Robot Mbot

Ce robot a plusieurs avantages: ludique, robuste, évolutif, disposant d'un gros écosystème, abordable financièrement, programmable simplement (mblock 5) ou professionnellement (arduino), pouvant utiliser la grande gamme de composants arduino Grove …
Il permet une transition aisée avec l'arduino pur qui est « la » référence en IEOC, mais qui est plus technique et fragile.
Ce robot a souvent été utilisé au collège et programmé avec mblock. Pour certains cela risque d'être trop simple. Pour d'autres cela sera un bon point d'entrée vers ce thème. C'est aussi un bon outil, évolutif, pour des mini-projets pour peu qu'on demande à l'élève de ne pas en rester à l'aspect «jolis blocs de programmation », mais qu'on fasse le lien avec le code arduino. La continuité de cette séquence pédagogique est le codage réel sur arduino. Mais c'est du langage C, pas du python …
Dans l'optique de la formation de masse: beaucoup d'élèves, sans trop de matériel, je propose d'utiliser un simulateur mbot de la société IRAI (https://www.iraifrance.com/mbot-simulator-simulateur-mbot). Ce simulateur, en conjonction avec l'achat de quelques mbot réels, peut être une bonne solution pour traiter le thème « Info Embarquée et objets connectés » avec beaucoup d'élèves, sans trop de moyen technique et en peu de temps.

L'inconvénient de mBot par rapport à une solution comme micro:bit, c'est le fait que la programmation n'est pas encore « python native ».
On peut espérer que cela va changer vu la réactivité de la communauté qui le gère.

Contenus	Capacités attendues
Systèmes informatiques embarqués	Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs, l'IHM et les actions des actionneurs dans des systèmes courants.
Interface homme-machine (IHM)	Réaliser une IHM simple d'un objet connecté.
Commande d'un actionneur, acquisition des données d'un capteur	Écrire des programmes simples d'acquisition de données ou de commande d'un actionneur.
Exemples d'activités	
Réaliser une IHM pouvant piloter deux ou trois actionneurs et acquérir les données d'un ou deux capteurs. Gérer des entrées/sorties à travers les ports utilisés par le système.	

Activités

- Analyse fonctionnelle et structurelle du robot => identifier les capteurs et les actionneurs
- Programmation du robot: interface Mblock et identification avec code arduino
- Programmation d'un avatar: interface Mblock et identification avec code python
- Programmation d'un IHM avec AppInventor afin de commander le robot

Présentation

Il s'agit d'un robot Mbot du commerce.
Il est doté d'une carte électronique programmable.Celle-ci peut être programmé grâce à un ordinateur soit par fil (port USB), soit en Bluetooth
On utilisera le logiciel « Mblock » pour créer nos programmes. Le langage graphique utilisé est du type « scratch ».

Le robot mBot interagit avec son environnement en fonction du programme qu'on lui implante.

Pour cela, il est capable de collecter des informations grâce **à ses capteurs** et de réaliser des actions grâce **à ses actionneurs**.

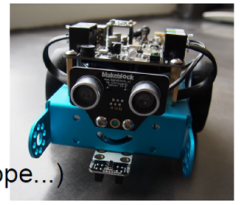
Actions et actionneurs :

- le robot vendu de base, est capable de **se déplacer** : il est équipé de **deux moteurs** indépendants reliés chacun à une roue (qui devient donc **une roue motrice**).
- il peut **émettre des sons** grâce à un **buzzer**.
- il peut **émettre de la lumière** grâce à **2 DEL 3 couleurs** (RGB) dont la couleur est paramétrable.
-d'autres actionneurs peuvent être branchés **en option** (afficheur 128 led, motoréducteur, blocs 4 led, afficheur 7 segments...)

Boutons et capteurs :

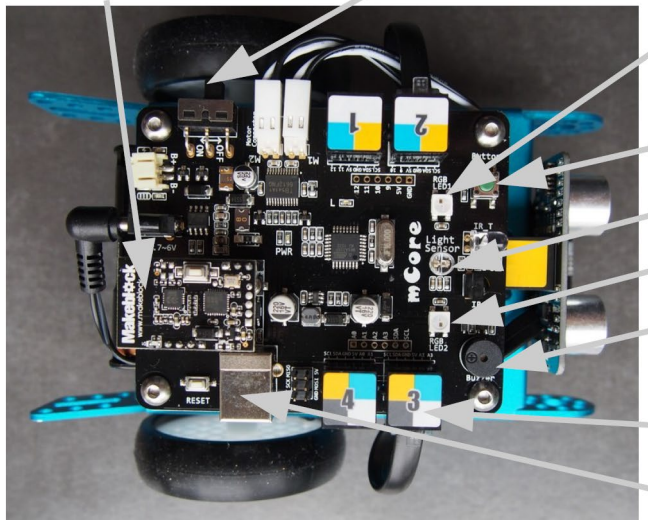
Pour interagir avec son environnement et y recueillir des informations, on retrouve sur le robot :

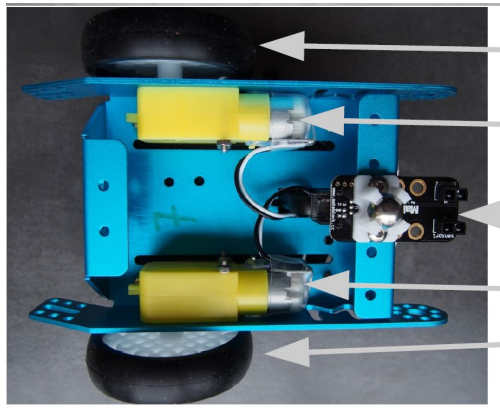
- un **module Wifi** qui permet de recevoir les ordres émis par l'ordinateur.
- un **capteur de luminosité** qui le renseigne sur la luminosité ambiante.
- un **module à ultrasons** qui lui permet de « voir » les obstacles à l'avant et d'en connaître la distance.
- un **module de suivi de ligne au sol** à infrarouge.
- un **bouton** paramétrable.
- un **bouton de mise sous tension**.
-d'autres capteurs peuvent être branchés **en option** (humidité, flamme, fumée,gyroscope...)



carte Bluetooth

Bouton de mise sous tension





Activité 1) Analyse fonctionnelle et structurelle du robot

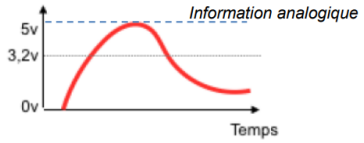
Types de signaux

Nature d'une information : logique ou analogique

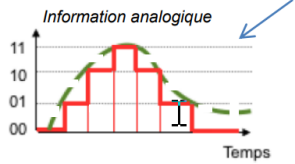
Nombres de valeurs possibles du signal qui porte l'information	Nature de l'information	Exemples
2	**LOGIQUE** Chaque valeur correspond à un état du système. Les deux états possibles sont contraire, opposés au sens logique.	Portail, interrupteur ouvert/fermé Jour/nuit Passage, quelqu'un / pas de passage, personne
Plusieurs valeurs distinctes (>2), jusqu'à une infinité	**ANALOGIQUE** Chaque valeur correspond à une valeur d'une grandeur physique à un instant donné	Evolution au cours du temps : - de la valeur de la température en °C - de la tension électrique dans un circuit en V -de l'éclairement en lux

Nature d'un signal : analogique ou numérique

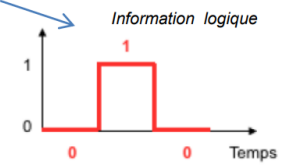
SIGNAL ANALOGIQUE	SIGNAL NUMERIQUE
Le signal peut avoir **toutes les valeurs possibles sur un intervalle de temps donné**. A chaque instant est associée une valeur : on dit aussi que c'est un signal **continu.**	Le signal, au cours du temps, ne peut avoir qu'un **certain nombre fini de valeurs**. On dit aussi que le signal est **« discret »** : on passe d'une valeur à l'autre par saut, par pallier. Les valeurs sont codées en binaire sur un certain nombre de bits (0/1). Si le codage est réalisé sur un bit, le signal ne peut avoir que 2^1= 2, 0 valeurs (0 ou 1) ; l'information dans ce cas est de type **logique**. Codé sur 2 bits , le signal peut prendre 2^2=4 valeurs, etc. Si on code sur plus d'un bit, l'information transportée est **analogique**.



Exemple : le signal analogique prend toutes les valeurs possibles entre 0 et 5V au cours du temps



Exemple : le signal numérique ne peut avoir que 4 valeurs codées (>2) en binaire au cours du temps 00,01,10,11



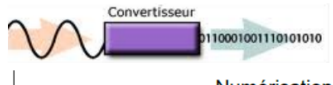
Ici, le signal numérique ne peut avoir que 2 valeurs binaires au cours du temps : 0 ou 1

Traitement numérique et numérisation

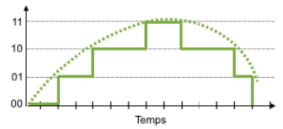
Un signal analogique doit être converti en signal numérique pour que l'étape « traiter » de la chaîne d'information puisse avoir lieu. En effet, le traitement du signal reposant sur une programmation informatique ne peut avoir lieu que **sur des 0 et des 1 (code binaire),** puisque c'est le seul « langage » que comprend un microprocesseur ou un microcontrôleur, ou toute autre interface de traitement. **On parle de traitement numérique.** Cette conversion s'appelle **la numérisation**, qui s'effectue avec un dispositif appelé **Convertisseur Analogique/ Numérique (CAN).**

Exemple un capteur de température :

NB : Parfois, le CAN est juste utilisé pour que le capteur ait un affichage numérique



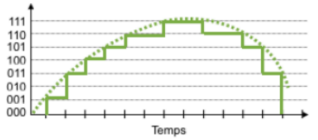
Numérisation sur 2 bits



Soit 4 valeurs possibles : de 0 à 3

Puissance de 2	2^1	2^0
Décimal	2	1
0	0	0
1	0	1
2	1	0
3	1	1

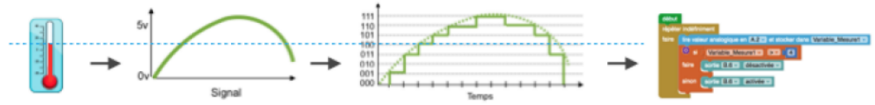
Numérisation sur 3 bits



Soit 8 valeurs possibles : de 0 à 7

Exemple :
100 en binaire correspond à 4 en décimal.

Puissance de 2	2^2	2^1	2^0
Décimal	4	2	1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1



Température extérieure
32°C

Acquisition en analogique
3,2 Volts

Numérisation
Binaire : 100

Traitement
Décimal : 4

Numérisation =

Échantillonnage dans le temps (prélèvement du signal à différents instants)

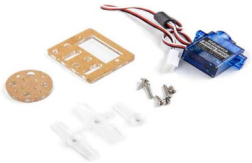
+

Quantification (attribution de la valeur binaire la plus proche ; dépend du nombre de bits de codage).

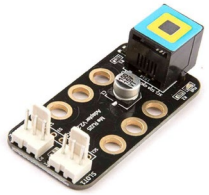
Plus le nombre de bits utilisé pour la numérisation est grand plus le signal numérique est proche du signal analogique et plus la numérisation est précise !

http://cache.media.eduscol.education.fr/file/Signal_et_information/42/2/RA16_C3_ST_signa_et_infol_N.D_581422.pdf

Modules



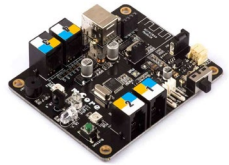
1 : 9g Micro Servo Motors
Tension : 4,8 Vcc à 6 Vcc.
Consommation typ. : 80 à 100 mA.
Consommation (stall) : 650 à 750 mA.
Torque: 1,3 à 1,7 kg/cm.



4: Cette carte permet de convertir le brochage RJ45 en brochage 6 broches. Le but est de pouvoir connecter des modules venant d'autres fabricants (température, servo moteur).

8: Temperature Sensor. Intègre un capteur de type DS18B20 à sortie numérique 1-wire Alimentation 3 à 5 Vcc. Plage de température du capteur: -55 °C à + 125°C.

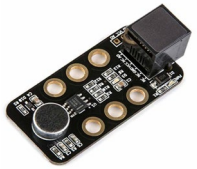
12: Ce module embarque un mini joystick 2 axes avec fonction bouton-poussoir. Il sera destiné à être raccordé sur 2 entrées "analogique" de votre microcontrôleur.



14: carte programmable composée d'un microprocesseur compatible arduino UNO-328 (base ATmega328) associé à un driver moteur lequel sera directement capable de piloter 2 moteurs "cc" (6 à 12 V / 1 A max.). C'est la platine utilisée dans les robots "mBot".

2 : potentiometer
Ce module embarque un potentiomètre rotatif. Il est destiné à être raccordé sur une entrée "analogique" de votre microcontrôleur.

5: Ce capteur de couleur est capable de reconnaître un total de six couleurs (noir, blanc, rouge, bleu, jaune et vert). Il se connecte sur le port standard bleu et blanc et utilise le protocole I2C pour la communication.

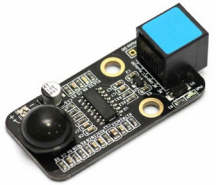


9: Ce module intègre un microphone électret associé à un amplificateur LM386. Sa sortie est destinée à être reliée sur une entrée de conversion "analogique/numérique" de votre platine afin que votre robot puisse réagir en fonction des bruits ambiants. La sensibilité de détection est réglable via un potentiomètre ajustable.

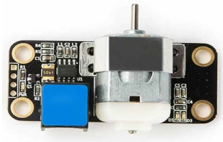
2 capteurs infrarouges émetteur /récepteur pour permettre à votre robot de suivre une ligne noir sur un sol blanc.
La portée de détection est de l'ordre de 1 à 2 cm max.

3 : Me Ultrasonic Sensor
Ce module capteur ultrason est capable de détecter un obstacle se trouvant à une distance comprise entre 3 cm et 4 m (avec un angle d'environ 30°).

6: Ce module Bluetooth est un transceiver Bluetooth (dual mode) spécialement conçu pour permettre à votre système makeblock ou votre arduino



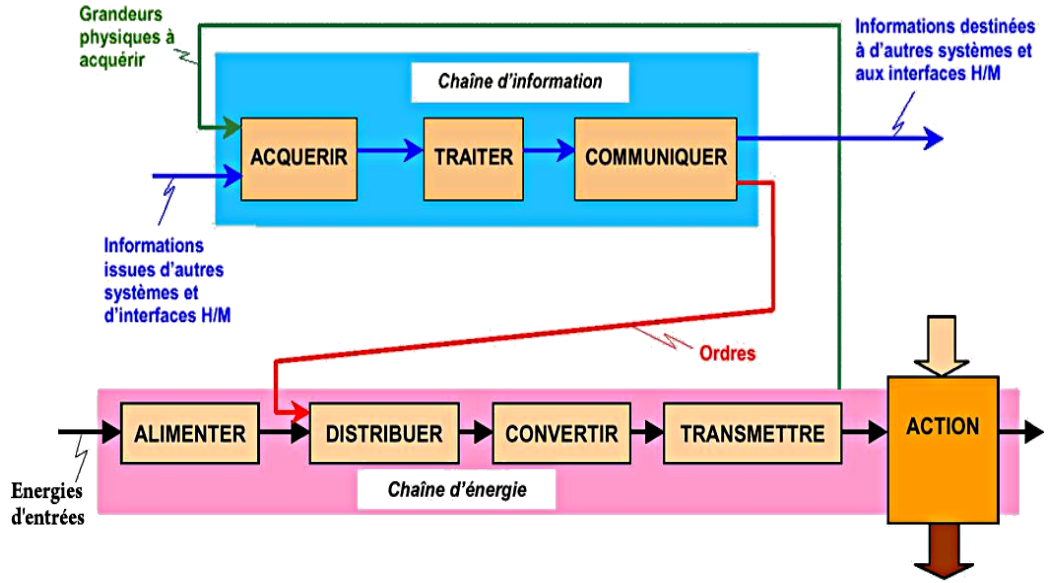
11: Ce module possède une capteur infrarouge passif PIR (pour un usage intérieur). Il est capable de détecter un déplacement humain sur une portée max. (réglable) de 6 m sur un angle d'environ 120°. Se connecte sur une entée/sortie numérique.



13: "Me Dual DC Motor Driver". Il est equipé d'une petite hélice permettant de visualiser plus facilement sa rotation.
- Alimentation: 12V DC.
- Vitesse (à vide): 16000 ±10 % rpm.
- Consommation (à vide): 90 ± 10% mA.

Travail à faire

La découpe fonctionnelle d'un système est présentée sur le schéma ci-dessous.



Question Identification composants

Identifier les composants par leurs numéros (voir figure « mbot composants matériels ») en exploitant la documentation fournie ci-après.

Remplir un tableau du modèle ci-dessous.

Vous trouverez des informations complémentaires sur le web.

https://www.makeblock.com/maker-platform/electronic-modules#

Acquérir	
Traiter	
Communiquer	
Alimenter	
Distribuer	
Convertir	
Transmettre	

Remarque : les composants servant à l'interconnexion d'autres composants seront à classer dans la rubrique « communication ». Certains composants peuvent ne plus exister, postérieurement à la récupération sur internet de la photo

Remarque sur la fonction DISTRIBUER

Elle est difficile à appréhender. Vous n'allez pas pouvoir identifier les modules matériels assurant cette fonction. Ils sont cachés à l'intérieur du bloc « unité centrale de commande »

3 modes de distribution:

- *allumer / éteindre*
- *envoyer l'énergie pour tourner dans un sens ou dans un autre*
- *envoyer plus ou moins d'énergie pour tourner vite ou pas.*

Ici, le moteur est un servomoteur, dont la distribution d'énergie est "transparente" pour l'installateur et l'utilisateur.

Question capteur infrarouge

a) Quelle est la nature de la grandeur mesurée par ce capteur ?
b) Quelle est la nature du signal envoyé par ce capteur, à destination de l'unité centrale (fonction Traiter) ?
c) Sur quelles bornes de l'unité centrale serait-il possible de connecter ce capteur ?
d) A l'inverse quelles sont les bornes inappropriées ? Expliquer pourquoi ?

Question détecteur de ligne

a) Quelle est la nature de la grandeur mesurée par ce capteur ?
b) Quelle est la nature du signal envoyé par ce capteur, à destination de l'unité centrale (fonction Traiter) ?
c) Sur quelles bornes de l'unité centrale serait-il possible de connecter ce capteur ?
d) A l'inverse quelles sont les bornes inappropriées ? Expliquer pourquoi ?

Question sorties

a) Quelles sont les sorties assurant une conversion « énergie électrique => énergie lumineuse » ?
b) Quelles sont les sorties assurant une conversion « énergie électrique => énergie mécanique » ?
c) Quelles sont les sorties assurant une conversion « énergie électrique => énergie ondulatoire » ?

Question carte à processeur

a) Quel est le type du processeur qui est au cœur de cette carte ?
b) Comment est alimentée cette carte ?
c) Quelles sont les entrées (fonction Acquérir) présentes sur cette carte ?
d) Quelles sont les sorties (fonction Communiquer ou fonction Distribuer) présentes sur cette carte ?

Question implantation du logiciel

Où doit être implanté le logiciel qui va piloter le mbot ?

a) Dans l'ordinateur ?
b) Dans le robot ?
c) Dans les deux ?

Question commande du robot

Comment est-il possible de commander le mbot ?

a) Avec le clavier du PC ?
b) Avec une télécommande infrarouge ?
c) Avec une manette de jeu bluetooth ?
d) Avec un joystick ?

Question langages de programmation

Parmi les langages de programmation ci-dessous, lesquels sont adéquats, lesquels ne le sont pas ?

a) mBlock 4
b) mBlock 5
c) des applications Androïd
d) des applications IOS
e) langage C pour arduino
f) App inventor

Réponses

CORRECTION

Réponse Identification composants

1 = convertir
2 = acquérir
3 = acquérir
4 = communication
5 = acquérir
6 = communiquer
7 = inconnu
8 = acquérir
9 = acquérir
10 = inconnu
11 = acquérir
12 = acquérir
13 = converti
14 = traiter

Réponse capteur infrarouge

a) Mesure de la distance
b) Signal numérique envoyé à la carte mcore, via un bus I2C
c) On peut connecter ce capteur sur un port étiqueté en blanc et bleu (bus I2C)
d) On ne peut surtout pas le connecter sur un port rouge (alimentation) ou un port noir (signal analogique)

Réponse détecteur de ligne

a) Détection de la couleur noire (en fait de l'absence de réflexion de la lumière)
b) Signal numérique envoyé à la carte mcore, via un bus I2C (notion de bus à éventuellement présenter dans un document annexe)
c) On peut connecter ce capteur sur un port étiqueté en blanc et bleu (bus I2C)
d) On ne peut surtout pas le connecter sur un port rouge (alimentation) ou un port noir (signal analogique)

Réponse sorties

a) Deux LED RGB
b) Deux moteurs
c) Un buzzer

Réponse carte à processeur

a) Processeur de type arduino
b) Alimentation par piles (4 fois 1,2V), par batterie ou par port USB
c) Un bouton poussoir est intégrée à la carte ainsi qu'un bouton On/off de mise sous tension générale
d) Deux LED RGB ainsi qu'un buzzer sont intégrés, ainsi qu'un éventuel module de communication (à rajouter, soit en bluetooth, soit en WIFI (2,4G)

Réponse implantation du logiciel

Il faut l'implanter dans le robot. Le programme se lancera automatiquement à la mise sous tension. Mais il pourra rester en attente d'une action opérateur, via (par exemple) l'appui sur le bouton poussoir intégré à la carte mère.
Parfois, certains logiciels permettent le pilotage du robot mbot par le PC. Mais c'est un peu particulier. En outre il faut pour cela que la carte mère du robot soit en attente d'ordres envoyés par le PC via le câble USB (ou une connexion sans fil). C'est trop spécifique et trop difficile à appréhender pour un élève.

Réponse commande du robot

a) Avec le clavier du PC, difficilement (voir ci-dessus)
b) Avec une télécommande infrarouge, oui (voir https://www.makeblock.com/project/me-ir-remote-controller)
c) Avec une manette de jeu bluetooth, oui (voir https://www.makeblock.com/project/bluetooth-controller)
d) Avec un joystick, oui (voir https://www.makeblock.com/project/me-joystick)

Réponse langages de programmation

a) mBlock 4, oui mais un peu vieillot
b) mBlock 5, oui, à privilégier
c) applications andoïd, oui
d) applications IOS, oui
e) langage C pour arduino, oui, mais demande un bon investissement
f) appinventor, très intéressant, mais demande un bon investissement

Activité 2) Programmation de mblock à arduino

On utilise mBlock 5 for PC http://www.mblock.cc/mblock-software/

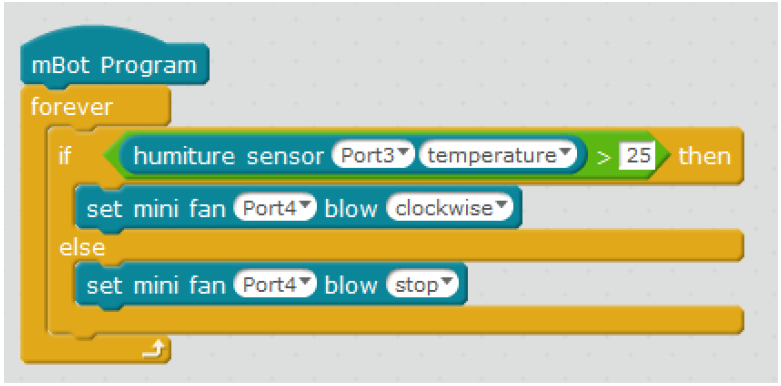
Objectifs: se confronter au réel, télécharger le programme, rajouter des modules matériels, en choisir (et acheter) de nouveaux, comprendre que les connexions sont délicates …

Sous mBlock 5, installer le périphérique Mbot (en plus du Codey qui est installé par défaut). Brancher un mbot réel avec un câble USB. Passer en mode connecté.
Il faut utiliser les périphériques liés au mbot et au code qui va désormais être embarqué dans le processeur du petit robot (processeur de type arduino). On dispose donc d'un petit bouton sur le mBot, de divers capteurs ainsi que de modules additionnels possibles. On dispose surtout d'une télécommande !
Une fois le code téléchargé dans le robot, le cordon ombilical avec le PC est coupé.

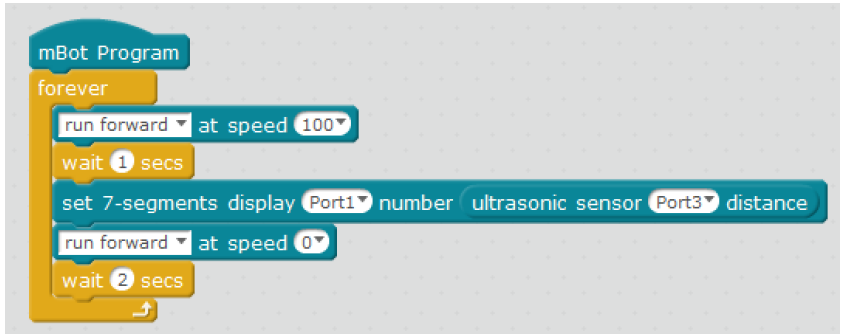
Le but de cette activité n'est pas la programmation mblock à proprement parler. Elle est certainement trop simple pour des élèves qui ont vu des produits similaires au collège.
Le but est d'identifier le code réel (arduino) qui est caché sous l'interface graphique.

Voici quand même quelques exemples de programme pour s'entrainer

Active le ventilateur en fonction de la température



Afficher la distance à un obstacle



Ce module intègre un afficheur 7 segments à leds rouges de 4 digits pilotés par un circuit intégré "TM1637".

Matrice 8 x 16 LEDs. Permet l'animation du robot mBot à la place du capteur ultrason. Chaque LED est paramétrable via le logiciel mBlock.

Suiveur de ligne

```
#include <MeMCore.h>
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

MeUltrasonicSensor ultrasonic_3(3);
MeDCMotor motor_9(9);
MeDCMotor motor_10(10);
void move(int direction, int speed) {
int leftSpeed = 0;
int rightSpeed = 0;
if(direction == 1) {
leftSpeed = speed;
rightSpeed = speed;
} else if(direction == 2) {
leftSpeed = -speed;
rightSpeed = -speed;
} else if(direction == 3) {
leftSpeed = -speed;
rightSpeed = speed;
} else if(direction == 4) {
leftSpeed = speed;
rightSpeed = -speed;
}
motor_9.run((9) == M1 ? -(leftSpeed) :
(leftSpeed));
motor_10.run((10) == M1 ? -(rightSpeed) :
(rightSpeed));
}

_loop();
}
```

VOIR SUITE =>

<= SUITE

```
void _delay(float seconds) {
long endTime = millis() + seconds *
1000;
while(millis() < endTime) _loop();
}

void setup() {

}

void _loop() {
}

void loop() {
if(ultrasonic_3.distanceCm() > 50){
move(1, 50 / 100.0 * 255);

}else{
move(2, 50 / 100.0 * 255);
_delay(5);

}
```

- Où se trouve le programme principal en rapport avec le formalisme mblock ?
- Retrouver dans le code les numéros des broches de la carte mère sur lesquelles sont branchés le capteur ultrason, le moteur droit et le moteur gauche
- Comment est codée la temporisation ?
- La commande des moteurs est assurée par un sous-programme. Colorier en vert le code correspondant à sa définition. Colorier en bleu le code correspondant à son appel.
- Indiquer comment sont indiqués les sens de « direction » des moteurs

Suiveur de ligne

```
#include <MeMCore.h>
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

MeUltrasonicSensor ultrasonic_3(3);
MeDCMotor motor_9(9);
MeDCMotor motor_10(10);

void move(int direction, int speed) {
int leftSpeed = 0;
int rightSpeed = 0;
if(direction == 1) {
leftSpeed = speed;
rightSpeed = speed;
} else if(direction == 2) {
leftSpeed = -speed;
rightSpeed = -speed;
} else if(direction == 3) {
leftSpeed = -speed;
rightSpeed = speed;
} else if(direction == 4) {
leftSpeed = speed;
rightSpeed = -speed;
}
motor_9.run((9) == M1 ? -(leftSpeed) :
(leftSpeed));
motor_10.run((10) == M1 ? -(rightSpeed) :
(rightSpeed));
}

_loop();
}
```

VOIR SUITE =>

<= SUITE

```
void _delay(float seconds) {
long endTime = millis() + seconds *
1000;
while(millis() < endTime) _loop();
}

void setup() {

}

void _loop() {
}

void loop() {
if(ultrasonic_3.distanceCm() > 50){
move(1, 50 / 100.0 * 255);

}else{
move(2, 50 / 100.0 * 255);
_delay(5);

}
```

CORRECTION

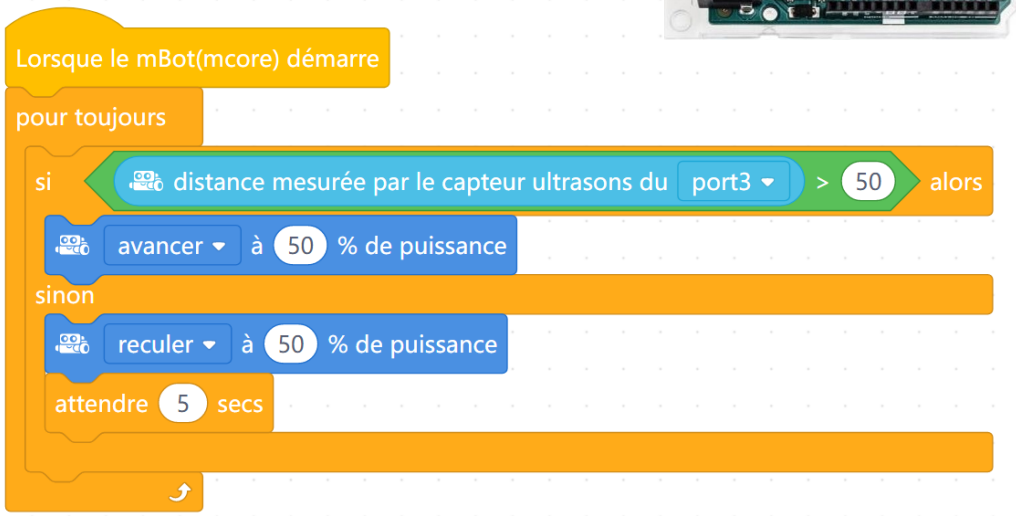
- Où se trouve le programme principal en rapport avec le formalisme mblock ?
- Retrouver dans le code les numéros des broches de la carte mère sur lesquelles sont branchés le capteur ultrason, le moteur droit et le moteur gauche
- Comment est codée la temporisation ?
- La commande des moteurs est assurée par un sous-programme. Colorier en vert le code correspondant à sa définition. Colorier en bleu le code correspondant à son appel.
- Indiquer comment sont indiqués les sens de « direction » des moteurs

La direction est codée par 4 numéros
1 = avance
2 = recule
3 = tourne à droite
4 = tourne à gauche
Ceci se retrouve dans les 4 « if » du sous programme move()

Oscillation devant un obstacle détecté par ultrason

```
#include <MeMCore.h>
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

MeUltrasonicSensor ultrasonic_3(3);
MeDCMotor motor_9(9);
MeDCMotor motor_10(10);
void move(int direction, int speed) {
  int leftSpeed = 0;
  int rightSpeed = 0;
  if(direction == 1) {
    leftSpeed = speed;
    rightSpeed = speed;
  } else if(direction == 2) {
    leftSpeed = -speed;
    rightSpeed = -speed;
  } else if(direction == 3) {
    leftSpeed = -speed;
    rightSpeed = speed;
  } else if(direction == 4) {
    leftSpeed = speed;
    rightSpeed = -speed;
  }
  motor_9.run((9) == M1 ? -(leftSpeed) : (leftSpeed));
  motor_10.run((10) == M1 ? -(rightSpeed) :
(rightSpeed));
}
VOIR SUITE =>
```

```
<= SUITE
void _delay(float seconds) {
  long endTime = millis() + seconds * 1000;
  while(millis() < endTime) _loop();
}

void setup() {

}

void _loop() {
}

void loop() {
  if(ultrasonic_3.distanceCm() > 50){
      move(1, 50 / 100.0 * 255);

  }else{
      move(2, 50 / 100.0 * 255);
      _delay(5);

  }

  _loop();
}
```



- Chercher la partie du code effectuant le test de distance. Colorier en rouge les marqueurs de début et de fin
- Chercher la partie du code faisant avancer à 50% de la puissance. Expliquer comment est exprimé ce pourcentage. L'information étant codée en numérique sur 8 bits, expliquer l'origine du nombre « 255 »
- Sachant que la variable speed contient l'information de vitesse souhaitée, essayer de comprendre l'usage qu'en fait le sous-programme de pilotage de la vitesse. Ce sous-programme est nommé « move() ».

Oscillation devant un obstacle détecté par ultrason

CORRECTION

```
#include <MeMCore.h>
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

MeUltrasonicSensor ultrasonic_3(3);
MeDCMotor motor_9(9);
MeDCMotor motor_10(10);
void move(int direction, int speed) {
  int leftSpeed = 0;
  int rightSpeed = 0;
  if(direction == 1) {
    leftSpeed = speed;
    rightSpeed = speed;
  } else if(direction == 2) {
    leftSpeed = -speed;
    rightSpeed = -speed;
  } else if(direction == 3) {
    leftSpeed = -speed;
    rightSpeed = speed;
  } else if(direction == 4) {
    leftSpeed = speed;
    rightSpeed = -speed;
  }
  motor_9.run((9) == M1 ? -(leftSpeed) : (leftSpeed));
  motor_10.run((10) == M1 ? -(rightSpeed) :
(rightSpeed));
}
VOIR SUITE =>
```

```
<= SUITE
void _delay(float seconds) {
  long endTime = millis() + seconds * 1000;
  while(millis() < endTime) _loop();
}

void setup() {

}

void _loop() {
}

void loop() {
  if(ultrasonic_3.distanceCm() > 50){
      move(1, 50 / 100.0 * 255);

  }else{
      move(2, 50 / 100.0 * 255);
      _delay(5);

  }

  _loop();
}
```

- Chercher la partie du code effectuant le test de distance. Colorier en rouge les marqueurs de début et de fin
- Chercher la partie du code faisant avancer à 50% de la puissance. Expliquer comment est exprimé ce pourcentage. L'information étant codée en numérique sur 8 bits, expliquer l'origine du nombre « 255 »
- Sachant que la variable speed contient l'information de vitesse souhaitée, essayer de comprendre l'usage qu'en fait le sous-programme de pilotage de la vitesse. Ce sous-programme est nommé « move() ».

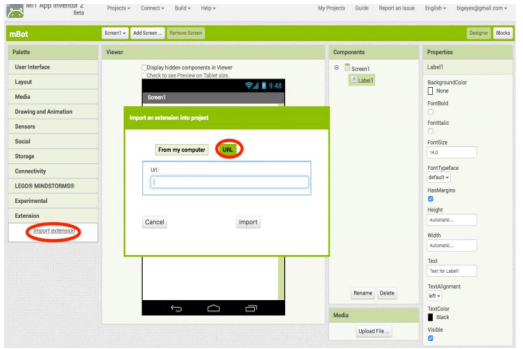
Sur 8 bits les valeurs codées en binaire vont de 0 à 255. Il faut mettre dans la variable « speed » qui est passée au SP « move » un nombre correspondant à 50% de cette valeur maximale

Les marqueurs de début et de fin sont des accolades

Lors de l'appel on passe le chiffre correspondant au sens de déplacement, puis la vitesse codée sur 8 bits. Dans le SP cette valeur est mise dans deux autres variables « leftSpeed » et « rightSpeed en l'inversant si nécessaire.
Ensuite deux autres SP (motor_9 …) les exploitent. Ces SP proviennent de la bibliothèque MeMCore.h

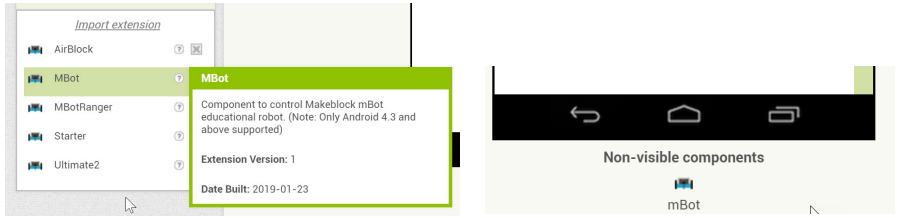
Activité 3) IHM AppInventor pour piloter Mbot et acquérir des entrées

La commande des objets connectés se fait de plus en plus souvent à distance, avec un téléphone ou une tablette.
Nous allons développer un IHM (Inteface Homme Machine) avec App Inventor.
Cet environnement de programmation, géré par le MIT, permet de développer des applications pour les téléphones Androïd. La communication se fera par bluetooth. La commande se fera grâce à App Inventor qui pilotera directement les entrées/sorties.

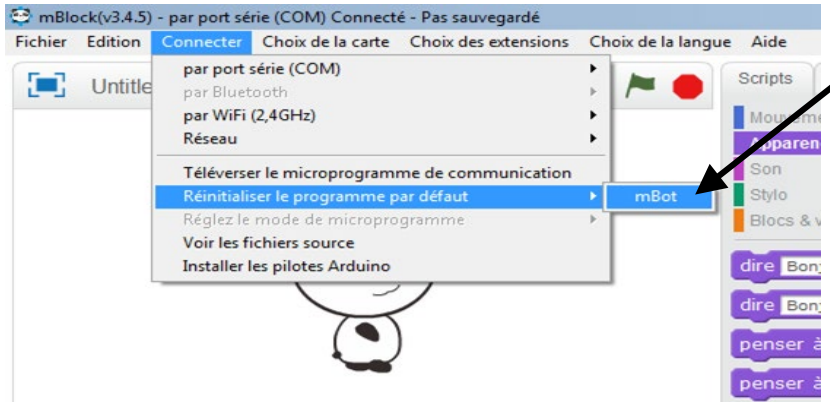


Présentation de AppInventor
http://appinventor.mit.edu/explore/ et https://makeblock.com/project/use-mbot-with-app-inventor-v-1-9

Il faut surtout faire attention à bien installer l'extension mbot.
La procédure est indiquée ici: https://makeblock.com/project/use-mbot-with-app-inventor.
Mais il faut utiliser l'extension disponible sur le site suivant: https://appstatic.makeblock.com/extension/makeblock_2.0.4.aix
Il faut ensuite déposer l'extension sur l'écran. Le composant est invisible, mais présent.



Pour que la commande avec App Inventor fonctionne, il faut commencer par effacer tout programme existant dans le robot. Pour cela il faut connecter le robot au PC (via un câble USB) et télécharger un programme vide. Sinon on peut se faire piéger. La connexion bluetooth fonctionne, mais le robot ne répond pas.



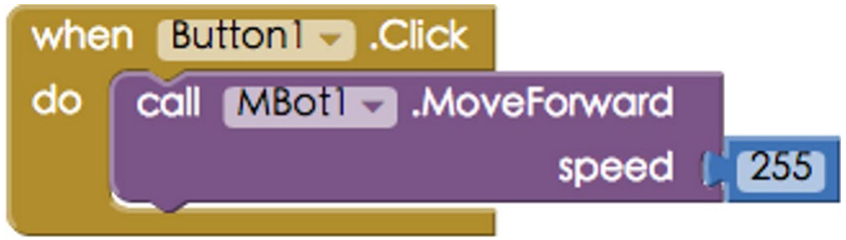
Pour pouvoir **piloter le robot** mbot d'App Inventor, il est **impératif** de le **reprogrammer avec son programme par défaut** à partir du logiciel **mBlock**.

Modules de programmation

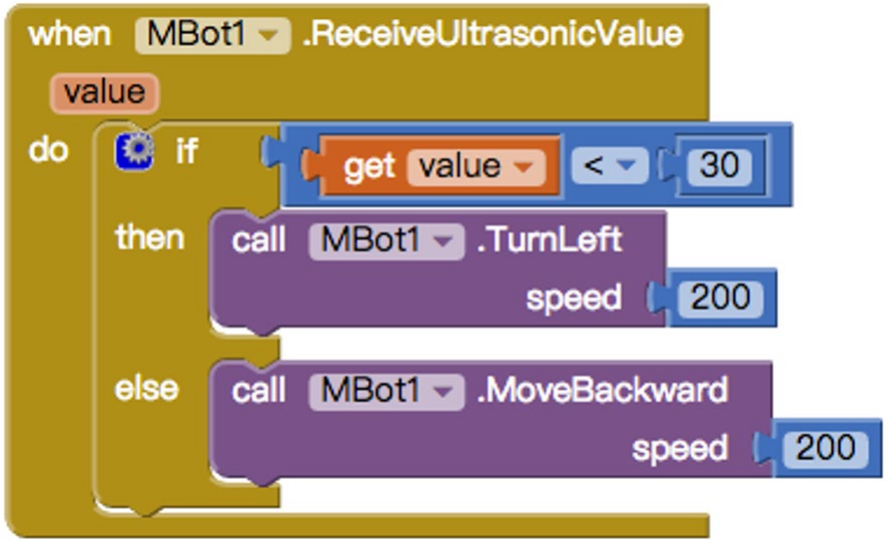
Pour le Bluetooth

Pour la lecture d'un capteur

Pour la commande de sorties (moteur)



Pour la lecture des capteurs et le lancement d'une action en conséquence



Travail à faire

L'objectif est de piloter le robot à distance, avec un téléphone android.
Il faut noter que la programmation Appinventor est une programmation orientée objet. C'est-à-dire qu'il faut commencer par designer son interface, avant d'écrire des blocs de commande associés à chacun des éléments de l'interface.

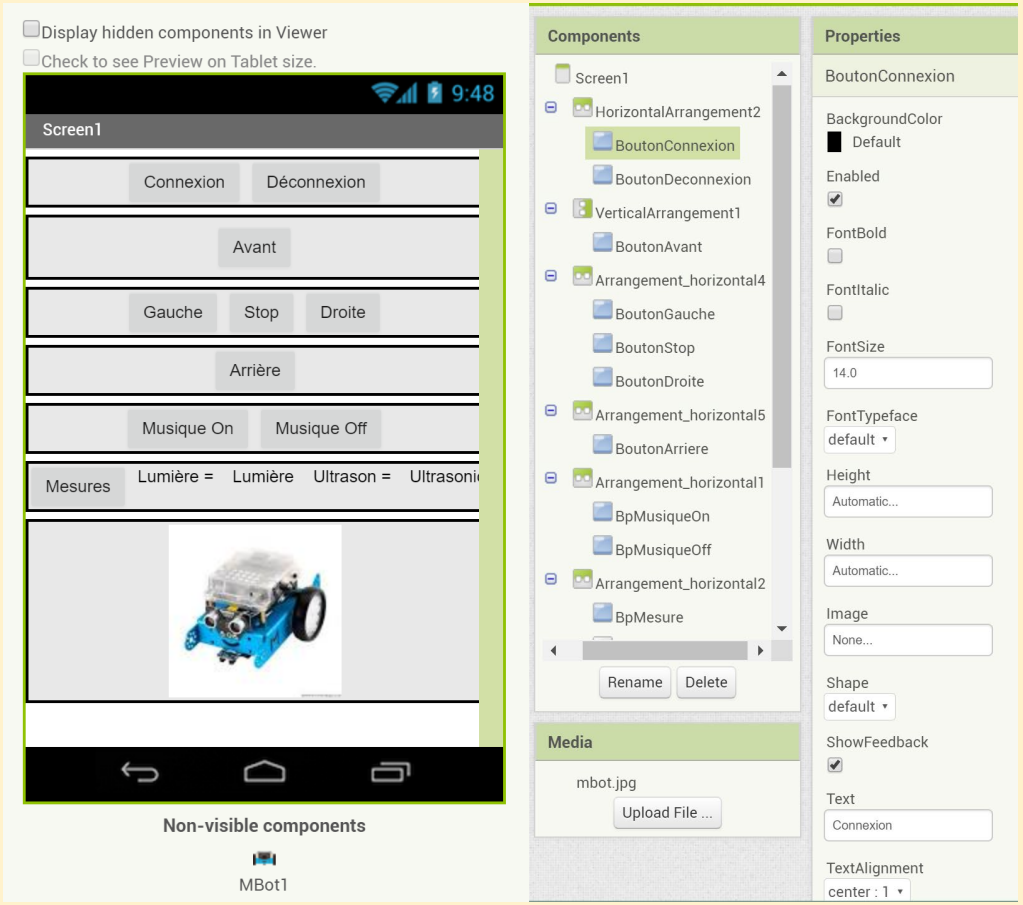
Avec l'aide du professeur, effectuer les tâches suivantes.

- Déposer un quelconque élément sur l'interface (en mode designer) et essayer de le faire apparaitre sur votre téléphone ou tablette. C'est là où résident les sources majeures de problèmes. Il est déconseillé d'essayer de travailler avec le simulateur intégré dans appinventor. Si vous n'avez ni wifi, ni téléphone (ou tablette), il vaut mieux abandonner. Vous allez perdre beaucoup de temps.
- Faire l'interface et la programmation du module d'établissement de la connexion bluetooth. Eventuellement prévoir d'afficher des messages « connexion ok ». Tester.
- Faire l'interface de commande des moteurs en marche avant. Tester. Si ok, compléter pour les 4 directions.
- Faire l'interface de récupération puis d'affichage des valeurs des capteurs.

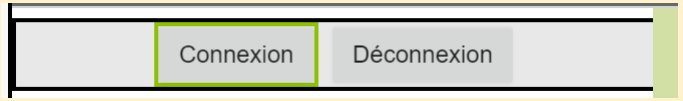
Solution

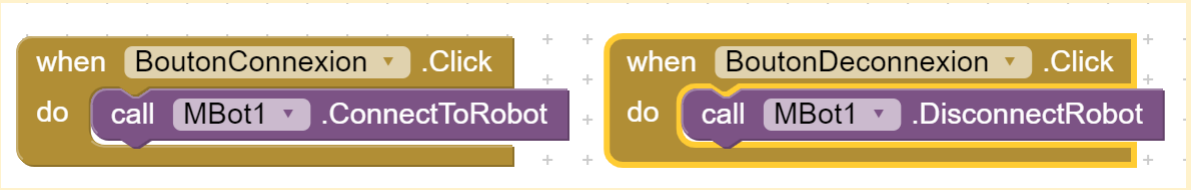
CORRECTION

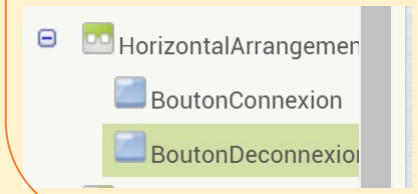
Design



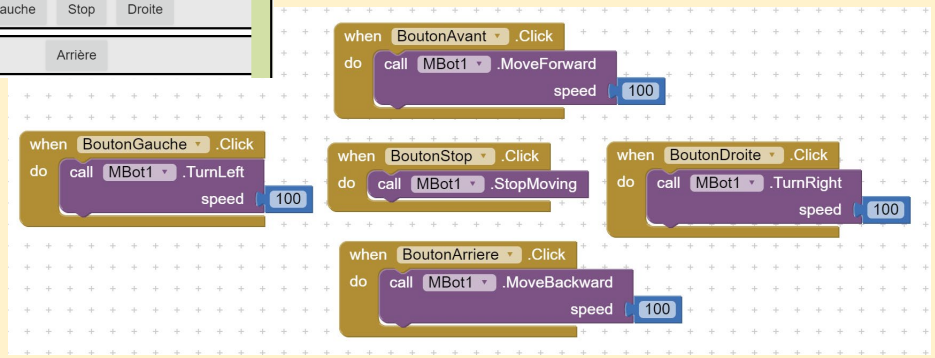
Bluetooth





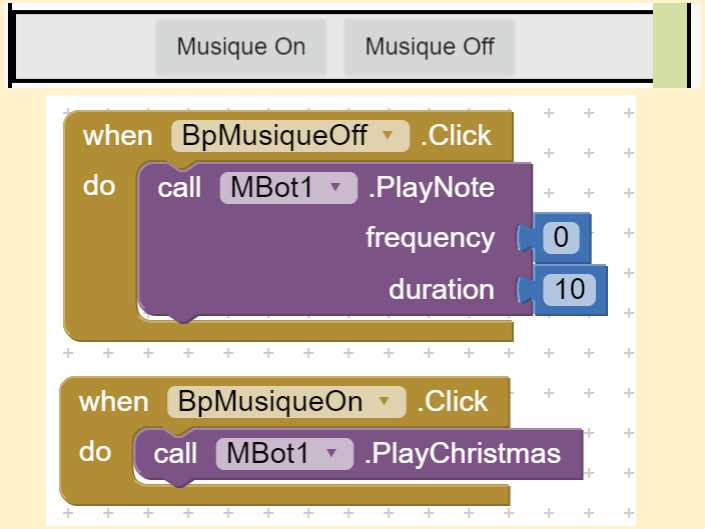


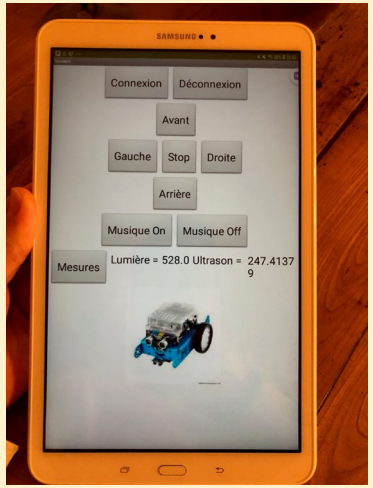
Déplacement



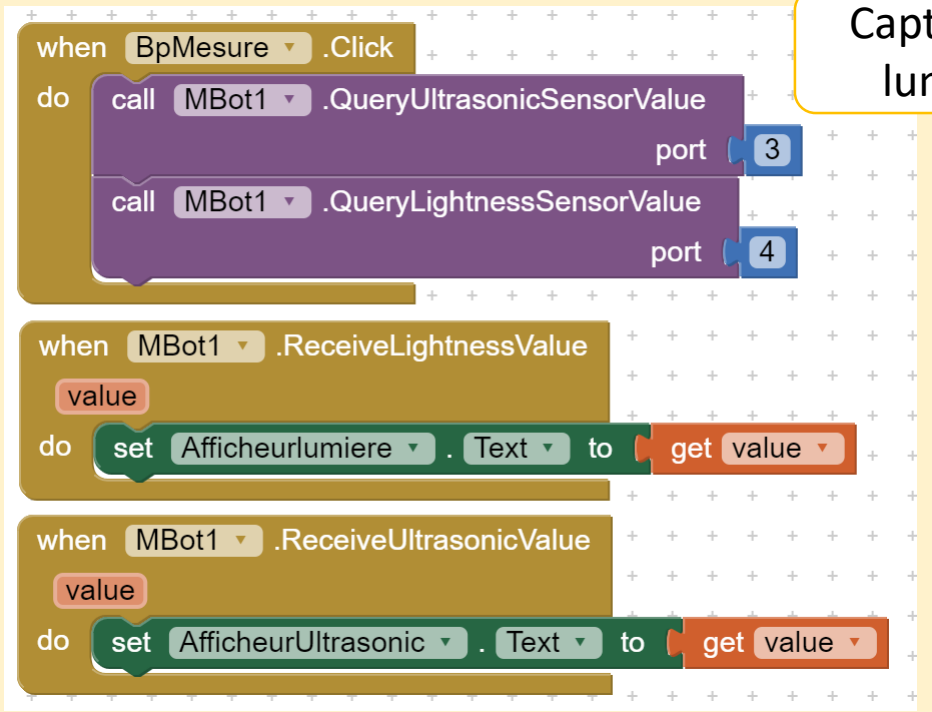
CORRECTION

Musique

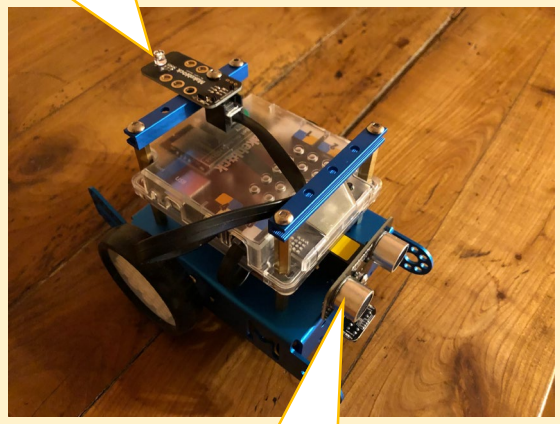




Capteur de lumière et de distance



Capteur de lumière



Capteur de distance