GO TRONIC
ROBOTIQUE ET COMPOSANTS ÉLECTRONIQUES

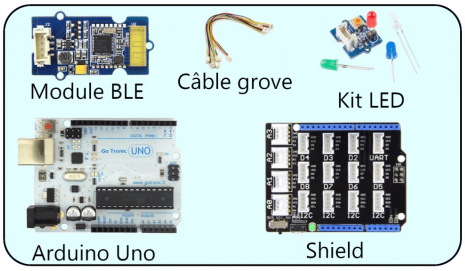


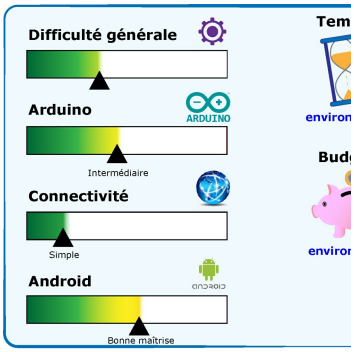
Allumage d’une LED avec un smartphone android en utilisant un module BLE

Aujourd’hui, nous allons allumer une LED grâce à une carte Arduino,
un module BLE (Bluetooth Low Energy) et un téléphone Android.

Matériel pour la réalisation :

- Une carte Arduino Uno
- Un kit LED
- Un module BLE
- Un Base Shield
- Des câbles groves



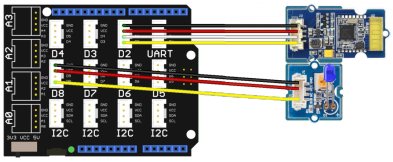


Montage des éléments

Étape 1, Insérez le Shield sur la carte Arduino Uno.

Étape 2, raccordez les modules au Shield de la façon suivante :

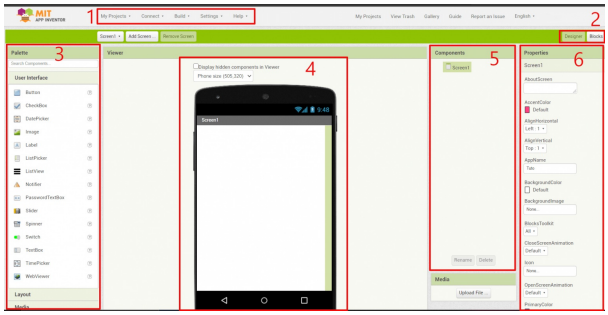
- Le module **BLE** en **D2**
- Le module **LED** en **D8**
- La LED de votre choix sur le module LED (grande patte sur le +)



L’application Android

Nous allons maintenant créer l’application Android avec **App Inventor**. Celle-ci nous permettra de contrôler la LED sur notre smartphone.

Il est possible d’accéder à AppInventor sur navigateur à cette adresse. Une fois celui-ci lancé, on commence un nouveau projet vide afin d’atterrir sur l’interface suivante :



L’écran est composé des parties suivantes :

1.
    1. Le menu de base permettant principalement de gérer les différents projets et de **compiler** l’application en .apk afin de la transférer sur smartphone.
    2. Permet le passage du mode **design** au **codage de l’application** et inversement.
    3. Menu présentant les différents composants à glisser sur notre écran.
    4. Prévisualisation de l’application.
    5. Menu montrant les différents composants présents dans l’application.
    6. Propriétés du composant sélectionné.

Design de l’application

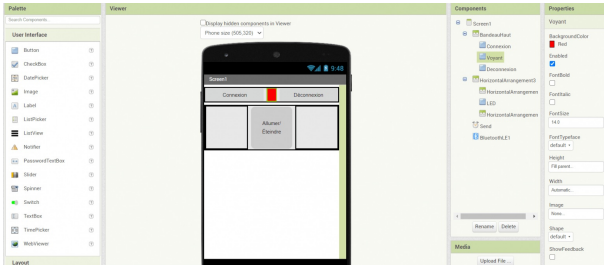
Étape 1, On ajoute dans l’écran un premier **HorizontalArrangement** (présent dans **Layout**) dans lequel on ajoute 3 **boutons** (dans **UserInterface**).

1. Le premier bouton servira à nous connecter au module BLE.
2. Le second sera simplement un indicateur et ne sera pas cliquable.
   Pour cela, on décoche dans ses propriétés la case **ShowFeedback** et on le colore de base en rouge.
3. Enfin le dernier bouton servira à nous déconnecter.

Étape 2, On ajoute un second **HorizontalArrangement** dans lequel on placera un dernier bouton qui servira à allumer et éteindre la LED.

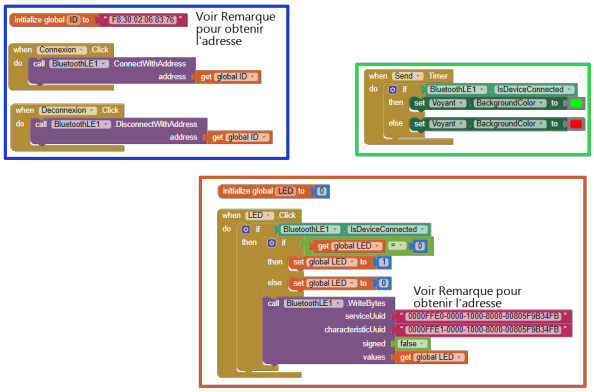
Étape 3, Finalement on ajoute une **Clock** (présent dans **Sensors**) et on ajoute l'extension **BluetoothLE** trouvable **ici** (importer le fichier BluetoothLE.aix).

À la fin de ces étapes, votre application devrait ressembler à quelquechose comme ça :



Code de l'application

Afin d'éditer le code de l'application, on clique sur le bouton "Blocks" en haut à gauche de App Inventor.



Le code est composé de trois parties :

En Bleu, la partie **connexion**, assignant aux boutons prévus à cet effet leurs fonctions permettant de gérer la communication avec le module BLE.

En Vert, la partie qui actualise le voyant de connexion, le passant en vert si la connexion est correctement établie.

En Orange, la partie **transmission**, envoyant l'état du bouton à la carte Arduino. Chaque appui sur le bouton, passe l'état de la variable de 0 à 1 ou inversement et transmet l'information à la carte Arduino. On récupérera cette information dans le programme Arduino afin de gérer l'allumage de la LED.

Remarque : L'adresse du module BLE (ID), le « service UUID » et la « characteristic UUID » sont facilement trouvables avec des applications Android telles que LightBlue par exemple. Pour cela, lancez l'application, connectez vous au module BLE (nom par défaut HMSoft) puis appuyez sur le dernier élément de la liste.

Programme Arduino

Nous allons éditer un programme Arduino permettant de contrôler l'allumage de la LED à l'aide des données envoyées par l'application Android précédemment créée :

- La LED est **allumée** quand la variable reçue vaut 1.
- La LED est **éteinte** quand la variable reçue vaut 0.

```
 #include <SoftwareSerial.h>   //Bibliothèque permettant la gestion du module BLE
  #define RxD 2
  #define TxD 3

  SoftwareSerial BLE(RxD,TxD); // Déclare le module BLE branché en 2 et 3

  void setup()
  {
    Serial.begin(9600);
    pinMode(RxD, INPUT);
    pinMode(TxD, OUTPUT);
    setupBleConnection();
    pinMode(8, OUTPUT);

  }
void loop()
   {
int LED;
      if(BLE.available()){//vérifie si des données ont été envoyées depuis le module BLE
        LED = BLE.read(); //Permet de lire les données reçues
      }
if (LED == 1) {digitalWrite(8,HIGH);} // La LED est allumée quand on reçoit un 1
if (LED == 0) {digitalWrite(8,LOW);} // La LED est éteinte quand on reçoit un 0
}

  void setupBleConnection() // Fonction paramétrant le module BLE
  {
    BLE.begin(9600); //
    BLE.print("AT+CLEAR");
    BLE.print("AT+ROLE1"); //Configure le module BLE en tant que maître
    BLE.print("AT+SAVE1");
  }
```

Mise en marche

Une fois le programme téléversé, lancez l'application sur votre smartphone et connectez vous sur le module BLE à l'aide du bouton prévu.

Une fois la connexion établie, le voyant vert s'allume.

Tout est prêt ! Maintenant vous contrôlez la LED avec votre smartphone !

Démonstration

Voir la vidéo sur YouTube

Pour aller plus loin ...

Une fois son fonctionnement compris, le module BLE peut être utilisé dans beaucoup d'applications plus complexes.

Par exemple, reprenons un ancien article sur le bras robotique Joy-it.

Dans cet article, un bras robotique est contrôlé par une manette reliée par un fil.

Nous pouvons donc essayer de remplacer cette manette par notre smartphone en connectant notre module BLE.

On refait donc une application avec App Inventor selon le même principe sauf qu'ici nous modifions le design pour que celui-ci ressemble à une manette.

On modifie le programme Arduino du bras robotique en ajoutant le contrôle des différentes informations envoyées par les multiples boutons selon le même principe que pour allumer la LED.

Pour les personnes intéressées, vous pouvez télécharger les fichiers ici.

Quant à la connexion du module BLE, on utilise un câble grove comme celui-ci avec un côté femelle qu'on vient connecter sur la pin 8 pour le Rx(fil blanc) et sur la pin 9 pour le Tx(fil jaune) du shield gravity.

Note : Un conflit entre les bibliothèques « SoftwareSerial » et « Servo » nous oblige à utiliser les bibliothèques « ServoTimer2 » et « AltSoftSerial » facilement trouvables sur le net qui sont compatibles entre elles. Cependant, « AltSoftSerial » utilise les pins 8 et 9 pour la connexion de notre module BLE !

Et voilà, on est maintenant capable de contrôler le bras robotique avec notre smartphone !

