Programmer l'arduino avec mBlock

2ème approche

Ce support a été complété des résultats de l'atelier arduino du 5 février 2017

http://www.planete-sciences.org/iledefrance/index.php/robotique/60-aller-plus-loin-en-robotique

http://learn.makeblock.com/getting-started-programming-with-mblock/?utm_source=software&utm_medium=mblock&utm_campaign=mblocktorumeng

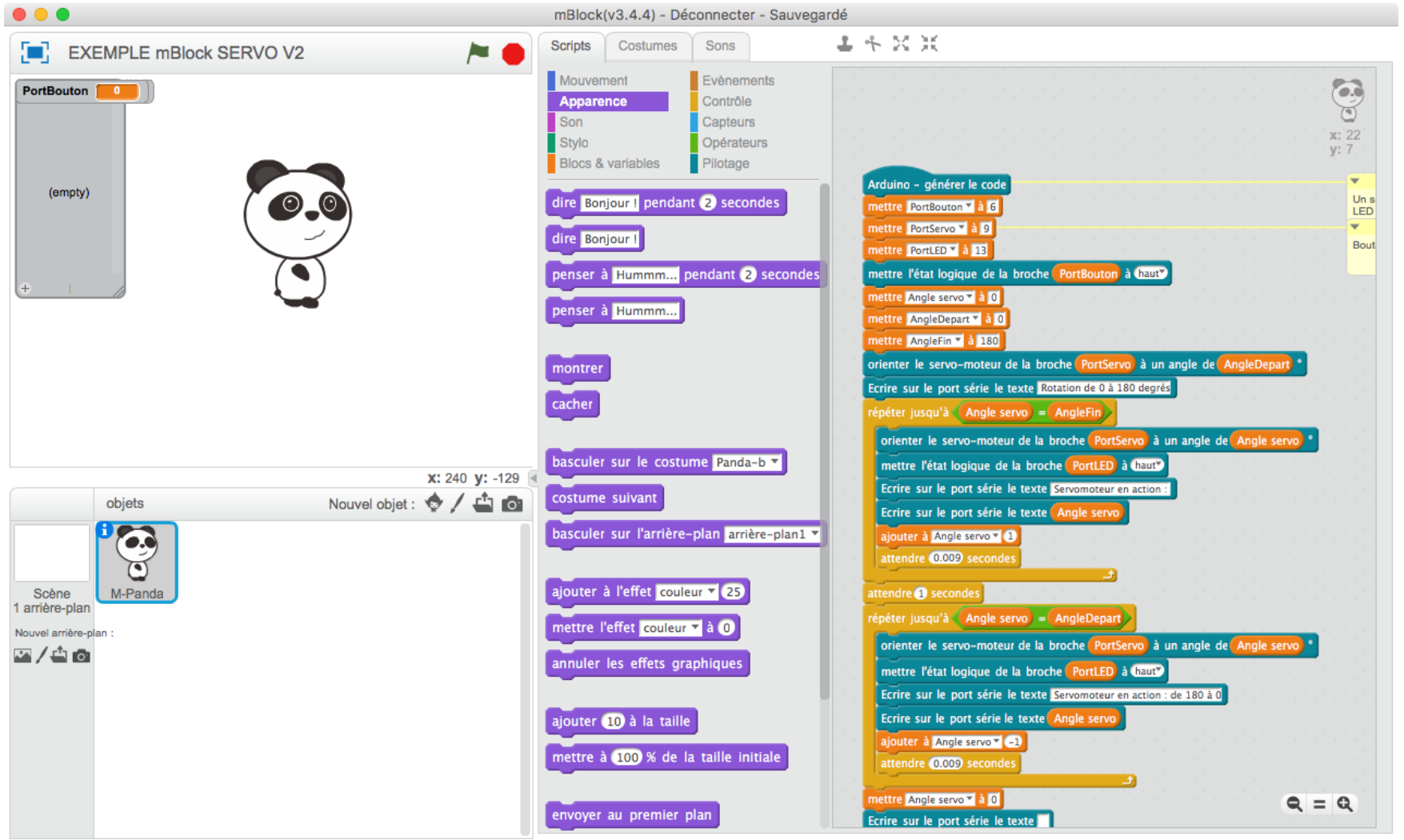
http://download.makeblock.com/mblock/mblock_extension_guide.pdf

Présentation de mBlock

- La programmation visuelle par blocs est un atout majeur dans l'apprentissage de l'algorithmique et du codage d'applications
- Historiquement, le MIT a développé un atelier très pédagogique : Scratch, qui permet de créer des applications et d'interagir avec elles
- Plus tard, avec le développement de l'Arduino, une version adaptée permettant de piloter un microcontrôleur, S4A (Scratch for Arduino) a été mise au point par une équipe de chercheurs espagnols. Toutefois, le contrôle de l'Arduino nécessite une liaison constante avec l'environnement Scratch

- **L'environnement mBlock constitue un nouveau développement qui permet de réaliser des programmes avec Scratch et d'en dériver une variante téléversable dans l'Arduino, qui fonctionne alors de façon autonome**

En mode Scratch : un environnement visuel et ludique avec une convivialité qui n'est plus à démontrer



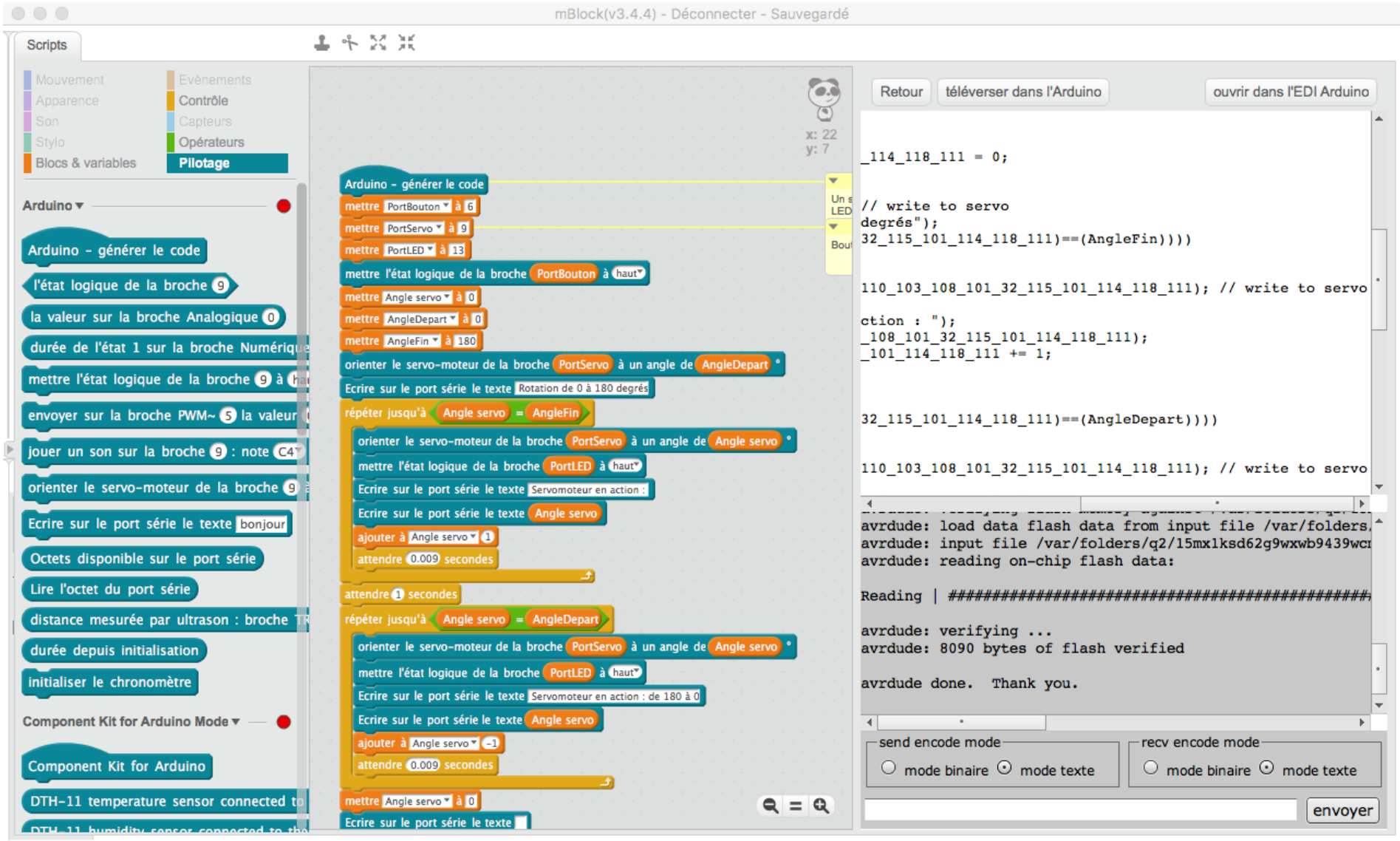
La logique du « glisser-déposer » est la même et des fonctions supplémentaires permettent des interactions et de nombreux effets visuels et sonores.

Il est possible de programmer des jeux et même de participer à des concours ou de mettre en commun ses programmes (scripts) Scratch

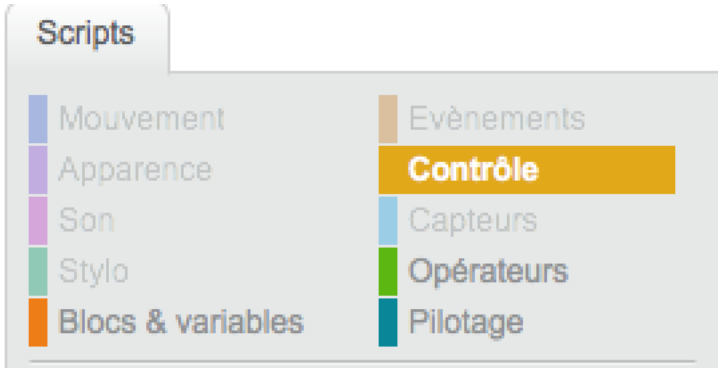
En mode Arduino : un environnement de programmation qui fait bonne figure

La logique est simple : les instructions sont sélectionnées dans le cadre de gauche et combinées par glisser-déposer sur la page centrale. Le code généré pour l'arduino s'affiche dans le cadre de droite.

Il n'y a plus d'erreur de syntaxe et le progammeur se concentre sur son algorithme



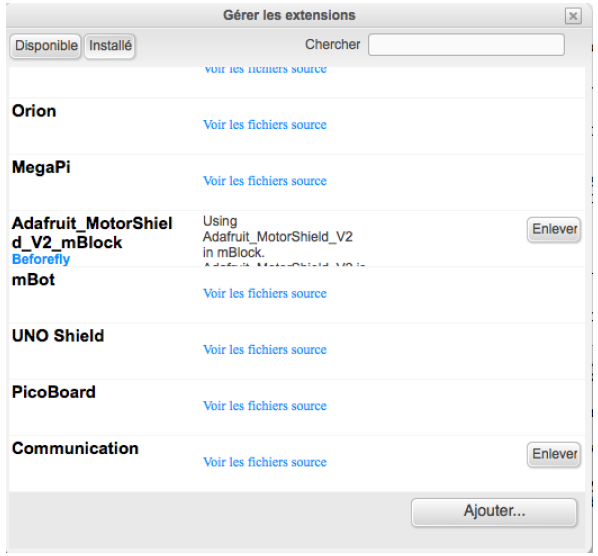
En mode Arduino : la programmation visuelle



4 familles de blocks sont utilisables pour la programmation :

- Blocs et variables : définir des variables, paramètre et procédures
- Contrôle : les structures usuelles de contrôle du code
- Opérateurs : arithmétiques et autres
- Pilotage : gère les interactions avec l'Arduino

Les instructions peuvent être complétées par des extensions correspondant aux primitives de commande des modules mBlot ou des shields arduino ou encore des capteurs et actionneurs personnalisés



Le guide de démarrage de mBlock est disponible à l'adresse
http://learn.makeblock.com/getting-started-programming-with-mblock/?utm_source=software&utm_medium=mblock&utm_campaign=mblocktorumeng

Le guide de développement des extensions est disponible sur le site
http://download.makeblock.com/mblock/mblock_extension_guide.pdf

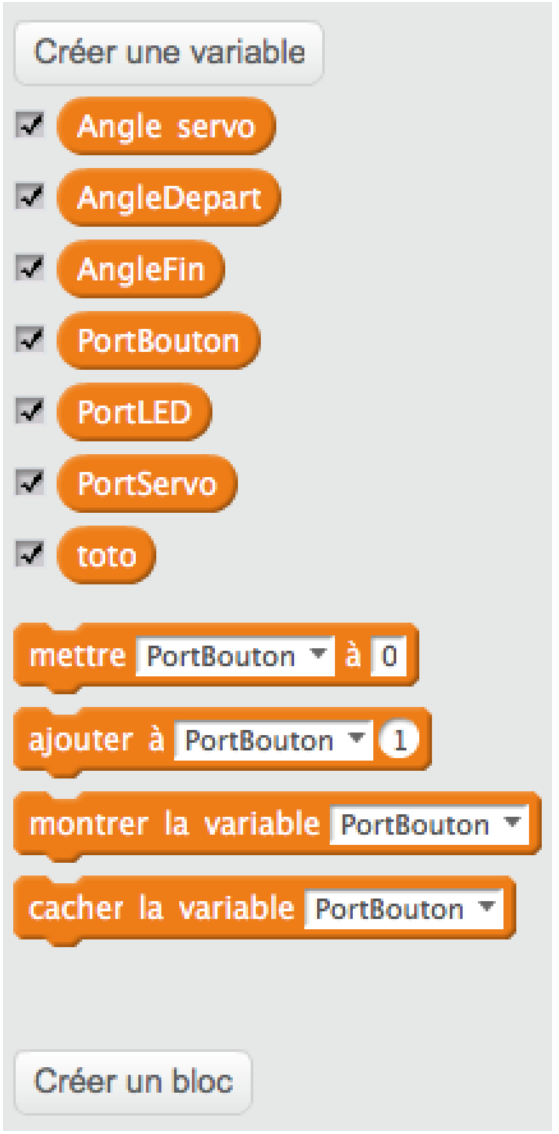
En mode Arduino : les familles de blocks

Les contrôles

Elle comprend les structures de contrôles basiques en algorithmique :

- La répétition jusqu'à ce qu'une condition soit remplie
- l'alternative « si alors sinon »

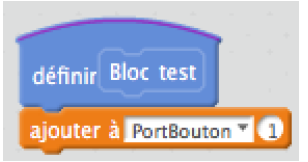
Ainsi que l'équivalent des instructions delay() = attendre n secondes et loop() = répéter indéfiniment

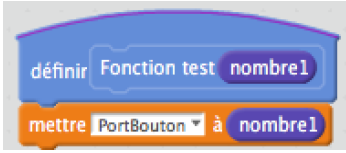


Les blocs et variables

Elle comprend un bouton qui permet de créer une variable et des blocs qui permettre de lui affecter une valeur, de l'incrémenter (ou de la décrémenter par une valeur négative) et deux blocs qui permettent de montrer ou de cacher une variable mais qui ne sont pas utilisables avec l'arduino

Elle comprend un bouton qui permet de créer des blocs en tant que fonction ou de procédure





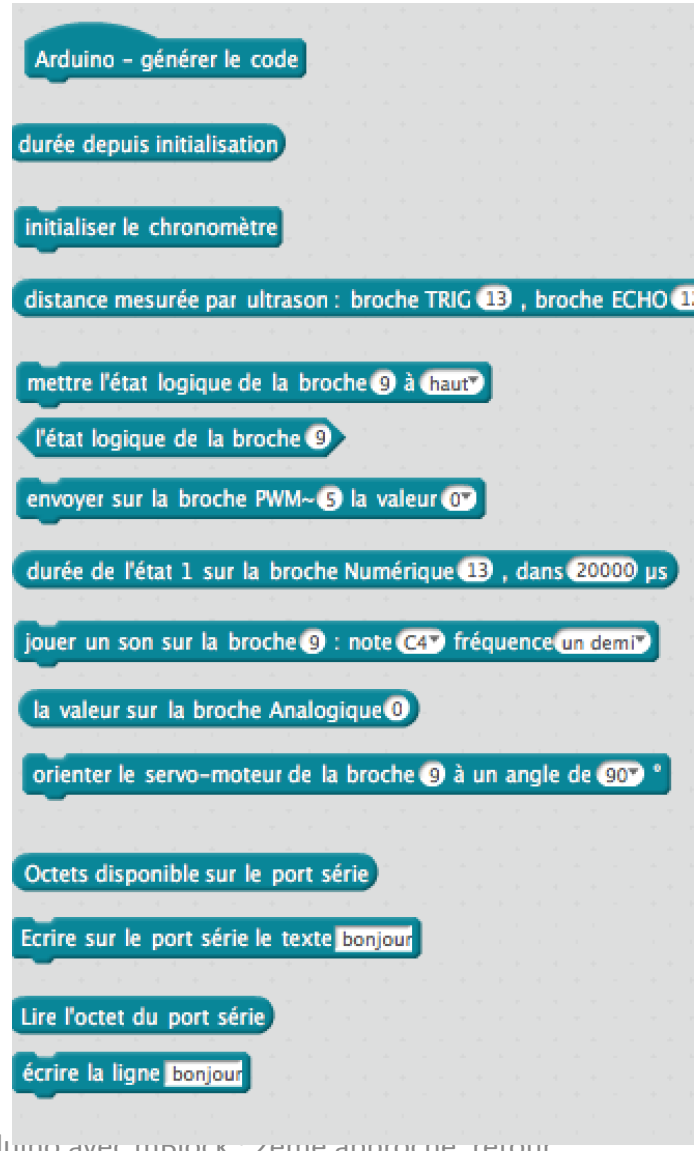
En mode Arduino : les familles de blocks

Les opérateurs

Elle comprend les opérateurs arithmétiques et logiques basiques ainsi que les fonctions mathématiques courantes

Il y a également un générateur de nombre aléatoire et des fonctions de traitement de tableaux de caractères

Les opérateurs sont utilisés dans les conditions



Le pilotage

Elle comprend un en-tête de bloc qui permet de passer en mode de génération de code et les opérations de manipulation de port

A noter : l'apport inestimable de la getsion d'un servomoteur par une instruction unique ainsi que la récupération directe d'une distance mesurée par un capteur à ultrasons et la simplicité d'écriture des messages sur le moniteur série

APPLICATION

Quelques exemples ià partir des fiches Planète Sciences

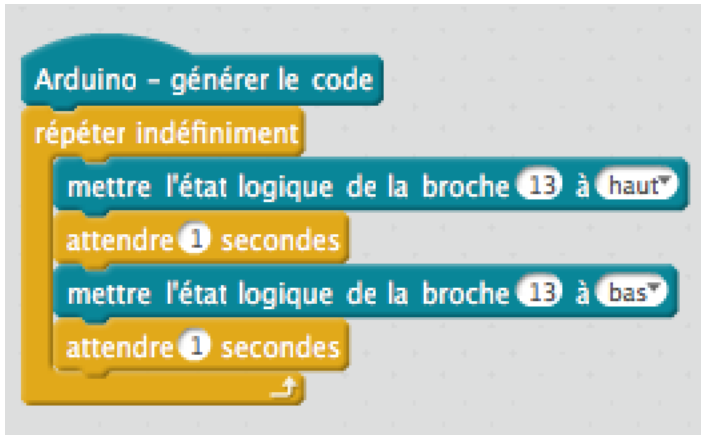
Faire clignoter une LED

Il n'est plus nécessaire de définir ni d'initialiser les PIN

Pour faire clignoter une LED avec Arduino, on se souvient qu'il suffit :

1. De mettre le PIN de la LED à l' état haut
2. D'attendre un certain temps (que l'on fixe)
3. De mettre le PIN de la LED à l'état bas
4. D'attendre encore un certain temps (que l'on fixe)
5. De recommencer

On relie l'anode (patte longue de la LED au PIN 13)




```
void setup(){
    pinMode(13,OUTPUT);
}

void loop(){
    digitalWrite(13,1);
    _delay(1);
    digitalWrite(13,0);
    _delay(1);
    _loop();
}

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

void _loop(){
}
```

Le code Arduino est généré automatiquement

Faire clignoter une LED – version PRO

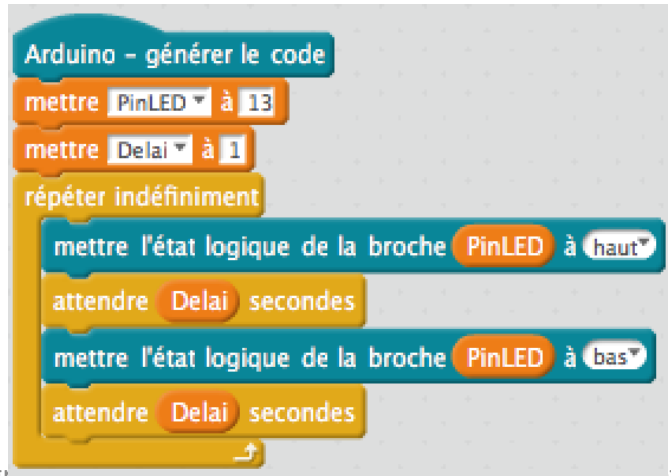
Bien évidemment, cette version du « Hello World » des programmeurs du monde entier n'est pas une avancée majeure de l'algorithmique mais elle mérite quelques améliorations :

- Si on veut que la LED ne soit pas systématiquement sur le PIN 13, on peut utiliser une variable qui définira le PIN en début de programme et la modifier (la personnaliser) à un seul endroit (portabilité)
- Si on veut modifier simplement le clignotement, on peut mettre le délai d'attente en variable pour n'avoir à le changer qu'à un seul endroit ou le faire varier par le programme, par exemple, plus un objet est proche et plus le clignotement est rapide

On crée 2 variables

On leur affecte une valeur

On les utilise ensuite dans le programme



Le code Arduino devient un peu plus « verbeux »

```
double PinLED;
double Delai;


void setup(){
    PinLED = 13;
    Delai = 1;

    pinMode(PinLED,OUTPUT);
}

void loop(){

    digitalWrite(PinLED,1);
    _delay(Delai);
    digitalWrite(PinLED,0);
    _delay(Delai);

    _loop();
}

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

void _loop(){

}
```

Contrôler un servomoteur

On se souvient que la bibliothèque « servo » d'Arduino permet de commander un servomoteur en ne lui donnant que la valeur de l'angle à laquelle il devait se positionner

Pour un servomoteur à rotation continue, cette valeur d'angle conditionnait le sens et la vitesse de sa rotation (0 = à gauche, 90 = arrêt, 180 = à droite)

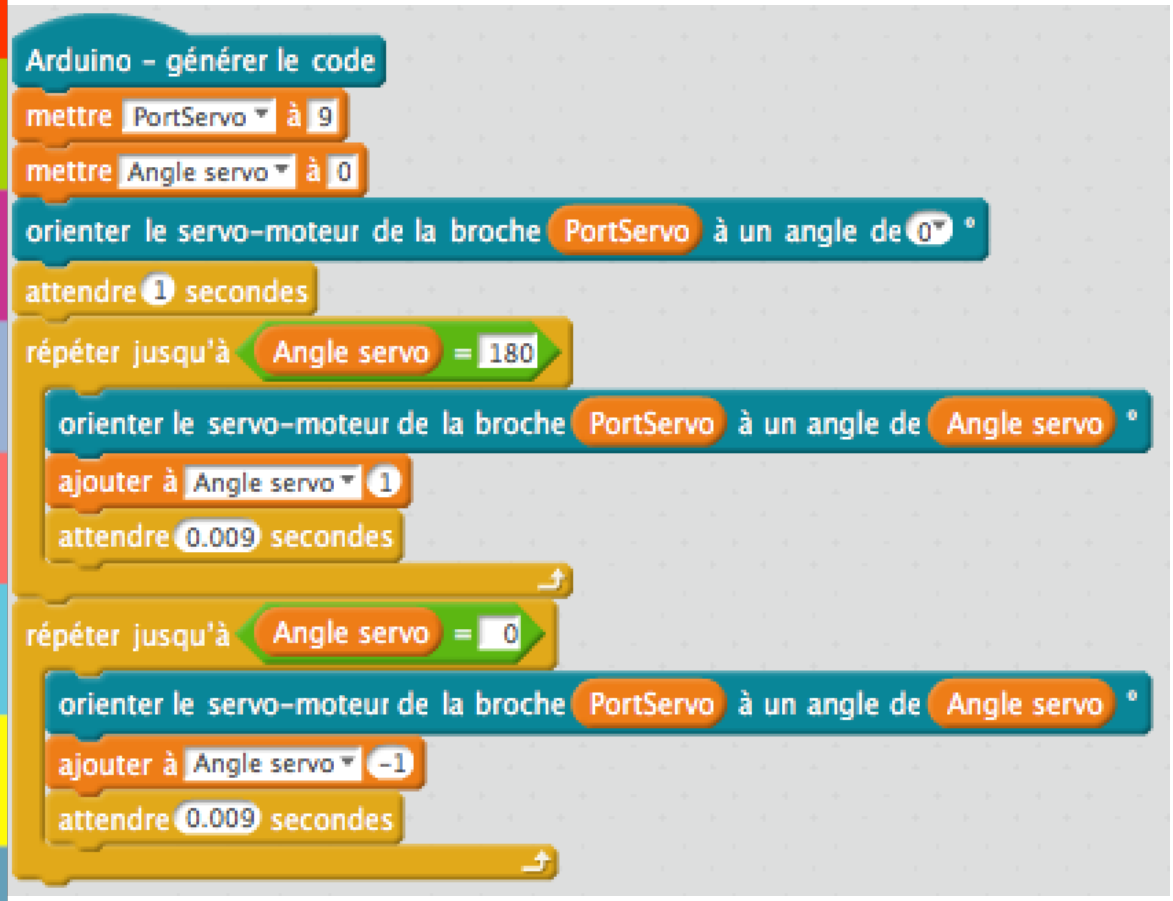
On se souvient également qu'il fallait déclarer un objet servo, « l'attacher » à un PIN préalablement déclaré lui aussi

mBlock a simplifié considérablement la gestion des servomoteurs : 1 seul bloc suffit

Si on veut ou si l'on doit utiliser plusieurs fois, ce bloc, on peut le variabiliser

Contrôler un servomoteur : les classiques

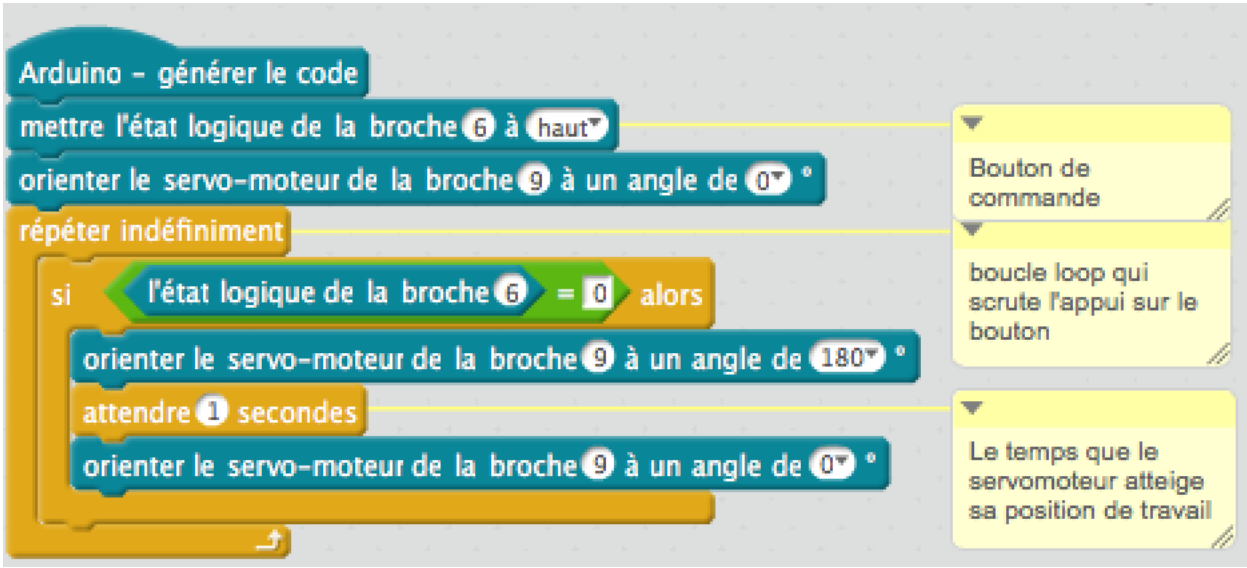
Le balayage progressif* des angles par le servomoteur dans un sens puis dans



(*) A noter : pour obtenir un mouvement progressif et non saccadé d'un servomoteur, on peut utiliser une boucle répétitive

La séquence de bascule d'un bac à l'appui d'un bouton poussoir : rotation à 90°, pause, rotation inverse

Notez la condition de test de la valeur lue sur le PIN du bouton qui sera relié à GND lors de l'appui (on détecte un passage à l'état 0 ou LOW)



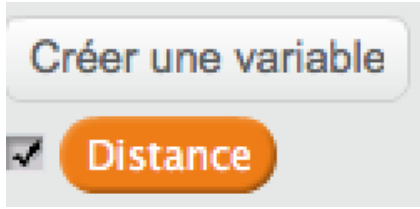
Les commentaires permettent de garder la trace des bonnes idées

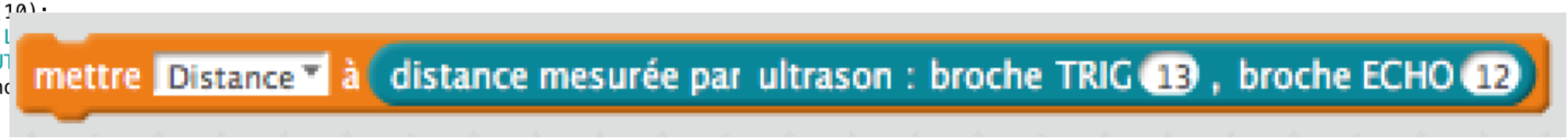
Mesurer une distance

On se souvient que l'utilisation d'un sonar (capteur à ultrasons) n'était pas évidente au premier abord

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

#include <Servo.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
double Angle;
double Distance;
float getDistance(int trig,int echo){
    pinMode(trig,OUTPUT);
    digitalWrite(trig,LOW);
    delayMicroseconds(2);
    digitalWrite(trig,HIGH);
    delayMicroseconds(10);
    digitalWrite(trig,L
    pinMode(echo, INPUT
    return pulseIn(echo
}
Servo servo_9;



void setup(){
    Angle = 0;

    servo_9.attach(9); // init pin
}
```

mBlock a également simplifié considérablement la gestion du module « sonar »: 1 seul bloc et une variable « cible » suffisent





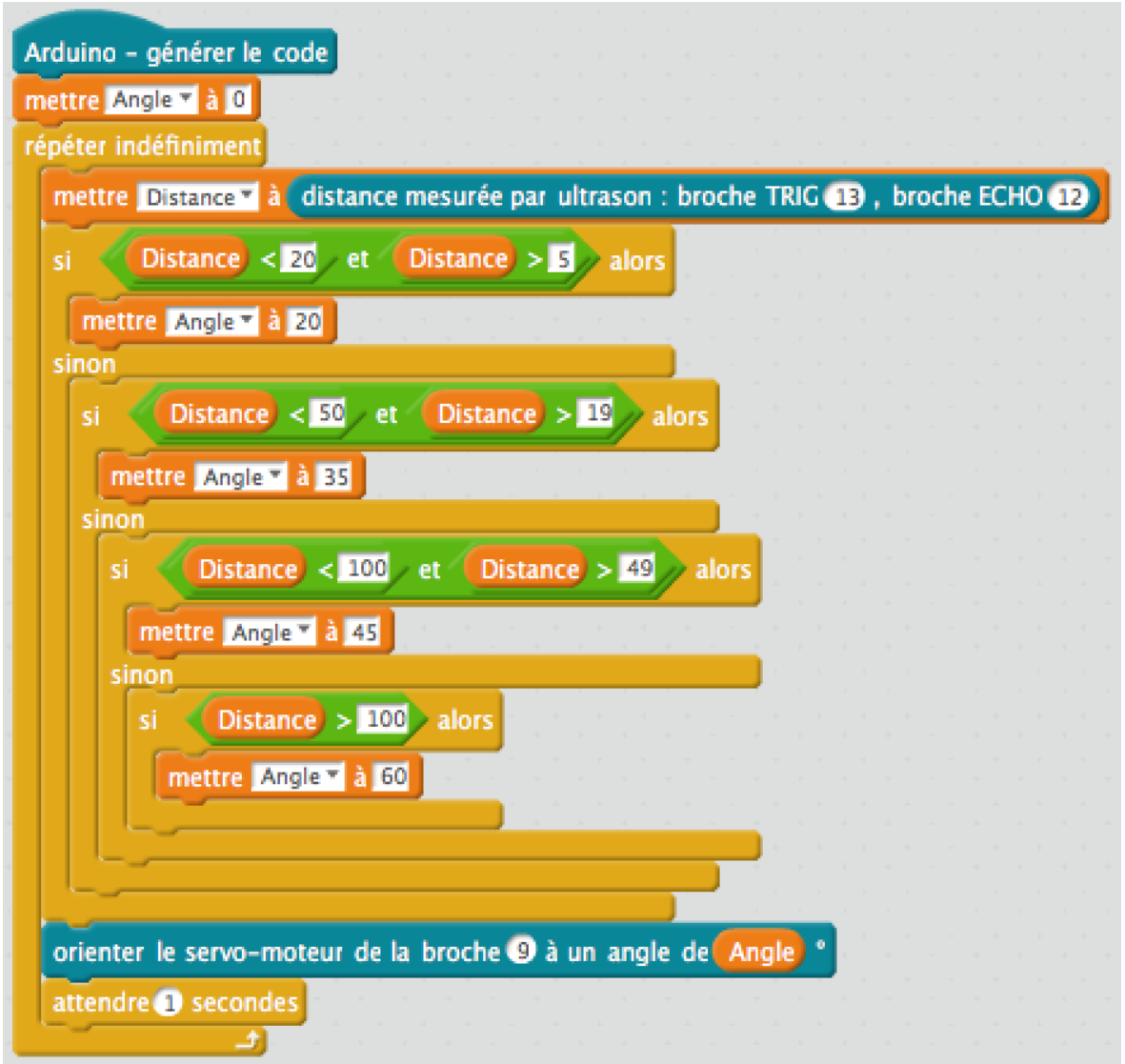
Mesurer une distance et asservir un servomoteur

Comme on sait mesurer une distance et positionner un servomoteur à un angle précis avec mBlock, on sait réaliser un asservissement d'un servomoteur

Par exemple, pour ajuster un angle de tir ou de lancer de quelque chose à une distance mesurée, ce qui peut être utile comme action de jeu

Mais il manque encore des blocs permettant de gérer des tableaux afin de faire une correspondance entre la valeur de distance mesurée et la valeur d'angle correspondante

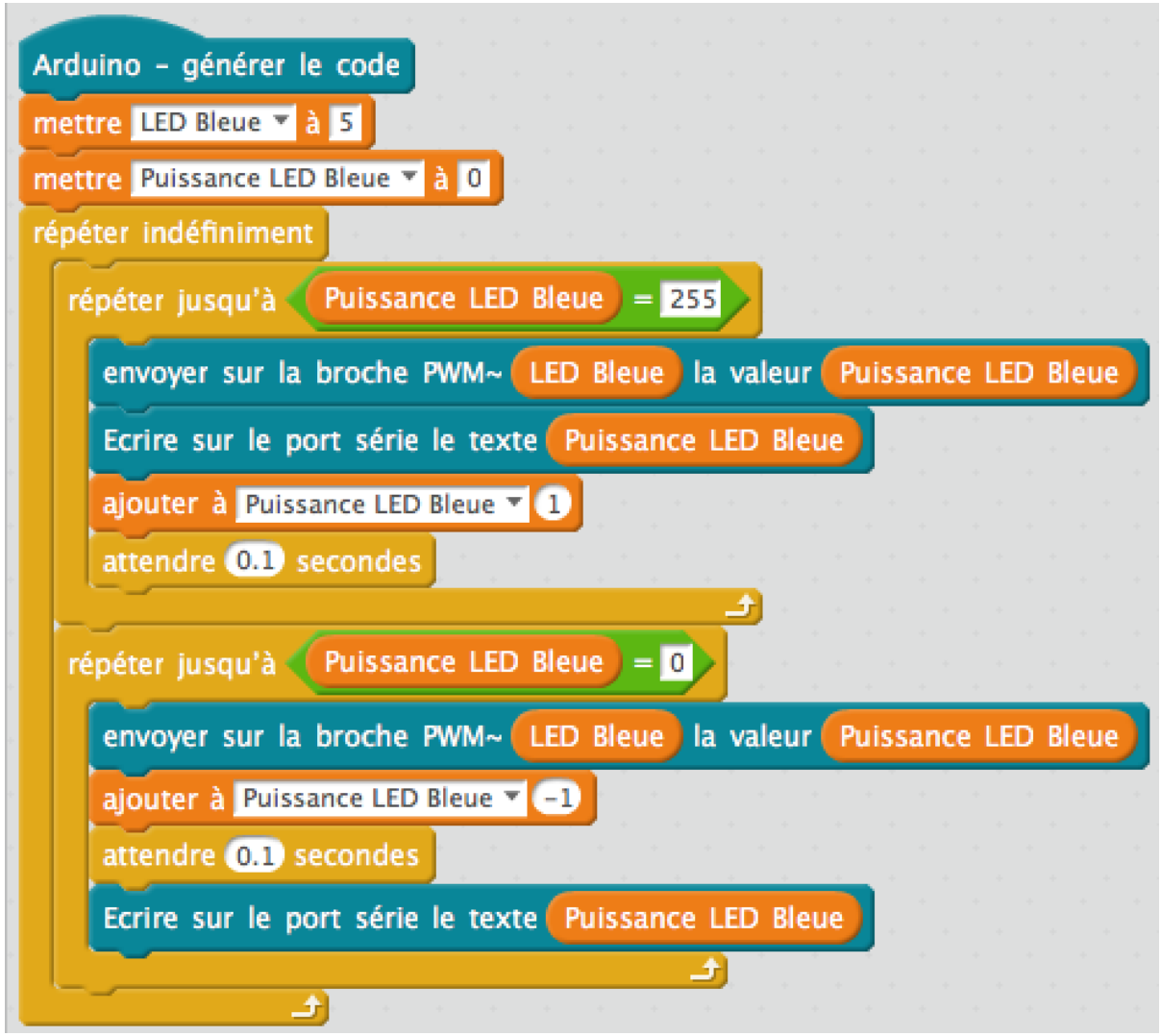
Toutefois, on peut faire une approximation par une cascade d'alternatives « si… alors… sinon… » ou par une fonction



Contrôler la puissance avec le PWM

Le principe : certains ports de l'Arduino peuvent moduler le signal en sortie selon la la méthode du PWM (voir la fiche F3 sur le site de Planète Sciences) **avec l'instruction analogWrite()**

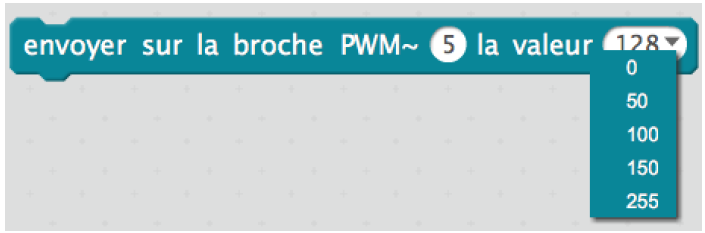
Des valeurs sont prédéfinies mais on peut mettre une valeur personnalisée dans la mesure où elle se situe dans l'intervalle [0 ; 255] l'exemple ci-contre faire croître puis décroître la puissance d'une LED



Contrôler la puissance avec le PWM

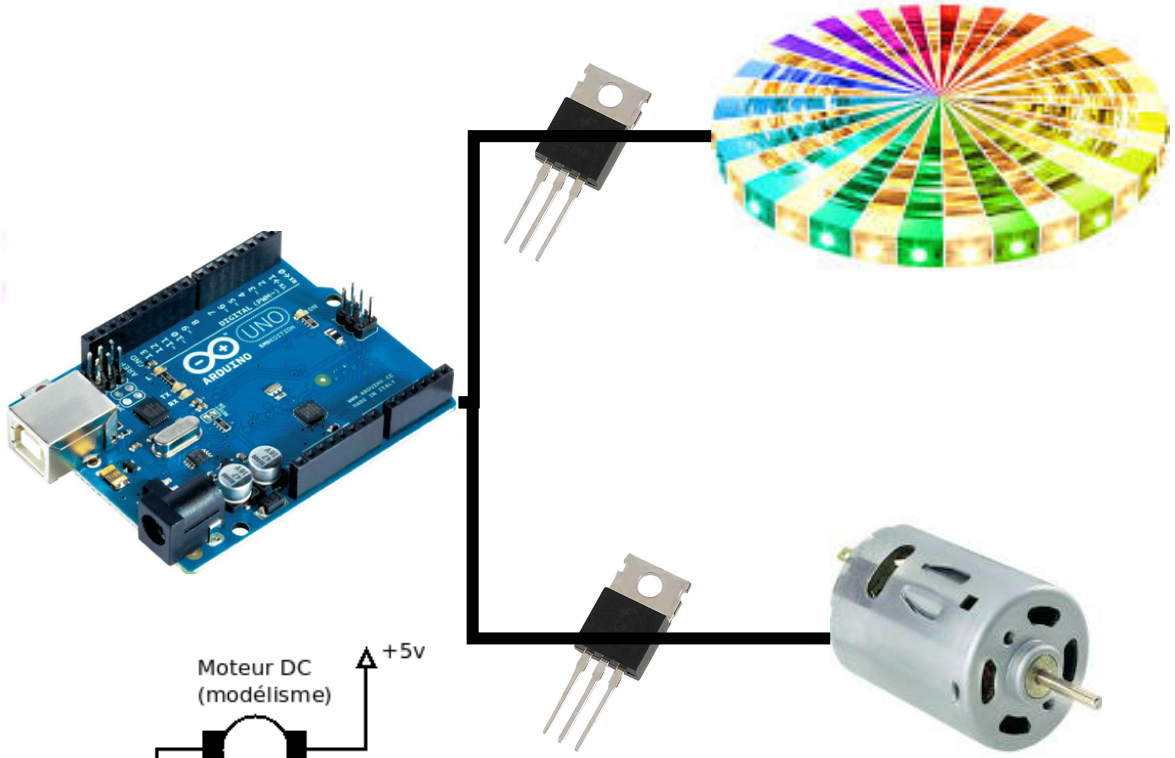
Applications :

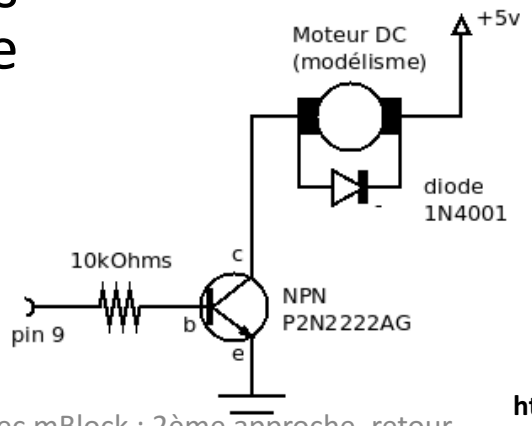
- moduler la puissance d'un groupe de LED ou d'une LED RGB pour avoir une lumière particulière ou des effets lumineux



- Faire varier la puissance d'un moteur électrique en rotation dans un seul sens avec un potentiomètre (pour changer de sens, utiliser un pont en H)

ATTENTION : il faut penser à mettre une diode en parallèle du moteur pour éviter un courant retour qui risque d'endommager le circuit



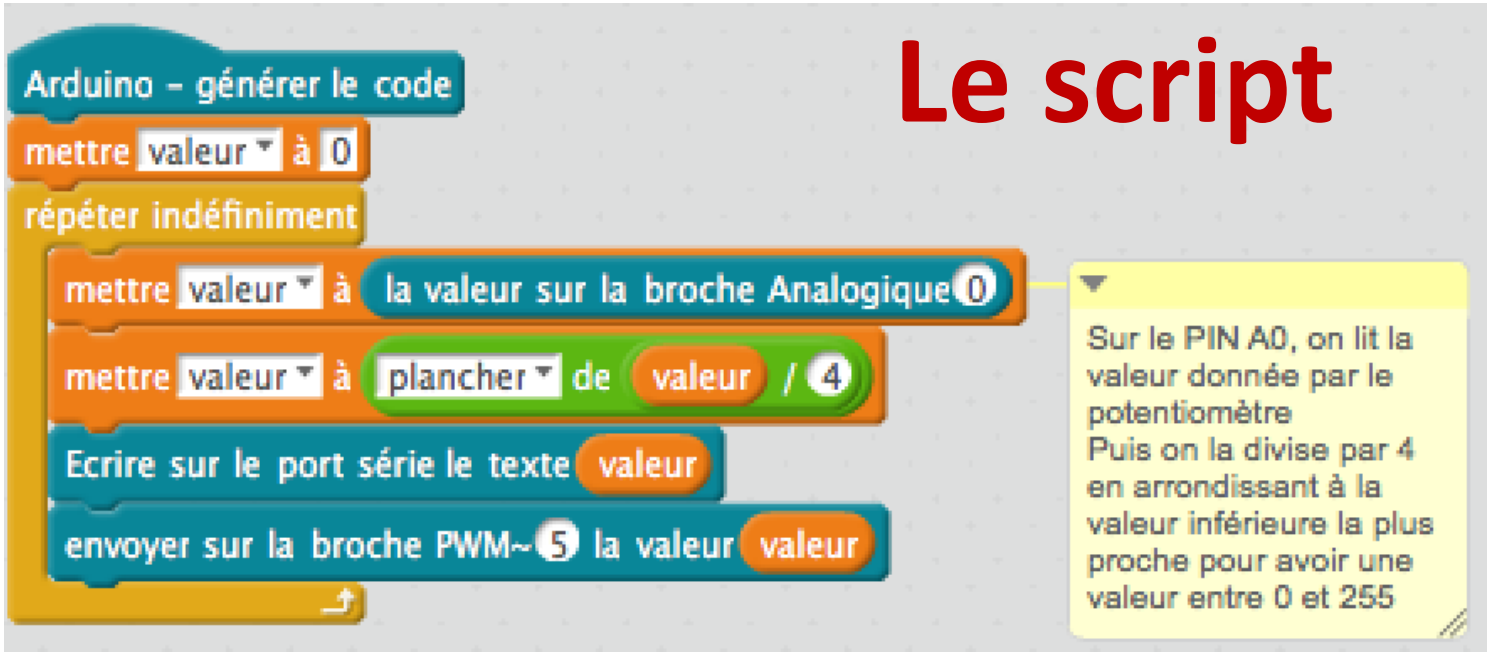


Pour gérer l'alimentation en courant fort, on utilise un transistor de puissance

http://arduino103.blogspot.fr/2011/05/controle-moteur-dc-via-transistor-pwm.html

Contrôler la puissance avec le PWM

Le script

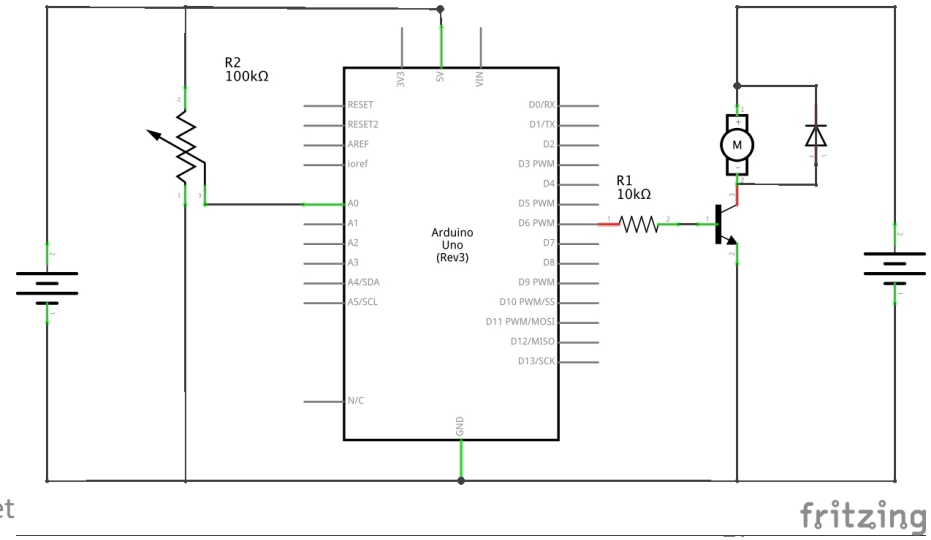


Un potentiomètre réglable par vis (trimmer) permet d'ajuster une valeur pour la conserver

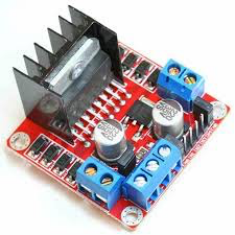
Ne pas oublier la diode anti-retour!

La taille du transistor dépend de la puissance du courant utilisé par le moteur

Un potentiomètre est utilisé pour donner une valeur de PWM, cette valeur est lue sur port A0 de l'Arduino et un signal PWM correspondant à la valeur lue est envoyé sur la base du transistor qui module l'alimentation du moteur en conséquence



Contrôler un moteur avec un pont en H



La fiche F3 explique le principe du pont en H pour contrôler le sens de rotation et la puissance d'un moteur électrique à courant continu

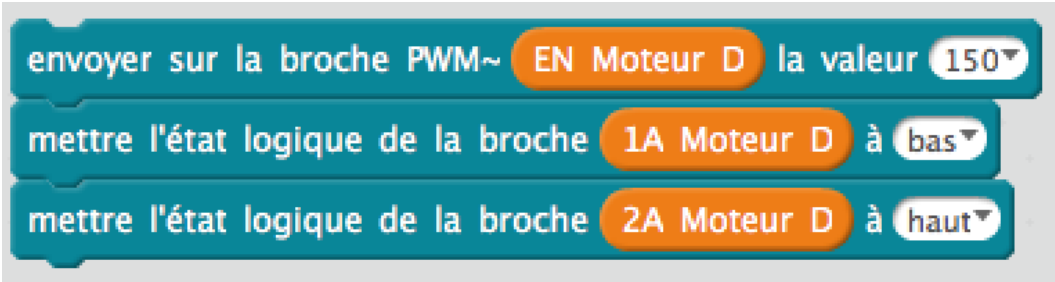
On se base sur la table de vérité du circuit qui implémente le pont en H pour envoyer les signaux de commande par 3 PIN pour chaque moteur

EN	1A (3A)	2A (4A)	Fonction
HIGH	LOW	HIGH	Tourne à droite
HIGH	HIGH	LOW	Tourne à gauche
HIGH	LOW	LOW	Arrêt rapide
HIGH	HIGH	HIGH	Arrêt rapide
LOW	Indifférent	Indifférent	Arrêt

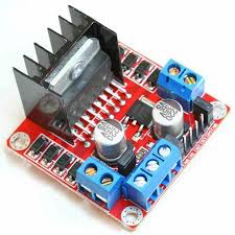
La vitesse est modulée par l'envoi de sa valeur via l'instruction analogWrite(PIN, valeur), ce qui est pratique pour contrôler la vitesse par un joystick

Par exemple pour faire tourner le moteur à droite à mi-vitesse, les blocs correspondant à la première ligne du tableau seraient les suivants

L'algorithme est plus clair et plus sûr en déclarant des variables correspondants aux PIN d' l'Arduino à utiliser (PWM pour EN)

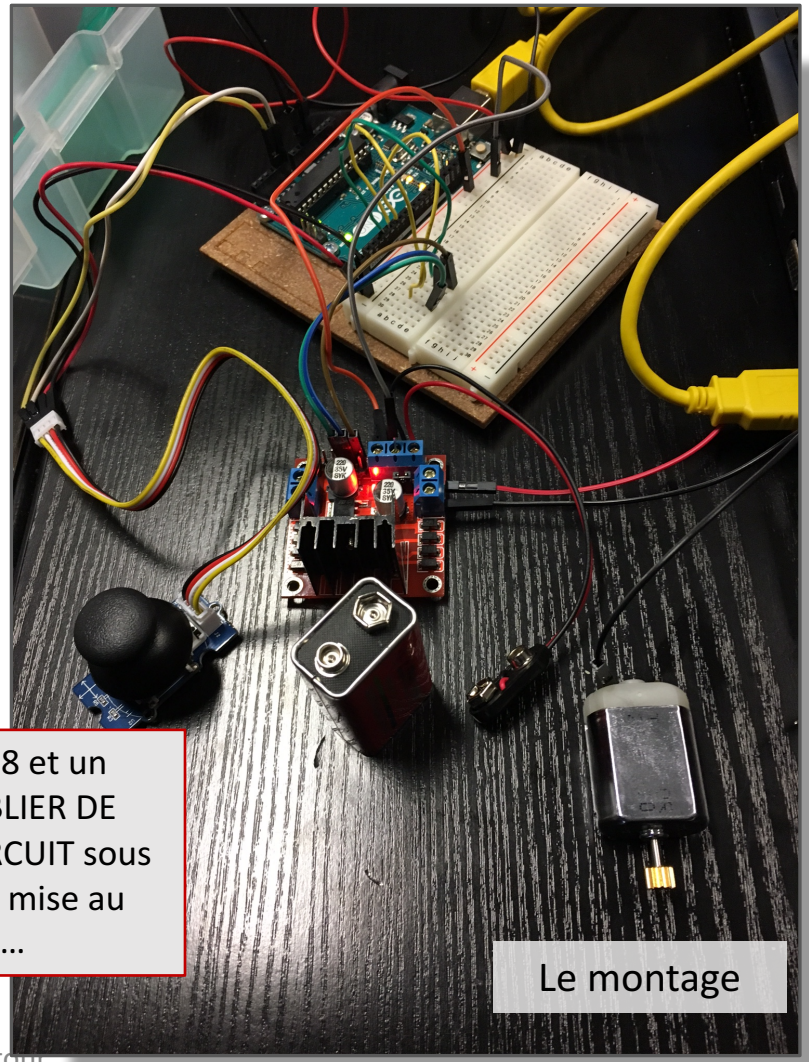


Contrôler un moteur avec un pont en H



```
Arduino - générer le code
mettre pin In 1 Moteur 1 à 4
mettre pin In 2 Moteur 1 à 5
mettre pin 1 Joystick 1 à 1
mettre pin 2 Joystick 1 à 0
répéter indéfiniment
  Ecrire sur le port série le texte la valeur sur la broche Analogique pin 1 Joystick 1
  Ecrire sur le port série le texte la valeur sur la broche Analogique pin 2 Joystick 1
  mettre ValeurPWMConvertie à abs de la valeur sur la broche Analogique pin 1 Joystick 1 - 512 / 2
  Ecrire sur le port série le texte ValeurPWMConvertie
  envoyer sur la broche PWM~ pin En Moteur 1 la valeur ValeurPWMConvertie
  si la valeur sur la broche Analogique pin 1 Joystick 1 > 562 alors
    mettre l'état logique de la broche pin In 1 Moteur 1 à haut
    mettre l'état logique de la broche pin In 2 Moteur 1 à bas
  sinon
    si la valeur sur la broche Analogique pin 1 Joystick 1 < 462 alors
      mettre l'état logique de la broche pin In 1 Mote
      mettre l'état logique de la broche pin In 2 Mote
    sinon
      mettre l'état logique de la broche pin In 1 Mote
      mettre l'état logique de la broche pin In 2 Mote
```

Le script réalisé par Nicolas : le joystick donne une valeur entre 0 et 1023 et de 512 au point de repos. Le script lit cette valeur et la convertit en une valeur de PWM puis l'analyse pour déterminer le sens de rotation ou une consigne d'arrêt



Le montage

Le montage utilise un circuit pont en H à base de L298 et un moteur alimenté par une pile 9V. IL NE FAUT PAS OUBLIER DE RELIER LA MASSE DE L'ARDUINO AVEC LA MASSE DU CIRCUIT sous peine d'interférence. Avec mBlock, la réalisation (et la mise au point) du montage et du script a pris une heure…

Fin