# Programmation du microcontrôleur Arduino

Prérequis : Installation du logiciel Arduino IDE (téléchargeable sur le site Internet : http://www.arduino.cc)

L'IDE Arduino est un logiciel permettant de programmer les microcontrôleurs Arduino en langage C. Il est simple d'utilisation. C'est une référence dans le monde amateur. Je vais vous donner le minimum à savoir pour l'utiliser. Toutes les informations sur son utilisation peuvent se trouver facilement sur Internet.

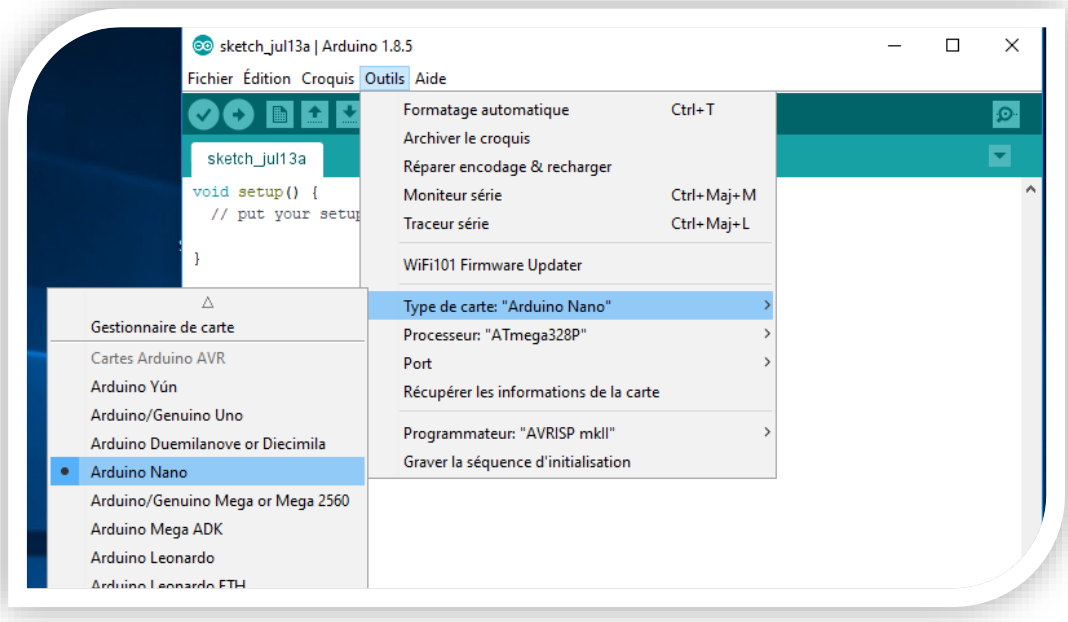
## Exécuter le logiciel Arduino IDE

Après avoir téléchargé/installé le logiciel Arduino, Double cliquer sur l'icône suivante 

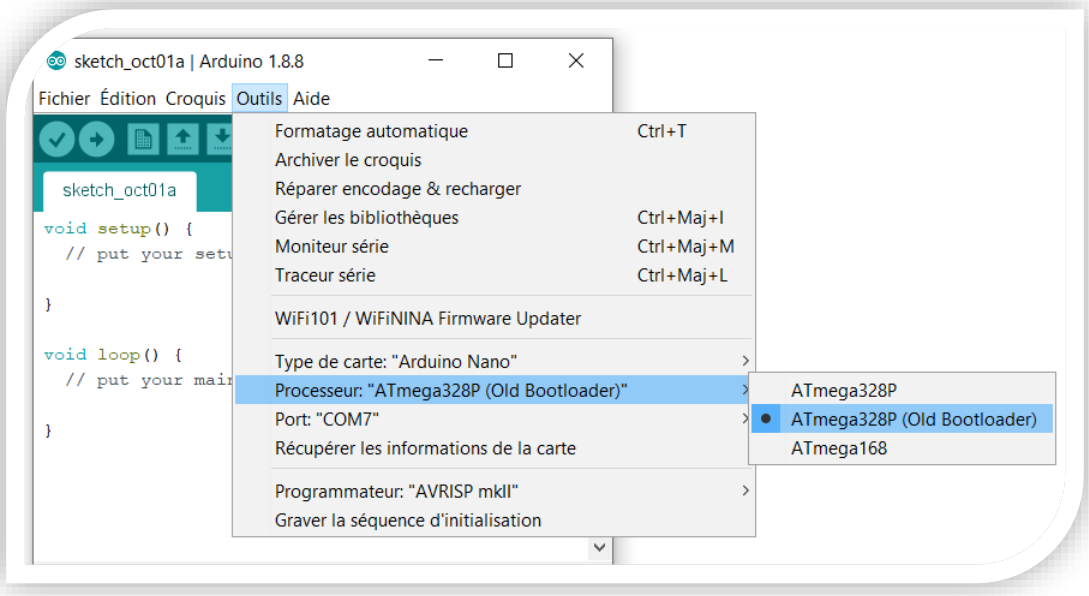

➔ Brancher la carte Arduino à l'ordinateur

## Paramétrer le logiciel

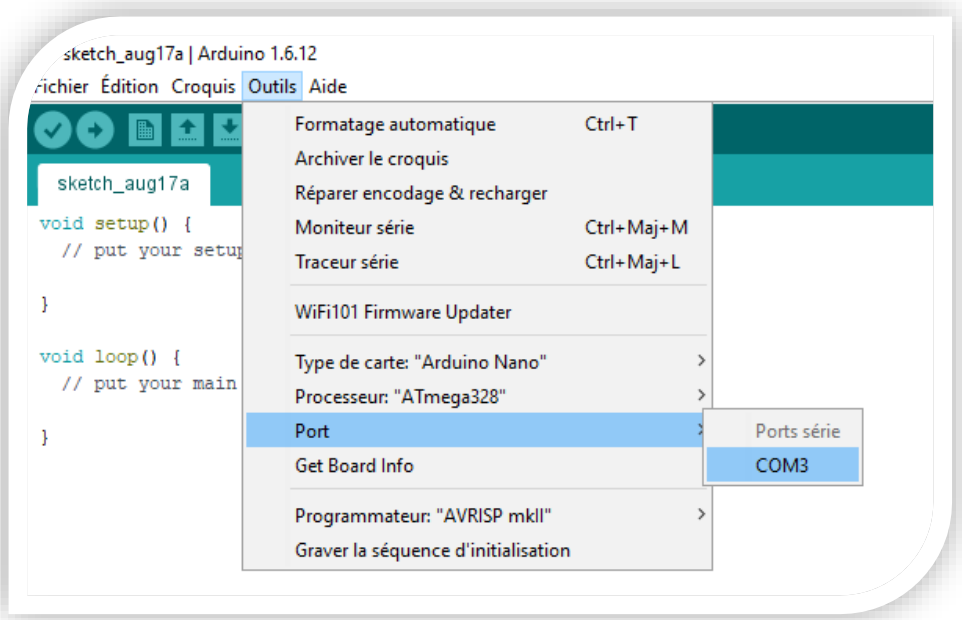
A la première exécution du logiciel. Il faut indiquer la carte utilisée. Dans notre cas choisir « Arduino Nano »




Ensuite, il faut choisir le type de Processeur. Dans notre cas : ATmega328P (Old Bootloader)

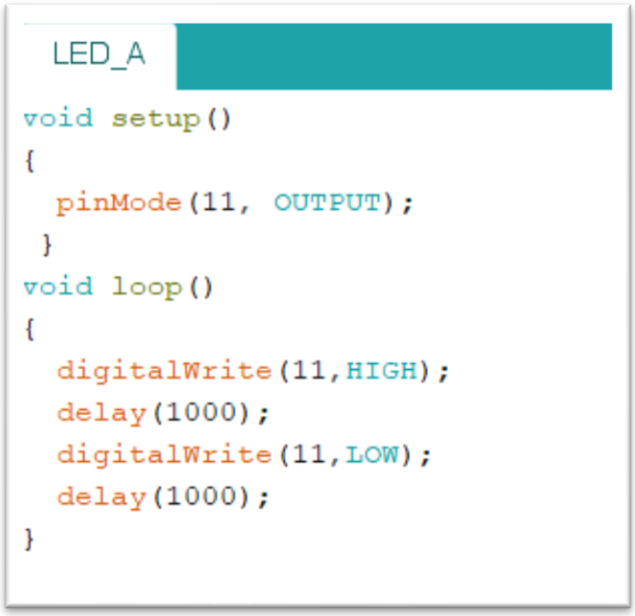


Enfin, choisir le port utilisé pour programmer l'Arduino. C'est-à-dire trouver où est branché le câble USB. Dans cet exemple le port COM 3 est utilisé




## Écrire un programme

Directement dans le document ouvert, la page blanche devant toi, écrire les lignes suivantes (telles quelles) :



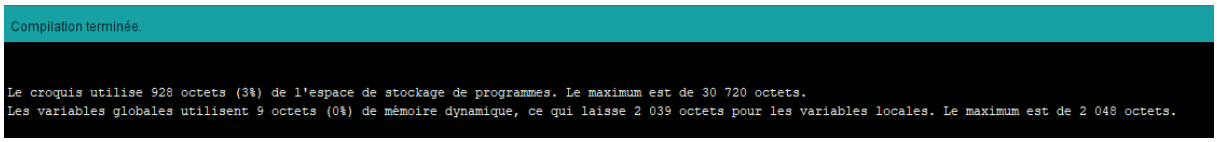

Note : Ce programme est présent dans le fichier exemple Blink.

## Compilation du code :

Pour que le programme puisse être compris et exécuter par le microcontrôleur, il faut qu'il soit traduit dans son langage. C'est ce que l'on appelle compiler le code.

Appuyer sur le bouton : 

S’il n’y a pas d’erreur le message suivant doit s’afficher.




Lors de la compilation, le logiciel Arduino vérifie le code, il permet de détecter les erreurs de syntaxe. C'est-à-dire les fautes d’orthographe et de grammaire dans les mots qu’il connait. Si tu fais une soustraction à la place d’une addition, l’erreur ne sera pas détectée. Le logiciel ne sait pas ce que tu veux faire, il reconnait juste si tu le dis correctement. Par exemple : *« **La terre est bleue comme une orange** » Paul Éluard*. Cette phrase est syntaxiquement et grammaticalement juste, mais mettons là dans un robot et il fera n’importe quoi.

Note : Le robot ne comprend pas la poésie.

# Lancer le téléversement

Une fois l’étape de la simulation passée, nous pouvons enfin programmer le microcontrôleur. Pour cela il suffit d’appuyer sur le bouton « Téléverser » : 

Note : Si une erreur apparait, vérifier les branchements, et que l’étape de paramétrage a bien été réalisée. Si ça ne fonctionne toujours pas, connecter le microcontrôleur sur un autre port USB et recommencer l’étape de paramétrage.

## Waouh ça marche !

La LED clignote ! La carte fonctionne donc !

Et maintenant à nous la conquête de l’univers !

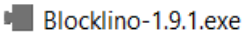
…

- Mais au fait ? A part çà il fait quoi ce programme ? Parce que là ça fait 10 minutes que je regarde une LED clignoter.

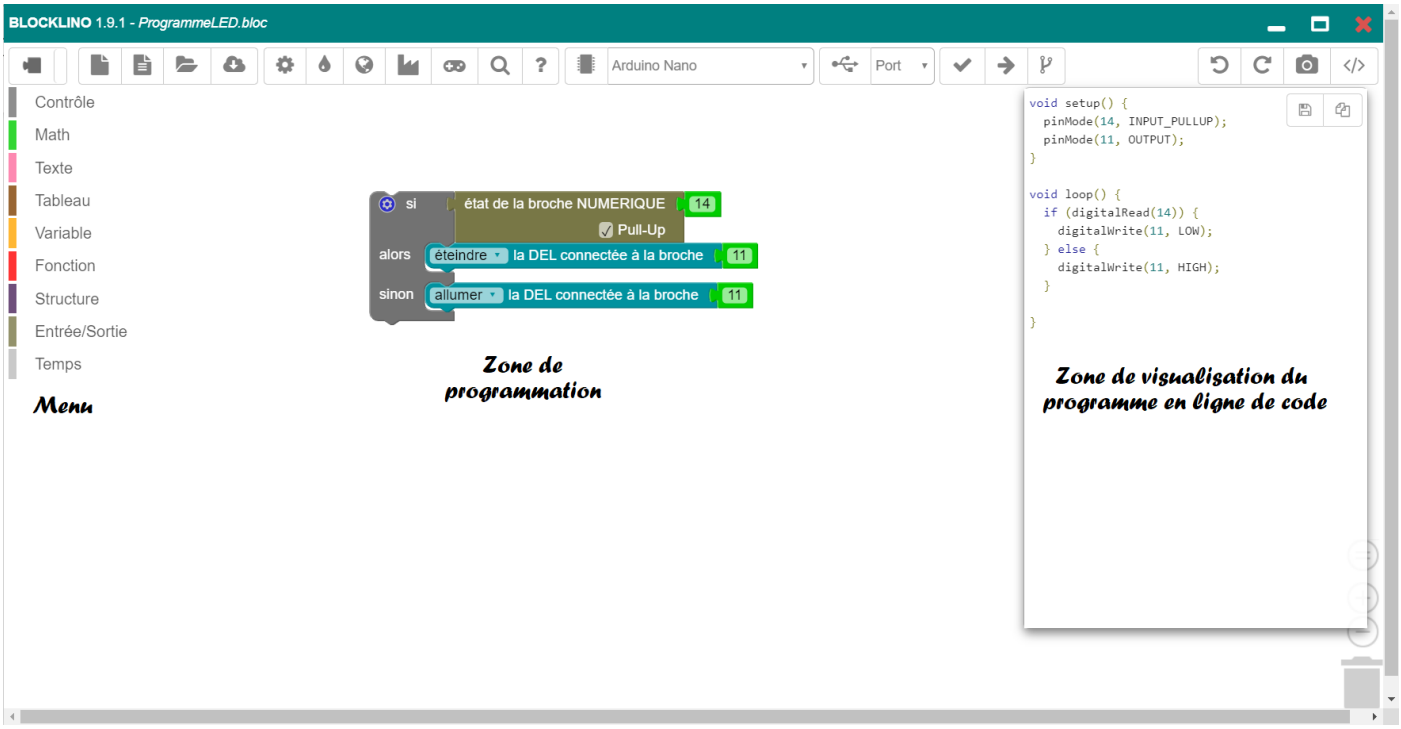
# Programmation par blocs

Pour ceux qui débutent et qui ne se sentent pas prêt pour programmer en ligne de code il existe des logiciels graphiques. Ils peuvent être utiliser pour se familiariser avec la programmation de manière ludique. Il en existe plusieurs comme ArduBlock, DuinoEdu, mBlock, MakeCode, Blocklino, et d'autres encore.

Je vais vous présenter Blocklino qui a ma préférence. Il est intuitif et produit un code en C lisible, ce qui permet une bonne transition entre la programmation par blocs et les lignes de code.

Après avoir télécharger Blocklino-1.9.1.exe il suffit de double cliquer dessus pour le lancer. 


Note : **Soyez patient !** il met du temps à se démarrer (sur mon ordinateur environ 50 secondes), mais une fois lancé, il n'y a pas de lenteur.




*Logiciel Blocklino*

Détails des icones :

| | |
|---|---|
|  | Passer du mode programmation par blocs au mode programmation par ligne de code |
|  | Nouveau programme |
|  | Ouvrir un exemple |
|  | Ouvrir un fichier de programme enregistré |
|  | Enregistrer |
|  | Préférences (paramètres et affichage) |
|  | Aide pour trouver les codes couleurs |

| | |
|---|---|
|  | Ouvre BLOCKY-WEB |
|  | Utilitaire pour fabriquer des blocs de code |
|  | Jeux pour apprendre à programmer avec Blocklino |
|  | Ouvre un moniteur série (écran pour voir les données envoyées par le microcontrôleur) |
|  | A propos |
|  | Permet de choisir le type de carte à programmer (pour nous Arduino Nano) |
|  | Choisir le port sur lequel est branché l'arduino |
|  | VERIFIER : Compile le code et vérifie s'il contient des erreurs |
|  | TELEVERSER : Transfert le programme compilé vers le microcontrôleur |
|  | EXPORTER : Créer un fichier contenant le code compilé (Nous ne nous servons pas de cette fonction) |
|  | Annuler |
|  | Rétablir |
|  | Faire une copie d'écran |
|  | Aperçu du code Arduino. Permet de voir le code générer afin d'apprendre sa syntaxe |

# Présentation de la carte Arduino Nano

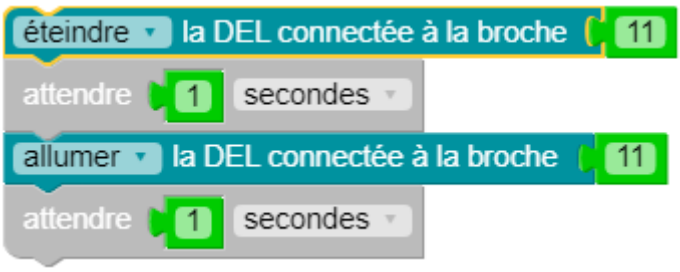
(…Descriptif à venir…)

| N° Entrée/Sortie (E/S) | N° utilisé en E/S numérique | Description | Utilisation sur la carte robot |
|---|---|---|---|
| 0 | | RX : ligne série, utilisée lors du transfert | Non utilisée |
| 1 | | TX : ligne série, utilisée lors du transfert | Non utilisée |
| 2 | 2 | E/S numérique | Moteur droit : Sens 1 |
| 3 | 3 | E/S numérique avec PWM<br>PWM = sortie « analogique » (MLI en Français) | Moteur droit : vitesse |
| 4 | 4 | E/S numérique | Moteur droit : Sens 2 |
| 5 | 5 | E/S numérique avec PWM | Moteur gauche : Sens 1 |
| 6 | 6 | E/S numérique avec PWM | Moteur gauche : vitesse |
| 7 | 7 | E/S numérique | Moteur gauche : Sens 2 |
| 8 | 8 | E/S numérique | Capteur Ultrason 1 : echo |
| 9 | 9 | E/S numérique avec PWM | Servomoteur |
| 10 | 10 | E/S numérique avec PWM | LED gauche |
| 11 | 11 | E/S numérique avec PWM | LED droite |
| 12 | 12 | E/S numérique | Capteur Ultrason 2 : echo |
| 13 | 13 | E/S numérique | LED de la carte Arduino Nano |
| A0 | **14** | Entrée analogique ou E/S numérique | Bouton |
| A1 | **15** | Entrée analogique ou E/S numérique | Capteur de ligne droit |
| A2 | **16** | Entrée analogique ou E/S numérique | Capteur de ligne arrière |
| A3 | **17** | Entrée analogique ou E/S numérique | Capteur de ligne gauche |
| A4 | **18** | Entrée analogique ou E/S numérique | Capteur Ultrason 1 : trigger |
| A5 | **19** | Entrée analogique ou E/S numérique | Capteur Ultrason 2 : trigger |
| A6 | | Entrée analogique | Capteur analogique optionnel |
| A7 | | Entrée analogique | Capteur analogique optionnel |

**Programme : LED**

Ce programme fait :

Clignoter la LED



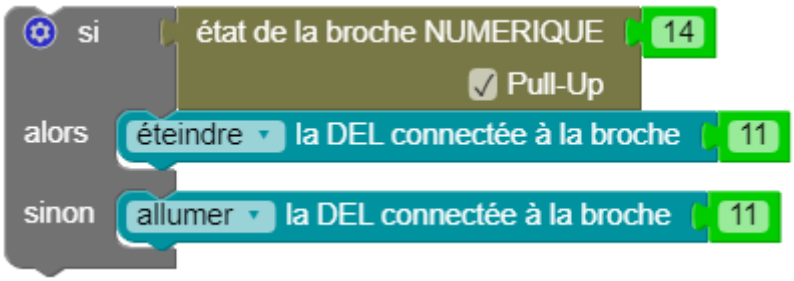

LED_A

```
void setup()
{
  pinMode(11, OUTPUT);
 }
void loop()
{
  digitalWrite(11,HIGH);
  delay(1000);
  digitalWrite(11,LOW);
  delay(1000);
}
```

**Programme : Bouton**

Ce programme fait :

La LED s'allume quand le bouton est appuyé, elle s'éteint si on relâche




BOUTON

```
void setup()
{
  pinMode( A0 , INPUT_PULLUP);
  pinMode( 11 , OUTPUT);
}

void loop()
{
  if(digitalRead(A0) == LOW)
  {
    digitalWrite(11 , HIGH);
  }
  else
  {
    digitalWrite(11 , LOW);
  }
}
```

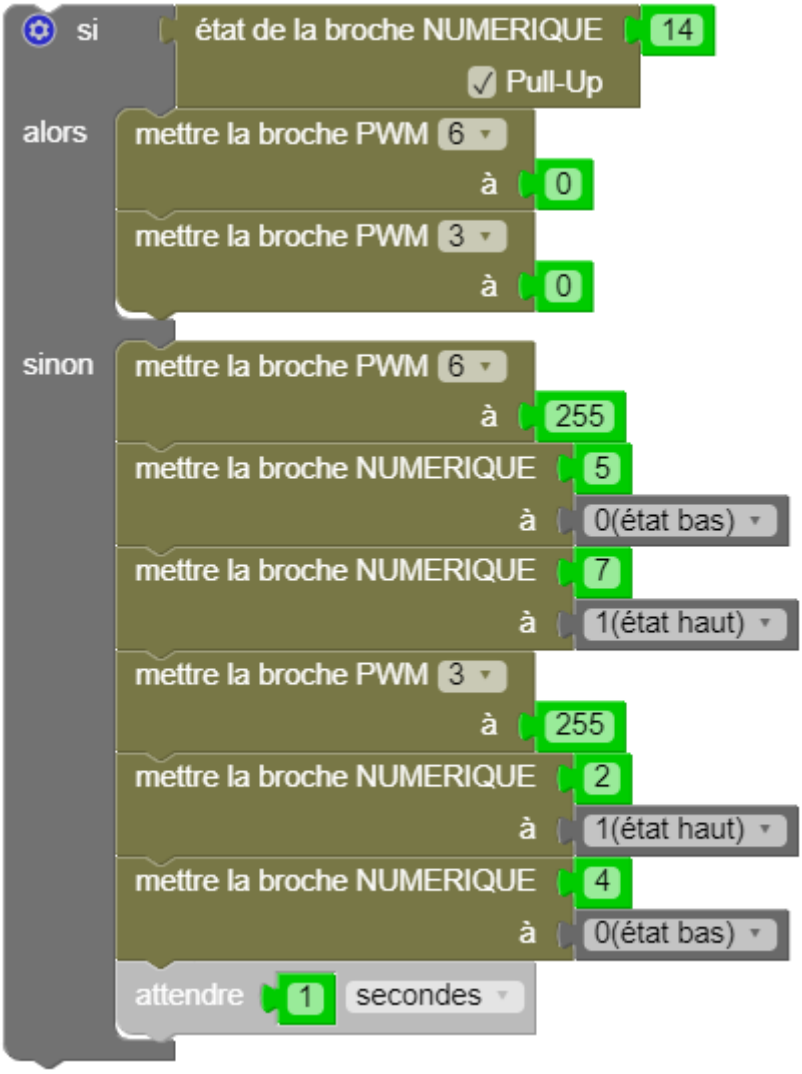
**Programme : Moteur CC**

Ce programme fait :

Lorsque l'on appuie sur le bouton, les 2 moteurs tournent dans un sens pendant une seconde.

Note : Les sorties 5, 6 et 7 gèrent le moteur droit. (6 pour la vitesse avec une valeur de 0 à 255)

Les sorties 2, 3 et 4 gèrent le moteur gauche. (3 pour la vitesse)



MOTEURS §

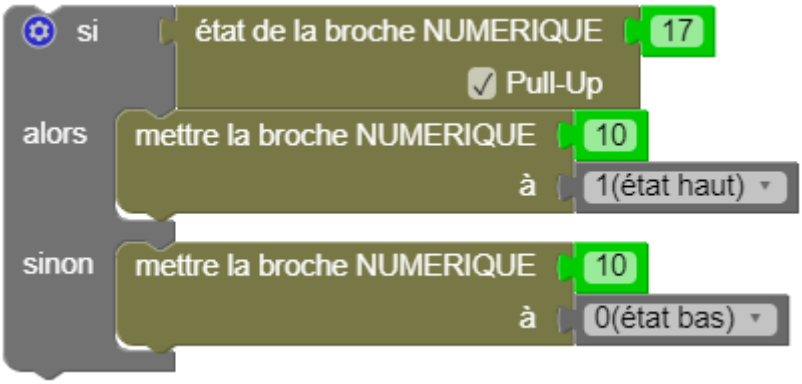
```
void setup() {
  pinMode(14, INPUT_PULLUP);
  pinMode(6, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(4, OUTPUT);
}

void loop() {
  if (digitalRead(14))     // Si le bouton est relâché
  {
//Arrêt des moteurs
    analogWrite(6, 0);
    analogWrite(3, 0);
  }
  else     // Si le bouton est appuyé
  {
//Moteur Droit
    analogWrite(6, 255);
    digitalWrite(5, LOW);
    digitalWrite(7, HIGH);
// Moteur Gauche
    analogWrite(3, 255);
    digitalWrite(2, HIGH);
    digitalWrite(4, LOW);
// Attente 1 seconde
    delay(1000);
  }
}
```

**Programme : Capteur de ligne**

Ce programme fait :

Quand le capteur de ligne voit du noir, la LED s'allume




LIGNE_BLANCHE

```
void setup() {
  pinMode(17, INPUT_PULLUP);
  pinMode(10, OUTPUT);
}

void loop() {
  if (digitalRead(17))
  {
    digitalWrite(10, HIGH);
  }
  else
  {
    digitalWrite(10, LOW);
  }
}
```

**ou**

LIGNE_BLANCHE

```
void setup() {
  pinMode(A3, INPUT_PULLUP);
  pinMode(10, OUTPUT);
}

void loop() {
  if (digitalRead(A3) == HIGH)
  {
    digitalWrite(10, HIGH);
  }
  else
  {
    digitalWrite(10, LOW);
  }
}
```

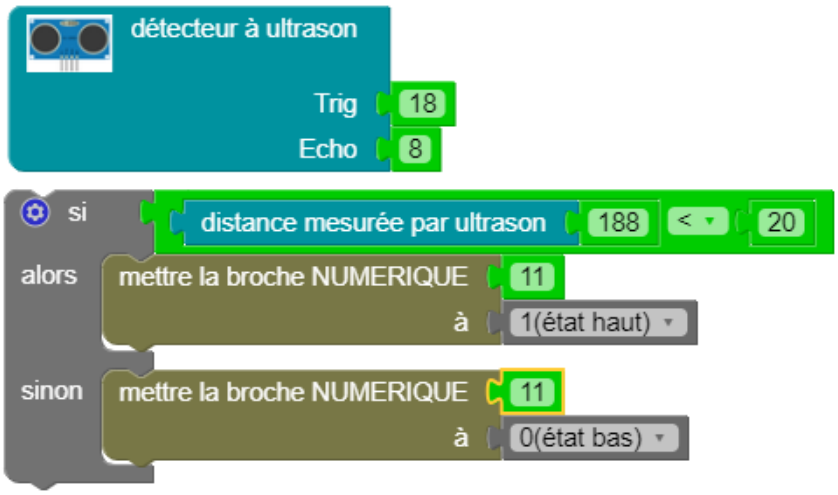
Note : On peut voir dans les deux programmes ci-dessus que **A3** est égal à **17**

Et que `if (digitalRead(A3) == HIGH)` est égal à `if (digitalRead(A3))`

**Programme : Capteur ultra-son**

Ce programme fait :
Quand le capteur ultrasons voit un obstacle à moins de 20cm, la LED s'allume.

Pour trouver le bloc Ultra-son cliquer sur ⚙, choisir le niveau « initié » et cocher « Capteurs »




CAPTEUR_ULTRASON

```
long distance;
int trigPin = A4;
int echoPin = 8;

void setup()
{
  pinMode(11, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop()
{
  mesureDistance();   // Appel à la fonction
  if(distance < 20)   // Si la distance est inférieure à 20cm
  {
    digitalWrite(11 , HIGH);
  }
  else
  {
    digitalWrite(11 , LOW);
  }
}

int mesureDistance()  // Fonction de mesure de distance
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(20);
  digitalWrite(trigPin, LOW);
  distance = pulseIn(echoPin, HIGH);
  distance = distance / 59;
  return distance;
}
```

**Programme : Servo-moteur**

Ce programme fait :

Le servo-moteur tourne dans un sens, puis dans l'autre avec un angle de 90° pendant 1 seconde


SERVO

```
#include <Servo.h>

Servo monServo;  // créer un objet servo pour controler un servomoteur

void setup() {
  monServo.attach(9);  // attacher le servo sur la patte 9
}

void loop() {
  monServo.write(0);  // tourne le servo à 0°
  delay(1000);
  monServo.write(90); // tourne le servo à 90°
  delay(1000);
}
```