

SYNTHÈSE DE TECHNOLOGIE

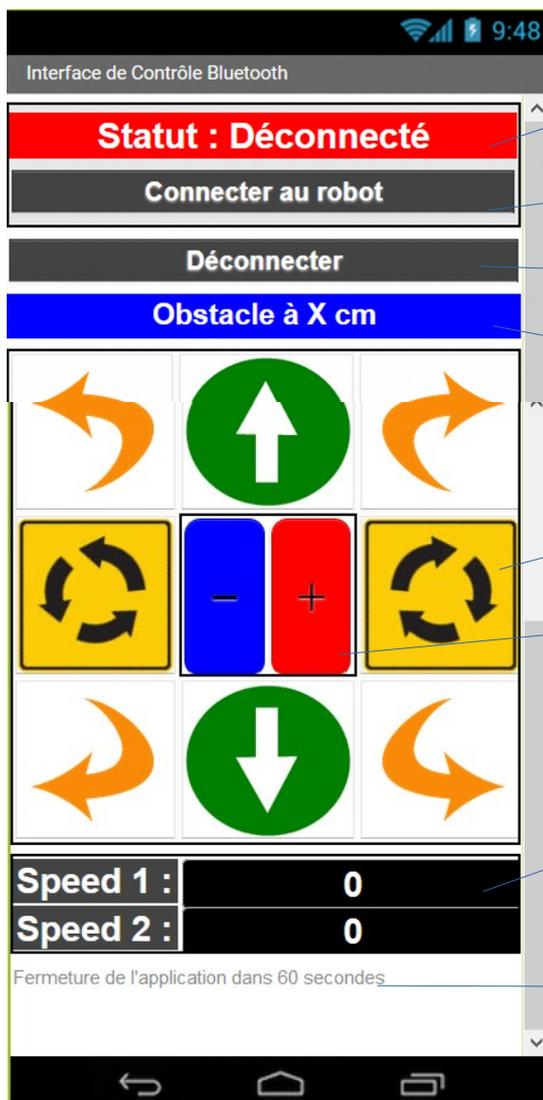
- +++Sommaire
- ++Présentation
- ++App Inventor
 - +L'interface
 - +Les Blocs et le fonctionnement
- ++Mblock
 - +Les Commandes du Robot
 - +La détection d'Obstacles
- ++Conclusion

PRÉSENTATION

+J'étais le chef de projet du groupe et je travaillais sur App Inventor ainsi que sur Mblock ; c'est pourquoi ma synthèse sera rédigé en trois autres parties (autres que la présentation) :

- La première partie sera dédiée à App Inventor,
- La seconde à Mblock,
- Une conclusion qui rassemblera tout ce que j'ai appris au long de ce projet fort intéressant.

1°/ L'Interface



« Label » affichant le statut de connexion au robot, devient vert s'il est connecté

Bouton de connexion, il n'apparaît que lorsque le robot est déconnecté

Bouton de déconnexion, il n'apparaît uniquement quand le robot est connecté.

« Label » affichant la distance d'un obstacle par rapport au robot.

« Table arrangement » où se trouvent des boutons de contrôle du véhicule

Autre « Table arrangement » dans le premier « Table arrangement » où se trouvent des boutons de contrôle de la vitesse, il remplace le bouton « Stop ».

« Table arrangement » où sont affichés la vitesse principale (Speed1) et la vitesse secondaire (Speed2) pour les avancées et reculs en tournant à droite ou à gauche.

« Label » affichant le temps restant avant la fermeture de l'application, tant que le robot n'est pas connecté. Visible uniquement si le robot n'est pas connecté

+Sont aussi présent des objets invisibles sur l'interface :

-3 « clocks »

-1 « BluetoothClient »

+Mon interface comprenait initialement un joystick et un deuxième moyen de changer la vitesse en plus, mais un problème de transfert de données à mBlock m'a contraint à les retirer.

2°/ Les Blocs et le Fonctionnement

```

when Screen1.Initialize
do
  set Btn_speed_plus.Enabled to false
  set Btn_speed_minus.Enabled to false
  set Forward_and_Left.Enabled to false
  set Forward_and_Right.Enabled to false
  set Reverse_and_Left.Enabled to false
  set Reverse_and_Right.Enabled to false
  set btnForward.Enabled to false
  set btnReverse.Enabled to false
  set btnRight.Enabled to false
  set btnLeft.Enabled to false
  set Disconnectbtn.Visible to false
  set Disconnectbtn.Enabled to false
  set Clock1.TimerEnabled to false
  set Clock2.TimerEnabled to false
  set Speed1.Enabled to false
  set Speed2.Enabled to false
  set BluetoothClient1.Secure to true
  
```

Voici les blocs de l'initialisation de l'écran. Afin d'éviter toute erreur bluetooth, la plupart des boutons sont désactivés.

Ces blocs concernent la connexion au bluetooth, ils changent le texte et la couleur du label de statut, connecte au bluetooth grâce à un « list picker » et un « bluetoothclient » ; ils activent aussi les boutons par le biais de la procédure « Active_Interface ».

```

when listPairedBluetoothDevices.BeforePicking
do
  set listPairedBluetoothDevices.Elements to BluetoothClient1.AddressesAndNames

when listPairedBluetoothDevices.TouchDown
do
  set global Close to 60

when listPairedBluetoothDevices.AfterPicking
do
  evaluate but ignore result call BluetoothClient1.Connect
  address listPairedBluetoothDevices.Selection

set global Close to 60
if BluetoothClient1.IsConnected
then
  set lblConnectionStatus.Text to Statut: Connecté
  set lblConnectionStatus.TextColor to
  set lblConnectionStatus.BackgroundColor to
  set listPairedBluetoothDevices.Enabled to false
  set listPairedBluetoothDevices.Visible to false
  set Clock1.TimerEnabled to true
  set Clock3.TimerEnabled to false
  set Close_Label.Visible to false
  call Active_Interface
  
```

Le modèle d'un bouton directionnel (ici pour communiquer au robot d'avancer). Lorsque que le bouton est touché, un texte est envoyé au robot pour qu'il exécute une action définie dans le programme mBlock :
 var=1\n → avancer
 var=2\n → reculer
 var=3\n → tourner à gauche
 var=4\n → tourner à droite
 var=6\n → avancer en tournant à gauche
 Etc...

Quand un bouton directionnel est relâché, le texte envoyé au robot est « var=5\n », ce qui lui indique de se stopper, ainsi le robot roule seulement lorsqu'on appuie sur un bouton directionnel.

```

when Btn_speed_plus.TouchDown
do
  call BluetoothClient1.SendText
  text var=10\n
  set global Speed1 to get global Speed1 + 50
  set global Speed2 to get global Speed2 + 50
  set global Time to 0
  set Speed1.Text to get global Speed1
  set Speed2.Text to get global Speed2
  
```

Un bouton de changement de vitesse (+) qui modifie le texte des « text_box » « Speed1 » et « Speed2 » par le biais de deux variables des mêmes noms et envoie le texte « var=10\n »

Le bouton de changement de vitesse (-) suis le même principe mais en soustrayant et non en additionnant 50 aux variables et envoie le texte « var=11\n »

```

initialize global Speed2 to 80
initialize global Speed1 to 155
  
```

Les variables Speed1 et Speed2

```

to Active_Interface
do
  set TableArrangement1.Visible to true
  set Modifier_Vitesse2.Visible to true
  set Disconnectbtn.Visible to true
  set Disconnectbtn.Enabled to true
  set btnForward.Enabled to true
  set btnReverse.Enabled to true
  set btnLeft.Enabled to true
  set btnRight.Enabled to true
  set Reverse_and_Left.Enabled to true
  set Reverse_and_Right.Enabled to true
  set Forward_and_Left.Enabled to true
  set Forward_and_Right.Enabled to true
  set Btn_speed_plus.Enabled to true
  set Btn_speed_minus.Enabled to true
  
```

La définition de la procédure « Active_Interface »

```

initialize global Close to 60
initialize global Time to 0

when Clock1.Timer
do
  set global Time to get global Time + 1
  if get global Time >= 30
  then
    call deconnecter

when Clock3.Timer
do
  set global Close to get global Close - 1
  set Close_Label.Text to join | Fermeture de l'application dans |
  | get global Close | secondes.
  if get global Close = 0
  then
    close application
  
```

Ces blocs sont liés aux horloges et contrôlent les temps de fermeture de l'app (Clock3, 60 sec.) et de déconnexion bluetooth (Clock1, 30 sec.).

```

when Disconnectbtn.Click
do
  call deconnecter
  
```

Le bloc du bouton « Déconnecter » qui fait appelle à la procédure « deconnecter » pour déconnecter le bluetooth

```

do
  call BluetoothClient1 . Disconnect
  set Distance_Sonar . Visible to false
  set TableArrangement1 . Visible to false
  set Modifier_Vitesse2 . Visible to false
  set Speed1 . BackgroundColor to 
  set Speed2 . BackgroundColor to 
  set Speed1 . Text to 0
  set Speed2 . Text to 0
  set IstPairedBluetoothDevices . Enabled to false
  set IstPairedBluetoothDevices . Visible to false
  set lbiConnectionStatus . Text to Statut : Déconn
  set lbiConnectionStatus . TextColor to 
  set lbiConnectionStatus . BackgroundColor to 
  set Forward_and_Left . Enabled to false
  set Forward_and_Right . Enabled to false
  set Reverse_and_Left . Enabled to false
  set Reverse_and_Right . Enabled to false
  set btnForward . Enabled to false
  set btnReverse . Enabled to false
  set btnRight . Enabled to false
  set btnLeft . Enabled to false
  set IstPairedBluetoothDevices . Visible to true
  set IstPairedBluetoothDevices . Enabled to true
  set Clock1 . TimerEnabled to false
  set Clock2 . TimerEnabled to false
  set Clock3 . TimerEnabled to true
  set Disconnectbtn . Visible to false
  set Disconnectbtn . Enabled to false
  set Close_Label . Visible to true
  set global Close to 60
  set global Time to 0

```

La définition de la procédure « déconnecter ». Elle consiste en déconnectant la plupart des boutons, changer le texte et la couleur du « label » de statut, modifier le text des « text_box » en « 0 » et faire réapparaître le bouton de connexion.

```

when Clock2 . Timer
do
  if BluetoothClient1 . IsConnected and call BluetoothClient1 . BytesAvailableToReceive > 0
  then
    set Distance_Sonar . Text to join
    call BluetoothClient1 . ReceiveText
    numberOrBytes call BluetoothClient1 . BytesAvailableToReceive
    cm
  if get global Speed1 <= 154 and BluetoothClient1 . IsConnected
  then
    set Speed1 . BackgroundColor to 
  else if get global Speed1 >= 200 and BluetoothClient1 . IsConnected
  then
    set Speed1 . BackgroundColor to 
  else if get global Speed1 > 154 and get global Speed1 <= 200 and BluetoothClient1 . IsConnected
  then
    set Speed1 . BackgroundColor to 
  if get global Speed2 <= 79 and BluetoothClient1 . IsConnected
  then
    set Speed2 . BackgroundColor to 
  else if get global Speed2 >= 125 and BluetoothClient1 . IsConnected
  then
    set Speed2 . BackgroundColor to 
  else if get global Speed2 > 79 and get global Speed2 <= 125 and BluetoothClient1 . IsConnected
  then
    set Speed2 . BackgroundColor to 

```

Un grand ensemble de blocs, qui permet de récupérer la distance d'un obstacle au robot et de l'écrire sur le « label » « Distance_Sonar ». Il permet aussi de changer la couleur de fond des « text_box » en fonction de la vitesse (élevé = rouge, moyen = vert, faible = bleu foncé).

Tout ceci se produit lorsque que l'horloge « Clock2 » « Timer » à intervalle 50.

MBLOCK

1°/ Le Contrôle du Robot

+Cette première section de la rubrique « mBlock » concerne la réception des données envoyées par App Inventor, il s'agit de la plus grande partie du programme.

```

si une donnée est disponible? alors
  mettre rx à lire la commande var
  si rx > 0 alors
    écrire la ligne distance mesurée par le capteur ultrasons du D0rt4
    si rx = 1 alors
      si juli < 40 ou juli = 40 alors
        si Vara = 0 alors
          activer le moteur M1 à la puissance Speed1
          activer le moteur M2 à la puissance Speed1
        sinon
          activer le moteur M1 à la puissance 0
          activer le moteur M2 à la puissance 0
        sinon
          activer le moteur M1 à la puissance Speed1
          activer le moteur M2 à la puissance Speed1
      si rx = 2 alors
        activer le moteur M1 à la puissance -1 * Speed1
        activer le moteur M2 à la puissance -1 * Speed1
      si rx = 3 alors
        activer le moteur M1 à la puissance -1 * Speed1
        activer le moteur M2 à la puissance Speed1
      si rx = 4 alors
        activer le moteur M1 à la puissance Speed1
        activer le moteur M2 à la puissance -1 * Speed1
      si rx = 5 alors
        activer le moteur M1 à la puissance 0
        activer le moteur M2 à la puissance 0

```

Ce bloc « si une donnée est disponible » permet que les blocs suivants soient traités uniquement si le robot est connecté.

Ce bloc indique de lire la commande « var » envoyée par App Inventor comme « rx ».

Ces blocs indique au robot quoi en fonction des données reçu: par exemple, si rx=1 alors activer les 2 moteurs à la vitesse « Speed1 »; si rx=7 alors activer le moteur1 à la vitesse « Speed1 » et le moteur2 à la vitesse « Speed2 ».

Les variables « Speed1 » et « Speed2 » Sont initialement réglées à 155 (Speed1) et 80 (Speed2). « Speed1 » est la vitesse utilisée pour les avancées et reculs droits, pour les avancées et reculs en tournant, un des moteur est réglé à la vitesse « Speed2 » et l'autre « Speed1 » afin d'obtenir un décalage de vitesse entre les moteurs et donc obtenir un virage à gauche ou à droite selon quel moteur est réglé à la vitesse « Speed2 ». Ces variables sont utilisées pour pouvoir modifier la vitesse (voir plus bas).

On peut voir que chaque groupe de blocs concernant une avancée contient une condition: le robot ne peut avancer que si la variable « juli » est à plus de 40 ou que la variable « Vara » est à 0, la prochaine section concernera cela.

```

si rx = 6 alors
  si juli < 40 ou juli = 40 alors
    si Vara = 0 alors
      activer le moteur M1 à la puissance Speed2
      activer le moteur M2 à la puissance Speed1
    sinon
      activer le moteur M1 à la puissance 0
      activer le moteur M2 à la puissance 0
  sinon
    activer le moteur M1 à la puissance Speed2
    activer le moteur M2 à la puissance Speed1
  si rx = 7 alors
    si juli < 40 ou juli = 40 alors
      si Vara = 0 alors
        activer le moteur M1 à la puissance Speed1
        activer le moteur M2 à la puissance Speed2
      sinon
        activer le moteur M1 à la puissance 0
        activer le moteur M2 à la puissance 0
    sinon
      activer le moteur M1 à la puissance Speed1
      activer le moteur M2 à la puissance Speed2
  si rx = 8 alors
    activer le moteur M1 à la puissance -1 * Speed2
    activer le moteur M2 à la puissance -1 * Speed1
  si rx = 9 alors
    activer le moteur M1 à la puissance -1 * Speed1
    activer le moteur M2 à la puissance -1 * Speed2

```

```

sinon
si rx = 9 alors
activer le moteur M1 à la puissance -1 * Speed1
activer le moteur M2 à la puissance -1 * Speed2
sinon
si rx = 10 alors
si Speed1 = 255 alors
ajouter à Speed1 0
ajouter à Speed2 0
sinon
ajouter à Speed1 50
ajouter à Speed2 50
sinon
si rx = 11 alors
si Speed2 = 10 alors
ajouter à Speed1 0
ajouter à Speed2 0
sinon
ajouter à Speed1 -50
ajouter à Speed2 -50

```

Ces groupes de blocs concernent un changement de la vitesse :
 -Si rx = 10, les variables « Speed1 » et « Speed2 » se voient ajouter 50
 -Si rx = 11, les variables « Speed1 » et « Speed2 » se voient ajouter -50 (soit soustraire 50)

On peut voir que ces blocs contiennent une condition : pas d'augmentation si « Speed1 » = 255 ou de réduction si « Speed2 » = 10.
 Ce sont des blocages que j'ai intégré afin d'éviter des bugs de vitesse du robot en cas de vitesse trop élevée ou trop réduite. Ainsi, comme « Speed1 » et « Speed2 » ont toujours une différence de 75, on a :
 $85 < Speed1 < 255$
 $10 < Speed2 < 180$

2°/ La Détection d'obstacle

+Cette seconde section est centrée sur la détection d'obstacle, la variable juli ainsi que la communication de la distance d'obstacle à App Inventor.

```

Orion - générer le code
mettre Vara à 0
mettre Vara2 à 0
mettre Speed1 à 155
mettre Speed2 à 80
répéter indéfiniment
mettre juli à distance mesurée par le capteur ultrasons du Port4
si juli = 0 alors
répéter jusqu'à juli > 0
mettre juli à distance mesurée par le capteur ultrasons du Port4
si juli > 40 ou juli = 40 alors
mettre Vara à 1
si juli > 50 ou juli = 50 alors
mettre Vara à 0
si juli < 20 ou juli = 20 alors
activer le moteur M1 à la puissance -100
activer le moteur M2 à la puissance -100
attendre jusqu'à distance mesurée par le capteur ultrasons du Port4 > 40
activer le moteur M1 à la puissance 0
activer le moteur M2 à la puissance 0

```

Voici les blocs qui déterminent la variable « juli ».
 La valeur de « juli » correspond à la mesure d'obstacle en cm par le capteur d'ultrasons (il détecte l'obstacle le plus proche). Le deuxième groupe de blocs (si juli=0 etc...) a été ajouté avec l'aide de M.Maréchal à cause d'un bug dans la mesure d'obstacle, ce qui faisait reculer la voiture indéfiniment (cela est lié au groupe de blocs « si juli < 20 ou juli = 20 ... », à voir plus bas).

Ces autres blocs concernent la variable Vara, servant à l'hyperstésis des 50cm de distance d'un obstacle après recul depuis 40 cm ou moins : dans cette situation, la viture peut de nouveau avancer (voir plus haut), fonction étant bloquée lorsque Vara était égale à 1 (soit juli=40 donc ostacle à 40cm).

Ce groupe de blocs fait en sorte de faire reculer la voiture jusqu'à 40cm d'un obstacle, si celui-ci se trouve à 20cm ou moins.
 (modifié suite à quelques bugs avec l'aide de M.Maréchal)

Ce groupe de blocs se trouve plus bas dans le programme, dans le bloc « si une donnée est disponible ». Lorsque qu'une donnée d'un quelconque bouton d'App Inventor est reçue (traduit par si rx > 0), la distance d'un obstacle au robot est envoyé sur l'interface du programme App Inventor (écrire la ligne...), dans le label « Distance_Sonar » (voir rubrique App Inventor, section 2).
 (Le bloc « écrire la ligne » a été intégré au programme avec, encore une fois, l'aide de M.Maréchal Merci :))

```

si rx > 0 alors
écrire la ligne distance mesurée par le capteur ultrasons du Port4

```

CONCLUSION

+Je retiens de ce projet qu'il m'a apporté beaucoup de connaissances supplémentaires et qu'il m'a fort intéressé : j'ai passé de nombreuses heures à travailler chez moi pour peaufiner mes programmes sur App Inventor et mBlock et j'ai tenté d'ajouter des fonctionnalités supplémentaires, telles que la possibilité de choisir entre le joystick et les boutons et des boîtes de texte afin de pouvoir changer la vitesse en y inscrivant la vitesse désirée pour Speed1 et Speed2. Malheureusement, j'ai découvert en salle de Technologie vendredi dernier que ces ajouts ne fonctionnaient pas, car le robot ne pouvait traiter qu'une seule variable envoyée par App Inventor ! Malgré tout, après avoir réglé ces problèmes en enlevant ces variables, mes programmes étaient fonctionnels et ne présentaient plus de bugs (excepté la communication de la distance d'un obstacle, que j'ai réglé plus tard). Je suis fier de ces programmes, ayant travaillé seul sur mBlock (car Antonis ne faisait rien) mais par contre et heureusement, Marc-Olivier a contribué au programme sur App Inventor.

+Chaque fonction des programmes à son utilité, Sur mBlock, la détection d'obstacle permet d'éviter des incidents et assure la sécurité du robot (il ne peut pas foncer à toute vitesse dans un mur). Sur App Inventor, l'ergonomie et les différentes fonctionnalités de l'interface de contrôle du robot permet de mieux maîtriser ce dernier et simplifie les manœuvres (au départ, on devait appuyer sur un bouton stop pour arrêter le robot) et pour déconnecter le robot, sortir de l'app était nécessaire, l'ajout d'un bouton déconnecter régla ce gros inconvénient. De plus, aussi bien sur App Inventor que sur mBlock, les 4 boutons de mouvement ajoutés permirent au robot d'avancer ou de reculer en tournant à gauche ou à droite.

+Voici donc la fin de cette synthèse un peu trop longue, synthèse d'un projet hautement enrichissant et absolument inoubliable. À présent j'ai hâte de commencer à faire du robot un suiveur de ligne.

*Léo Larigauderie,
Elève de 3[°]A*