



Bluetooth & Arduino

Tutoriel d'application pour le Cocci-Bot

Nicolas LE GUERROUE

juillet 2020

Table des matières

1	Introduction	3
1.1	Présentation	3
1.2	Liste du matériel	3
1.3	Cahier des Charges	4
2	Appairage du module	6
3	Création de l'interface	7
3.1	Préparation	7
3.2	Sauvegarde du projet	9
3.3	Présentation des éléments graphiques	11
3.3.1	Principaux éléments	12
3.4	Présentation des propriétés	13
3.5	Renommer les éléments	16
3.6	Ajouter un client Bluetooth	17
3.7	Premier rendu et agencement des éléments	18
3.7.1	Générer un espace entre deux éléments	20
3.8	Alignement des éléments	21
3.9	Mise en place de l'ensemble des boutons	23
4	Gestion de l'application	25
4.1	Présentation	26
4.2	Configuration de la liste des périphériques Bluetooth	28
4.2.1	Principe	28
4.2.2	Définir les clients Bluetooth disponibles	29
4.3	Connexion au module Bluetooth	30
4.3.1	Principe	30
4.3.2	Emplacement des blocs	31
4.4	Gestion des boutons de direction	33

5	Installation de l'application	36
5.1	Installation de MIT App Inventor	36
6	Traitement des données	39
6.1	Branchements	39
6.2	Code Arduino	39
6.3	Conclusion	47

Chapitre 1

Introduction

1.1 Présentation

Ce tutoriel a pour objectif de créer une application pour diriger le robot en Bluetooth. L'application enverra des « commandes » au module Bluetooth de type Carius Cette application se fera à l'aide du logiciel en ligne «App Inventor 2», qui se divise en deux parties :

- une partie dédiée exclusivement à l'interface graphique (positionnement des boutons, etc).
- une seconde partie dédiée à la gestion des données et à leurs envois/réceptions

Nous verrons donc comment faire une interface spécifique pour le Cocci-Bot.

Remarque importante

Il est impératif que le téléphone portable tactile soit sous le système d'exploitation Android et qu'il dispose de la fonctionnalité Bluetooth

Par exemple, si j'appuie sur un bouton «avancer», je veux que l'application envoie en Bluetooth la donnée qui permettra à l'Arduino de comprendre l'instruction à l'aide du module Bluetooth.

1.2 Liste du matériel

Pour cette première partie, nous aurons besoin de :

- Un téléphone portable, comme dit précédemment, étant tactile et fonctionnant sous **Android**. Les dimensions importent peu, pourvu que celui-ci dispose du mode Bluetooth
- Un module Bluetooth de type Crius (Alimentation en 5V)
- Une carte Arduino
- Des fils de connexion pour brancher le module à la carte
- Une connexion internet

Et c'est tout... pour le moment.

1.3 Cahier des Charges

Maintenant que nous avons la liste du matériel nécessaire, ce serait bien de mettre en place un cahier des charges afin de savoir ce que le robot sera capable de faire...

1) L'application devra se connecter au module Bluetooth du robot. Pour cela, on se connectera à un module en passant par la liste des modules Bluetooth disponibles.

Cette méthode permet de se connecter sans avoir l'adresse MAC. En revanche, l'étape de l'appairage est indispensable. (Chapitre 2)

2) Le robot devra être contrôlé de façon manuelle, il y aura donc 5 boutons de commande :

- 1 bouton «Avancer»
- 1 bouton «Reculer»
- 1 bouton «Droite»
- 1 bouton «Gauche»
- 1 bouton «Stop»

3) Le robot devra également se diriger de façon autonome grâce à ses capteurs (distance, lumière). Il y aura donc un bouton «automatique» pour déclencher ce mode.

Application facultative

Le robot pourra également indiquer l'état de la batterie (en %, sur l'écran LCD) par simple appui d'un bouton.

Cette section ne sera pas abordée ici.

Chapitre 2

Appairage du module

Point-clé

Avant de créer l'application, il faut tout d'abord que le portable puisse reconnaître le module Bluetooth.

Pour cela, il faut «l'appairer», c'est à dire que l'on va définir que le module sera apte à recevoir les données envoyées par le portable.

Cette étape est très rapide et ne sera effectuée qu'à chaque changement de module Bluetooth.

1) Tout d'abord, branchez le module : la broche +5V du module est reliée à la broche 5V de la carte Arduino et la broche GND du module est reliée à la GND de la carte.

2) Démarrez le Bluetooth sur votre téléphone portable puis allez dans **paramètres > Bluetooth**

Faites « **rechercher** » afin de trouver un périphérique. Vous devriez trouver un périphérique « BT Crius ». Cliquez dessus et faites « **associer** ».

A ce moment là, on vous demandera un mot de passe, qui par défaut, est **0000** (ou **1234**).

Lorsqu'il est entré, faites « **valider** » et au bout de quelques secondes, le clignotement du module devrait se stopper pour qu'il n'y ait qu'une lumière continue

Et voila, le module est appairé.

Chapitre 3

Création de l'interface

3.1 Préparation

Maintenant que nous nous sommes assurés que la communication s'effectuera, il est temps de faire l'application et de préparer notre outil (enfin, me direz-vous !)

Tout d'abord, saisissez dans un moteur de recherche l'adresse suivante : <http://appinventor.mit.edu/> (Site **App Inventor 2** »)

Une page comme ceci devrait apparaître :

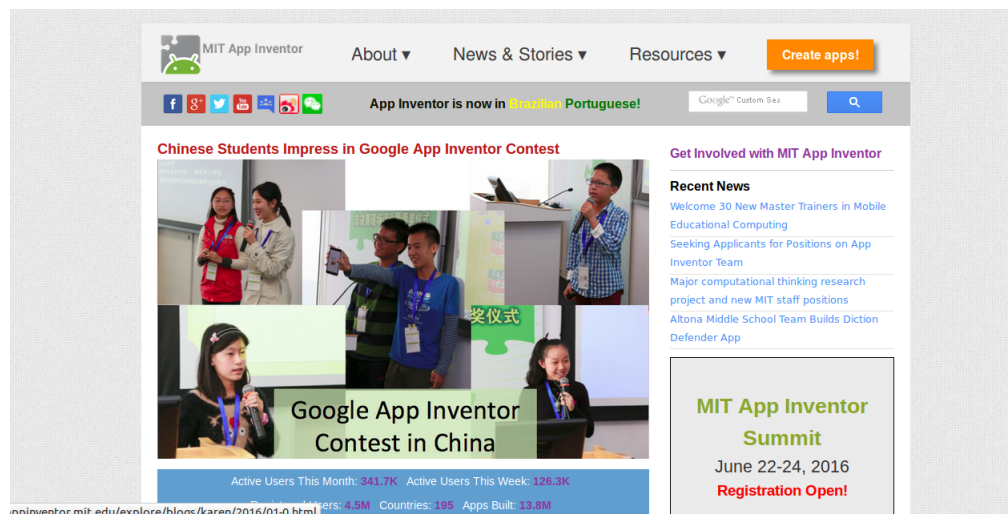


FIGURE 3.1 – Le portail App Inventor

Ensuite, cliquez en haut à gauche sur :



Cela devrait vous diriger vers ceci :

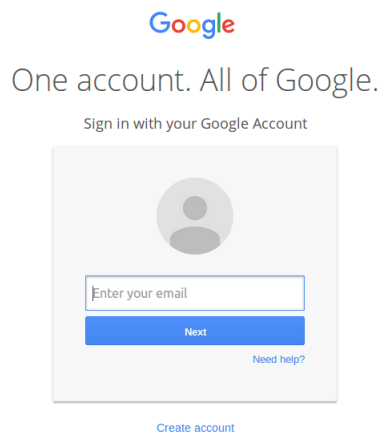


FIGURE 3.2 – La fenêtre de connexion

Puis connectez vous avec votre compte Google (ou Gmail). Si vous n'en avez pas, sa création est rapide... (create account). Vous tombez ensuite sur une interface similaire :

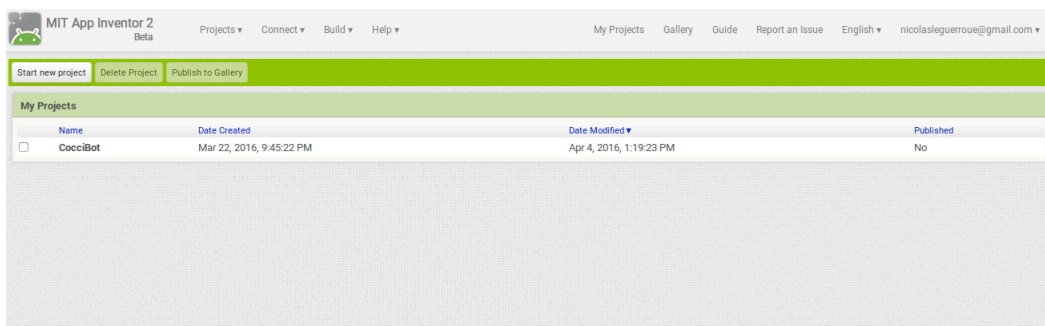


FIGURE 3.3 – La fenêtre des applications

Pour démarrer votre projet, cliquez sur « **Start new project** » (onglet « **Projects** ») et là, cette page apparaît :

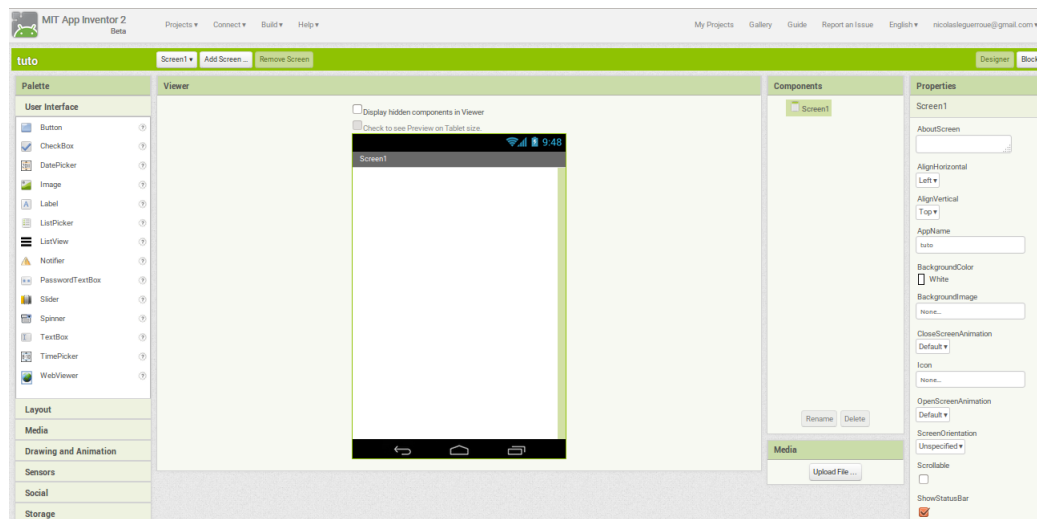


FIGURE 3.4 – La fenêtre de notre application

Avant toute chose, réglez la langue du site en français. Pour cela, sélectionnez l'onglet **English** et choisissez « **français** ».

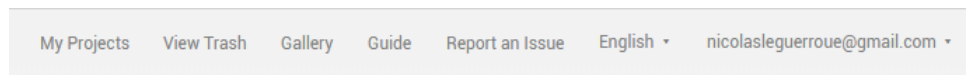


FIGURE 3.5 – Le choix de la langue

3.2 Sauvegarde du projet

Avant toute chose, voici une étape non négligeable : la sauvegarde du projet

Remarque importante

L'application App Inventor étant en ligne, un problème de réseau peut ruiner votre projet en cas de mauvaise sauvegarde. Il convient de sauvegarder **régulièrement** votre travail

Il faut se reporter au menu du haut et aller dans **Projets** puis **Enregistrer le projet**

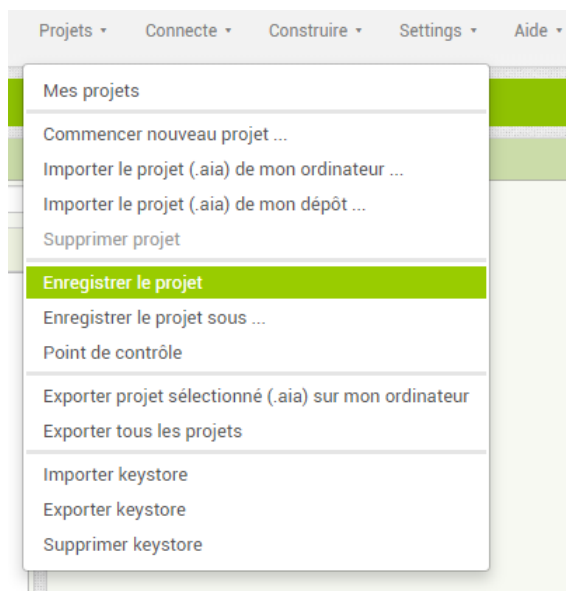


FIGURE 3.6 – Sauvegarde du projet

Quant à l'interface, un petit tour d'horizon s'impose.

3.3 Présentation des éléments graphiques

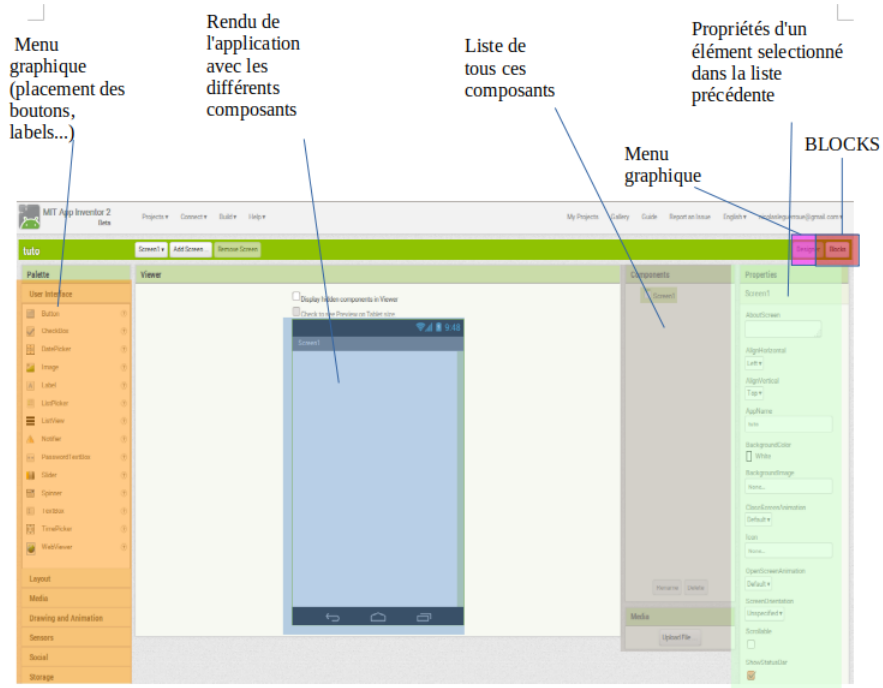


FIGURE 3.7 – Les outils de App Inventor

Par convention, dans ce tutoriel :

- la partie **orange** sera la partie menu
- la partie **bleue** sera l'interface
- la partie **grise** sera la partie composants
- la partie **verte** sera la partie propriétés
- la partie **rouge** garde son nom : blocks
- la partie **magenta** est la partie "graphique" (regroupement des quatre premières parties)

Maintenant, prenons un **sélecteur de liste** dans le menu, et faisons le **glisser sur l'interface**. Pour cela, maintenez enfoncé le « **sélecteur de**

liste » dans le menu et faites le glisser sur l'écran virtuel. Relâchez ensuite la souris. On obtient :

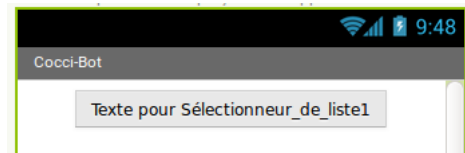


FIGURE 3.8 – Rendu de l'interface

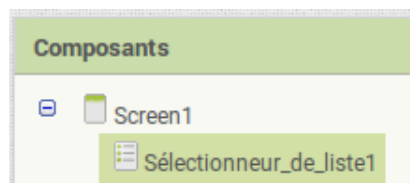


FIGURE 3.9 – Rendu des composants

Veuillez cliquer sur **Sélecteur_de_list1**

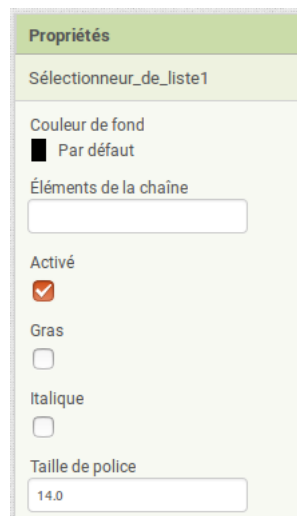


FIGURE 3.10 – Rendu des propriétés

3.3.1 Principaux éléments

Maintenant, on peut voir que le menu offre plusieurs choix d'objets. Voici les plus importants :

Il vous est notamment possible de faire les actions suivantes :

- Créer un bouton
- Créer une case à cocher
- Sélectionner une date
- Afficher une image
- Afficher du texte
- Créer une liste de selection
- Créer une liste classique
- Envoyer une notification
- Afficher un curseur
- Créer une zone de saisie de texte (mot de passe...)
- Sélectionner l'heure

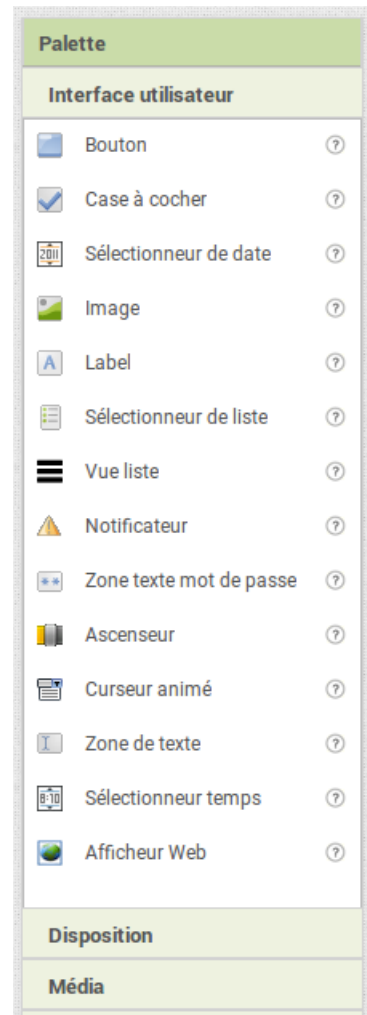


FIGURE 3.11 – La liste des composants

3.4 Présentation des propriétés

Nous allons présenter les propriétés principales des composants mis à notre disposition.

Tout d'abord, gardez le **sélectionneur de liste** sur l'**écran**. Ensuite, pour centrer la liste, allez dans le menu composants, sélectionnez **screen 1**.

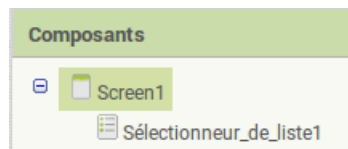


FIGURE 3.12 – Sélection de l'écran

Dans l'onglet **propriété** → alignement horizontal, sélectionner la liste sur **centrer**.

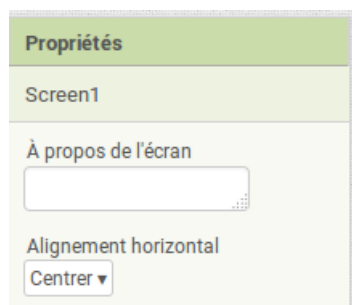


FIGURE 3.13 – Centrage de l'écran

Vous pouvez également changer la forme du bouton. Allez dans **Composants** → sélecteur de liste et dans **Propriétés**, trouvez l'onglet **forme** et sélectionnez celle voulue.

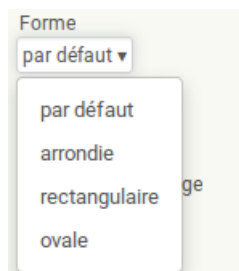


FIGURE 3.14 – Forme du bouton

Voici quelques paramètres de mise en forme :

- Vous pouvez modifier la couleur de la liste en sélectionnant **Couleur de fond**
- Pour modifier la taille de la liste, il suffit de rentrer la taille en pixels ou en % de l'écran (**Hauteur, Largeur**)
Il vaut mieux préciser en % afin que l'application soit compatible au niveau du format sur tous les appareils.



FIGURE 3.15 – La liste des propriétés

Enfin, pour décider du titre de la liste sur l'application, sélectionner la liste **Sélectionneur_de_liste1** et dans ses propriétés, modifier l'onglet **texte** et écrivez, par exemple « **Connexion** »



FIGURE 3.16 – Choisir le nom par défaut du bouton **Connexion**

3.5 Renommer les éléments

Dans un souci de clarté, il convient de renommer les éléments que nous plaçons sur l'écran.

Pour renommer le sélectionneur de liste, il faut aller dans les composants de l'interface, et sélectionnez « Renommer ». Je l'ai renommé en « connexion ».

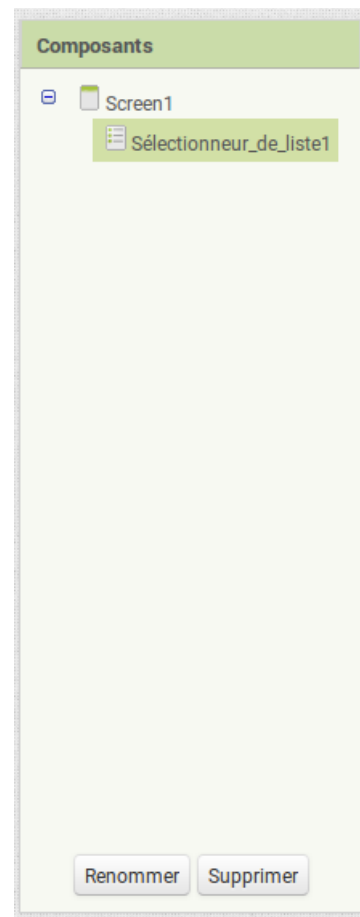


FIGURE 3.17 – Renommer un élément

Astuce

Pour changer la couleur d'arrière-plan de l'application, allez dans composants → screen1 et dans propriétés, sélectionnez couleur de fond

3.6 Ajouter un client Bluetooth

Par la suite, nous allons utiliser le client Bluetooth d'App Inventor. Pour le trouver, il suffit d'explorer le [menu latéral](#). Un ensemble d'éléments est disponible à la suite des principaux éléments énumérés dans la section 3.3.1.

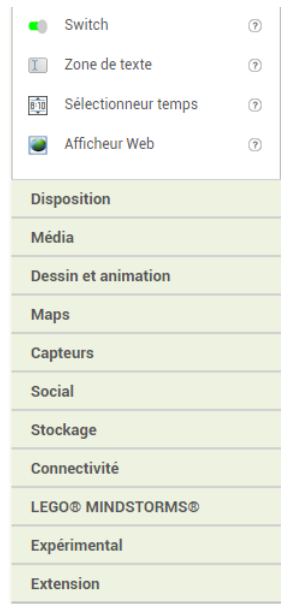


FIGURE 3.18 – Menu latéral

Le client Bluetooth se situe dans la section **Connectivité**

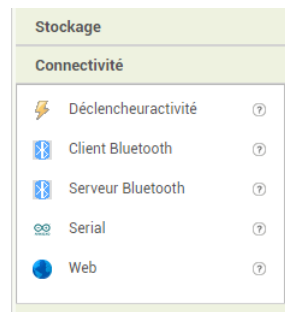


FIGURE 3.19 – Ajouter un client Bluetooth

Pour l'ajouter à notre application, il suffit de glisser le client Bluetooth sur l'interface. A ce moment là, sous l'écran virtuel, le client Bluetooth apparaît.

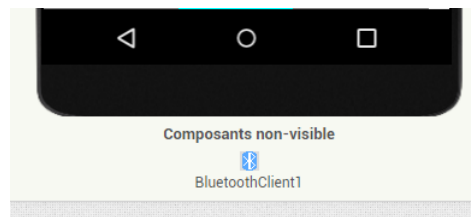


FIGURE 3.20 – Présence du client Bluetooth

Et voilà, le client Bluetooth est mis en place.

3.7 Premier rendu et agencement des éléments

En faisant toutes ces étapes, on obtient :



FIGURE 3.21 – Premier rendu de l'application

En ajoutant un bouton (glisser-déposer un bouton disponible dans le menu latéral) et en modifiant ses propriétés (forme, couleur...), on obtient ceci :



FIGURE 3.22 – Rendu avec un bouton "Avancer"

3.7.1 Générer un espace entre deux éléments

On souhaite maintenant espacer le bouton "Connexion" et le bouton "Avancer"

Rien de plus simple ! Il suffit de placer un élément **label** (outil dans menu) entre les deux boutons puis de choisir ses dimensions, ce qui correspondra à l'espacement des boutons.

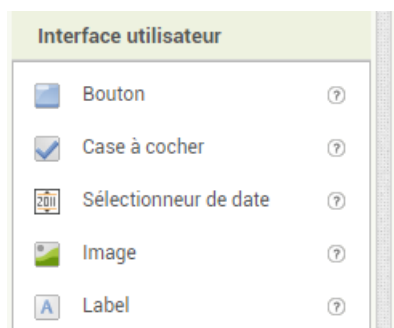


FIGURE 3.23 – Emplacement du label

Pour que le label ne soit pas visible, il suffira de mettre son texte vide.

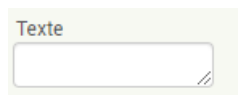


FIGURE 3.24 – Texte du label

Et voilà le résultat :



FIGURE 3.25 – Espace entre deux éléments

3.8 Alignement des éléments

Maintenant, ce serait bien de pouvoir aligner des boutons (ou autre) horizontalement afin d'obtenir quelque chose comme ceci :

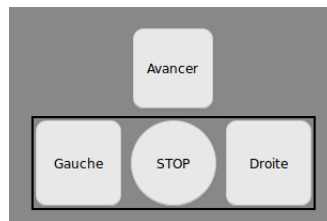


FIGURE 3.26 – Alignement d'éléments

Ne vous inquiétez pas, le cadre noir n'apparaîtra pas sur l'application ; c'est juste un moyen de distinguer les alignements.

Pour faire ceci, allez dans le **menu** et sélectionnez l'onglet **Disposition**

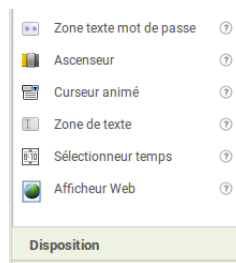


FIGURE 3.27 – Emplacement des éléments d'agencement

Une fenêtre comme ceci apparaît :

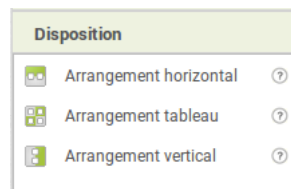


FIGURE 3.28 – Éléments d'agencement

Ensuite, faites glisser l'outil **Arrangement Horizontal** sous la liste « **Connexion** » et vous devriez avoir ceci :

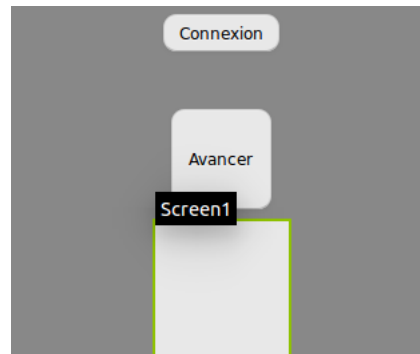
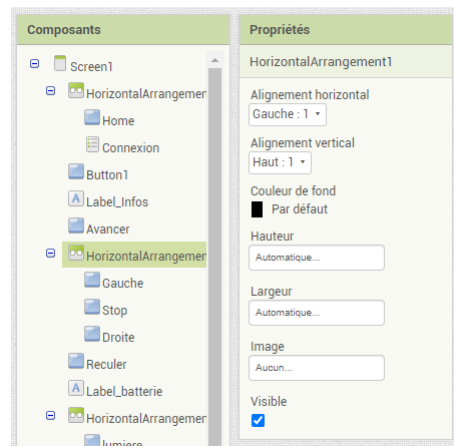


FIGURE 3.29 – Éléments d'agencement placé

Vous pouvez dorénavant faire glisser des boutons dans ce carré et ils se mettront automatiquement côte à côte. Il suffit de régler la taille des boutons sélectionnés dans [propriétés](#).

Astuce

Il est possible de changer les dimensions de l'outil **Arrangement Horizontal**

FIGURE 3.30 – Propriétés de l'objet **Arrangement Horizontal**

En revanche, il y a un fond blanc pour l'objet **Arrangement Horizontal** :

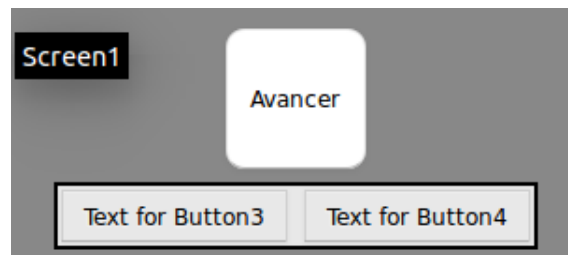


FIGURE 3.31 – Couleur d'arrière-plan de l'outil **Arrangement Horizontal**

En allant dans les **propriétés** de l'objet **Arrangement Horizontal**, mettez la couleur d'arrière-plan en « **aucun** »

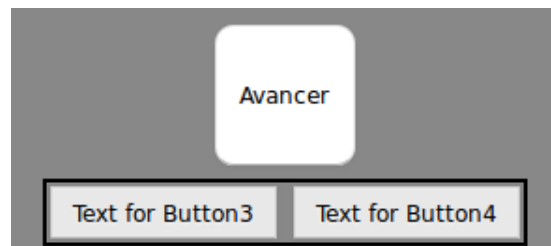


FIGURE 3.32 – Couleur d'arrière-plan transparent

3.9 Mise en place de l'ensemble des boutons

Après avoir mis les boutons "**Droite**", "**Stop**", "**Gauche**", "**Reculer**", "**Batterie**" et "**Auto**" et mis un peu de couleur, vous pouvez obtenir une interface de ce type :

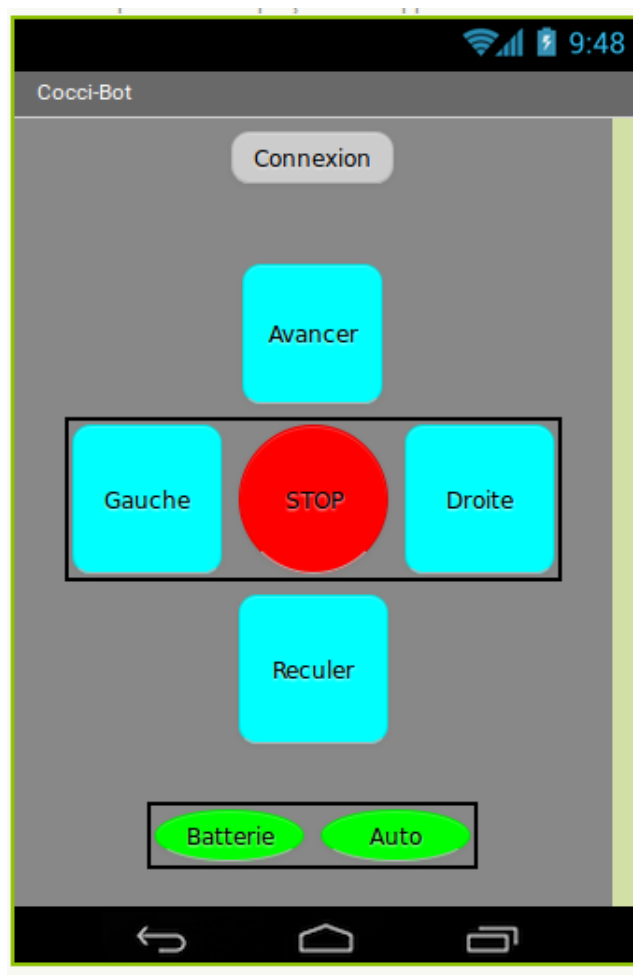


FIGURE 3.33 – Rendu de l'application

Remarque importante

Par la suite, veuillez mettre l'arrière-plan de « connexion » en rouge (non représenté ici)

Chapitre 4

Gestion de l'application

Nous avons réalisé et mis en place tous nos élément graphiques. Il ne nous reste donc plus qu'à les faire interagir afin d'envoyer les données Bluetooth. Pour cela, commencez par aller dans la section **Bloc**, disponible en haut à droite de la page



FIGURE 4.1 – Emplacement du menu "**Bloc**"

4.1 Présentation

A l'ouverture du menu **Bloc**, voici le rendu

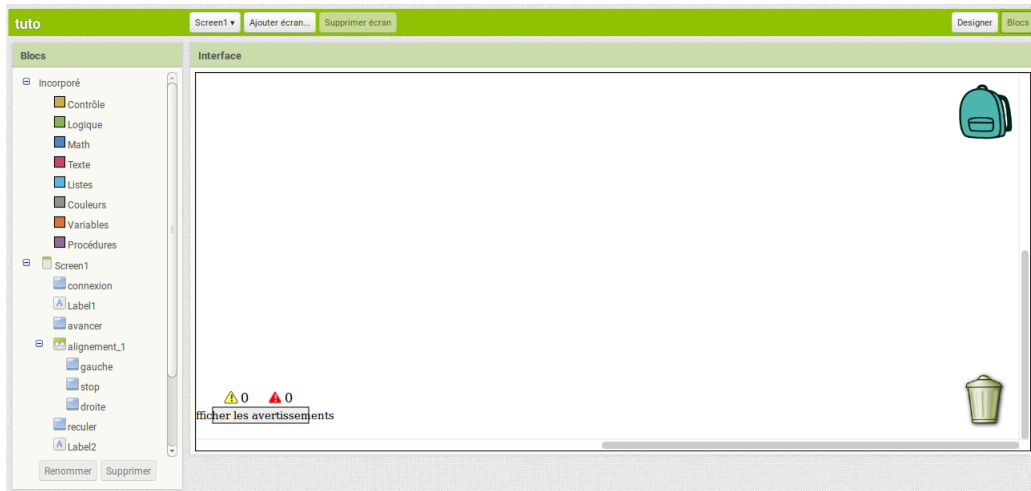


FIGURE 4.2 – Menu "Bloc"

Sur le menu de gauche, nous retrouvons une liste de blocs

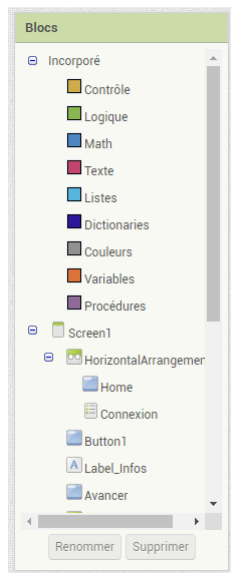


FIGURE 4.3 – Menu latéral

mais également nos éléments graphiques agencés précédemment :

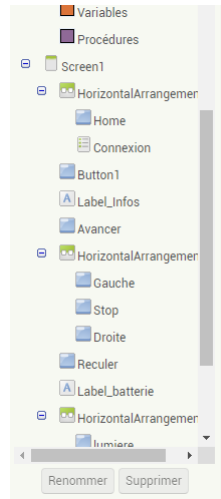


FIGURE 4.4 – Nos éléments ajoutés

En cliquant sur chaque onglet (Contrôle, Logique, Math...), une liste de blocs apparaît : (ici, en cliquant sur Contrôle)



FIGURE 4.5 – Blocs Controls

Tous ces blocs sont faits pour pouvoir être glissés dans la partie blanche centrale...

4.2 Configuration de la liste des périphériques Bluetooth

4.2.1 Principe

Nous allons configurer notre liste **Connexion** afin qu'elle nous indique les modules Bluetooth disponibles.

Cette étape se fera en deux temps :

- Configuration de la liste **avant** le choix de l'utilisateur
- Mise à jour de la liste **après** le choix de l'utilisateur

Commencer par sélectionner la liste **Connexion** dans le menu latéral

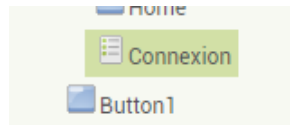
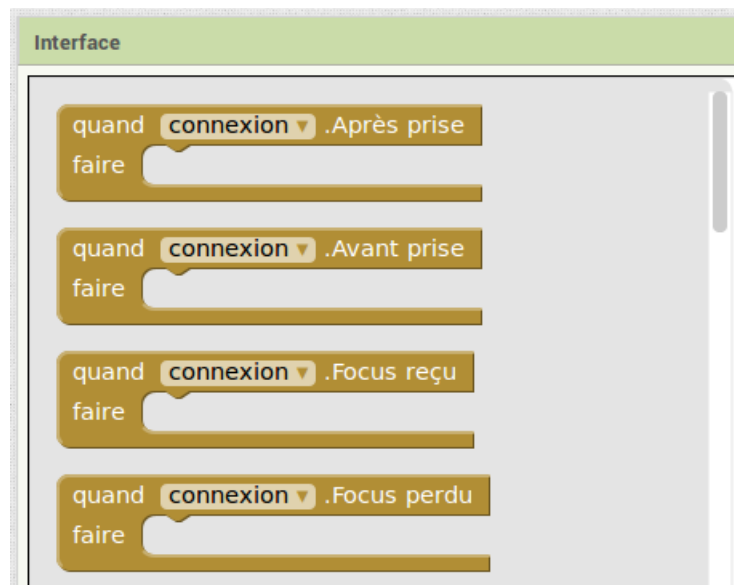


FIGURE 4.6 – Sélection de la liste

Le menu suivant apparaît :

FIGURE 4.7 – Éléments de l'objet **Connexion**

- Le "**Quand connexion.Avant prise**" représente le bloc qui contiendra les instructions de la liste, c'est à dire que dans ce bloc, nous annoncerons que la liste contiendra toutes les adresses Bluetooth disponibles.
- Le "**Quand connexion.Après prise**" représente le bloc d'instructions après avoir choisi le module Bluetooth dans la liste. Dans ce bloc, après avoir récupéré l'adresse du module, nous nous connecterons à ce dernier.

Il faut donc placer ces deux blocs dans la zone blanche (glisser-déposer).

4.2.2 Définir les clients Bluetooth disponibles

Le bloc pour définir les éléments de la liste **Connexion** est le suivant :

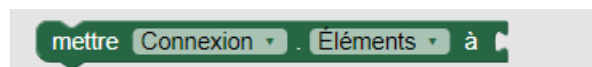


FIGURE 4.8 – Définition de la liste

Il attend en argument (zone de puzzle à droite) une liste. Nous allons lui donner une liste des modules Bluetooth disponibles grâce à

notre Client Bluetooth installé (Section 3.6).

Ce bloc est disponible dans le menu latéral, partie **BluetoothClient1**

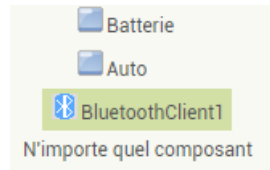


FIGURE 4.9 – Sélection du client Bluetooth

Puis

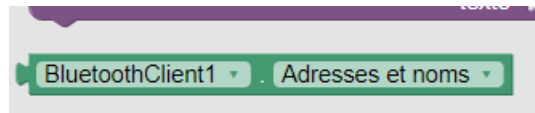


FIGURE 4.10 – Sélection du bloc des adresses

Au final, on obtient le bloc suivant :



FIGURE 4.11 – Bloc de configuration de la liste

4.3 Connexion au module Bluetooth

Une fois que l'utilisateur a sélectionné le module auquel il souhaite se connecter, il faudra mettre à jour l'état de la liste et se connecter au module Bluetooth

4.3.1 Principe

Une fois que la personne a fait son choix dans "**Connexion**", on vérifie la couleur de l'arrière plan de **connexion** :

- Si la couleur est rouge (par défaut), cela veut dire que nous ne sommes pas connectés. On remplace donc le mot « Connexion » par l'adresse MAC¹ sélectionnée et on définit la couleur d'arrière-plan en vert, pour obtenir ceci :

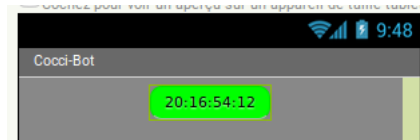


FIGURE 4.12 – Rendu de l'état de connexion

Visuellement, on sait que nous sommes connectés.

Ensuite, pour se connecter réellement, on établit une connexion Bluetooth sécurisée à l'adresse donnée par la sélection de la liste « connexion ». Quand c'est fait, on envoie en Bluetooth la lettre « c » que l'Arduino se chargera d'interpréter, et pourra, en conséquent, jouer une mélodie pour confirmer la connexion.

- Si la couleur est verte , cela veut dire que nous sommes déjà connectés. Il faut donc remettre la couleur et le texte d'origine (« connexion ») et se déconnecter

4.3.2 Emplacement des blocs

Dans le bloc **Quand connexion. Après prise**, nous allons avoir besoin d'une structure de contrôle, c'est à dire en l'occurrence un bloc conditionnel **Si-Alors-Sinon**

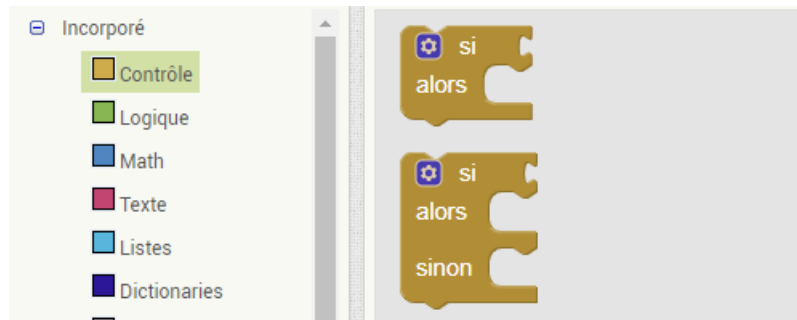


FIGURE 4.13 – Emplacement des blocs conditionnels

1. L'adresse MAC est une adresse physique pour identifier de manière unique un composant.

Nous aurons également besoin d'un bloc de comparaison :



FIGURE 4.14 – Emplacement des blocs de comparaison

Les blocs de couleurs sont disponibles à l'emplacement suivant :

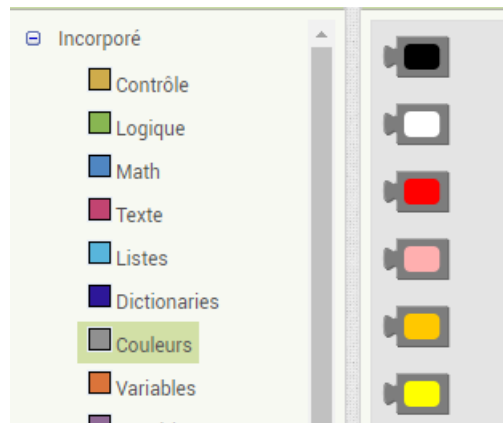


FIGURE 4.15 – Emplacement des blocs de couleurs

Enfin, les chaînes de caractères sont disponibles à l'emplacement suivant :

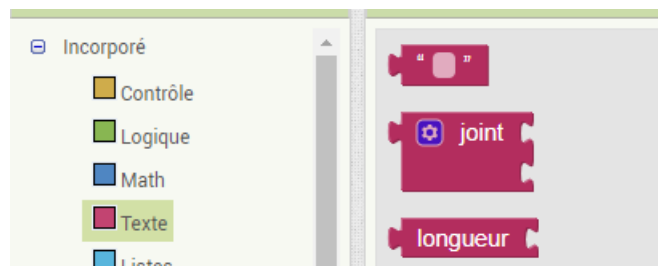


FIGURE 4.16 – Emplacement des chaînes de caractères

Astuce

Pour savoir où se situe une instruction, il faut regarder le nom de l'instruction. Elle sera de la forme :

nom_de_l'instruction.fonction

le "nom_de_l'instruction" sera affiché dans un des onglets de gauche (Figure 4.1)

Par exemple, pour "**mettre Connexion.elements**", il faut aller dans l'onglet "**Connexion**"

Maintenant que vous savez où trouver les blocs, je vous invite à recopier ce bloc d'instructions, qui n'est que la mise en pratique de la partie "Principe" de cette section

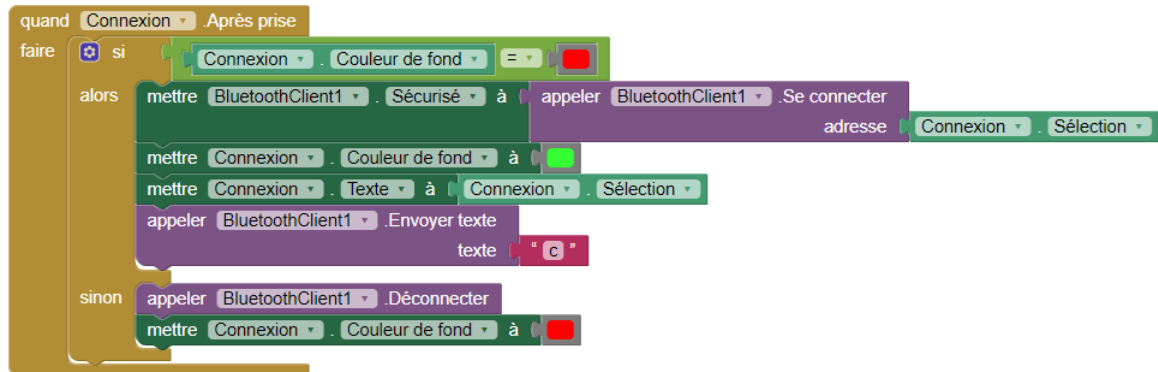


FIGURE 4.17 – Gestion de la liste après sélection

Le plus dur est fait, reste maintenant à gérer les boutons directionnels.

4.4 Gestion des boutons de direction

Nous allons sélectionner le bouton **Avancer** dans le menu de gauche



FIGURE 4.18 – Menu du bouton Avancer

A l'intérieur de ce bloc, on mettra les instructions pour envoyer le mot « c » en Bluetooth

Pour le langage Arduino, c'est l'équivalent de :

```
if (bouton==appui) {allumer led ;}
```

Equivalent Arduino

Après, nous avons d'autres paramètres (appui long, après avoir retiré le doigt, sur le bouton...)

Placez le **"Quand avancer. Click"** sur la page blanche.

Ensuite, nous allons vérifier si nous sommes connectés. Pour cela, nous mettrons un **"Si connexion.Couleur_de_fond = vert"** afin d'attester la connexion.

Enfin, sélectionnez l'onglet "**BluetoothClient1**" et insérez la fonction "**envoyer texte**" en mettant le mot "a" (comme avancer) en texte à envoyer.² Pour obtenir ceci :

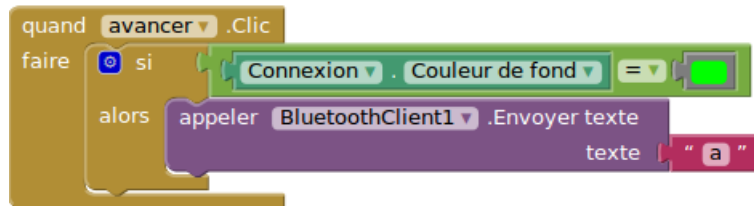


FIGURE 4.19 – Code du bouton Avancer

Il suffit de réaliser la même étape pour tous les boutons. Il faudra juste changer le "**Quand avancer.Click**" en "**Quand 'bouton'.Click**" et ne pas oublier de changer le texte à envoyer.

Par exemple, un deuxième bouton donnerait : ("g" pour gauche, "d" pour droite, "r" pour reculer, "s" pour stop, "b" pour batterie, "A" pour automatique)

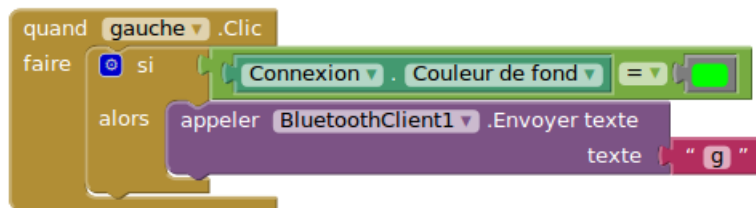


FIGURE 4.20 – Code du bouton Gauche

Après avoir fait ceci pour les autres boutons, l'application est terminée...

Nous pouvons donc passer à l'installation de l'application sur votre téléphone

2. il faut privilégier de petites chaînes de caractère afin que le module puisse lire plus facilement. Il y a moins de perte de données

Chapitre 5

Installation de l'application

5.1 Installation de MIT App Inventor

Pour installer l'application que vous avez réalisé, il est préférable d'installer l'application MIT AI2 Companion disponible sur Play Store. Cette application (relativement légère (50 Mo) va faire le lien entre le site App Inventor et votre téléphone. Une fois que votre application est terminée, voici la façon la plus simple pour l'installer :

- Installer MIT AI2 Companion sur votre téléphone

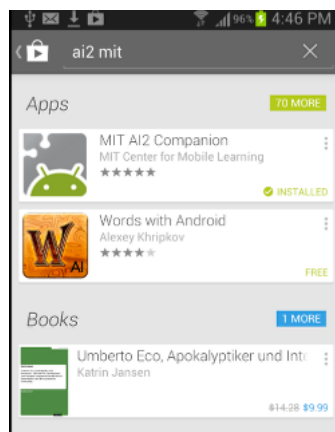


FIGURE 5.1 – Logo de l'application

- Lancer l'application MIT AI2 Companion lorsque'elle est installée **sur votre téléphone**

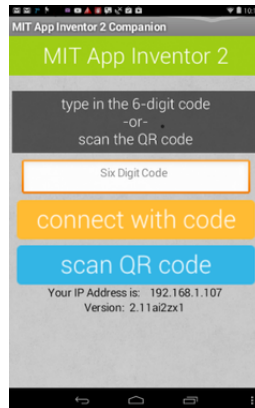


FIGURE 5.2 – Rendu de l'application MIT AI2 Companion

- Connecter vous au réseau Wifi de votre maison **depuis votre téléphone**
- Ouvrez votre application (Cocci-Bot) dans App Inventor (<http://appinventor.mit.edu/>) **depuis votre ordinateur**
- Rendez-vous dans le menu du haut, sélectionner **Construire** puis **App** (**Donnez le code QR pour fichier .apk**)

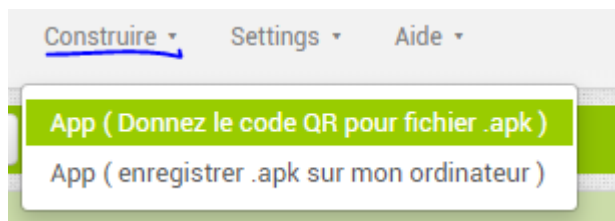


FIGURE 5.3 – Emplacement du menu "Construire"

Une barre de progression devrait apparaître.

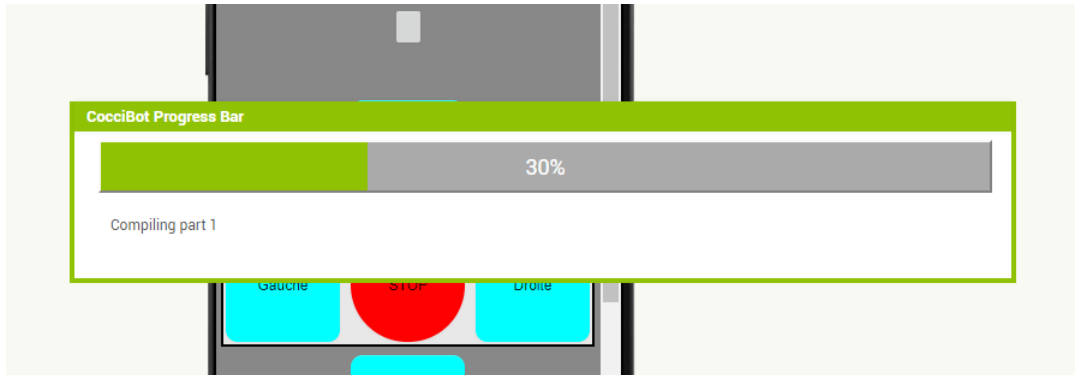


FIGURE 5.4 – Barre de progression

Une fois la barre de progression disparue, un QR Code apparaît.

Ne surtout pas cliquer sur **ok**



FIGURE 5.5 – QR code

- Il est temps de scanner le code avec l'application MIT AI2 Companion sur votre téléphone ("Scan with QRCode"). Une fois le processus terminé, vous pouvez cliquer sur **ok**, il faudra suivre les consignes sur votre téléphone portable (gestion des autorisations...)
- Et voilà !

Nous pouvons passer au code Arduino et au traitement des données.

Chapitre 6

Traitement des données

6.1 Branchements

Comme dit au chapitre 2, il faut relier le « +5V » du module au 5 Volts de la carte Arduino et la masse du module à celle de la carte. Ensuite, nous allons relier la broche TX du module à la broche 12 de la carte et la broche RX à la broche 10. Voilà, le branchement est terminé.

6.2 Code Arduino

Point-clé

Afin de lire les données du module, nous allons "émuler" une voie série, en l'occurrence les broches 10 et 12.

C'est-à-dire que nous allons déclarer que ces broches recevront et enverront des données.

Pour cela, il faut utiliser la bibliothèque **SoftwareSerial**. Dans le logiciel Arduino, allez dans **croquis** puis **inclure une bibliothèque** et sélectionnez **SoftwareSerial**.

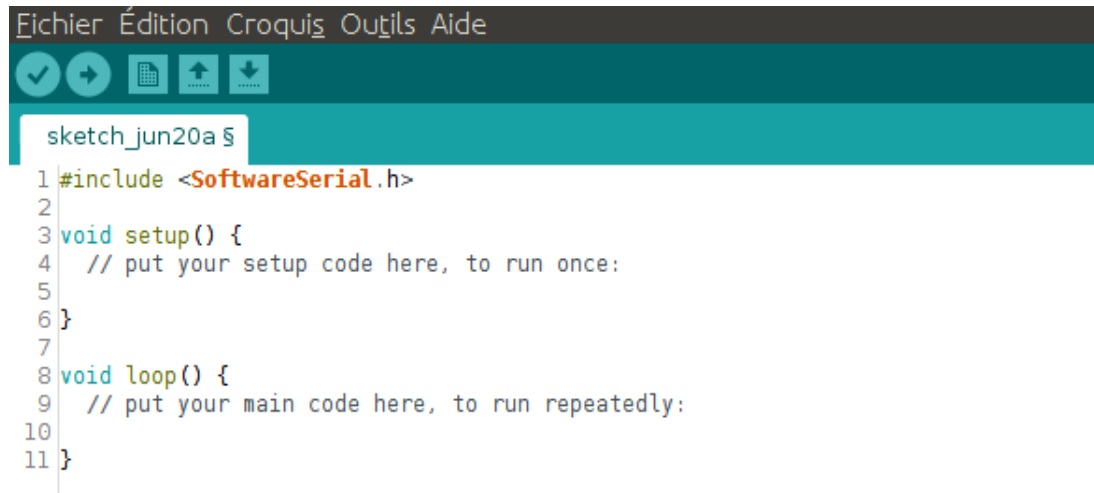


FIGURE 6.1 – Inclusion de la bibliothèque SoftwareSerial

Maintenant, nous allons définir les broches du module (toujours avant le « void setup ») :

```
const int RX = 10;
const int TX = 12
```

Définitions des broches RX et TX

Après ceci, il faut déclarer un objet **SoftwareSerial** qui prendra en argument respectivement les broches TX et Rx, un peu comme lors de la déclaration d'un écran LCD (« LiquidCrystal lcd (RS,E,D4,D5,D6,D7) ; »).

On obtient donc :

```
#include <SoftwareSerial.h>

const int RX = 10;
const int TX = 12;

SoftwareSerial crius(RX,TX);
```

Définition de l'objet SoftwareSerial

Bien entendu, "crius" peut être remplacé par ce que vous voulez. Ensuite, on déclare que la communication carte-module peut débuter avec "crius.begin(115200);" Où 115200 correspond à la vitesse de transmission en bauds (comme "Serial.begin(115200);");

```
SoftwareSerial crius(RX,TX);
```

```
void setup() {  
    crius.begin(115200);  
}
```

Vitesse de communication

Remarque importante

Par la suite, en cas d'erreurs de transmission Bluetooth (pas de données...), il conviendra de vérifier le branchement des broches RX et TX (essayer de les intervertir) et d'éventuellement changer la vitesse de communication car certains modules communiquent à 9600 bauds !

Pour lire les données du module, ce sont presque les mêmes fonctions que pour le port série :

En effet :

Pour le port série :

- `Serial.begin(115200);`
- `Serial.available();`
- `Serial.read();`
- `Serial.print();`
- `Serial.println();`

Pour le module Bluetooth :

- `crius.begin(115200);`
- `crius.available();`
- `crius.read();`
- `crius.print();`
- `crius.println();`

Tant que des données (caractères) sont disponibles, nous allons les assembler en une chaîne de caractère (concaténation).
Ensuite, avec un **if**, nous allons voir si cette chaîne en question correspond par exemple à "a".
Si c'est la cas, nous allons appeler la fonction **robot.enAvant(100)** ;

Il faut donc définir un caractère x et une chaîne de caractère. Donc dans le programme Arduino, avant la fonction **setup**, on rajoute :

```
#include <SoftwareSerial.h>

const int RX = 10;
const int TX = 12;

char c;
String message;
```

Définition des structures du message

La boucle while va permettre de lire les données : Dans la boucle **loop** : on écrit :

```
void loop() {

    while() {

    } //Fin while

} //Fin void loop
```

Boucle de lecture partielle

Maintenant que la boucle va attendre des données, il suffit de les lire et de les transformer en chaîne de caractère.
Pour cela :

- on lit le premier caractère c
- on définit que la chaîne message = message + c

On obtient :

```
void loop() {  
  
    while(crius.available()>0) {  
  
        c = crius.read();  
        message = message + c;  
    }//Fin while  
  
}//Fin void loop
```

Boucle de lecture complète

La structure conditionnelle est très simple :
Après la boucle « while », mettez :

```
void loop() {  
  
    while(crius.available()>0) {  
  
        c = crius.read();  
        message = message + c;  
    }//Fin while  
  
    if(message=="c") {  
  
        message="";  
        Melodie(1);  
    }//Fin if message=="c"  
  
}//Fin void loop
```

Structure conditionnelle

« c » correspond au message envoyé par l'application lors de l'appui du bouton connexion.

Attention!!! il est important de vider la chaîne de caractère avec "message= " ";", sinon, lorsque le module recevra un autre message (par exemple "s"), la chaîne sera égale à "cs"

Après avoir fait ceci pour les boutons connexion, avancer, droite, gauche, stop, batterie et reculer, voici le résultat :

```
void loop() {  
  
    while(crius.available()>0) {  
  
        c = crius.read();  
        message = message + c;  
    }//Fin while  
  
    if(message=="c") {  
  
        message="";  
        Melodie(1);  
    }//Fin if message=="c"  
  
    if(message=="c") {message="";  
                        Melodie(1);}  
  
    else if(message=="a"){message="";  
                           robot.enAvant(100);  
                           etatMoteurs=true;}  
  
    else if(message=="r") {message="";  
                           robot.enArriere(100);  
                           etatMoteurs=false;}  
  
    else if(message=="d") {message="";  
                           robot.tourneDroite(100);  
                           etatMoteurs=true;}  
  
    else if(message=="g") {message="";  
                           robot.tourneGauche(100);}  
  
    else if(message=="s") {message="";  
                           robot.arret();}  
  
    else if(message=="b") {message="";  
                           TestBatterie();}
```

```
else if(message=="A") {gestion_mouvement();}  
  
} //Fin void loop
```

Code des if

Pour finir, nous allons nous occuper de la partie automatique, qui s'activera avec l'appui sur le bouton auto.
Le début est le même, avec le **if**, en revanche, cette fois-ci, nous ne viderons pas la chaîne de caractère.

```
void gestion_mouvement() {  
  
    while(message=="A") {  
  
        robot.enAvant(100);  
  
        distance = moyenneMesure(30, A2);  
        fin_automatique();  
        if(distance>500) {  
  
            robot.tourneGauche(100);  
            delay(1500);  
            distance = moyenneMesure(30, A2);  
            fin_automatique();  
            if(distance>500) {  
  
                robot.tourneDroite(100);  
                delay(3000);  
  
                distance = moyenneMesure(30, A2);  
                fin_automatique();  
                if(distance>500) {  
  
                    robot.tourneGauche(100);  
                    delay(1500);  
                }  
            }  
            else{robot.enAvant(100);}  
        }  
    }  
}
```

```
        else {robot.enAvant(100);}

    }
    else {robot.enAvant(100);}

} //fin while
} //fin gestion mouvement
```

Code gestion_mouvement()

On remarque que dans cette fonction, il y a une boucle "while". La boucle "while(message== "A") " s'exécutera tant que l'on n'appuiera pas sur un autre bouton ;

En effet, si j'appuie sur le bouton stop, la chaîne message ne sera plus égale à "A", la boucle "while" ne sera plus respectée et le robot s'arrêtera.

Afin de sortir de la boucle "while", il faut intercaler dans l'algorithme plusieurs fois une fonction **fin_automatique** » qui comprend :

```
void fin_automatique() {

if(blueetooth.available()>0) {
    robot.arret();
    message="";
    etatMoteurs=false;
} //fin if
} //fin fin_automatique
```

Code fin_automatique()

Dès qu'une donnée est présente, cela arrête le robot, vide la chaîne de caractère. Le programme revient donc à la boucle "while" de départ, au tout début du **void loop**.

Si vous voulez ajouter un bouton sur l'application, vous savez le faire. Pour ajouter ce bouton dans le code Arduino, il suffit de rajouter une ligne :

```
if(message=="valeur envoyée depuis l'application") {message=
    "";

    action();}
```

Code d'ajout de touche

6.3 Conclusion

Le programme complet du cocci-bot, l'application (que vous pourrez modifier) ainsi que les branchements se situent en annexe du dossier