Transmis par Guillaume Martin / académie-Grenoble

Algorithmique et Arduino
Programmation d'une carte Arduino

On peut actuellement programmer une carte Arduino de multiples manières, mais celles identifiées pour cette formation sont :

- mBlock ;
- Ardublock ;
- ide Arduino.

Aucun de ces environnements de programmation n'est explicitement au programme. Ainsi, il n'est pas nécessaire de formaliser les connaissances et compétences liées à ces environnements de programmation.

Par contre, elles peuvent aider les élèves à acquérir la pensée algorithmique, pour peu que l'on fasse l'effort de prendre du recul sur la programmation, en faisant le lien avec les concepts algorithmiques fondamentaux.

En effet, les environnements de programmation sont appelés à évoluer, à devenir de plus en plus ergonomiques. Par contre, la pensée algorithmique, elle, restera.

Cependant, il est absolument indispensable par ailleurs de formaliser les notions techniques relatives

Table des matières

1 Partie 1 – Arduino et Algorithmique 3
1.1 Notion 1 : Évènement 3
1.2 Notion 2 : Action 4
1.3 Notion 3 : Structure conditionnelle SI ALORS SINON 5
1.4 Répéter jusqu'à / tant que. 6
1.5 Répéter indéfiniment. 7
1.6 Notion de SETUP et de Programme avec Ardublock. 7
1.7 Notion 4 : Variable 8
1.8 Notion 5 : Types de Variable 10
1.9 Pilotage en Bluetooth. 11
2 Notions techniques 12
2.1 Diagramme des blocs internes classique avec Arduino. 12
2.2 Communication entre votre ordinateur et la carte Arduino. 12
2.3 Contrariétés fréquentes avec Arduino. 13
2.4 Entrées digitales de la carte Arduino. 13
2.5 Entrées analogiques de la carte Arduino. 14
2.6 Sorties Digitales de la carte Arduino. 14
2.7 Sorties Analogiques de la carte Arduino : le PWM 15
2.8 Entrées/sorties « protocoles » 16

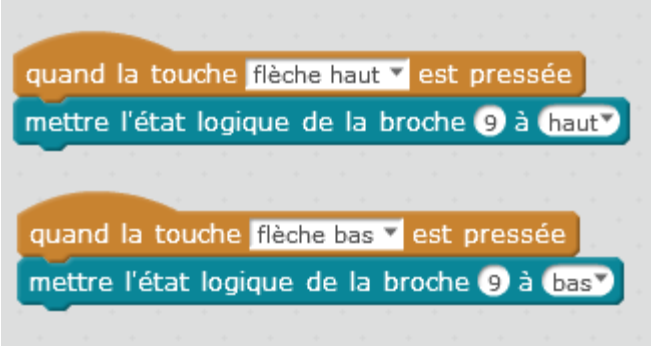
2.8.1 Détecteur de distance à ultra-son. 16
2.8.2 Pilotage d’un servomoteur. 16
2.9 Intensité délivrée dans les sorties 17
2.10 Le BUS I2C 17
2.11 Le port UART. 17

1 Partie 1 – Arduino et Algorithmique

1.1 Notion 1 : Évènement

La notion d'événement est primordiale en algorithmique. Un événement est un changement d'état d'une variable extérieure au programme, qui déclenche le lancement de code informatique.

Dans l'exemple ci-dessous, sous mBlock un appui sur les flèches *haut* et *bas* allume ou éteint la lampe :

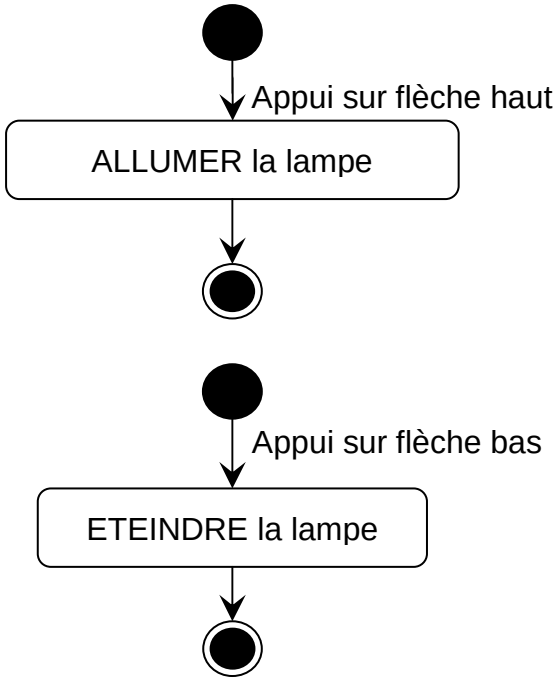


L'algorithme associé est

QUAND appui sur flèche haut
DEBUT
    ALLUMER la lampe
FIN

QUAND appui sur flèche bas
DEBUT
    ETEINDRE la lampe
FIN

Le diagramme d'activité associé est :



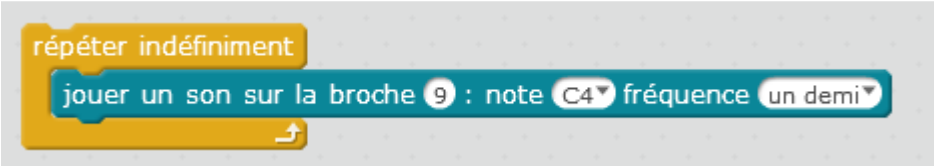
appui sur Bouton Poussoir ON et **appui sur Bouton Poussoir OFF** sont des évènements, de même pour **appui sur flèche haut** et **appui sur flèche bas**.

Remarque : dans Ardublock, il est plus difficile de programmer un évènement, ce qui est un inconvénient.

1.2 Notion 2 : Action

Une action est une activité d'un programme. Elle est exprimée par un verbe à l'infinitif.
Les exemples ci-dessous permettent de jouer une note avec un buzzer.

Sous mBlock :



Sous Ardublock :

En C :

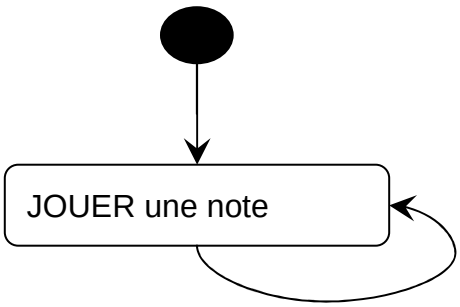
```
void setup()
{
}

void loop()
{
  tone(2, 440);
}
```

On remarque que sous Ardublock, les actions ne sont pas forcément écrites avec un verbe à l'infinitif, ce qui peut prêter à confusion pour les élèves. Par contre en C, c'est bien le cas.

L'algorithme associé est :

REPETER indéfiniment
    JOUER un son
FIN REPETER

Le diagramme d'activité associé est :



JOUER un son est une action.

Attention, les élèves confondent fréquemment l'infinitif et le participe passé pour les verbes du premier groupe, ce qui rend l'algorithme incompréhensible. Exemple : Jouer, ~~joué~~, Connecter, ~~connecté~~ ; Avancer, ~~avancé~~.

1.3 Notion 3 : Structure conditionnelle SI ALORS SINON

La structure SI ALORS SINON permet de prendre des décisions quant à l'action à réaliser.
On teste une condition qui doit être soit vraie, soit fausse. (C'est une condition binaire)

Exemple avec une orientation de servomoteur après appui sur bouton poussoir branché sur D8 :

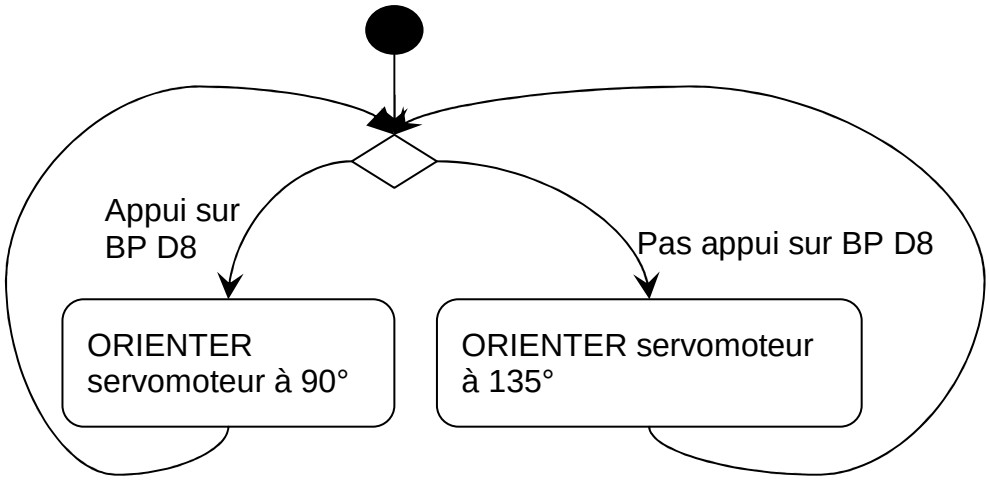
Sous mBlock :

Sous Ardublock :

L'algorithme associé est :

REPETER indéfiniment
    SI Appui sur le bouton branché sur D8 ALORS
        Orienter le servomoteur à 90°
    SINON
        Orienter le servomoteur à 135 °
    FIN SI
FIN REPETER

Le diagramme d'activité associé est :



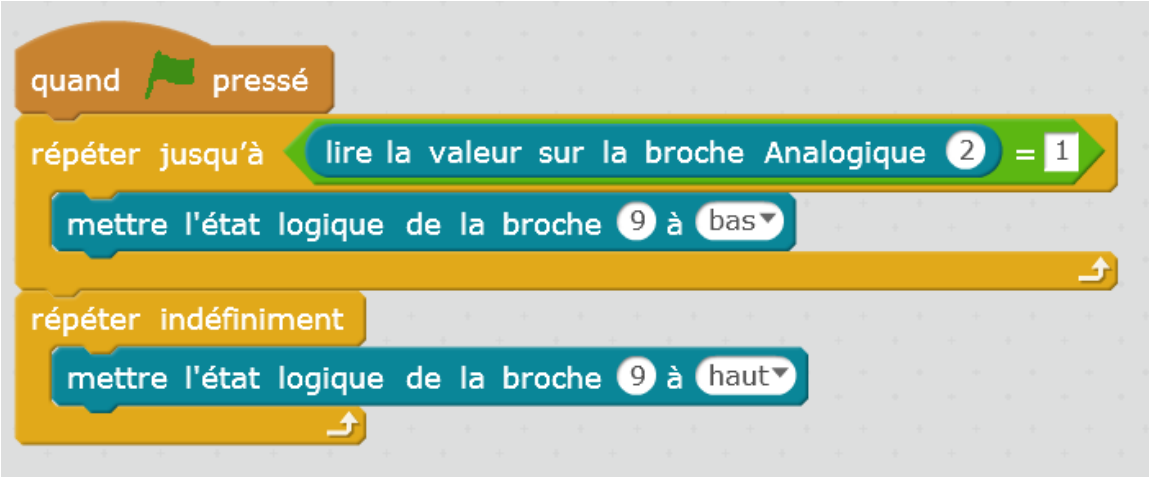
(Appui sur Bouton Poussoir D8 = VRAI) est une condition, soit vraie, soit fausse.

Attention : pour tester une condition, il est important de connaître le type de donnée que l'on manipule (voir plus bas)

1.4 Répéter jusqu'à / tant que.

Il peut être utile de répéter une action JUSQU'A ce qu'une condition soit vraie, ou TANT QUE une condition soit vraie.
Par exemple, pour ne faire démarrer un système que lorsqu'on appuie sur le bouton poussoir 2, on peut utiliser :

Avec mBlock, l'instruction **Répéter jusqu'à** :

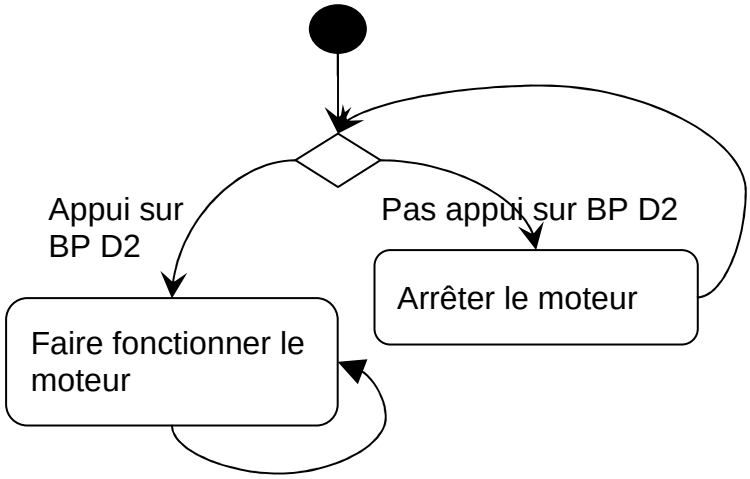


Avec Ardublock, l'instruction **répéter tant que** :

L'algorithme correspondant est, pour mBlock :

QUAND Drapeau pressé
REPETER jusqu'à Bouton poussoir pressé
    Arrêter le moteur
FIN REPETER jusqu'à
REPETER indéfiniment
    Faire fonctionner le moteur
FIN REPETER

Le diagramme d'activité :



1.5 Répéter indéfiniment.

La boucle répéter indéfiniment est tellement courante qu'il n'est pas besoin de la détailler plus ici.

1.6 Notion de SETUP et de Programme avec Ardublock.

Dans Ardublock, on programme ce qui ne sera fait qu'une seule fois en début de programme dans la partie Setup :

1.7 Notion 4 : Variable

Une variable est le **nom d'un espace de mémoire** réservée par un programme informatique.

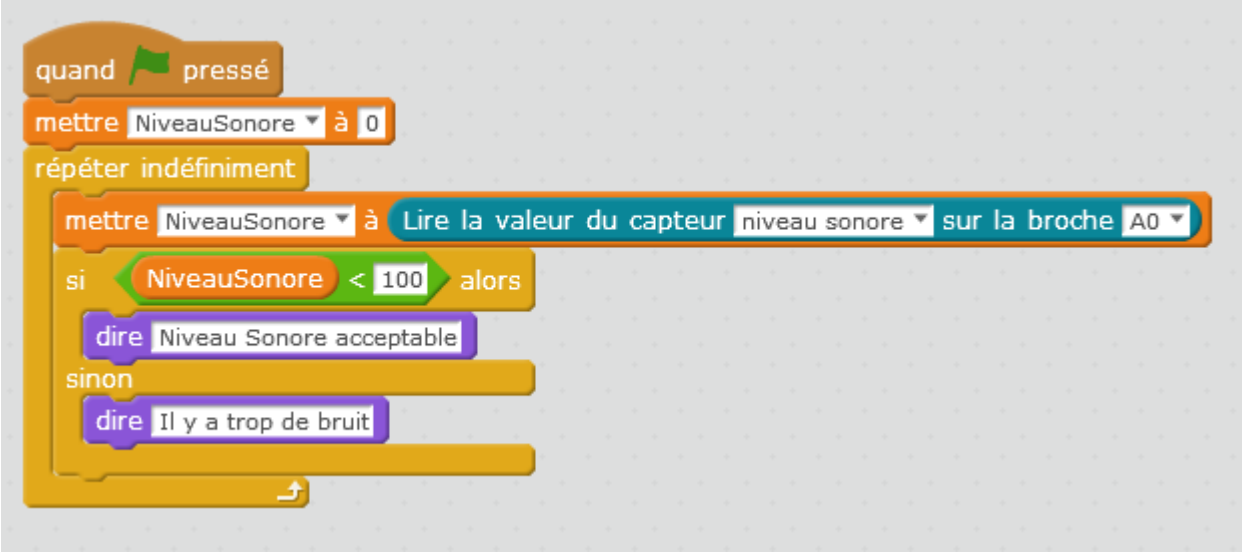
Lorsque le programme évolue, il affecte une valeur dans cet espace, qui peut changer au cours du temps. On dit alors que la valeur est affectée à la variable.

Attention 1 : En algorithmique, l'affectation s'écrit : Nom variable ← Valeur
C'est normal ! :

- Si l'on écrivait *Nom variable* = *Ancienne valeur*, on ne saurait pas s'il s'agit d'un test ou d'une affectation.
- Et écrire *Valeur* ➔ *Nom variable* serait écrire dans le sens inverse de l'habitude.

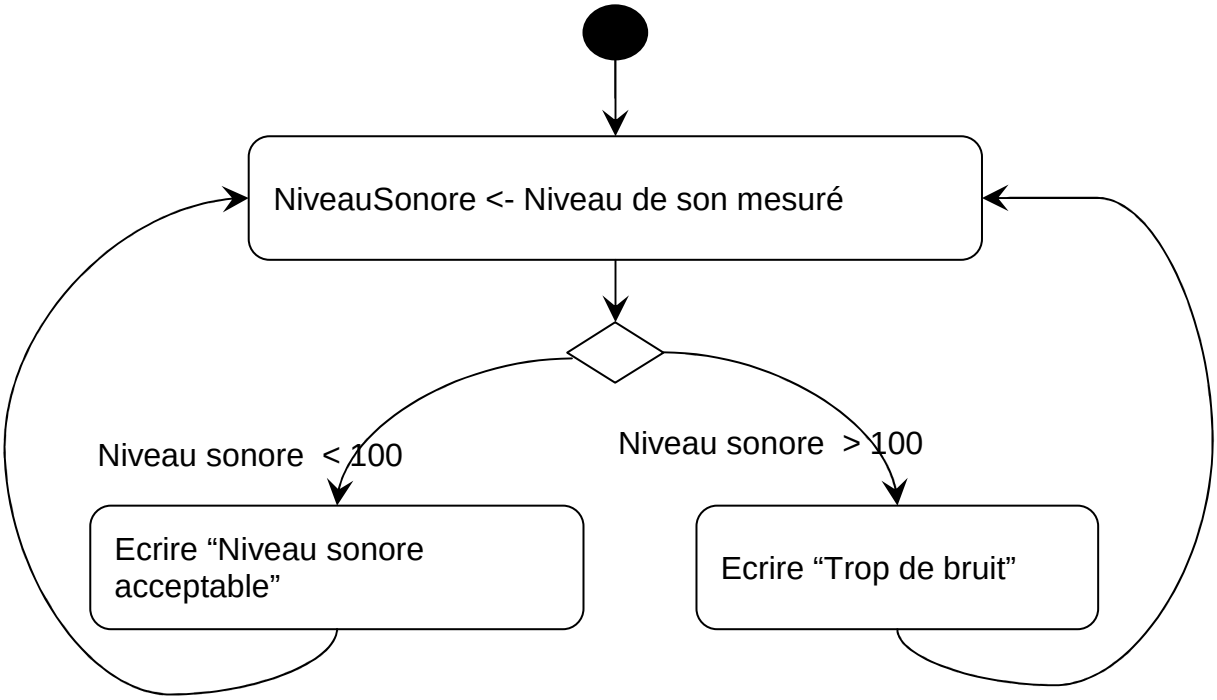
Exemple de test de niveau sonore :

mBlock :



Ardublock :

Le diagramme d'activité associé est :



L'algorithme associé est :

```
REPETER indéfiniement
        NiveauSonore ← Capteur son
        SI NiveauSonore < 100 ALORS
                AFFICHER "Niveau sonore acceptable"
        SINON
                AFFICHER "Trop de bruit"
        FIN SI
FIN REPETER
```

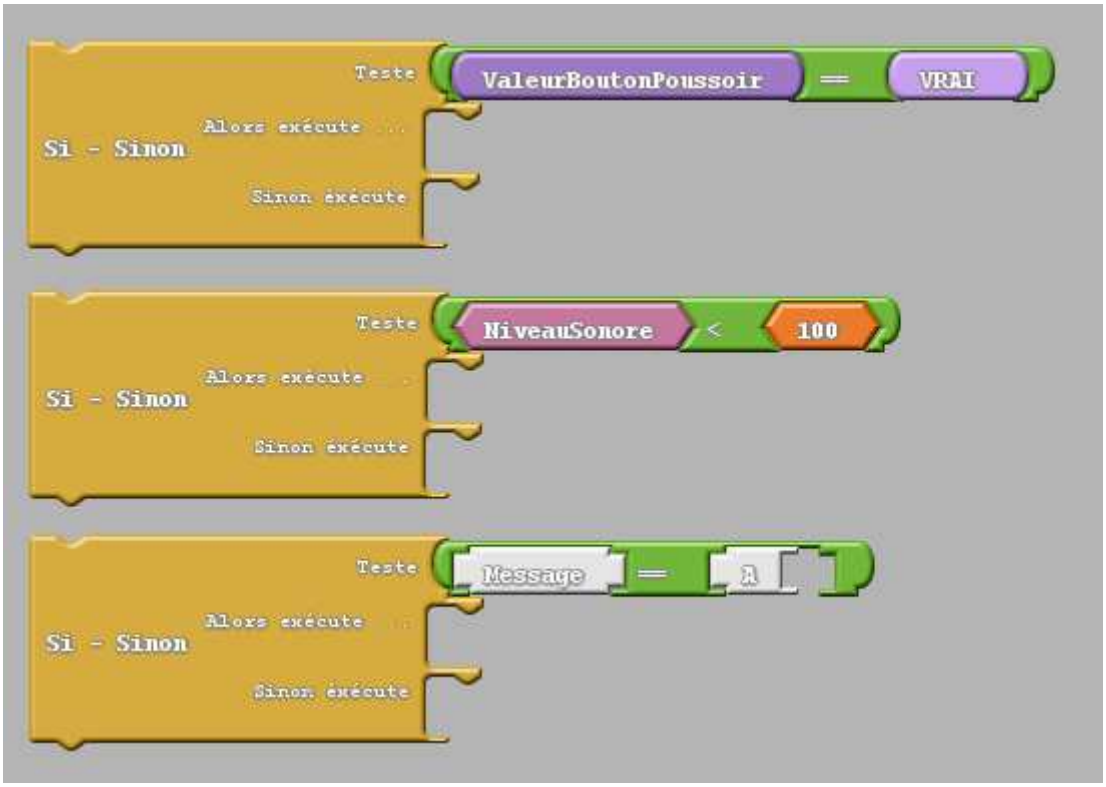
1.8 Notion 5 : Types de Variable

Au collège, on considérera que les variables peuvent principalement être de trois types :

- binaire : contiennent un état, vrai ou faux (0 ou 1)
- Entier : contiennent des nombres ;
- Chaine de caractère : contiennent du texte

Il est très important de connaître le type de la variable quand l'on veut faire un test.

- un binaire sera comparé à 0/1, ou VRAI/FAUX ;
- un entier, sera comparé à un nombre ;
- une chaîne de caractère sera comparée à du texte.



Les variables ci-dessus sont de type, dans l'ordre : Binaire, Entier, Chaine de caractère.

Remarque :

- dans mBlock, le type de la variable n'apparaît pas explicitement.
- dans Ardublock, le type de la variable est indiqué par la forme des bords (Arrondi pour du binaire, triangulaire pour des entiers, angles droits pour des chaînes de caractère)

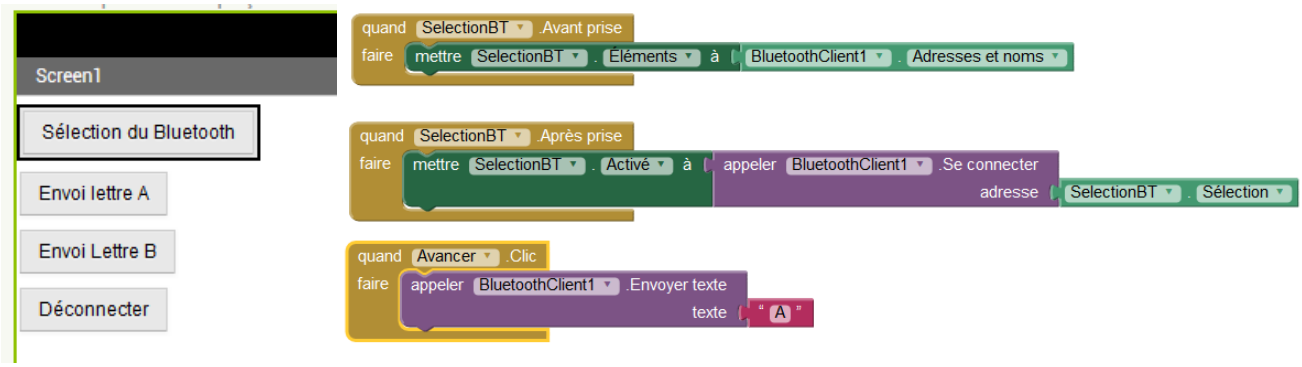
Attention :

- Dans Ardublock, le nom de la variable est toujours un peu plus foncé que la valeur de la variable. C'est un moyen de les reconnaitre.
- Dans Ardublock, il existe des types de variables que l'on utilisera rarement au niveau collège (long et décimal notamment)

1.9 Pilotage en Bluetooth.

Pour piloter une carte en Bluetooth, le principe est toujours le même : un programme, sur une tablette ou autre, envoie des codes. La carte Arduino reçoit ces codes et agit en conséquence.

Par exemple, sous AppInventor, pour envoyer les codes on utilise invariablement :



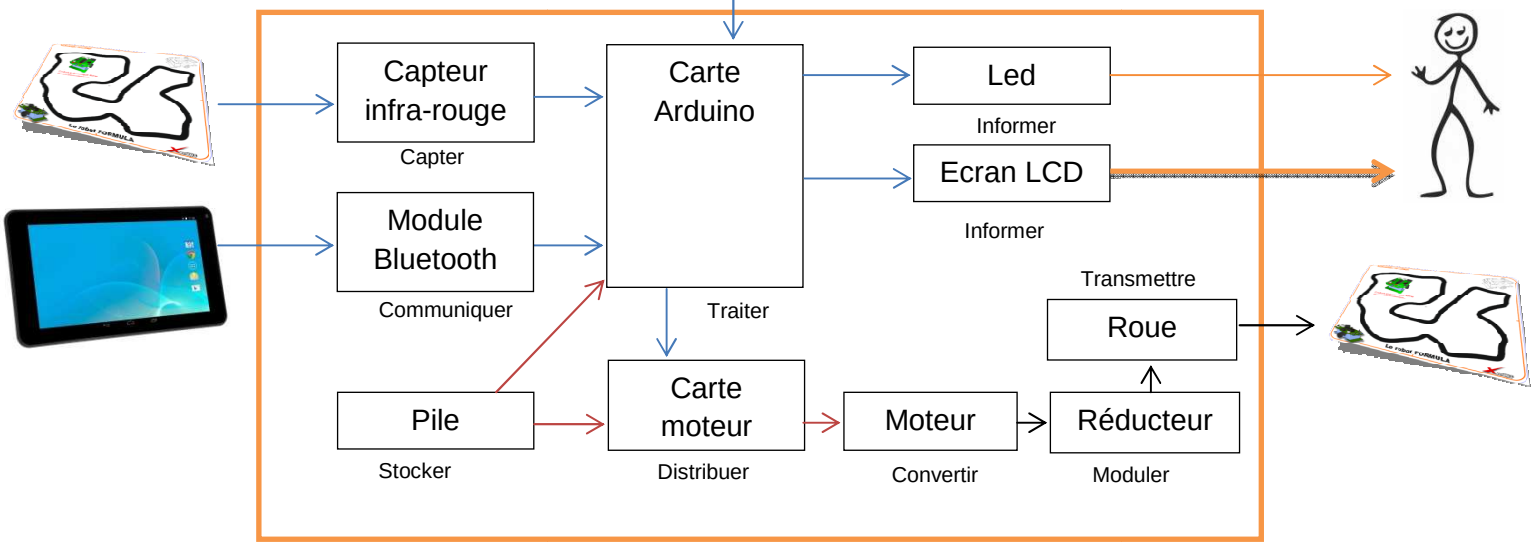
Et sous Ardublock :

2 Notions techniques

Dans la partie 1, nous avons traité des notions purement algorithmiques, qui pourraient être abordées avec d'autres environnements de programmation, tel que Scratch, AppInventor, Picaxe etc. Cependant, si l'on veut savoir programmer, il faut nécessairement avoir pris conscience des notions ci-dessous.

2.1 Diagramme des blocs internes classique avec Arduino.

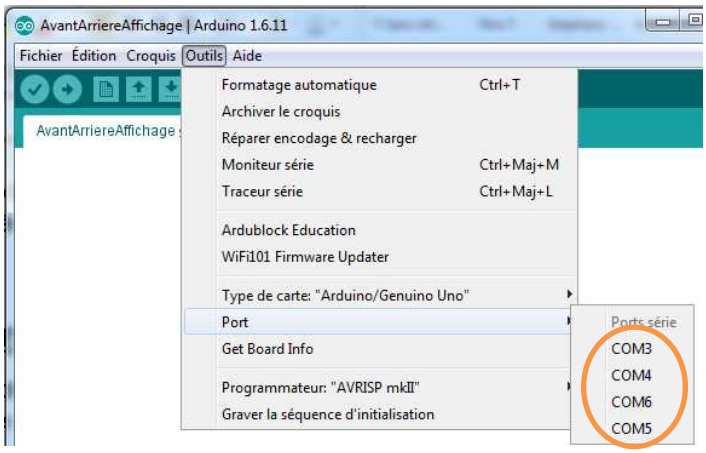
Exemple classique du robot de suivi de ligne :



La carte Arduino présente des entrées et des sorties, visibles dans le diagramme des blocs internes du robot. Il est nécessaire de caractériser. Ce n'est pas si simple que cela et c'est un passage obligatoire ! ! ! !

2.2 Communication entre votre ordinateur et la carte Arduino.

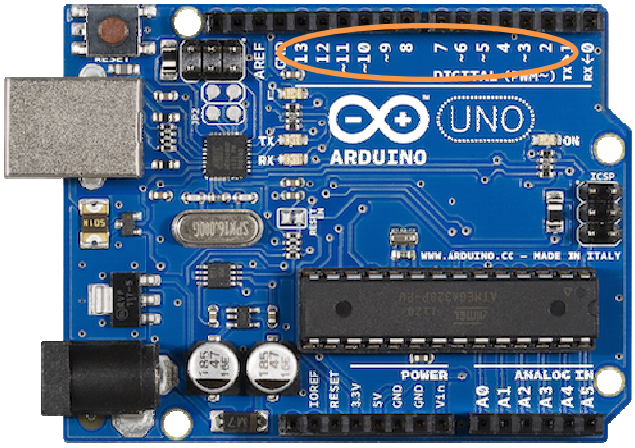
Pour communiquer avec votre carte Arduino, votre ordinateur émule un port COM. Avant de télécharger un programme sur votre Carte Arduino, il faudra sélectionner le port COM :



2.3 Contrariétés fréquentes avec Arduino.

Lorsque les sorties demandent trop d'intensité, Arduino « plante ». Il faut donc la débrancher et la rebrancher. Lorsque vous branchez et débranchez « à chaud » un composant, Arduino peut planter. Il faut donc la réinitialiser. D'une manière générale, il vaut mieux débrancher Arduino avant de lui rajouter des composants.

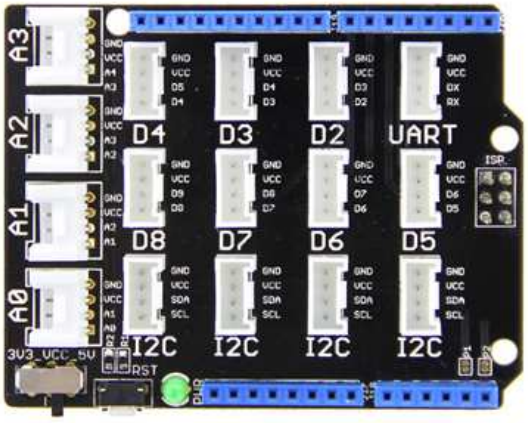
2.4 Entrées digitales de la carte Arduino.



Les entrées numériques de la carte Arduino sont numérotées de 0 à 13.
0 et 1 ne sont pas utilisables, car elles sont reliées au port série (communication avec l'extérieur)

On accède à la valeur de l'entrée digitale par l'instruction :

La valeur renvoyée sera un 0 ou un 1.



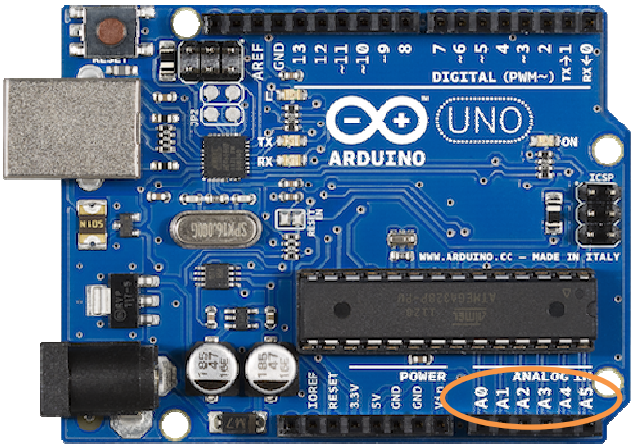
Si l'on branche une carte Grove dessus, on a accès aux entrées (pas toutes). Chaque connectique Grove propose l'alimentation pour son module.

Mais attention !!! Grove a apposé un grand repère D2 en bas de chaque connectique, qui peut prêter à confusion.

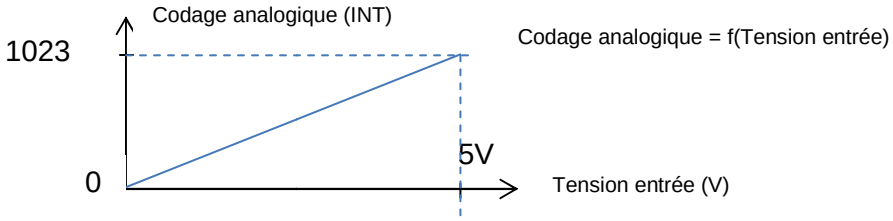
Le repère « D2 en dessous de la connectique » donne accès aux « entrées de la carte Arduino D2 et D3 ».

- Très souvent, les modules que l'on branche sur la carte Arduino n'ont qu'un seul PIN connecté (le fil jaune) le « repère D2 en dessous de la connectique » correspond donc au « repère D2 de la carte Arduino ».
- Mais parfois, les deux fils de Grove sont branchés (le jaune et le blanc) comme par exemple pour le capteur à ultrasons. A ce moment-là, le « repère D2 en dessous de la connectique » correspondra aux entrées D2 et D3 d'Arduino.

2.5 Entrées analogiques de la carte Arduino.



La carte Arduino comporte 6 entrées analogiques, qui convertissent une tension de 0 à 5V en un code allant de 0 à 1023 :



Exemples :

- Capteur de luminosité ;
- Potentiomètre.

On accède à la valeur de l'entrée analogique par l'instruction :

La valeur renvoyée sera comprise entre 0 et 1024.

2.6 Sorties Digitales de la carte Arduino.

Les entrées digitales de la carte Arduino peuvent également servir de sorties digitales. Il y a juste à les déclarer ainsi lors de la programmation.

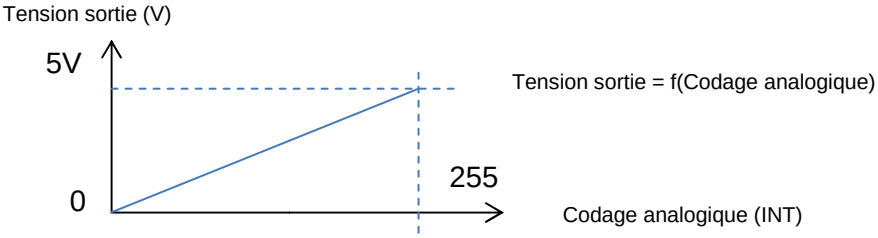
Si Arduino rencontre cette instruction, il saura que D2 est une sortie :

2.7 Sorties Analogiques de la carte Arduino : le PWM

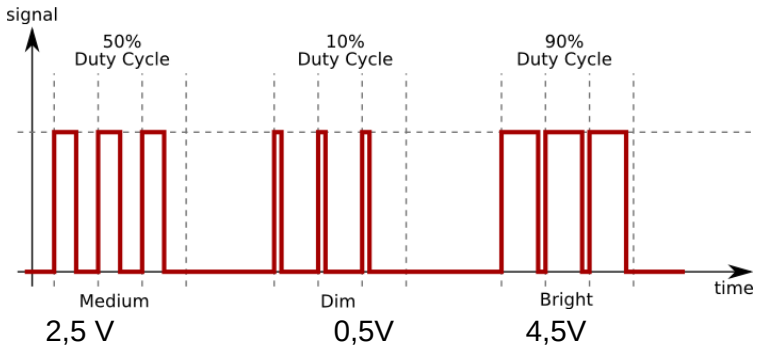
Quand on regarde une carte Arduino, celle-ci ne dispose pas de sortie Analogique. Pourtant, l'instruction

Existe bel et bien.

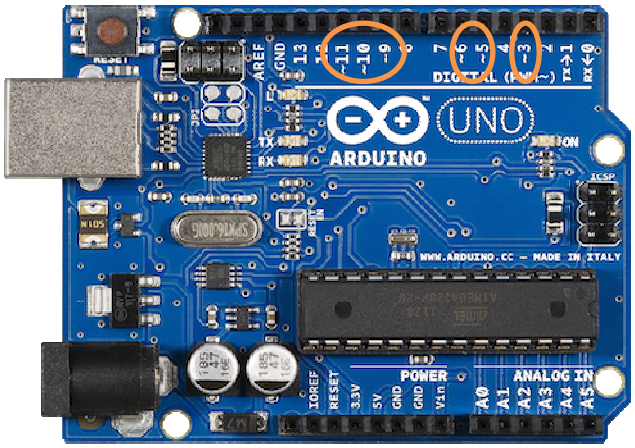
Cette instruction convertit une valeur allant de 0 à 255 en une tension de 0 à 5V.



Pour cela, Arduino utilise la technique dite du PWM (Pulse Width Modulation) : la valeur de sortie bascule de 0 à 5V, et la valeur moyenne donne la tension de sortie apparente :



Cependant, attention, toutes les sorties ne sont pas capable de réaliser cela. Seulement les sorties « rapides ». Elles sont précédées du signe « ~ » :

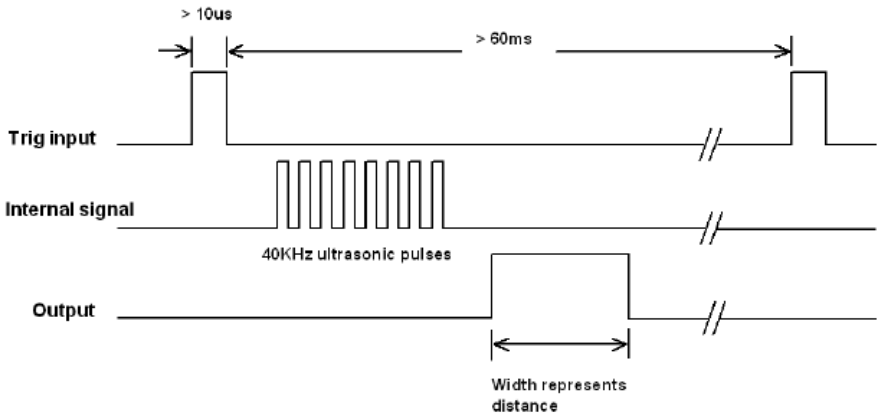


Seules les sorties 3, 5, 6, 9, 10 et 11 peuvent donc servir de sorties analogiques.

2.8 Entrées/sorties « protocoles »

2.8.1 Détecteur de distance à ultra-son.

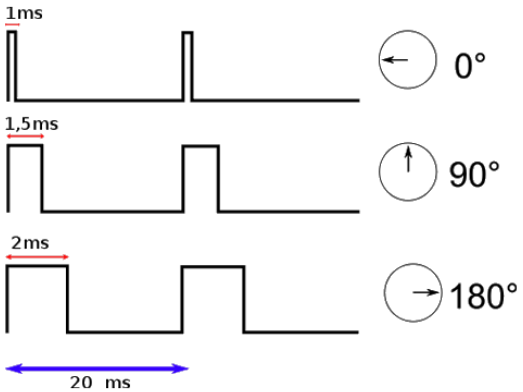
Le détecteur de distance à ultra son est une entrée qui peut sembler de type analogique. Or, il n'en est rien : la carte Arduino et le détecteur à ultra-son échangent des données au moyen d'un protocole de données numériques.
C'est une bibliothèque Arduino qui interprète ce protocole, et renvoie la distance mesurée.



Lorsque vous déclarez l'utilisation du capteur à ultra-son, vérifiez si vous possédez un capteur Grove (Une seule broche suffit) ou un capteur bon marché (Deux broches à piloter)

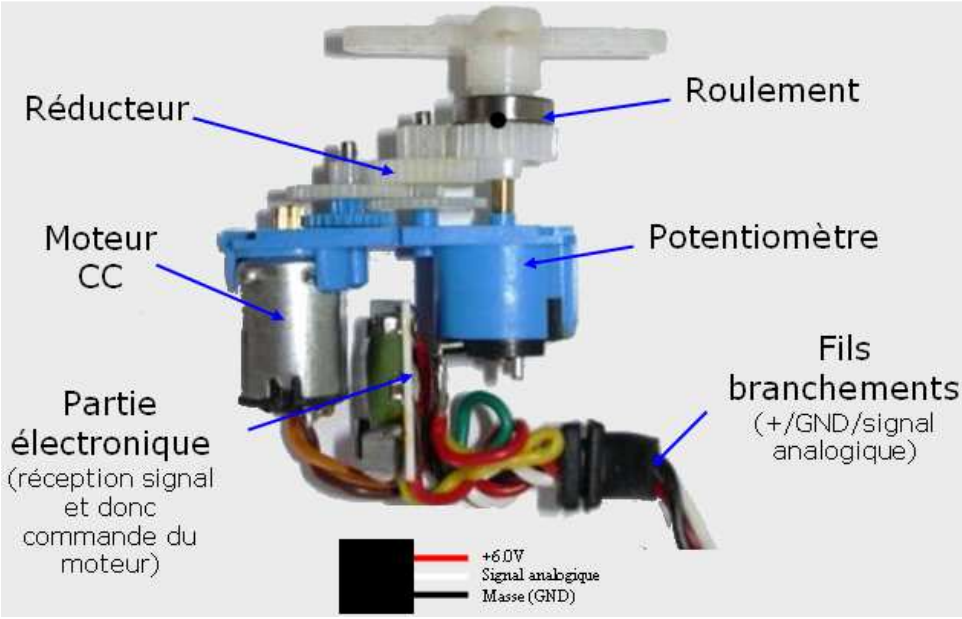
2.8.2 Pilotage d'un servomoteur.

Un servomoteur est lui aussi piloté au moyen d'un protocole :



Au vu de la rapidité nécessaire 1 à 2 ms de basculement, il vaut mieux brancher les servomoteurs « bas de gamme » (ceux que nous achetons le plus souvent…) sur les broches PWM, c'est-à-dire D3, D5, D6 (D10, D11).

La littérature indique que, en principe, on peut brancher un servomoteur n'importe où ; mais en pratique, le brancher sur les broches mentionnées ci-dessus résout souvent les problèmes.



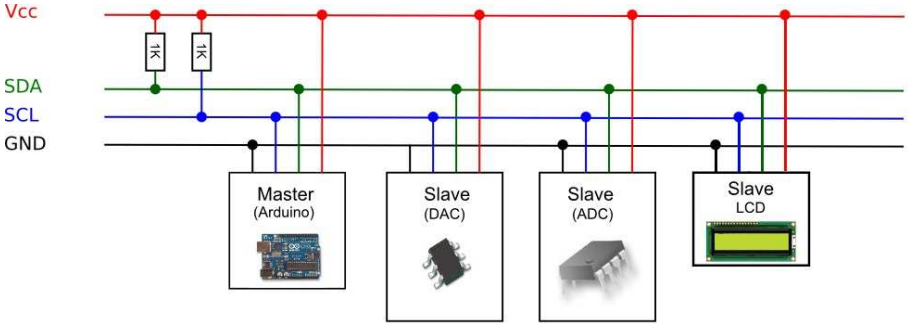
Structure interne d'un servomoteur. Pour un servomoteur à rotation continu, l'axe entre le potentiomètre et la sortie moteur est coupée.

2.9 Intensité délivrée dans les sorties

L'Arduino ne peut délivrer que des intensités de 40mA, donc insuffisante pour piloter un moteur. D'où la nécessité d'installer une carte moteur pour piloter un moteur à courant continu.

2.10 Le BUS I2C

Le bus I2C est une sorte de réseau présent sur la carte Arduino. Il est utilisé pour piloter des équipements tels que les cartes moteur ou l'écran LCD.



Chaque équipement possède son adresse sur le bus I2C. Le protocole est capable de faire dialoguer Arduino avec n'importe lequel de ces équipements.

2.11 Le port UART.

Le port UART est un port de communication série, qui permet de faire communiquer deux processeurs entre eux. Il sera rarement utilisé au collège.