Pilotage objets connectés

Table des matières

1 Généralités

Le projet “e-DOLLY”

Le projet est de piloter un chariot Dolly motorisé, équipé d’un appareil photo. La Dolly est équipée d’une carte arduino Uno. Le projet se découpe en plusieurs phases.

1. Gérer le déplacement de la Dolly et le nombre de prises de vues
    - Distance
    - Temps de parcours
    - Sens de déplacement
    - Nombre de photos
2. Dans un deuxième temps
    - Gérer la rotation de l’appareil photo
    - Gérer les roues directrices
    - Gérer les obstacles

Les solutions de pilotage

Dans un premier temps nous avions opté pour un pilotage à l'aide d'un écran LCD équipé de boutons . Mais cette solution est contraignante et manque de souplesse

Nous avons rapidement recherché une solution pour piloter la Dolly à l'aide d'un smartphone .

Nous avons retenu deux outils

1. Appinventor pour un pilotage local via une connexion Bluetooth
2. Blink pour un pilotage à distance via internet

Principe généraux

- Pilotage via APPinventor
  le pilotage s'effectue via une connexion Bluetooth, l'échange des données de pilotage se fait via le port série .
- Pilotage via Blink
  le pilotage s'effectue via une connexion internet . Il est nécessaire d'utiliser un service sur internet . L'échange de données de pilotage se fait via des ports virtuels

Nous avons opté pour APPinventor, dans un premier temps.

APPINVENTOR

Historique de l’application Appinventor

App Inventor pour Android est une application développée par Google. Elle est actuellement entretenue par le Massachusetts Institute of Technology (MIT).

Elle simplifie le développement des applications sous Android et le rend accessible même pour les novices et ceux qui ne sont pas familiers avec les langages de programmation. Elle est basée sur une interface graphique similaire à Scratch et à celle de StarLogo TNG **(en)**. Grâce à son interface entièrement graphique et à l'absence totale de ligne de code, elle est particulièrement adaptée à l'initiation des enfants à la programmation, et ce dès l'école primaire.

Google publie l'application le 15 décembre 2010 et met fin à son activité le 31 décembre 2011. Dès l'été 2011, Google travaille sur un projet similaire Blockly2, développé cette fois en javascript. Depuis le retrait de Google, c'est le centre d'études mobiles au MIT qui gère le support technique de cette application sous le nouveau nom "MIT App Inventor"3.

Connexion au centre éducation MIT

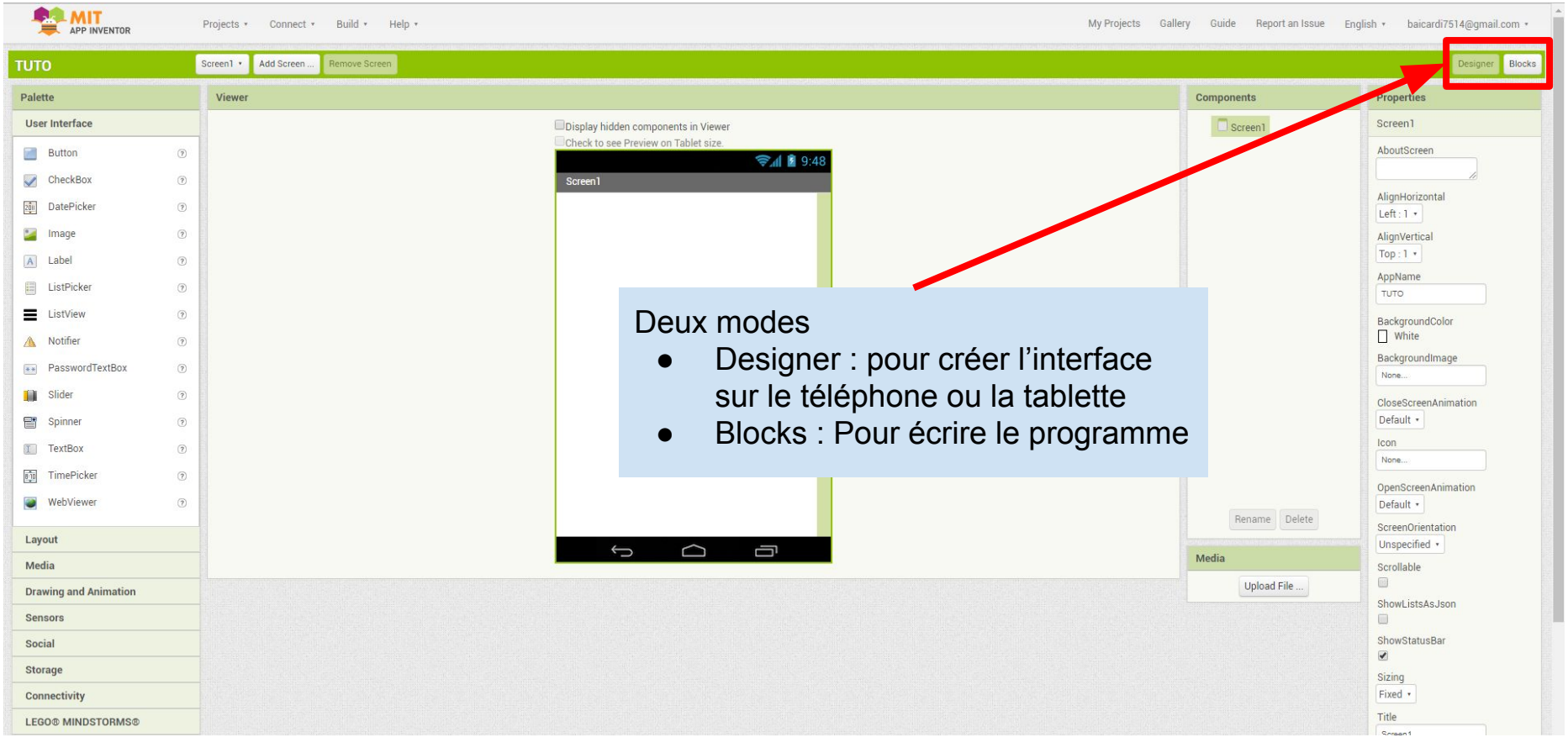
Il faut disposer d’un compte GMAIL

Pour créer une application, allez sur le site : http://ai2.appinventor.mit.edu/

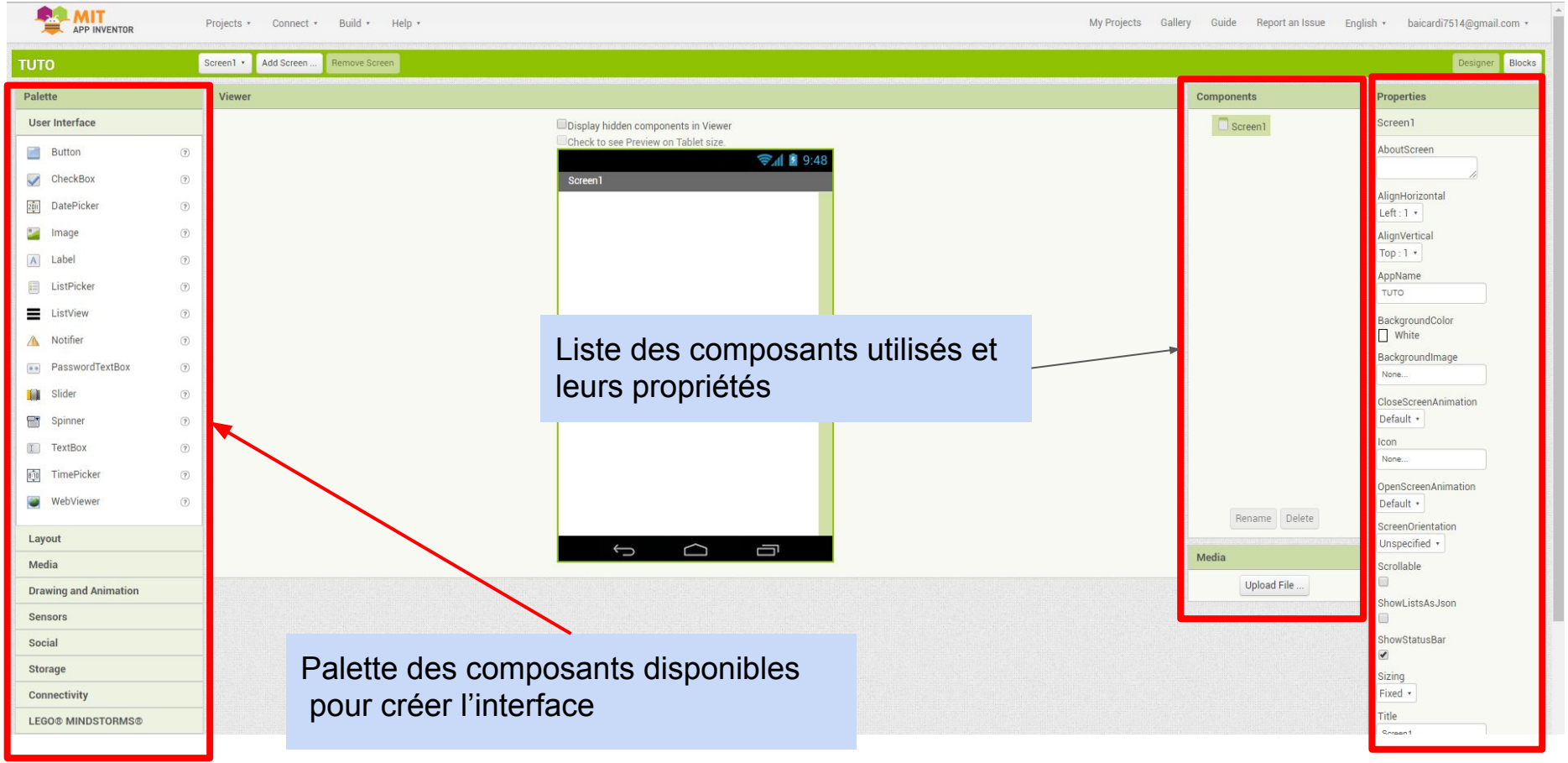
Le développement de l’application se fait entièrement en ligne … rien à installer sur votre PC.

Les projets sont sauvegardés sur le site du MIT.

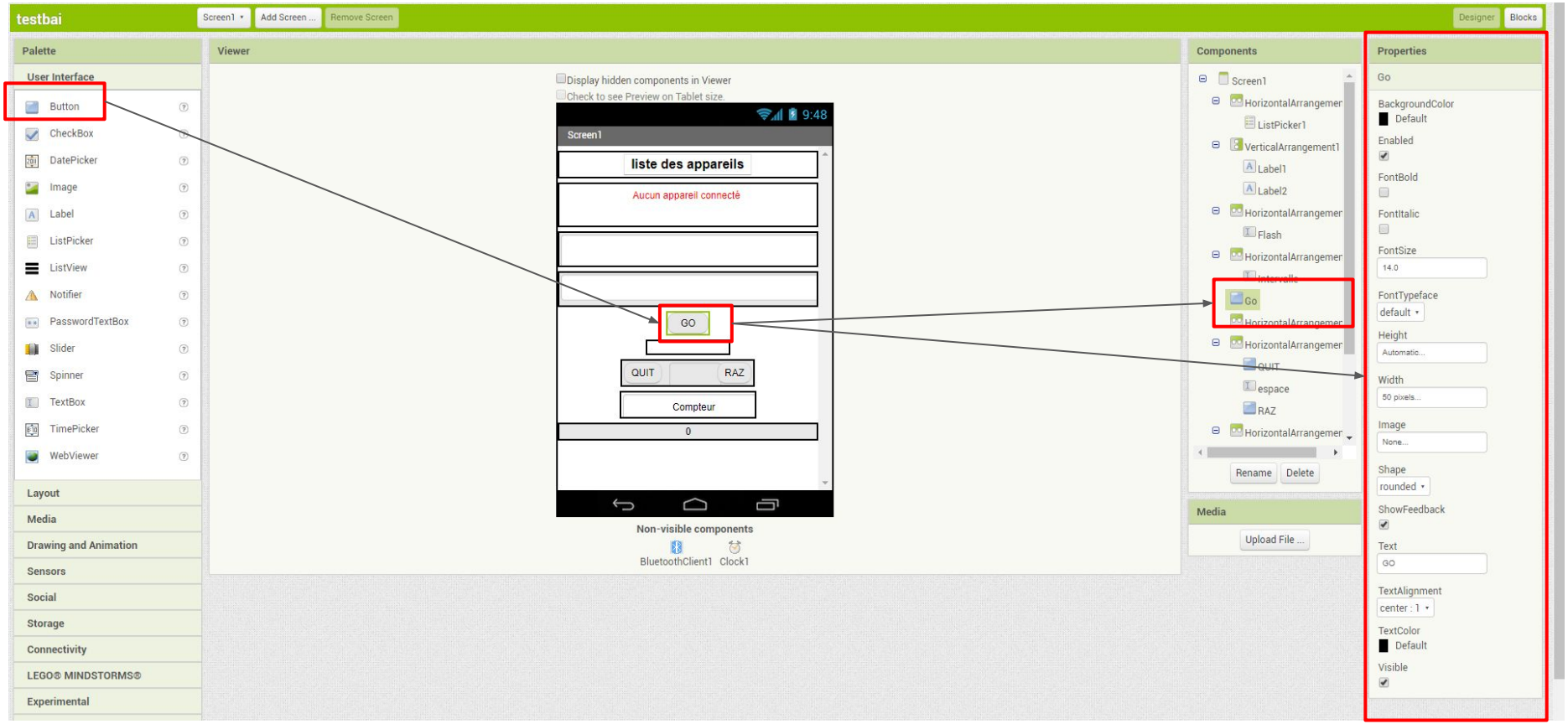
Ecran Principal de l'interface de développement



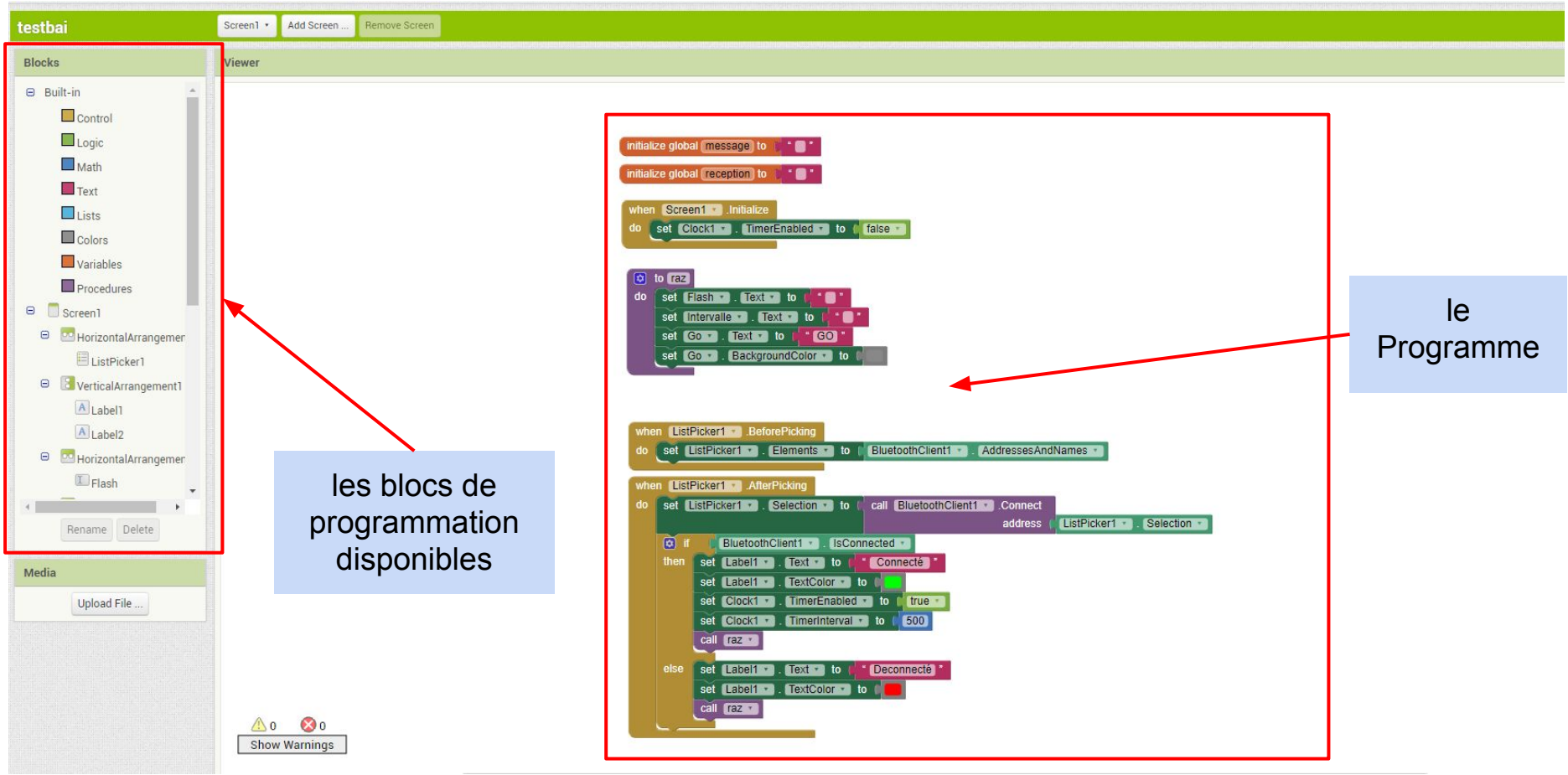
Mode “DESIGNER”



Exemple réalisation d'une interface



Exemple Interface de programmation



les blocs de programmation disponibles

le Programme

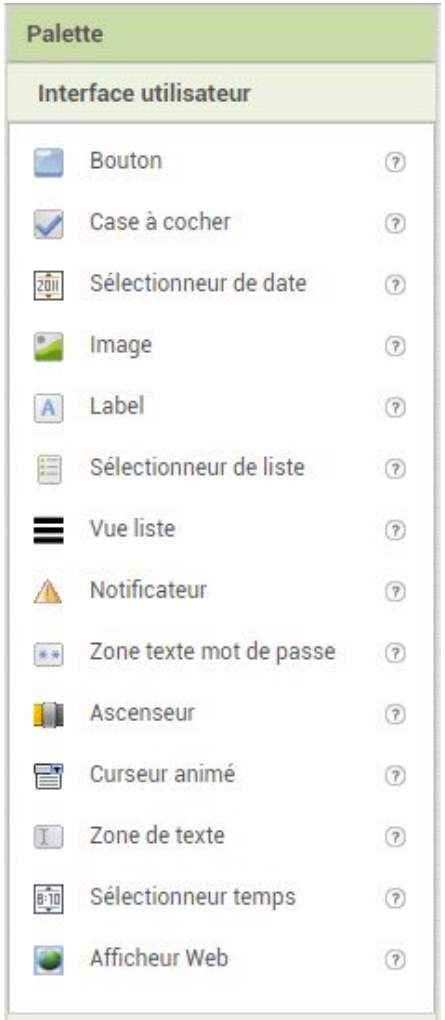
Création de l’interface utilisateur

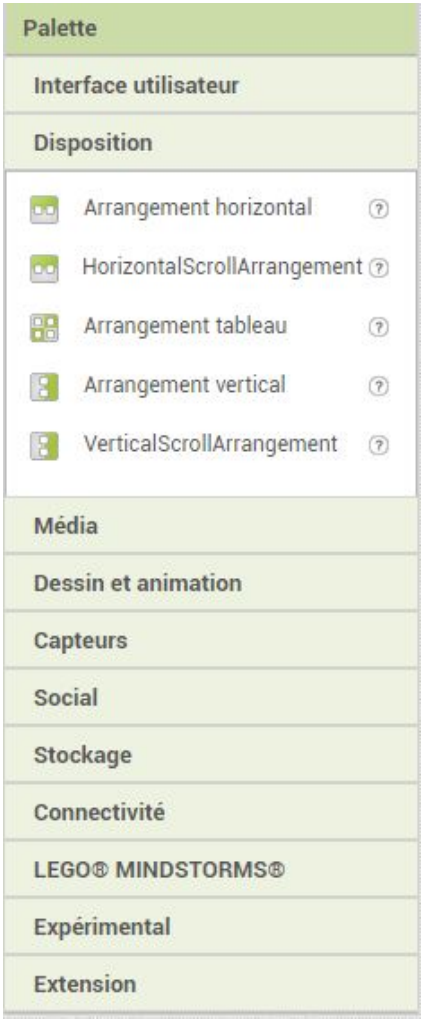
Pour créer l’interface , on dispose d’une large palette de composants sur la gauche de l’écran “bouton”, “checkbox”, “image”, etc...

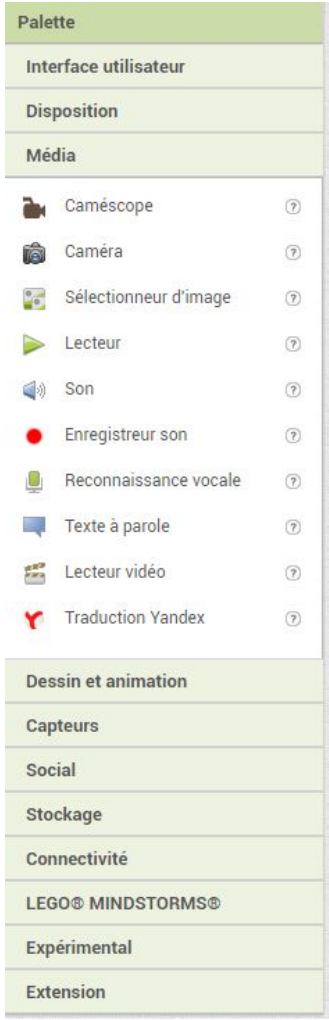
Pour choisir un composant il suffit de cliquer dessus pour le sélectionner et de déposer (glisser déposer) ce dernier sur la zone de travail. Voir la liste des composants disponibles sur les slides suivants.

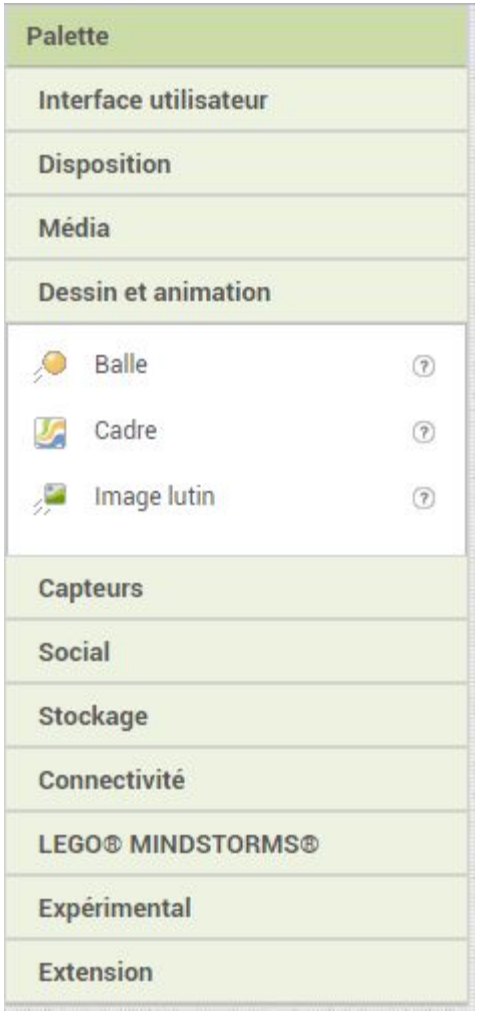
Chaque composant dispose d’un ensemble de propriétés pour le personnaliser sur la droite de l’écran “properties”.

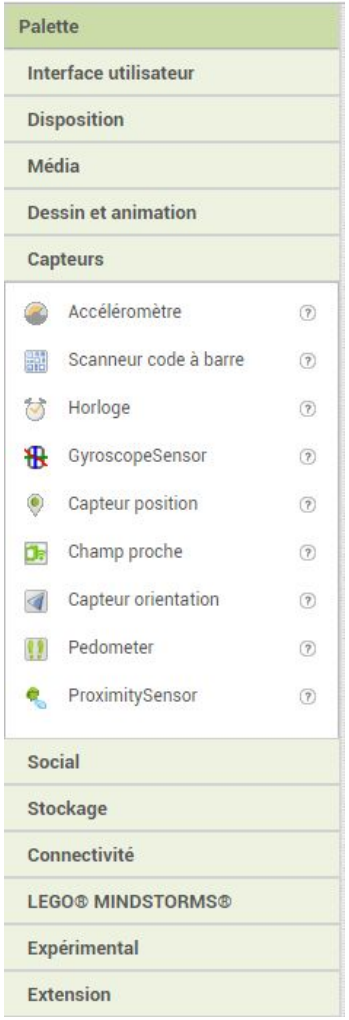
Pour visualiser et tester l’interface il faut connecter un équipement Android via USB ou Wifi ou utiliser l’émulateur Android.

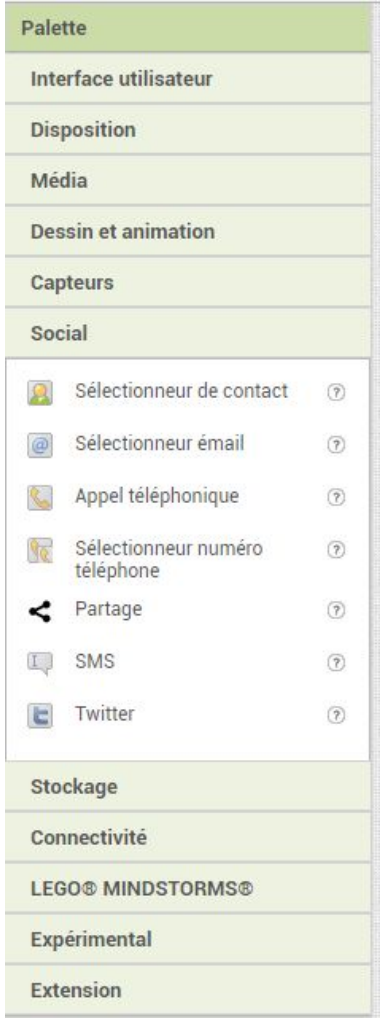

Palette
Interface utilisateur
Bouton
Case à cocher
Sélectionneur de date
Image
Label
Sélectionneur de liste
Vue liste
Notificateur
Zone texte mot de passe
Ascenseur
Curseur animé
Zone de texte
Sélectionneur temps
Afficheur Web

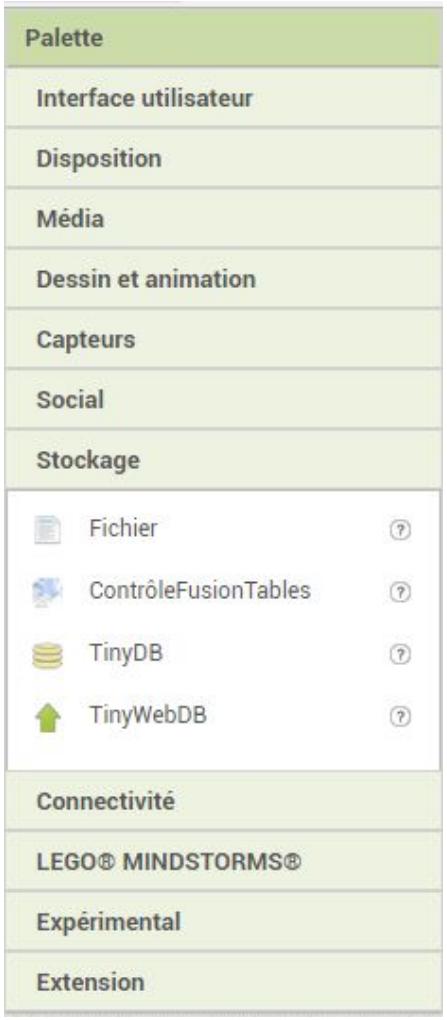

Palette
Interface utilisateur
Disposition
Arrangement horizontal
HorizontalScrollArrangement
Arrangement tableau
Arrangement vertical
VerticalScrollArrangement
Média
Dessin et animation
Capteurs
Social
Stockage
Connectivité
LEGO® MINDSTORMS®
Expérimental
Extension

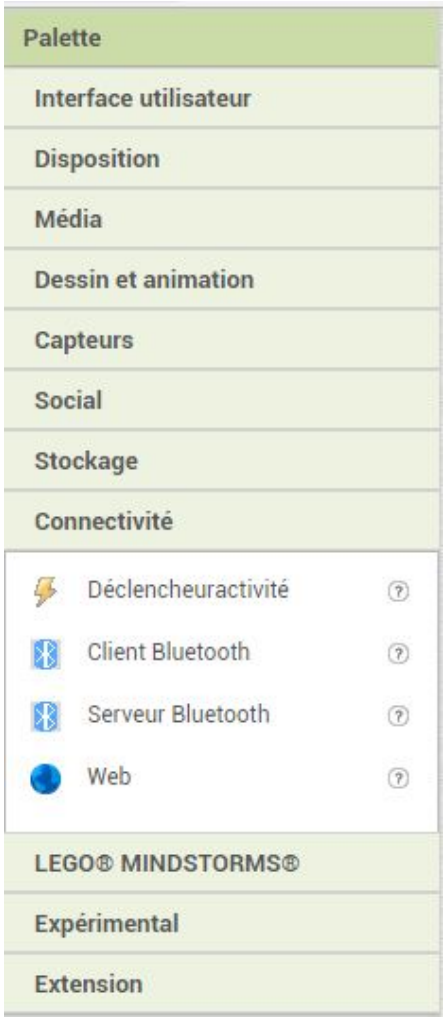

Palette
Interface utilisateur
Disposition
Média
Caméscope
Caméra
Sélectionneur d'image
Lecteur
Son
Enregistreur son
Reconnaissance vocale
Texte à parole
Lecteur vidéo
Traduction Yandex
Dessin et animation
Capteurs
Social
Stockage
Connectivité
LEGO® MINDSTORMS®
Expérimental
Extension

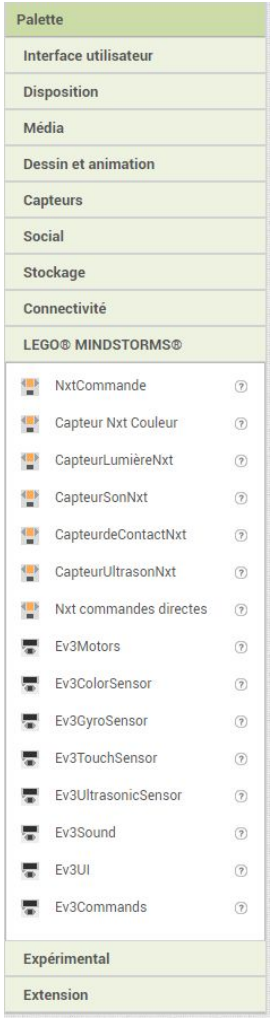

Palette
Interface utilisateur
Disposition
Média
Dessin et animation
Balle
Cadre
Image lutin
Capteurs
Social
Stockage
Connectivité
LEGO® MINDSTORMS®
Expérimental
Extension


Palette
Interface utilisateur
Disposition
Média
Dessin et animation
Capteurs
Accéléromètre
Scanneur code à barre
Horloge
GyroscopeSensor
Capteur position
Champ proche
Capteur orientation
Pedometer
ProximitySensor
Social
Stockage
Connectivité
LEGO® MINDSTORMS®
Expérimental
Extension


Palette
Interface utilisateur
Disposition
Média
Dessin et animation
Capteurs
Social
Sélectionneur de contact
Sélectionneur émail
Appel téléphonique
Sélectionneur numéro téléphone
Partage
SMS
Twitter
Stockage
Connectivité
LEGO® MINDSTORMS®
Expérimental
Extension

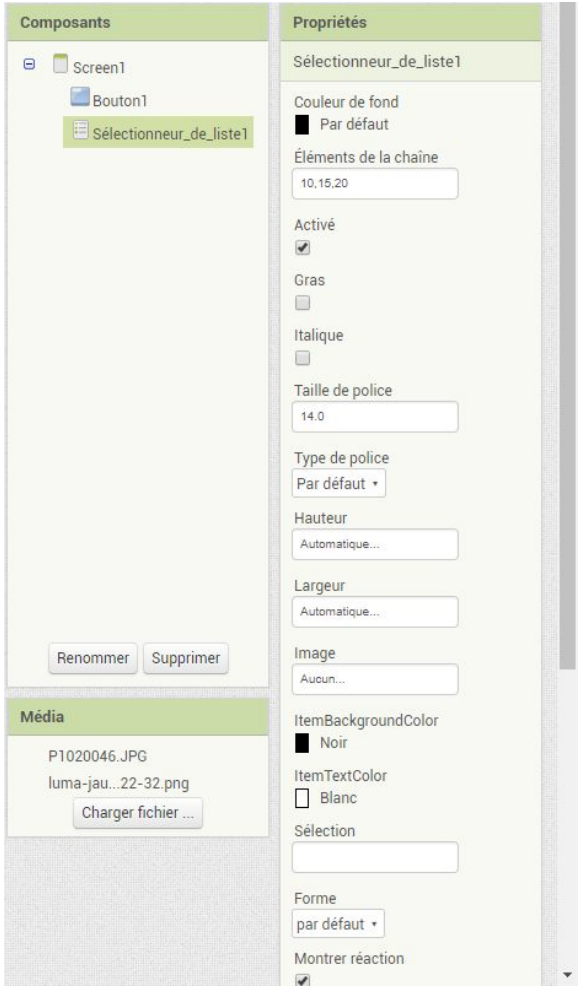

Palette
Interface utilisateur
Disposition
Média
Dessin et animation
Capteurs
Social
Stockage
Fichier
ContrôleFusionTables
TinyDB
TinyWebDB
Connectivité
LEGO® MINDSTORMS®
Expérimental
Extension


Palette
Interface utilisateur
Disposition
Média
Dessin et animation
Capteurs
Social
Stockage
Connectivité
Déclencheuractivité
Client Bluetooth
Serveur Bluetooth
Web
LEGO® MINDSTORMS®
Expérimental
Extension


Palette
Interface utilisateur
Disposition
Média
Dessin et animation
Capteurs
Social
Stockage
Connectivité
LEGO® MINDSTORMS®
NxtCommande
Capteur Nxt Couleur
CapteurLumièreNxt
CapteurSonNxt
CapteurdeContactNxt
CapteurUltrasonNxt
Nxt commandes directes
Ev3Motors
Ev3ColorSensor
Ev3GyroSensor
Ev3TouchSensor
Ev3UltrasonicSensor
Ev3Sound
Ev3UI
Ev3Commands
Expérimental
Extension


Palette
Interface utilisateur
Disposition
Média
Dessin et animation
Capteurs
Social
Stockage
Connectivité
LEGO® MINDSTORMS®
Expérimental
FirebaseDB
Extension


Palette
Interface utilisateur
Disposition
Média
Dessin et animation
Capteurs
Social
Stockage
Connectivité
LEGO® MINDSTORMS®
Expérimental
Extension
Import extension


Composants
Screen1
Bouton1
Sélectionneur_de_liste1
Renommer
Supprimer
Média
P1020046.JPG
luma-jau...22-32.png
Charger fichier ...
Propriétés
Sélectionneur_de_liste1
Couleur de fond
Par défaut
Éléments de la chaîne
10,15,20
Activé
Gras
Italique
Taille de police
14.0
Type de police
Par défaut
Hauteur
Automatique...
Largeur
Automatique...
Image
Aucun...
ItemBackgroundColor
Noir
ItemTextColor
Blanc
Sélection
Forme
par défaut
Montrer réaction

Visualisation et tests de l’application

Pour visualiser et tester l’interface il faut soit :

- Connecter un équipement Android (tablette ou téléphone) via un câble USB ou en Wifi
- Utiliser l’émulateur inclu dans appinventor

Voir dans les slides suivants les procédures de connexion.

Visualisation sur un smartphone ou une tablette

Il est nécessaire de charger sur le smartphone ou la tablette l'application MIT AIE Companion

Sur le PC choisir AI compagnon dans l'onglet connect

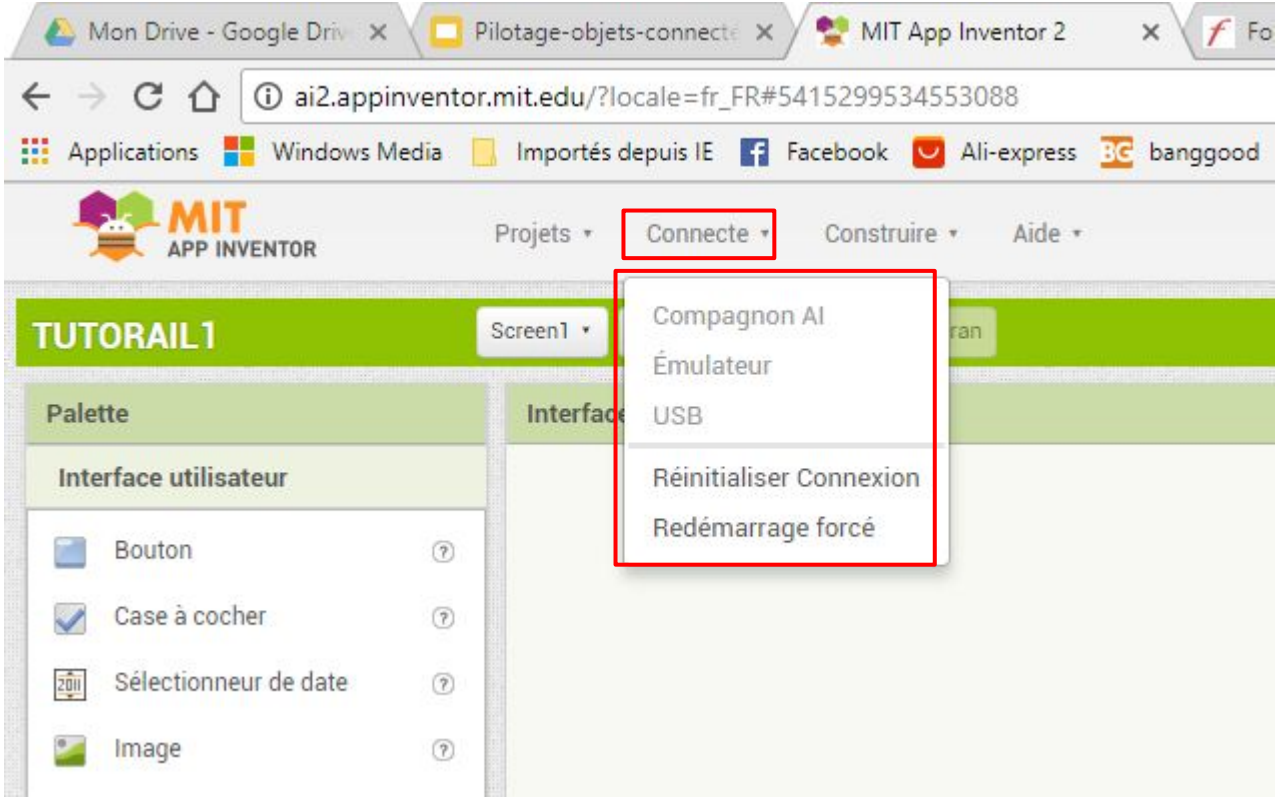
Un QR code s'affiche

Ouvrir l'application AI Compagnon sur le smartphone ou la tablette et flasher le QR code

L'interface s'affiche sur la tablette ou le smartphone

Il est possible de saisir le code au lieu de flasher le QR code

Visualisation et test
sur un smartphone ou une tablette



Visualisation sur un émulateur

Il faut télécharger MIT_App_Inventor pour installer aiStarter sur le PC
http://appinventor.mit.edu/explore/ai2/windows.html

Ensuite on démarre aiStarter sur le PC

Puis on choisit Emulateur dans l'onglet de connexion

Le temps de connexion est relativement long.

Exemple de pilotage d’une lampe à distance

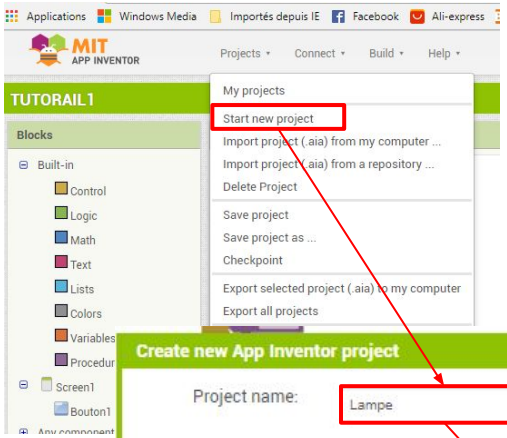
Commençons par un exemple simple:

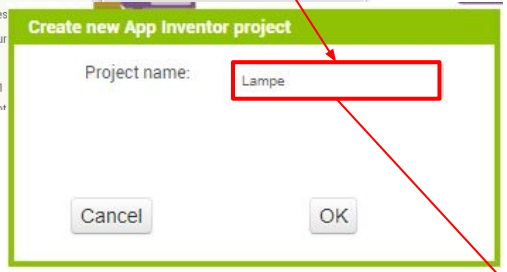
Construisons une application qui va permettre d’allumer et d’éteindre une lampe.

Nous allons procéder en trois phases:

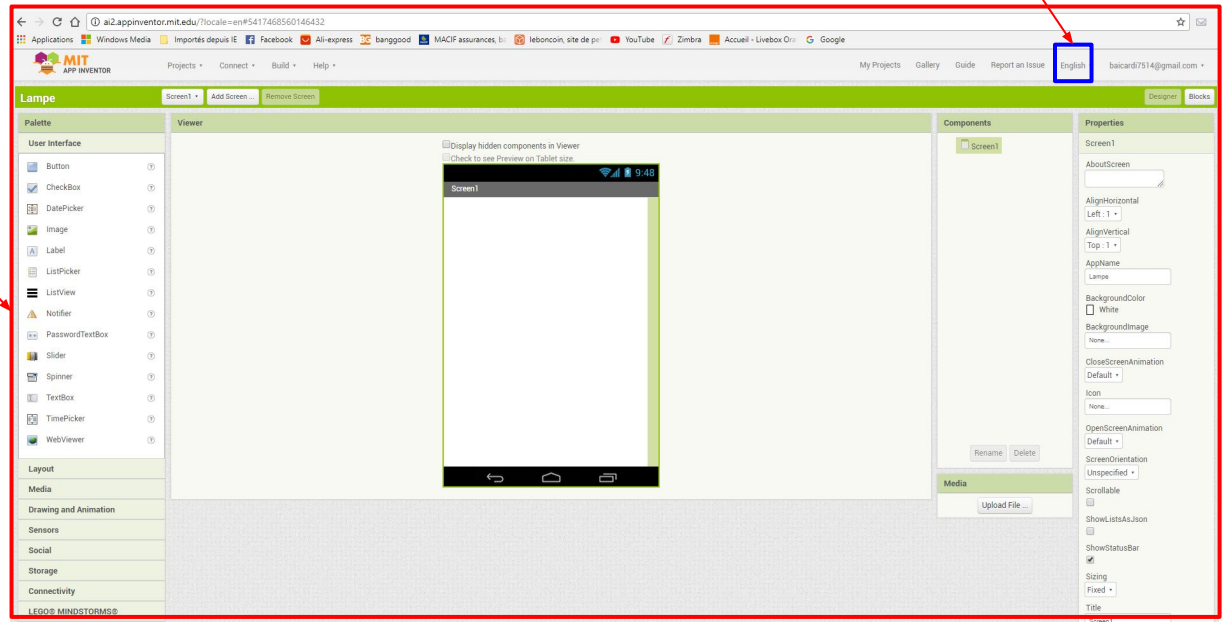
1. Construction de l’interface
2. Ajout de la connexion Bluetooth
3. Réalisation du programme Arduino

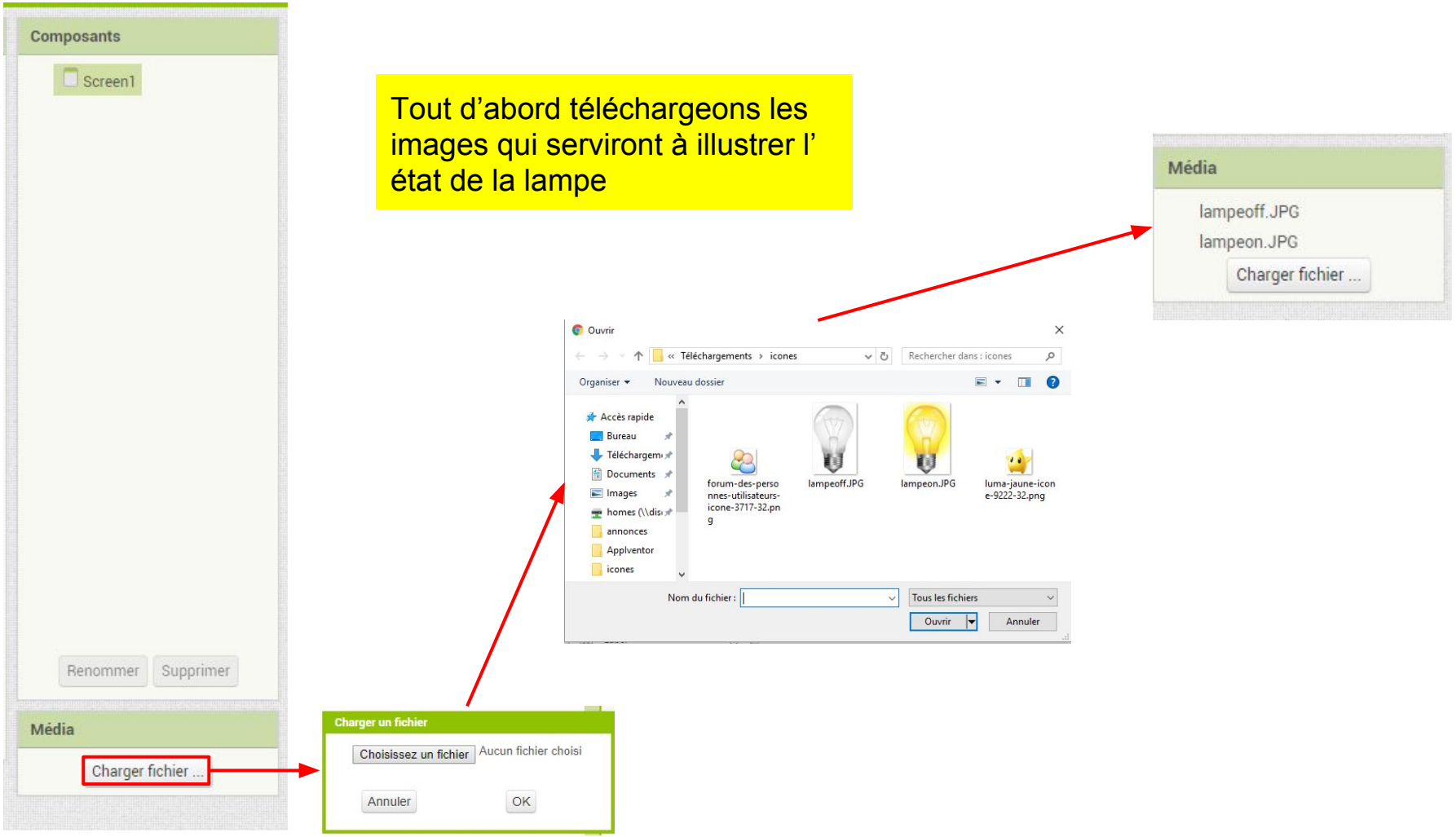
Tout d’abord créons un nouveau projet que nous appellerons “Lampe”

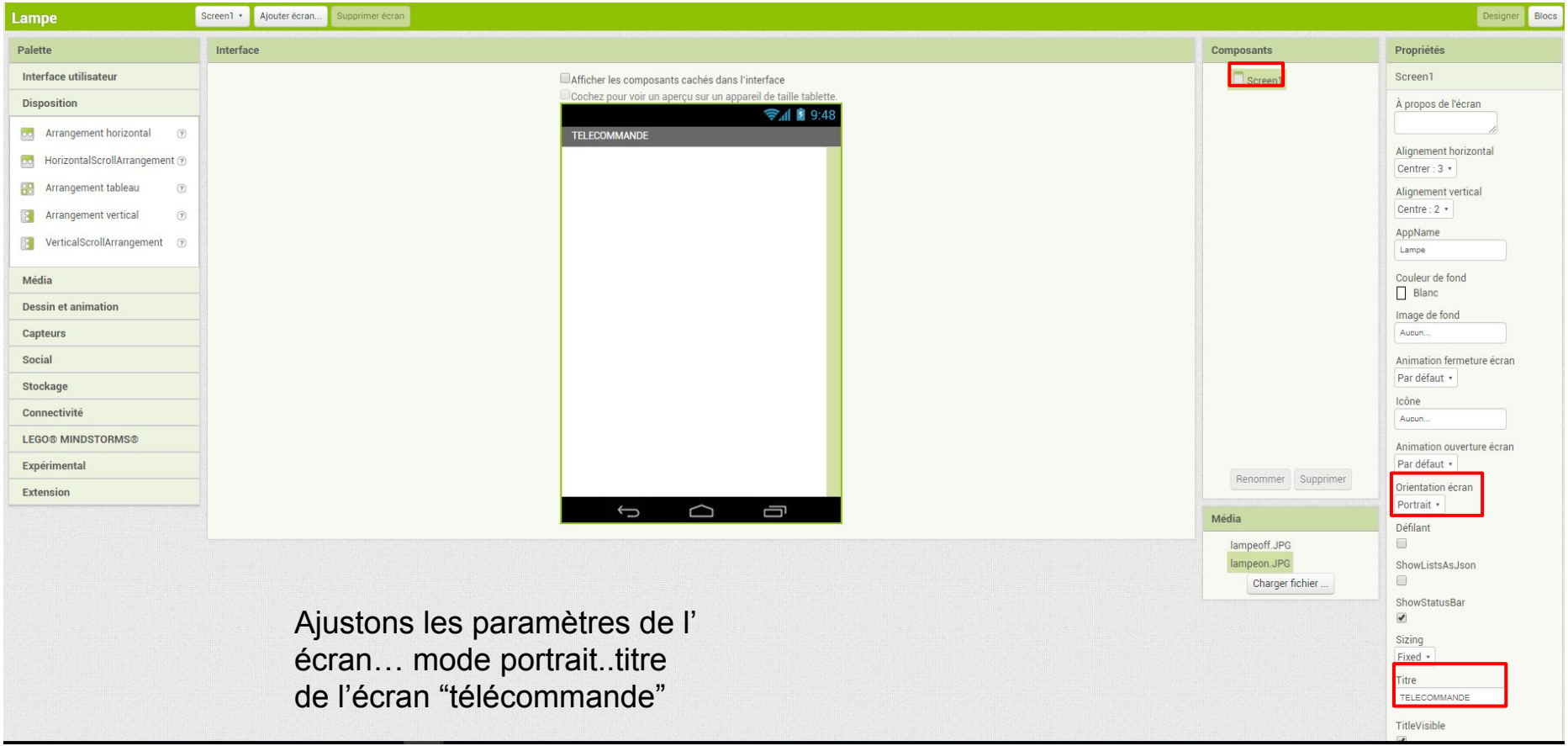




Changer la langue

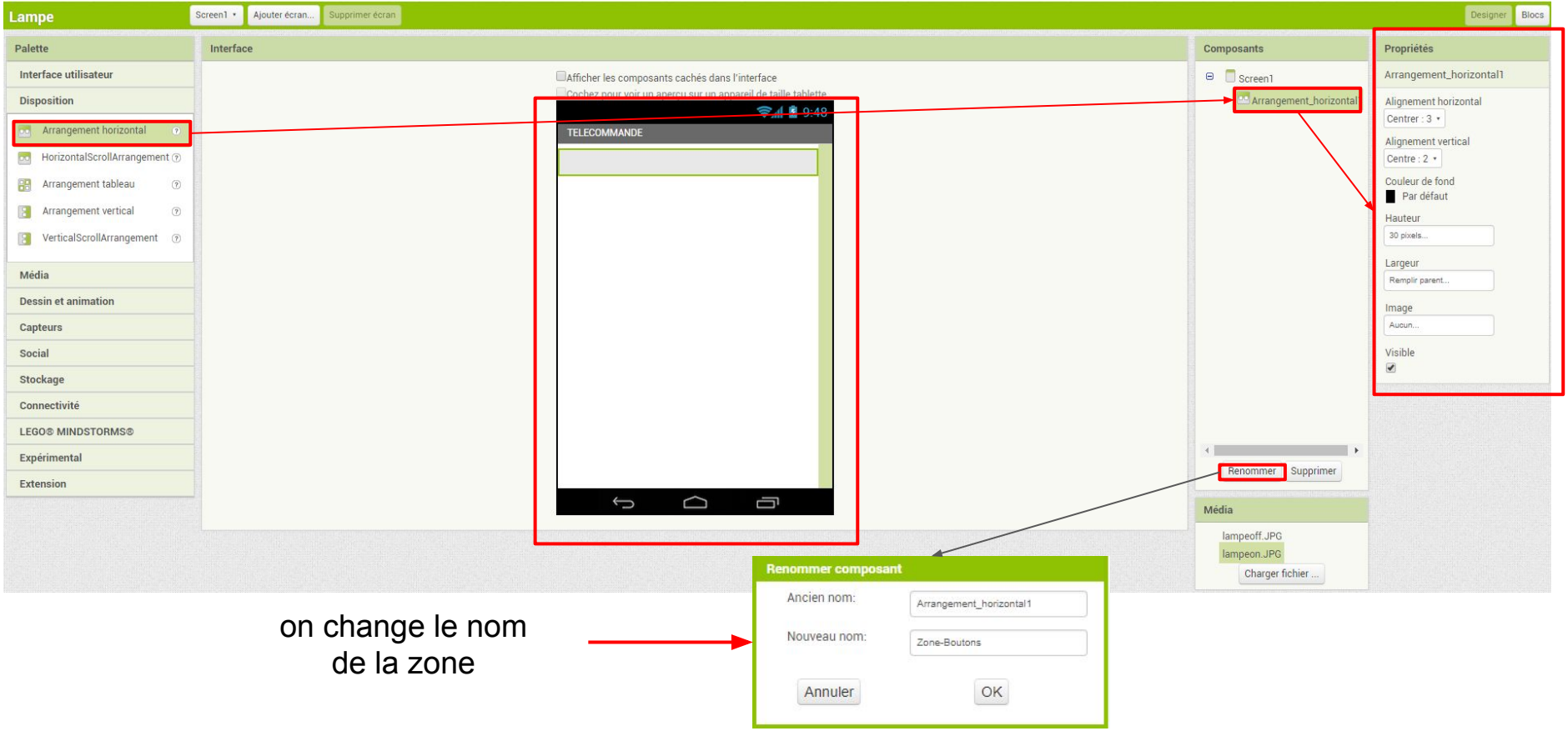



Composants
Screen1
Renommer
Supprimer
Média
Charger fichier ...
Tout d'abord téléchargeons les images qui serviront à illustrer l' état de la lampe
Média
lampeoff.JPG
lampeon.JPG
Charger fichier ...
Ouvrir
Téléchargements › icones
Rechercher dans : icones
Organiser
Nouveau dossier
Accès rapide
Bureau
Documents
Images
annonces
AppInventor
icones
forum-des-personnes-utilisateurs-icone-3717-32.png
lampeoff.JPG
lampeon.JPG
luma-jaune-icone-9222-32.png
Nom du fichier :
Tous les fichiers
Ouvrir
Annuler
Charger un fichier
Choisissez un fichier
Aucun fichier choisi
Annuler
OK



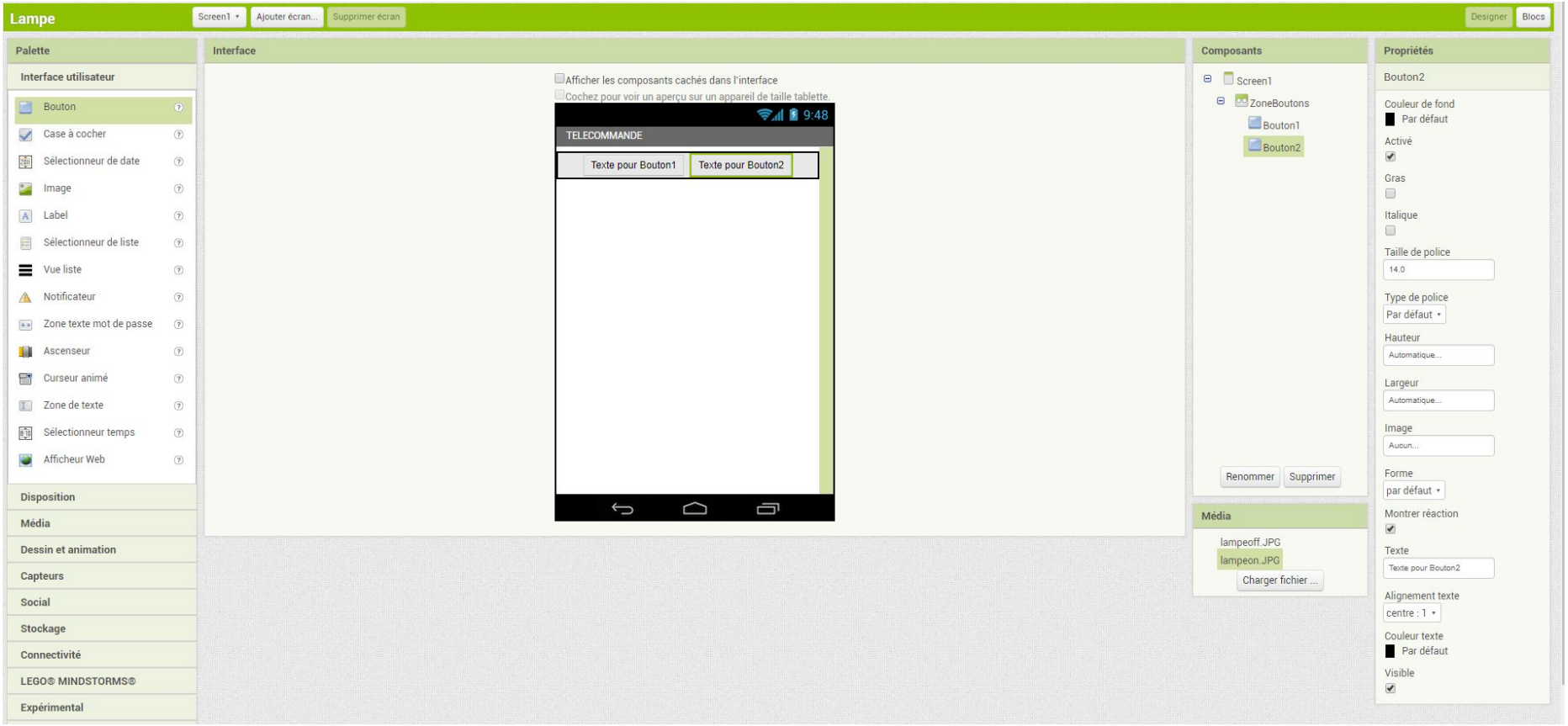
Ajustons les paramètres de l'écran… mode portrait..titre de l'écran "télécommande"

créons une zone qui contiendra les boutons côte à côte

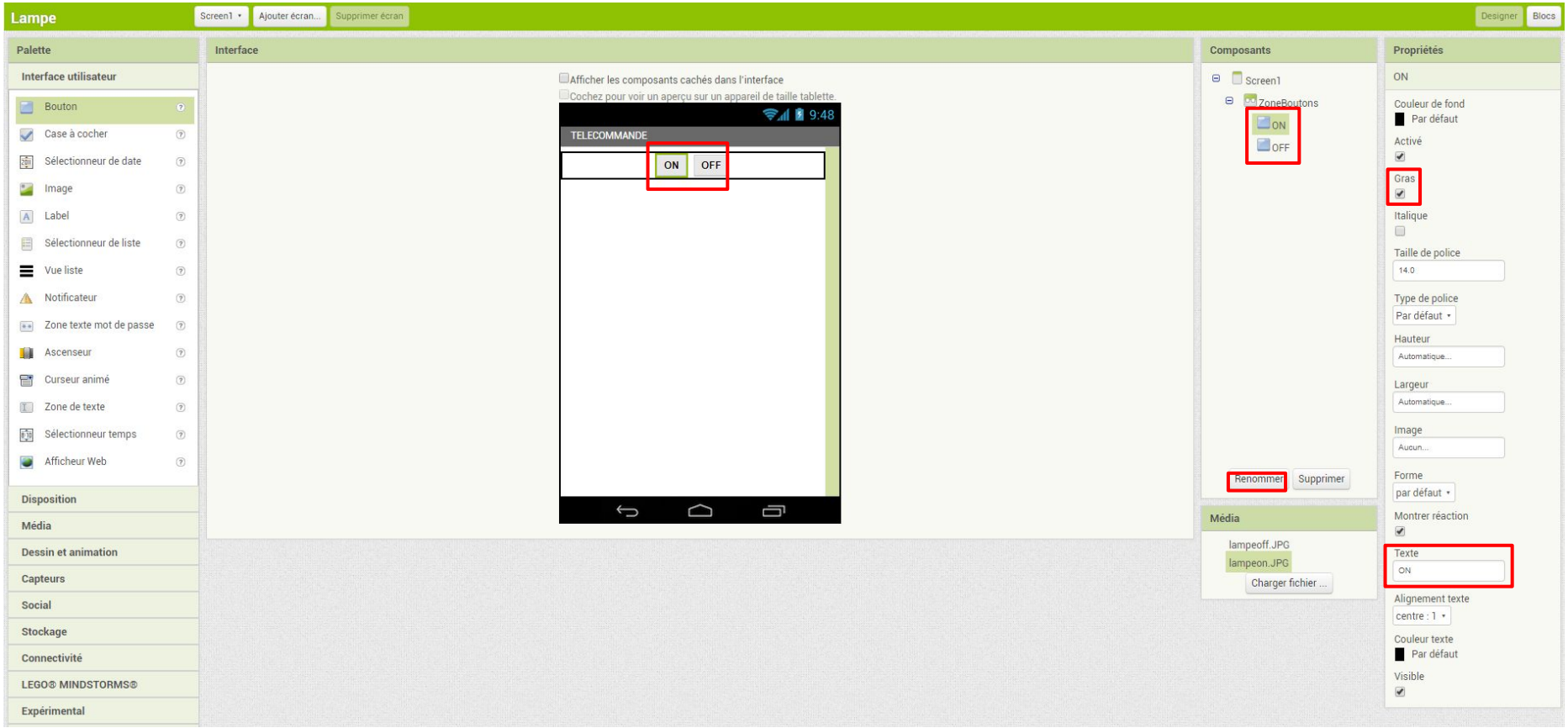


on change le nom de la zone

Insertion de deux boutons dans la zone boutons



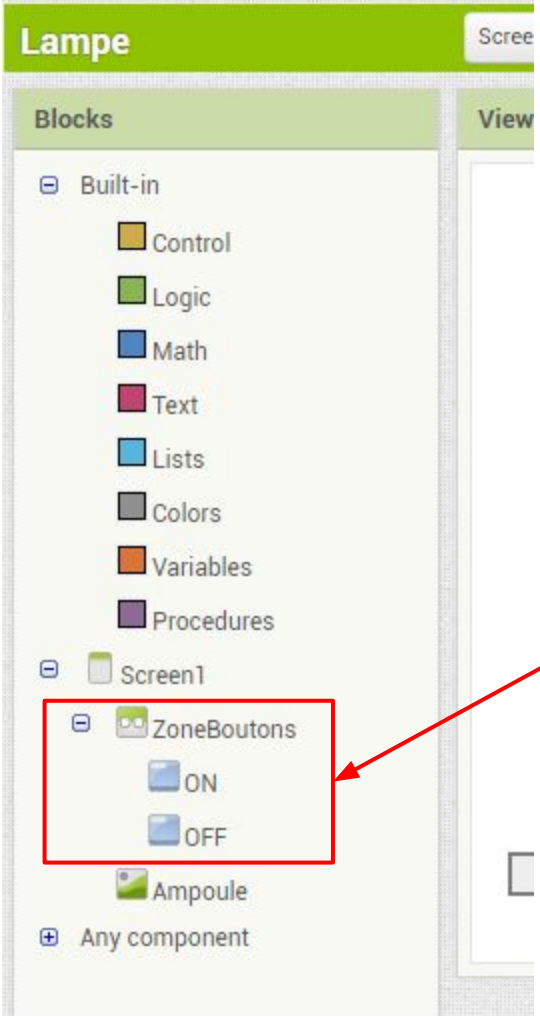
On renomme les boutons et on leur donne des propriétés



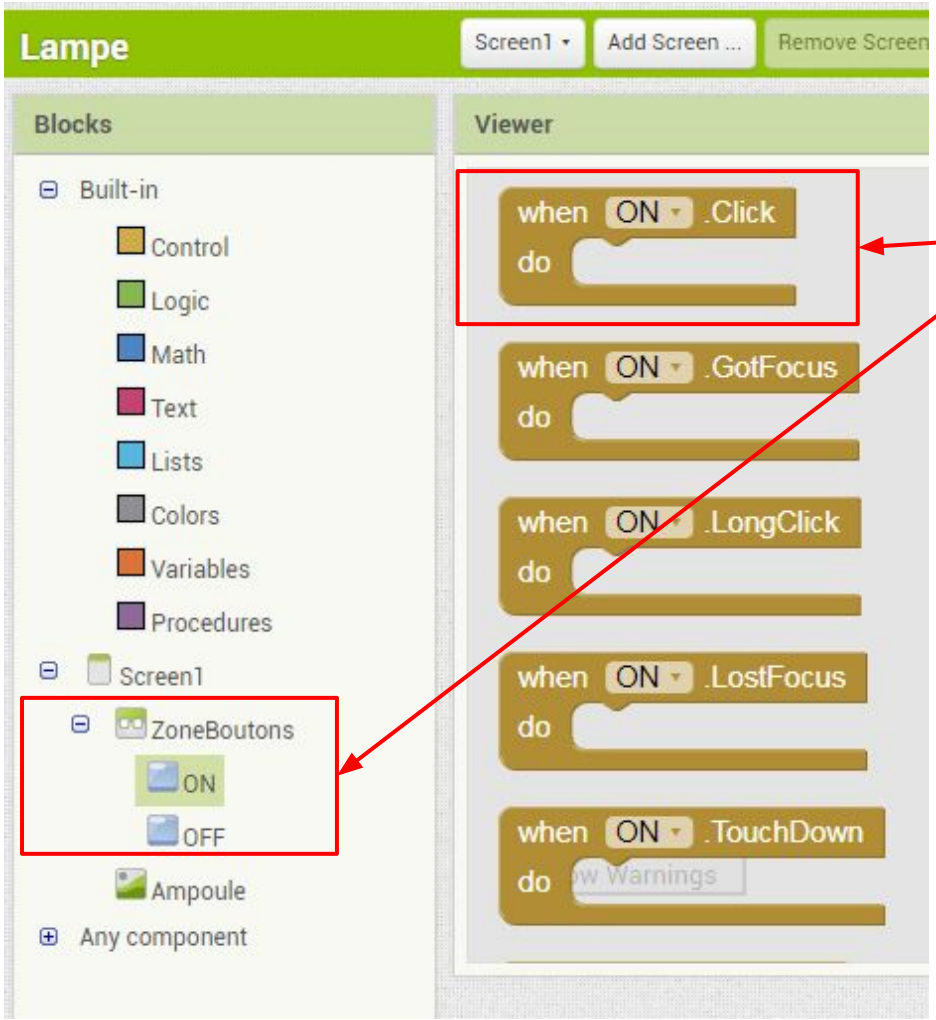
Voyons maintenant pour le code

Il s’agit pour l'instant d’afficher l’ampoule allumée quand on appuie sur “ON” et afficher l’ampoule éteinte quand on appuie sur “OFF”

on va donc se mettre sur l’interface de programmation en cliquant sur l’onglet “block”



Sur la gauche on voit apparaître notre zone avec les deux boutons

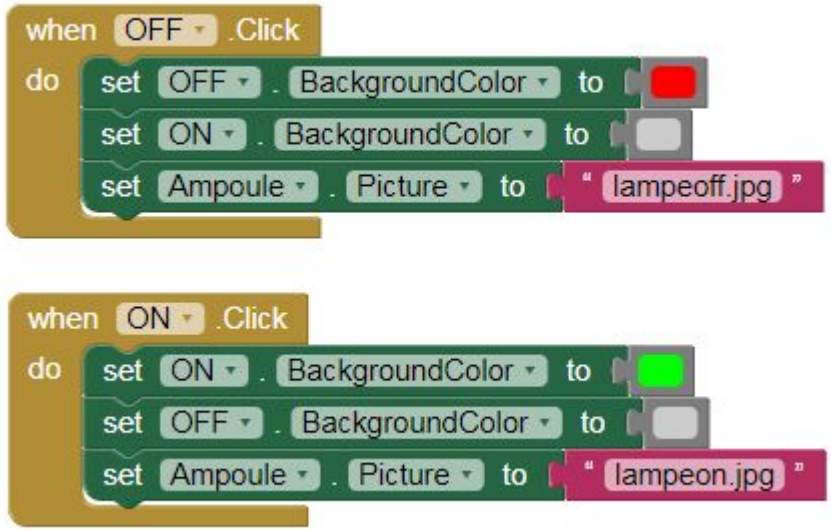


lorsque l'on clique sur le bouton "ON"
un ensemble de blocks de programmation s'affiche

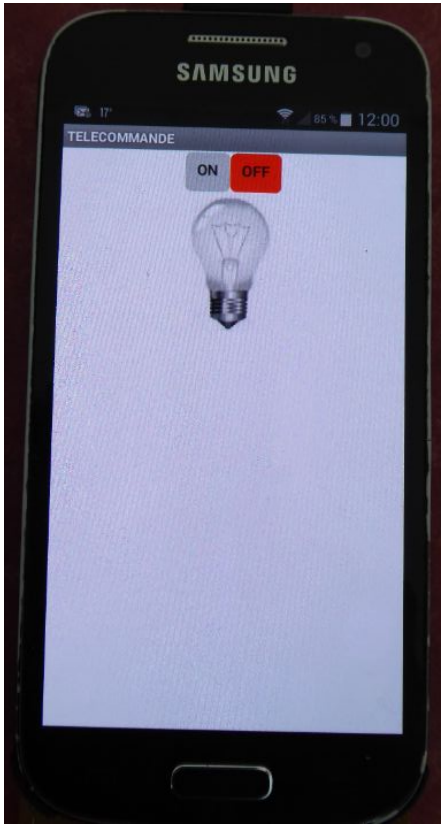
Dans notre cas,nous allons choisir le premier block

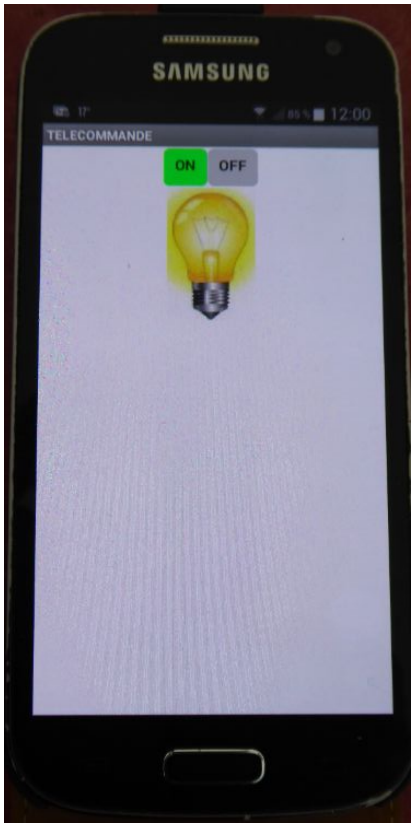
ensuite on descend avec l'ascenseur pour chercher les instructions nécessaires à notre programme .
par exemple dans notre programme on souhaite modifier la couleur de fond du bouton quand on clique dessus

Voici le programme complet qui permet de changer la couleur de fond du bouton et afficher l'image correspondante à l'état de la lampe



Et voilà ce que ça donne la
partie interface est terminée

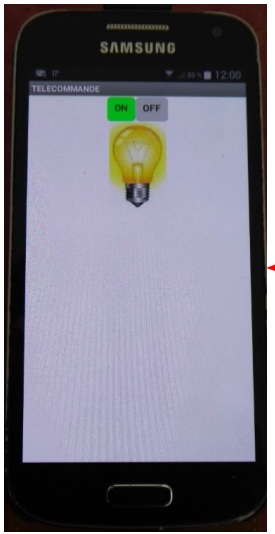




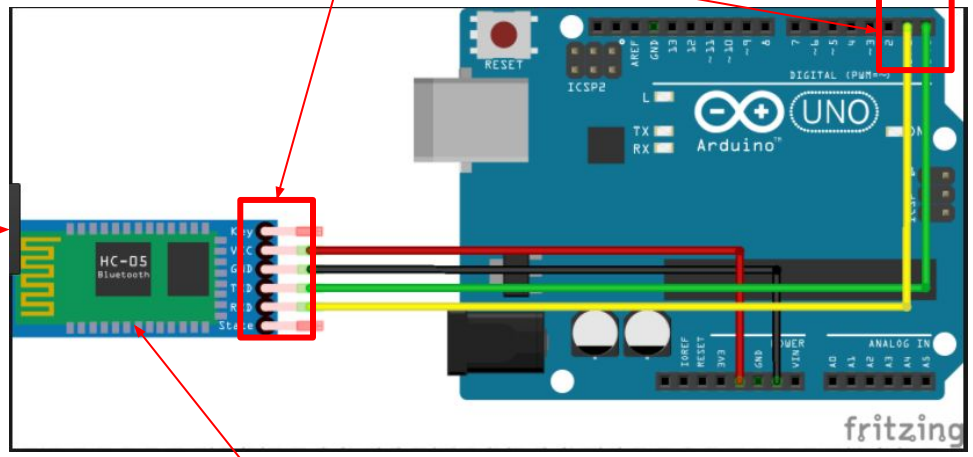
Nous allons maintenant nous occuper de la partie communication Bluetooth avec la carte Arduino

le principe est assez simple , la communication via bluetooth est similaire à une communication sur le port série de la carte arduino … attention pour éviter d'utiliser les Port RX/TX par défaut une solution est proposée plus loin dans les slides suivants

il faut en premier lieu appairer le smartphone et la carte arduino

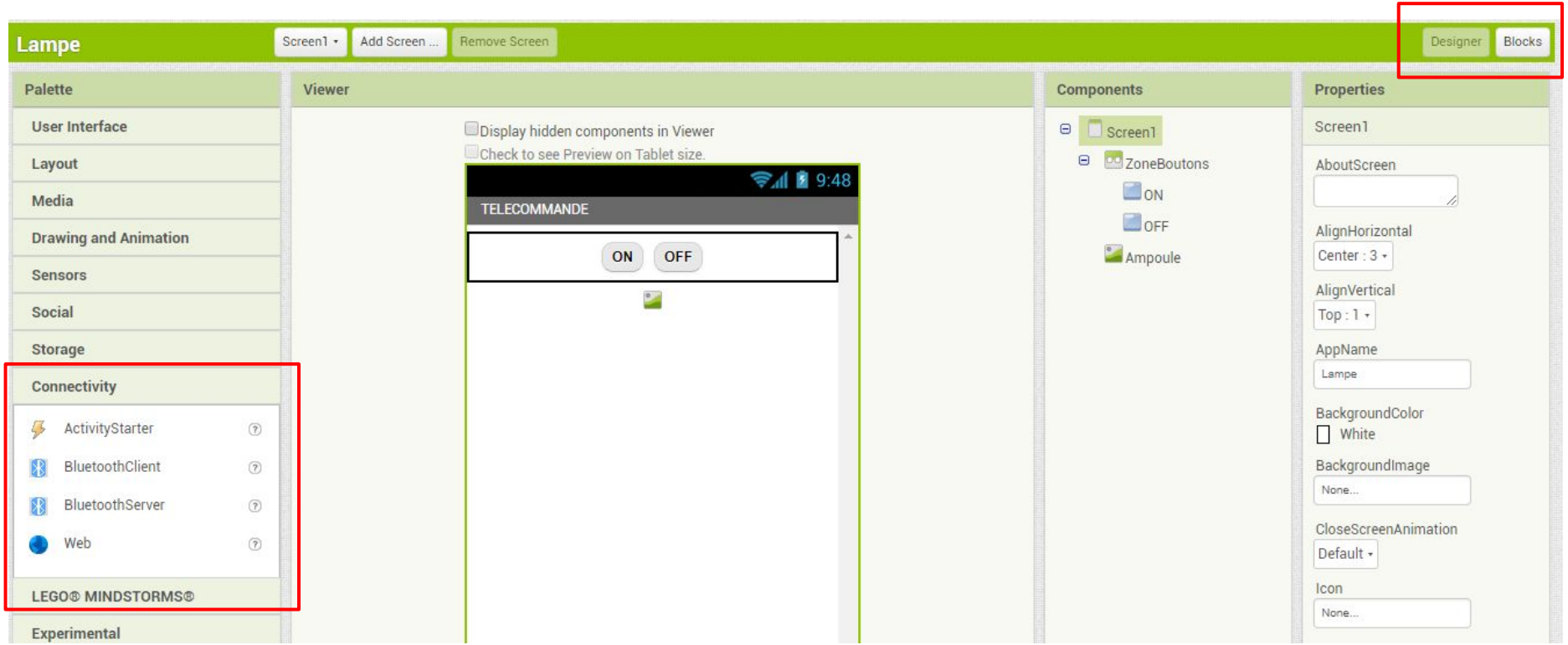


connexion Rx sur Tx et Tx sur Rx



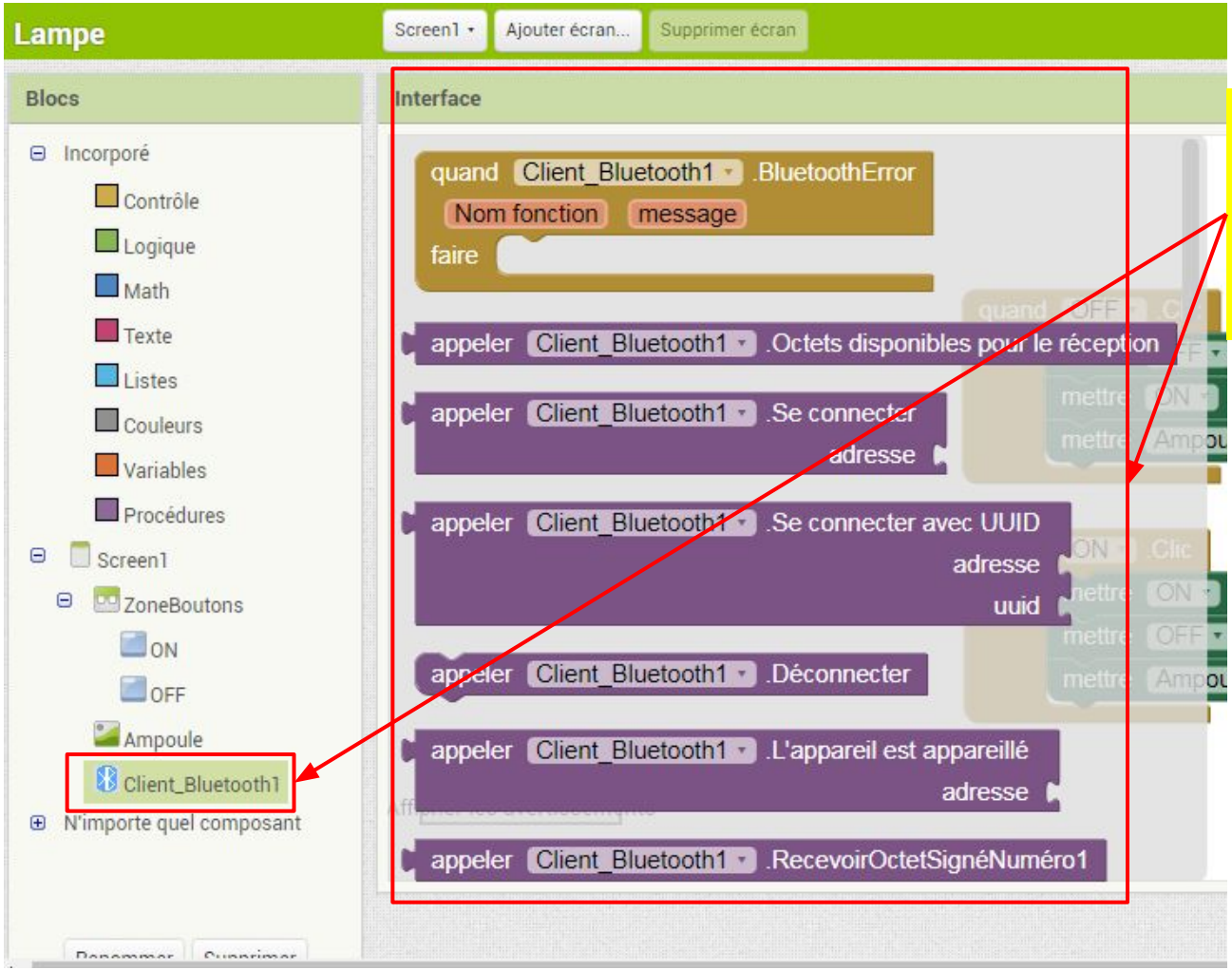
Composant bluetooth

Pour cela ,retournons sur l'onglet "designer" et ouvrons dans la palette le menu "Connectivité"



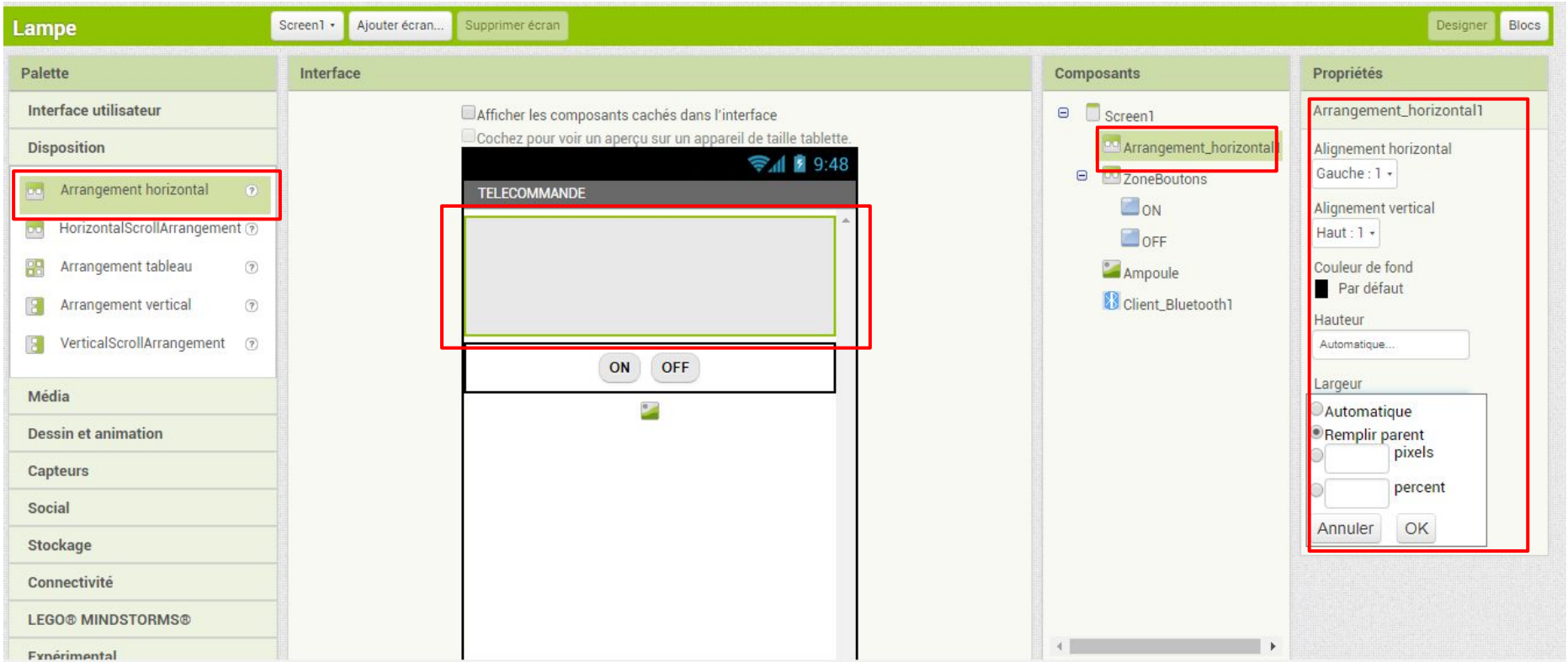
Tirer le composant “Client bluetooth” sur l’écran

le composant vient se placer sous l’écran car c’est un composant non visible

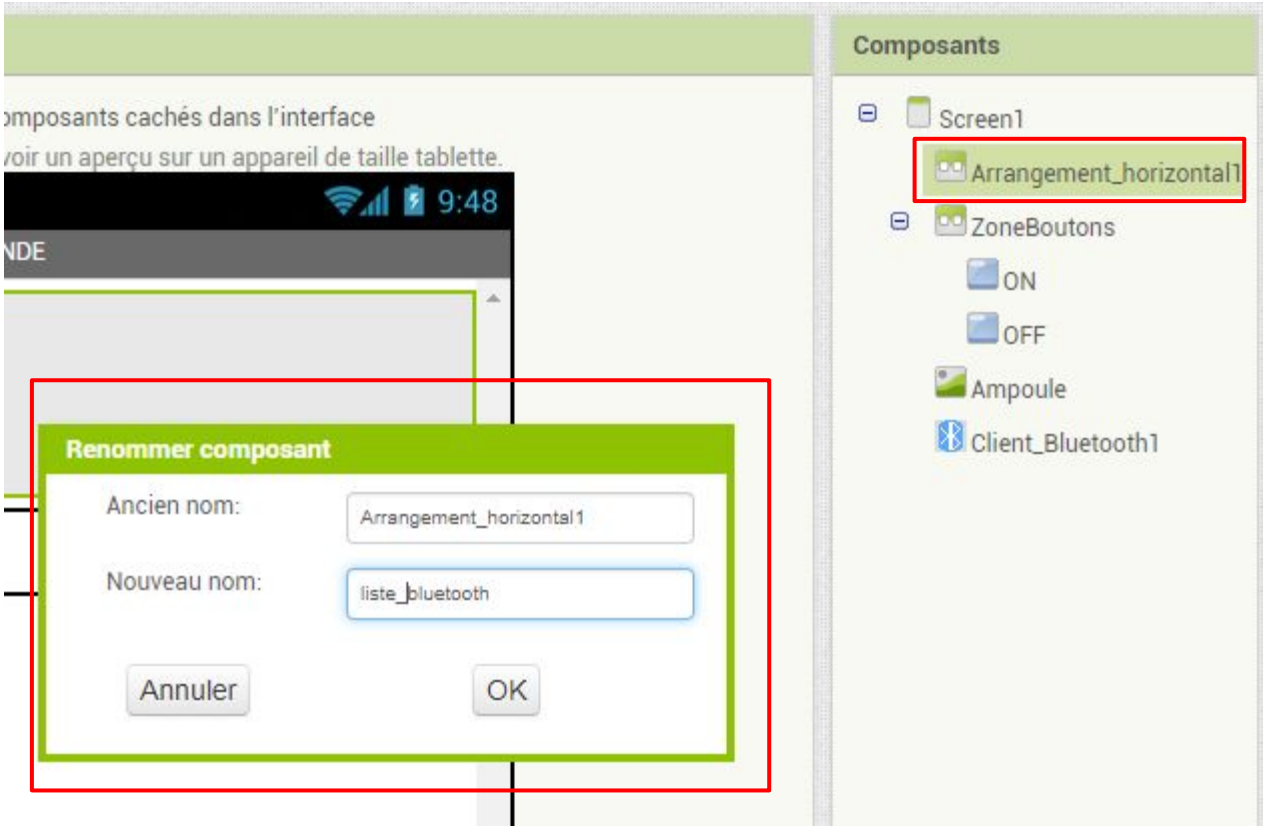


Retounons dans l'onglet "block" le composant "client_bluetooth" est maintenant disponible avec tous ses blocks de programmation

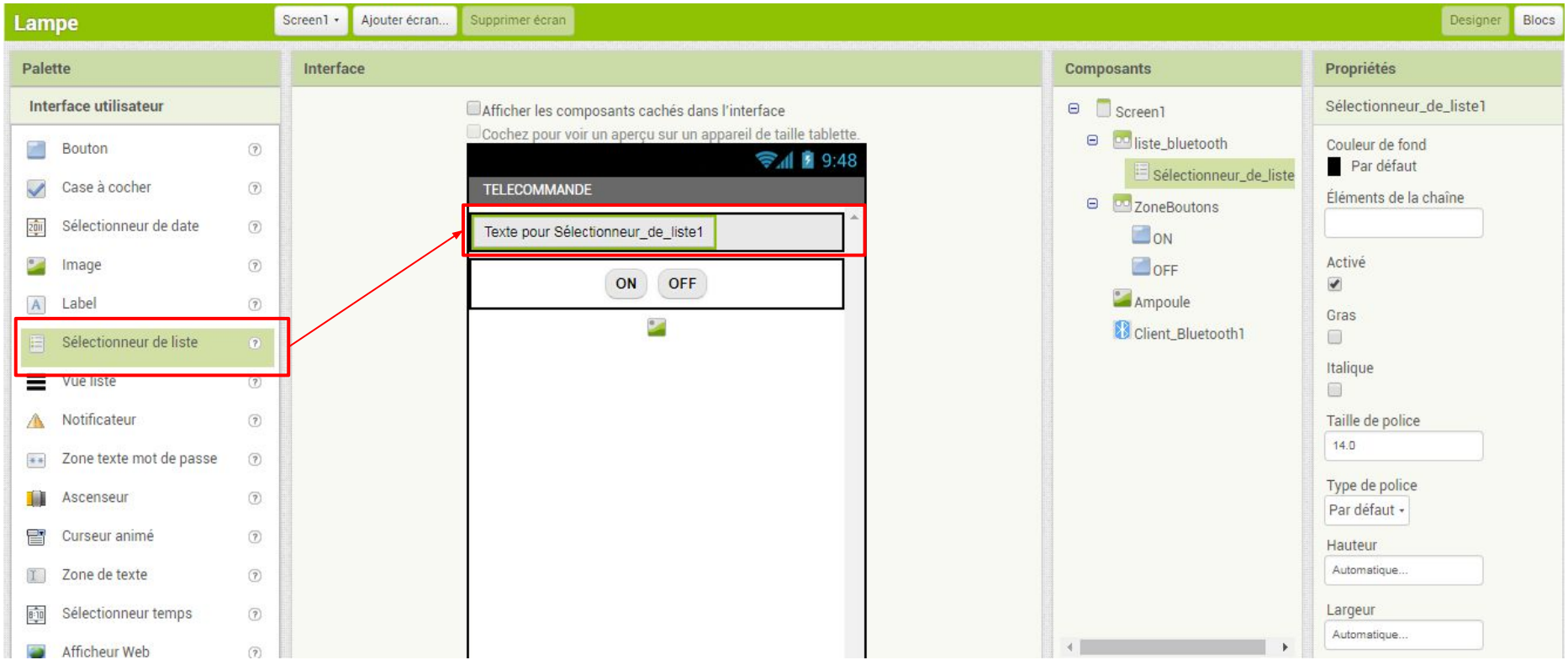
Nous allons tout d'abord retourner dans l'onglet " designer" pour créer une boîte qui servira à afficher la liste des équipements bluetooth disponibles autour de nous

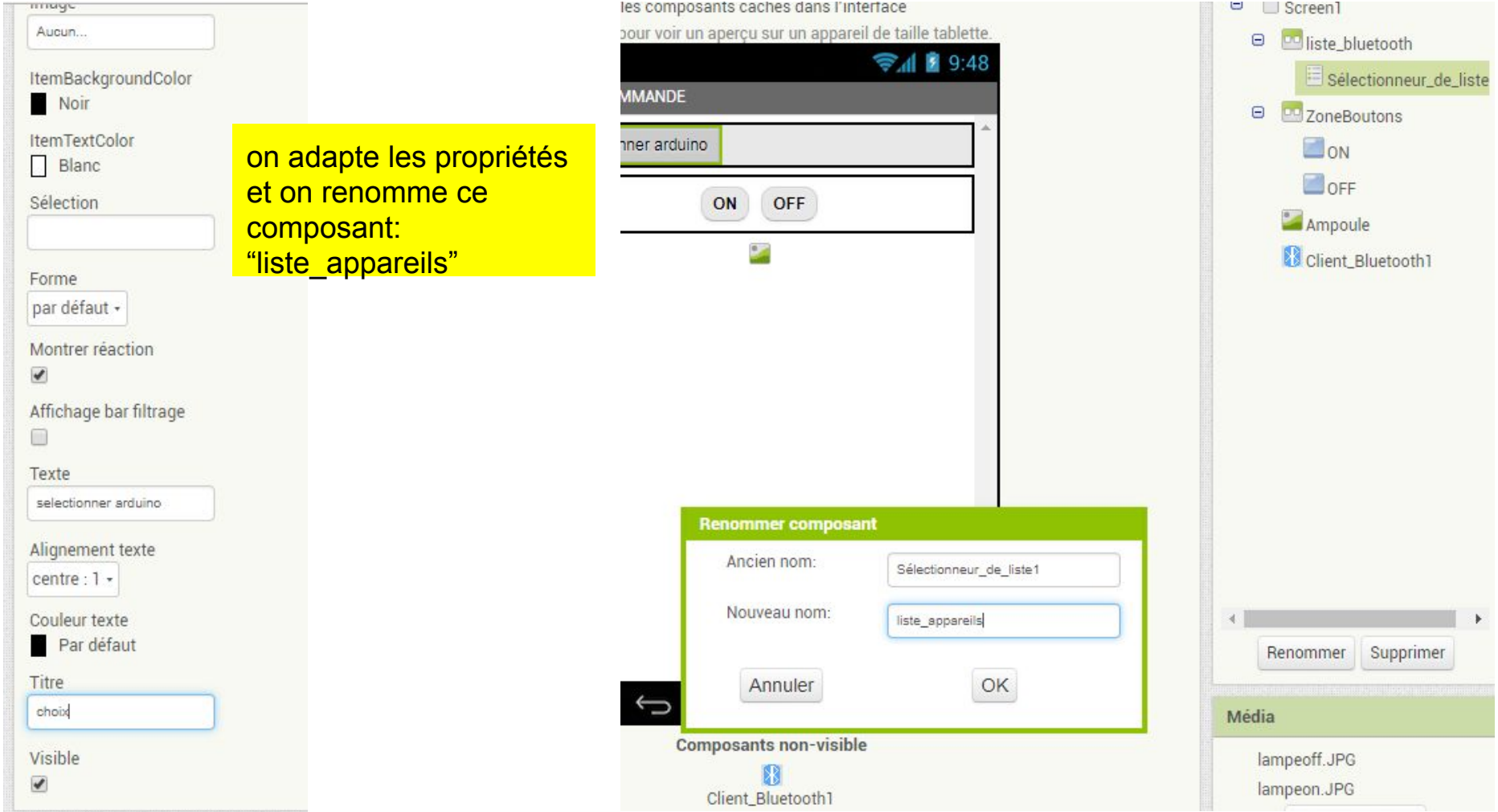


On change le nom de la zone

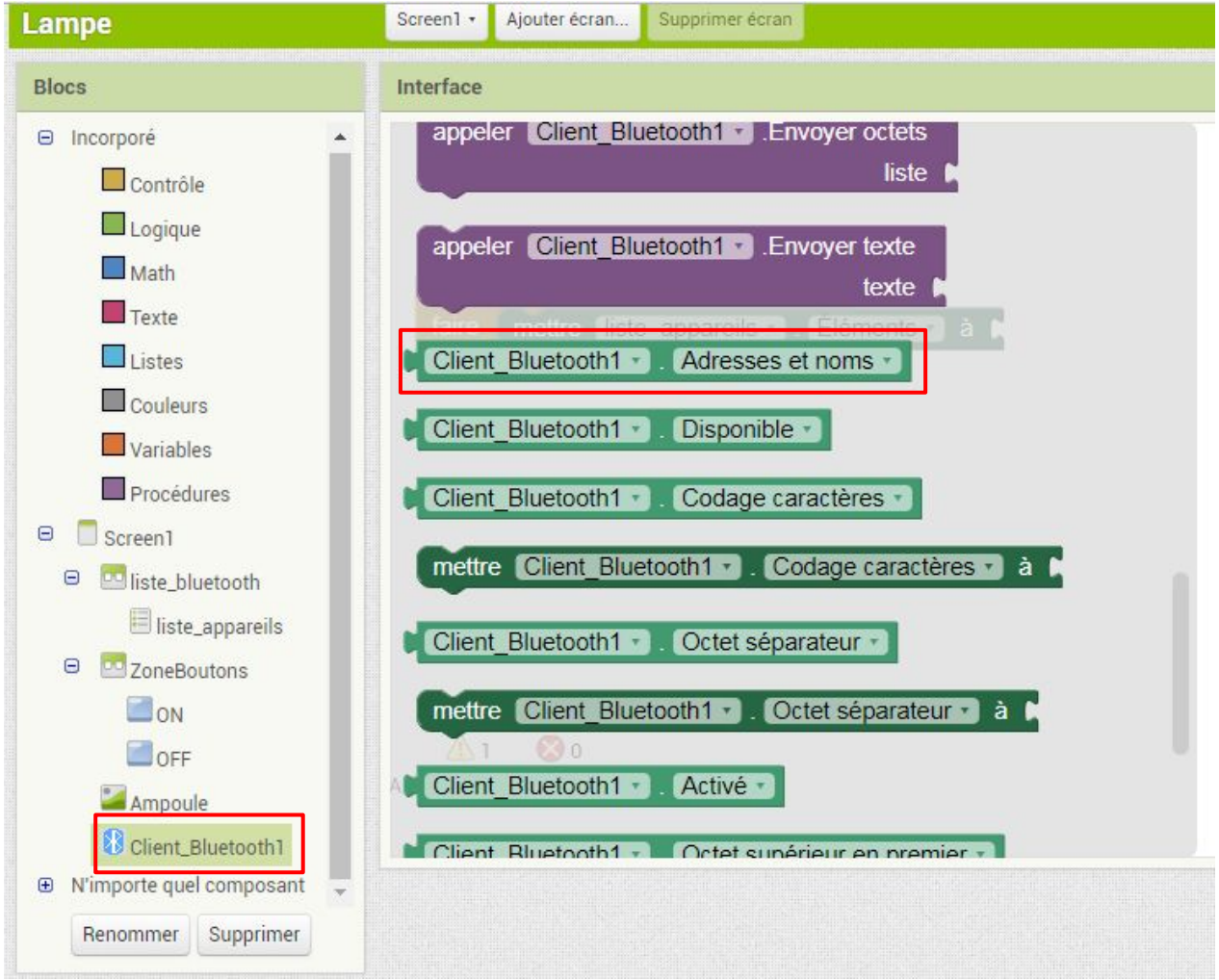


On choisit le composant “sélectionneur de liste” qui permettra de choisir un équipement dans la liste

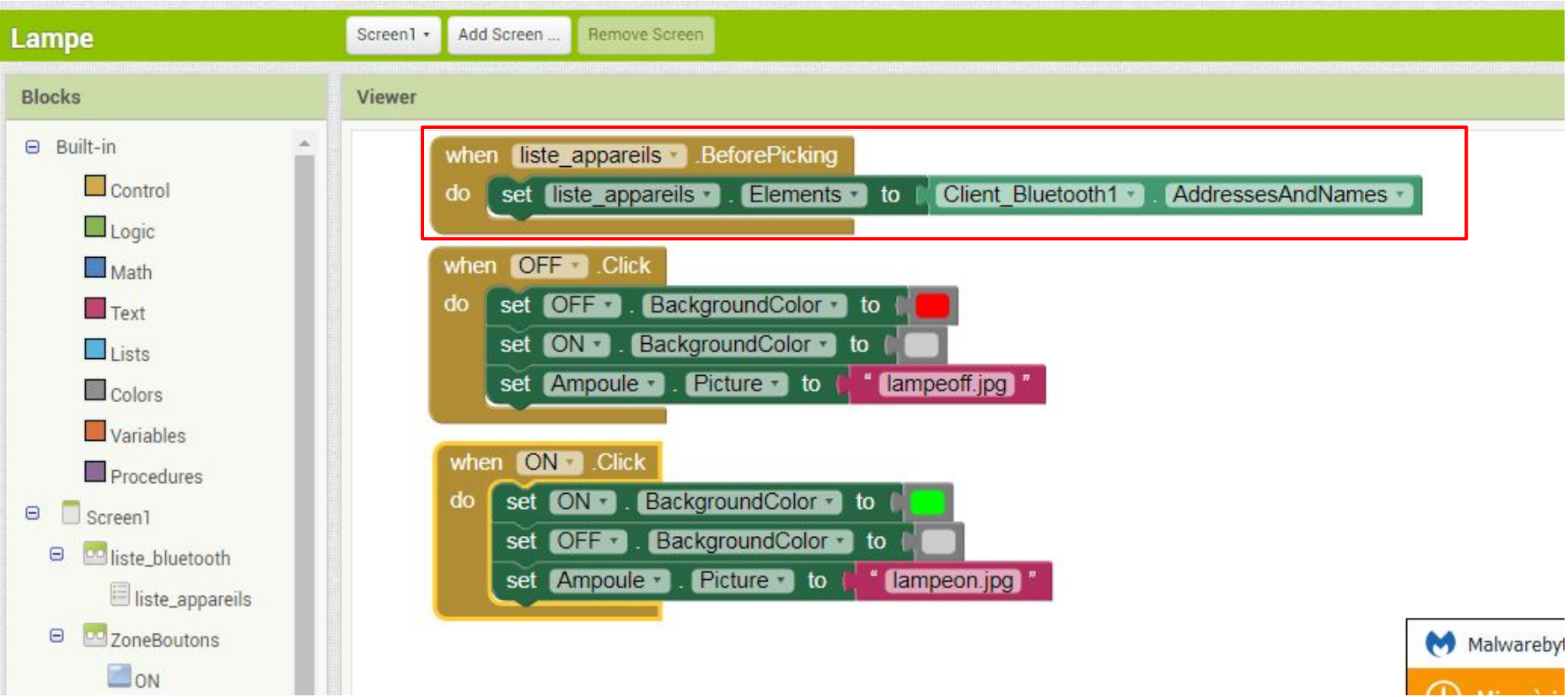



Aucun...
ItemBackgroundColor
Noir
ItemTextColor
Blanc
Sélection
Forme
par défaut
Montrer réaction
Affichage bar filtrage
Texte
selectionner arduino
Alignement texte
centre : 1
Couleur texte
Par défaut
Titre
choix
Visible
on adapte les propriétés et on renomme ce composant: “liste_appareils”
les composants caches dans l'interface
pour voir un aperçu sur un appareil de taille tablette.
9:48
MMANDE
nner arduino
ON
OFF
Renommer composant
Ancien nom:
Sélectionneur_de_liste1
Nouveau nom:
liste_appareils
Annuler
OK
Composants non-visible
Client_Bluetooth1
Screen1
liste_bluetooth
Sélectionneur_de_liste
ZoneBoutons
ON
OFF
Ampoule
Client_Bluetooth1
Renommer
Supprimer
Média
lampeoff.JPG
lampeon.JPG

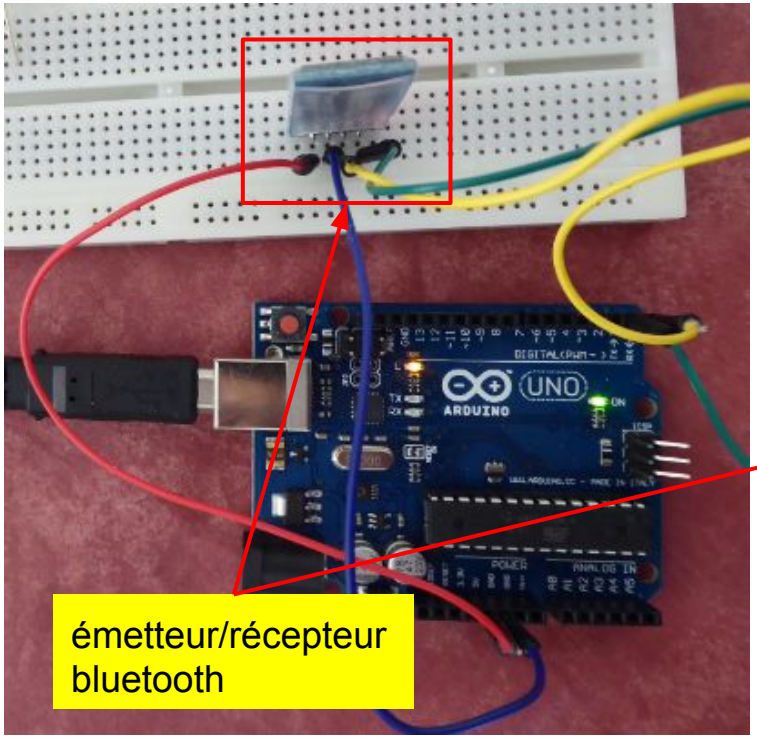

Lampe
Screen1
Ajouter écran...
Supprimer écran
Blocs
Interface
Incorporé
Contrôle
Logique
Math
Texte
Listes
Couleurs
Variables
Procédures
Screen1
liste_bluetooth
liste_appareils
ZoneBoutons
ON
OFF
Ampoule
quand liste_appareils .Avant prise
faire mettre liste_appareils . Éléments à
choisir les blocs pour afficher la liste des équipements bluetooth à proximité
1
0
Afficher les avertissements

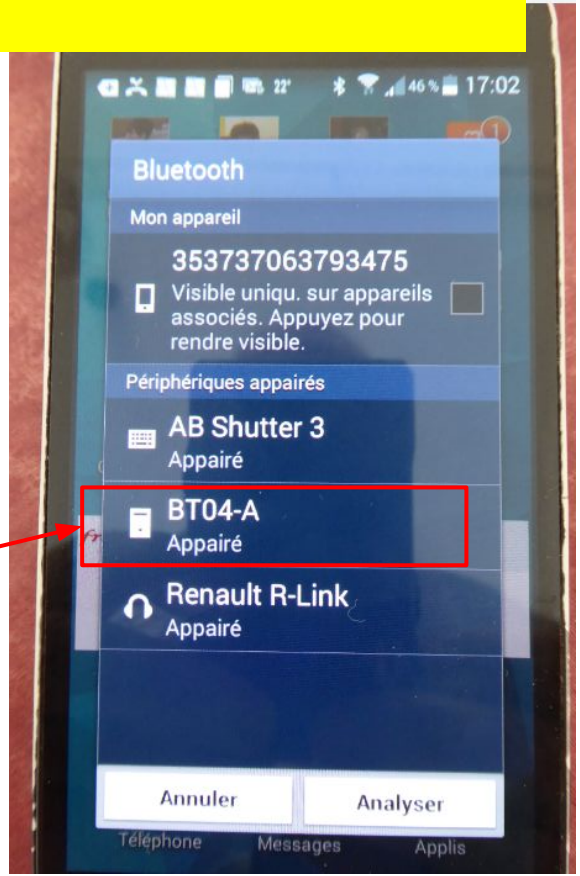


choisir les blocs pour afficher la liste des équipements bluetooth à proximité

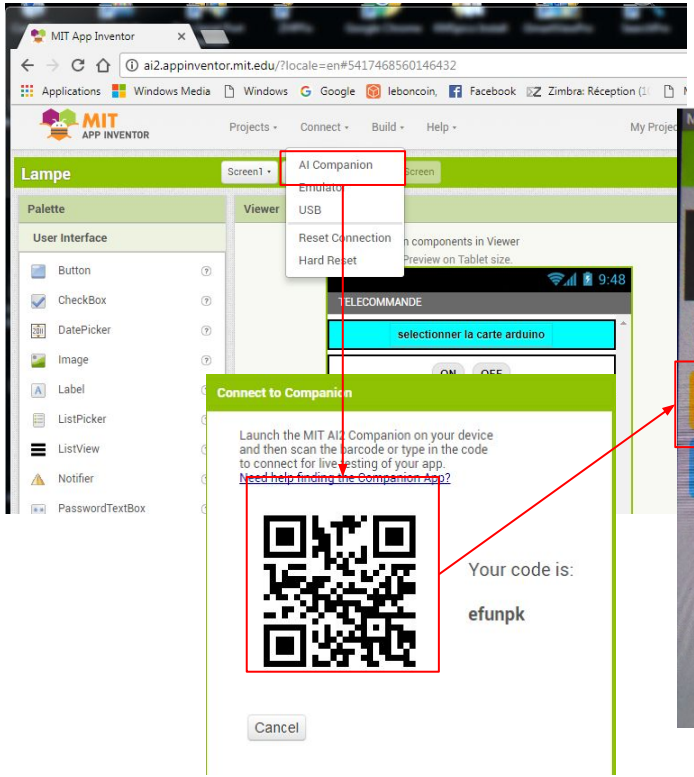

Lampe
Screen1
Add Screen ...
Remove Screen
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Colors
Variables
Procedures
Screen1
liste_bluetooth
liste_appareils
ZoneBoutons
ON
when liste_appareils .BeforePicking
do set liste_appareils . Elements to Client_Bluetooth1 . AddressesAndNames
when OFF .Click
do set OFF . BackgroundColor to
set ON . BackgroundColor to
set Ampoule . Picture to " lampeoff.jpg "
when ON .Click
do set ON . BackgroundColor to
set OFF . BackgroundColor to
set Ampoule . Picture to " lampeon.jpg "

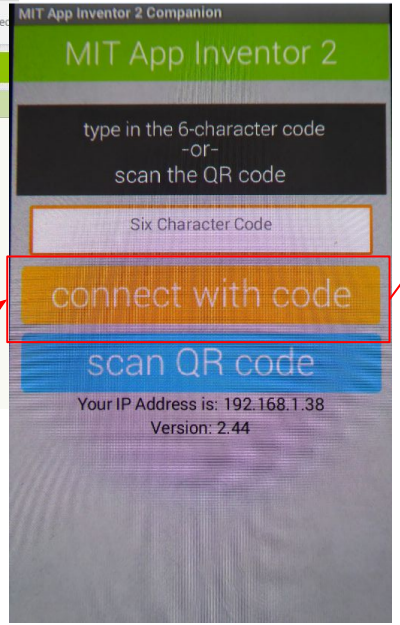
Réalisons tout d'abord un petit montage … une carte arduino et un émetteur /récepteur bluetooth . Vérifions avec un smartphone que nous voyons bien notre "carte arduino" et procédons à un appairage le code est en général "0000"

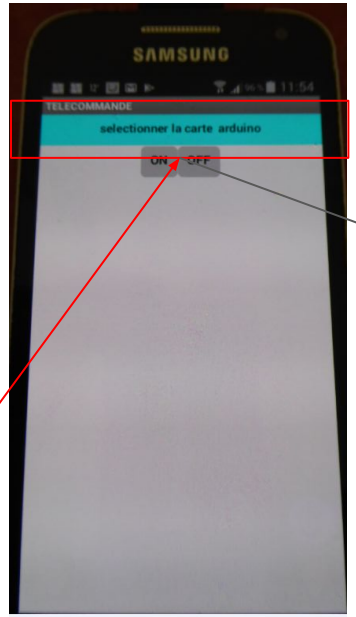


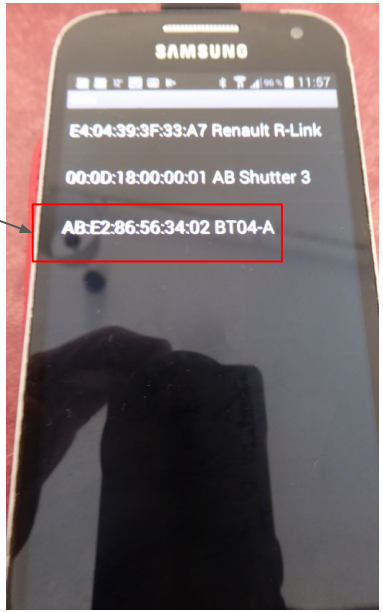


maintenant testons notre application de recherche des équipements bluetooth sur le smartphone

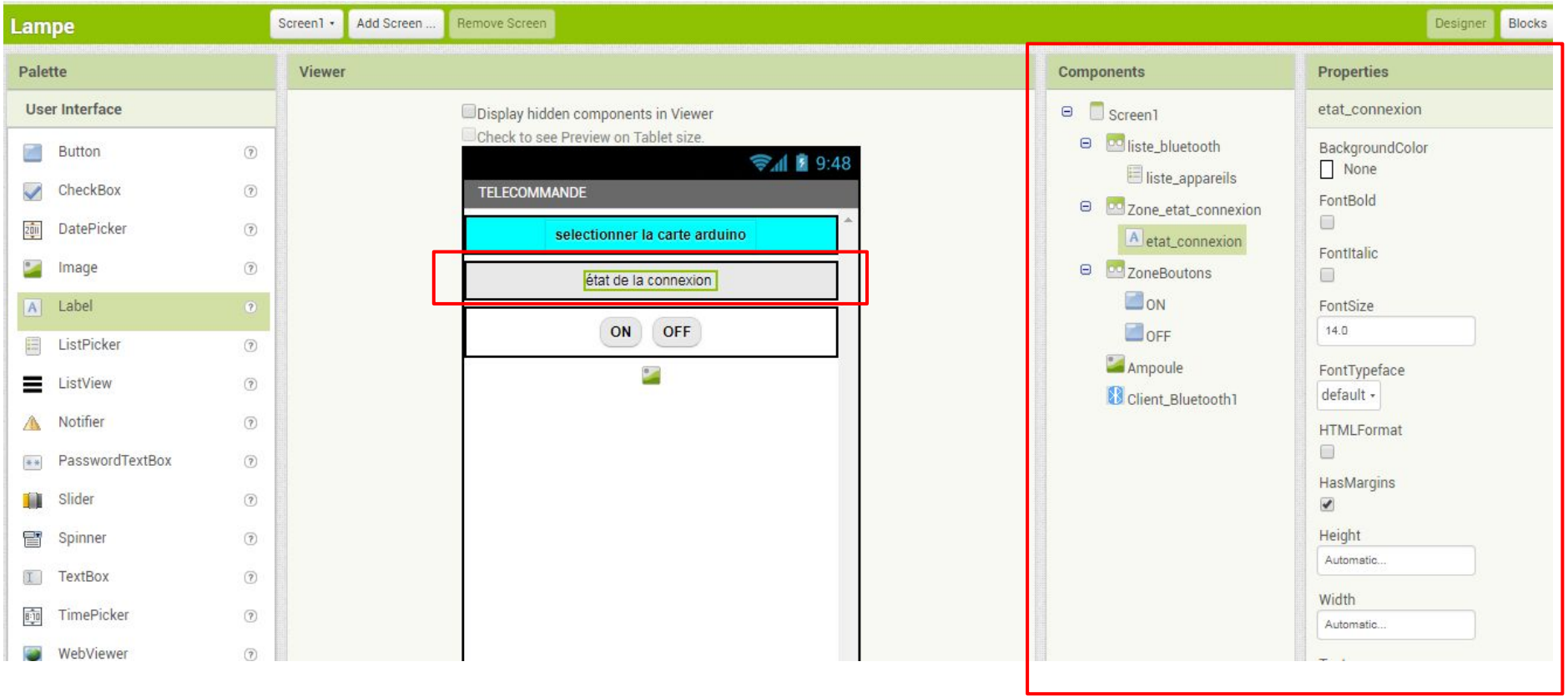




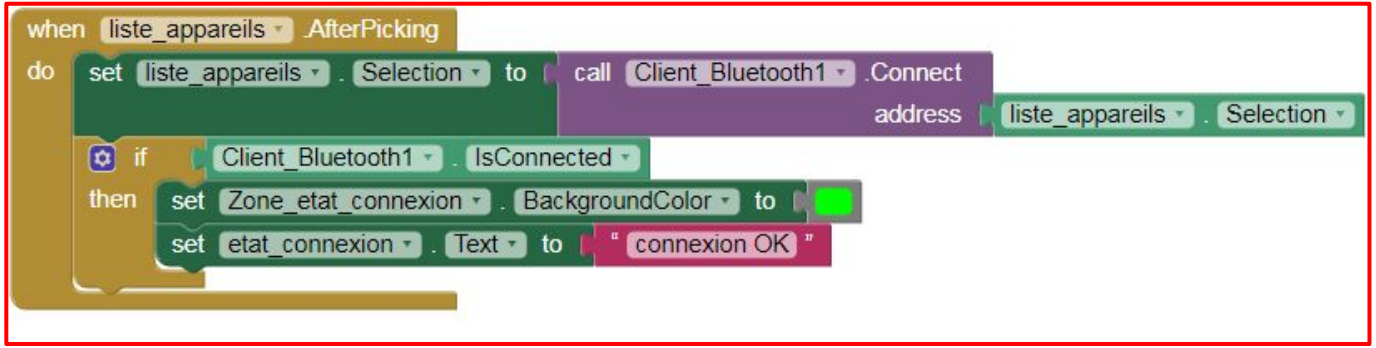


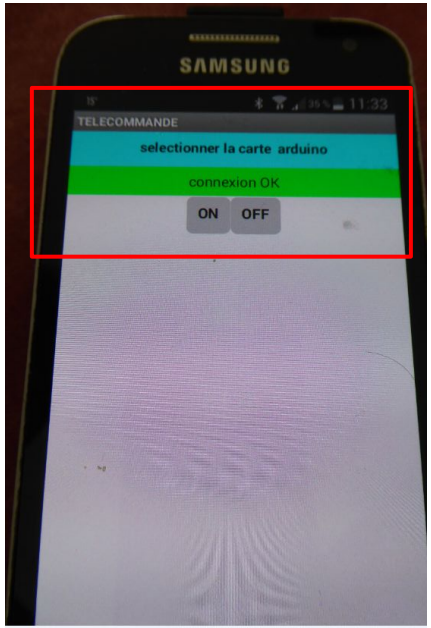


Nous allons maintenant effectuer la connexion bluetooth entre notre smartphone et la carte Arduino nous allons tout d'abord créer une zone horizontale avec un label pour afficher l'état de la connexion



Voyons maintenant le code pour lancer la connexion depuis le smartphone.
On récupère la ligne sélectionnée dans la liste des appareils bluetooth,
on met en vert le fond de la zone d'état et on envoi le message connexion OK
voilà la connection est établie





Maintenant que nous avons établi la connexion bluetooth entre notre smartphone et notre carte arduino , il va falloir établir une communication .

La communication s'effectuera au travers du port série.

L'appui sur la touche "ON" du smartphone enverra la valeur "1" et l'appui sur "OFF" enverra la valeur "0"

Avant de programmer notre application "appinventor" commençons à écrire le petit programme Arduino permettant d'allumer et d'éteindre une LED.

Pour des raisons de facilité nous utiliserons la Led de test raccordée à la pin 13

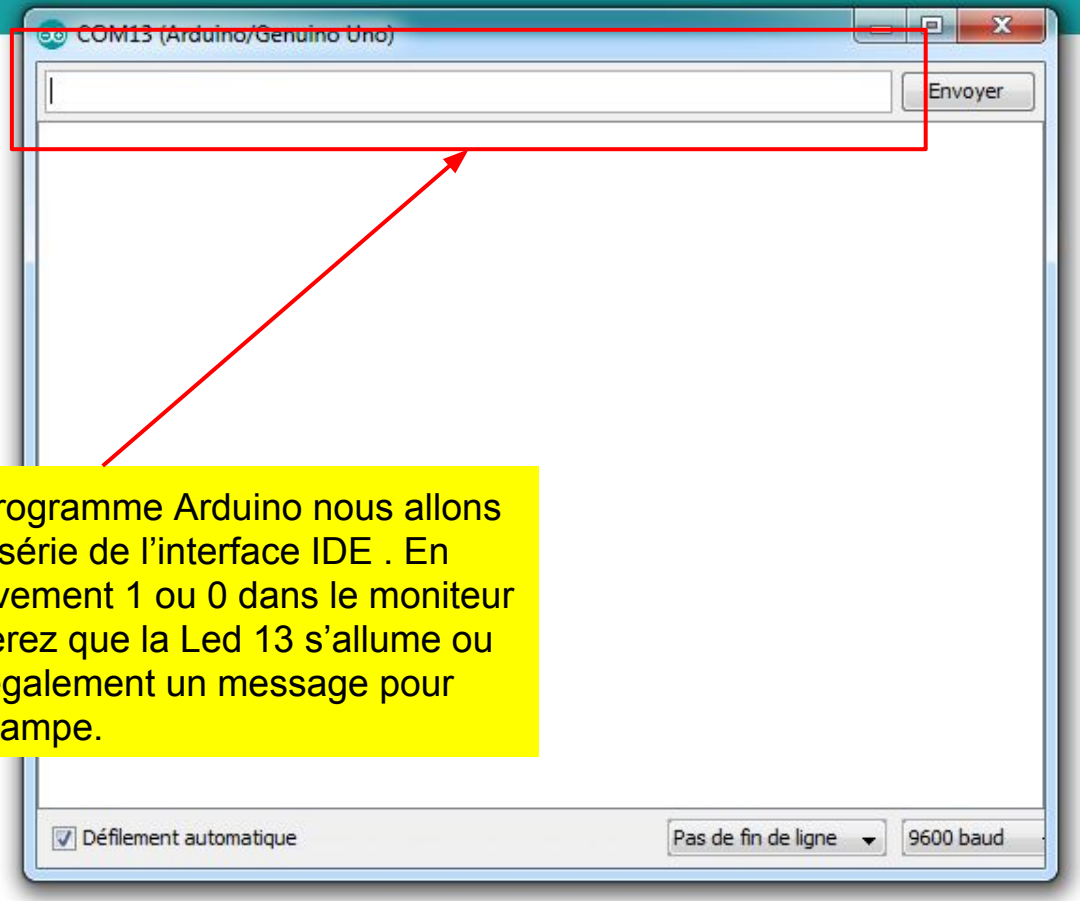
```
int LED = 13;
int etat_lampe  = 0;    //etat de la lampe
//-------------------------------------------------------------
void setup()
{
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
}
//-------------------------------------------------------------
void action()
{
  switch (etat_lampe)
  {
    case 0:
    digitalWrite(LED, LOW);
    Serial.println("eteinte");
    break;
    case 1:
    digitalWrite(LED, HIGH);
    Serial.println("allumée");
    break;

  }
}
//-------------------------------------------------------------
void loop()
{
//---------------------------------------------
//  reception du message depuis le smarphone
//---------------------------------------------
  while (Serial.available() > 0) {
  etat_lampe = Serial.parseInt();
  Serial.println(etat_lampe);
  action();

  }
}
```

Serial.available
Cette instruction permet de tester l'arrivée d'une information sur le port série… donc dans notre cas l'appui sur un bouton du smartphone.

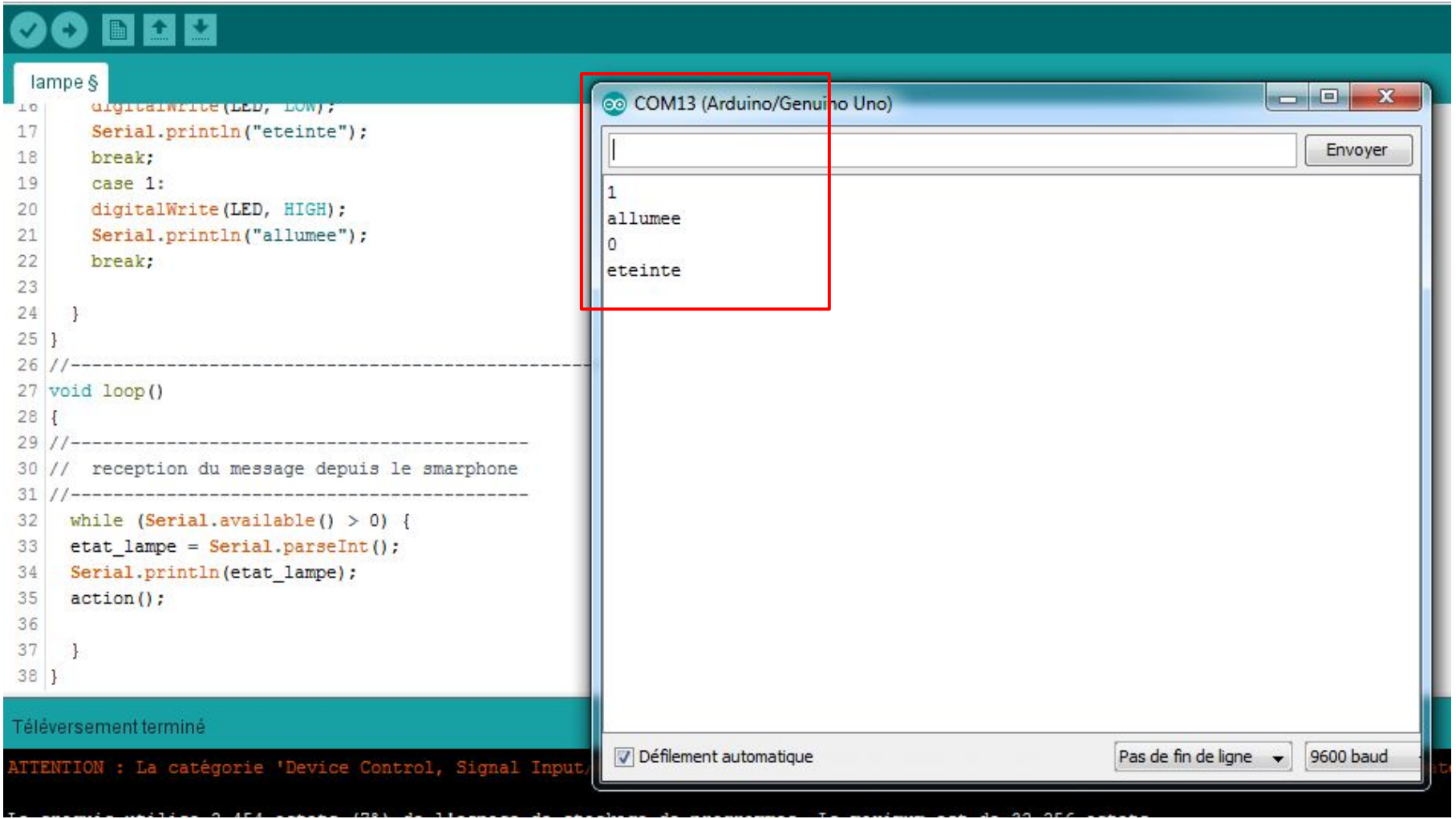
Serial.parseInt
Cette instruction permet de lire les données reçues sur le port série.
Si on souhaite recevoir plusieurs données groupées, elles doivent être séparées par des virgules. Dans ce cas il faut écrire autant d'instructions parseint qu'il y a de données à récupérer.

lampe

```
void action()
{
  switch (etat_lampe)
  {
    case 0:
    digitalWrite(LED, LOW);
    Serial.println("eteinte");
    break;
    case 1:
    digitalWrite(LED, HIGH);
    Serial.println("allumée");
    break;

  }
}
//----------------------
void loop()
{
//----------------------
//  reception du message
//----------------------
  while (Serial.availabl
  etat_lampe = Serial.pa
  Serial.println(etat_lampe);
  action();

  }
}
```



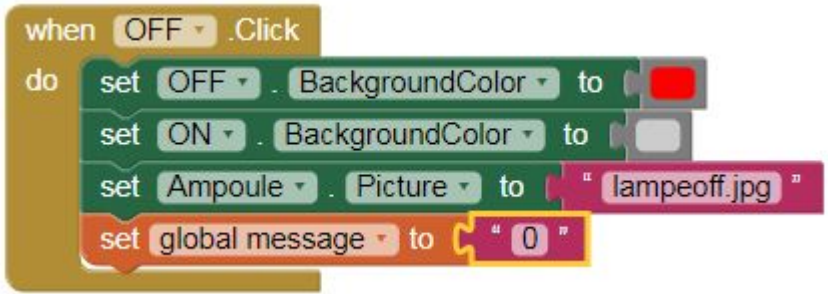
Pour tester notre programme Arduino nous allons utiliser le moniteur série de l'interface IDE . En saisissant alternativement 1 ou 0 dans le moniteur série vous constaterez que la Led 13 s'allume ou s'éteint . J'envoie également un message pour donner l'état de la lampe.

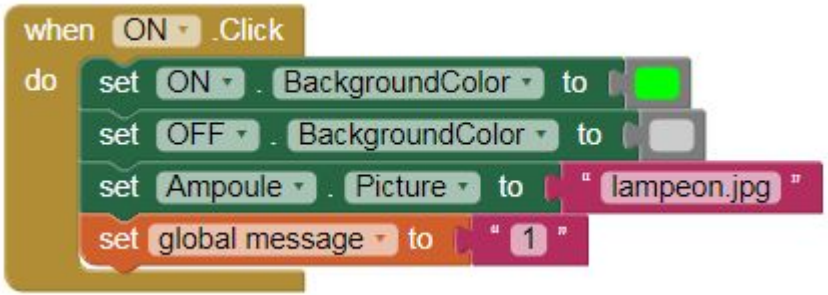

lampe §
digitalWrite(LED, LOW);
Serial.println("eteinte");
break;
case 1:
digitalWrite(LED, HIGH);
Serial.println("allumee");
break;
}
}
//--
void loop()
{
//---
// reception du message depuis le smarphone
//---
while (Serial.available() > 0) {
etat_lampe = Serial.parseInt();
Serial.println(etat_lampe);
action();
}
}
Téléversement terminé
ATTENTION : La catégorie 'Device Control, Signal Input,
COM13 (Arduino/Genuino Uno)
Envoyer
1
allumee
0
eteinte
Défilement automatique
Pas de fin de ligne
9600 baud

Maintenant que nous avons testé notre programme, nous allons modifier le programme “appinventor” pour émettre un message 0 ou 1 depuis le smartphone.

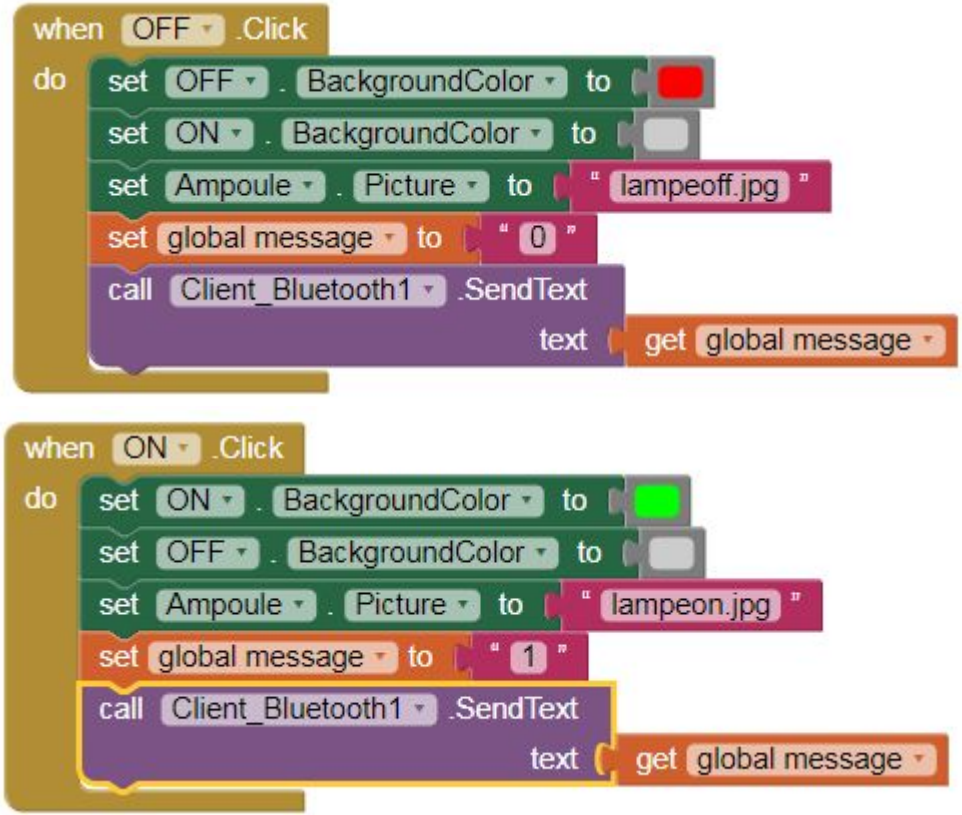
Tout d’abord créons une variable qui va contenir notre message et que l’on initialise à “ ” (vide)

Ajoutons dans notre code des boutons On / OFF la valorisation de la variable “message”





Enfin ajoutons l'envoi du message via bluetooth



Attention n'oubliez pas de faire la connexion bluetooth avant de cliquer sur les boutons

Voilà nous avons réalisé une petite application simple pour allumer et éteindre une lampe .
Elle n’est pas parfaite, nous pourrions faire une application beaucoup plus compliquée, en paramétrant une durée d’éclairage, un clignotement, changer la couleur etc…
Nous avons émis des informations depuis le smartphone , mais nous pouvons également recevoir des informations de la carte arduino.

Dans les diapositives suivantes vous trouverez l’exemple d’une lampe que l’on fait clignoter

Sur le smarphone on indique le nombre de flash souhaité et la durée entre chaque flash.

En retour on a sur le téléphone le décompte des flashs effectués et des flashs restants.

Partie 1: programme Arduino

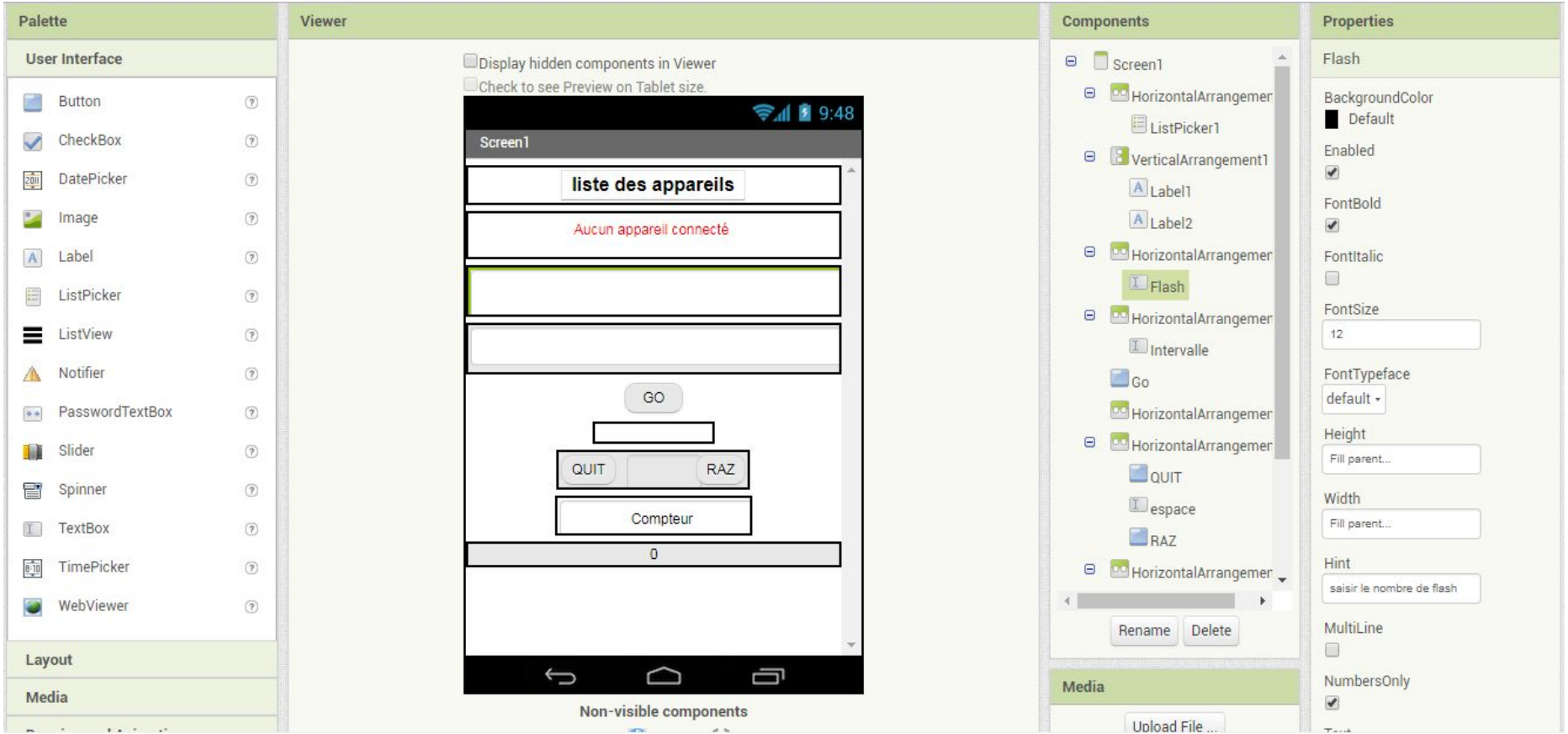
```
#define LED  12
int compteur = 0;       //compteur impulsions
int flash = 10;         //nb de clignotement
int intervalle = 1000;  //intervalle entre 2 impulsions
int reste = 0;          //nb restant d'impulsions
bool go = false;        //lancement

void setup()
{
  Serial.begin(9600);
}

void loop()
{
```

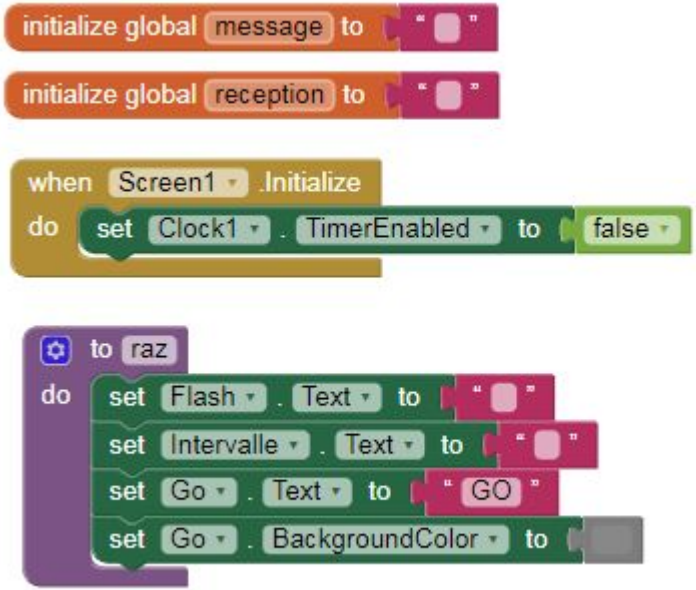
```
/* ----------------------------------------
réception du message depuis le smartphone
----------------------------------------*/
 while (Serial.available() > 0) {

 flash = Serial.parseInt();
 intervalle = Serial.parseInt();
 intervalle = intervalle * 1000;
 reste = flash;
 go = Serial.parseInt();

 Serial.println(flash);
 Serial.println(intervalle);
 Serial.println(go);
 }
```

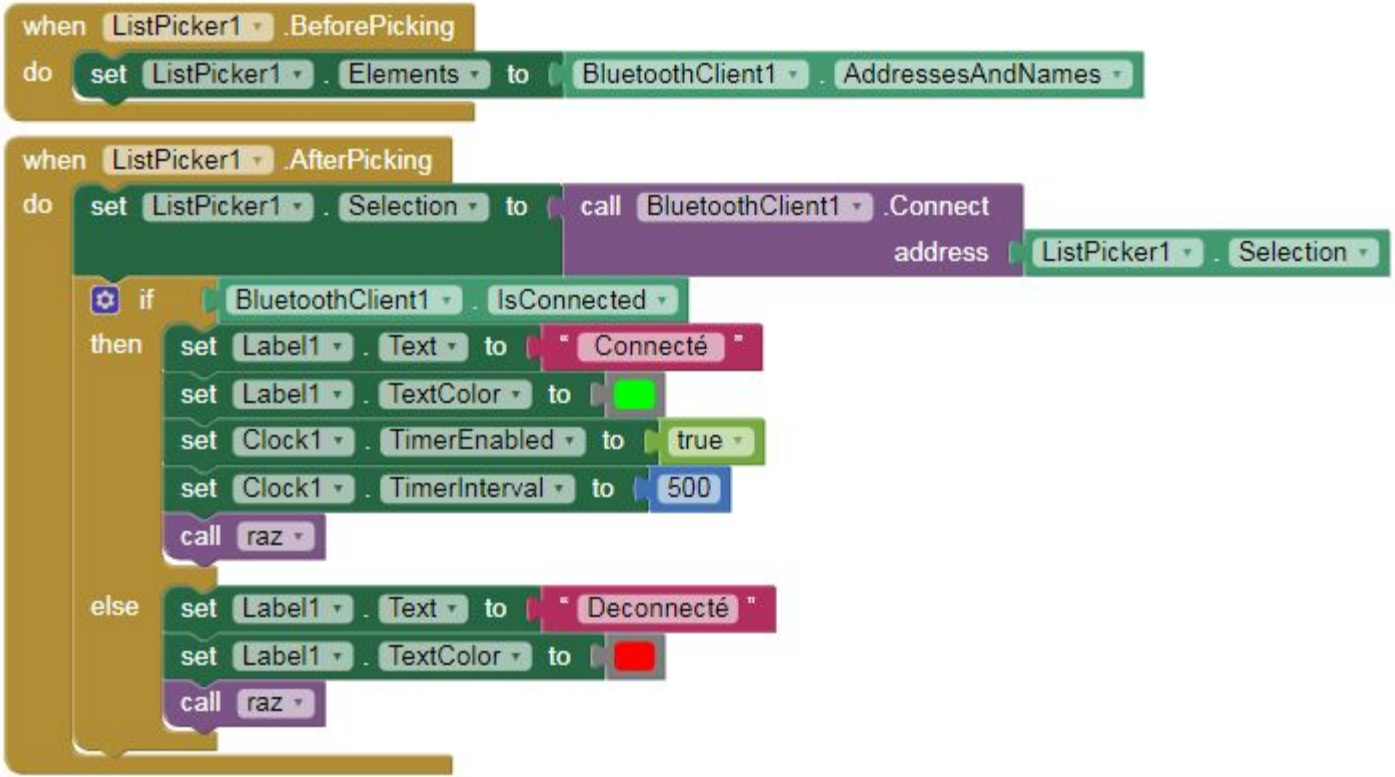
```
/* ----------------------------------------
boucle de clignotement de la led
----------------------------------------*/
 while ((compteur < flash) && (go != false))   //tant que compteur est différent du nb d'impulsions demandées
    {
      compteur ++ ; //On incrémente d’une unité
      reste --;

      digitalWrite(LED, HIGH);
      delay(intervalle);
      digitalWrite(LED, LOW);
      delay(intervalle);
      Serial.println("");
      Serial.print("-----Nb de flash réalisés----------------> ");
      Serial.println(compteur);
      Serial.print("-----Nb de flash restant----------------> ");
      Serial.println(reste);

  }
```

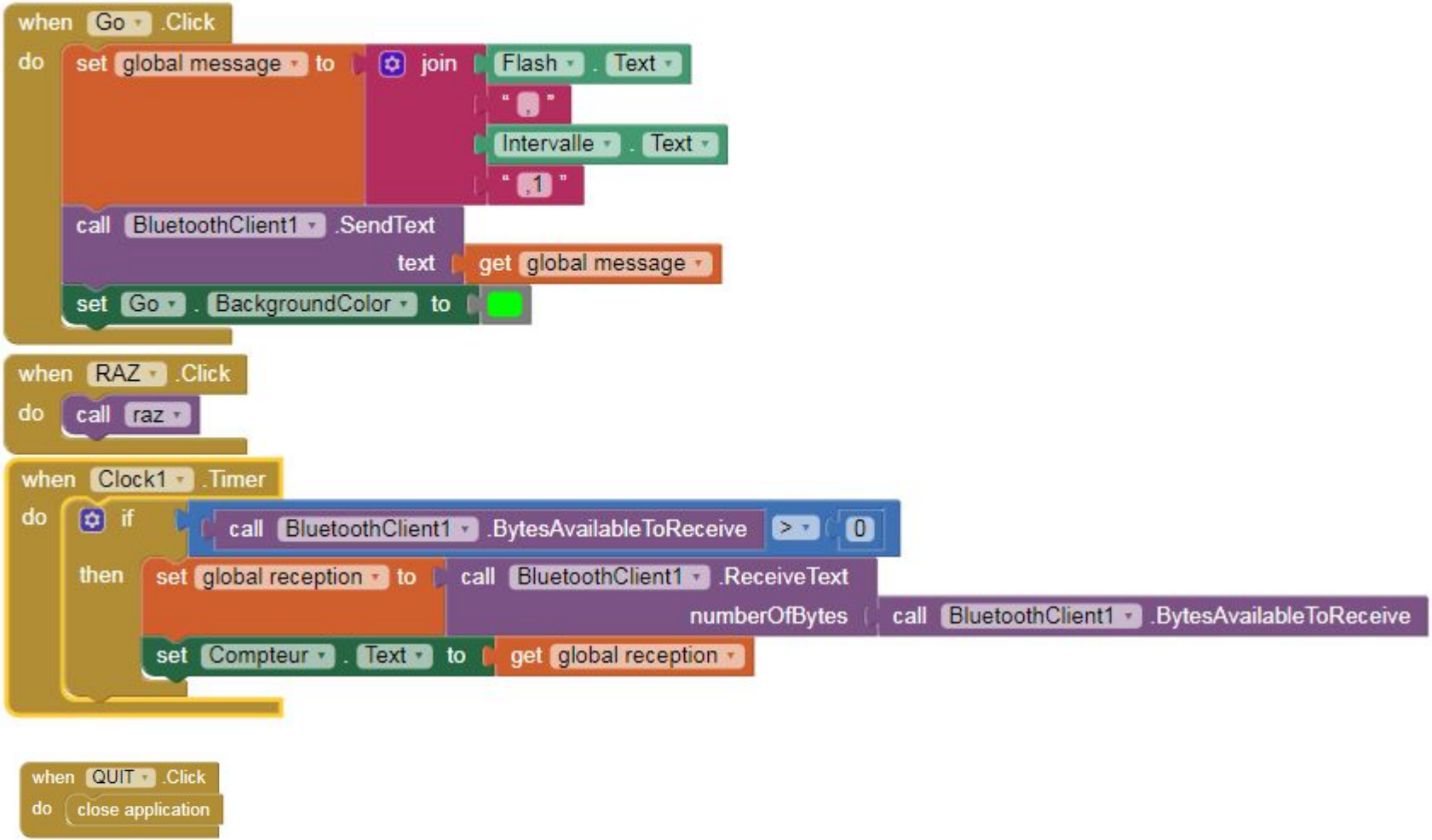
```
 compteur = 0;       //compteur impulsions
 flash = 10;         //nb de clignotement
 intervalle = 1000;  //intervalle entre 2 impulsions
 reste = 0;          //nb restant d'impultion
 go = false;         //lancement

   //fin du programme
}
```

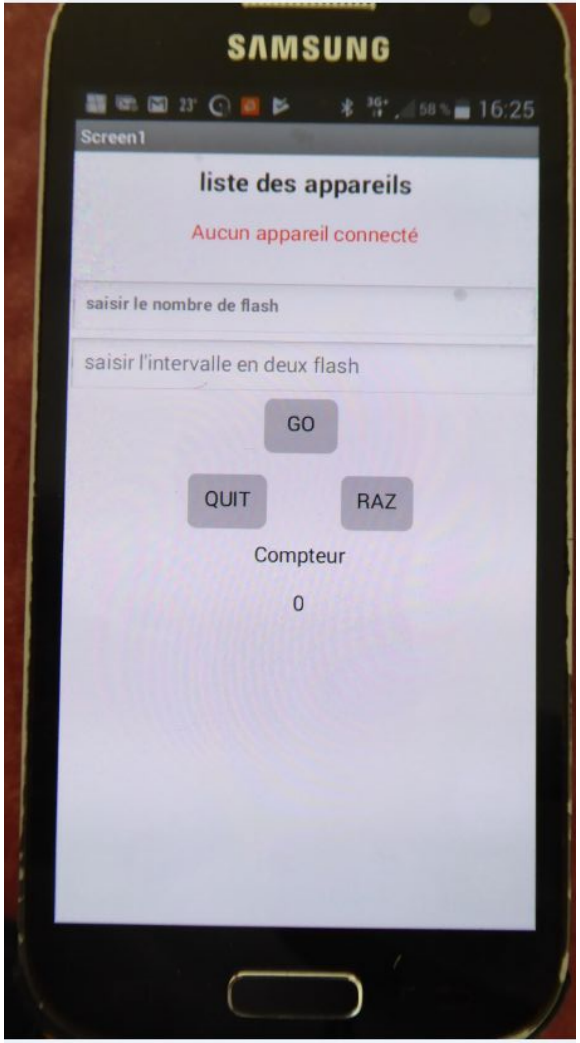
Partie 2: programme Appinventor




initialize global message to " "
initialize global reception to " "
when Screen1 .Initialize
do set Clock1 . TimerEnabled to false
to raz
do set Flash . Text to " "
set Intervalle . Text to " "
set Go . Text to " GO "
set Go . BackgroundColor to


when ListPicker1 .BeforePicking
do set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
when ListPicker1 .AfterPicking
do set ListPicker1 . Selection to call BluetoothClient1 .Connect
address ListPicker1 . Selection
if BluetoothClient1 . IsConnected
then set Label1 . Text to " Connecté "
set Label1 . TextColor to
set Clock1 . TimerEnabled to true
set Clock1 . TimerInterval to 500
call raz
else set Label1 . Text to " Deconnecté "
set Label1 . TextColor to
call raz


when Go .Click
do set global message to join Flash . Text
" , "
Intervalle . Text
" ,1 "
call BluetoothClient1 .SendText
text get global message
set Go . BackgroundColor to
when RAZ .Click
do call raz
when Clock1 .Timer
do if call BluetoothClient1 .BytesAvailableToReceive > 0
then set global reception to call BluetoothClient1 .ReceiveText
numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
set Compteur . Text to get global reception
when QUIT .Click
do close application


SAMSUNG
16:25
Screen1
liste des appareils
Aucun appareil connecté
saisir le nombre de flash
saisir l'intervalle en deux flash
GO
QUIT
RAZ
Compteur
0

Conflit d'utilisation du port série

Pour réaliser notre montage nous avons utilisé, le port série standard de la carte Arduino. Or cette façon de faire provoque un conflit d'utilisation. En effet si nous souhaitons modifier notre programme et le téléverser dans la carte , le port série est vu comme étant occupé par le composant bluetooth. Il est donc impossible de téléverser notre programme sans enlever le composant bluetooth.

La solution de contournement est d'utiliser deux pin's de la carte arduino pour émuler unecommunication série

Pour cela nous allons utiliser la bibliothèque "SoftwareSerial" et créer notre propre port série.

```
#include <SoftwareSerial.h>
SoftwareSerial mavoieserie(Rx, Tx);
```

N° des pins utilisées pour Rx et Tx

nom que l'on veut donner à notre port série

Ensuite il suffit d'utiliser ce nom dans les instructions de commande du port série

```
#include <SoftwareSerial.h>
SoftwareSerial mavoieserie(10, 11); //RX et TX sur pin 10 et 11

int LED = 13;
int etat_lampe  = 0;    //etat de la lampe
//---------------------------------------------------------
void setup()
{
  Serial.begin(9600);
  pinMode(LED, OUTPUT);

  mavoieserie.begin(9600);// démarre la voie série à la vitesse speed
  Serial.println("port serie ok sur pins 10 et 11");

}
```

```
//-------------------------------------------------------
void action()
{
  switch (etat_lampe)
  {
    case 0:
    digitalWrite(LED, LOW);
    Serial.println("eteinte");
    break;
    case 1:
    digitalWrite(LED, HIGH);
    Serial.println("allumee");
    break;
     }
}
//-------------------------------------------------------
void loop()
{
//-----------------------------------------
//  réception du message depuis le smartphone
  while (mavoieserie.available() > 0) {
  etat_lampe = mavoieserie.parseInt();
  Serial.println(etat_lampe);
  action();
  }
```

Fabrication et programmation du Dolly

- voir document “Présentation Dolly OFS”

……. c’est tout pour le moment