

# Comment commander une lampe à distance ?



NOM :

Prénom :

Classe :

## Ma première application

 <https://tinyurl.com/yagajxgs>



App INVENTOR est ....

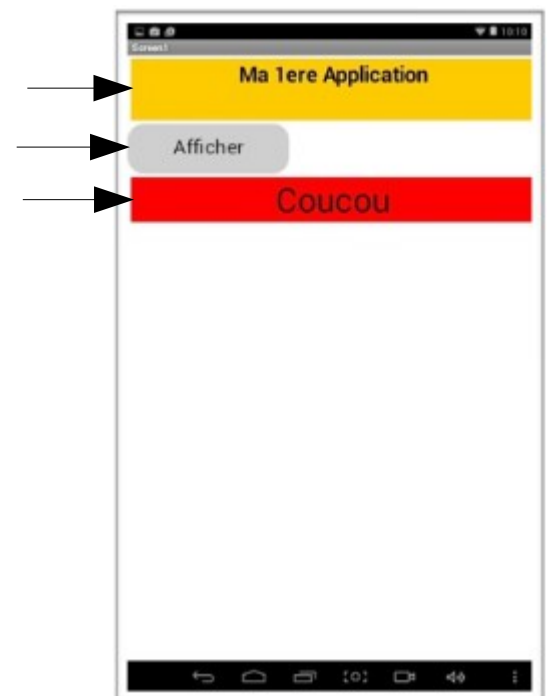


 <http://appinventor.mit.edu/>

Son interface de développement comprend deux parties.  
Lesquelles ? Et Pourquoi ?



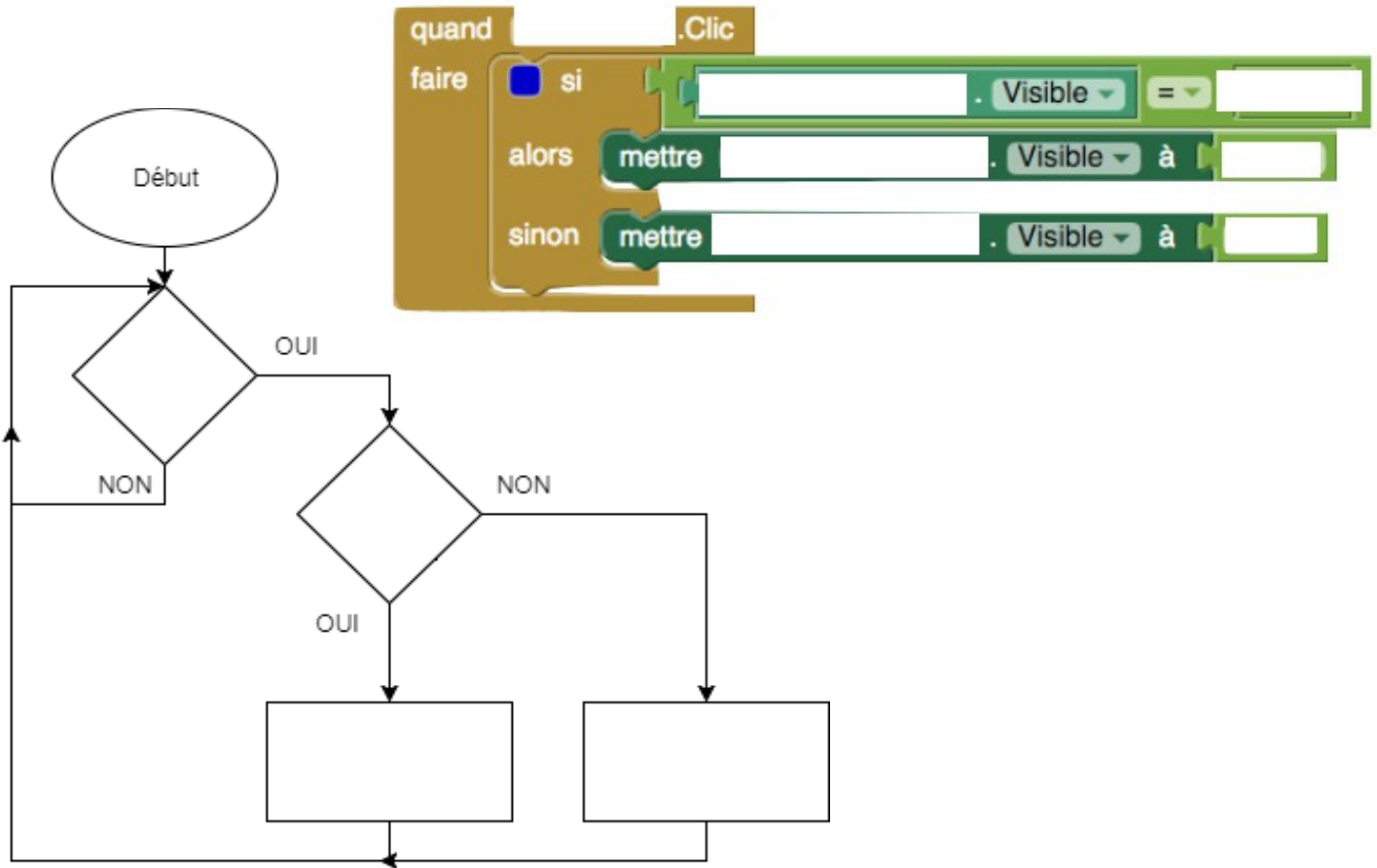
Ma première application consiste à .....



Les composants utiles pour la réaliser sont :

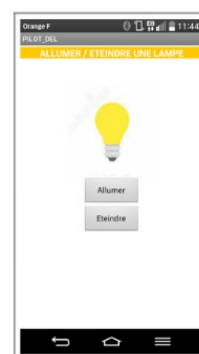
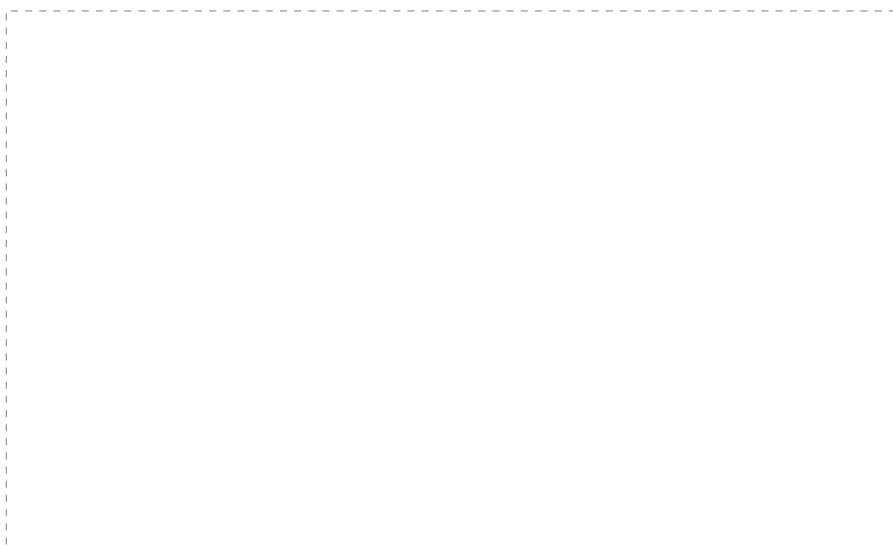
A l'aide du tableau Evènements/Actions, je complète l'algorithme et le script :

Evènements	Actions
Zone_à_afficher.visible = Vrai ?	Mettre Zone_à_afficher.visible à VRAI
Bouton1.clic=Vrai ?	Mettre Zone_à_afficher.visible à FAUX



## Simulation de la commande de la lampe

J'explique sous forme d'un texte le fonctionnement de l'application.



Je repère et je désigne les composants utilisés.



J'indique quelle est l'image qui doit être visible au démarrage de l'application et celle qui ne l'est pas. Je justifie ma réponse.

Empty dashed box for justification.

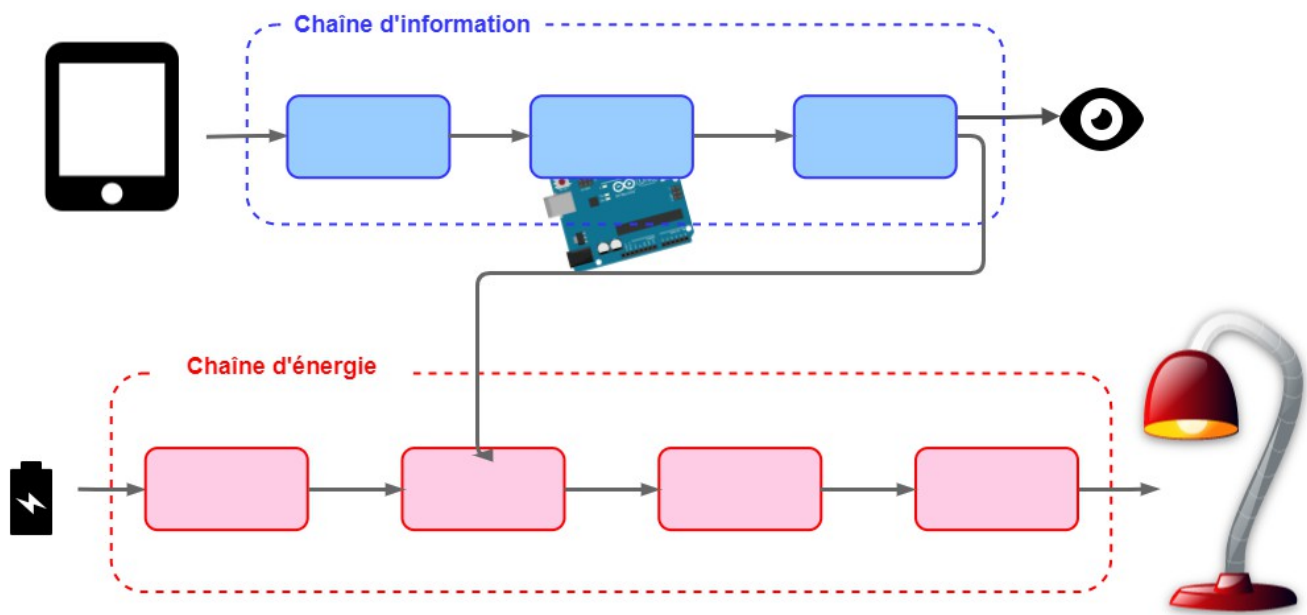
Je complète les deux scripts qui suivent.

```
quand BP_ON .Clic
faire
mettre [ ] . Visible à [ ]
mettre [ ] . Visible à [ ]
```

```
quand BP_OFF .Clic
faire
mettre [ ] . Visible à [ ]
mettre [ ] . Visible à [ ]
```

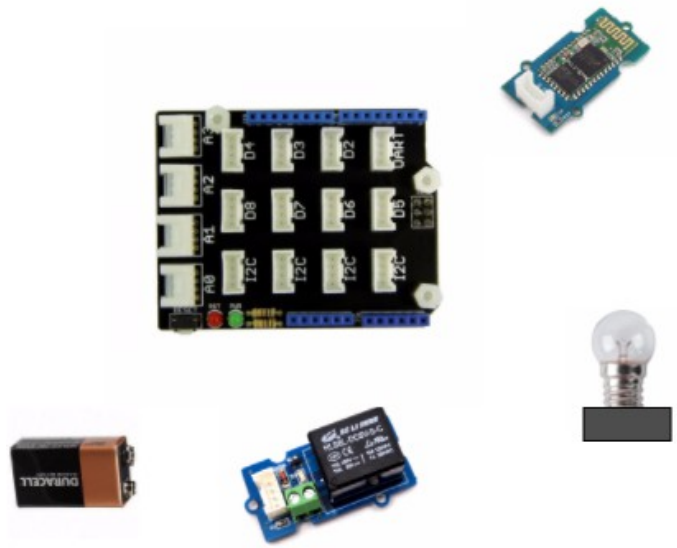
# Mise en oeuvre de la lampe Bluetooth

Je complète la chaîne d'informations et la chaîne d'énergie de mon système de lampe BLUETOOTH.



Je représente mon câblage et complète le tableau en associant un port/composant.

PORTS	COMPOSANTS
D2	
D3	
D4	
D5	
D6	
D7	
D8	
D9	



## Programmation (APPINVENTOR)

Je résume à quoi servent ces deux scripts de mon application.

```

quand BP_Connecte .Après prise
faire
mettre BP_Connecte . Activé à appeler Client_Bluetooth1 .Se connecter
adresse BP_Connecte . Sélection
mettre BP_Connecte . Visible à faux
mettre BP_Deconnecte . Visible à vrai
    
```

```

quand BP_Deconnecte .Clic
faire
  appeler Client_Bluetooth1 .Déconnecter
  mettre BP_Connecte . Visible à vrai
  mettre BP_Deconnecte . Visible à faux

```

En m'aidant du tableau Evènements/actions, je complète les **algorithmes**.

```

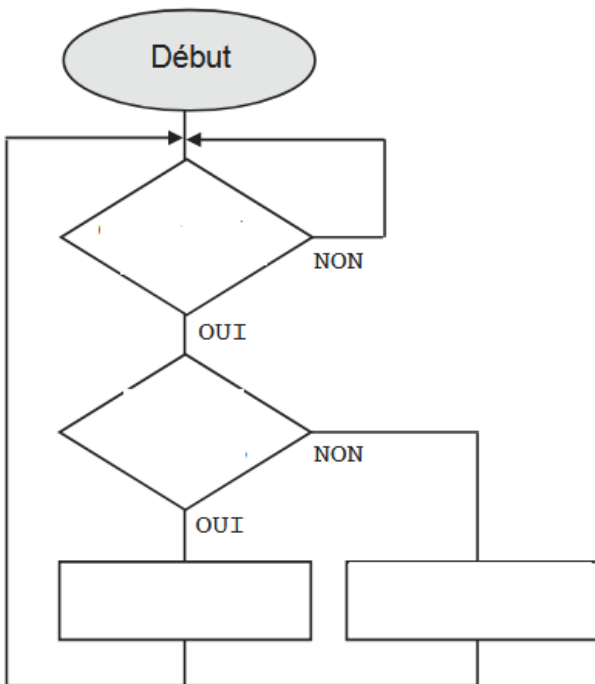
quand BP_ON .Clic
faire
  mettre LAMPE_ON . Visible à [ ]
  mettre LAMPE_OFF . Visible à [ ]
  appeler Client_Bluetooth1 .Envoyer texte
  texte joint [ ] [ ]

quand BP_OFF .Clic
faire
  mettre LAMPE_ON . Visible à [ ]
  mettre LAMPE_OFF . Visible à [ ]
  appeler Client_Bluetooth1 .Envoyer texte
  texte joint [ ] [ ]

```

Evènements	Actions
BP_ON.clic= vrai ?	Mettre Lampe_ON.visible à VRAI
BP_OFF.clic= vrai ?	Mettre Lampe_ON.visible à FAUX
	Mettre Lampe_OFF.visible à VRAI
	Mettre Lampe_OFF.visible à FAUX
	Envoyer texte « BP:0 »
	Envoyer texte « BP:1 »

## Programmation (mBLOCK)



Evènements	Actions
BT ; données disponibles = 1 ?	Mettre le port de la lampe à HAUT
Valeur reçue = « BP=1 » ?	Mettre le port de la lampe à BAS

```

UNO et Grove - générer le code
répéter indéfiniment
  si BT: données disponibles sur le port [ ] = [ ] alors
  si BT: recevoir la valeur de [ ] sur le port [ ] = [ ] alors
    Mettre [ ] sur la broche [ ] à [ ]
  sinon
    Mettre [ ] sur la broche [ ] à [ ]

```