

# APP INVENTOR



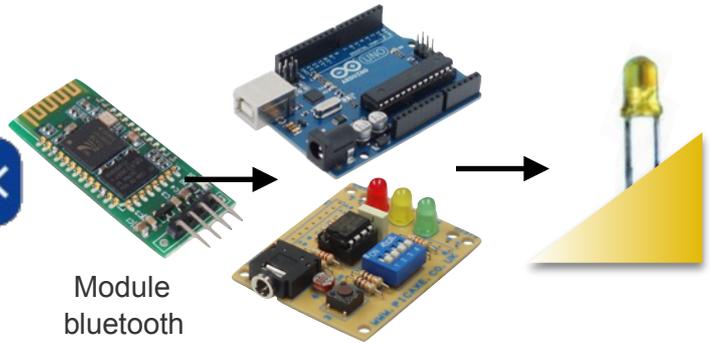
Application Android



Communication sans fil bluetooth



Module bluetooth



Interface programmable (Arduino ou Picaxe)



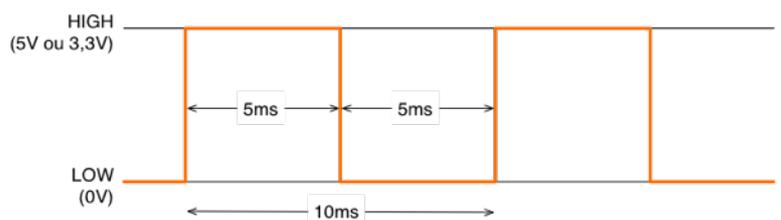
Dans cet exemple il s'agit, de piloter la puissance d'éclairage une del (variation de lumière) depuis le smartphone (application Android)

Pour cela nous allons utiliser les sorties « analogiques » (PWM) des microcontrôleurs (Picaxe ou Arduino) pour faire varier la puissance lumineuse de la del. Voir autre ressource pour davantage de précisions.

Côté application, nous allons utiliser un curseur qui permet de communiquer une valeur entre 0 et 255 (soit les 256 possibilités en 8 bits) en bluetooth.

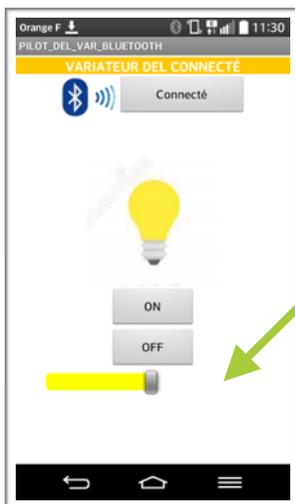
### Exemple avec PWM à 50%

La fréquence est de 100Hz, le rapport cyclique de 50%



Une sortie PWM sur un microcontrôleur est une sortie Numérique dont les signaux ont toujours une valeur LOW (0 logique) ou HIGH (1 logique). Mais le principe est de construire un signal qui est alternativement LOW et HIGH et de répéter très vite cette alternance en faisant varier la fréquence du signal.

Dans le cas d'une DEL, elle est alternativement allumée et éteinte mais le cycle est tellement rapide que la persistance rétinienne nous donne l'illusion d'une DEL allumée en permanence. Prenons par exemple une période de 10ms, soit une fréquence de 100Hz. Si la DEL est allumée pendant 5ms et éteinte pendant 5ms, comme sur la figure ci-contre, l'impression sera une luminosité de 50% de la luminosité maximum.



Curseur de 0 à 255 et initialement à 128 (valeur milieu)



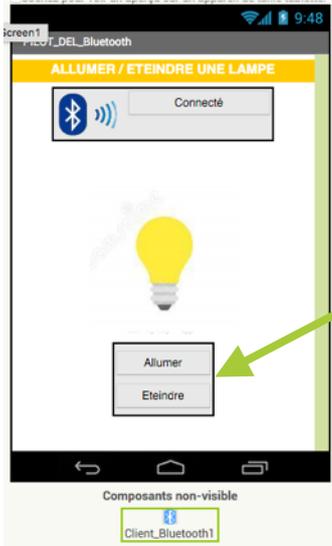
1  
Reprendre l'application Appli\_Lampe\_Bluetooth vue précédemment et l'enregistrer sous un nouveau nom de projet

Ajouter les images que l'on va utiliser par la suite :

2  
Image variation et nouveau logo de l'application



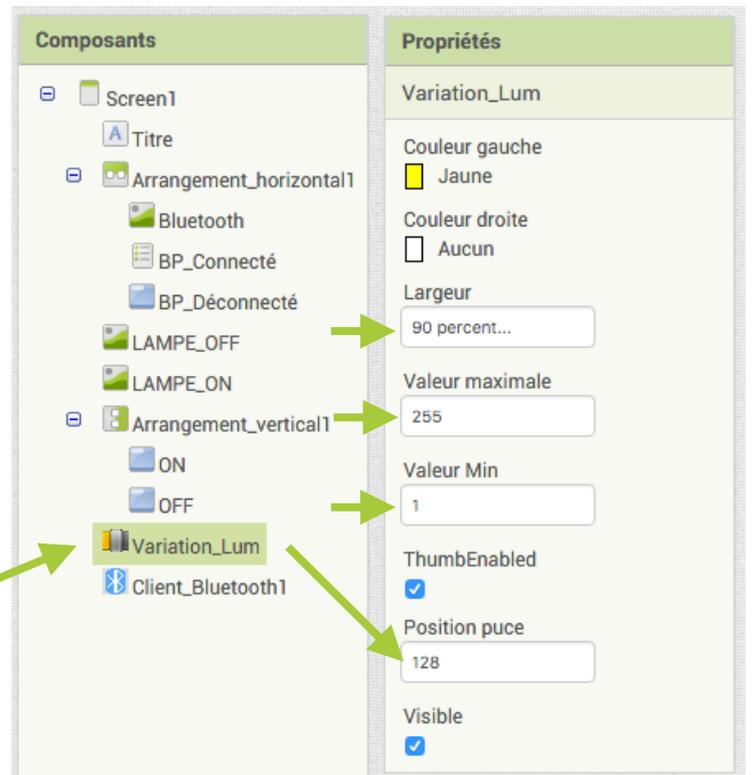
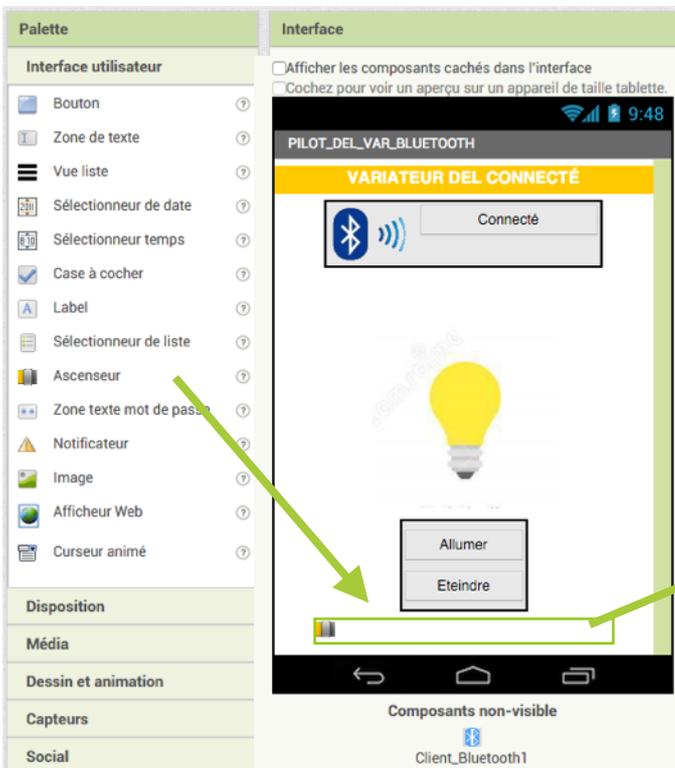
3  
Changez les propriétés de l'application : Logo et nomme l'application



4  
Côté interface design de l'application, il suffit d'ajouter un curseur (renommé « Variation\_Lum »): Ascenseur en dessous des 2 boutons « Allumer » « Eteindre ».



L'ascenseur doit avoir pour valeur max et min respectivement 255 et 1 (car nous sommes en 8 bits, voir tableau en bas de page). Egalement il peut être initialisé en position milieu (Position puce) soit à 128.



Rapport Cyclique %	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
Valeur sur 8 bits	0	13	26	38	51	64	77	89	102	115	128	140	153	166	179	191	204	217	230	242	255

Côté programmation ... Quelques ajouts et modifications sont à réaliser pour communiquer les valeurs de l'ascenseur via le bluetooth

```

quand BP_Connecte .Avant prise
faire
mettre BP_Connecte . Eléments à Client_Bluetooth1 . Adresses et noms

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
  
```



La partie de code correspondant à la fonction bluetooth ne change pas

```
initialise global var_lum à 128
```



Initialiser une variable **var\_lum** à 128 (valeur milieu du curseur par défaut)

```

quand ON .Clic
faire
mettre LAMPE_ON . Visible à vrai
mettre LAMPE_OFF . Visible à faux
appeler Client_Bluetooth1 .Envoyer1Octet
nombre obtenir global var_lum
  
```



Le bouton « ON » envoie donc maintenant le contenu de la variable **var\_lum**

```

quand Variation_Lum .Position changée
Position pouce
faire
mettre global var_lum à arrondi Variation_Lum . Position pouce
appeler Client_Bluetooth1 .Envoyer1Octet
nombre obtenir global var_lum
  
```



Quand le curseur de l'ascenseur change de position :

Le Client\_Bluetooth envoie la position actuelle du curseur :

Soit une valeur **entière** entre 1 et 255, d'ou l'utilisation du bloc **arrondi**

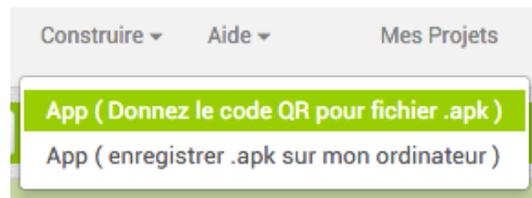
```

quand OFF .Clic
faire
mettre LAMPE_OFF . Visible à vrai
mettre LAMPE_ON . Visible à faux
appeler Client_Bluetooth1 .Envoyer1Octet
nombre 0
  
```



Côté bouton « OFF » rien ne change : il communique 0 via le client bluetooth

L'application est terminée, vous pouvez la tester et l'installer sur la tablette ou smartphone Android



Lien code à barre pour Ma1ereApplication



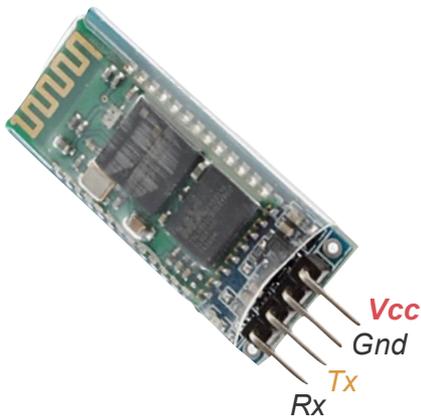
OK

Note: ce code à barre n'est valide que 2 heures. Regardez la FAQ pour voir des informations sur comment partager votre application avec les autres.



Il reste maintenant à réaliser un montage électronique qui permet de recevoir en bluetooth les valeurs de 1 à 255 générés par l'application.

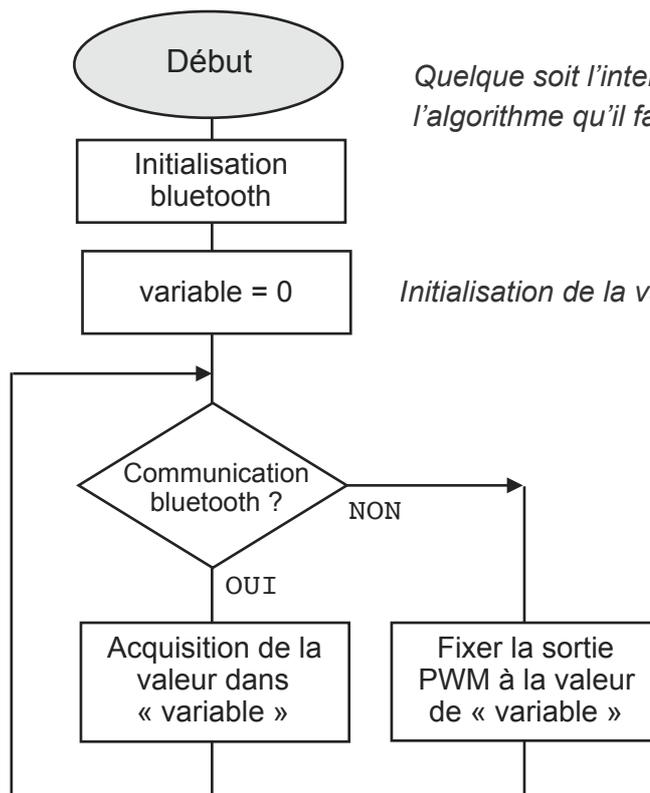
La solution la plus simple étant d'utiliser une interface programmable Picaxe ou Arduino et leurs sorties PWM associées



**Vcc** pour l'alimentation 3,3V ou 5V

**Gnd** pour l'alimentation : 0V

**Tx et Rx** pour la communication



Quelque soit l'interface choisie, voici l'algorithme qu'il faut programmer

Initialisation de la variable à 0

# Exemple avec un microcontrôleur Picaxe sous Blockly



Projets ▾ Édition ▾ Paramètres ▾ PICAXE ▾ Mode: Blocks Code

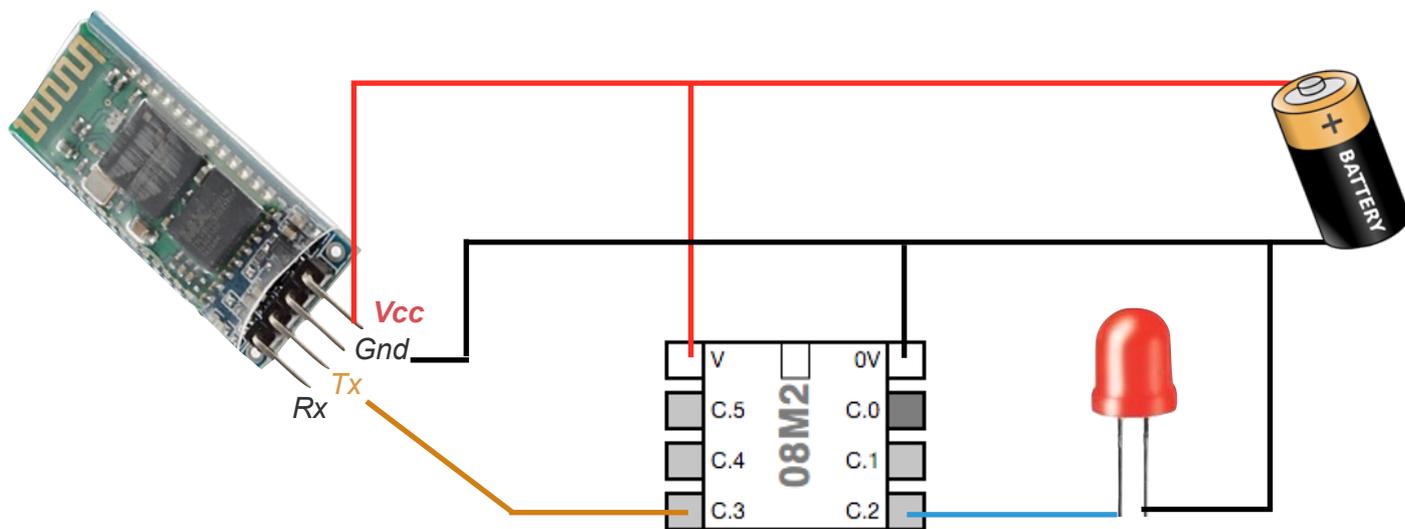
Simulateur × Blocks PICAXE BASIC Javascript XML

Sorties  
Entrées  
Délais  
Boucles  
Variables  
Maths  
Procédures  
Tâches  
Moteurs  
Liaison série  
Avancé

```

début
  BASIC setfreq m8
  répéter indéfiniment
    faire
      fixer varA à 0
      BASIC serin C.3, T9600_8, varA
      si varA <> 0
        faire
          signal pwm de periode 255 rapport cyclique varA sur C.2
      si varA = 0
        faire
          signal pwm de periode 255 rapport cyclique 0 sur C.2
    
```

Rapport Cyclique %	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
Valeur sur 8 bits	0	13	26	38	51	64	77	89	102	115	128	140	153	166	179	191	204	217	230	242	255



Ne pas oublier une résistance pour limiter le courant dans la del : 300 à 500 Ohms

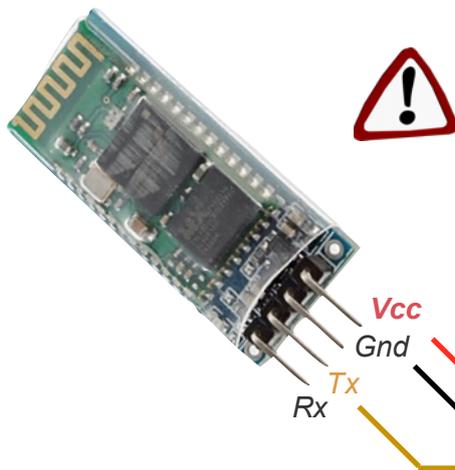
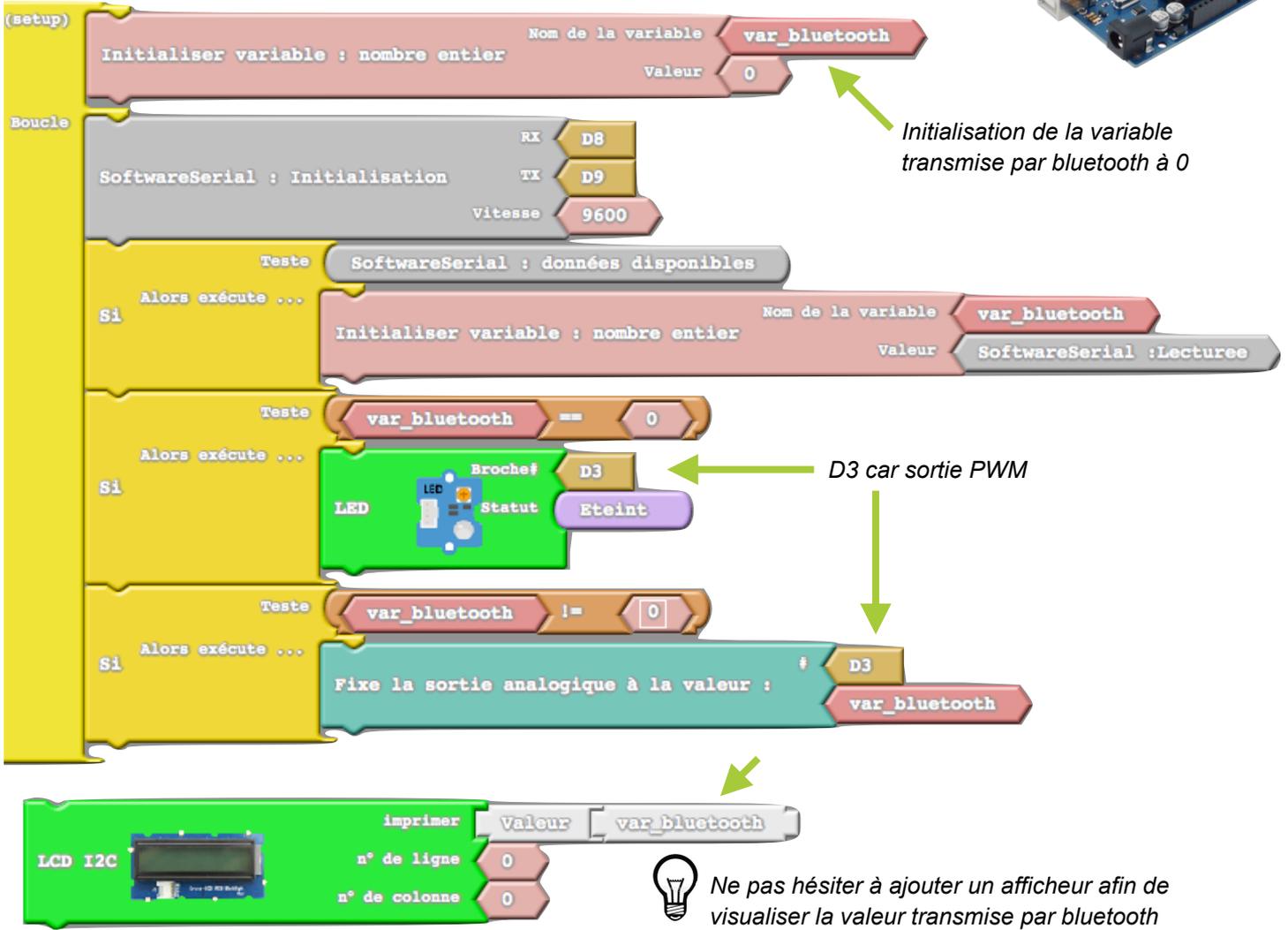
## PICAXE-28X1

Reset	1	28	Output 7
ULPWU / ADC 0 / In a0	2	27	Output 6
ADC 1 / In a1	3	26	Output 5
ADC 2 / In a2	4	25	Output 4 / hpwm D
ADC 3 / In a3	5	24	Output 3
Serial In	6	23	Output 2 / hpwm B
Serial Out	7	22	Output 1 / hpwm C
0V	8	21	Output 0
Resonator	9	20	+V
Resonator	10	19	0V
timer clk / Out c0 / In 0	11	18	In 7 / Out c7 / hserin / kb data
pwm 1 / Out c1 / In 1	12	17	In 6 / Out c6 / hserout / kb clk
hpwm A / pwm 2 / Out c2 / In 2	13	16	In 5 / Out c5 / spi sdo
spi sck / i2c scl / Out c3 / In 3	14	15	In 4 / Out c4 / i2c sda / spi sdi

## PICAXE-08M PICAXE-08M2

+V	1	8	0V
Serial In	2	7	Out 0 / Serial Out / Infraout
ADC 4 / Out 4 / In 4	3	6	In 1 / Out 1 / ADC 1
Infrain / In 3	4	5	In 2 / Out 2 / ADC 2 / pwm 2

# Exemple avec une interface Arduino sous Ardublock



Attention ici on utilise le port D3 car seuls les ports 3, 5, 6, 9, 10 et 11 peuvent fournir une sortie analogique (PWM). Ils sont repérés par le symbole :

Ne pas oublier une résistance pour limiter le courant dans la del : 300 à 500 Ohms

