Programme ton robot OTTO

Atelier du 9 fevrier 2019

avec Jacques, Michel et Philippe

www.beauxboulons.com

Programme de l'atelier du 9 février 2019

1. Copie des fichiers / mise en place des environnements Arduino et Mblock sur les postes des Participants
   - Ou utilisation des PCs portables du fablab préchargés/préconfigurés
2. Installation / prise en main de mBlock
3. Calibration du robot avec l'IDE Arduino
4. Programmation mBlock mode interactif/pilotage du robot
5. Programmation mBlock mode Arduino
6. Programmation du robot au travers de l'IDE Arduino

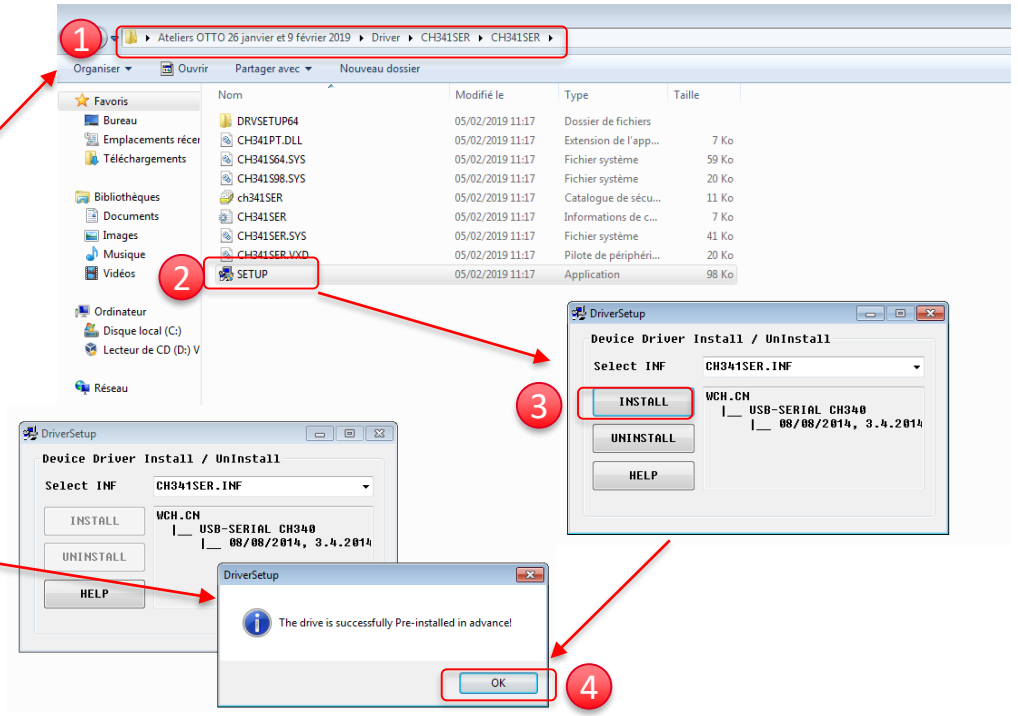
Copie / Installation du dossier « Ateliers OTTO … »

Utilisateurs Windows

1. **Copier depuis la clé USB le dossier « Ateliers OTTO 26 janvier et 9 février 2019 » sur votre disque dur ou sur le Bureau**

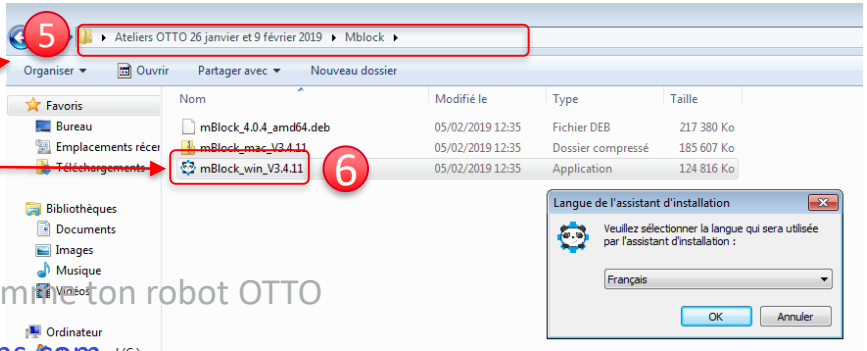
2. **Installer le « driver » USB spécifique lié à la carte Nano utilisée**

   - Aller dans le sous-dossier « **Driver\CH341SER\CH341SER** » puis cliquer/exécuter le programme **SETUP**
   - Et cliquer le bouton « **INSTALL** »
   - Cliquer enfin la bouton **OK** lorsque la boite de dialogue suivante s'affiche



3. **Installer Mblock**

   - Aller dans le sous-dossier « **Mblock** » puis cliquer/exécuter le programme **mBlock_win_V3.4.11**

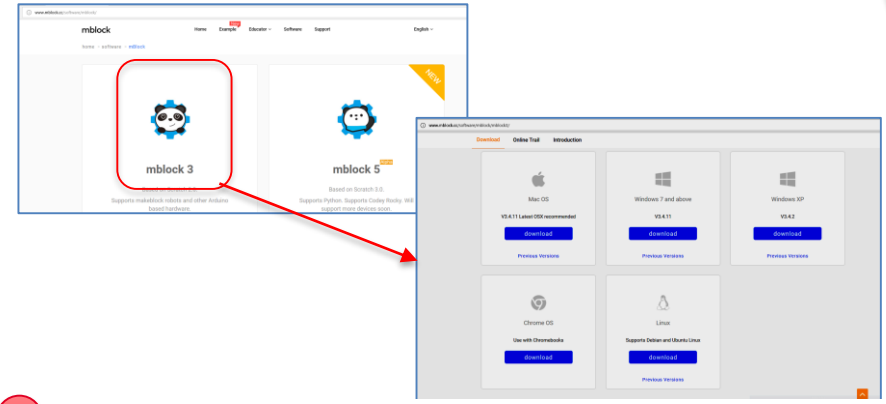


- **Puis aller au point 2 voir page suivante**

Installation de Mblock version 3.4.11

Aller directement au point 2 si vous avez copié l'environnement depuis la clé USB

Aller sur le site http://www.mblock.cc/software/mblock



1 Téléchargement

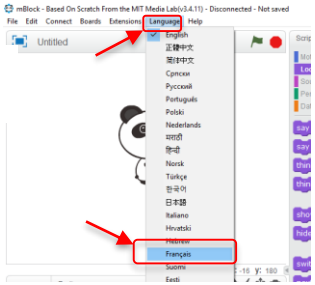
Sélectionner / cliquez Mblock3 puis choisir la version adaptée à votre OS

- Pour Windows, télécharger la version 3.4.11 de mblock3

2 Installation, lancement et configuration

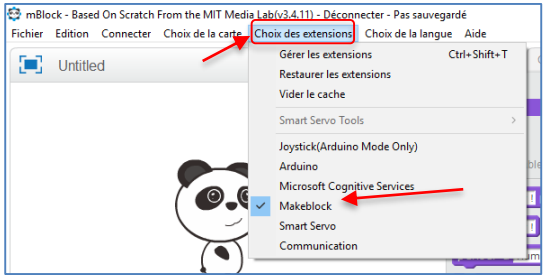
- choisir la langue

→ Menu Language > Français

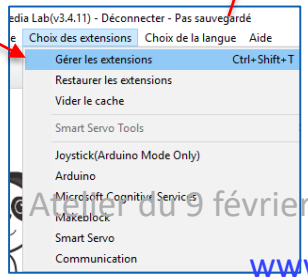


3 Installation des extensions Otto

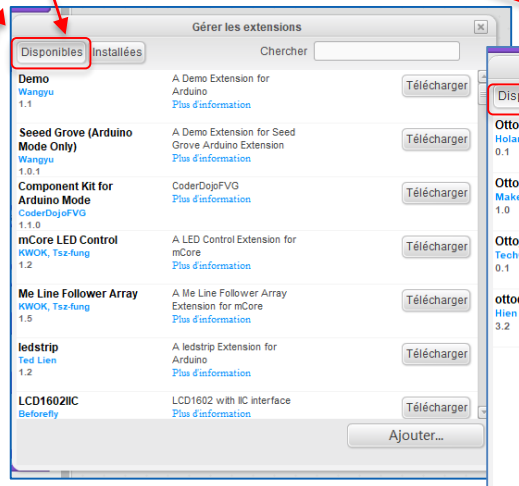
- Menu Choix des extension
- Déselectionner Makeblock

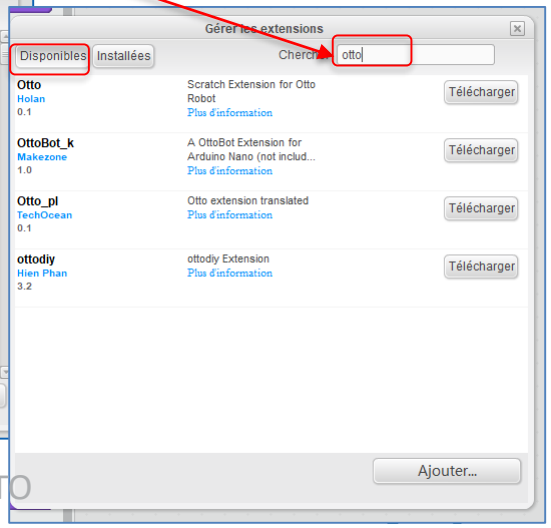


puis cliquer **Gérer les extension**

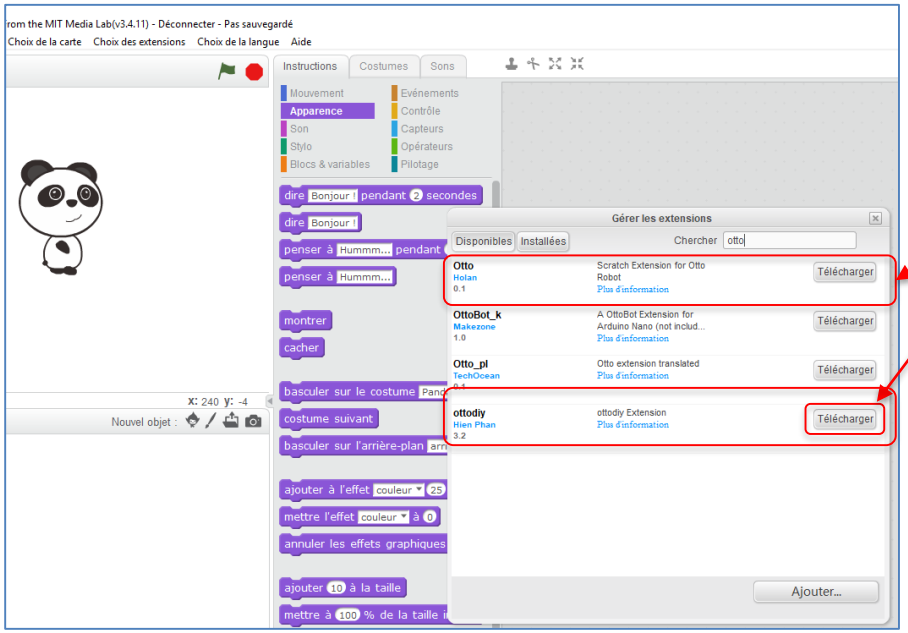


- Cliquer l'onglet Disponibles puis saisir **Otto** dans la zone Chercher





mBlock : installation des extensions Otto

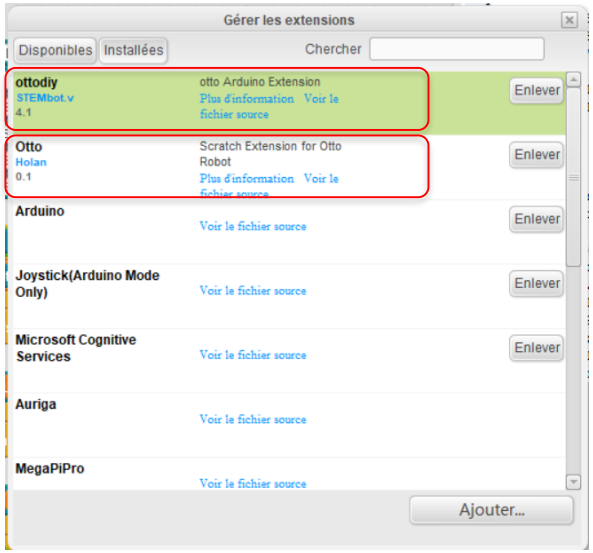


4- Installation des extensions Otto (suite)

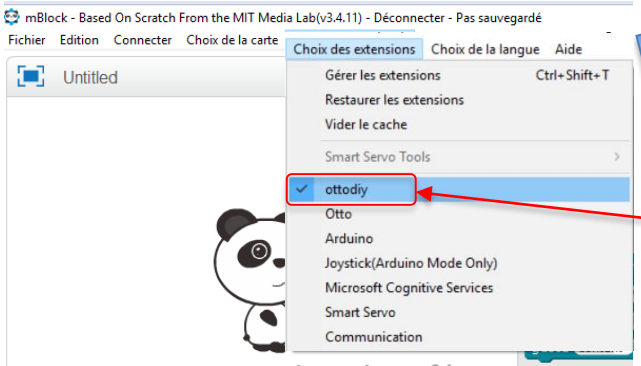
- Dans la fenêtre Gérer l'extension, onglet Disponible, cliquez « Télécharger » sur les lignes otto et ottodiy

1

- Cliquez ensuite l'onglet « Installées » pour vérifier la bonne installation de l'extension ottodiy téléchargée



5 – Enfin, vérifier dans Menu → Choix des extensions leur activation



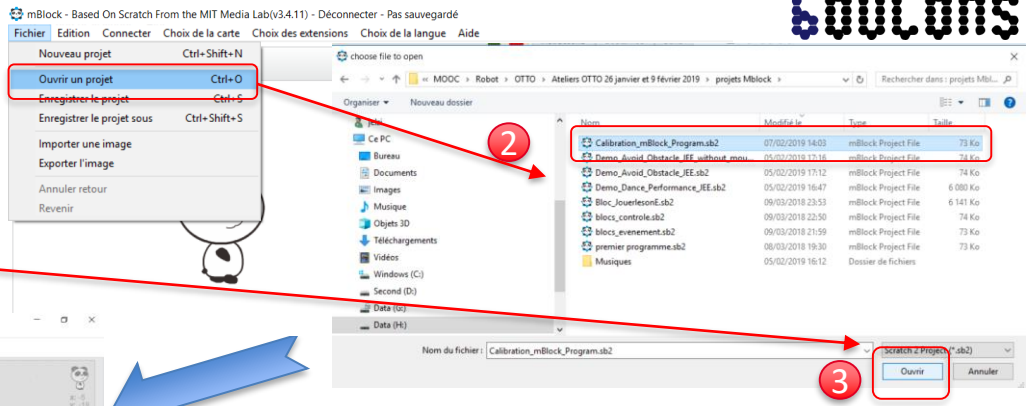
et quittez l'écran de gestion des extensions

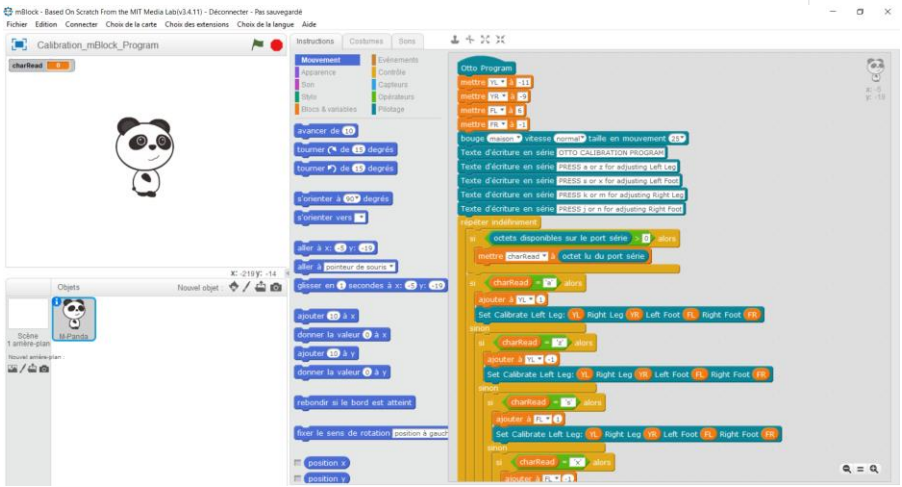
2 **Activez uniquement l'extension ottodiy dans un premier temps**

Calibrer le robot OTTO avec Calibration_mBlock (1/3)

1- Charger le programme « Calibration_mBlock_Program.sb2 »

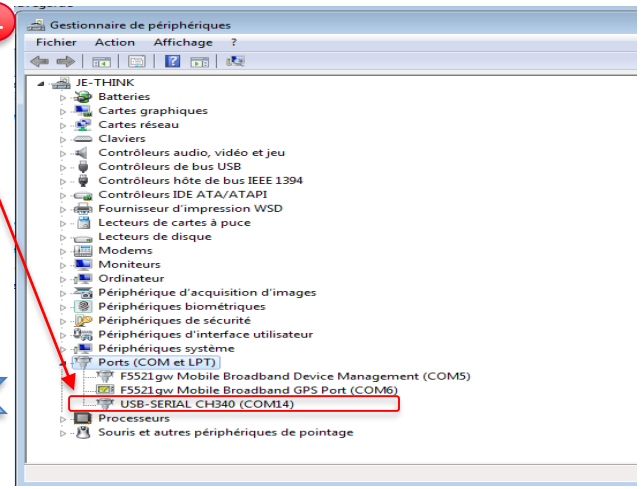
- Menu **Fichier – Ouvrir un projet**
- Aller dans le sous-dossier « **projets mBlock** »
- et sélectionner le fichier « **Calibration_mBlock_Program.sb2** » puis cliquer le bouton **Ouvrir**





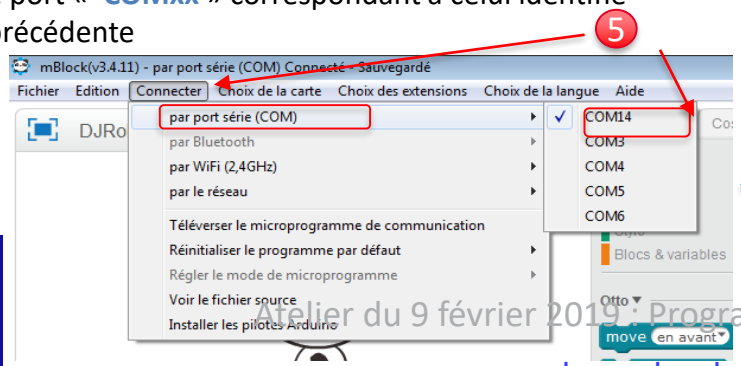
2- identifier le port COM auquel est connecté le robot Otto

- - Sous Windows; pour identifier le port COM auquel est connecté le robot Otto, aller dans « **Panneau de Configuration** » → « **Gestionnaire de périphériques** »



3- menu « Connecter » et choisir « par port série (COM) »

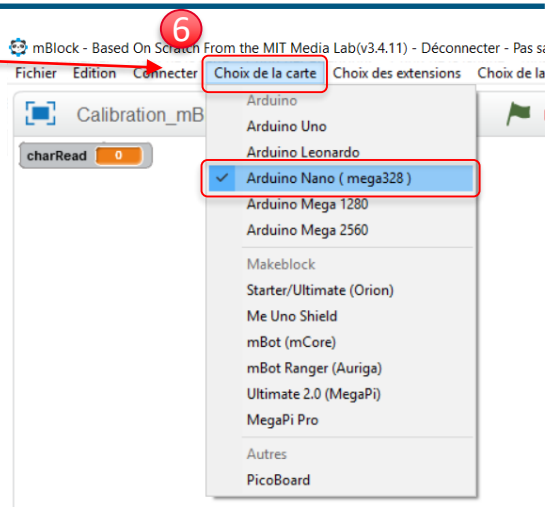
- Puis cliquer le port « **COMxx** » correspondant à celui identifié dans l'étape précédente



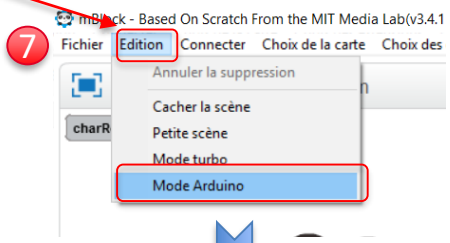
Calibrer le robot OTTO avec mBlock (2/3)

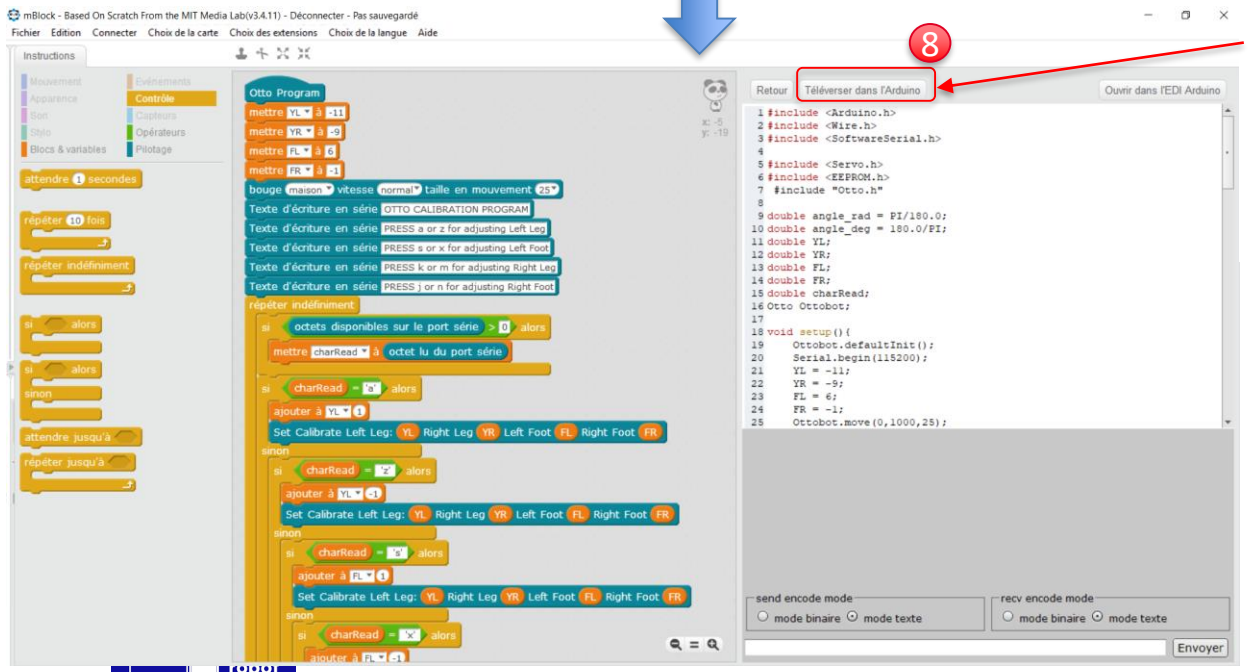
4- Menu « Choix de la carte »

- sélectionner **« Arduino Nano (mega328) »**



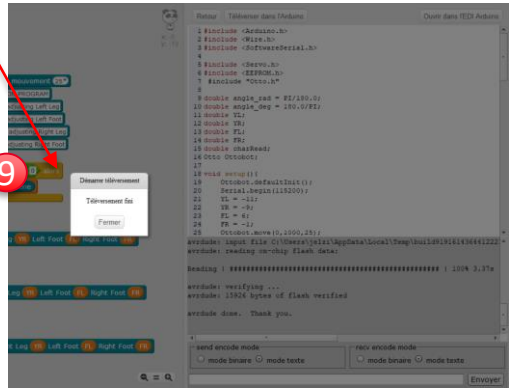
5- Puis menu « Edition » et cliquer « **mode Arduino** »





6- Cliquer le bouton « **Téléverser dans l'Arduino** »

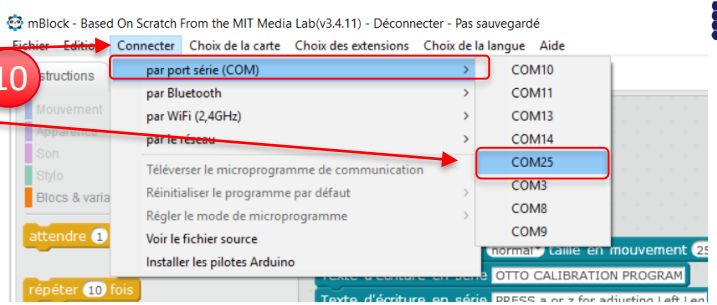
- Une boite de dialogue s'affiche « **Téléversement en cours** »
- Cliquer « **Fermer** » quand « **Téléversement fini** » est affiché

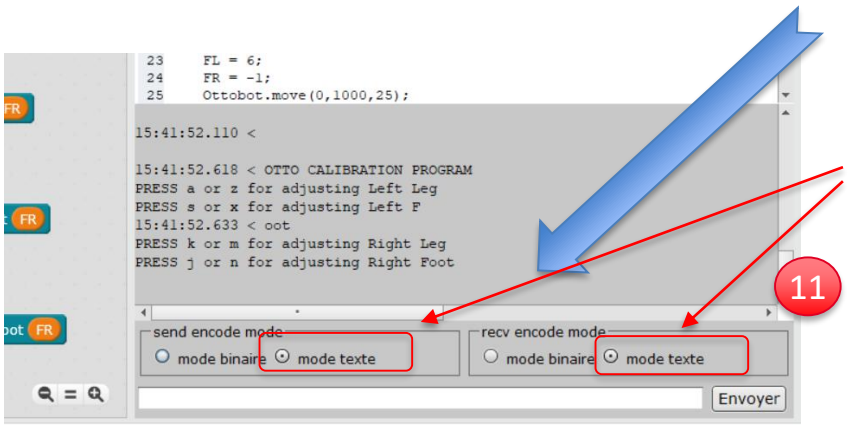


Calibrer le robot OTTO avec mBlock (3/3)

7- Reconnecter le robot au travers du menu « Connecter » et choisir « par port série (COM) »

- Puis cliquer le port « COMxx » correspondant à celui identifié dans l'étape précédente

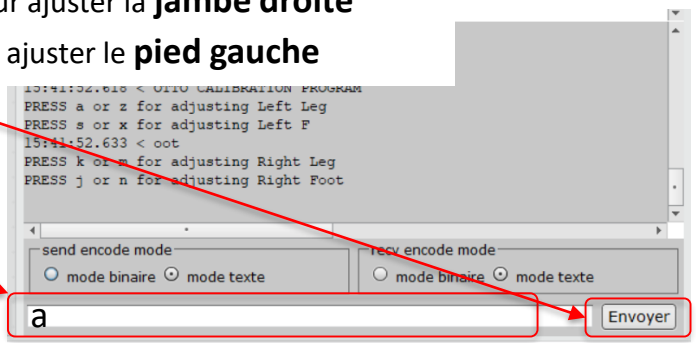




8- Cliquer le mode « mode texte » dans les cadres « send encode mode» et « receive encode mode »

9- Renverser / mettre le robot sur la tête face à vous puis

- Utiliser / taper les touches **a** ou **z** et cliquer le bouton **Envoyer** pour ajuster la **jambe gauche**
- Utiliser / taper les touches **s** ou **x** et cliquer le bouton **Envoyer** pour ajuster le **pied gauche**
- Utiliser / taper les touches **k** ou **m** et cliquer le bouton **Envoyer** pour ajuster la **jambe droite**
- Utiliser / taper les touches **j** ou **n** et cliquer le bouton **Envoyer** pour ajuster le **pied gauche**



Programmer OTTO AVEC mBlock

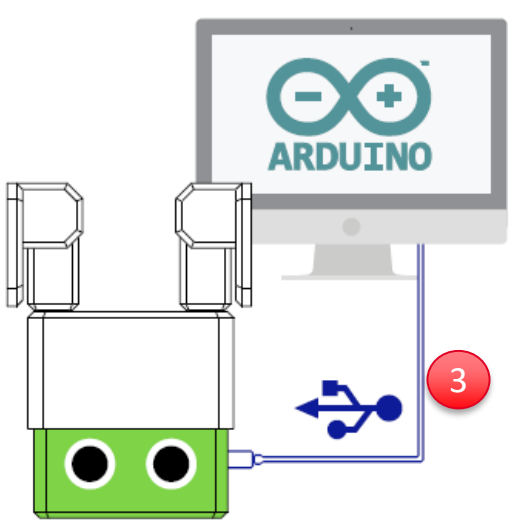
Lancement de l'IDE Arduino : connexion du robot Otto

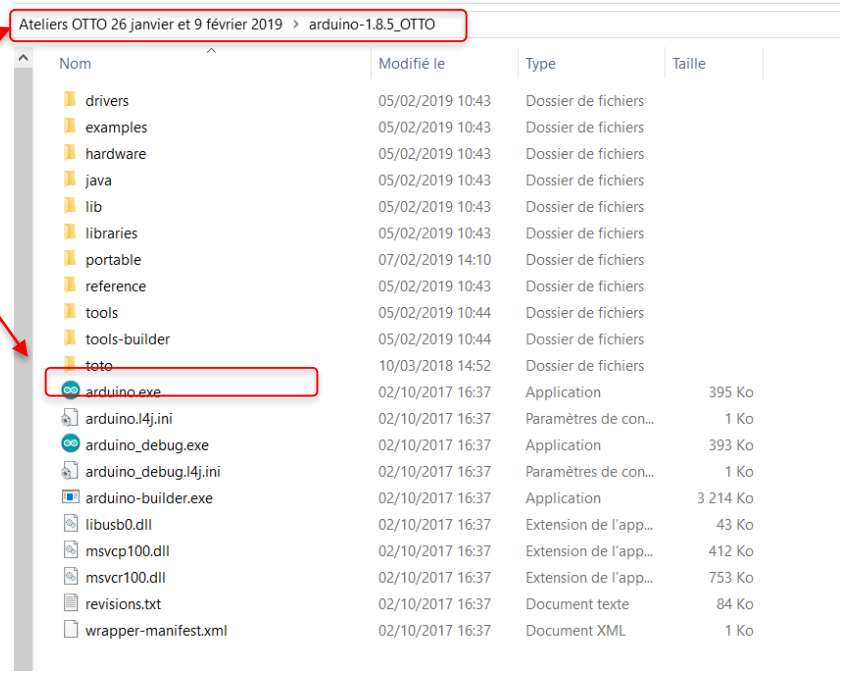
→ **Objectif : précharger le programme Arduino qui permet le dialogue avec mBlock**

a Aller dans le sous-dossier « **arduino-1.8.5_OTTO** »

puis cliquer/exécuter le programme « **arduino.exe** »

b Connecter le robot Otto sur un port USB



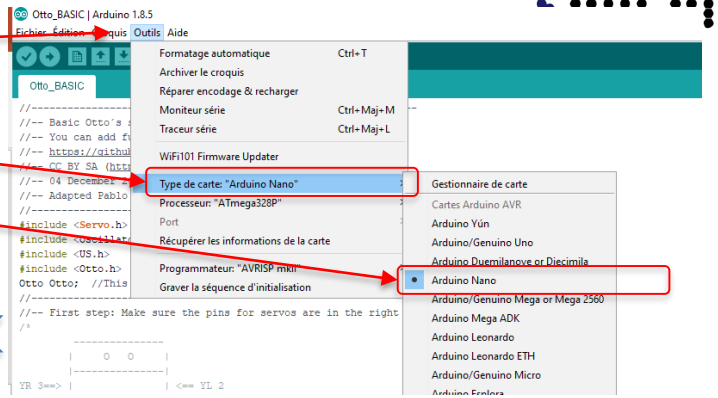


_V3.4.11

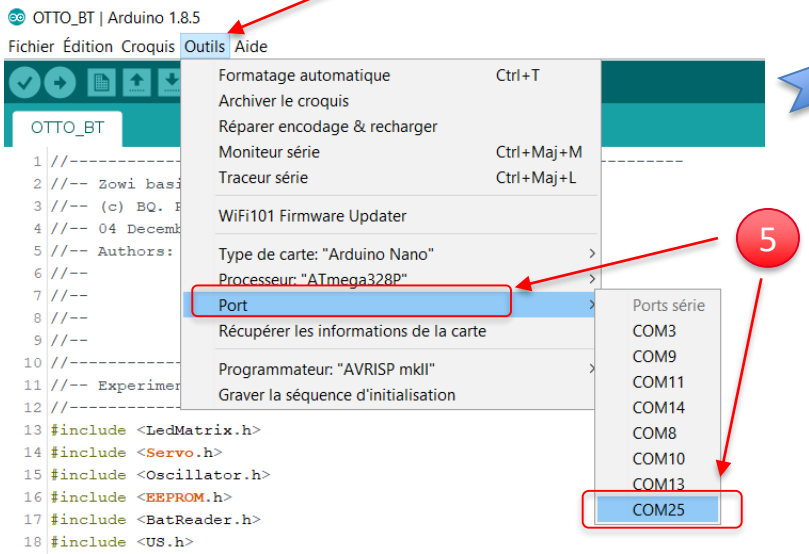
e

Téléchargement du programme BT.ino

c Sélectionner dans le menu **Outils**
- Type de carte **Arduino Nano**
- Processeur **ATATmega328P**

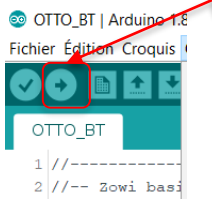


d puis toujours dans le menu **Outils,** sélectionner
- Port : **COM#** sur lequel est connecté le robot Otto

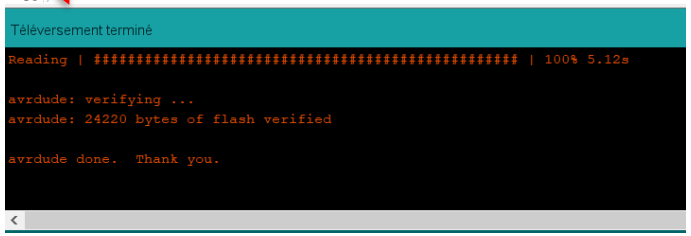


e Menu **Fichier** → **Ouvrir** ou cliquer le bouton - aller dans le dossier « **scripts INO** » et charger le script « **OTTO_BT** »

f puis cliquer le bouton  pour télécharger le script sur la carte Arduino du robot puis …



… attendre l'affichage du message « téléversement terminé »

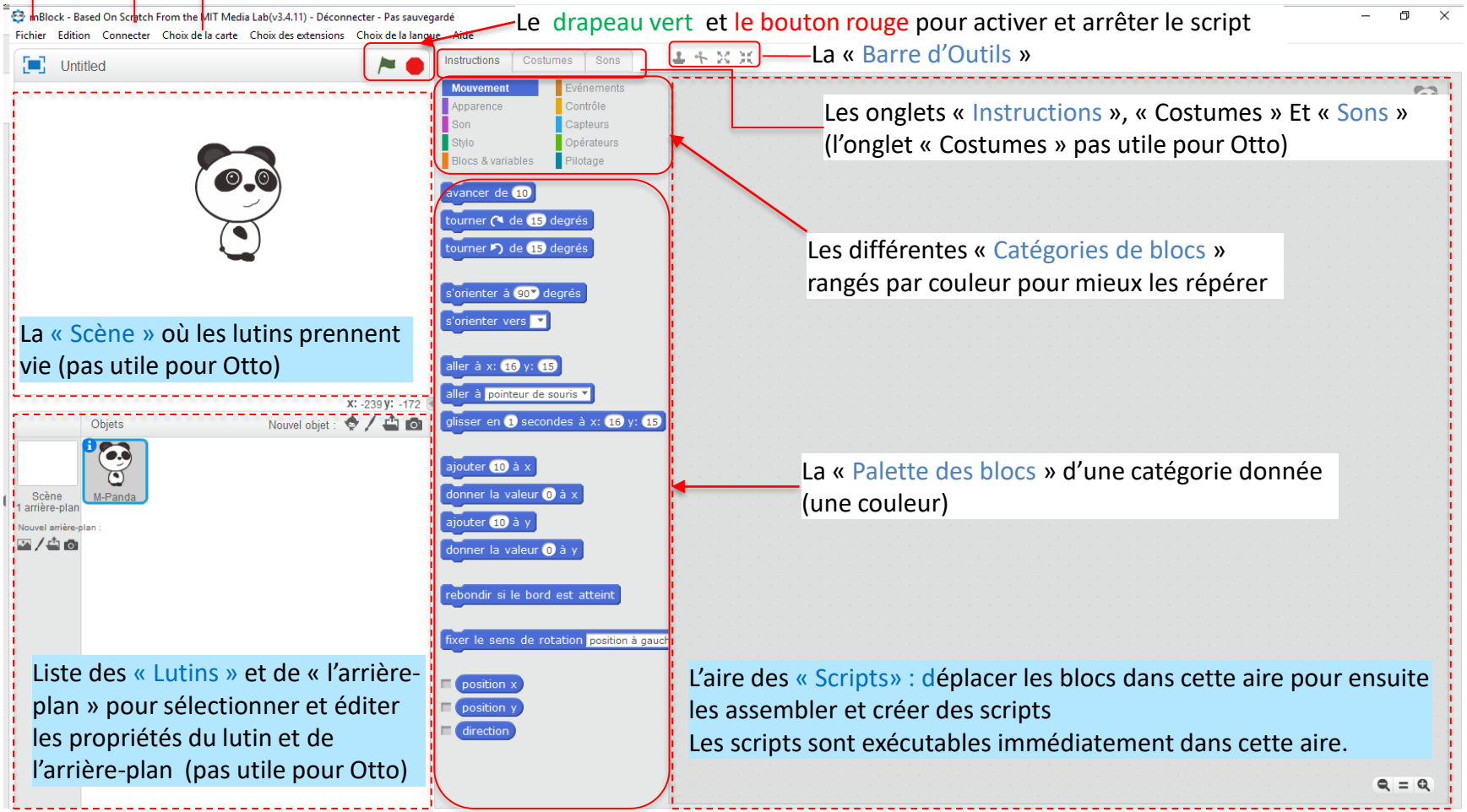


Retour à mBlock : L'interface Scratch / mBlock

Menu « Fichier » pour charger un programme existant ou enregistrer un nouveau programme

Menu « Connecter » pour la connexion au robot

Menu « Choix de la carte » → sélectionner « **Arduino Nano (mega328)** »



Le drapeau vert et le bouton rouge pour activer et arrêter le script

La « Barre d'Outils »

Les onglets « Instructions », « Costumes » Et « Sons »
(l'onglet « Costumes » pas utile pour Otto)

Les différentes « Catégories de blocs »
rangés par couleur pour mieux les répérer

La « Palette des blocs » d'une catégorie donnée
(une couleur)

La « Scène » où les lutins prennent vie (pas utile pour Otto)

Liste des « Lutins » et de « l'arrière-plan » pour sélectionner et éditer les propriétés du lutin et de l'arrière-plan (pas utile pour Otto)

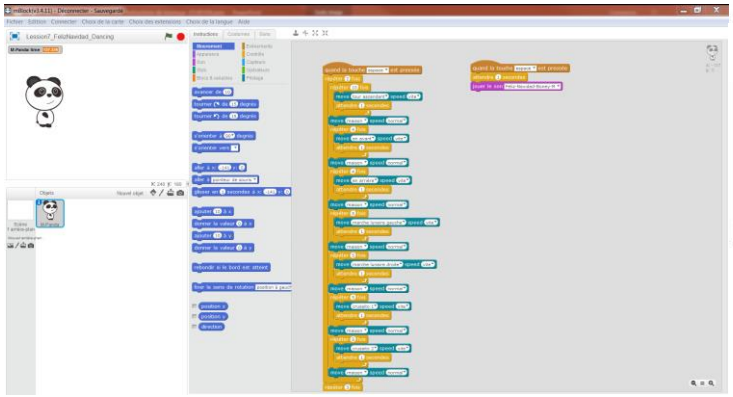
L'aire des « Scripts» : déplacer les blocs dans cette aire pour ensuite les assembler et créer des scripts
Les scripts sont exécutables immédiatement dans cette aire.

mBlock : Mode Scratch et mode Arduino

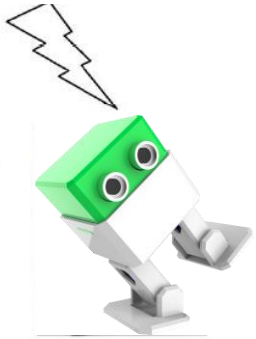
mBlock a deux modes de programmation : le mode **Scratch** et le mode **Arduino**

1- le mode Scratch : Le script s'exécute directement sur l'ordinateur et transfère les commandes liées au robot au robot via une liaison permanente filaire (USB) ou sans fil (Bluetooth). Le robot et l'ordinateur doivent toujours être connectés.

Avantages: La programmation est effectuée « en direct » et interagit immédiatement avec le robot.

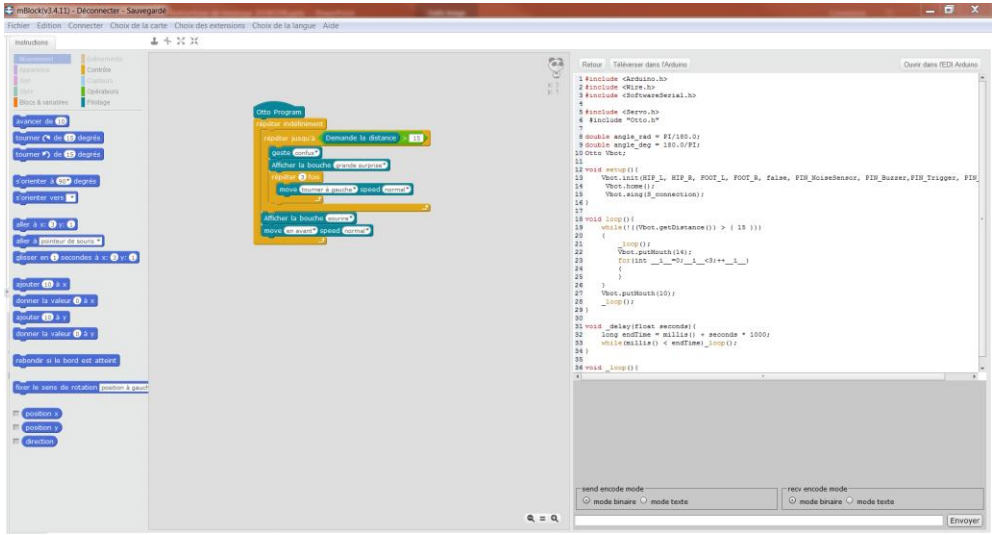


Envoi des commandes liées au robot la liaison USB



2- le mode ARDUINO : Le script est compilé en code Arduino et téléchargé dans le robot, de sorte que le robot peut fonctionner indépendamment sans être connecté de manière permanente à l'ordinateur. en continu.

Attention : Après avoir téléchargé le programme, le mode Scratch ne fonctionnera plus avec le robot jusqu'à ce que vous reveniez au mode initial



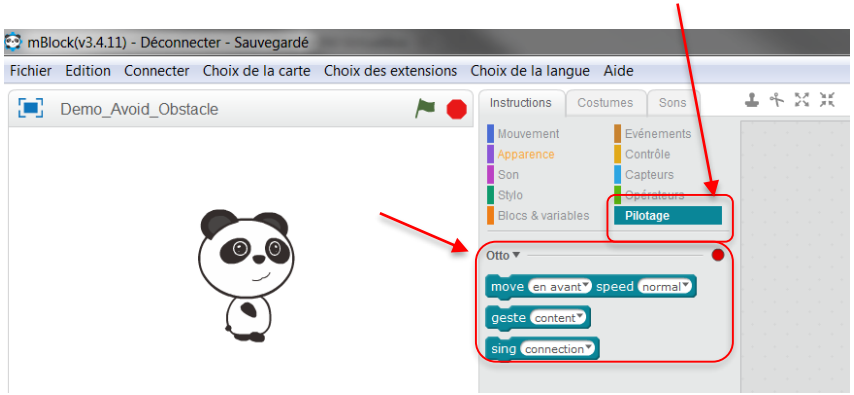
Programmer le robot OTTO « en direct » (Mode Scratch)

Pré-requis : activer l'extension « otto » → Menu Choix des extension

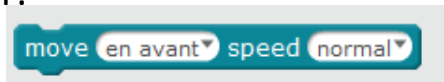
Pour travailler avec le robot Otto;, les principaux types de blocs qui seront utilisés sont

- Les blocs **Evénements**, les blocs **Contrôle**, les blocs **Pilotage**, et les blocs **Son**.

Les blocs spécifiques définis dans **Pilotage** pour le robot Otto sont :

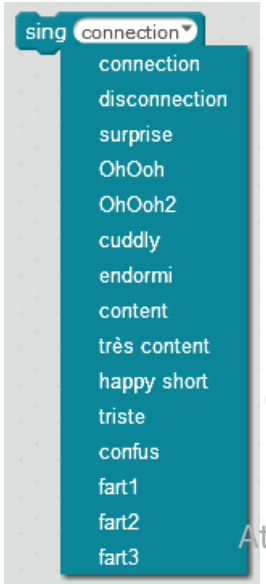


1. **le bloc de mouvement**
   avec 2 propriétés à choisir:
   - type de mouvement
   - vitesse de déplacement / « speed »



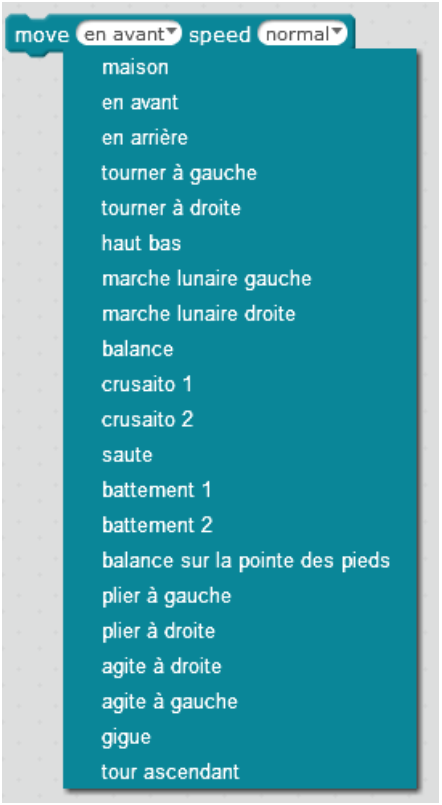
→ *Voir liste détaillée page suivante*

2. **Les ordres comportementaux et émotionnels**
   (une combinaison de mouvements et de sons)
   avec un attribut « geste »
   à choisir.



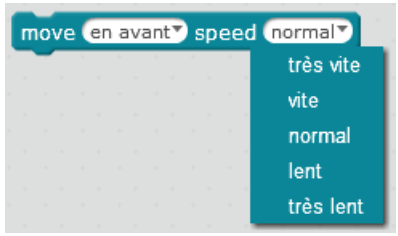
3. **Le bloc jouer le son** avec un attribut « sing » qui exprime un état « émotionnel »

Programmer le robot OTTO avec le bloc mouvement



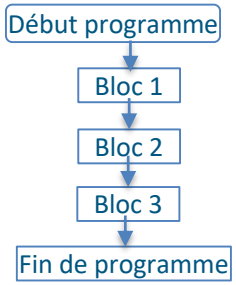
Les propriétés « move » :

maison : état de repos, articulations du robot en état de repos
en avant : avancer
en arrière : reculer
tourner à gauche / tournez à droite
haut bas: effectue un mouvement haut et bas
marche lunaire gauche / droite : deplacement façon moonwalker / Michael Jackson vers la gauche ou vers la droite
balance : balancement jambe gauche – jambe droite
crusalto 1 : « crusalto » mode « espagnol » ki chiki »
crusalto 2: « crusalto » mode « chiki chiki »
saute: se met sur la pointe des pieds
battement 1 / 2: agitation « joyeuse »
balance sur la pointe des pieds
Plier à gauche / à droite : incliner vers la gauche ou vers la droite
agite à droite / à gauche : agite le pied droit ou gauche
gigue : tourne sur les deux pieds style hip-hop
tour ascendant : faire pivoter les jambes pour se déplacer



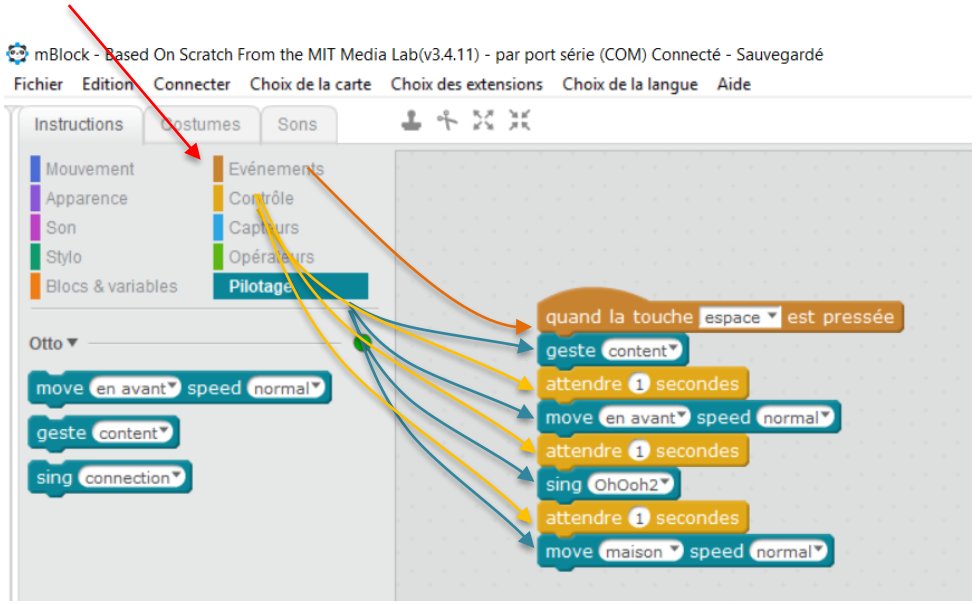
La propriété « speed » / vitesse

Programmer le robot OTTO « en direct » (mode scratch)



Structurer le programme

- le script démarre par l'exécution du premier bloc puis en exécutant en séquence (séquentiellement) les blocs suivants.
- Le diagramme ci-contre décrit une structure de programmation hiérarchique typique



Essayer avec un programme simple

- Bloc « Quand la touche espace est pressé » qui appartient à la catégorie « Evènements » pour démarrer
- Bloc « Geste » qui appartient à la catégorie « Pilotage »
- Bloc « attendre 1 seconde » qui appartient à la catégorie « Contrôle »
- … etc …
- Et enfin bloc « move maison » qui appartient à la catégorie « Pilotage » pour termine le programme

→ il est conseillé de terminer systématiquement les scripts Otto par ce bloc « move maison »

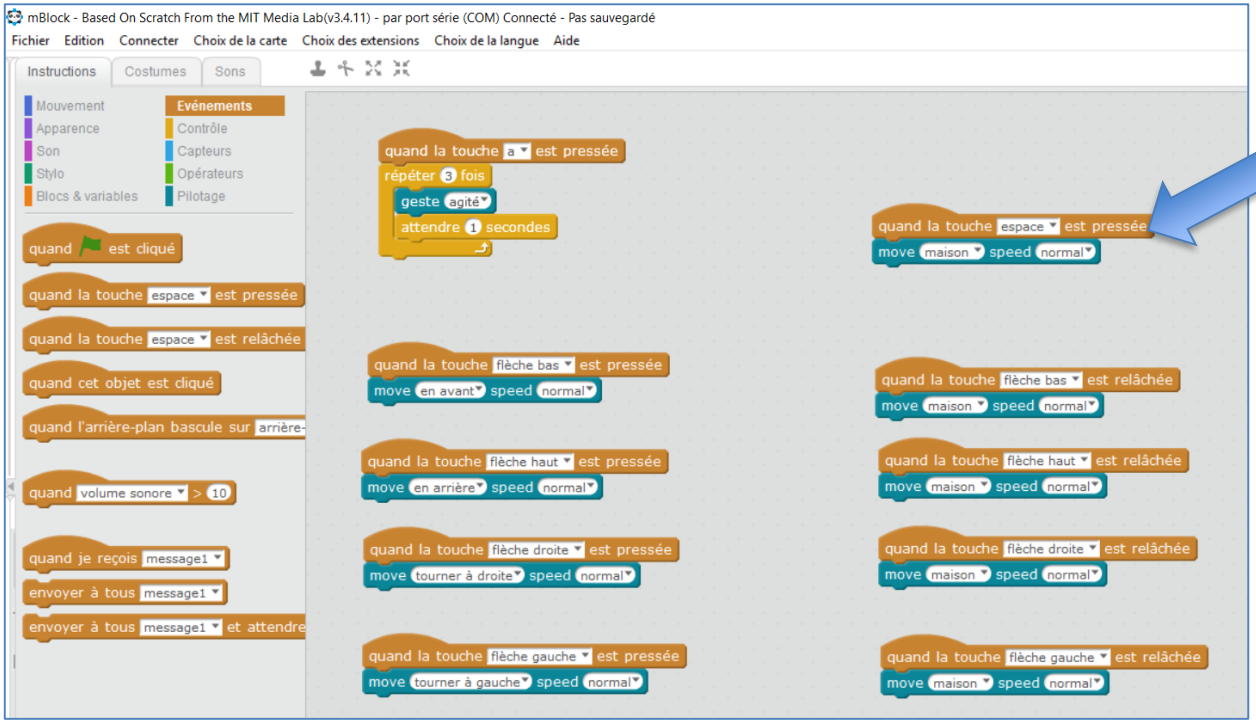
Utilisation des blocs « Evènements »

Exercice 1: développez un script pour le robot Otto exécute l'action sélectionnée lorsque :

- vous cliquez sur le bouton Exécuter 

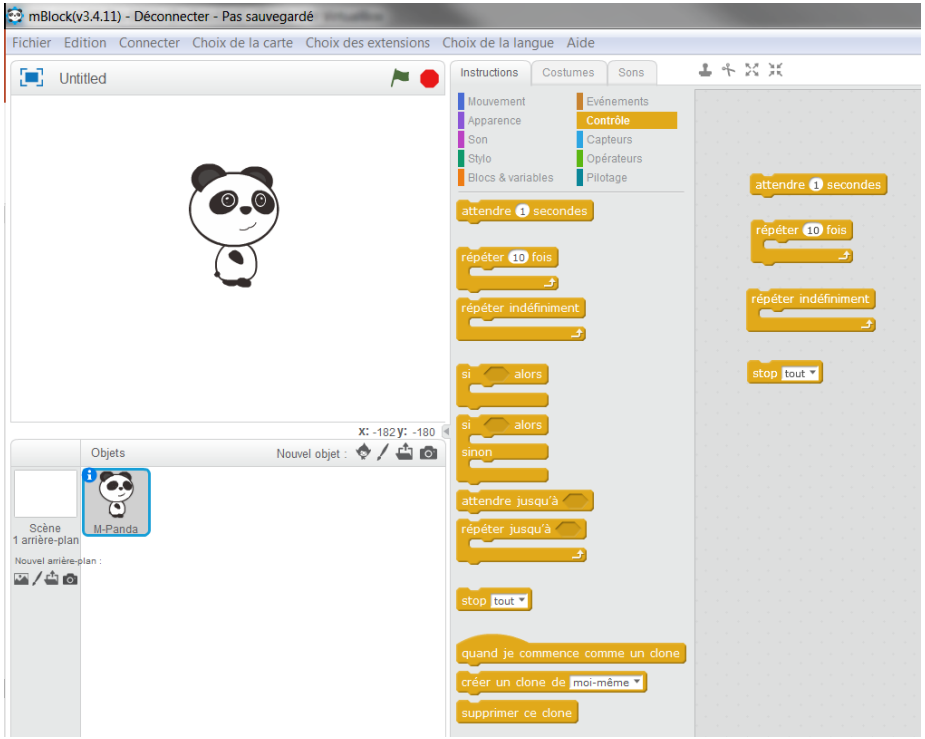
- Qu'une touche pressée 

- Ou qu'une touche est relachée: 

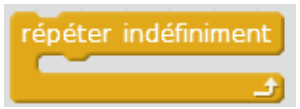


Exemple de programme qui gère les déplacements du robot avec les touches flèche droite, flèche gauche, flèche vers le haut et flèche vers le bas

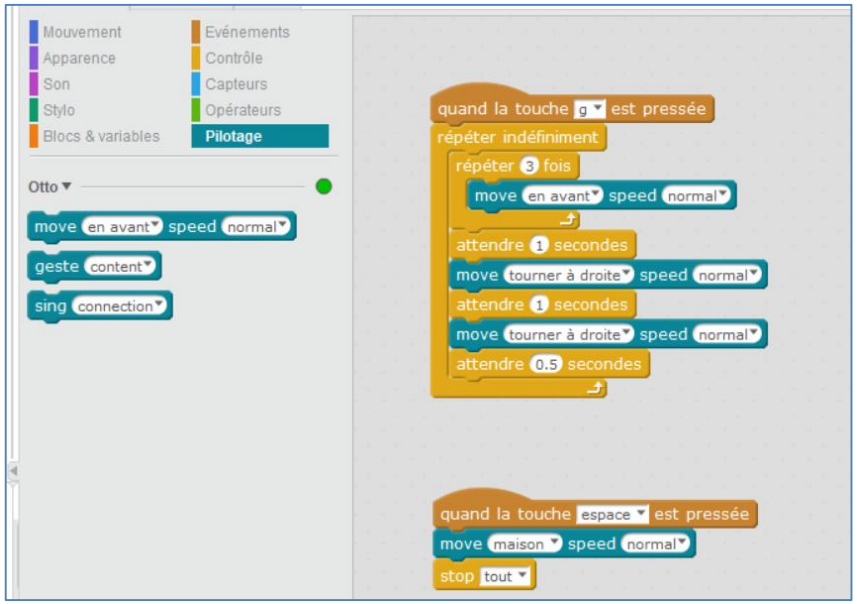
Utilisation des blocs « Contrôle »



Exercice 2: développez un script pour le robot Otto qui utilise les contrôles suivants:



Exemple de programme qui répète indéfiniment une séquence d'instructions dès que la touche « G » est cliquée et qui est stoppé par l'appui de la touche Espace



Utilisation du SON : ajout de pistes dans mBlock

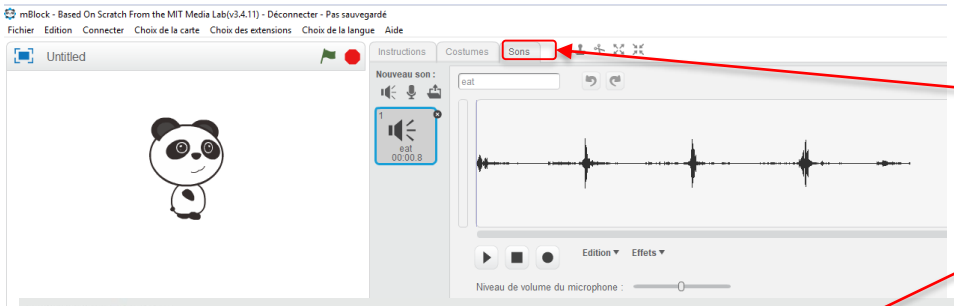
Configurer les pistes dans mBlock

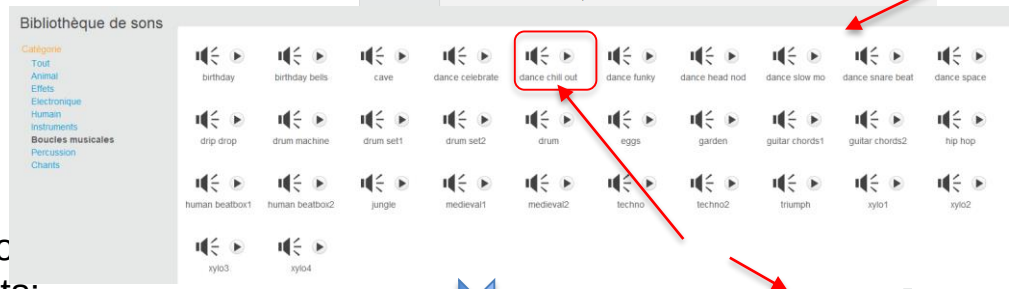
- cliquer l'onglet **Sons**
- Puis cliquer un des trois boutons suivants

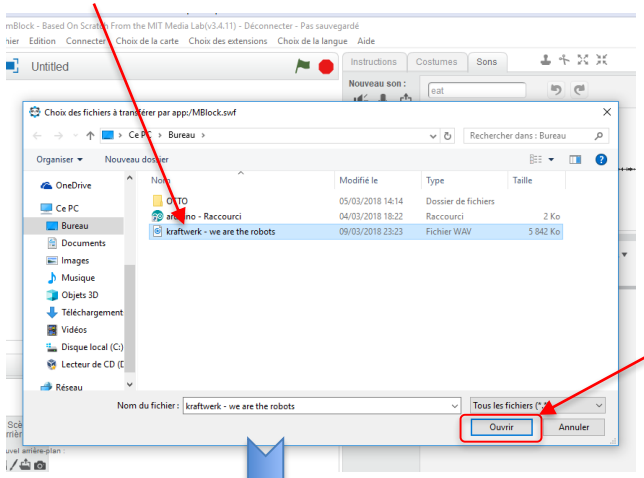
Choisir un son dans la bibliothèque

Enregistrer un nouveau son

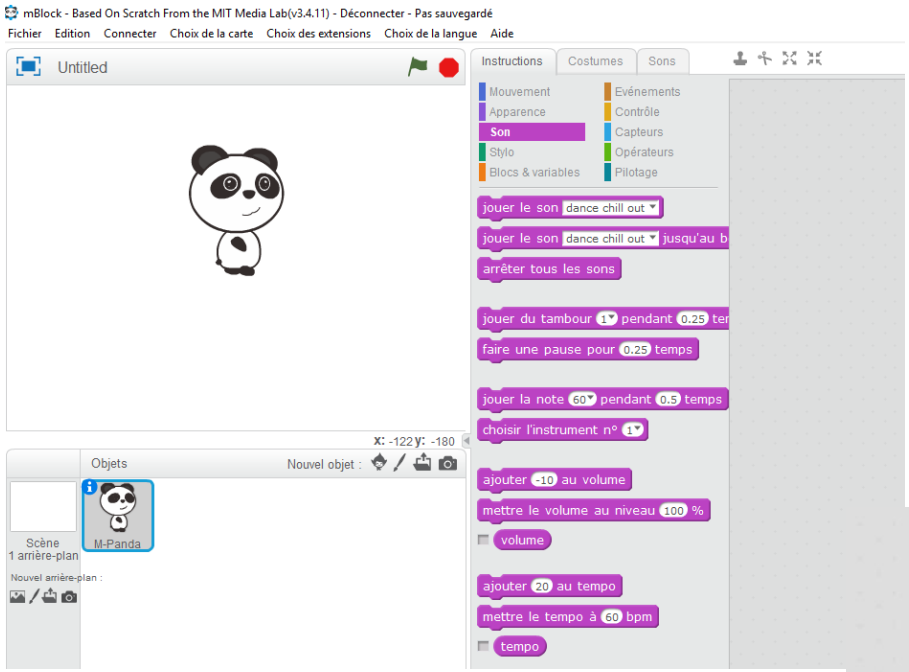
Importer un son depuis un fichier



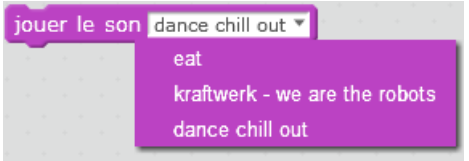




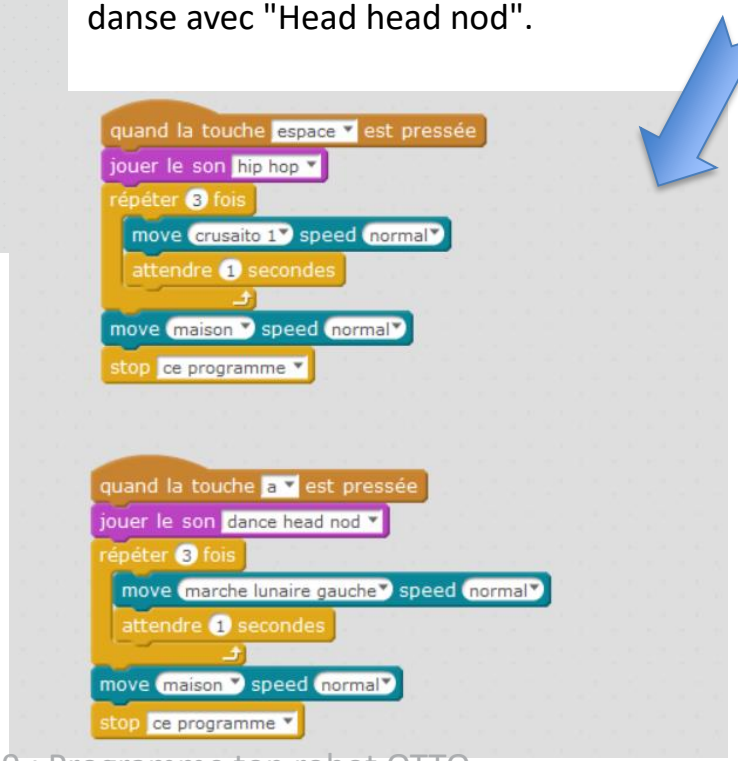
Utilisation du SON : ajout de pistes dans mBlock



Exercice 3: développez un script pour le robot Otto utilise le bloc



<u>Par exemple</u> : un programme de sorte que lorsque vous appuyez sur la touche "Espace", le robot danse sur le HipHop. Lorsque vous appuyez sur "a", le robot danse avec "Head head nod".



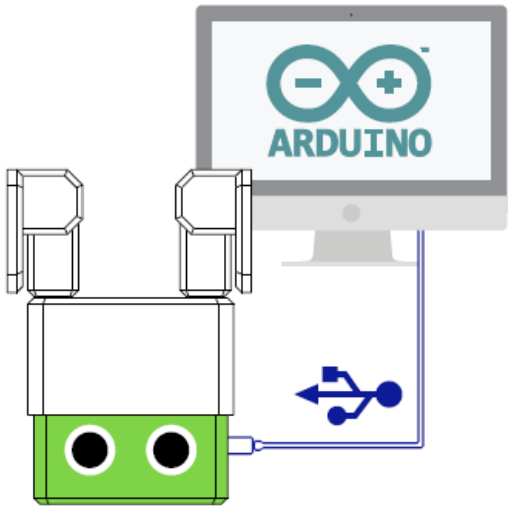
Programmer OTTO avec l'IDE Arduino

Lancement de l'IDE Arduino : connexion du robot Otto

Voir en annexe pour téléchargement / installation / configuration de l'IDE arduino dans le cas où sa mise en place n'aurait pas été effectuée à partir de la clé USB fournie par le fablab

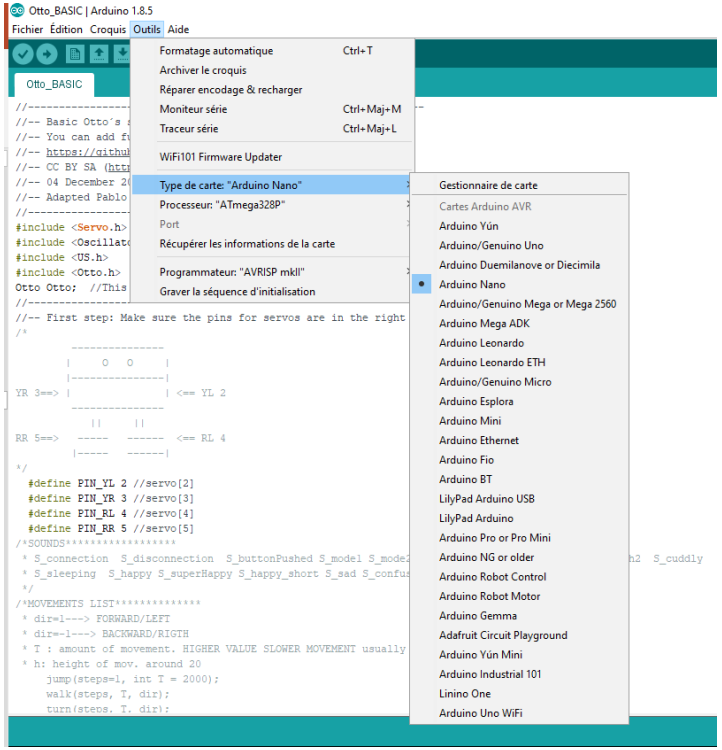
a Ouvrir le programme
c:/arduino-1.8.5_OTTO/arduino.exe

b Connecter le robot Otto sur un port USB



c Sélectionner dans le menu **Outils**

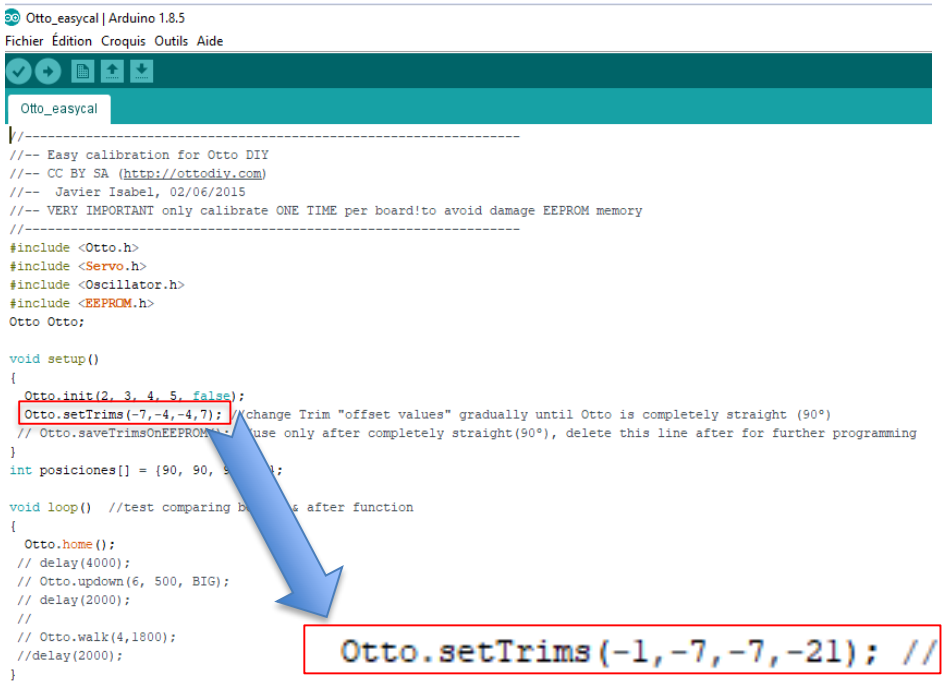
- Type de carte **Arduino Nano**
- Processeur **ATATmega328P**
- Port : **COM#** sur lequel est connecté le robot Otto



Calibration des servos et enregistrement

Attention : Cette étape n'est pas utile si la calibration du robot a déjà été effectuée sous mBlock

- charger le script **Otto_easycal.ino** du 5/2/2018



Reporter / remplacer dans la ligne 16 les quatre valeurs numériques « (-7,-4,-4,-7) » par les valeurs TRIM relevées dans le tableau page 15 du tutoriel de l'atelier Otto du 24 février 2018

AVANT:

Otto.setTrims(-7,-4,-4,7);

- remplacer par TRIM ligne broche 2
- remplacer par TRIM ligne broche 3
- remplacer par TRIM ligne broche 4
- remplacer par TRIM ligne broche 5

Exemple de valeurs relevées dans le tableau de la page page 15 du tutoriel de l'atelier Otto du 24 février 2018

Servo	code	Servo n°	Broche	Valeur TRIM
Left hip – *hanche/jambe gauche*	LH/YL/LL	0	2	**-1**
Right hip – *hanche/jambe droite*	RH/YR/RL	1	3	**-7**
Left ankle - *cheville/pied gauche*	LA/RL/LF	2	4	**-7**
Right ankle - *cheville / pied droite*	RA/RR/RF	3	5	**-21**

APRES

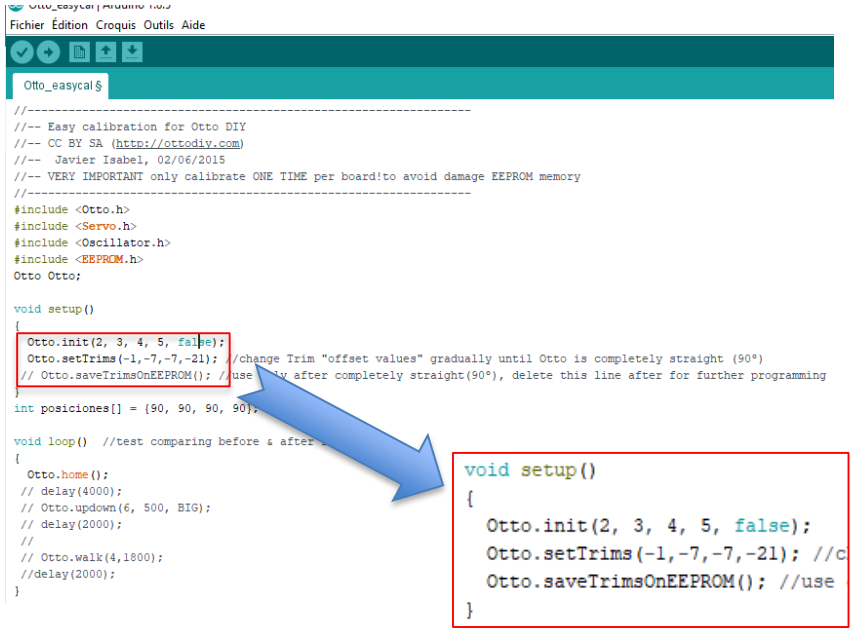
Otto.setTrims(-1,-7,-7,-21);

Calibration des servos et enregistrement (suite & fin)

e Vérifier / compiler le script

f puis télécharger le script

h



i et télécharger à nouveau le script

j et enfin, enregistrer le script modifié

g ***Vérifier l'orientation à 90° des jambes et pieds du robot***

- *Le déconnecter*
- *Couper l'alimentation via l'interrupteur*
- *Puis le remettre sous tension*

Si l'orientation des jambes et pieds est satisfaisante :

- ***Reconnecter le robot via le port USB***
- ***Et effectuer la modification suivante dans le script***

Supprimer les caractères « // » en début de ligne 17

AVANT:

// Otto.saveTrimsOnEEPROM();

APRES

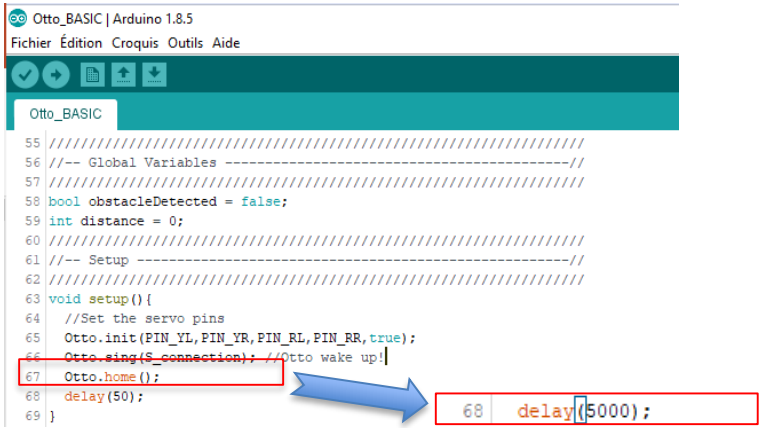
Otto.saveTrimsOnEEPROM();

En résumé : charger un programme au travers l'IDE Arduino

a charger le script

Par exemple : charger le script **Otto_BASIC.ino**

b effectuer éventuellement vos modifications



Par exemple : changer ligne 68 la durée de la pause avant que le robot Otto commence à se déplacer

AVANT:

delay(50); = 50 millisecondes (5/100ème secondes)

APRES:

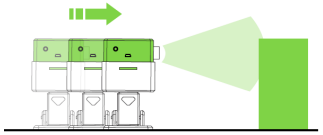
delay(5000); = 5000 millisecondes (5 secondes)

c Vérifier / compiler le script

Ou menu Croquis → Vérifier Compiler (**Ctrl-R**)

d Télécharger le script

Ou menu Croquis → Téléverser (**Ctril-U**)

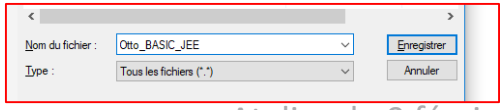


Remarque : l'exécution du script va démarrer dès la fin du téléchargement

e enregistrer le script si vous avez effectué des modifications à l'étape **b**

Ou menu Fichier → Enregistrer (**Ctril-S**)
ou Enregistrer sous (**Ctrl-Maj-S**)

Par exemple : Ctrl-Maj-S et saisir dans nom de fichier « Otto_BASIC_*XY* » *(XY = vos initiales)*



Structure des scripts Otto / Arduino : exemple

1- début du script « Otto_avoid_TRIMS,ino »

```
//-----------------------------------------------------
//-- CC BY SA (http://ottodiy.com)
//-- 20 January 2017
//-- TRIMS servo calibration values added
//-- double check your US (Ultra sound) module some have TRIG and ECHO swapped!
//-----------------------------------------------------------------
//-- Otto will avoid obstacles with this code!
//-----------------------------------------------------------------
#include <Servo.h>
#include <Oscillator.h>
#include <US.h>
#include <Otto.h>

/*
                 ---------------
                |     o   o     |
                |---------------|
RIGHT LEG 3==>  |               | <== LEFT LEG 2
                 ---------------
                    ||     ||
RIGHT FOOT 5==>    -----   ------  <== LEFT FOOT 4
                   |-----   ------|
*/

#define PIN_LEFT_LEG   2
#define PIN_RIGHT_LEG  3
#define PIN_LEFT_FOOT  4
#define PIN_RIGHT_FOOT 5


#define TRIM_LEFT_LEG   0
#define TRIM_RIGHT_LEG  0
#define TRIM_LEFT_FOOT  0
#define TRIM_RIGHT_FOOT 0

Otto Otto;  //This is Otto!

void setup()
{
  Serial.begin(9600);
  Serial.println("hello");

  //Set the servo pins
  Otto.init(PIN_LEFT_LEG,PIN_RIGHT_LEG,PIN_LEFT_FOOT,PIN_RIGHT_FOOT,true);
  Otto.setTrims(TRIM_LEFT_LEG,TRIM_RIGHT_LEG, TRIM_LEFT_FOOT, TRIM_RIGHT_FOOT);

  Otto.sing(S_connection); //Otto wake
  Otto.home();

}
```

commentaires

Librairies standard (servo..h) et spécifiques Otto

commentaires

Attachement des servo

Données de calibrage (non prises en compte – voir + bas)

ensemble d'instructions d'Initialisation
Exécutées une seule fois au démarrage

Appel de la fonction home = lecture des données de calibrage mémorisées dans le robot, initialisation positions jambes et pieds, puis mise au repos des servos

2- Suite du script « Otto avoid TRIMS,ino »

```
//////////////////////////////////////////////////////////////////
void loop()
{
  bool obstacleDetected = false;

  obstacleDetected = obstacleDetector();

  if(obstacleDetected)
  {
      Otto.sing(S_surprise);
      Otto.walk(7,1000,BACKWARD);
      delay(1000);
      Otto.sing(S_happy);
      Otto.turn(10,1000,RIGHT);
      delay(2000);
  }
  else
  {
      Otto.walk(1,1000,FORWARD);
  }
}

//////////////////////////////////////////////////////////////////
//-- Function to read distance sensor & to actualize obstacleDetected variable
bool obstacleDetector()
{
  int distance = Otto.getDistance();

  Serial.println(distance);

  if(distance<15)
  {
     return true;
  }
  else
  {
     return false;
  }

}
```

Boucle principale
Ensemble des instructions exécutées continuellement jusqu'à l'arrêt / mise hors tension du robot

Actions effectuées en cas de détection d'obstacle

Appel fonction détection obstacle

Appel des fonctions OTTO « Jouer le son » et « Mouvement » → voir documentation pages suivantes

Action effectuée si pas d'obstacle

Fonction détection obstacle
Évalue distance et retourne détection ou non d'un obstacle

Otto / Arduino : fonctions « mouvement »

Fonction « mouvement »

```
Otto.walk(2,1300,-1);
```

- `Otto.walk` : fonction
- `(2,1300,-1)` : (nombre, durée, direction)
  - `2` : nombre de mouvements, pas, gestes, …
  - `1300` : Durée en millisecondes (1000ème de secondes) Plus la valeur est grande → plus le mouvement est lent – valeur conseillée = 1000 (de 600 à 1400)
  - `-1` : Direction : 1 vers l'avant, à gauche selon type de mouvement ; -1 vers l'arrière, à droite selon type de mouvement

→ Pour changer le type de mouvement, remplacer **Otto.walk** par un des mouvements suivants

`Otto.walk(1,1000,1);`	`Otto.walk(1,1000,-1);`	**walk :** avancer (1), reculer (-1)
`Otto.turn(3,1000,1);`	`Otto.turn(3,1000,-1);`	**turn :** tourner à gauche (1) tournez à droite (-1)
`Otto.bend(2,1000,1);`	`Otto.bend(2,500,-1);`	**bend :** incliner vers la gauche ou vers la droite – valeur conseillée : (1,2000,1) ou -1
`Otto.shakeLeg(1,1000,1);`	`Otto.shakeLeg(1,500,-1);`	**shakeLeg:** balancement jambe gauche – jambe droite

`fonction (nombre, période, taille, direction)`

« moveSize = taille / ampleur du mouvement – valeur conseillée : autour de 20

`Otto.moonwalker(1,1000,moveSize,1);` `Otto.moonwalker(1,1000,30,1);`	**moonwalker :** deplacement façon Michael Jackson vers la gauche ou vers la droite
`Otto.crusaito(1,1000,moveSize,1);`	**crusalto :** « crusalto » mode « espagnol » ou « chiki chiki »
`Otto.flapping(1,1000,moveSize,1);`	**flapping :** agite le pied droit ou gauche
`Otto.swing(1,1000,moveSize);`	
`Otto.updown(1,1000,moveSize);`	**updown:** effectue un mouvement haut et bas
`Otto.tiptoeSwing(1,1000,moveSize);`	**tiptoeSwing :** faire pivoter les jambes pour se déplacer
`Otto.jitter(1,1000,moveSize);`	**jitter :** tourne sur les deux pieds style hip-hop
`Otto.ascendingTurn(1,1000,moveSize);`	**ascendingTurn :** agitation « joyeuse »
`Otto.jump(1,1000);`	**jump :** se met sur la pointe des pieds – valeur conseillée : (1,2000)

Otto / Arduino : fonctions « jouer le son » & « geste »

Fonction « jouer le son »

```
Otto.sing(S_surprise);
```

fonction (le son à jouer)

→ Pour changer le son, remplacer **S_surprise** par un des sons suivants

(S_surprise);	(S_OhOoh);	(S_OhOoh2);
(S_cuddly);	(S_sleeping);	
(S_happy);	(S_superHappy);	(S_happy_short);
(S_sad);	(S_confused);	
(S_fart1);	(S_fart2);	(S_fart3);
(S_mode1);	(S_mode2);	(S_mode3);
(S_connection);	(S_disconnection);	(S_buttonPushed);

Fonction « Geste »

```
Otto.playGesture(OttoFretful);
```

fonction (le geste à effectuer)

→ Pour changer le geste, remplacer **OttoFretful** par un des gestes suivants

(OttoSuperHappy);	(OttoSad);
(OttoSleeping);	(OttoFart);
(OttoConfused);	(OttoFretful);
(OttoLove);	(OttoAngry);
(OttoMagic);	(OttoWave);
(OttoVictory);	(OttoFail);

Structure des scripts Otto / Arduino : autre exemple

1- script « Otto_BASIC.ino »

```
/*SOUNDS*****************
 * S_connection  S_disconnection  S_buttonPushed S_mode1 S_mode2 S_mode3 S_surprise S_OhOoh  S_OhOoh2  S_cuddly
 * S_sleeping  S_happy S_superHappy S_happy_short S_sad S_confused S_fart1 S_fart2  S_fart3
 */
/*MOVEMENTS LIST**************
 * dir=1---> FORWARD/LEFT
 * dir=-1---> BACKWARD/RIGTH
 * T : amount of movement. HIGHER VALUE SLOWER MOVEMENT usually 1000 (from 600 to 1400)
 * h: height of mov. around 20
    jump(steps=1, int T = 2000);
    walk(steps, T, dir);
    turn(steps, T, dir);
    bend (steps, T, dir); //usually steps =1, T=2000
    shakeLeg (steps, T, dir);
    updown(steps, T, HEIGHT);
    swing(steps, T, HEIGHT);
    tiptoeSwing(steps, T, HEIGHT);
    jitter(steps, T, HEIGHT); (small T)
    ascendingTurn(steps, T, HEIGHT);
    moonwalker(steps, T, HEIGHT,dir);
    crusaito(steps, T, HEIGHT,dir);
    flapping(steps, T, HEIGHT,dir);
/*GESTURES LIST***************
OttoHappy OttoSuperHappy  OttoSad   OttoSleeping  OttoFart  OttoConfused OttoLove  OttoAngry
OttoFretful OttoMagic  OttoWave  OttoVictory  OttoFail*/
```

Commentaires documentant toutes les fonctions utilisables

```
//////////////////////////////////////////////////////////////////
void setup(){
  //Set the servo pins
  Otto.init(PIN_YL,PIN_YR,PIN_RL,PIN_RR,true);
  Otto.sing(S_connection); //Otto wake up!
  Otto.home();
  delay(50);
}
//////////////////////////////////////////////////////////////////
//-- Principal Loop -------------------------------------------//
//--Uncomment lines or add you own-------------------------------//
//////////////////////////////////////////////////////////////////
void loop() {
          //OBSTACLE MODE ON!!!!
          obstacleMode();
          /*Otto.walk(2,1000,1); //2 steps FORWARD
          Otto.walk(2,1000,-1); //2 steps BACKSWARD
          Otto.playGesture(OttoFretful);
          Otto.home();
          Otto.sing(S_sleeping);
          delay(1000);
          Otto.turn(2,1000,1);//2 steps turning RIGHT
          delay(50);
          Otto.turn(2,1000,-1);//2 steps turning RIGHT
          delay(50);
          Otto.moonwalker(3, 1000, 25, 1);
          Otto.home();
          Otto.bend (1, 2000, 1);
          Otto.home();
          Otto.ascendingTurn(1, 2000, 22);
          Otto.home();
          Otto.updown(1, 2000, 22);
          Otto.home();
          */
}
//////////////////////////////////////////////////////////////////
```

suppression des caractères « /* » (début de zone commentaire)

suppression des caractères « */ » (fin de zone commentaire)

Otto_BASIC_JEE §

```
  Otto.home();
//JEE  delay(50);
  delay(5000);

}
//////////////////////////////////////////////////////////////////
//-- Principal Loop -------------------------------------------//
//--Uncomment lines or add you own-------------------------------//
//////////////////////////////////////////////////////////////////
void loop() {
          //OBSTACLE MODE ON!!!!
          obstacleMode();
          Otto.walk(2,1000,-1); //2 steps BACKSWARD
          Otto.playGesture(OttoFretful);
          Otto.home();
          Otto.sing(S_sleeping);
          delay(1000);
          Otto.turn(2,1000,1);//2 steps turning RIGHT
          delay(50);
          Otto.turn(2,1000,-1);//2 steps turning RIGHT
          delay(50);
          Otto.moonwalker(3, 1000, 25, 1);
          Otto.home();
          Otto.bend (1, 2000, 1);
          Otto.home();
          Otto.ascendingTurn(1, 2000, 22);
          Otto.home();
          Otto.updown(1, 2000, 22);
          Otto.home();

          delay(3000);
          Otto.sing(S_superHappy);
          Otto.flapping(5,1000,20,1);
          delay(50);
          Otto.playGesture(OttoVictory);
          Otto.home();
}
```

Ajout de ces 6 lignes d'instruction

Annexes

Installation de l'IDE Arduino sur le PC

a download Arduino for FREE to your computer

Aller sur le site www.arduino.cc et télécharger la version 1.8.5 du programme Arduino IDE

⚠ choose the appropriate Operating System

Choisir la version appropriée en fonction du système d'exploitation installé sur votre PC

b Installer le programme téléchargé sur votre PC

→ Pour Windows, choisir Windows ZIP file for non admin install puis cliquer JUST DOWNLOAD

Installer l'IDE Arduino en mode **« portable »**

→ Sur Windows :

- déplacer le fichier téléchargé « arduino-1.8.5-windows.zip » dans le dossier où vous souhaiter l'installer (par exemple à la racine du disque C:\
- puis clic droit sur le « Extraire tout .. » créera un dossier « **c:\arduino-1.8.5** »
- Créer un dossier **c:\arduino-1.8.5 \portable**
- Envoyer vers le Bureau (créer un raccourci) **c:\arduino-1.8.5\arduino.exe**

Installation de l'IDE Arduino : ajout des librairies Otto

a go to ottodiy.com in the build it! section download and unzip OTTO_DIY_all.zip

Aller sur https://github.com/stembotvn/OttoDIY_Vbot et cliquer Clone or download puis Download ZIP pour télécharger le fichier OttoDIY_Vbot-master.zip

b from the "driver" folder install CH341SER

⚠ choose the appropriate Operating System installation package for your computer.

Pour Windows, aller sur http://www.wch.cn/download/CH341SER_EXE.html ou pour MAC sur http://www.wch.cn/download/CH341SER_MAC_ZIP.html Ou pour Linux sur http://www.wch.cn/download/CH341SER_LINUX_ZIP.html

c copy or move all "libraries" folders to:

C:\Documents\Arduino\libraries\
(or wherever your Arduino library folder is)

Ajouter la librairie suivante manuellement en utilisant le menu "Croquis → inclure une bibliothèque → ajouter la bibliothèque ZIP OttoDIY_Vbot-master.zip

Copier le dossier **ledmatrix** dans **c:\arduino-1.8.5\Portable\ sketchbook\librairies**

d copy or move all "OTTO_" folders to:

C:\Documents\Arduino\
(or wherever your Arduino sketch folder is)

Après installation de la Librairie, aller dans Fichier/Exemples/OttoDIY_Vbot-master et sélectionner le script que vous souhaitez charger sur la carte Arduino Nano