

Fiche de travail pour les étudiant.e.s

Equipe : .....

## C'est l'heure du brainstorming :

Que savez-vous des applications d'assistants vocaux embarqués dans les voitures ? Savez-vous si des technologies d'intelligence artificielle ont été utilisées pour optimiser ces applications ? *Recherchez des informations en ligne avec votre équipe et écrivez vos réponses ci-dessous.*

Pouvez-vous imaginer des cas concrets où les commandes vocales pour contrôler et/ou piloter une voiture autonome pourraient être utiles ou nécessaires ? *Discutez-en avec votre équipe et documentez vos réflexions ci-dessous.*

Pouvez-vous citer des avantages et des inconvénients liés à l'utilisation de commandes vocales pour contrôler et/ou piloter une voiture autonome ? Comment ces avantages et inconvénients peuvent-ils influencer sur le processus de conception ? *Discutez-en avec votre équipe et documentez vos réflexions ci-dessous.*

## Il est temps de créer l'application permettant de contrôler/piloter notre voiture robotisée par commandes vocales

### Enrichir notre application en intégrant des services d'IA

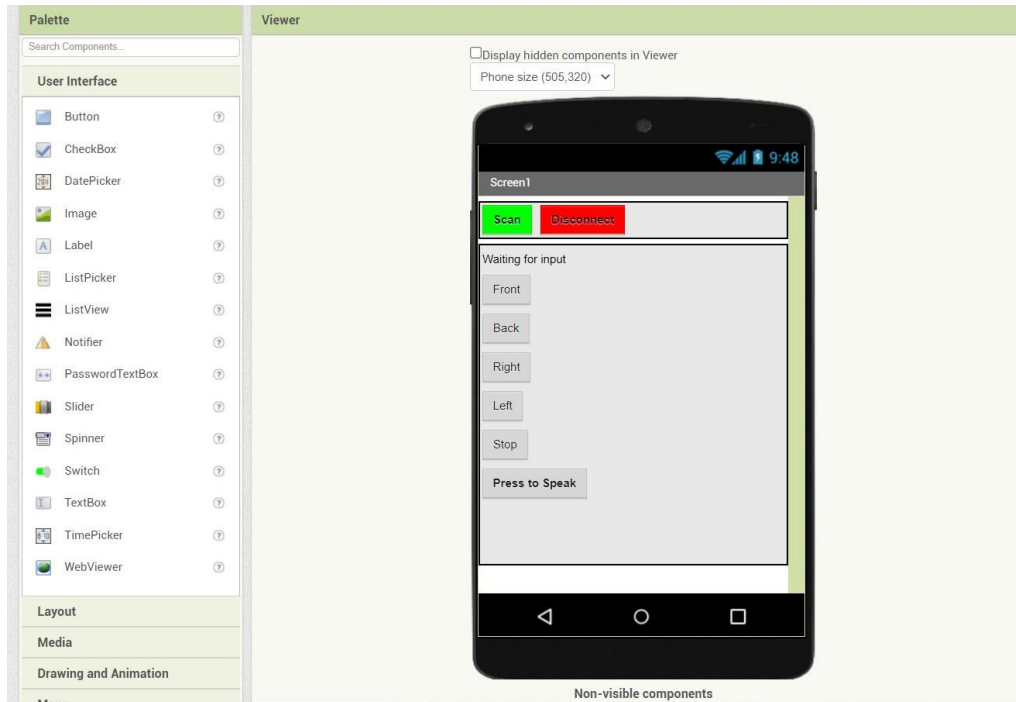
Afin de piloter notre voiture robotisée à l'aide de notre voix, nous ajouterons un bouton supplémentaire à notre application. Lorsque nous appuierons sur ce bouton, le microphone de notre appareil se déclenchera et, avec l'aide du service Google AI Speech-to-Text, nos commandes vocales seront converties en texte. Ce texte sera ensuite filtré comme suit : Si la commande vocale contient les mots "avant", "arrière", "gauche", "droite" ou "stop", notre application transmettra (par Bluetooth) le message correspondant à notre voiture robotisée, qui agira en conséquence.

Essayez de remplir le tableau suivant sur la base de la description susmentionnée. Quel message sera envoyé à notre voiture robotisée, et quel mouvement sera effectué en conséquence, lorsque chacune des commandes vocales sera donnée ? Conseil : vous pouvez également consulter le script Makecode.

<i>Si la commande vocale contient le mot :</i>	<i>Envoyez alors à la voiture robotisée le message suivant:</i>	<i>Ensuite, la voiture robotisée va:</i>
"en avant"		
"en arrière"		
"à gauche"		
"à droite"		
"stop"		

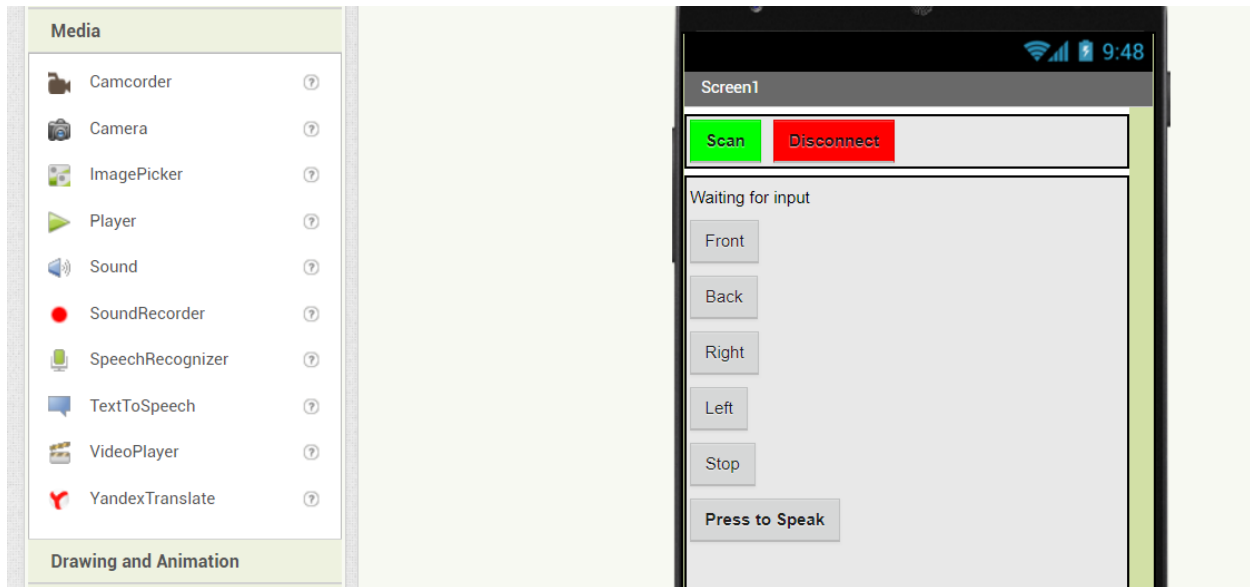
## 1) Ajout du nouveau bouton

Allez dans le menu Designer et faites glisser un bouton de l'onglet Interface Utilisateur vers la zone de conception de l'interface de l'application. Ensuite, changez le texte et le nom du bouton en quelque chose de significatif comme "Appuyer pour parler" ou "parler" par exemple. Après cette étape, l'aperçu de votre application ressemblera à l'image suivante.



## 2) Ajout du service d'IA de conversion de la parole en texte

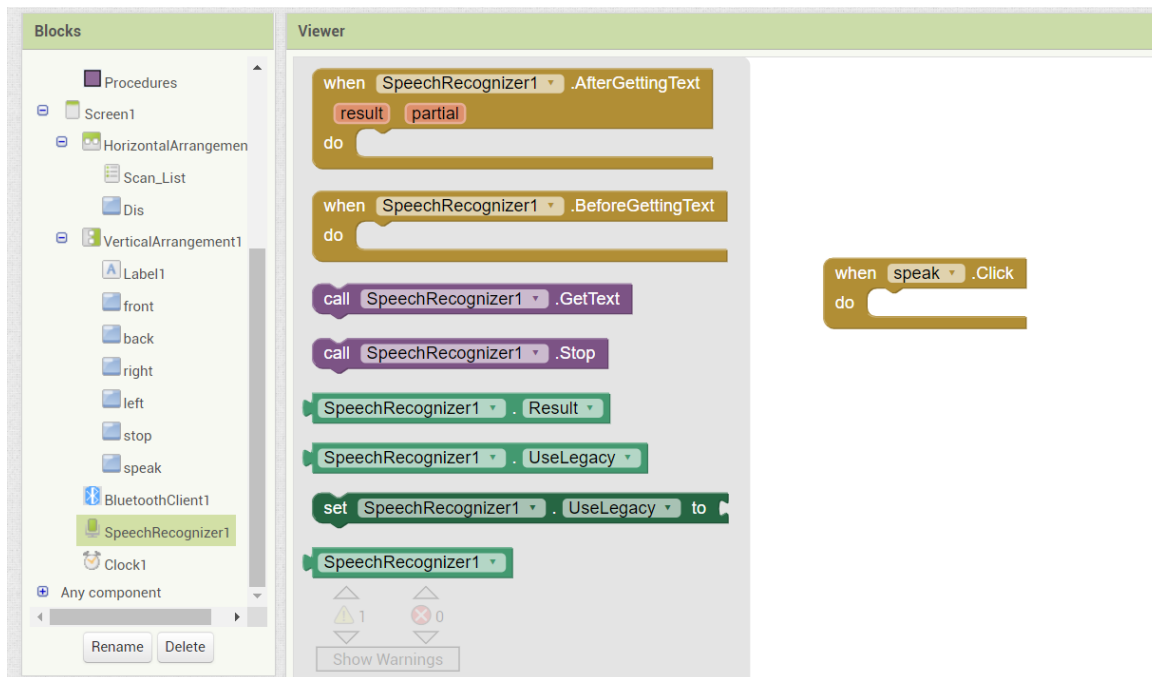
Il est maintenant temps d'ajouter un service qui permettra à notre application de **reconnaître** nos commandes vocales. Cliquez sur l'onglet Media. Parmi les services contenus, lequel pensez-vous qu'il faille utiliser ? *Saisissez votre réponse ci-dessous.*



Ensuite, faites glisser ce service vers l'écran de l'application.

### 3) Création du script pour le bouton "Appuyer pour parler".

Comme nous l'avons déjà mentionné, lorsque nous appuyons sur le bouton "parler", nos commandes vocales sont converties en texte. A partir des commandes contenues dans l'image suivante, quelle commande *SpeechRecogniser* devrions-nous placer dans le gestionnaire d'événement "*when\_speak\_click*", pour permettre à notre application d'**obtenir** un morceau de **texte** ? Discutez-en avec votre équipe et écrivez vos réflexions ci-dessous.



The screenshot shows the AI4STEM development environment. On the left is a 'Blocks' panel with a tree view of components: Procedures, Screen1, HorizontalArrangement1, Scan\_List, Dis, VerticalArrangement1, Label1, front, back, right, left, stop, speak, BluetoothClient1, SpeechRecognizer1, and Clock1. Below this is a search bar and 'Rename'/'Delete' buttons. On the right is a 'Viewer' panel showing a script for the 'when speak\_click' event. The script contains the following blocks:

```

when SpeechRecognizer1 .AfterGettingText
  result partial
do

when SpeechRecognizer1 .BeforeGettingText
do

call SpeechRecognizer1 .GetText

call SpeechRecognizer1 .Stop

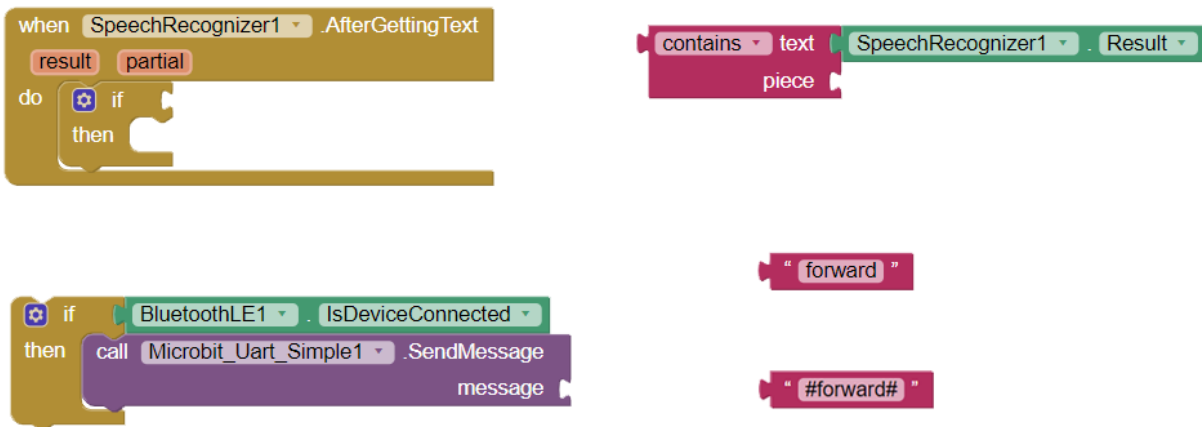
SpeechRecognizer1 . Result
SpeechRecognizer1 . UseLegacy
set SpeechRecognizer1 . UseLegacy to
SpeechRecognizer1
  
```

Below the script, there are two small icons: a yellow triangle with a '1' and a red circle with an 'X' and a '0'. A 'Show Warnings' button is at the bottom.

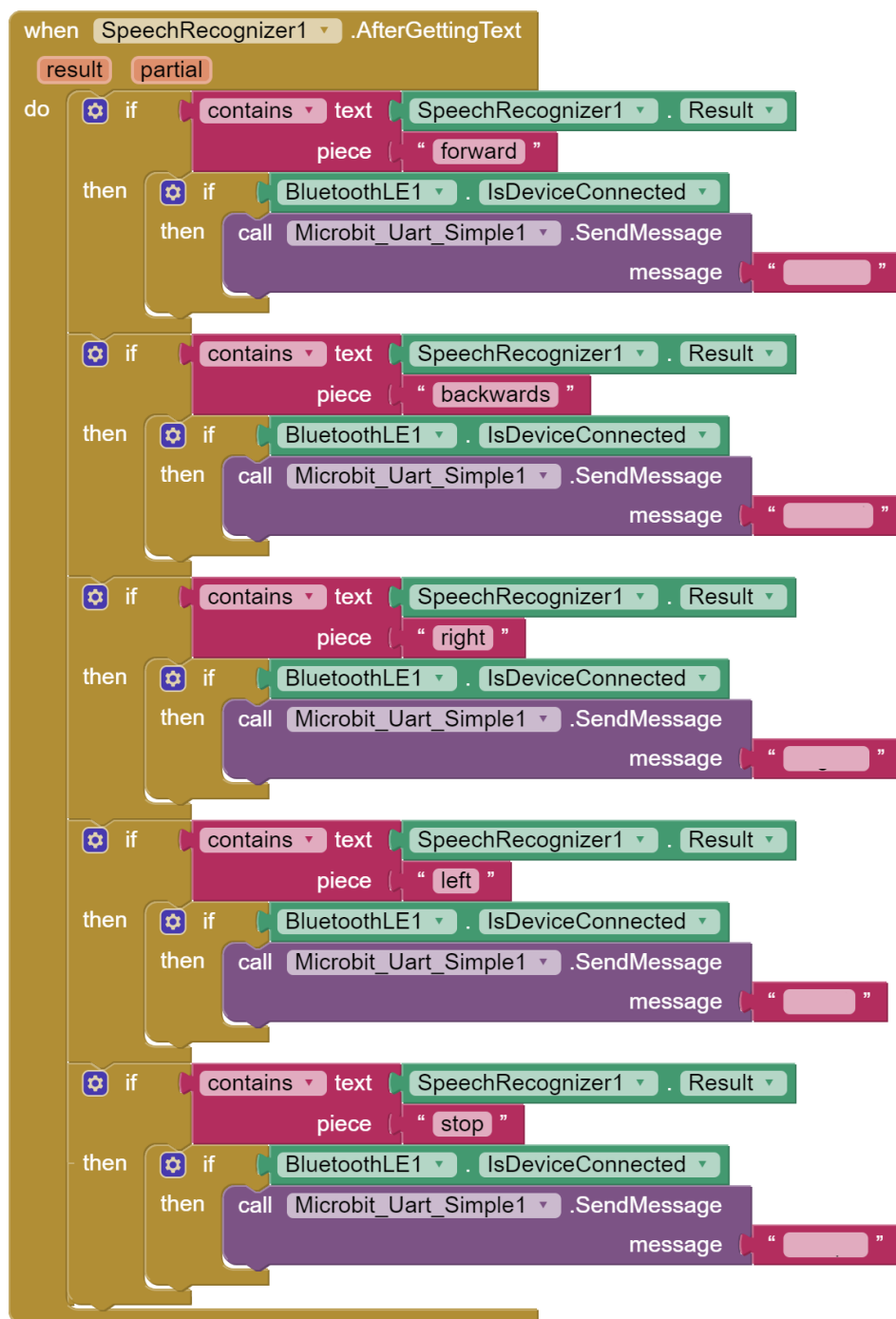
#### 4) Filtrage du texte reçu

Après avoir programmé, que se passera-t-il lorsqu'on cliquera sur le bouton (à savoir l'obtention des commandes vocales sous forme de texte) ? Il est temps de programmer notre application pour qu'elle reconnaisse des parties spécifiques du texte reçu afin de permettre à notre voiture robotisée d'effectuer les mouvements correspondants.

Le script suivant est semi-structuré. Essayez de placer les commandes dans le bon ordre pour programmer l'application afin qu'elle fasse avancer la voiture robotisée (en envoyant via le Bluetooth le message correspondant) *si* le morceau "avancer" est contenu dans le texte reconnu.



Remplissez maintenant le script précédent avec toutes les commandes nécessaires pour permettre à votre application de déplacer la voiture robotisée en conséquence, si la commande vocale correspondante est reconnue (c.-à-d. avant, arrière, droite, gauche et arrêt). Pour ce faire, essayez de remplir le script semi-structuré suivant.



### 5) Test de l'application

Il est maintenant temps de tester votre application. Allez dans le menu Build et générez votre application. Après avoir installé le fichier .apk sur votre appareil électronique, ouvrez-le et testez si votre voiture robotisée répond aux commandes vocales programmées, lorsque vous cliquez sur le bouton Appuyer pour parler.

Essayez d'utiliser différentes commandes vocales et vérifiez si votre voiture robotisée y répond ou non. Notez vos observations dans le tableau suivant (*par exemple, la commande vocale "Avancer" réussit, la commande vocale "Aller à l'avant" échoue, etc.*)

Commande vocale	Réussite	Echec



## Rendons notre application moins "ennuyeuse" en ajoutant un compteur

Jusqu'à présent, nous avons mis en œuvre le scénario suivant : Chaque fois que nous voulons donner une commande vocale, nous devons appuyer sur le bouton "Appuyer pour parler".

Cette solution est fonctionnelle, mais elle n'est peut-être pas très pratique, surtout si nous projetons ce scénario dans la vie réelle et dans des conditions de conduite réelles (c'est-à-dire au volant d'une voiture en ville).

Pour créer une solution optimale, nous allons modifier notre application comme suit :

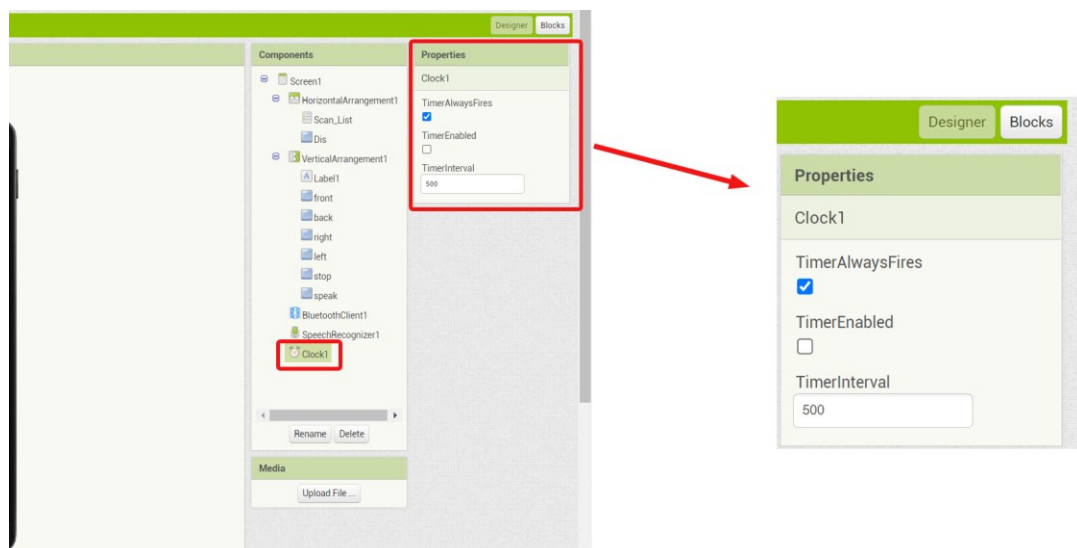
L'utilisateur ne devra appuyer qu'une seule fois sur le bouton "Appuyer pour parler", et seulement au début du processus, afin d'initialiser la reconnaissance vocale. L'application commencera alors à rechercher des commandes vocales valides. Après un intervalle de temps spécifique (par exemple, 2 secondes), l'application recherchera automatiquement une nouvelle commande vocale.

Voyons comment procéder :

### 1) Ajout d'un compteur

Allez à nouveau dans le menu Designer et faites glisser un capteur "Compteur" de l'onglet Sensors vers la zone de conception de l'application.

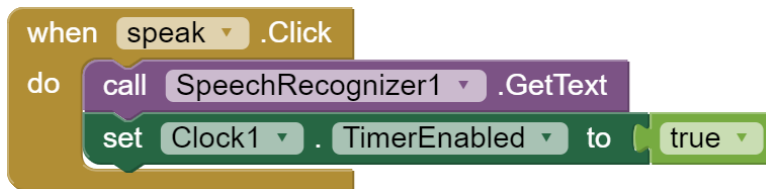
Dans les propriétés du Compteur, cochez la case TimeAlwaysFires et réglez TimeInterval sur 2000 (millisecondes).



## 2) Modification de l'application

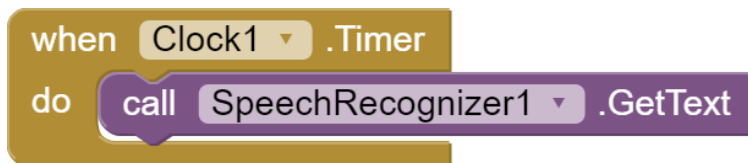
Allez maintenant dans le menu Blocks pour modifier certaines parties du script existant tout en ajoutant d'autres blocs de commandes.

Dans le gestionnaire "when speak.Click", ajoutez la commande **set Clock1 TimerEnabled** et cliquez sur une commande logique **True** sur le côté droit du bloc. Cela permettra à notre horloge de se déclencher lorsque le bouton "Press to speak" est cliqué.



Maintenant que nous avons activé le minuteur pour la première fois (après avoir appuyé sur le bouton "Parler"), l'étape suivante consiste à réactiver le SpeechRecognizer à chaque fois que le compte à rebours se termine. La durée de la minuterie est automatiquement renouvelée au cours de ce processus.

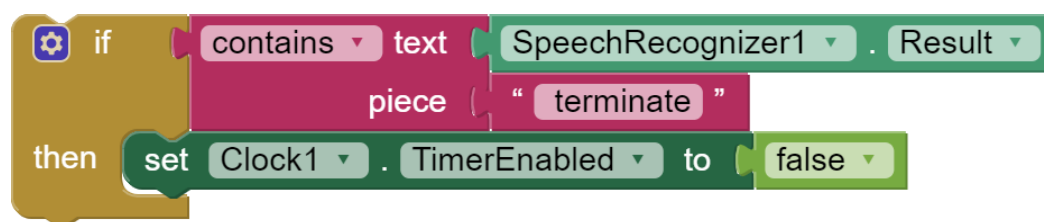
Pour ce faire, nous devons demander à notre application **de solliciter le SpeechRecognizer pour obtenir le texte** de ce qu'il entend, à chaque fois que la minuterie du Compteur se déclenche. Ceci peut être réalisé en créant le bloc de commandes suivant.



Testons l'application. Allez à nouveau dans le menu Build et générez la nouvelle version de l'application.

Fonctionne-t-elle correctement ou y a-t-il des dysfonctionnements ? Discutez-en avec votre équipe et écrivez votre réponse ci-dessous.

Ajoutez le bloc de commandes suivant au texte "When SpeechRecognizer1.AfterGetting", et testez à nouveau l'application.



Le dysfonctionnement persiste-t-il ?

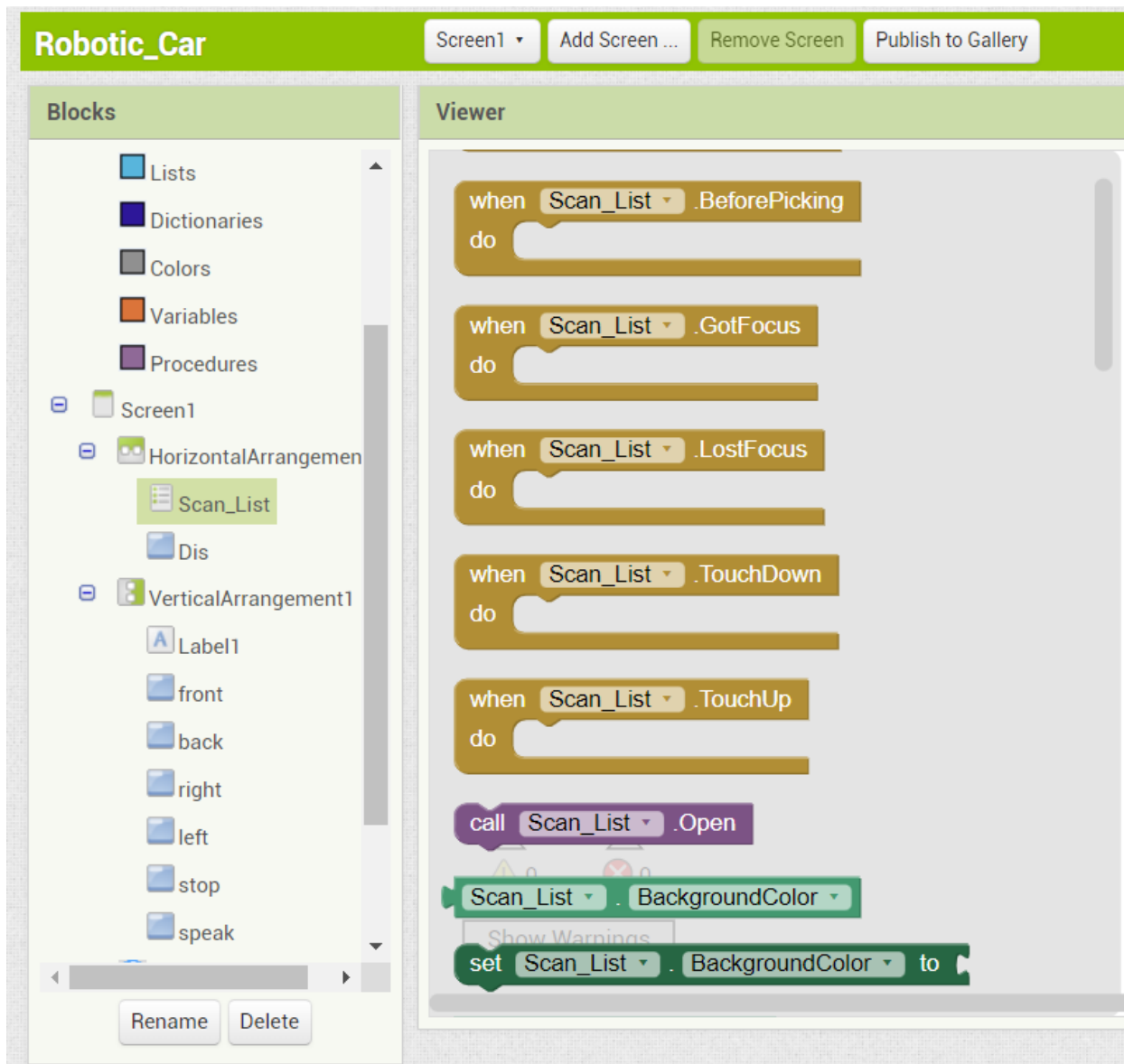
Faites d'autres expériences avec l'application pour optimiser la solution actuelle. Par exemple, essayez de modifier la durée de l'intervalle de temps et inscrivez vos observations dans le tableau ci-dessous. N'oubliez pas que le temps est compté en millisecondes (1 sec = 1000 ms).

TimerInterval	Observation

## Conseils

### Trouver les commandes :

Pour trouver certaines des commandes nécessaires, cliquez sur l'élément correspondant (c'est-à-dire Scan\_List dans l'exemple suivant) et cherchez dans le menu flottant qui apparaît.



The screenshot shows the Robotic\_Car interface. The top bar includes a title "Robotic\_Car" and buttons for "Screen1", "Add Screen ...", "Remove Screen", and "Publish to Gallery". The interface is divided into two main panels: "Blocks" on the left and "Viewer" on the right.

**Blocks Panel:** This panel contains a list of categories and objects. The categories are Lists, Dictionaries, Colors, Variables, and Procedures. Under the "Screen1" category, there is a "HorizontalArrangement" object, which contains a "Scan\_List" object. Below "HorizontalArrangement" is a "VerticalArrangement1" object, which contains several sub-objects: "Label1", "front", "back", "right", "left", "stop", and "speak".

**Viewer Panel:** This panel displays a sequence of commands for the "Scan\_List" object. The commands are:

- when Scan\_List .BeforePicking do
- when Scan\_List .GotFocus do
- when Scan\_List .LostFocus do
- when Scan\_List .TouchDown do
- when Scan\_List .TouchUp do
- call Scan\_List .Open
- Scan\_List .BackgroundColor
- set Scan\_List .BackgroundColor to

Certaines commandes contiennent plus d'une option. L'image suivante présente une telle commande.

