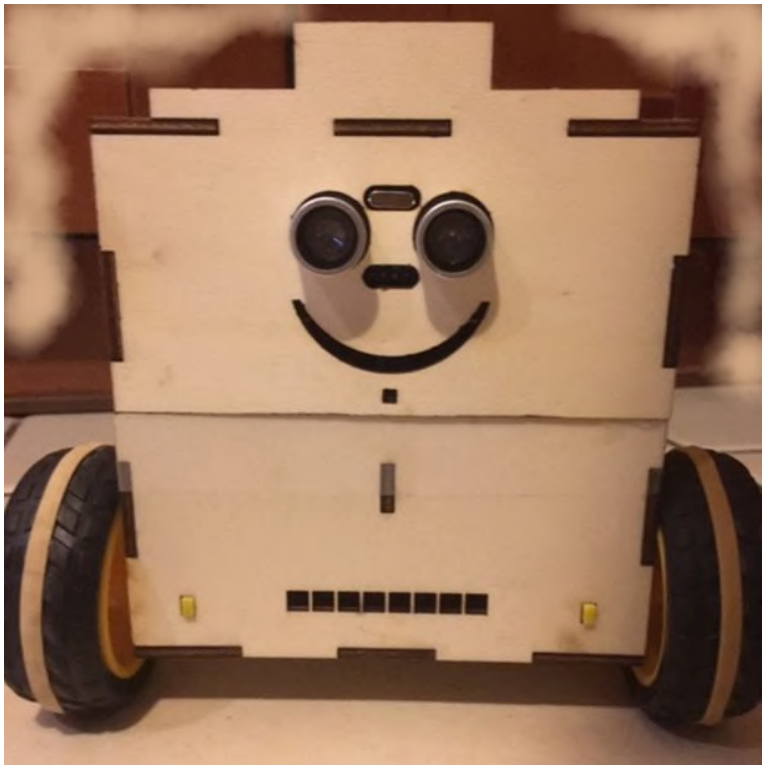


WHEELY_BOT V0

Arduino & App Inventor

RÉSUMÉ
Robot évitant les obstacles ou piloté par un smartphone Android

MS-56.
Bâtiment CFA 56
Fabriquer un objet connecté

TABLE DES MATIÈRES

WHEELY_BOT_V0

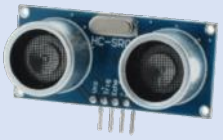
WheelyBot est un petit robot qui évite les obstacles. Il peut aussi être piloté par un smartphone sous Android.

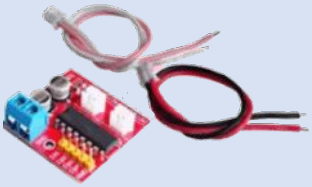
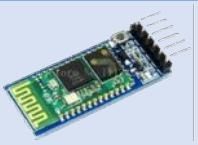
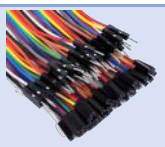
Objectifs pédagogiques :

- Dessin vectoriel ; Dessin 2D et 3D
- Découpe laser (Réalisée par leMS56 par manque de ce matériel au CFA);
- Algorithme ;
- Programmation ;
- Câblage électrique ;

La finalité est d'initier les apprentis aux nouvelles technologies en fabriquant un objet connecté tout en utilisant différents langages de programmation.

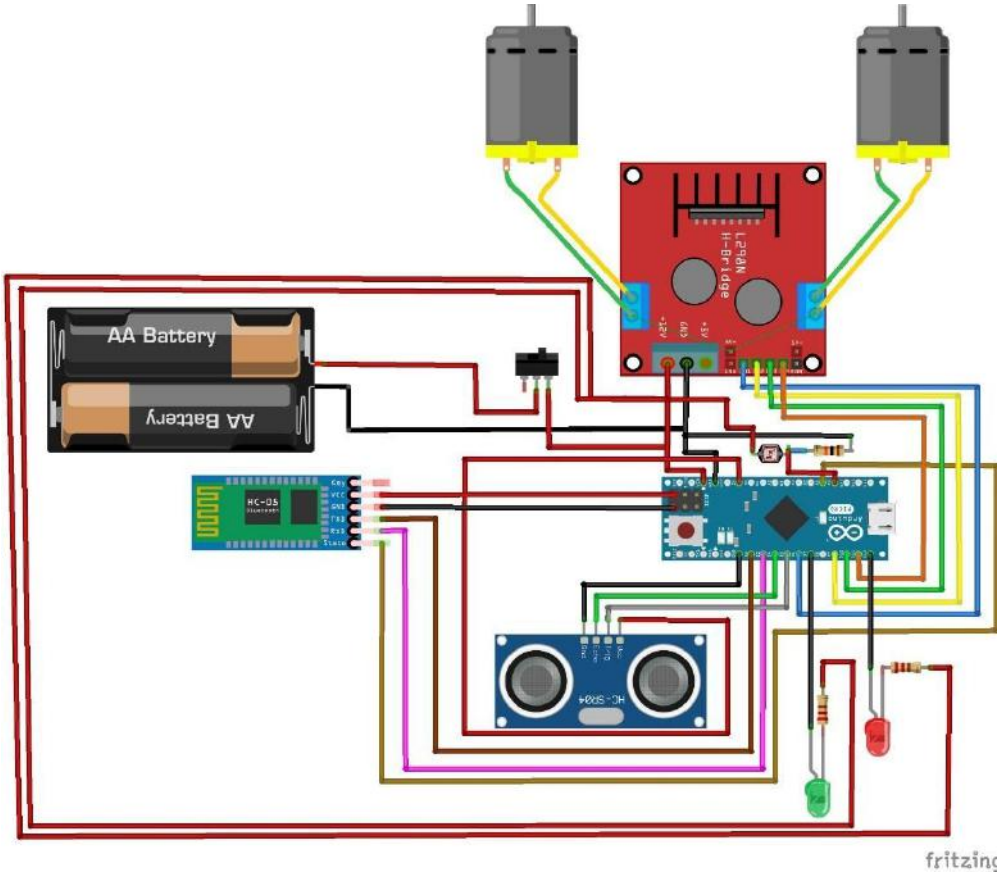
1. Le matériel :

Désignation	Description	Quantité	Lien
accu 18650		2	
Power bank		1	
Boitier accu 18650		1	lien Ali
moteurs + roues		2	lien 2
HC-SR04 (capteur ultrasons)		1	lien
interrupteur		1	lien

Module de pilotage des moteurs L298N PWM régulateur de vitesse		1	lien
Nano strong (carte Arduino)		1	lien
HC 05 (récepteur Bluetooth)		1	lien
LEDs		2	lien
résistances 220Ω		2	lien
Roulette (œil de taureau)		1	lien
photorésistance		1	lien
Résistance 10 kΩ		1	
Fils Dupont		plein	lien
Vis 2- 30mm		20	lien
Boulons		20	lien

2. Câblage :

A. Schéma de montage



B. Câblage Arduino

1) Port Digital

pin	élément	pin	élément
1		8	
2	RX	9	IN2(A)
3	TX	10	IN3(B)
4	écho	11	IN4(B)
5	trig	12	LED
6	IN1(A)	13	
7	LED		

2) Port Analog

pin	élément
A0	photorésistance
A1	État (State) BT

3. Arduino :

Arduino, et son synonyme **Genuino**, est une marque de cartes électroniques sur lesquelles se trouve un microcontrôleur (d'architecture Atmel AVR comme l'Atmega328p).

Les schémas de ces cartes électroniques sont publiés en licence libre. Cependant, certains composants, comme le microcontrôleur par exemple, ne sont pas sous licence libre.

Le microcontrôleur peut être programmé pour analyser et produire des signaux électriques, de manière à effectuer des tâches très diverses comme la domotique (le contrôle des appareils domestiques — éclairage, chauffage...), le pilotage d'un robot, de l'informatique embarquée, etc.

C'est une plateforme basée sur une interface entrée/sortie simple.

Arduino peut être utilisé pour construire des objets interactifs indépendants (prototypage rapide), ou bien peut être connecté à un ordinateur pour communiquer avec ses logiciels. Des informations sont fournies pour ceux qui souhaitent assembler ou construire une carte Arduino eux-mêmes (open source).

Arduino Uno

Arduino Nano

Nano V3.Strong

A. Arduino IDE

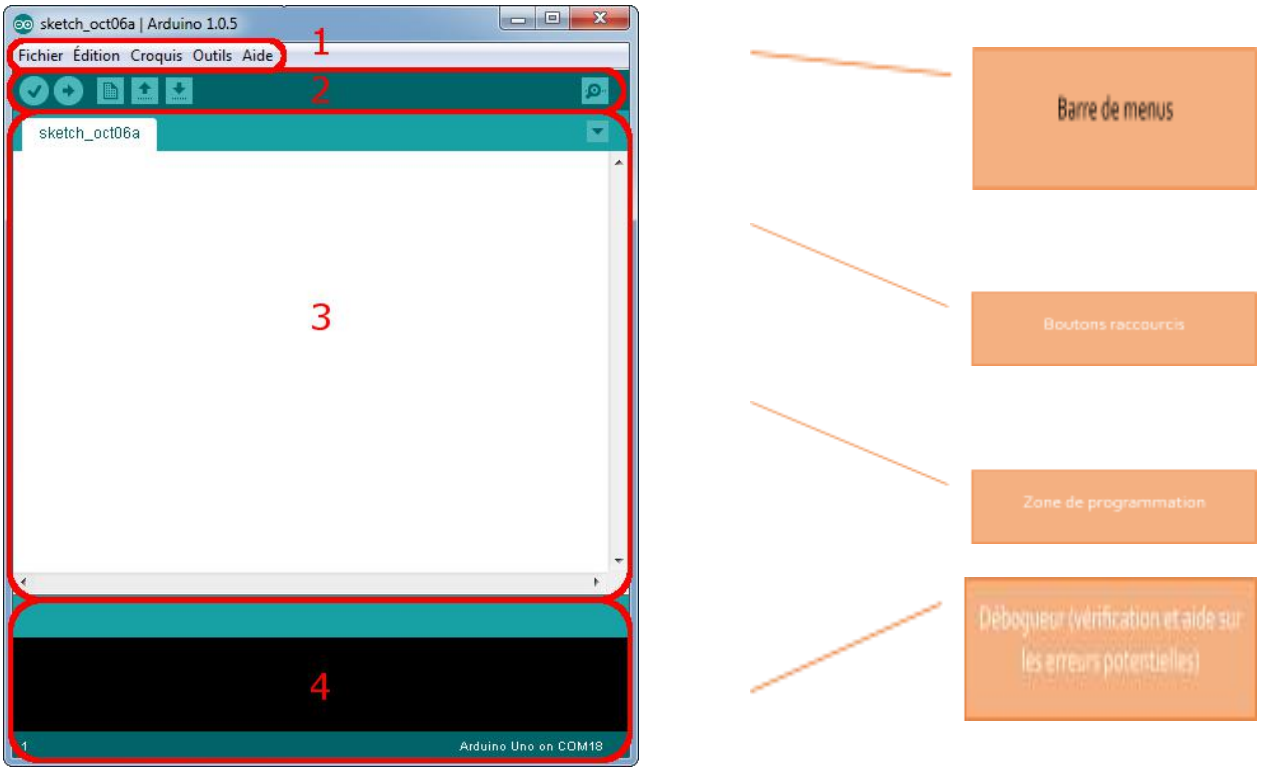
Le logiciel Arduino open source (IDE) facilite l'écriture de code et le téléchargement sur la carte. Il fonctionne sous Windows, Mac OS X et Linux. L'environnement est écrit en Java et basé sur Processing et autres logiciels open source.
Ce logiciel peut être utilisé avec n'importe quelle carte Arduino.

B. Logiciel Arduino

Un programme est une suite d'actions qui est exécutée par un système.

C'est une liste d'instructions que le processeur va exécuter dans l'ordre.

Le logiciel Arduino permet d'écrire, traduire et de charger le programme dans la carte.

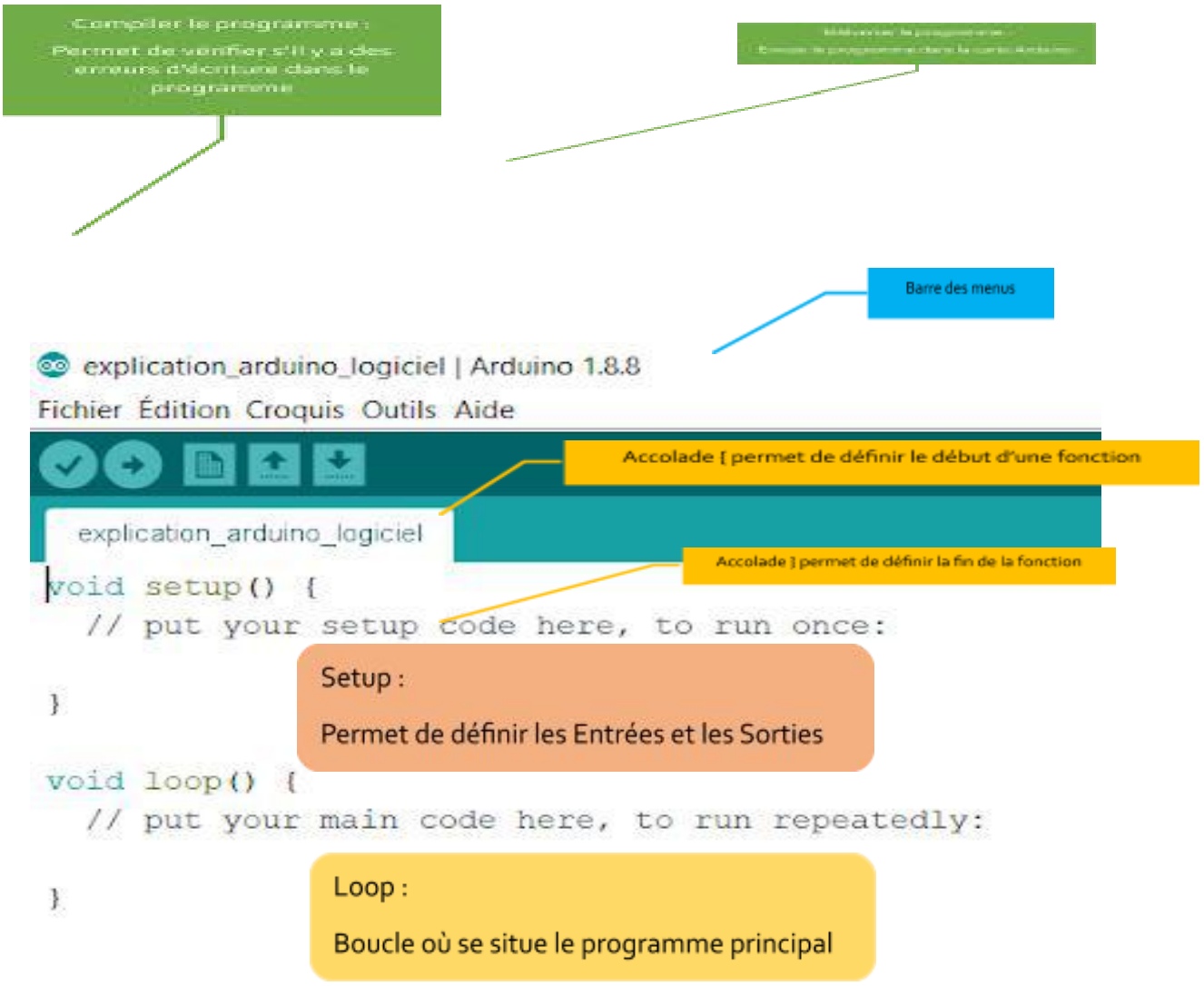


Le programme Arduino s'écrit dans la zone de programmation. Il comprend trois parties.

- Déclaration des variables, valeurs et noms à utiliser,
- Setup,
- Loop.

La première partie sert principalement à dire à la carte de **garder en mémoire quelques informations** qui peuvent être : l'emplacement d'un élément connecté à la carte, par exemple une LED en broche 7, ou bien une valeur quelconque qui sera utile dans le programme.

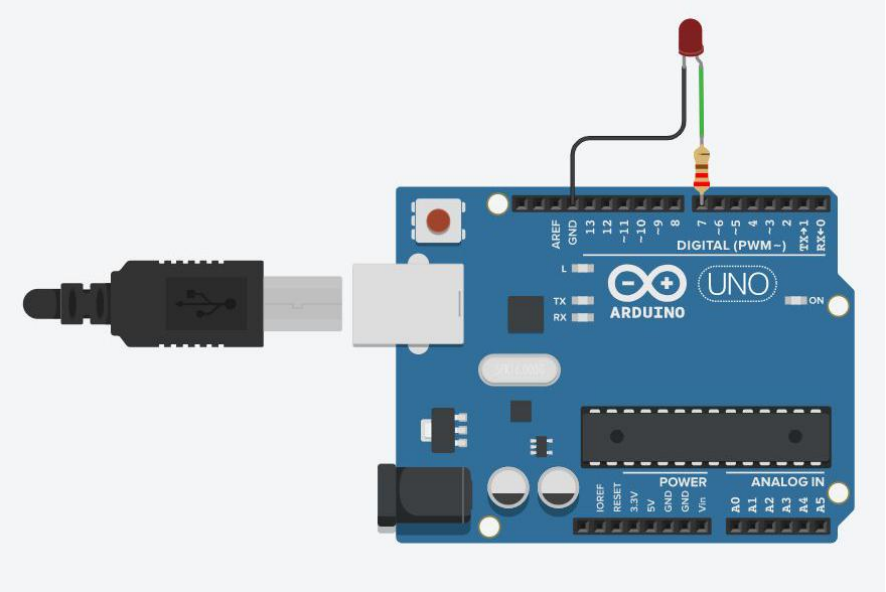
C. Interface



La deuxième partie « Setup » est le paragraphe où l'on va **initialiser certains paramètres** du programme. Par exemple, on va indiquer ce qu'elle devra faire de la LED qui est connectée sur sa broche 7. Elle n'est effectuée qu'une seule fois au lancement du programme.

La troisième partie est la **zone principale où se déroulera le programme**. Tout ce qui va être écrit dans cette zone sera exécuté par la carte ligne après ligne. Ces instructions se répètent en boucle tant que la carte est alimentée.
Par exemple, c'est ici qu'on pourra lui dire de faire allumer et éteindre la LED sur sa broche 7.

D. Premier programme : Connecter une lampe



Il faut définir sur quelle broche de la carte ARDUINO nous allons brancher la LED et lui donner le nom de « LED » pour la suite du programme

```
const int LED = 7; // broche 7 du micro-contrôleur se nomme maintenant : LED
```

Setup : Nous allons définir que la LED est une Sortie avec la fonction « PinMode (LED, OUTPUT) ; » (ne pas oublier le point-virgule avant de passer à une autre ligne)

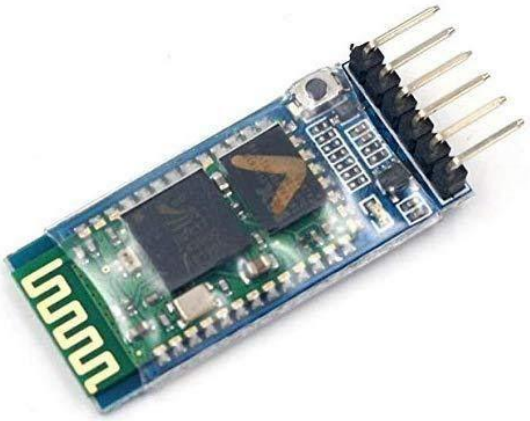
```
void setup() {
  pinMode(LED, OUTPUT); //LED est une broche de sortie
  }
```

DigitalWrite permet d'écrire 1 dans « LED » afin d'allumer la LED, HIGH correspond à la valeur 1
DigitalWrite permet d'écrire 0 dans « LED » afin d'éteindre la LED, LOW correspond à la valeur 0

```
void loop() {
  digitalWrite(LED, HIGH); //allumer LED
  delay(1000); // attendre 1 seconde
  digitalWrite(LED, LOW); // Eteindre LED
  delay(2000); // attendre 2 secondes
  }
```

Delay (2000) permet de réaliser une temporisation donnée en ms
Ici, 2000 ms soit 2 Secondes

E. Deuxième programme : Connecter le Bluetooth



1) Zone Déclaration

#include <SoftwareSerial.h> permet d'aller chercher une bibliothèque (un programme déjà fait) pour la configuration de la carte Bluetooth

```
#include <SoftwareSerial.h> // TX RX librairie software pour le bluetooth
```

softwareSerial monBT (2,3) ; permet de définir le nom de la carte Bluetooth : « MonBT » et de définir TX en 2 et RX en 3. Tx : Transmission, Rx : Réception des données.

```
SoftwareSerial monBT(2, 3);  // Connection au module BlueTooth HC05
```

Il faut créer une variable « valeur reçue » pour stocker les données envoyées par le téléphone via le Bluetooth (App Inventor).

```
int valeur_recue;
```

Définition de la broche A1 pour « bluetoothconnecte »de la carte nano afin de vérifier la connexion avec le téléphone.

```
#define bluetoothconnecte A1 // entrée pour vérifier la connection Bluetooth
```

2) Zone Setup

Nous définissons « bluetoothconnecte » comme une entrée.
Nous mettons la variable « valeur reçue » à O.

```
void setup() {
  // initialisation du module BlueTooth HC05
  pinMode(bluetoothconnecte, INPUT);
  valeur_recue = 0;       //  RAZ de toutes les sorties
```

monBT.begin(9600) permet de définir la vitesse de transmission (9600) entre le module Bluetooth et la carte Arduino. Le baud est l'unité de rapidité de modulation. Serial.begin(9600) permet de définir la vitesse de transmission entre la carte et le moniteur série. Le moniteur permet de visualiser les messages sur logiciel Arduino.

```
  monBT.begin(9600);
  Serial.begin(9600);
}
```

3) Zone Loop

While : « tant que »
Tant que la valeur lue sur Bluetoothconnecte est supérieure à 512, on écrit sur le moniteur « ..... ». Cela permet de vérifier la connexion.

```
void loop() {

  while (analogRead(bluetoothconnecte) > 512 ) {
    Serial.print("... ");
```

If : « Si »
Si monBT est « disponible » alors écrire sur le moniteur « Bluetooth CONNECTE » , « Octet reçu : », le contenu de la variable « valeur_ reçue » et enfin une tempo de 50 ms.

```
    if (monBT.available())
    {
      valeur_recue = monBT.read();
      Serial.println("      --BlueTooth CONNECTE--");
      Serial.print("octet recu : ");
      Serial.println(valeur_recue);
      delay(50);
    }
  }
}
```

F. Troisième programme : Commander une LED avec son téléphone et envoyer un message « c'est pas Versailles ici »

1) Zone déclaration

Définition du Bluetooth, déjà vue dans le paragraphe précédent.

```
#include <SoftwareSerial.h>
SoftwareSerial monBT(2, 3);

int valeur_recue;
#define bluetoothconnecte A1
```

Il faut définir la broche 12 de la carte ARDUINO. Nous allons brancher la LED et lui donner le nom de « ledPin » pour la suite du programme.

```
// Initialisation des constantes :
const int ledPin = 12;
```

« Unsigned long » définit un nombre positif de 0 à 4 294 967 295 qui va permettre de mémoriser le temps, on a donné le nom de « temps prévu » à cette variable, nous la mettons à zéro pour l'initialiser

« Const long » définit un nombre constant que nous avons appelé « Interval » et l'avons mis à 5000 ms soit 5 secondes.

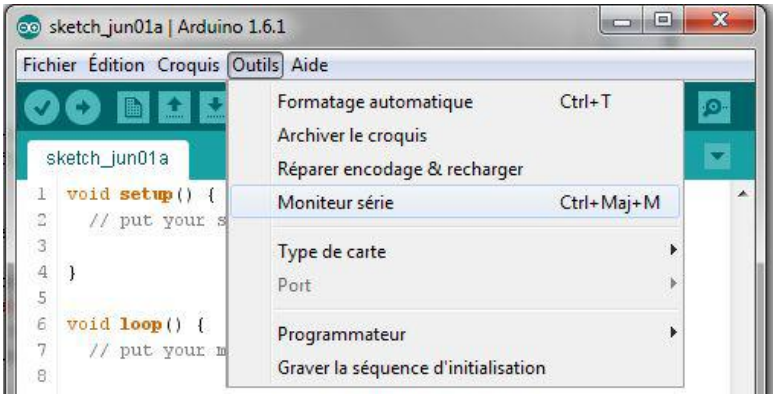
```
unsigned long temps_prevu = 0;
unsigned long temps_courant = 0;

const long interval = 5000;
```

Pour la suite, nous avons déjà vu l'ensemble des commandes :

2) Zone Setup

pinMode(Bluetoothconnecte, INPUT) : indique que « bluetooth connecte » est une entrée.
On initialise à zéro « la valeur reçue »
Vitesse de transmission 9600 bauds pour « monBT »
Vitesse de transmission 9600 bauds pour le moniteur
Définition de ledPin est une sortie pour la led

```
// le code dans cette fonction est exécuté une fois au début
void setup() {
  // initialisation du module BlueTooth HC05
  pinMode(bluetoothconnecte, INPUT);
  valeur_recue = 0;      //  RAZ de toutes les sorties
  monBT.begin(9600);
  Serial.begin(9600); // ouvre le port série en 9600 bps:
  // indique que la broche ledPin est une sortie :
  pinMode(ledPin, OUTPUT);
  valeur_recue = 0;
}
```



3) Zone Loop

```
// le code dans cette fonction est exécuté en boucle
void loop(){

  if (analogRead(bluetoothconnecte) > 512 ) {
    Serial.println("Connecté");
    if (monBT.available())
    { valeur_recue = monBT.read();
      Serial.println("--BlueTooth CONNECTE--");
      Serial.print("octet recu : ");
      Serial.println(valeur_recue);
      temps_prevu = millis();
    }
    Serial.print("valeur recue est de :");
    Serial.println(valeur_recue);
  }

      temps_courant = millis();
```

Nous avons remplacé le while par un if. Si la valeur de Bluetoothconnecte est > 512 alors on écrit sur le moniteur « connecté »
Si monBT est « disponible » alors écrire sur le moniteur « Bluetooth CONNECTE »
« Octet reçu : »
Affichage de la valeur : « valeur_reçue »
Tempo de 50 ms

Fonction millis()
Renvoie le nombre de millisecondes écoulées depuis que la carte Arduino a commencé à exécuter le programme actuel. On écrit ce temps dans la variable « Temps courant »

```
if (valeur_recue == 1) {
  // on allume la LED
  Serial.println("led allumée");
  digitalWrite(ledPin, HIGH);
}
```

Nous avons défini 1 pour allumer la lampe, cette valeur est envoyée par le Bluetooth.
Si valeur reçue est égale à 1
On écrit sur le moniteur « led allumée »
On met à la valeur haute la led, on met à 1 la Led pour l'allumer.

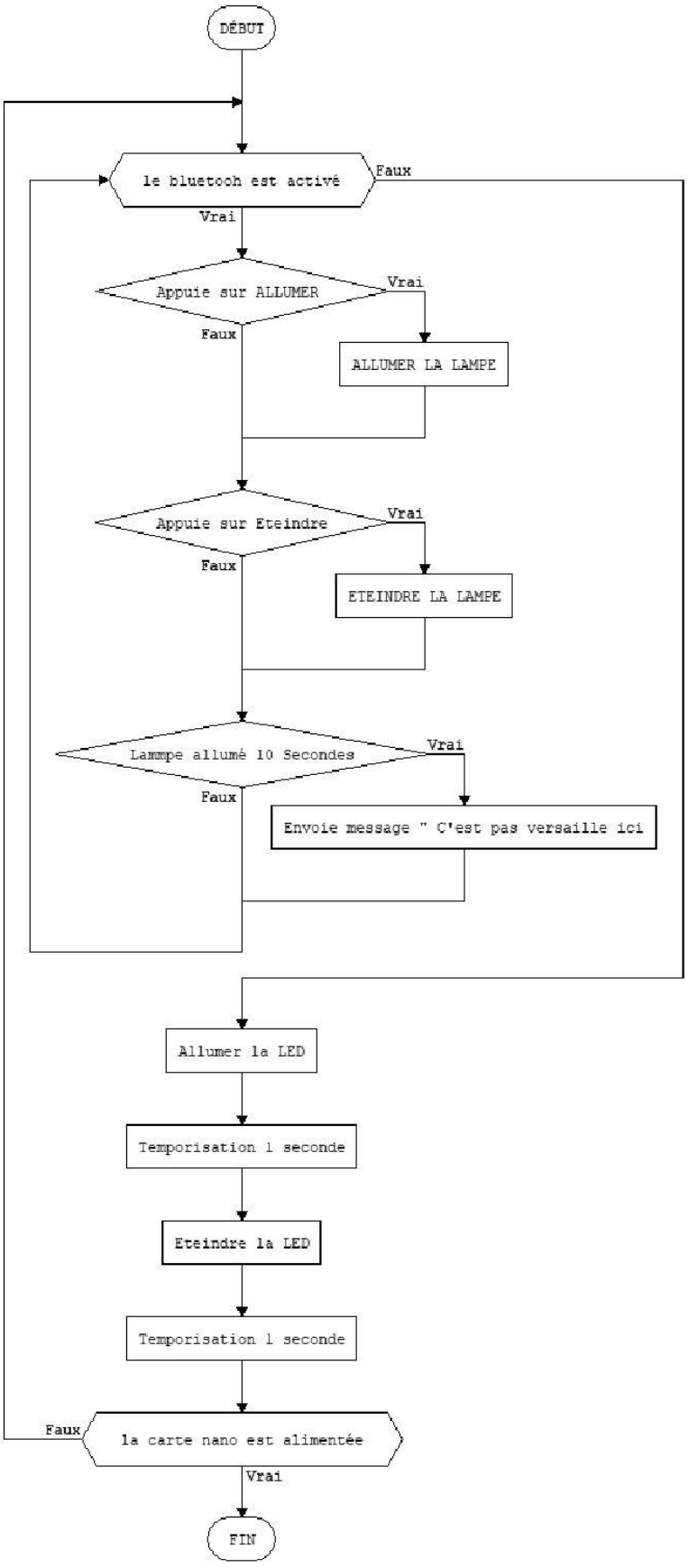
```
if (valeur_recue == 2) {
  digitalWrite(ledPin, LOW);
  Serial.println("led éteinte");
  monBT.print("                                ");
  temps_prevu = temps_courant;
}
```

Nous avons défini 2 pour éteindre la lampe, cette valeur est envoyée par le Bluetooth.
Si valeur reçue est égale à 2
On met à la valeur basse la led, on met à O la Led pour l'éteindre. On écrit sur le moniteur « led éteinte ».
On met la valeur du « temps courant » dans la variable « temps prévu » afin que les deux soient égales

```
if (valeur_recue == 1){
  Serial.print("temps_prevu : ");
  Serial.println(temps_prevu);
  Serial.print("temps_courant : ");
  Serial.println(temps_courant);
    if (temps_courant - temps_prevu >= interval) {
    Serial.println("c'est pas VERSAILLES, ici");
    monBT.print("c'est pas VERSAILLES, ici");
    temps_prevu = temps_courant;
  }}}
```

On vérifie que le temps courant - temps prévu >= 5000 ms ». Dès que la différence est égale à 5 secondes on écrit sur le moniteur et sur le Bluetooth « c'est pas VERSAILLES, Ici »

Cette fonction permet de créer une temporisation de 5 secondes.


DÉBUT
le bluetooh est activé
Faux
Vrai
Appuie sur ALLUMER
Vrai
Faux
ALLUMER LA LAMPE
Appuie sur Eteindre
Vrai
Faux
ETEINDRE LA LAMPE
Lammpe allumé 10 Secondes
Vrai
Faux
Envoie message " C'est pas versaille ici
Allumer la LED
Temporisation 1 seconde
Eteindre la LED
Temporisation 1 seconde
Faux
la carte nano est alimentée
Vrai
FIN

4. App Inventor :

A. L'application App Inventor

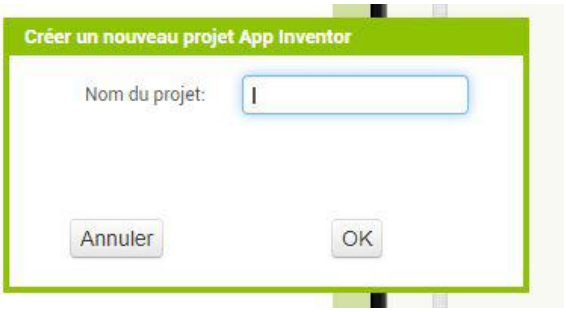
APP INVENTOR est un IDE (environnement de développement intégré) qui permet la création d'applications destinées à des systèmes équipés de plates-formes Androïd.

Cet environnement de programmation permet une programmation graphique aisée, basée sur l'assemblage de blocs (langage Scratch). En effet des blocs de propriétés, méthodes et évènements seront directement proposés dès la création d'un objet.

1) Création d'un compte Gmail

L'application est disponible en ligne gratuitement sur le site http://ai2.appinventor.mit.edu/
Après avoir créé un compte Gmail, il est possible de concevoir une application Android téléchargeable sur un téléphone Android.

En premier lieu, le projet doit être nommé. Il sera alors toujours disponible sur le compte créé. Les sauvegardes sont automatiques.



Outre la zone de travail, il est possible de visualiser les constructions sur son propre téléphone en se connectant grâce à un QR code via l'application MIT AIA Companion.
Cela permet de tester en direct le projet et de faire les modifications adaptées à la taille de son téléphone .

Le projet peut être conservé sous forme aia ou apk.
L'apk est le format de l'application finale. Celle-ci peut être téléchargée par toute personne disposant du QR code. Sous cette forme, elle n'est plus modifiable et la conception n'est plus accessible.

2) Création d'une application

Interface designer

Interface block

B. Application

On veut créer une application qui permettra d'allumer et d'éteindre une lampe gérée par une carte Arduino grâce à deux boutons.

Pour cela, on utilisera une connexion Bluetooth que l'on pourra choisir et désactiver.

1) « Interface designer »

La première étape consiste à construire l'apparence de l'application :

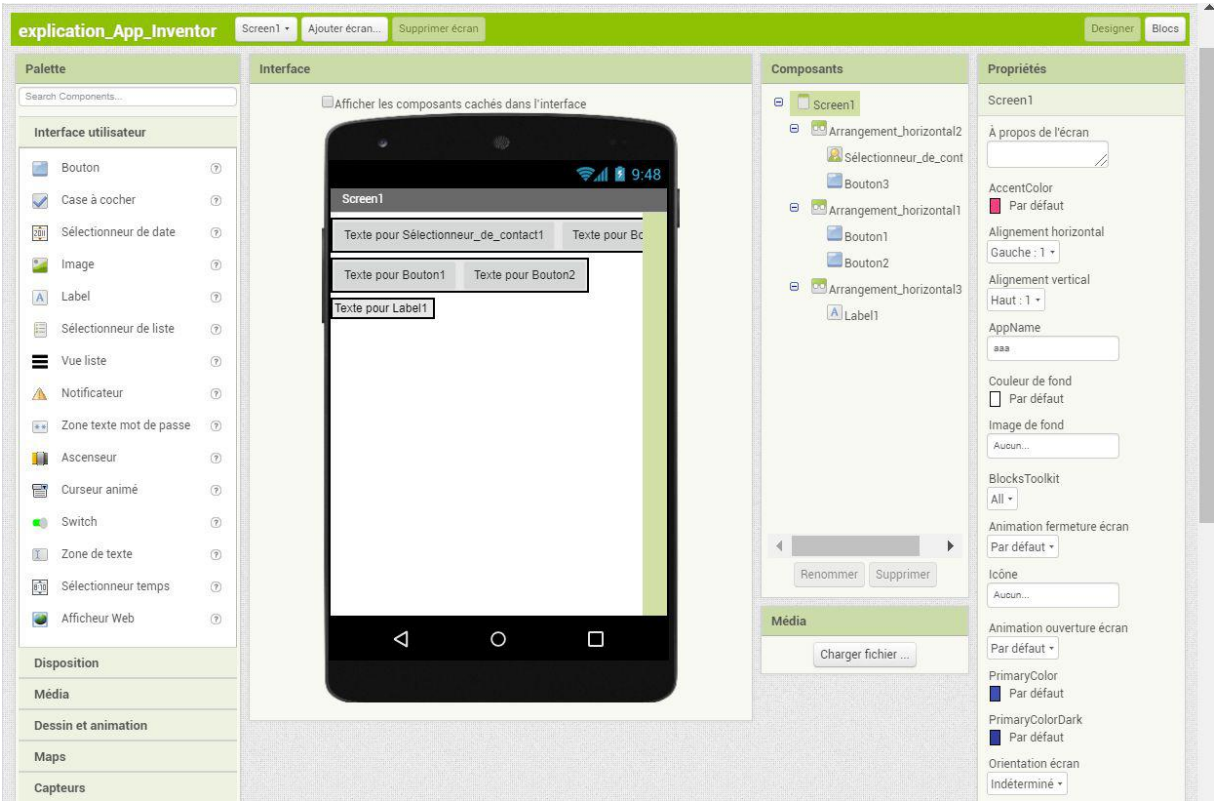
Pour réaliser une application, ces 4 étapes seront répétées :

1 - Insérer un composant par glisser-déposer,

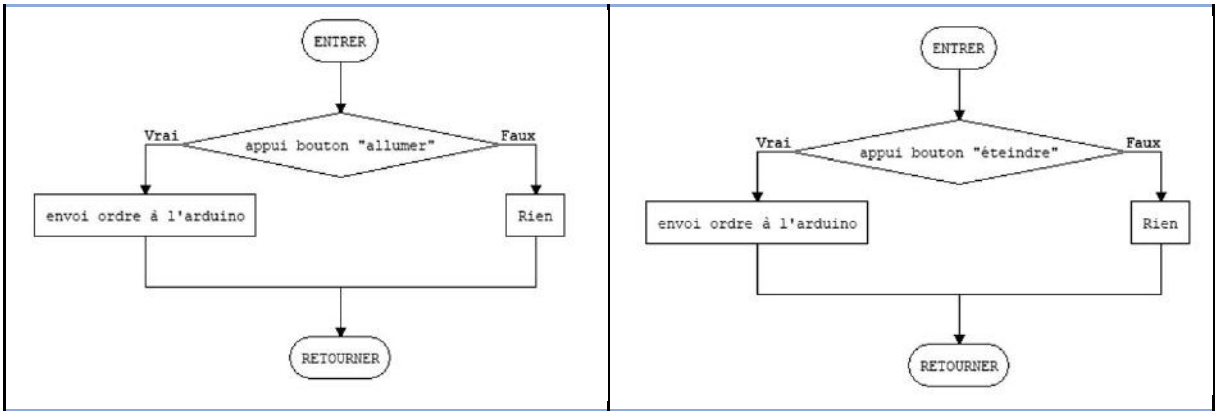
2 - Renommer le composant,

3 - Configurer ses propriétés,

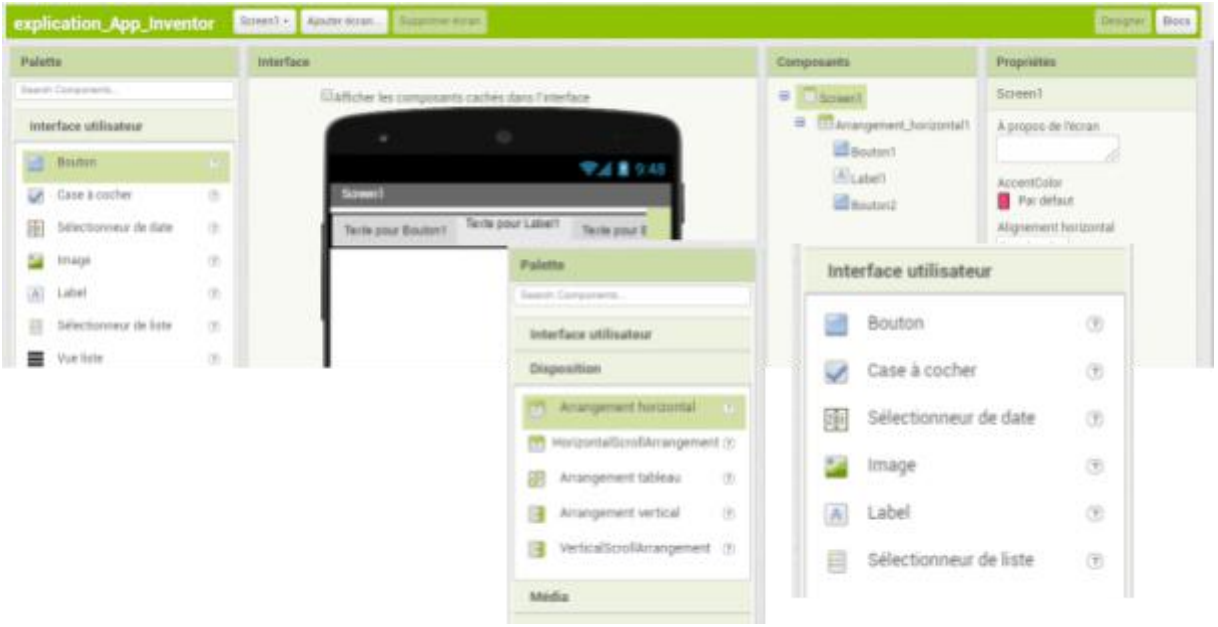
4 - Programmer son fonctionnement.



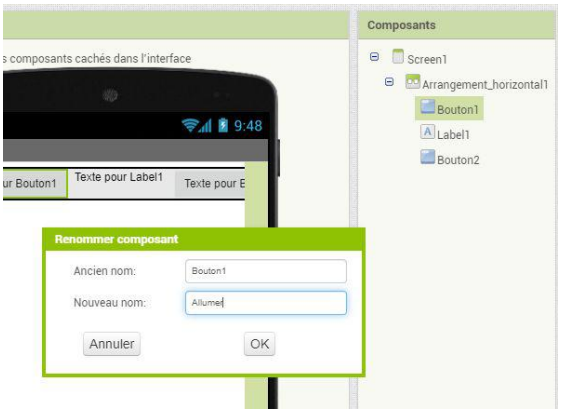
2) Création des boutons « allumer » et « éteindre »



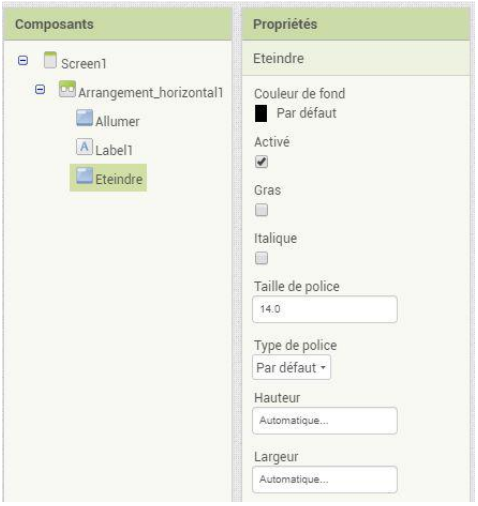
Pour insérer les boutons « Allumer » et « Éteindre » côte à côte, on va utiliser un **arrangement horizontal**. On va ensuite glisser 2 **boutons** dans l'espace en insérant entre eux une zone **Label**



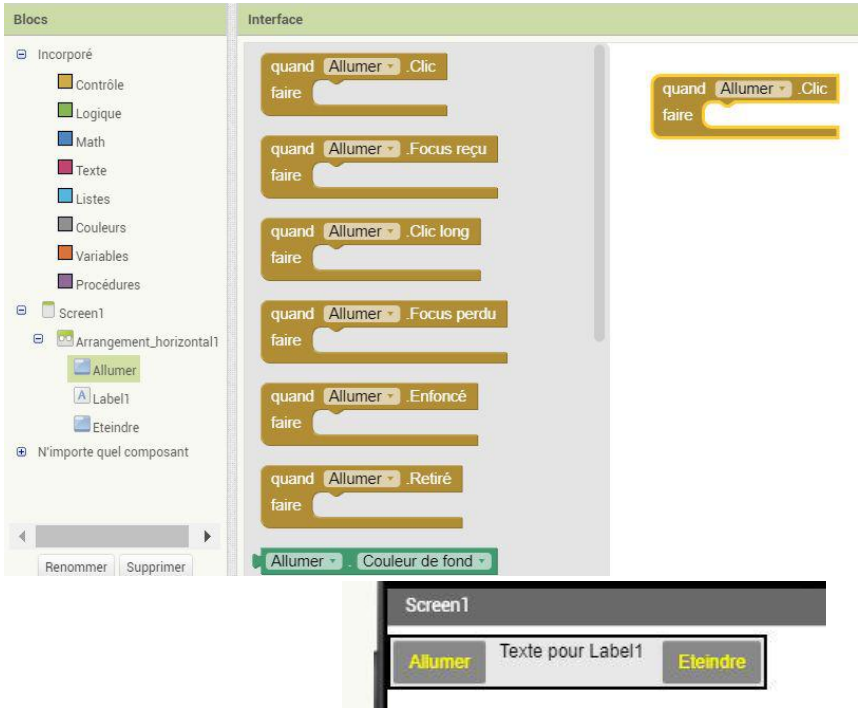
Renommer les boutons.



Choisir l'apparence et le libellé des boutons



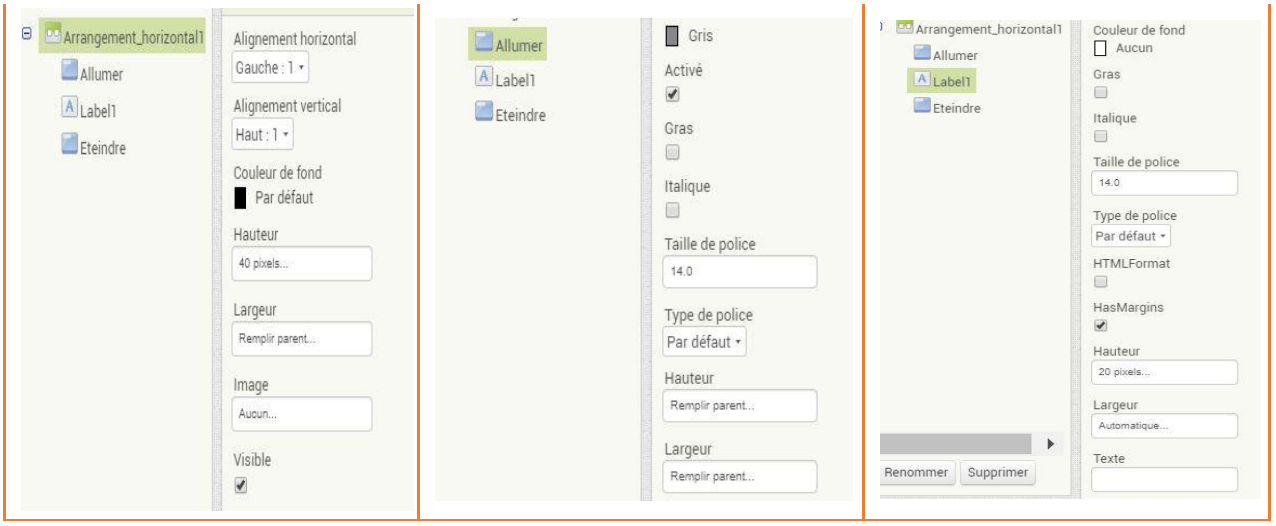
3) « Interface block »



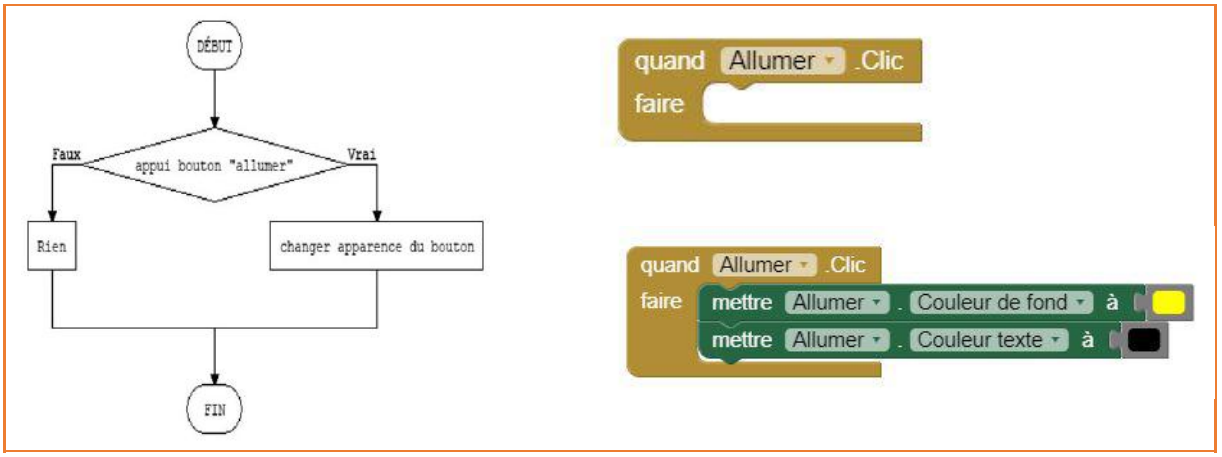
En sélectionnant un composant, on accède à une série de fonctions disponibles (glisser-déposer)

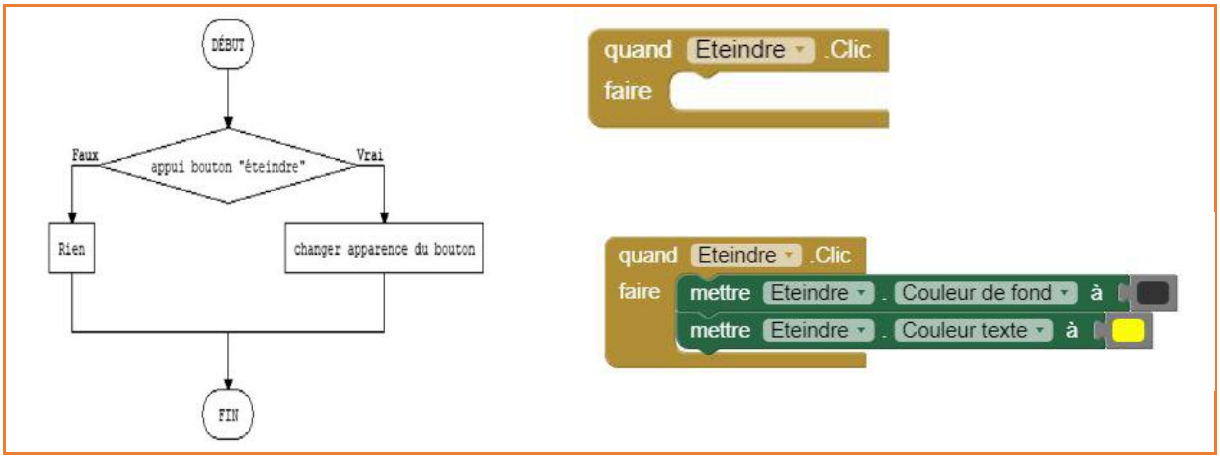
On termine la mise en page

tableau	boutons	label

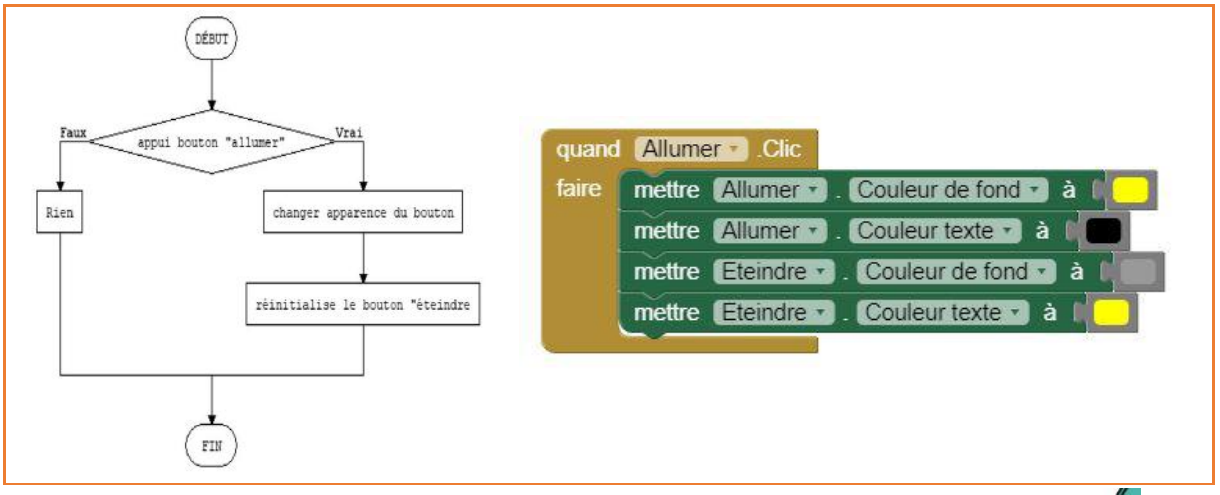


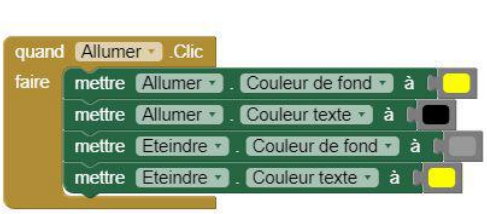
4) Programmation des boutons « allumer » et « éteindre »





Le clic d'un bouton remet l'autre bouton dans son apparence initiale



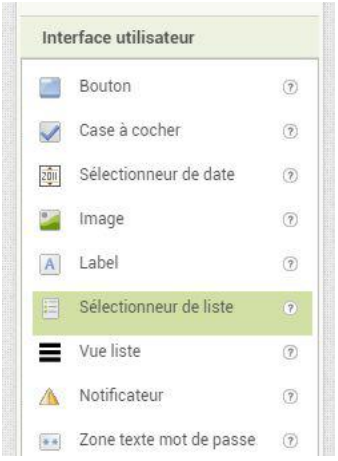


5) Connexion Bluetooth

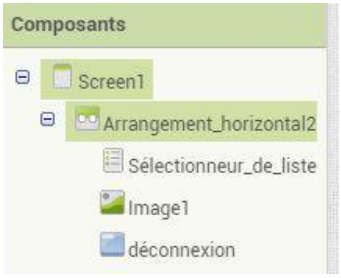
On veut envoyer des ordres et recevoir des informations avec Arduino.

L'application utilise les capteurs et fonctions présentes dans le téléphone. Le **sélectionneur de liste** (texte « connexion ») ouvre une fenêtre sur le téléphone qui affiche les connexions Bluetooth disponibles. Le **composant client Bluetooth** va utiliser le réseau du téléphone. Le Capteur Horloge va déclencher périodiquement les émissions et réceptions

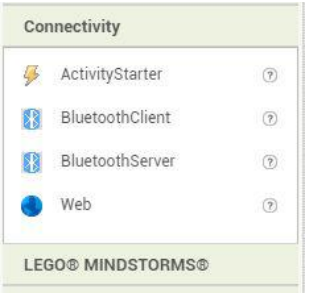
Recherche la liste des connexions BT



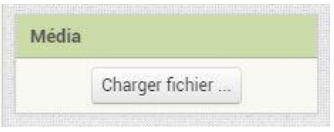
Création d'un bouton « déconnexion » et d'une image



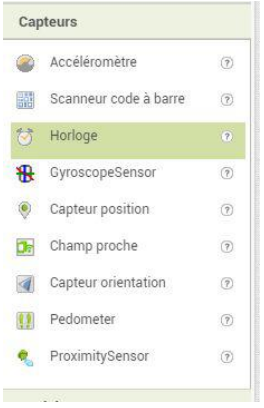
Insertion du composant client Bluetooth



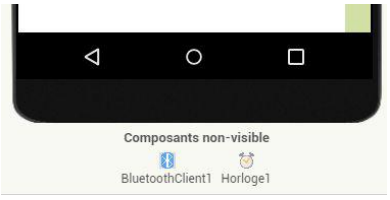
Insertion d'images présentes dans l'ordinateur



Capteurs Horloge

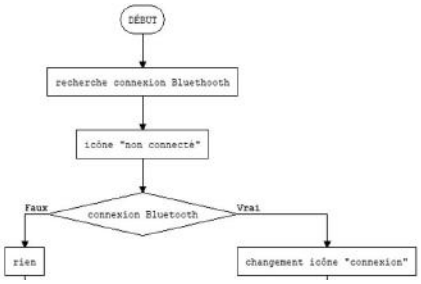


Un glisser-déposer insère les composants en dessous de l'écran

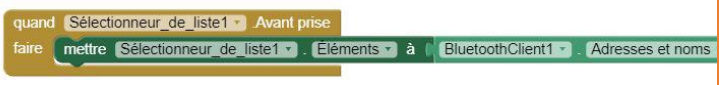



Screen1
Connexion
déconnexion
Allumer
Eteindre

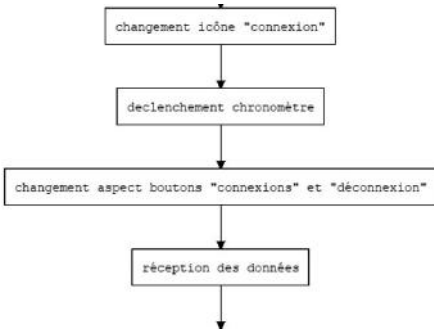
6) Programmation Bluetooth



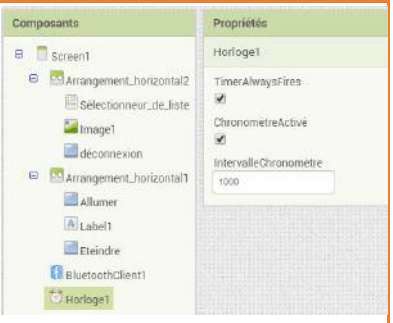
On récupère la liste des périphériques Bluetooth appariés sur le Smartphone



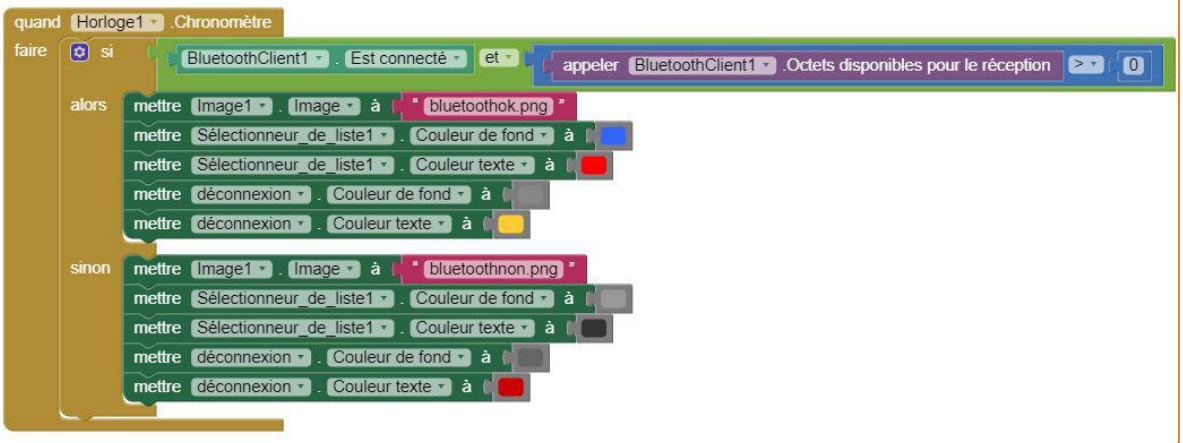
On se connecte au périphérique sélectionné



L'horloge a pour fonction de déclencher périodiquement le traitement.



Si le Bluetooth est connecté, l'aspect des boutons change et l'icône connexion apparait. Sinon, les boutons prennent un autre aspect et l'icône déconnecté apparait



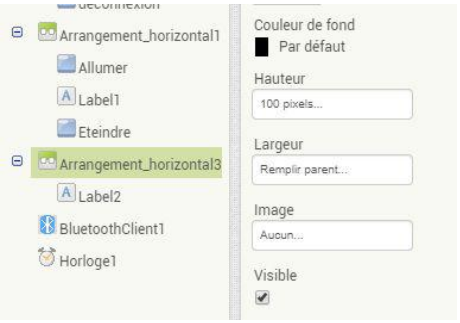
On désactive la connexion en appuyant sur le bouton

7) Réception d'information Bluetooth

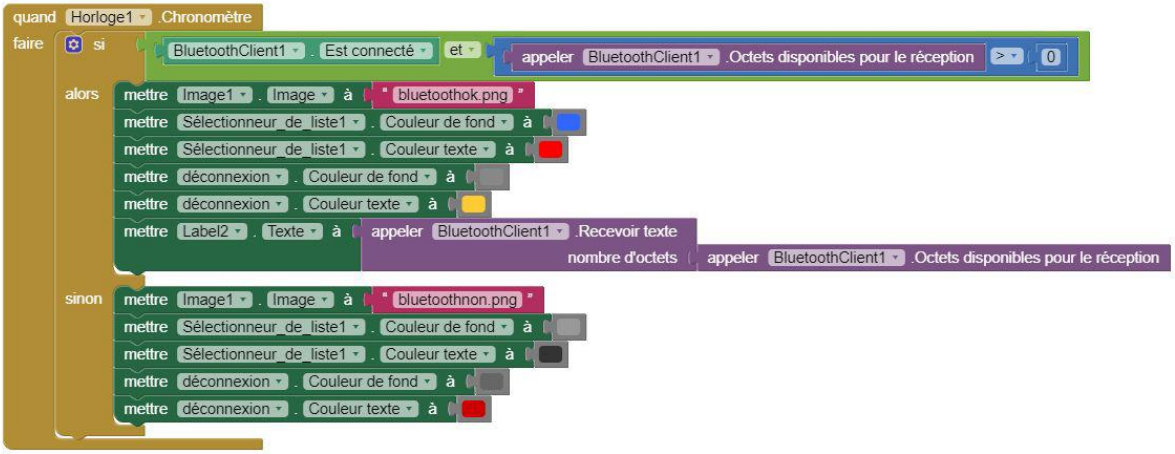
Le programme Arduino est conçu pour envoyer des messages textes selon des conditions prédéfinies.

L'application doit avoir une zone susceptible de la recevoir.

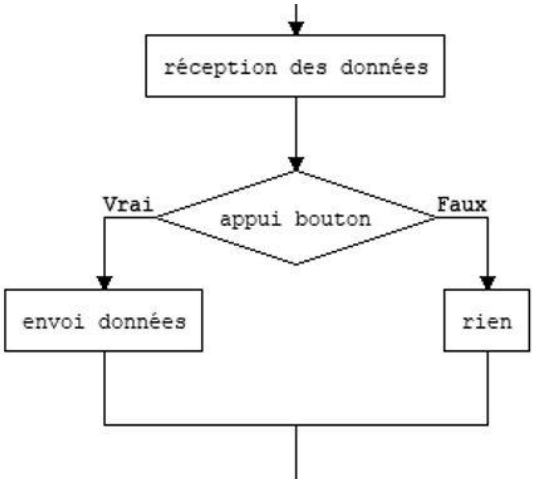
Dans l'onglet **Designer**, on crée un tableau dans lequel on place un Label dont on définit la taille.



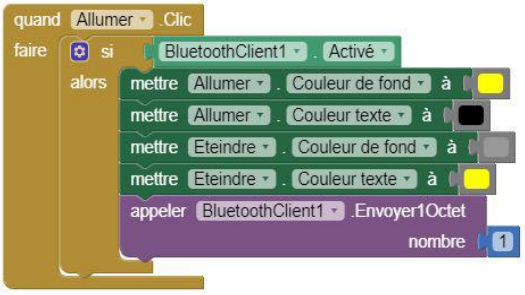
Dans l'onglet **Block**, les données reçues doivent être écrites dans la zone Label2
On rajoute une fonction dans le bloc chronomètre

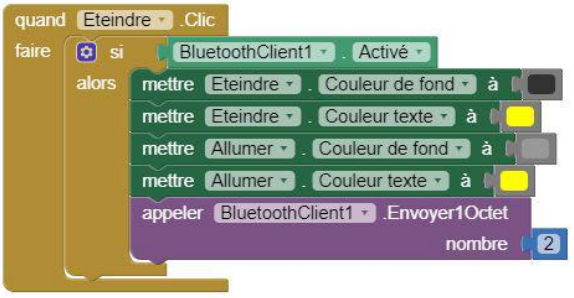


8) Envoi d'information Bluetooth



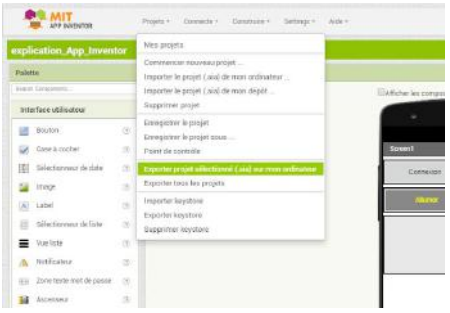
Lorsqu'on appuie sur le bouton « allumer », l'application envoie un nombre. Ce nombre est reçu par l'Arduino qui est capable de l'interpréter dans son programme.
Lorsqu'on appuie sur le bouton « éteindre », un autre nombre est envoyé.





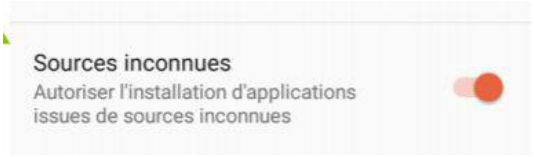
C. Enregistrement du projet

App Inventor enregistre toutes les modifications en ligne. Il est possible aussi d'exporter le projet sur le PC.
De la même façon, il est possible de télécharger et modifier une application existante tant que son extension est « aia ».



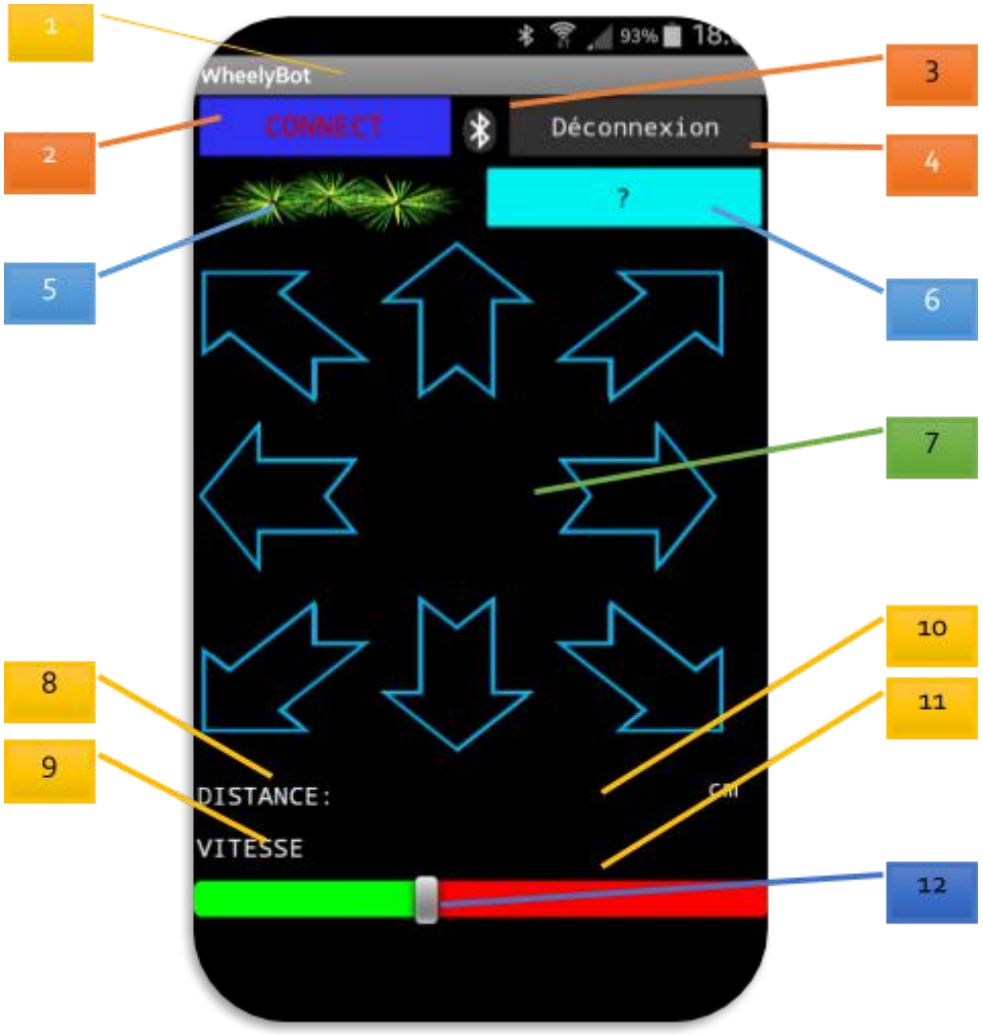
D. Téléchargement de l'application sur le téléphone

L'application créée est terminée.
L'onglet Construire permet de générer un QR code pour télécharger l'application en « apk ».
Cliquer sur « Installer ».
L'installation est bloquée car « sources inconnues »



Il faut ensuite « autoriser l'installation d'applications issues de sources inconnues. Lancer l'application.

5. Interface du téléphone pour WheelyBot



1	Screen (écran)	7	Flèches de direction
2	Bouton connexion	8	Label distance
3	Indicateur Bluetooth	9	Label vitesse
4	Bouton déconnexion	10	Indicateur distance

5	Bouton lumière	11	Indicateur vitesse
6	Bouton autre	12	Variateur vitesse