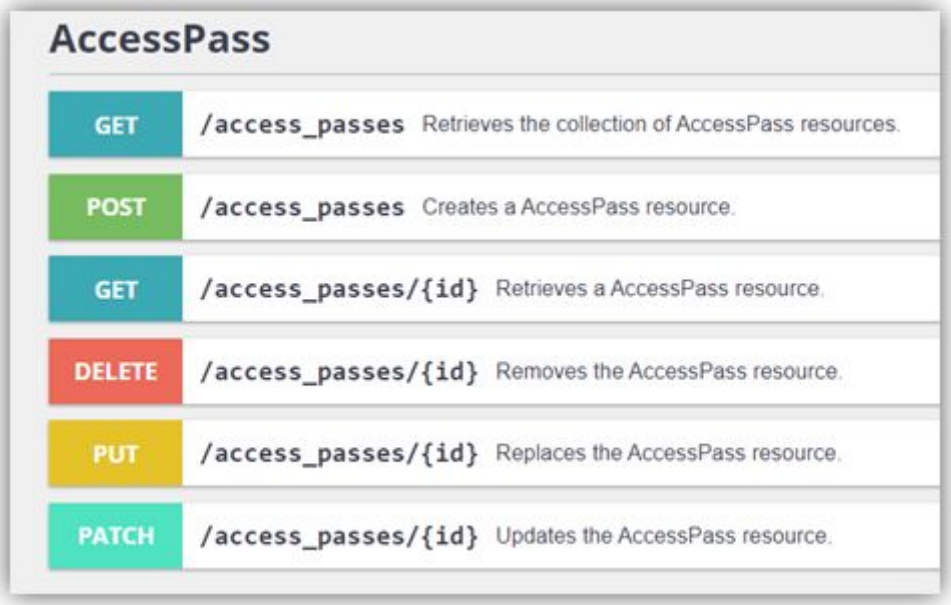
5 - Développement React Native

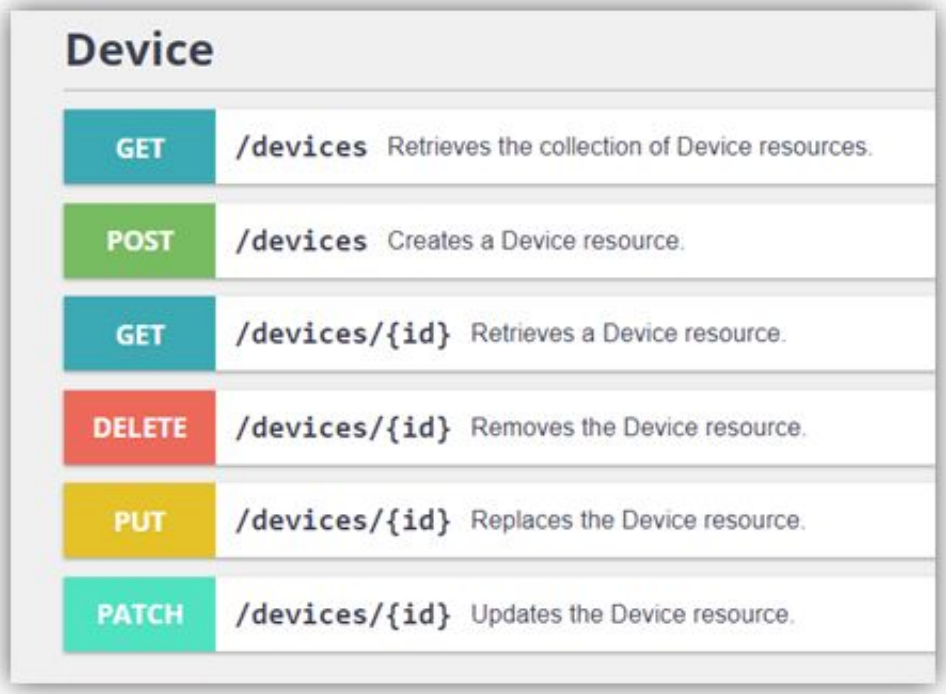
Grâce au composant “Client Generator React Native” proposé par Api-platform (et oui, il permet beaucoup de choses..), nous avons pu générer toute une appli mobile permettant de communiquer avec les différents endpoints de l’API.

En effet, l’application mobile est générée depuis le lien de la documentation de l’API qui représente toutes les routes/endpoints exposée par cette dernière.

Pour notre cas, nous avons générer l’application

API Symfony


AccessPass
GET /access_passes Retrieves the collection of AccessPass resources.
POST /access_passes Creates a AccessPass resource.
GET /access_passes/{id} Retrieves a AccessPass resource.
DELETE /access_passes/{id} Removes the AccessPass resource.
PUT /access_passes/{id} Replaces the AccessPass resource.
PATCH /access_passes/{id} Updates the AccessPass resource.


Device
GET /devices Retrieves the collection of Device resources.
POST /devices Creates a Device resource.
GET /devices/{id} Retrieves a Device resource.
DELETE /devices/{id} Removes the Device resource.
PUT /devices/{id} Replaces the Device resource.
PATCH /devices/{id} Updates the Device resource.

6 - Problèmes rencontrés

Driver / Pilote :

Il se peut que votre que le driver/pilote de votre XBEE ne soit pas installer ou alors il peut posséder une erreure dans l'installation. Vous pourrez trouver le pilote Windows à cette adresse :
https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

API SSL :

L'api REST ("générée" grâce à "api-platform") est accessible seulement et uniquement en HTTPS, avec un certificat "self-signed"/"auto-signé".
Cela a posé des problèmes à tous les niveaux puisque pour chaque client qui a eu à requêter l'API, il fallait augmenter la flexibilité au moment de la requête afin de forcer l'accès à l'endpoint.
Par exemple pour le serveur Node.JS, dans les options de la requête, il a fallu ajouter ce paramètre là:

```
strictSSL: false,
```

Erreur 502 API :

Exécuter la ligne de commande suivante dans le projet :
docker-compose exec php php-fpm -D

Erreur d'éxecution du container docker :

Il faut changer les paramètres "**DB**" du fichier docker-compose-yml qui se trouve à la racine du projet.

```
db:
    image: postgres:12-alpine
    environment:
      - POSTGRES_DB=api
      - POSTGRES_PASSWORD=!ChangeMe!
      - POSTGRES_USER=api-platform
    volumes:
      - db-data:/var/lib/postgresql/data:rw
      # you may use a bind-mounted host
directory instead, so that it is harder to
accidentally remove the volume and lose all
your data!
      # -
./api/docker/db/data:/var/lib/postgresql/dat
a:rw
    ports:
      - target: 5432
        published: 5432
        protocol: tcp
```