

Introduction à MIT App Inventor - création d'une application pour contrôler à distance une voiture robotisée DIY



Présentation des 5 grandes idées en intelligence artificielle utilisant l'Internet des objets dans l'éducation STEM

T2.4 Conception de projets IoT et développement de ressources

Projets IoT AI4STEM

Projet: Voiture robotique IoT

Copyright

© Copyright au AI4STEM Consortium
2022-1-FR01-KA220-SCH-000085611
Tous droits réservés.



Projets IoT AI4STEM Projet: Voiture robotique IoT © 2023 par [AI4STEM CONSORTIUM](#) est sous licence [Attribution - Utilisation non commerciale - Partage dans les mêmes conditions 4.0 International](#)

Table des matières

1. Création d'une application pour le contrôle à distance de la voiture robotique DIY	3
1.1 Introduction	3
1.2 Planification de la conception de l'application	3
1.3 Conception de l'application	3
1.4 Programmation de l'application	15
1.5 Création de l'application	22
1.6 Association de l'application à la voiture robotisée	23

1. Création d'une application pour le contrôle à distance de la voiture robotique DIY

1.1 Introduction

Ce document présente une activité de mise en condition pour initier vos étudiants à l'environnement App Inventor du MIT. Grâce à cette activité, les étudiants apprendront à créer une application qui permettra de contrôler à distance la voiture robotisée, par le biais d'un appareil intelligent. Pour ce faire, ils apprendront à concevoir l'interface de l'application et à programmer les éléments qu'elle contient. Avant de faire cette activité, il est vivement conseillé de demander à vos élèves de créer le script décrit dans le fichier "T2.4_Programming_the_robotic_car.pdf", ou de le télécharger sur la voiture robotisée, en utilisant le fichier .hex correspondant.

1.2 Planification de la conception de l'application

Avant de passer à la conception de l'application, il est essentiel de connaître les composants qui devront être inclus. L'un des principaux objectifs est de trouver un moyen d'établir une connexion et une communication entre notre appareil intelligent et notre voiture robotisée. Pour ce faire, nous allons définir des messages qui seront envoyés/transmis par notre appareil intelligent et qui seront reçus par notre voiture robotisée via Bluetooth. Chaque fois qu'un message est envoyé/transmis par l'appareil intelligent et reçu par la carte micro:bit, la voiture robotique répondra d'une manière différente.

Dans cette optique, nous devons concevoir une application qui pourra :

- contrôler les mouvements de notre voiture robotique. Pour ce faire, nous avons besoin de 5 boutons : 4 boutons permettront à notre voiture robotisée d'avancer, de reculer, d'aller à droite et à gauche, et 1 bouton permettra d'arrêter tout mouvement.
- permettre à notre application d'être connectée via le Bluetooth de notre appareil
- permettre à notre application de scanner/rechercher les appareils Bluetooth disponibles et de se connecter à l'appareil choisi
- déconnecter notre application de l'appareil Bluetooth connecté
- éventuellement, nous informer de l'état actuel de la connectivité

Tous ces éléments nous aident à nous faire une idée concrète du résultat du processus de conception.

1.3 Conception de l'application

La conception est un processus assez libre, qui repose principalement sur l'esthétique du créateur. Les instructions suivantes sont indicatives et présentent une version assez simplifiée de l'apparence de l'interface que notre application peut avoir.

La figure 1 présente un aperçu de l'interface que nous allons créer sur la base des besoins que nous avons enregistrés dans la section précédente.

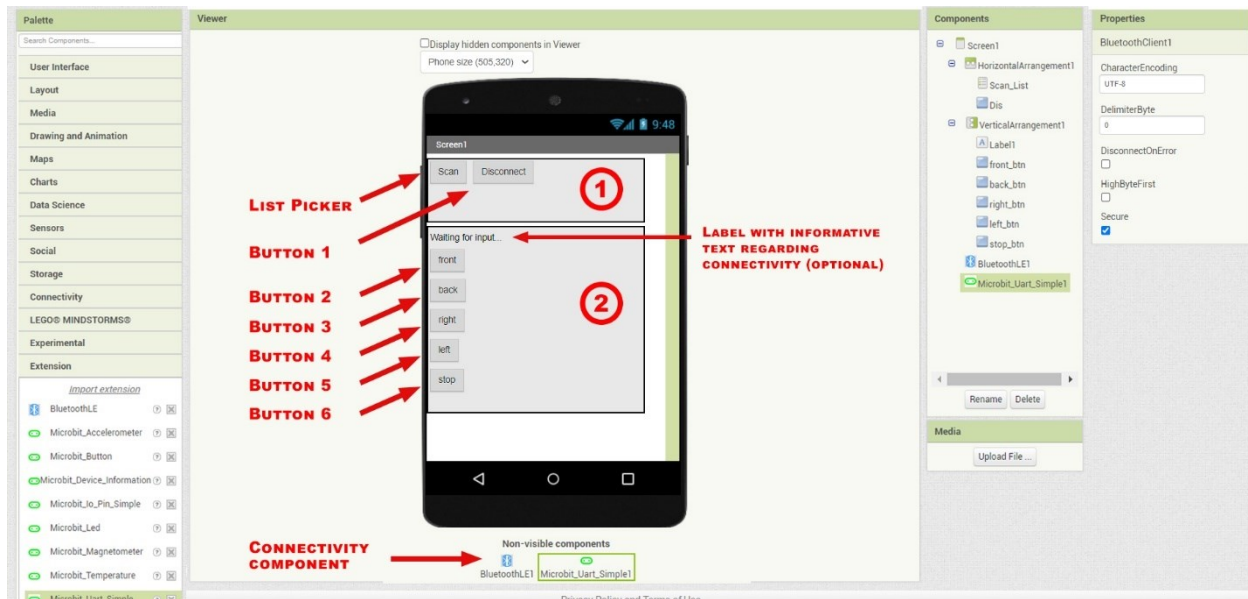


Figure 1 : Aperçu de l'interface que vous êtes sur le point de concevoir

Pour mieux organiser tous les composants sur notre écran, nous voulons créer deux présentations. L'une accueillera le bouton Scan/Search et le bouton Disconnect et nous permettra de les disposer en ligne (c'est-à-dire l'un à côté de l'autre) (1), et l'autre accueillera les boutons de navigation et nous permettra de les disposer en colonne (c'est-à-dire l'un par dessous de l'autre) (2).

Examinons de plus près la fonctionnalité de chaque bouton :

- 1) **Scanner** : Le bouton "Scan" devrait ouvrir une liste de tous les appareils Bluetooth Low Energy disponibles dans la zone. Dans cette liste, l'utilisateur doit choisir l'adresse Bluetooth du Micro:bit. La connexion sera alors automatiquement établie. Ce bouton est différent car il redirige l'utilisateur vers une liste de toutes les connexions Bluetooth disponibles. Pour activer cette fonction, nous ajouterons un bouton "ListPicker", qui sera décrit plus loin dans ce guide.
- 2) **Déconnexion** : Lorsque le bouton "Disconnect" est enfoncé, la connexion entre le micro:bit et l'appareil intelligent de l'utilisateur est désactivée.
- 3) **Boutons de navigation (avant, arrière, etc.)** : Lorsque l'on appuie sur l'un de ces boutons, notre voiture robotisée se déplace dans la direction correspondante.

Conseil : Au cours de cette étape, vous pouvez conseiller/encourager vos élèves à créer un croquis ou un diagramme de l'interface et des éléments qui y sont inclus, ou/et une liste de tous les éléments nécessaires. De cette manière, ils seront en mesure de mieux organiser les étapes vers la réalisation de la tâche actuelle.

Création de la mise en page pour placer/arranger les boutons

La première étape de la création de notre application consiste à définir les deux dispositions qui accueilleront et organiseront tous les boutons et toutes les étiquettes nécessaires. Pour ce faire, nous

allons ajouter deux éléments de mise en page, à savoir une mise en page horizontale qui accueillera le ListPicker et les boutons de déconnexion, et une mise en page verticale qui accueillera les boutons de navigation. Par conséquent, à partir de l'onglet "Disposition", nous ferons glisser les composants "Disposition horizontale" (1) et "Disposition verticale" (2), et nous les déposerons sur l'écran (Figure 2).

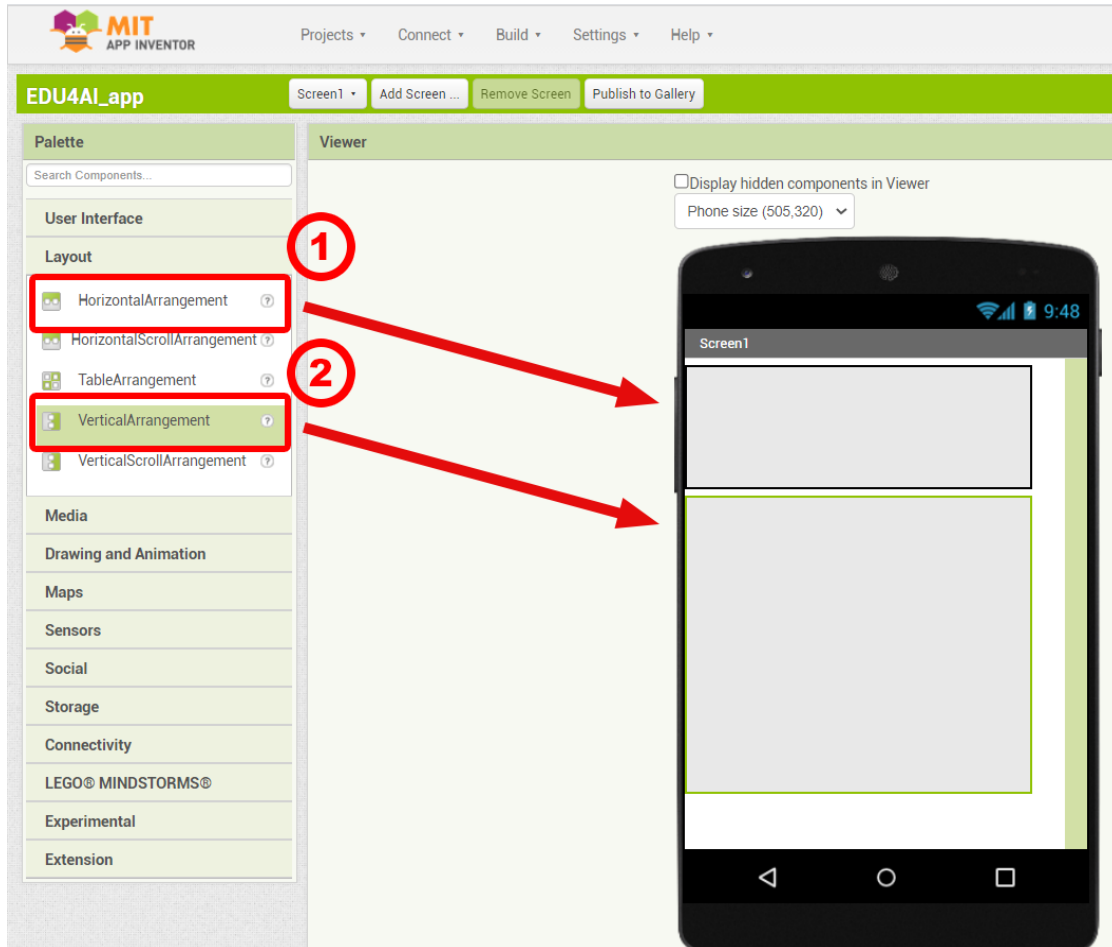


Figure 2 : Glisser-déposer les deux éléments de la mise en page sur l'écran

Après avoir placé les deux composants de mise en page sur notre écran, nous pouvons ajuster un certain nombre de caractéristiques telles que leur hauteur et leur largeur à partir de la section "Properties" (4). Pour ce faire, nous devons sélectionner le composant correspondant dans la liste des composants (3). Dans l'exemple présenté dans la figure 3, nous avons fixé la largeur du composant "HorizontalArrangement" à 90 %, et la largeur et la hauteur du composant "VerticalArrangement" à 60 et 90 % respectivement.

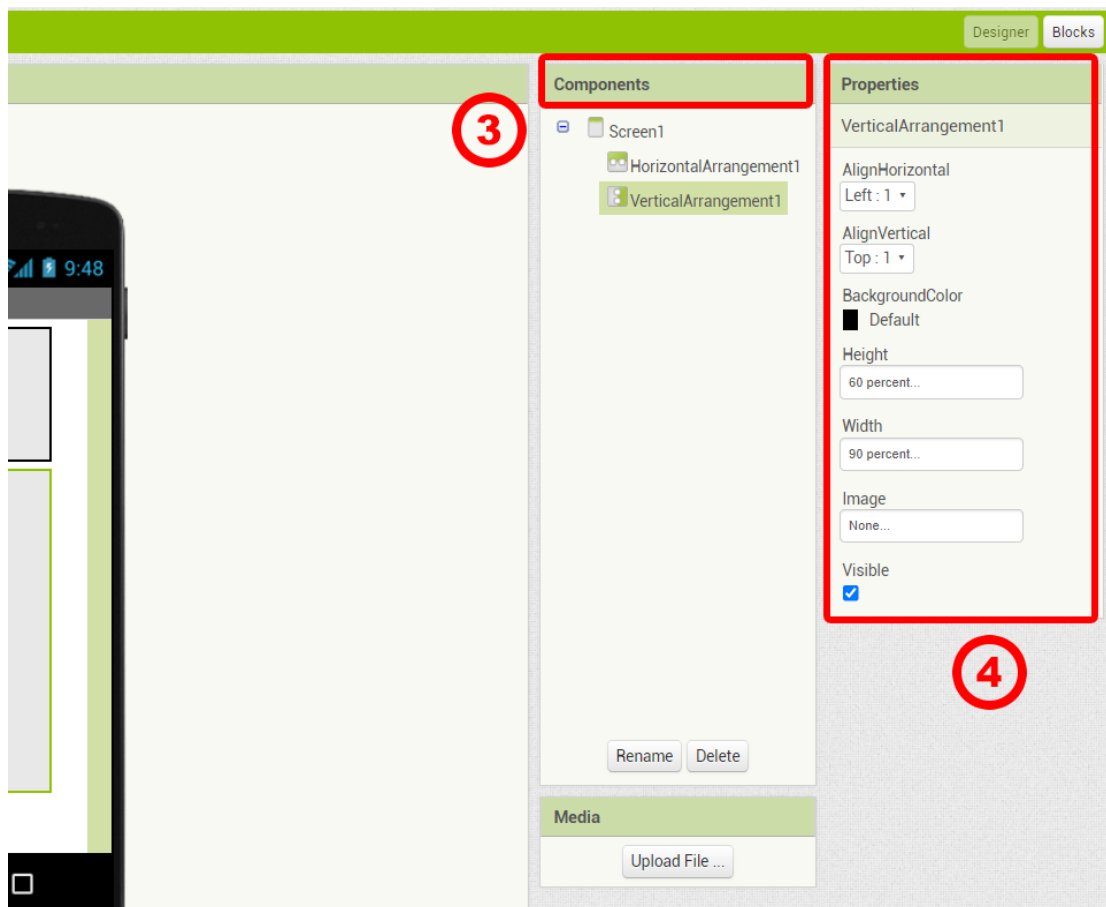


Figure 3 : Ajustement des caractéristiques de chaque composant

Conseil : si vous le souhaitez, vous pouvez revoir et modifier les caractéristiques susmentionnées après avoir ajouté les boutons dans les composants de la mise en page.

Ajout du ListPicker et des boutons

L'étape suivante consiste à ajouter le ListPicker et les boutons à l'écran. Pour ce faire, nous allons dans le sous-menu "Interface utilisateur" et à partir de la section "Palette", nous allons glisser et déposer sur l'écran 7 éléments, à savoir un "ListPicker" et 6 boutons. Le ListPicker sera utilisé pour rechercher/scanner et révéler tous les périphériques Bluetooth disponibles.

Plus précisément, nous faisons glisser l'élément "ListPicker" **(2)** et un bouton **(1)** (c'est-à-dire le bouton de déconnexion) et nous les déposons dans la disposition "HorizontalArrangement", puis nous faisons glisser 5 autres boutons **(1)** (c'est-à-dire les boutons de navigation) et nous les déposons dans la disposition "VerticalArrangement" (Figure 4).

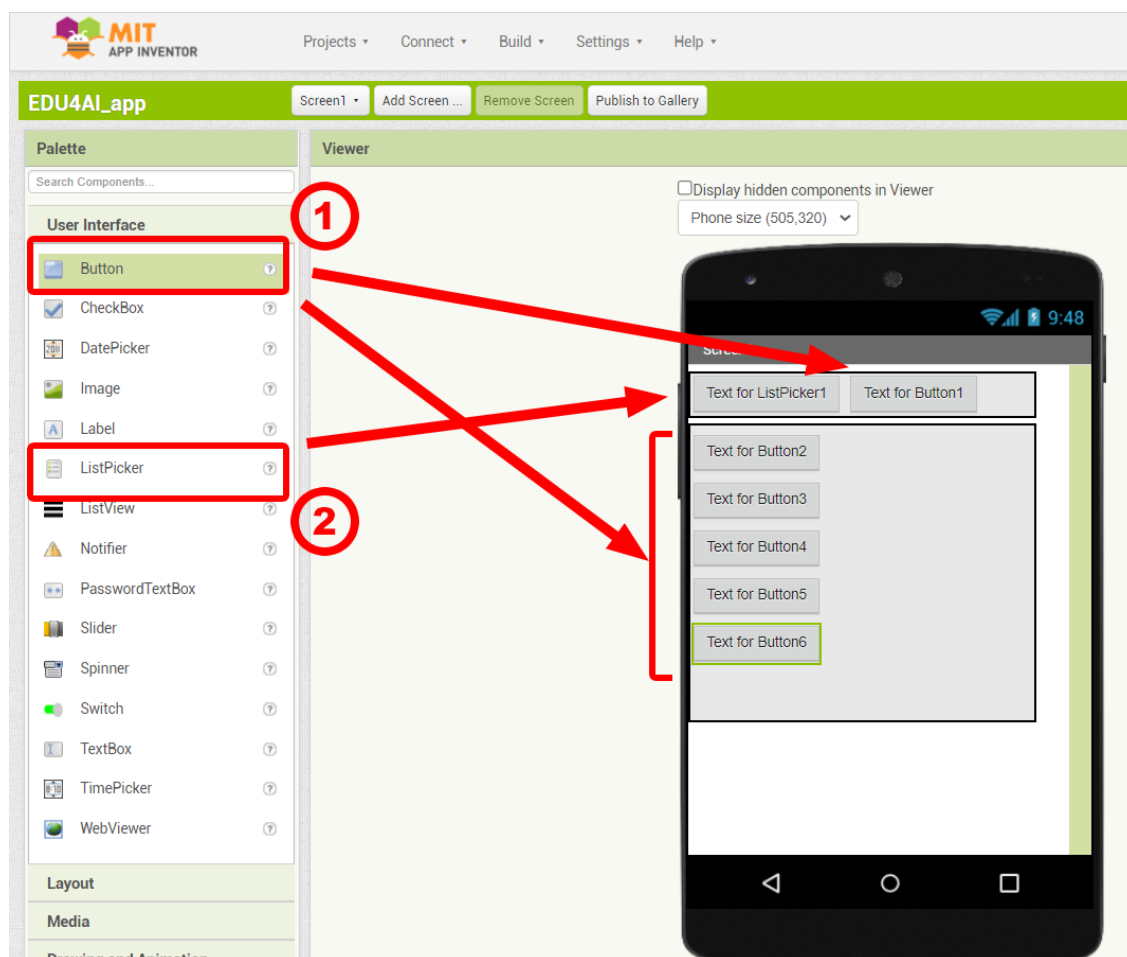


Figure 4 : Ajout d'un élément ListPicker et de 6 boutons à l'écran

Afin d'optimiser l'aspect de notre interface, nous pouvons modifier le nom de chaque bouton via l'onglet "Texte" (4) situé dans le menu "Propriétés". Pour cela, il faut sélectionner l'élément correspondant dans la liste "Composants" (3) (le bouton 3 dans notre exemple) et changer le nom manuellement (Figure 5).

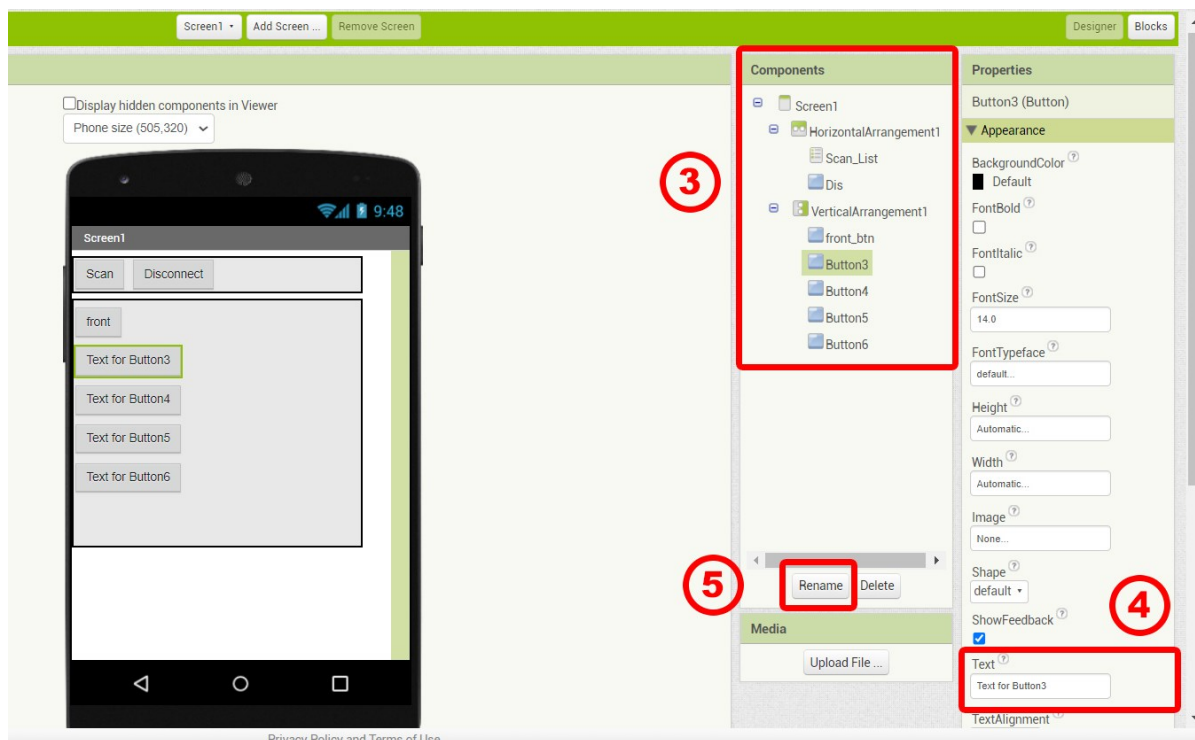


Figure 5 : Modification des noms des boutons

Une autre bonne pratique consiste à changer le nom des boutons. Cela facilitera également la phase de codage ultérieure. Vous pouvez le faire en cliquant sur le bouton "Renommer" (5), et en insérant le nouveau nom dans la zone de texte "Nouveau nom" située dans le menu contextuel (Figure 5, Figure 6).

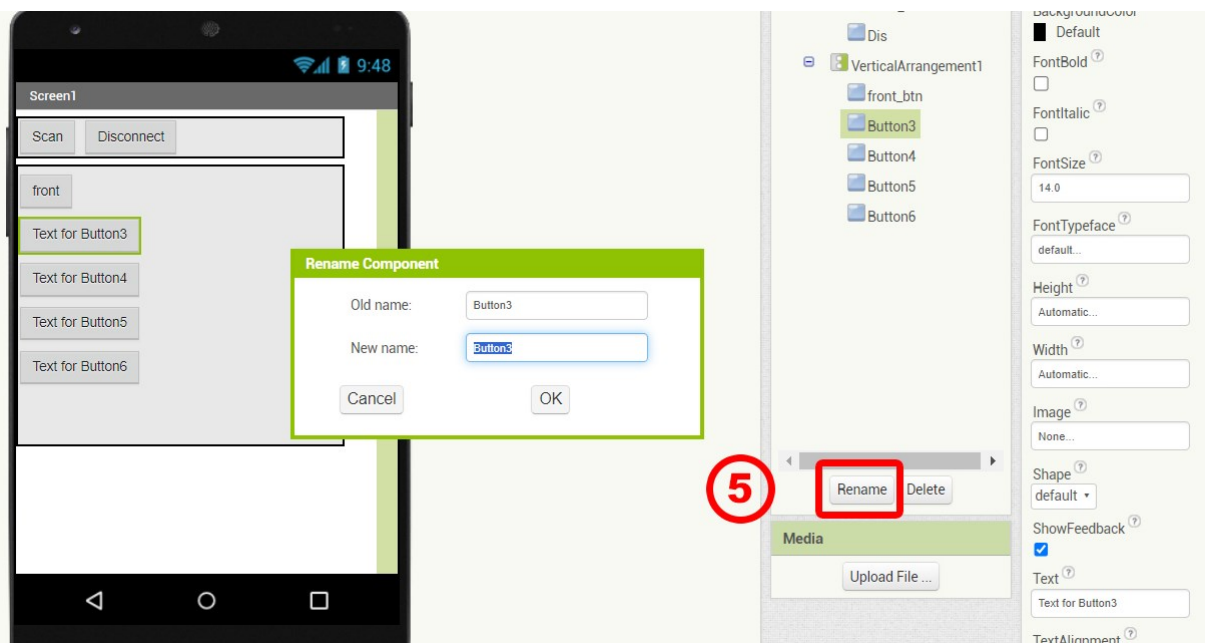


Figure 6 : Renommer le composant Bouton

Conseil : les noms des boutons sont indicatifs et n'affectent pas leur fonctionnalité. Toutefois, une bonne pratique consiste à utiliser des noms significatifs pour l'ensemble du processus (par exemple, donner le nom "front_btn" au bouton qui fera avancer la voiture, "back_btn" à celui qui fera reculer la voiture, etc.) Dans notre exemple, nous renommons les composants comme suit :

Composant	Nom du texte (modifié à partir de l'onglet Texte dans les Caractéristiques)	Nouveau nom (remplacé par Renommer dans la liste des composants)
ListPicker	Scanner	Liste Scan_List
Bouton1	Déconnexion	Dis
Bouton2	avant	front_btn
Bouton 3	retour	back_btn
Bouton 4	droit	right_btn
Bouton5	gauche	left_btn
Bouton6	arrêter	stop_btn

Remarque importante : n'utilisez pas le même mot pour le nom du texte et le nom du bouton, car cela entraînerait un dysfonctionnement d'App Inventor, l'empêchant de créer l'application.

En résumé, le ListPicker "Scan" recherchera les appareils Bluetooth disponibles, tandis que le bouton "Disconnect" interrompra la connexion entre notre appareil intelligent et le module Bluetooth. Les autres boutons seront utilisés pour naviguer notre voiture robotisée.

Conseil : gardez à l'esprit que les lignes directrices susmentionnées sont indicatives. N'hésitez pas à expérimenter avec les formes et les couleurs différentes pour chaque bouton, voire avec les dispositions différentes, afin de créer une interface unique et plus attirante sur le plan visuel.

Ajout d'une notification textuelle

L'étape suivante consiste à ajouter une étiquette de texte à notre écran. Cette étape n'est pas obligatoire, mais elle est très utile, car elle nous informera si la connexion avec notre appareil intelligent est établie ou non. Trouvez "Label" dans le sous-menu "User Interface", puis glissez et déposez-le sur l'écran (*Figure 7*). Changez le contenu du texte en "Waiting for input" (ou en quelque chose de similaire, selon votre préférence, de la même manière que vous avez changé le contenu du texte pour les autres composants).

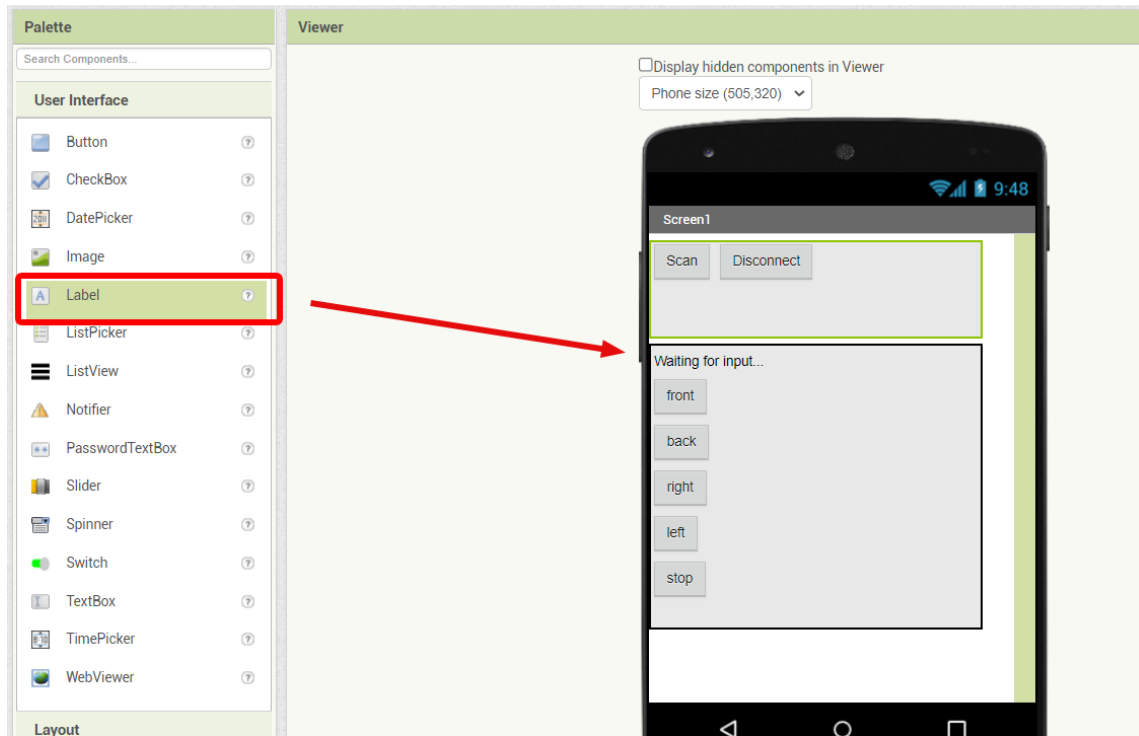


Figure 7 : Ajout d'une étiquette qui nous informe sur l'état de la connectivité

Ajout d'extensions

L'étape suivante consiste à ajouter quelques composants qui permettront la connexion entre notre application et la voiture robotisée. Pour ce faire, nous devons ajouter les extensions suivantes à notre application : l'extension BluetoothLE et l'extension Microbit_Uart_Simple. La première est utilisée pour établir la connexion Bluetooth entre notre appareil intelligent et le Micro:bit, tandis que la seconde sert à envoyer les messages appropriés une fois la connexion établie.

Pour pouvoir utiliser ces extensions, vous devez les télécharger localement sur votre ordinateur. Pour ce faire, cliquez ici <https://mit-cml.github.io/extensions/> et téléchargez sur votre ordinateur les fichiers *BluetoothLE.aix* et *Microbit.aix* (Figure 8).

MIT APP INVENTOR

Home Directory Documentation

Supported:

Name	Description	Author	Version	Download .aix File	Source Code
BluetoothLE	Adds as Bluetooth Low Energy functionality to your applications. See BluetoothLE Documentation and Resources for more information.	MIT App Inventor	20230728	BluetoothLE.aix	Via GitHub
FaceMeshExtension	Estimate face landmarks with this extension.	MIT App Inventor	20210414	Facemesh.aix	Via GitHub
LookExtension	Adds object recognition using a neural network compiled into the extension.	MIT App Inventor	20181124	LookExtension.aix	Via GitHub
Microbit	Communicate with micro:bit devices using Bluetooth low energy (needs BluetoothLE extension above).	MIT App Inventor	20200518	Microbit.aix	Via GitHub
PersonalAudioClassifier	Use your own neural network classifier to recognize sounds with this extension.	MIT App Inventor	20200904	PersonalAudioClassifier.aix	Via GitHub
PersonalImageClassifier	Use your own neural network classifier to recognize images with this extension.	MIT App Inventor	20210315	PersonalImageClassifier.aix	Via GitHub
PosenetExtension	Estimate pose with this extension.	MIT App Inventor	20200226	Posenet.aix	Via GitHub
TeachableMachine	Use vision models trained in TeachableMachine with your device's camera.	MIT App Inventor	1	TeachableMachine.aix	Via GitHub

Figure 8 : Recherche des extensions à télécharger

Après avoir téléchargé les extensions, retournez dans App Inventor. Dans la section Palette, cliquez sur l'onglet Extension, puis sur la sélection Importer une extension (6) (Figure 9).

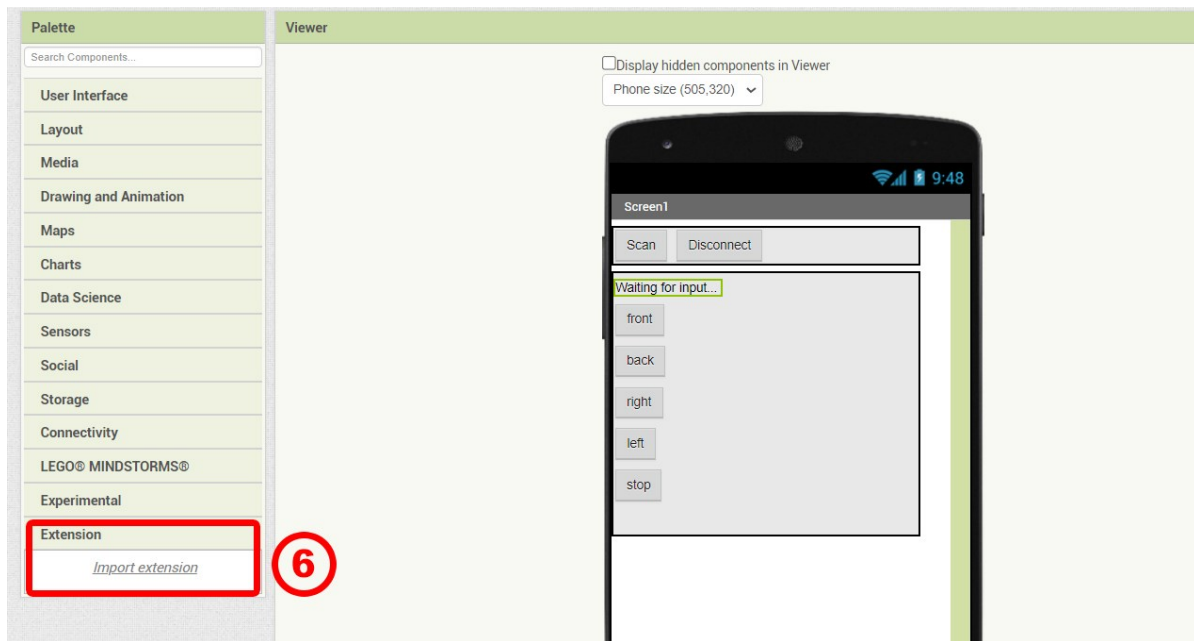


Figure 9 : La sélection de l'extension d'importation

Dans le menu contextuel, cliquez sur le bouton *Choose File* (7) pour rechercher dans votre dossier local et sélectionner l'extension téléchargée (figure 10). Assurez-vous que l'option "A partir de mon ordinateur", située au-dessus du bouton *Choisir un fichier*, est sélectionnée.

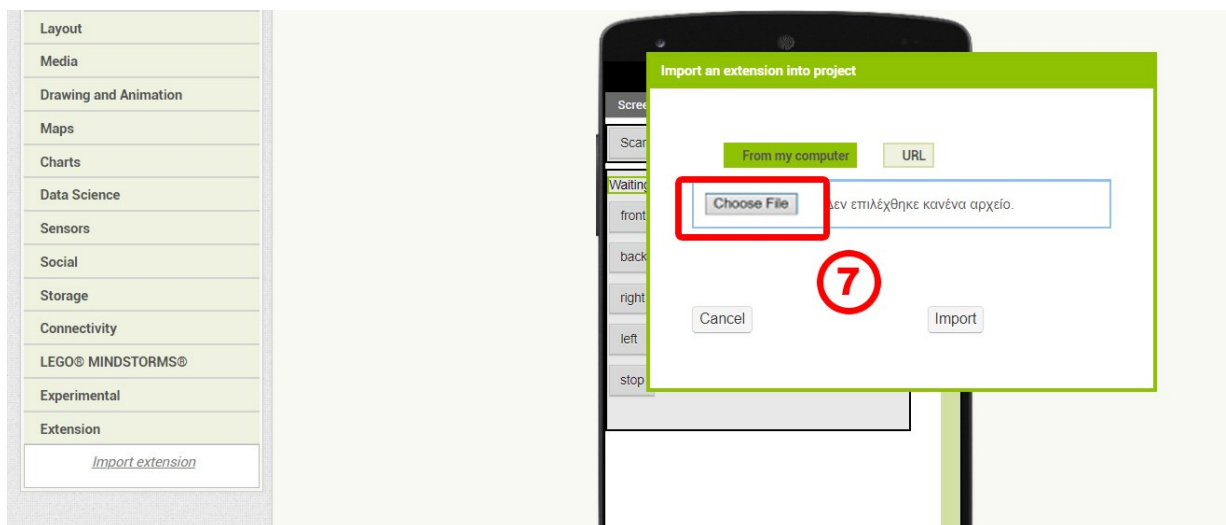


Figure 10 : Cliquez sur le bouton Choisir un fichier pour rechercher le fichier d'extension dans votre dossier local.

Après avoir trouvé et sélectionné le fichier d'extension, cliquez sur le bouton Importer (8) (Figure 11).

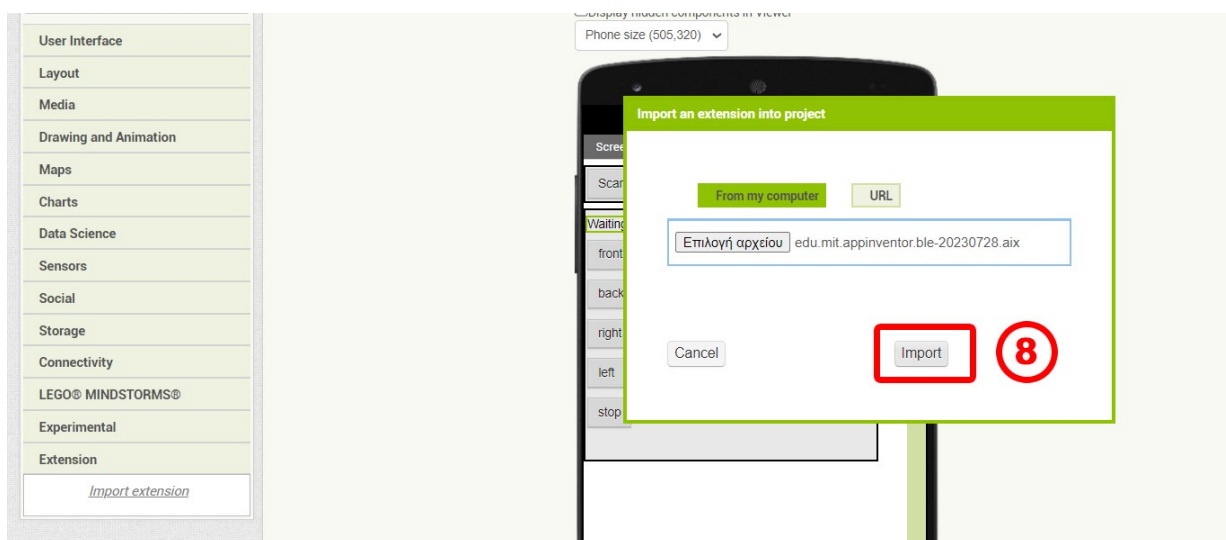


Figure 11 : Cliquez sur le bouton Importer pour importer l'extension sélectionnée

Après un certain temps, la nouvelle extension apparaîtra sous extension Importation, dans l'onglet Extension (par exemple, dans la figure 12, l'extension BluetoothLE a été importée).

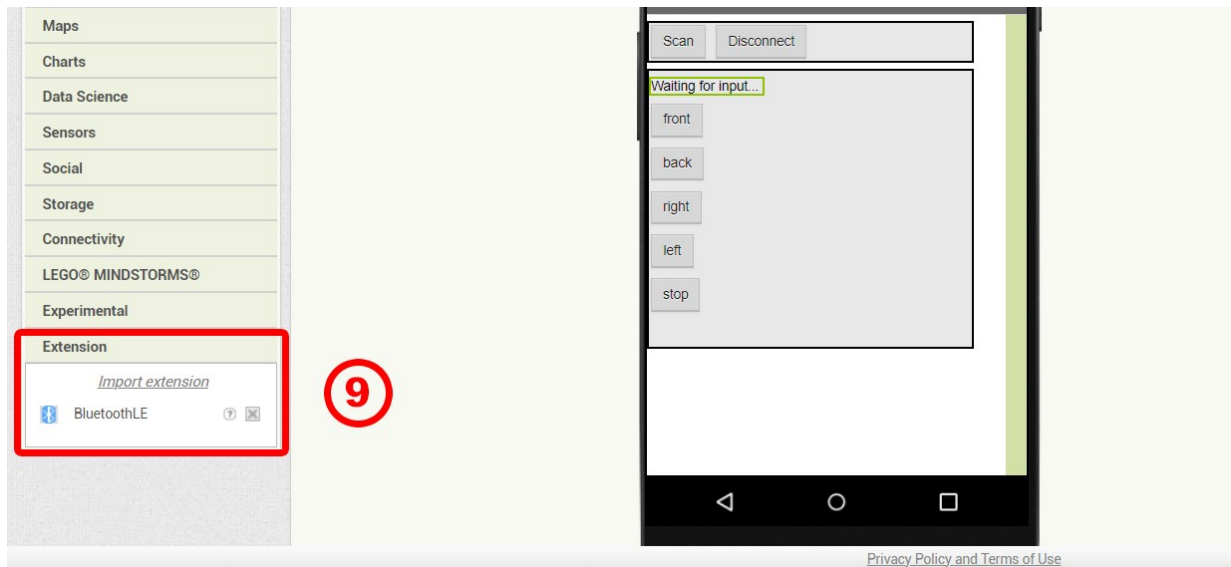


Figure 12 : L'extension BluetoothLE a été importée

Pour ajouter cette extension à l'application conçue, faites-la glisser et déposez-la dans la zone de conception (figure 13). Les extensions sont normalement des composants non visibles. Par conséquent, ces composants apparaissent sous la zone de conception, dans la section "Composants non visibles".

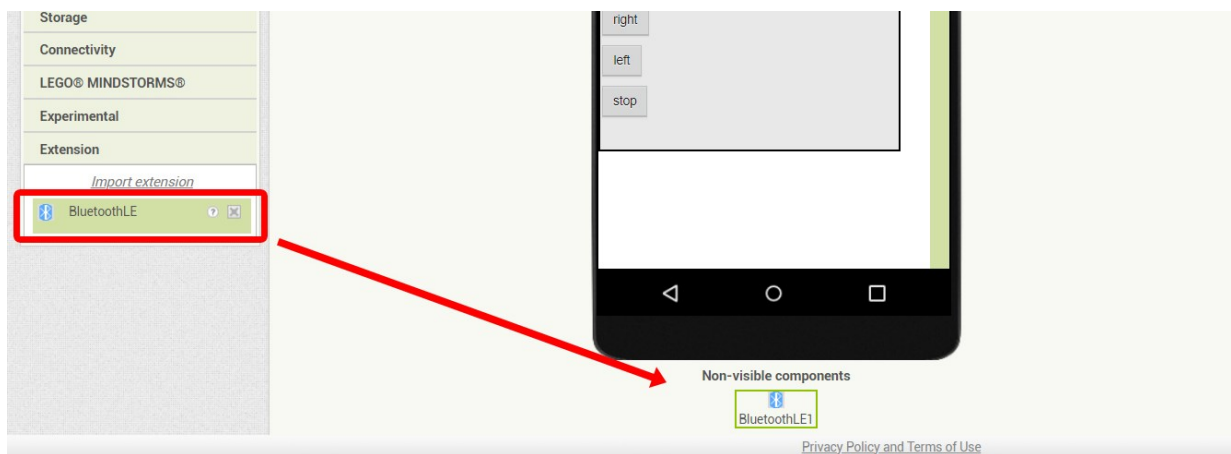


Figure 13 : Glissez-déposez l'extension BluetoothLE sur la zone de conception

Répétez le même processus pour importer l'extension Microbit_Uart_Simple.

Remarque importante : après avoir choisi et importé le fichier *Microbit.aix*, vous remarquerez que plusieurs extensions apparaissent sous l'onglet Extension. Pour cette activité, vous n'aurez qu'à glisser et déposer dans la zone de conception l'extension Microbit_Uart_Simple (Figure 14).

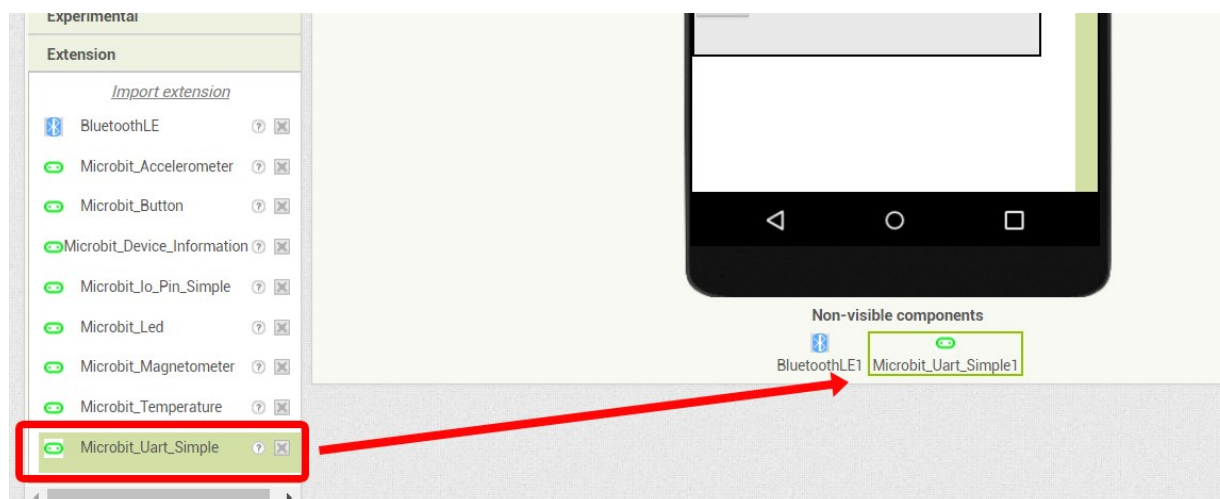
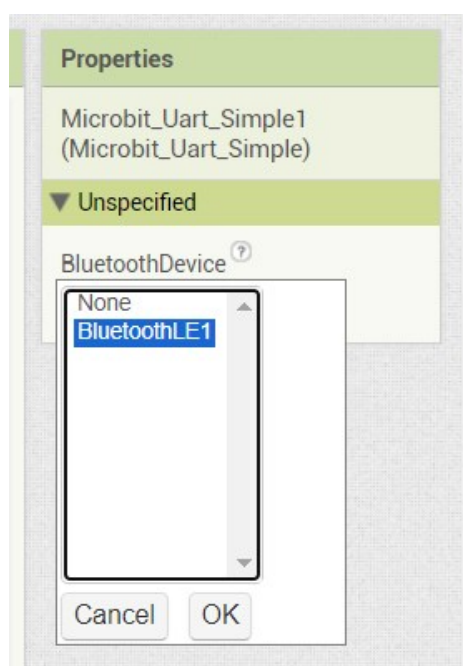


Figure 14 : Seul le Microbit_Uart_Simple doit faire l'objet d'un glisser-déposer.

Après avoir importé l'extension Microbit_Uart_Simple, allez dans le menu Propriétés et dans le champ Bluetooth device, sélectionnez BluetoothLE1 dans le menu flottant.



Après avoir ajouté toutes les extensions nécessaires, nous sommes prêts à passer à la programmation de l'application.

1.4 Programmation de l'application

Afin de faire fonctionner notre application, nous devons déterminer comment les composants ajoutés fonctionneront en spécifiant les actions qu'ils déclencheront lorsqu'ils seront enfoncés (sur l'écran de notre application). Pour ce faire, nous devons aller dans le menu Blocs et créer notre code dans l'espace de visualisation (Figure 21) en faisant glisser et en assemblant les blocs de commandes appropriés.

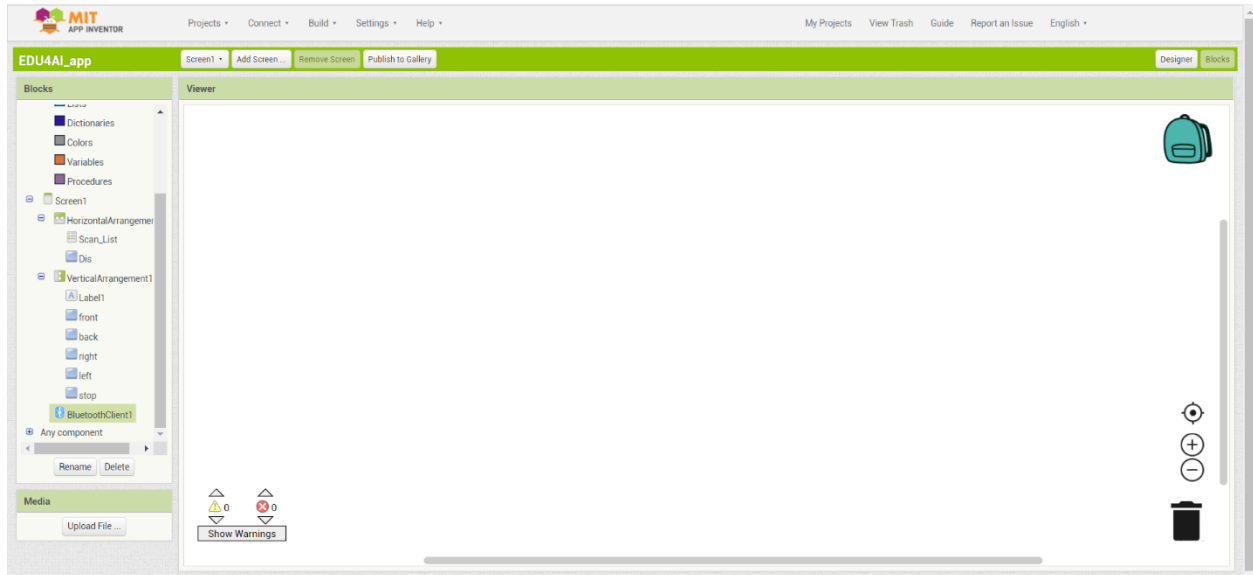
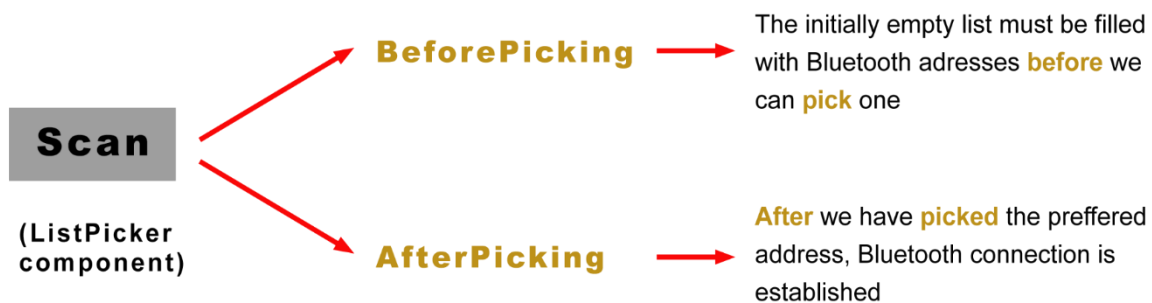


Figure 15 : Menu Bloc de l'App Inventor

Codage du composant ListPicker (c'est-à-dire Scan_List)

Nous commencerons par coder le composant ListPicker, ou le composant "Scan_List", tel qu'il est nommé dans notre exemple. Le diagramme suivant décrit les opérations/actions que nous voulons effectuer lorsque le composant ListPicker est actionné, ainsi que la commande d'événement correspondante qui doit être appliquée à cette fin. Voyons maintenant comment ces codes seront structurés.



Commande BeforePicking

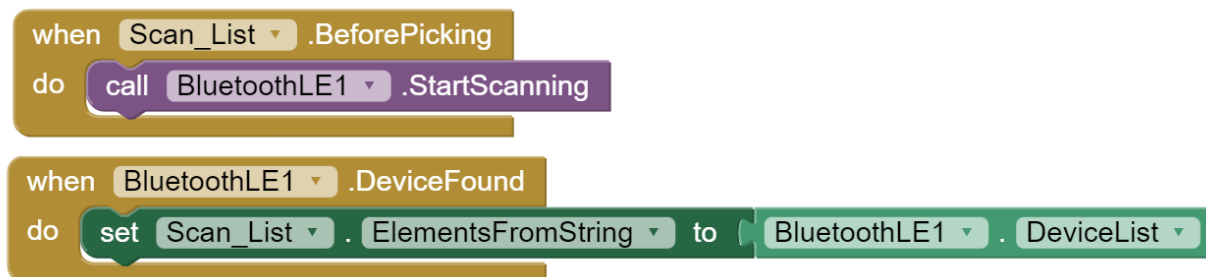
Cliquez sur l'élément "Scan_List" (1) et dans le menu flottant, sélectionnez la commande d'événement "When Scan_List BeforePicking" (2) (Figure 16).

Lorsque vous appuyez sur Scan_List, une liste des périphériques Bluetooth disponibles devrait s'afficher. "BeforePicking" signifie que nous n'avons pas encore choisi l'un des appareils Bluetooth affichés.



Figure 16 : Cliquez sur l'élément Liste d'analyse et sélectionnez la commande nécessaire dans le menu flottant.

Pour pouvoir choisir des éléments dans notre liste, celle-ci doit être initialement remplie de tous les appareils Bluetooth disponibles. (La liste est vide au début, donc avant de choisir un élément, elle doit d'abord être remplie). Pour ce faire, nous devons commencer à rechercher les appareils BLE. Ensuite, la liste est remplie avec tous les appareils disponibles trouvés :



Note : Les blocs de commandes utilisés dans le script ci-dessus peuvent être trouvés de la même manière, en cliquant sur l'élément correspondant (Scan_List ou BluetoothLE1) et en trouvant les blocs pertinents dans le menu flottant correspondant (quelques exemples sont présentés dans la Figure 17).

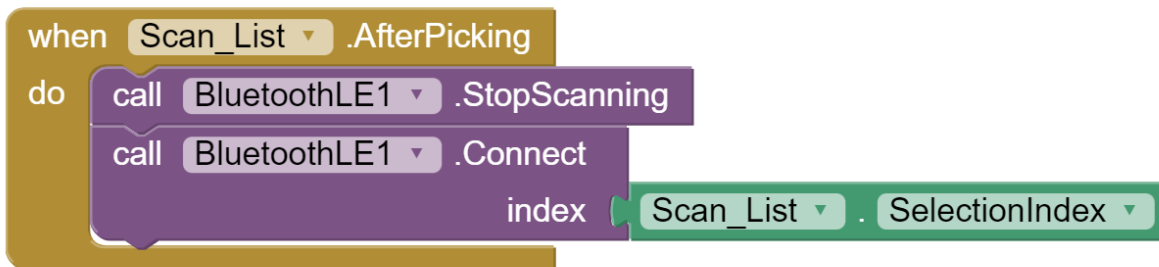


Figure 17 : Recherche de certaines commandes de bloc

Commande AfterPicking

Nous devons maintenant déterminer ce qui se passe lorsque l'utilisateur sélectionne l'adresse Bluetooth du micro:bit afin d'établir une connexion entre l'application et la voiture robotisée. Pour ce faire, nous aurons besoin de la commande d'événement **"When Scan_List AfterPicking"**.

Dans la **commande d'événement**, nous placerons quelques commandes qui détermineront l'action qui sera exécutée après avoir choisi une adresse Bluetooth dans la liste. Nous utiliserons donc la commande **"call BluetoothLE1.StopScanning"**, afin que notre appareil cesse de rechercher d'autres appareils Bluetooth LE. Ensuite, nous ajouterons la commande **"call BluetoothLE1.Connect"**, et dans son côté droit, nous insérerons la commande **"Scan_List.SelectionIndex"**, pour permettre à l'application d'être connectée au périphérique Bluetooth sélectionné.



Codage de l'étiquette (c'est-à-dire l'étiquette 1)

Pour s'assurer que la connexion Bluetooth a bien été établie, nous allons programmer l'application pour qu'elle envoie une notification pertinente. Pour ce faire, nous programmerons l'élément Label et utiliserons les blocs de code suivants :



En particulier, nous utilisons la commande "set Label1.Text to" et, sur son côté droit, nous insérons un bloc de **texte** "", dans lequel nous tapons "Connection Established".

Si la connexion est réussie, la notification "Connexion établie" apparaît à l'écran, remplaçant le message "Waiting for input..." (Figure 7). Si la connexion n'est pas établie avec succès, le message "Waiting for input..." reste inchangé.

Ainsi, après la sélection du dispositif Bluetooth (à savoir le Bluetooth du micro:bit), le composant BluetoothLE tentera de se connecter à l'adresse préférée (c'est-à-dire le micro:bit utilisé par la voiture robotisée). Si la tentative est réussie, le texte de l'étiquette "Waiting for input..." devient "Connection established!".

Note : le bloc de saisie de texte se trouve dans le menu flottant de l'onglet "Texte" (Figure 18).

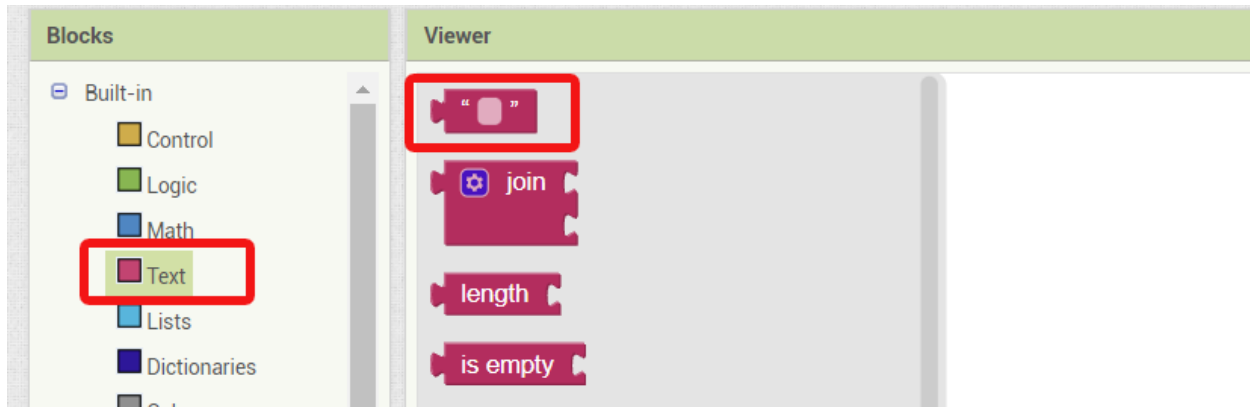


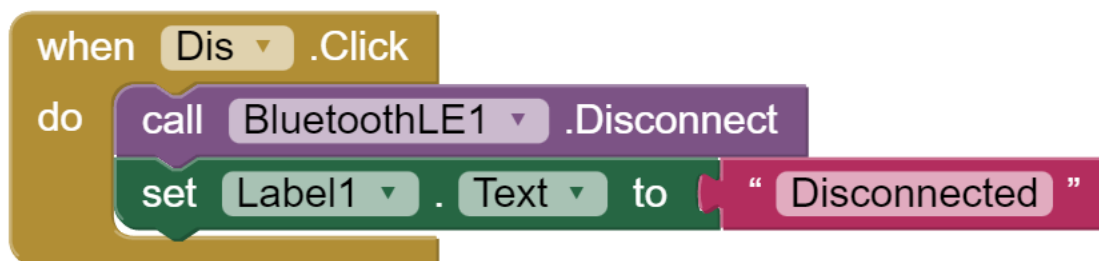
Figure 18 : Recherche du bloc de saisie de texte

Codage du bouton de déconnexion (Dis)

Dans cette étape, nous allons programmer le composant BluetoothLE pour qu'il mette fin à la connexion lorsque le bouton "Disconnect" est actionné. Le texte de l'étiquette deviendra également "Connection timed out".

Pour cette étape du codage, nous aurons besoin de la commande d'événement **"When Dis .Click...do"**.

Ainsi, grâce au script suivant, **lorsque le bouton "Disconnect" est enfoncé**, l'application est chargée d'**appeler le BluetoothLE** et de le **déconnecter** de l'appareil présélectionné. De plus, le **texte de l'étiquette est "Déconnecté"**, afin d'informer l'utilisateur que la connexion a été interrompue.



Remarque : le script d'activation des boutons "Scan" et "Disconnect" peut être difficile pour vos élèves. Par conséquent, en fonction de leur niveau, vous pouvez leur fournir ces parties du script et leur enseigner la partie suivante du script de manière approfondie.

Codage des boutons de navigation de la voiture robotisée

La dernière étape consiste à programmer les boutons de navigation. Comme mentionné précédemment dans ce document, lorsqu'un bouton de navigation est enfoncé, notre appareil intelligent transmet (via Bluetooth) un message spécifique à la carte micro:bit de notre voiture robotisée. Lorsque ce message est reçu, la voiture robotisée se comporte en conséquence, en effectuant un mouvement spécifique (c'est-à-

dire en avançant, en reculant, etc.). Le tableau suivant présente le message transmis lorsqu'un bouton spécifique est enfoncé :

Bouton enfoncé	Commande envoyée
"front"	"#forward#"
"dos"	"#backwards#"
"droit"	"#right#"
"gauche"	"#gauche#"
"stop"	"#stop#"

Note importante : Chaque message que nous voulons transmettre doit commencer et se terminer par le symbole "#", afin que le micro:bit puisse distinguer les limites du message.

Par exemple, lorsque nous transmettons le mot "#front#", le micro:bit fait tourner les moteurs CC vers l'avant.

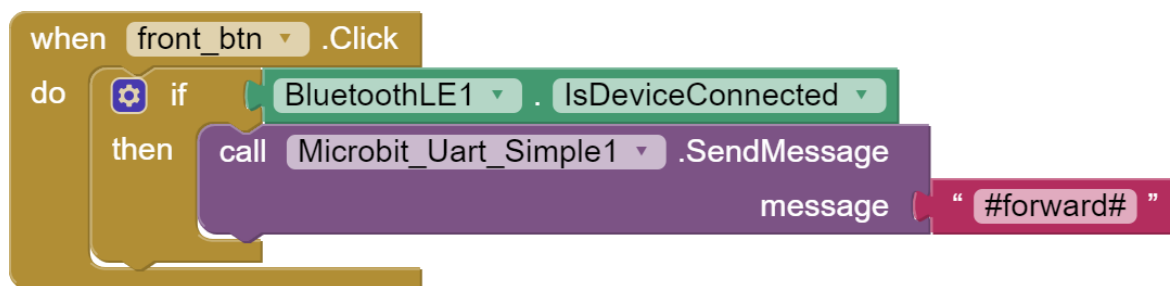
Pour cette partie du script, et pour chacun des boutons, nous aurons besoin de la commande d'événement **"When .Click...do"** et d'une condition **"if..then"**.

Remarque : la condition de bloc "if-then" se trouve sous l'onglet "Contrôle" (figure 19).



Figure 19 : Recherche de la condition du bloc "if...then" (si...alors)

Ainsi, grâce au bloc de commandes suivant, nous demandons à notre application de vérifier si le composant **BluetoothLE s'est connecté à l'appareil** souhaité, et si c'est le cas, le **message** correspondant ("**#forward#**" dans cet exemple) est **envoyé en appelant** le composant **Microbit_Uart_Simple**.



Note : la valeur "front", écrite à l'intérieur du texte, est une valeur déclarée dans le script créé dans l'environnement de programmation MakeCode, et assignée au mouvement "avancer".

Répétez le même processus pour les quatre autres boutons. Le résultat doit être similaire à celui de la figure 20.

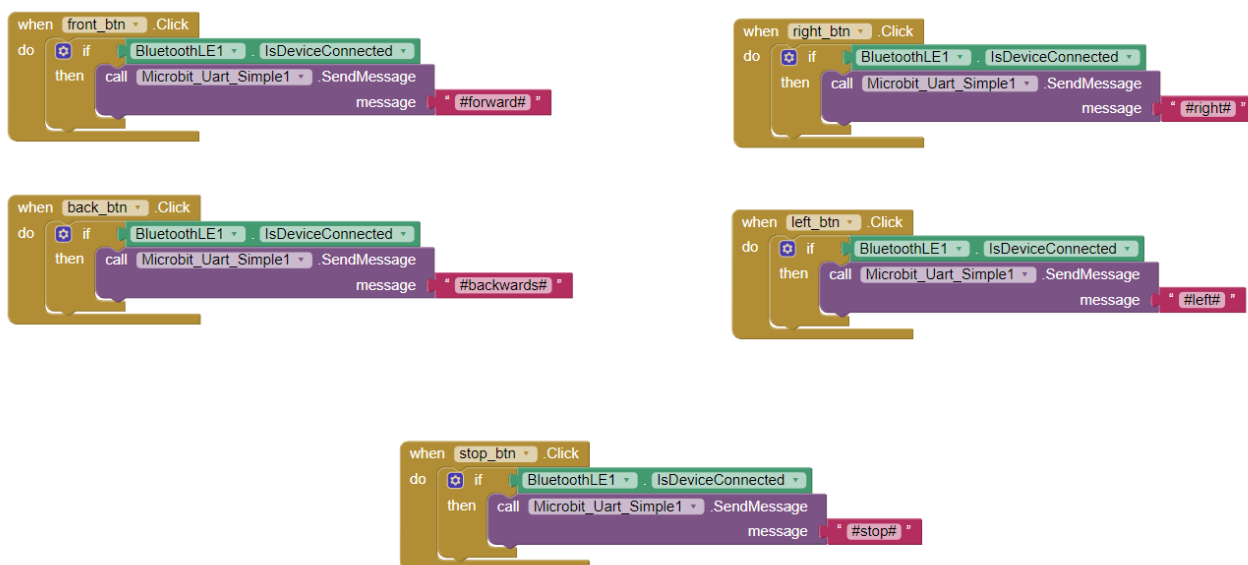


Figure 20 : Création des scripts pour tous les boutons de l'application

Lorsque vous avez terminé toutes les étapes susmentionnées, le script complet doit ressembler à celui illustré dans la figure 21.

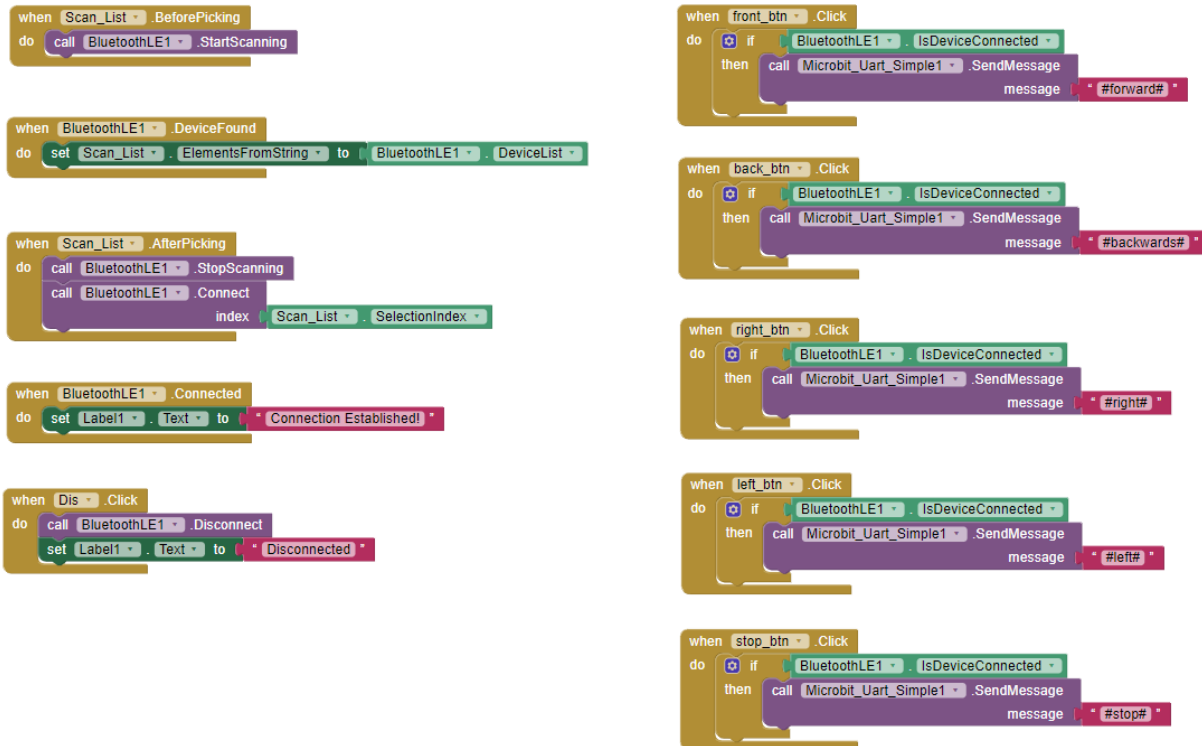


Figure 21 : Le script complet de cette activité d'échauffement

Maintenant que l'application est prête, vous pouvez la créer et la télécharger sur votre appareil intelligent.

1.5 Création de l'application

Lorsque vous avez terminé toutes les étapes susmentionnées (et que l'ensemble du code est similaire à celui illustré dans la *figure 21*), l'application est prête à être téléchargée et installée sur votre appareil intelligent.

Allez donc dans le *menu Build (1)* (*Figure 22a*) et sélectionnez "Android App (.apk)" dans le menu déroulant pour commencer le processus de production du fichier .apk. Cela peut prendre quelques minutes.

Lorsque le processus de construction est terminé, une nouvelle fenêtre s'ouvre **(2)** (*Figure 22a*). Vous pouvez soit télécharger le fichier .apk produit, soit scanner avec votre appareil intelligent le code QR intégré, via l'**application AI2 Companion du MIT** (*Figure 22b*), que vous aurez préalablement téléchargée et installée sur votre appareil intelligent.

Note 1 : MIT AI2 Companion est une application/service disponible (gratuitement) et vous pouvez la trouver dans le service "Play Store" de votre appareil intelligent. Cette application agit comme un médiateur, facilitant l'installation réussie du fichier .apk produit sur votre appareil intelligent.

Note 2 : Pendant l'installation du fichier .apk, un certain nombre de notifications concernant la sûreté/sécurité de ce fichier peuvent apparaître. Ignorez-les toutes et demandez à votre appareil de poursuivre le processus d'installation.

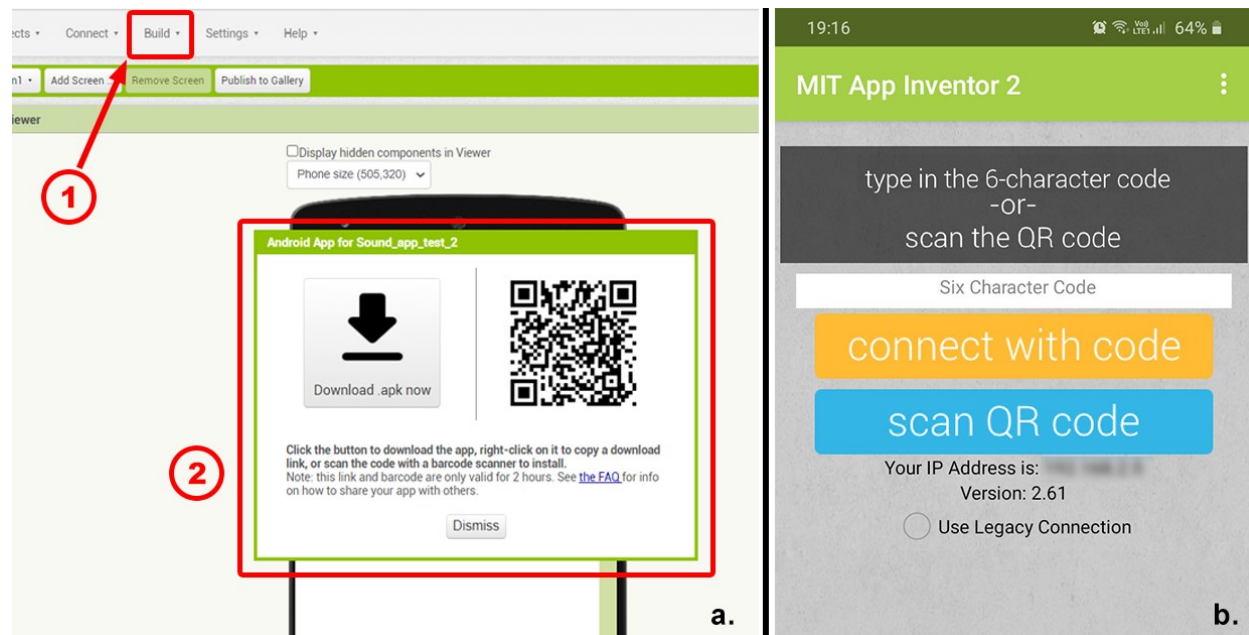


Figure 22 : a. QR Code produit à l'issue du processus de construction ; b. Capture d'écran de l'interface du MIT AI2 Companion

Après avoir installé avec succès l'application sur votre appareil intelligent, vous pouvez tester son bon fonctionnement. **N'oubliez pas** que vous devez avoir téléchargé sur votre voiture robotisée le script produit dans l'environnement de programmation Makecode.

1.6 Association de l'application à la voiture robotisée

Comme indiqué ci-dessus, pour connecter l'application à la voiture robotisée, vous devez appuyer sur le bouton "Scan" et sélectionner l'adresse Bluetooth du micro:bit dans la liste. Cependant, il est probable que le micro:bit ne figure pas dans la liste des périphériques Bluetooth disponibles. Pour résoudre ce problème, retournez au menu principal de l'application et **activez le mode "avion"** de votre appareil intelligent pendant quelques secondes. Désactivez-le ensuite et appuyez à nouveau sur le bouton Scan. L'adresse Bluetooth de Micro:bit sera alors disponible.

Remarque : Assurez-vous que la fonction "Localisation" est également activée sur votre appareil intelligent.