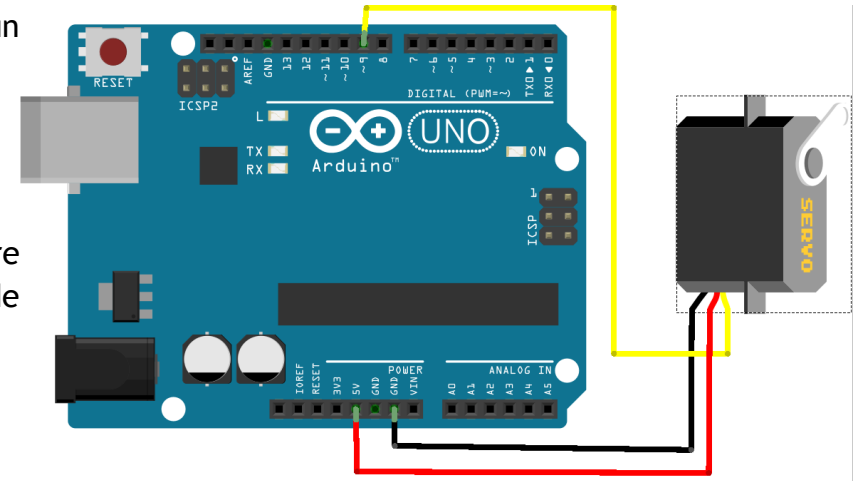
3 Troisième partie : piloter un servomoteur

```
/* Améliore la lisibilité du programme
 * en rajoutant un en-tête en commentaire comportant
 * ton nom + prénom + classe, la date,
 * la version du programme et ce qu'il fait.
*/
```

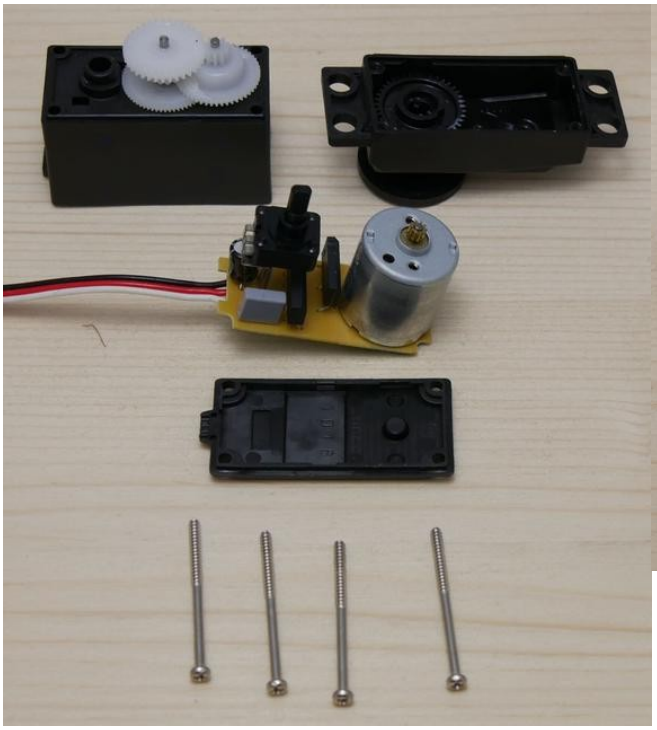
En électronique, un servomoteur est un moteur géré en position (et non en rotation).

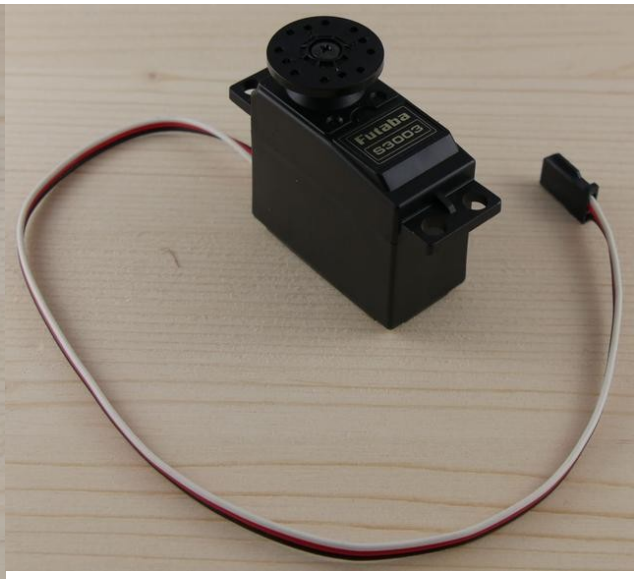
Il est commandé grâce à un ensemble de 3 fils :

- une masse [fil noir] ;
- une alimentation [fil rouge] ;
- un fil de commande [autre couleur] qui reçoit l'ordre de position.

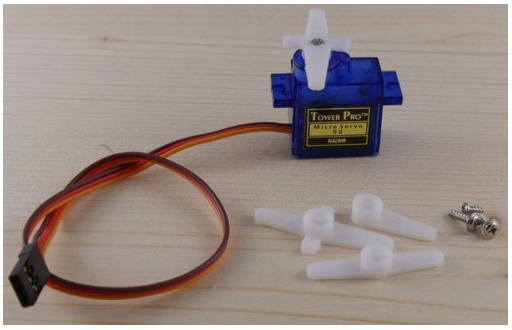


Il existe divers types de servomoteurs, de taille, poids et couple (force) différents. La photographie ci-dessous présente un servomoteur très classique en modélisme : le Futuba S3003.





On utilisera un autre servomoteur, communément appelé "servomoteur 9 grammes", par souci de consommation électrique et d'intégration car bien plus petit.



3.1 Essuie-glace

1. Connecte-toi sur Tinkercad et créé un nouveau circuit.

2. Place une carte « Arduino Uno R3 » et un servomoteur « Micro Servo ».

3. Relie le servomoteur à la carte Arduino sur la patte 9, sans oublier l'alimentation *(voir schéma au début)*.

> Comme le servomoteur demande des instructions techniques précises, nous allons utiliser des codes déjà préparés et rassemblés dans ce qu'on appelle une bibliothèque. Au début du programme, il faut donc dire que nous faisons appel à elle.
>
> Tape dans Tinkercad le programme ci-dessous en faisant attention aux majuscules/minuscules, aux virgules, etc (*inutile de recopier ce qui est après // car ce sont des commentaires qui ne seront pas pris en compte par la carte*) :

```
#include <Servo.h> //on inclut la bibliothèque pour piloter un servomoteur
Servo monServo; //on créée l'objet monServo

void setup() {
    monServo.attach(9); //on définit la broche (pin) utilisée par le servomoteur
}

void loop() {
    //on crée une variable position qui prend des valeurs entre 0 à 180 degrés
    for (int position = 0; position <=180; position ++) {
        //le bras du servomoteur prend la position de la variable position
        monServo.write(position);
        //on attend sinon le servomoteur n'a pas le temps de bouger
        delay(15);
    }
    //cette fois la variable position passe de 180 à 0°
    for (int position = 180; position >=0; position -) {
        //le bras du servomoteur prend la position de la variable position
        monServo.write(position);
        delay(15);
    }
}
```

4. Raccorde le servomoteur sur la carte réelle, copie-colle le code dans le logiciel Arduino et téléverse le pour valider ton programme.

3.2 Déclenchement de la rotation

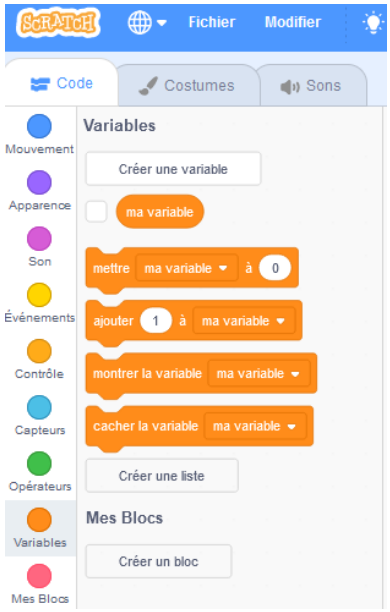
Nous allons rajouter un bouton poussoir pour modifier l'angle du servomoteur. Reprends le montage de la séance précédente pour te rappeler comment câbler un **bouton poussoir** (*sans les diodes bien sûr*) **sur la broche 8** et rajoute le **servomoteur sur la broche 9**.

Tout simplement nous allons faire exécuter un aller-retour SI on appuie sur le bouton poussoir :

```
#include <Servo.h>                    // permet de faire appel à la bibliothèque Servo

Servo monservo;                      // crée une constante appelée « monservo » utilisable par la bibliothèque Servo.
                                     // (vous pouvez la nommez comme vous voulez, exemple : servo_1, mais pensez à le remplacer dans le programme).

int bouton = 8;                      // déclare le bouton poussoir sur la PIN 8.
int etatbouton = 0;                  // variable représentant le bouton, soit ouvert ou fermé, ici au départ à 0 donc fermé.

void setup() {
   pinMode (bouton, INPUT) ;         // on indique que le bouton est une entrée.
   monservo.attach(9);               // commande de la bibliothèque Servo, qui consiste à « attacher » le servomoteur à la PIN 9
}

void loop() {
   etatbouton = digitalRead (bouton);   // On lit l'état du bouton pour savoir s'il est ouvert ou fermé

   if (etatbouton==HIGH) {              // Si la variable "etatbouton" est ouverte, à l'état haut, donc laisse passer le courant.
         monservo.write (90);           // alors le servomoteur prend un angle de 90°.
         delay (5000) ;                 // on attend un délais de 5 secondes
         monservo.write (0);            // le servomoteur revient à l'angle 0, sa position de départ.
         delay (150);                   // crée un délais de 150 millisecondes avant de refaire la boucle et de tester à nouveau l'état du bouton.
         }
}
```

Cette fois nous faisons appel à des **variables**.

Rappel dans Scratch c'est ça :



Pour mémoriser sur quelle broche est connecté le bouton poussoir, on stocke le numéro de la broche dans la variable 'bouton' :

int bouton = 8 ;

Comme tu peux le voir, dans la même ligne de code on lit l'état du bouton poussoir grâce à **digitalRead(bouton)** et on le stocke dans une autre variable nommée '**etatbouton**'.

Teste ton programme dans Tinkercad, câble la maquette réelle et téléverse ton programme dans la carte Arduino.

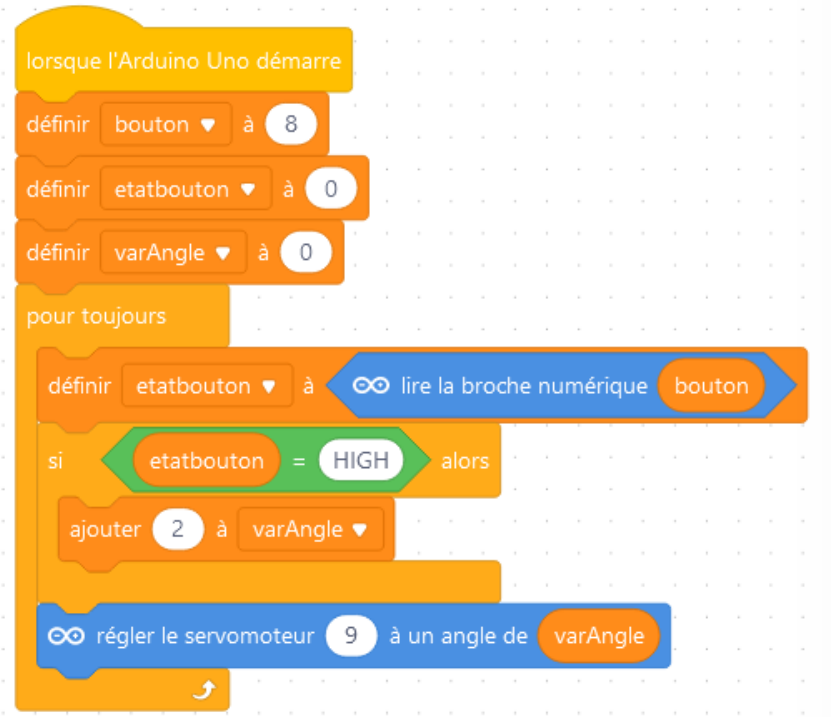
Dans la déclaration de la broche sur laquelle est connecté le bouton poussoir, change le 'INPUT' par 'INPUT_PULLUP'.

Que constates-tu comme changement ?

3.3 Pilotage de l'angle

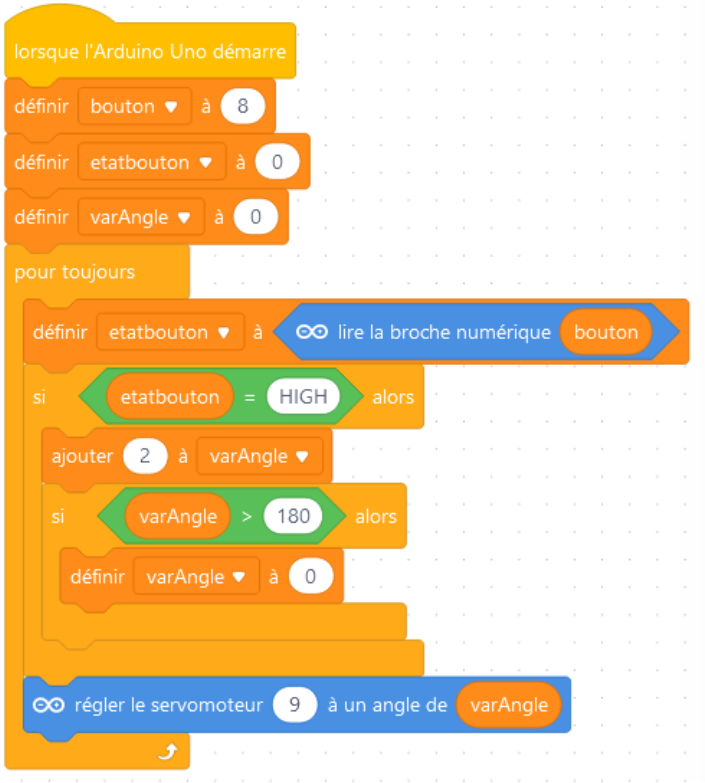
À chaque fois qu'on appuie sur le bouton poussoir, le servomoteur augmente de 2°. Tu auras besoin d'une variable supplémentaire, rajoute '**int varAngle = 0** ;'

L'algorithme à respecter est le suivant (*on ne voit que la partie loop*) :



Teste ton programme dans Tinkercad, câble la maquette réelle et téléverse ton programme dans la carte Arduino.

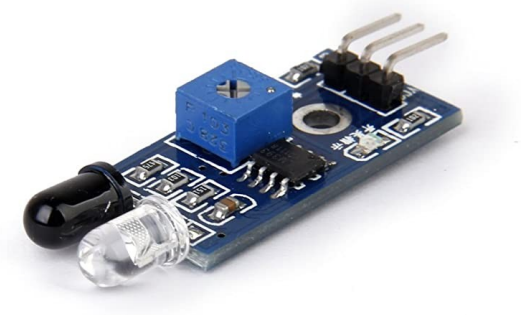
Mais pour éviter la casse il faut vérifier qu'on ne dépasse pas 180°, si varAngle > 180 alors il revient à 0°. L'algorithme devient :



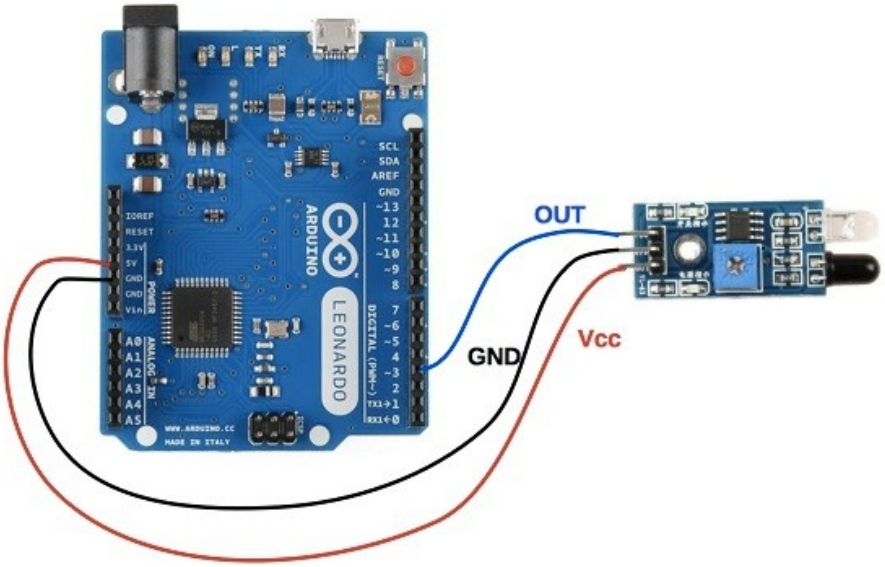
Teste ton programme dans Tinkercad, câble la maquette réelle et téléverse ton programme dans la carte Arduino.

3.4 Déclenchement sans contact

Nous allons remplacer le bouton poussoir par un capteur d'obstacle infrarouge :



Le montage va être aussi simple que ça :



Ne garde que le servomoteur et rajoute le capteur d'obstacle IR (*pas forcément sur la broche 3 comme la photo ci-dessus…*).

Sans rien changer au programme dans la carte Arduino, câble la maquette réelle et vérifie ce qu'il se passe.