

# Pilotage objets connectés

## Table des matières

- Généralités
- Appinventor
- Exemple de pilotage d'une lampe à distance
- Fabrication et programmation du Dolly

# 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 avons 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

The screenshot displays the MIT App Inventor web interface. At the top, there is a navigation bar with the MIT App Inventor logo, a menu (Projects, Connect, Build, Help), and user information (My Projects, Gallery, Guide, Report an Issue, English, baicardi7514@gmail.com). Below this is a green header bar with the project name 'TUTO' and buttons for 'Screen1', 'Add Screen...', and 'Remove Screen'. In the top right corner of this header, two buttons labeled 'Designer' and 'Blocks' are highlighted with a red box and a red arrow pointing to them. The main workspace is divided into three panels: 'Palette' on the left containing various UI components like Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, TextBox, TimePicker, and WebViewer; 'Viewer' in the center showing a mobile device preview of 'Screen1' with a status bar at 9:48 and Android navigation icons at the bottom; and 'Properties' on the right showing settings for 'Screen1' such as AboutScreen, alignment, app name (TUTO), background color (White), and scrollability. A blue text box is overlaid on the center of the interface.

**Deux modes**

- Designer : pour créer l'interface sur le téléphone ou la tablette
- Blocks : Pour écrire le programme

# Mode "DESIGNER"

The image shows the MIT App Inventor Designer interface in the "DESIGNER" mode. The interface is divided into several sections:

- Top Bar:** MIT APP INVENTOR logo, navigation menus (Projects, Connect, Build, Help), and user information (My Projects, Gallery, Guide, Report an Issue, English, baicardi7514@gmail.com).
- Toolbar:** Buttons for "Screen1", "Add Screen...", and "Remove Screen".
- Viewer:** A central area showing a mobile device screen with the title "Screen1". It includes checkboxes for "Display hidden components in Viewer" and "Check to see Preview on Tablet size".
- Component Palette (Left):** A list of UI components categorized by type (User Interface, Layout, Media, Drawing and Animation, Sensors, Social, Storage, Connectivity). A red box highlights this palette, with a red arrow pointing to it from the text "Palette des composants disponibles pour créer l'interface".
- Components Panel (Right):** A list of components currently on the screen, including "Screen1". A red box highlights this panel, with an arrow pointing to it from the text "Liste des composants utilisés et leurs propriétés".
- Properties Panel (Far Right):** A detailed view of the properties for the selected component (Screen1), such as "AboutScreen", "AlignHorizontal", "AlignVertical", "AppName", "BackgroundColor", "BackgroundImage", "CloseScreenAnimation", "Icon", "OpenScreenAnimation", "ScreenOrientation", "Scrollable", "ShowListsAsJson", "ShowStatusBar", "Sizing", and "Title". A red box highlights this panel.

Two blue text boxes provide additional context:

- A blue box with the text "Liste des composants utilisés et leurs propriétés" has an arrow pointing to the Components Panel.
- A blue box with the text "Palette des composants disponibles pour créer l'interface" has a red arrow pointing to the Component Palette.

# Exemple réalisation d'une interface

The screenshot illustrates the design process in the testbai software. The central 'Viewer' shows a mobile application screen with the following elements:

- Header: **liste des appareils**
- Message: **Aucun appareil connecté** (in red)
- Buttons: **GO**, **QUIT**, **RAZ**
- Field: **Compteur**
- Value: **0**

The 'User Interface' palette on the left lists various components, with **Button** highlighted. The 'Properties' panel on the right shows the configuration for the selected **Go** button:

- Go**
- BackgroundColor: **Default**
- Enabled:
- FontBold:
- FontItalic:
- FontSize: **14.0**
- FontTypeface: **default**
- Height: **Automatic...**
- Width: **60 pixels...**
- Image: **None...**
- Shape: **rounded**
- ShowFeedback:
- Text: **go**
- TextAlignment: **center : 1**
- TextColor: **Default**
- Visible:

# Exemple Interface de programmation

The screenshot displays the 'testbai' programming environment. On the left, a 'Blocks' palette is visible, containing categories like 'Built-in', 'Screen1', and 'Media'. A red box highlights this palette, with a blue callout box containing the text 'les blocs de programmation disponibles' and a red arrow pointing to the palette. The main area is a 'Viewer' showing a block-based program. A red box highlights the code editor, with a blue callout box containing the text 'le Programme' and a red arrow pointing to the code. The code includes global initialization, screen initialization, and logic for a list picker and a timer.

testbai

Screen1 Add Screen ... Remove Screen

Viewer

Blocks

- Built-in
  - Control
  - Logic
  - Math
  - Text
  - Lists
  - Colors
  - Variables
  - Procedures
- Screen1
  - HorizontalArranger
    - ListPicker1
  - VerticalArrangement1
    - Label1
    - Label2
  - HorizontalArranger
    - Flash

Media

Upload File ...

Show Warnings

```
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 #ccc

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 #00ff00
set Clock1.TimerEnabled to true
set Clock1.TimerInterval to 500
call raz
else set Label1.Text to "Deconnecté"
set Label1.TextColor to #ff0000
call raz
```

# 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	<a href="#">?</a>
	Case à cocher	<a href="#">?</a>
	Sélecteur de date	<a href="#">?</a>
	Image	<a href="#">?</a>
	Label	<a href="#">?</a>
	Sélecteur de liste	<a href="#">?</a>
	Vue liste	<a href="#">?</a>
	Notificateur	<a href="#">?</a>
	Zone texte mot de passe	<a href="#">?</a>
	Ascenseur	<a href="#">?</a>
	Curseur animé	<a href="#">?</a>
	Zone de texte	<a href="#">?</a>
	Sélecteur temps	<a href="#">?</a>
	Afficheur Web	<a href="#">?</a>

Palette		
Interface utilisateur		
Disposition		
	Arrangement horizontal	<a href="#">?</a>
	HorizontalScrollArrangement	<a href="#">?</a>
	Arrangement tableau	<a href="#">?</a>
	Arrangement vertical	<a href="#">?</a>
	VerticalScrollArrangement	<a href="#">?</a>
Média		
Dessin et animation		
Capteurs		
Social		
Stockage		
Connectivité		
LEGO® MINDSTORMS®		
Expérimental		
Extension		

Palette		
Interface utilisateur		
Disposition		
Média		
	Caméscope	<a href="#">?</a>
	Caméra	<a href="#">?</a>
	Sélecteur d'image	<a href="#">?</a>
	Lecteur	<a href="#">?</a>
	Son	<a href="#">?</a>
	Enregistreur son	<a href="#">?</a>
	Reconnaissance vocale	<a href="#">?</a>
	Texte à parole	<a href="#">?</a>
	Lecteur vidéo	<a href="#">?</a>
	Traduction Yandex	<a href="#">?</a>
Dessin et animation		
Capteurs		
Social		
Stockage		
Connectivité		
LEGO® MINDSTORMS®		
Expérimental		
Extension		

Palette		
Interface utilisateur		
Disposition		
Média		
Dessin et animation		
	Balle	<a href="#">?</a>
	Cadre	<a href="#">?</a>
	Image lutin	<a href="#">?</a>
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 email	
 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	<a href="#">?</a>
 Capteur Nxt Couleur	<a href="#">?</a>
 CapteurLumièreNxt	<a href="#">?</a>
 CapteurSonNxt	<a href="#">?</a>
 CapteurdeContactNxt	<a href="#">?</a>
 CapteurUltrasonNxt	<a href="#">?</a>
 Nxt commandes directes	<a href="#">?</a>
 Ev3Motors	<a href="#">?</a>
 Ev3ColorSensor	<a href="#">?</a>
 Ev3GyroSensor	<a href="#">?</a>
 Ev3TouchSensor	<a href="#">?</a>
 Ev3UltrasonicSensor	<a href="#">?</a>
 Ev3Sound	<a href="#">?</a>
 Ev3UI	<a href="#">?</a>
 Ev3Commands	<a href="#">?</a>
<b>Expérimental</b>	
<b>Extension</b>	

Palette
Interface utilisateur
Disposition
Média
Dessin et animation
Capteurs
Social
Stockage
Connectivité
LEGO® MINDSTORMS®
Expérimental
 FirebaseDB <a href="#">?</a>
<b>Extension</b>

Palette
Interface utilisateur
Disposition
Média
Dessin et animation
Capteurs
Social
Stockage
Connectivité
LEGO® MINDSTORMS®
Expérimental
Extension
<i><a href="#">Import extension</a></i>

Composants	Propriétés
<ul style="list-style-type: none"> <li>Screen1           <ul style="list-style-type: none"> <li>Bouton1</li> <li>Sélectionneur_de_liste1</li> </ul> </li> </ul> <p>Renommer Supprimer</p>	<p>Sélectionneur_de_liste1</p> <p>Couleur de fond  <input checked="" type="checkbox"/> Par défaut</p> <p>Éléments de la chaîne  <input type="text" value="10,15,20"/></p> <p>Activé  <input checked="" type="checkbox"/></p> <p>Gras  <input type="checkbox"/></p> <p>Italique  <input type="checkbox"/></p> <p>Taille de police  <input type="text" value="14.0"/></p> <p>Type de police  <input type="text" value="Par défaut"/></p> <p>Hauteur  <input type="text" value="Automatique..."/></p> <p>Largeur  <input type="text" value="Automatique..."/></p> <p>Image  <input type="text" value="Aucun..."/></p> <p>ItemBackgroundColor  <input checked="" type="checkbox"/> Noir</p> <p>ItemTextColor  <input type="checkbox"/> Blanc</p> <p>Sélection  <input type="text"/></p> <p>Forme  <input type="text" value="par défaut"/></p> <p>Montrer réaction  <input checked="" type="checkbox"/></p>
<p><b>Média</b></p> <ul style="list-style-type: none"> <li>P1020046.JPG</li> <li>luma-jau...22-32.png</li> </ul> <p>Charger fichier ...</p>	

# 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

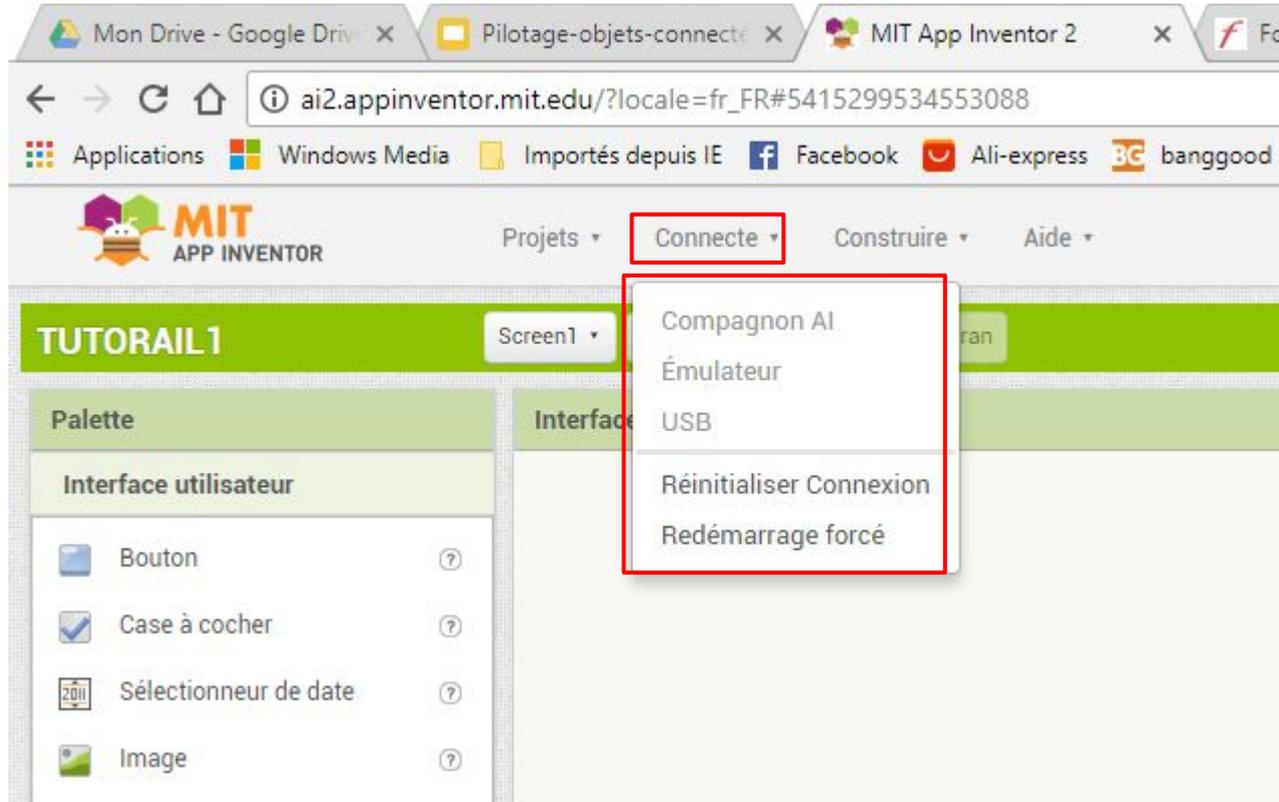
Un QR code s'affiche

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



The image shows a screenshot of the MIT App Inventor 2 web interface in a browser. The browser's address bar displays the URL `ai2.appinventor.mit.edu/?locale=fr_FR#5415299534553088`. The page header includes the MIT App Inventor logo and navigation links for 'Projets', 'Connecte', 'Construire', and 'Aide'. The 'Connecte' menu is open, showing options: 'Compagnon AI', 'Émulateur', 'USB', 'Réinitialiser Connexion', and 'Redémarrage forcé'. On the left, the 'TUTORAIL 1' section is visible, along with a 'Palette' containing 'Interface utilisateur' components: 'Bouton', 'Case à cocher', 'Sélecteur de date', and 'Image'. The main workspace area is partially visible, showing 'Screen1' and 'Interface'.

# 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.

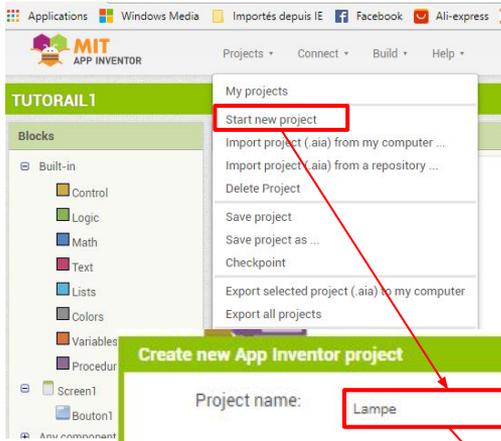
Exemple de pilotage d'une lampe à distance

# Commençons par un exemple simple:

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

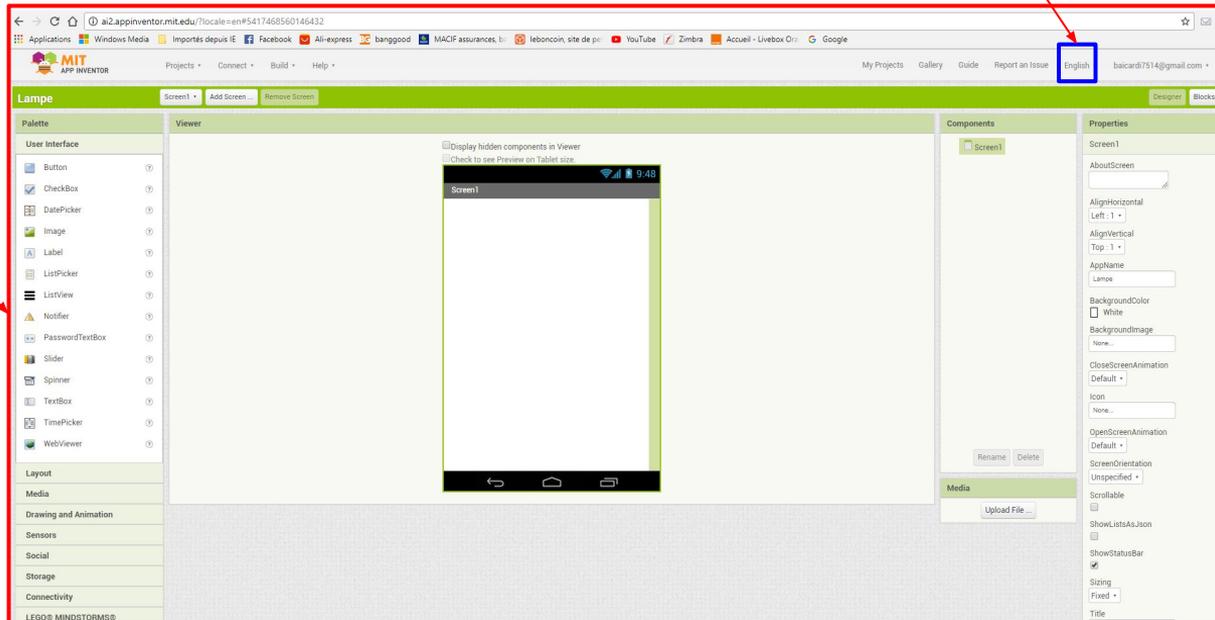
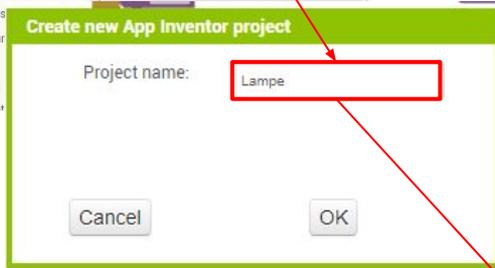
Nous allons procéder en trois phases:

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

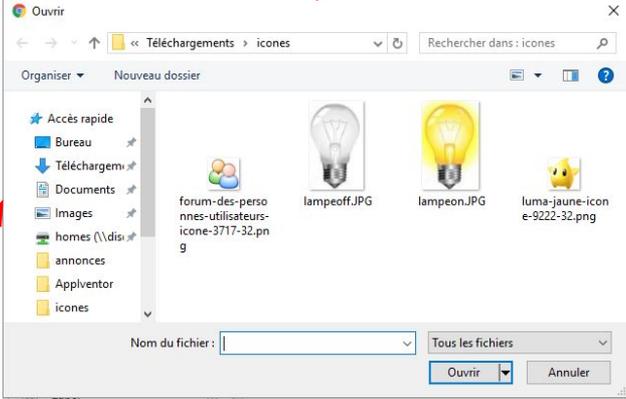
Charger fichier ...

Tout d'abord téléchargeons les images qui serviront à illustrer l'état de la lampe

Média

lampeoff.JPG  
lampeon.JPG

Charger fichier ...



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

The screenshot shows the 'Lampe' application designer interface. On the left, the 'PaLETTE' sidebar is open to the 'Disposition' section, where 'Arrangement horizontal' is selected. In the center, a mobile device preview displays a screen titled 'TELECOMMANDE'. A red box highlights a new horizontal arrangement zone on the screen. On the right, the 'Composants' panel shows 'Arrangement\_horizontal' selected, and the 'Propriétés' panel shows its configuration options. A red box highlights the 'Propriétés' panel. Below the 'Composants' panel, a 'Renommer' button is highlighted, which triggers a dialog box.

on change le nom  
de la zone

**Renommer composant**

Ancien nom: Arrangement\_horizontal1

Nouveau nom: Zone-Boutons

Annuler OK

# Insertion de deux boutons dans la zone boutons

The screenshot displays the 'Lampe' software interface for designing a mobile application. The main workspace shows a preview of a screen titled 'TELECOMMANDE' with two buttons labeled 'Texte pour Bouton1' and 'Texte pour Bouton2'. The interface is divided into several panels:

- PaLETTE**: A sidebar on the left containing various UI components like 'Bouton', 'Case à cocher', 'Sélectionneur de date', etc.
- Interface**: The central workspace where the design is being created.
- Composants**: A panel on the right showing the hierarchy of components, including 'Screen1', 'ZoneBoutons', 'Bouton1', and 'Bouton2'.
- Propriétés**: A panel on the right showing the properties for the selected 'Bouton2' component, such as 'Couleur de fond', 'Activé', 'Gras', 'Itallique', 'Taille de police', 'Type de police', 'Hauteur', 'Largeur', 'Image', 'Forme', 'Montrer réaction', 'Texte', 'Alignement texte', 'Couleur texte', and 'Visible'.

The top bar includes navigation options like 'Screen1', 'Ajouter écran...', and 'Supprimer écran', along with 'Designer' and 'Blocs' buttons. The bottom bar shows the time '9:48' and various status icons.

# On renomme les boutons et on leur donne des propriétés

The screenshot displays the LEGO Mindstorms IDE interface for a project named "Lampe". The interface is divided into several sections:

- PaLETTE**: A list of UI components on the left, including "Bouton", "Case à cocher", "Sélecteur de date", "Image", "Label", "Sélecteur de liste", "Vue liste", "Notificateur", "Zone texte mot de passe", "Ascenseur", "Curseur animé", "Zone de texte", "Sélecteur temps", and "Afficheur Web".
- Interface**: The central workspace showing a mobile app preview. The preview displays a screen titled "TELECOMMANDE" with two buttons labeled "ON" and "OFF". Red boxes highlight these buttons.
- Composants**: A panel on the right showing the components used in the interface. It includes "Screen1" and "ZoneBoutons". The "ZoneBoutons" component is highlighted with a red box, showing "ON" and "OFF" buttons. Below it are "Renommer" and "Supprimer" buttons, with "Renommer" highlighted in red.
- Propriétés**: A panel on the right showing the properties of the selected component. The "ON" property is highlighted in red, and the "Texte" property is set to "ON".

The "Propriétés" panel for the "ON" button shows the following settings:

- ON
- Couleur de fond: Par défaut
- Activé:
- Gras:
- Italique:
- Taille de police: 14.0
- Type de police: Par défaut
- Hauteur: Automatique...
- Largeur: Automatique...
- Image: Aucun...
- Forme: par défaut
- Montrer réaction:
- Texte: ON
- Alignement texte: centre : 1
- Couleur texte: Par défaut
- Visible:

# 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"



# Lampe

Screen

## Blocks

View

### Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

### Screen1

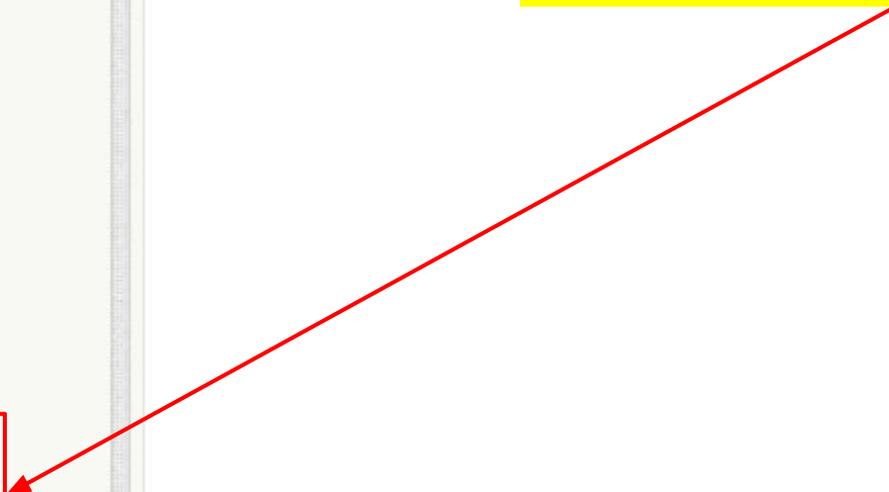
#### ZoneBoutons

- ON
- OFF

#### Ampoule

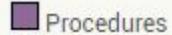
### Any component

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



## Blocks

### Built-in



### Screen1

#### ZoneBoutons



#### Any component

## Viewer

when ON ▾ .Click

do

when ON ▾ .GotFocus

do

when ON ▾ .LongClick

do

when ON ▾ .LostFocus

do

when ON ▾ .TouchDown

do show Warnings

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

## Blocks

## Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

## Screen1

## ZoneBoutons

- ON
  - OFF
  - Ampoule
- Any component

## Viewer

when ON ▾ . TouchUp

do

ON ▾ . BackgroundColor ▾

set ON ▾ . BackgroundColor ▾ to

ON ▾ . Enabled ▾

set ON ▾ . Enabled ▾ to

ON ▾ . FontBold ▾

set ON ▾ . FontBold ▾ to

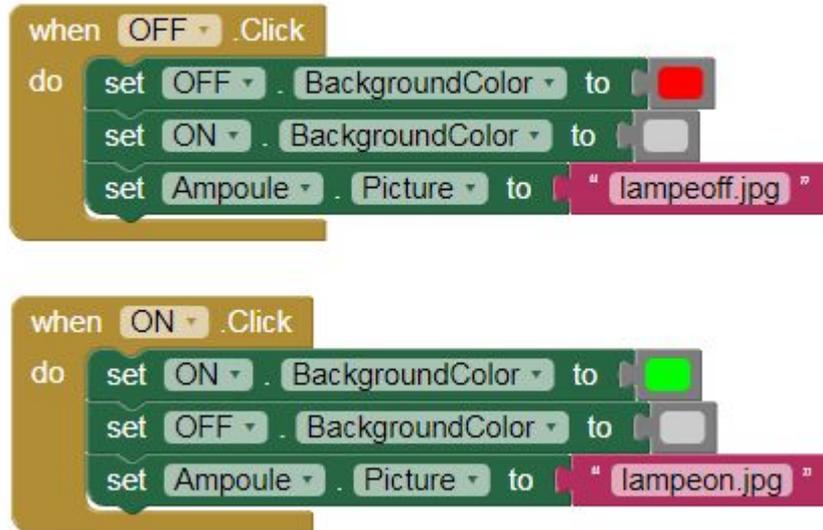
ON ▾ . FontItalic ▾

set ON ▾ . FontItalic ▾ to

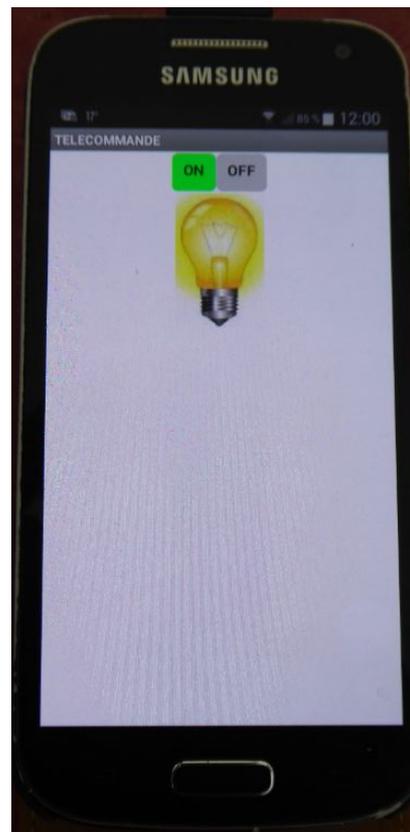
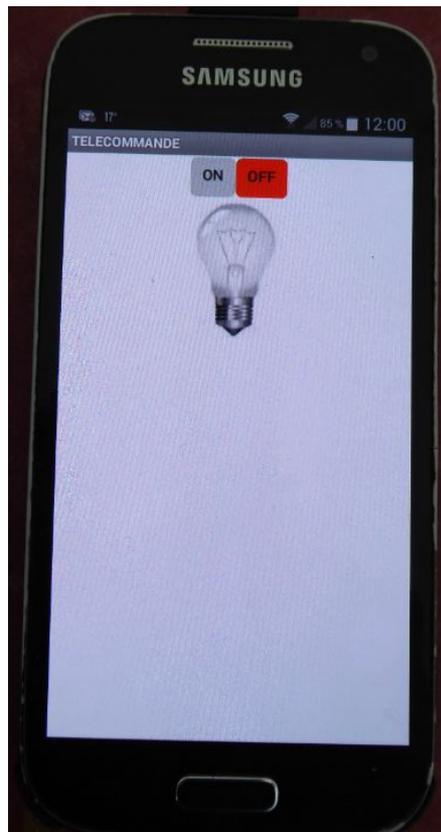
ensuite on descend avec l'ascenseur pour chercher les instructions nécessaires à notre programme .

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

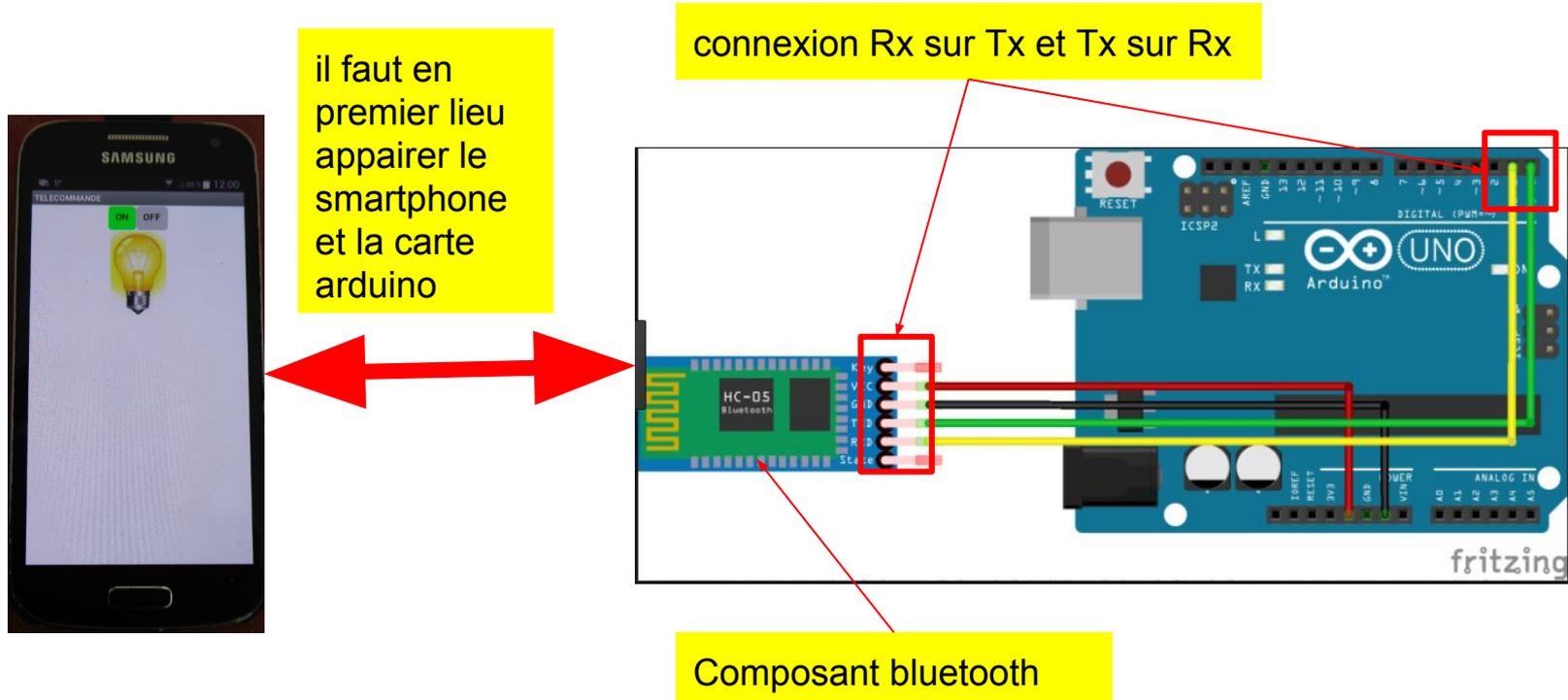


Et voilà ce que ça donne la  
partie interface est terminée



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



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

The screenshot displays the LEGO Mindstorms Designer software interface. At the top, a green header bar contains the title "Lampe" and navigation buttons: "Screen1", "Add Screen ...", and "Remove Screen". In the top right corner, two tabs are visible: "Designer" and "Blocks", both highlighted with a red rectangular box.

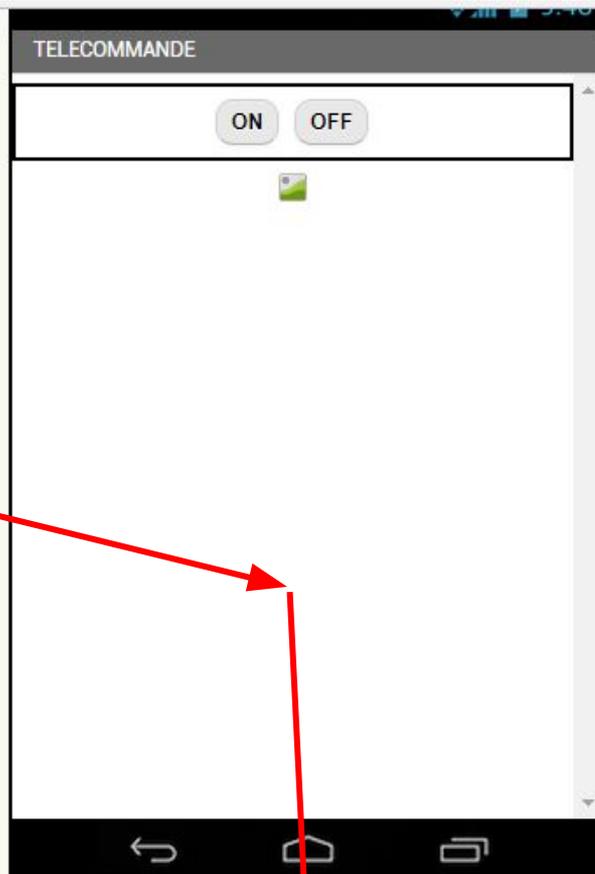
On the left side, there is a "Palette" with several categories: "User Interface", "Layout", "Media", "Drawing and Animation", "Sensors", "Social", "Storage", "Connectivity", and "Experimental". The "Connectivity" category is highlighted with a red rectangular box and contains the following items:

- ActivityStarter
- BluetoothClient
- BluetoothServer
- Web

The central "Viewer" area shows a mobile app preview. At the top, there are status icons for Wi-Fi, cellular signal, and battery, along with the time "9:48". Below this, a dark grey bar contains the text "TELECOMMANDE". Underneath, there are two buttons labeled "ON" and "OFF".

On the right side, there are two panels: "Components" and "Properties". The "Components" panel shows a tree view with "Screen1" expanded to show "ZoneBoutons", which contains "ON" and "OFF" components, and "Ampoule". The "Properties" panel shows settings for "Screen1", including "AboutScreen", "AlignHorizontal" (Center : 3), "AlignVertical" (Top : 1), "AppName" (Lampe), "BackgroundColor" (White), "BackgroundImage" (None...), "CloseScreenAnimation" (Default), and "Icon" (None...).

- Média
- Dessin et animation
- Capteurs
- Social
- Stockage
- Connectivité
  - ⚡ Déclencheuractivité ?
  - 📶 Client Bluetooth ?**
  - 📶 Serveur Bluetooth ?
  - 🌐 Web ?
- LEGO® MINDSTORMS®
- Expérimental
- Extension



Composant non-visible



Tirer le composant "Client bluetooth" sur l'écran

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

## Blocs

## Incorporé

- Contrôle
- Logique
- Math
- Texte
- Listes
- Couleurs
- Variables
- Procédures

## Screen1

## ZoneBoutons

- ON
- OFF

## Ampoule

## Client\_Bluetooth1

## N'importe quel composant

## Interface

quand Client\_Bluetooth1 ▾ .BluetoothError

Nom fonction message

faire

appeler Client\_Bluetooth1 ▾ .Octets disponibles pour le réception

appeler Client\_Bluetooth1 ▾ .Se connecter

adresse

appeler Client\_Bluetooth1 ▾ .Se connecter avec UUID

adresse

uuid

appeler Client\_Bluetooth1 ▾ .Déconnecter

appeler Client\_Bluetooth1 ▾ .L'appareil est appareillé

adresse

appeler Client\_Bluetooth1 ▾ .RecevoirOctetSignéNuméro 1

Retournons 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

**Lampe** Screen1 • Ajouter écran... Supprimer écran Designer Blocs

**Palette**

Interface utilisateur

Disposition

- Arrangement horizontal ?
- HorizontalScrollArrangement ?
- Arrangement tableau ?
- Arrangement vertical ?
- VerticalScrollArrangement ?

Média

Dessin et animation

Capteurs

Social

Stockage

Connectivité

LEGO® MINDSTORMS®

Expérimental

**Interface**

Afficher les composants cachés dans l'interface

Cochez pour voir un aperçu sur un appareil de taille tablette.

TELECOMMANDE

ON OFF

**Composants**

- Screen1
  - Arrangement\_horizontal
- ZoneBoutons
  - ON
  - OFF
- Ampoule
- Client\_Bluetooth1

**Propriétés**

Arrangement\_horizontal1

Alignement horizontal

Gauche : 1 ▾

Alignement vertical

Haut : 1 ▾

Couleur de fond

■ Par défaut

Hauteur

Automatique...

Largeur

Automatique

Remplir parent

pixels

percent

Annuler OK

On change le nom de la zone

Composants cachés dans l'interface  
voir un aperçu sur un appareil de taille tablette.

**Renommer composant**

Ancien nom:

Nouveau nom:

**Composants**

- Screen1
  - Arrangement\_horizontal1
- ZoneBoutons
  - ON
  - OFF
- Ampoule
- Client\_Bluetooth1

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

The image shows a software development interface for a mobile application named "Lampe". The interface is divided into several panels:

- PaLETTE**: A list of UI components. The "Sélectionneur de liste" component is highlighted with a red box. Other components include Bouton, Case à cocher, Sélectionneur de date, Image, Label, Vue liste, Notificateur, Zone texte mot de passe, Ascenseur, Curseur animé, Zone de texte, Sélectionneur temps, and Afficheur Web.
- Interface**: The main workspace showing a mobile device mockup. The mockup has a status bar at the top with the time 9:48 and icons for Wi-Fi, signal, and battery. Below the status bar is a header labeled "TELECOMMANDE". A red box highlights a text input field labeled "Texte pour Sélectionneur\_de\_liste1" within the mockup. Below the text field are two buttons labeled "ON" and "OFF".
- Composants**: A tree view of the application's components. The hierarchy is: Screen1 > liste\_bluetooth > Sélectionneur\_de\_liste. Other components include ZoneBoutons, ON, OFF, Ampoule, and Client\_Bluetooth1.
- Propriétés**: A panel showing the properties of the selected "Sélectionneur\_de\_liste1" component. Properties include: Couleur de fond (Par défaut), Éléments de la chaîne (empty text field), Activé (checked), Gras (unchecked), Italique (unchecked), Taille de police (14.0), Type de police (Par défaut), Hauteur (Automatique...), and Largeur (Automatique...).

Image  
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 cachés dans l'interface  
pour voir un aperçu sur un appareil de taille tablette.



**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

## Blocs

## ✚ Incorporé

■ Contrôle

■ Logique

■ Math

■ Texte

■ Listes

■ Couleurs

■ Variables

■ Procédures

## ✚ Screen1

✚ liste\_bluetooth

✚ liste\_appareils

✚ ZoneBoutons

■ ON

■ OFF

■ Ampoule

## Interface

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

## Blocs

## Incorporé

- Contrôle
- Logique
- Math
- Texte
- Listes
- Couleurs
- Variables
- Procédures

## Screen1

- liste\_bluetooth
  - liste\_appareils
- ZoneBoutons
  - ON
  - OFF
- Ampoule

Client\_Bluetooth1

## N'importe quel composant

Renommer

Supprimer

## Interface

appeler Client\_Bluetooth1 ▾ .Envoyer octets  
liste

appeler Client\_Bluetooth1 ▾ .Envoyer texte  
texte

Client\_Bluetooth1 ▾ . Adresses et noms ▾

Client\_Bluetooth1 ▾ . Disponible ▾

Client\_Bluetooth1 ▾ . Codage caractères ▾

mettre Client\_Bluetooth1 ▾ . Codage caractères ▾ à

Client\_Bluetooth1 ▾ . Octet séparateur ▾

mettre Client\_Bluetooth1 ▾ . Octet séparateur ▾ à

Client\_Bluetooth1 ▾ . Activé ▾

Client\_Bluetooth1 ▾ . Octet supérieur en premier ▾

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

## Blocks

## Built-in

Control

Logic

Math

Text

Lists

Colors

Variables

Procedures

## Screen1

liste\_bluetooth

liste\_appareils

ZoneBoutons

ON

## Viewer

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 

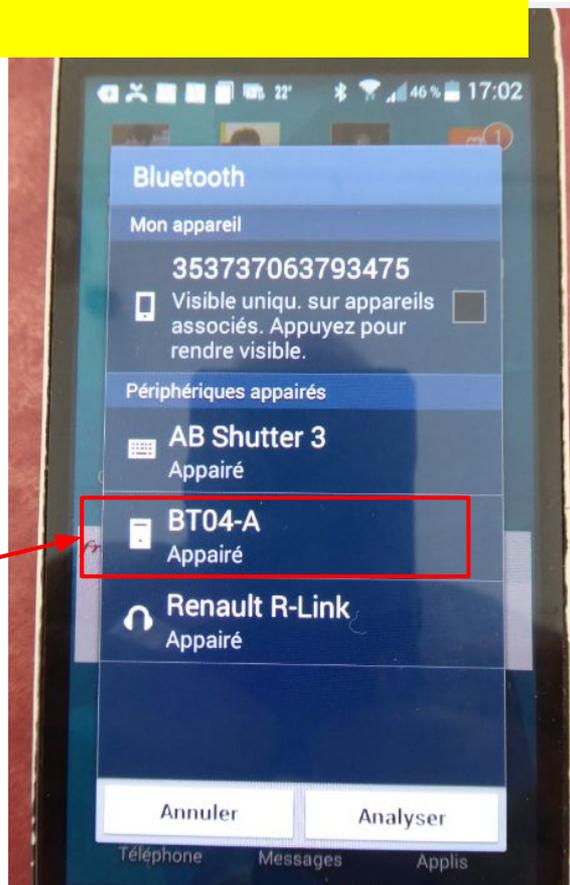
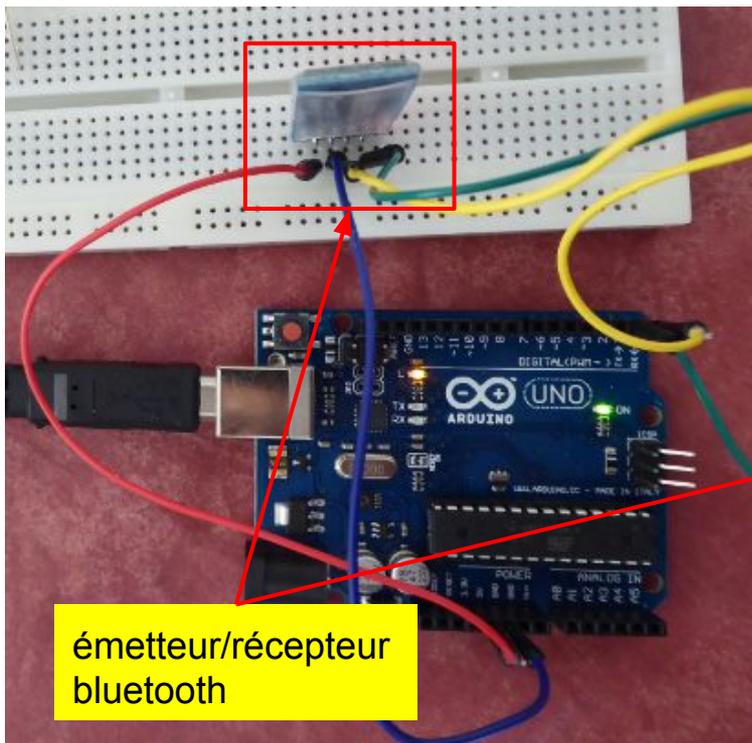
set Ampoule ▾ .Picture ▾ to "lampeoff.jpg"

when ON ▾ .Click

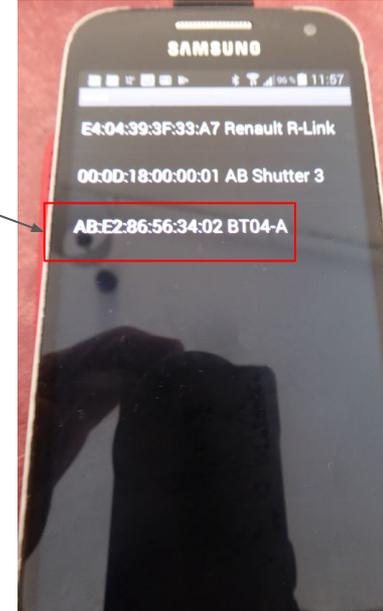
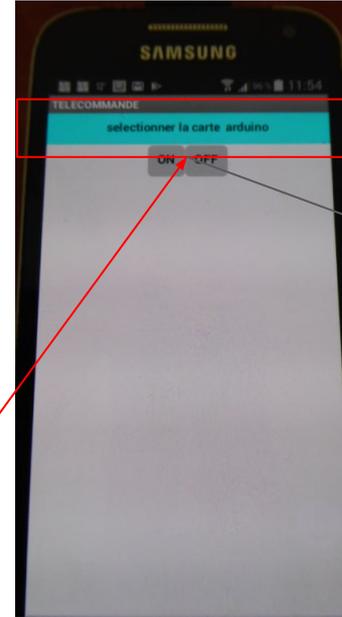
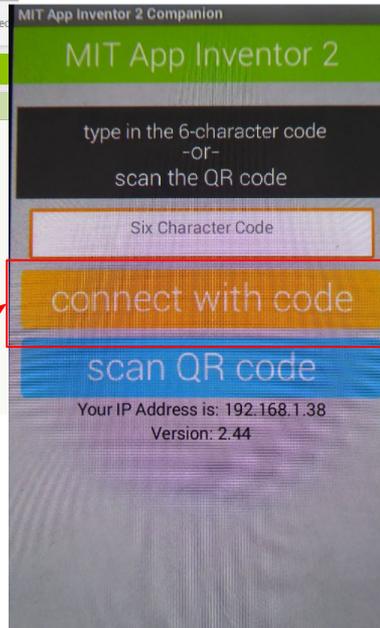
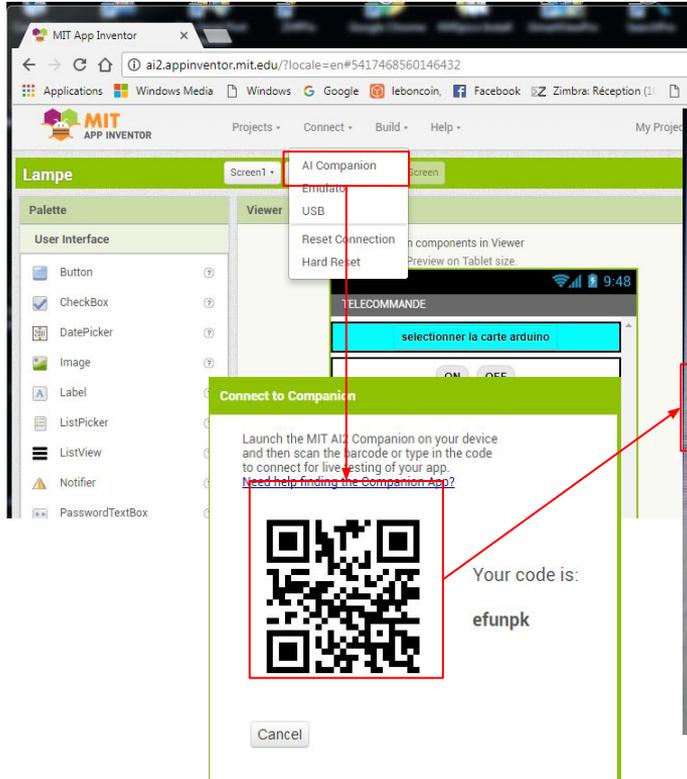
do set ON ▾ .BackgroundColor ▾ to set OFF ▾ .BackgroundColor ▾ to 

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

The image shows the Xamarin Designer interface for a mobile application. The top bar includes the title "Lampe" and navigation buttons for "Screen1", "Add Screen ...", and "Remove Screen". On the right, there are "Designer" and "Blocks" tabs.

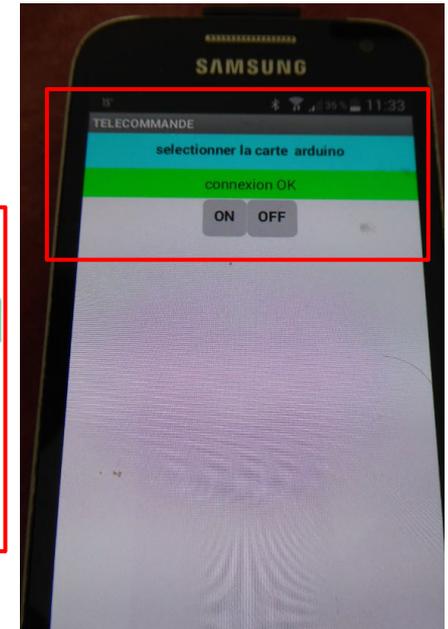
The interface is divided into several panels:

- Palette (User Interface):** Lists various UI components such as Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, TextBox, TimePicker, and WebViewer. The "Label" component is currently selected and highlighted in green.
- Viewer:** Displays a preview of the application screen. The screen has a title bar "TELECOMMANDE" and a status bar showing signal strength, Wi-Fi, and the time 9:48. Below the title bar, there is a cyan button labeled "selectionner la carte arduino". Below that, a horizontal zone is highlighted with a red box, containing a label "état de la connexion". At the bottom of the screen, there are "ON" and "OFF" buttons.
- Components:** A tree view showing the hierarchy of UI elements. The selected "Label" component is highlighted in green. The tree includes: Screen1, liste\_bluetooth, liste\_appareils, Zone\_etat\_connexion (containing etat\_connexion), ZoneBoutons (containing ON, OFF), Ampoule, and Client\_Bluetooth1.
- Properties:** A panel for configuring the selected "etat\_connexion" label. It shows various properties: BackgroundColor (None), FontBold (checkbox), FontItalic (checkbox), FontSize (14.0), FontTypeface (default), HTMLFormat (checkbox), HasMargins (checked), Height (Automatic...), and Width (Automatic...).

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 connexion est établie

```
when liste_appareils .BeforePicking  
do set liste_appareils . Elements to Client_Bluetooth1 . AddressesAndNames
```

```
when liste_appareils .AfterPicking  
do set liste_appareils . Selection to call Client_Bluetooth1 .Connect  
address liste_appareils . Selection  
if Client_Bluetooth1 . IsConnected  
then set Zone_etat_connexion . BackgroundColor to #00FF00  
set etat_connexion . Text to "connexion OK"
```



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 smartphone
  //-----
  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.

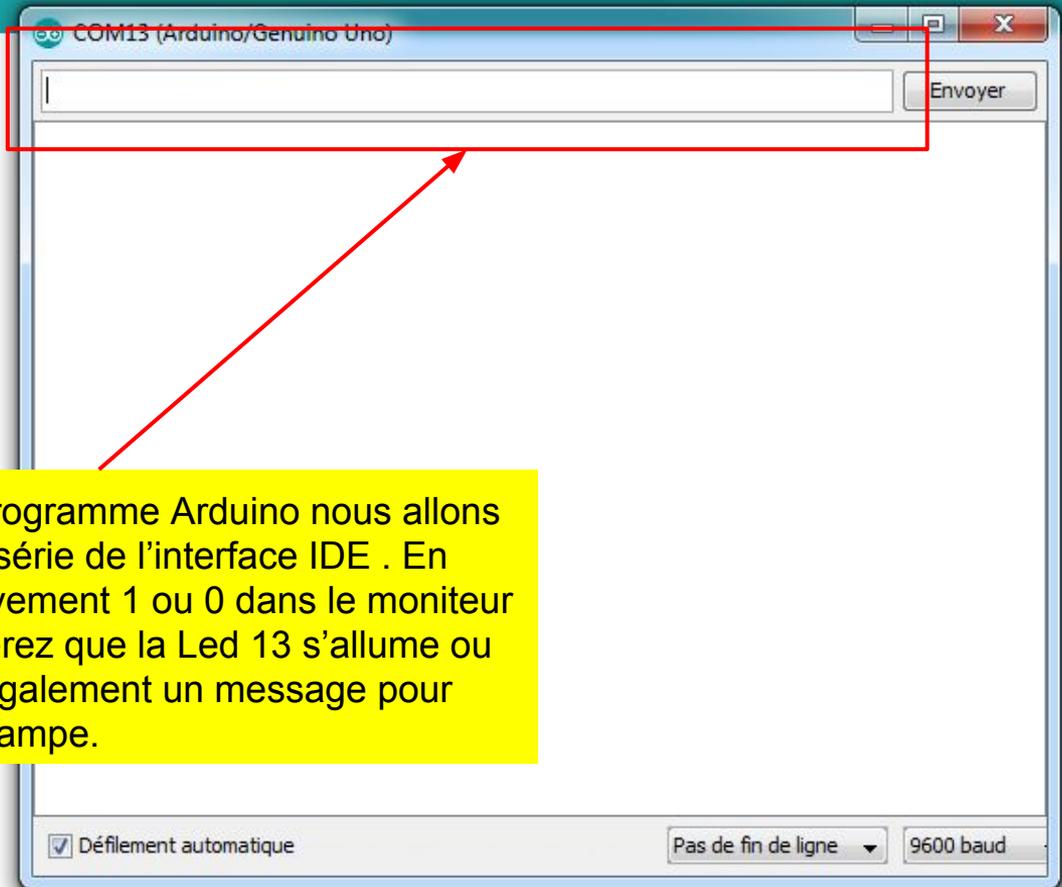
## Serial.parseInt

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
11 void action()
12 {
13   switch (etat_lampe)
14   {
15     case 0:
16       digitalWrite(LED, LOW);
17       Serial.println("eteinte");
18       break;
19     case 1:
20       digitalWrite(LED, HIGH);
21       Serial.println("allumée");
22       break;
23   }
24 }
25 }
26 //-----
27 void loop()
28 {
29   //-----
30   // reception du message
31   //-----
32   while (Serial.available())
33     etat_lampe = Serial.parseInt();
34   Serial.println(etat_lampe);
35   action();
36 }
37 }
38 }
```



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 \$

```
16 digitalWrite(LED, LOW);
17 Serial.println("eteinte");
18 break;
19 case 1:
20 digitalWrite(LED, HIGH);
21 Serial.println("allumee");
22 break;
23
24 }
25 }
26 //-----
27 void loop()
28 {
29 //-----
30 // reception du message depuis le smarphone
31 //-----
32 while (Serial.available() > 0) {
33 etat_lampe = Serial.parseInt();
34 Serial.println(etat_lampe);
35 action();
36
37 }
38 }
```

COM13 (Arduino/Genuino Uno)

Envoyer

1  
allumee  
0  
eteinte

Défilement automatique    Pas de fin de ligne    9600 baud

Téléversement terminé

ATTENTION : La catégorie 'Device Control, Signal Input,

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)

A Scratch code block with a tan background and a purple connector on the right. The text reads "initialize global message to" followed by a pink text box containing a single space character " ".

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

A Scratch code block with a tan background and a purple connector on the right. The text reads "when OFF .Click" followed by a "do" block containing four "set" blocks: "set OFF . BackgroundColor to" with a red color swatch, "set ON . BackgroundColor to" with a grey color swatch, "set Ampoule . Picture to" with a pink text box containing "lampeoff.jpg", and "set global message to" with a pink text box containing "0".A Scratch code block with a tan background and a purple connector on the right. The text reads "when ON .Click" followed by a "do" block containing four "set" blocks: "set ON . BackgroundColor to" with a green color swatch, "set OFF . BackgroundColor to" with a grey color swatch, "set Ampoule . Picture to" with a pink text box containing "lampeon.jpg", and "set global message to" with a pink text box containing "1".

Enfin ajoutons l'envoi du message via bluetooth

```
when OFF .Click
do
  set OFF . BackgroundColor to [red]
  set ON . BackgroundColor to [grey]
  set Ampoule . Picture to [lampeoff.jpg]
  set global message to [0]
  call Client_Bluetooth1 .SendText
  text [get global message]

when ON .Click
do
  set ON . BackgroundColor to [green]
  set OFF . BackgroundColor to [grey]
  set Ampoule . Picture to [lampeon.jpg]
  set global message to [1]
  call Client_Bluetooth1 .SendText
  text [get global message]
```

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'impulsion
go = false;     //lancement
```

```
    //fin du programme
```

```
}
```

# Partie 2: programme Appinventor

The screenshot displays the App Inventor web interface, divided into four main sections:

- Palette:** A list of UI components under the "User Interface" category, including Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, TextBox, TimePicker, and WebViewer. Below this are sections for "Layout" and "Media".
- Viewer:** A central workspace showing a mobile app preview. At the top, there are checkboxes for "Display hidden components in Viewer" and "Check to see Preview on Tablet size.". The app preview shows a screen titled "Screen1" with a status bar at the top displaying signal strength, Wi-Fi, battery, and the time 9:48. The app content includes:
  - A title bar with the text "liste des appareils".
  - A red text label: "Aucun appareil connecté".
  - A list view (currently empty).
  - A "GO" button.
  - A text input field.
  - Buttons labeled "QUIT" and "RAZ".
  - A label "Compteur" above a numeric display showing "0".
- Components:** A tree view of the app's component hierarchy. The root is "Screen1", which contains:
  - HorizontalArrangement1 (containing ListPicker1)
  - VerticalArrangement1 (containing Label1 and Label2)
  - HorizontalArrangement2 (containing the Flash component, which is highlighted in green)
  - HorizontalArrangement3 (containing an Intervalle component)
  - Go button
  - HorizontalArrangement4 (containing QUIT and espace buttons)
  - HorizontalArrangement5 (containing RAZ button)
  - HorizontalArrangement6 (containing the numeric display)
- Properties:** A panel for configuring the selected "Flash" component. It includes:
  - Flash: BackgroundColor (Default), Enabled (checked), FontBold (checked), FontItalic (unchecked), FontSize (12), FontTypeface (default), Height (Fill parent...), Width (Fill parent...), Hint (saisir le nombre de flash), MultiLine (unchecked), NumbersOnly (checked).

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 green
set Clock1 . TimerEnabled to true
set Clock1 . TimerInterval to 500
call raz
else set Label1 . Text to "Deconnecté"
set Label1 . TextColor to red
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 #00FF00
```

```
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

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 une communication 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