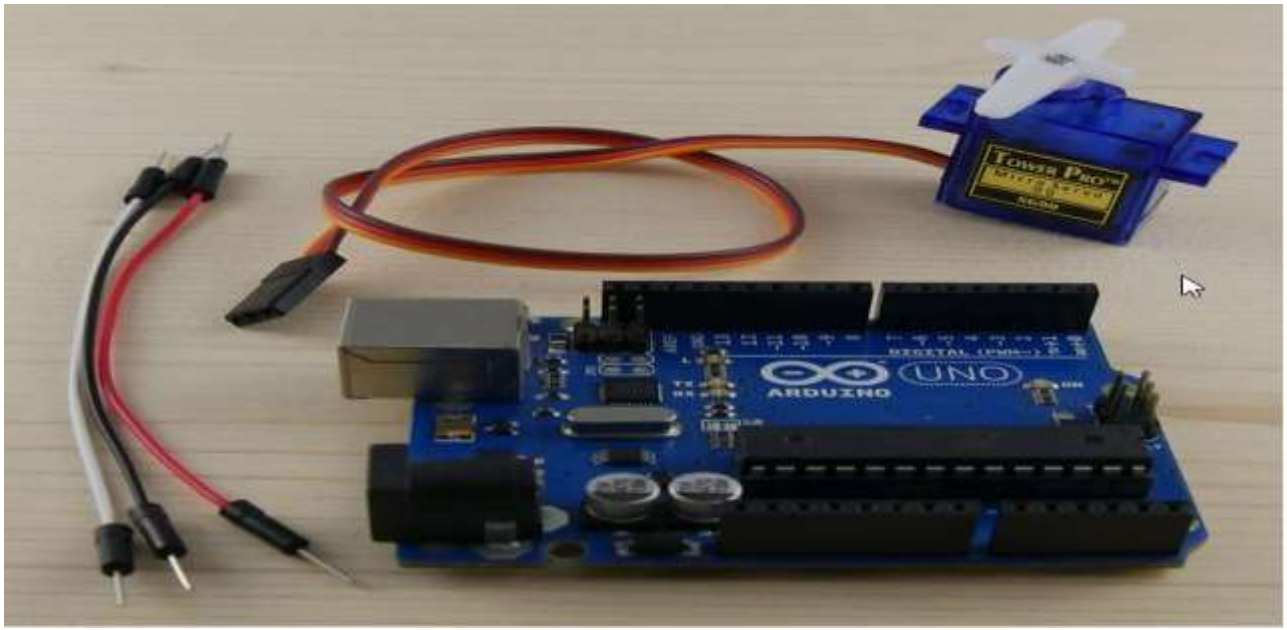
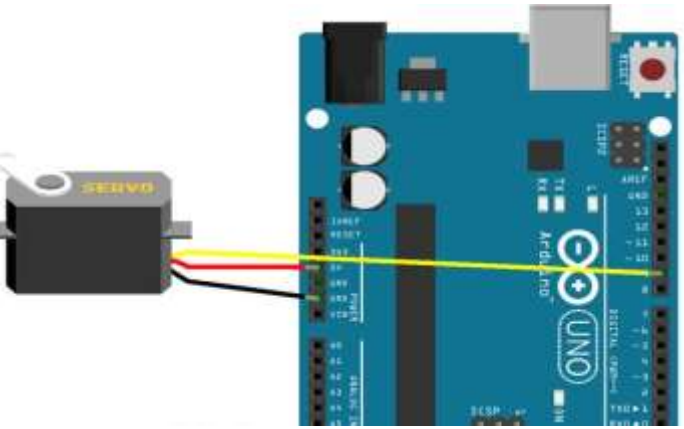
TP Mécatronique

1) Servo-moteur with arduino uno :

Schema :



Montage :

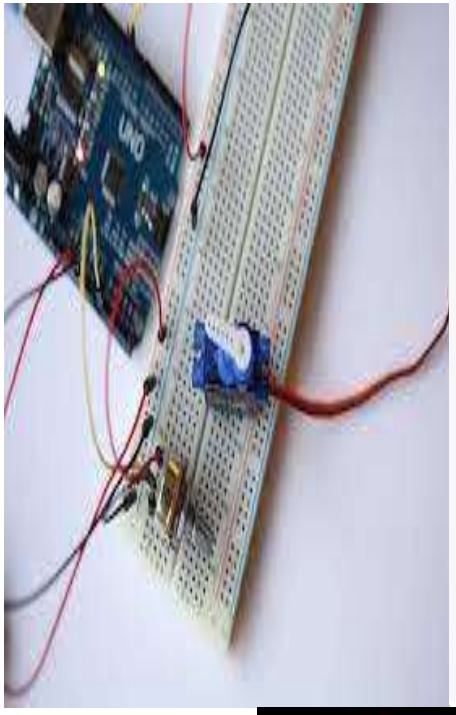


Code :

```
 * Exemple de code pour un servomoteur, il fait faire des va-et-vient à la tête du servomoteur.
 */

 /* Inclut la lib Servo pour manipuler le servomoteur */
#include <Servo.h>

/* Créer un objet Servo pour contrôler le servomoteur */
Servo monServomoteur;

void setup() {

  // Attache le servomoteur à la broche D9
  monServomoteur.attach(9);
}

void loop() {

  // Fait bouger le bras de 0° à 180°
  for (int position = 0; position <= 180; position++) {
    monServomoteur.write(position);
    delay(15);
  }

  // Fait bouger le bras de 180° à 10°
  for (int position = 180; position >= 0; position--) {
    monServomoteur.write(position);
    delay(15);
  }
}
```

2) Servo-moteur with potentiomètre

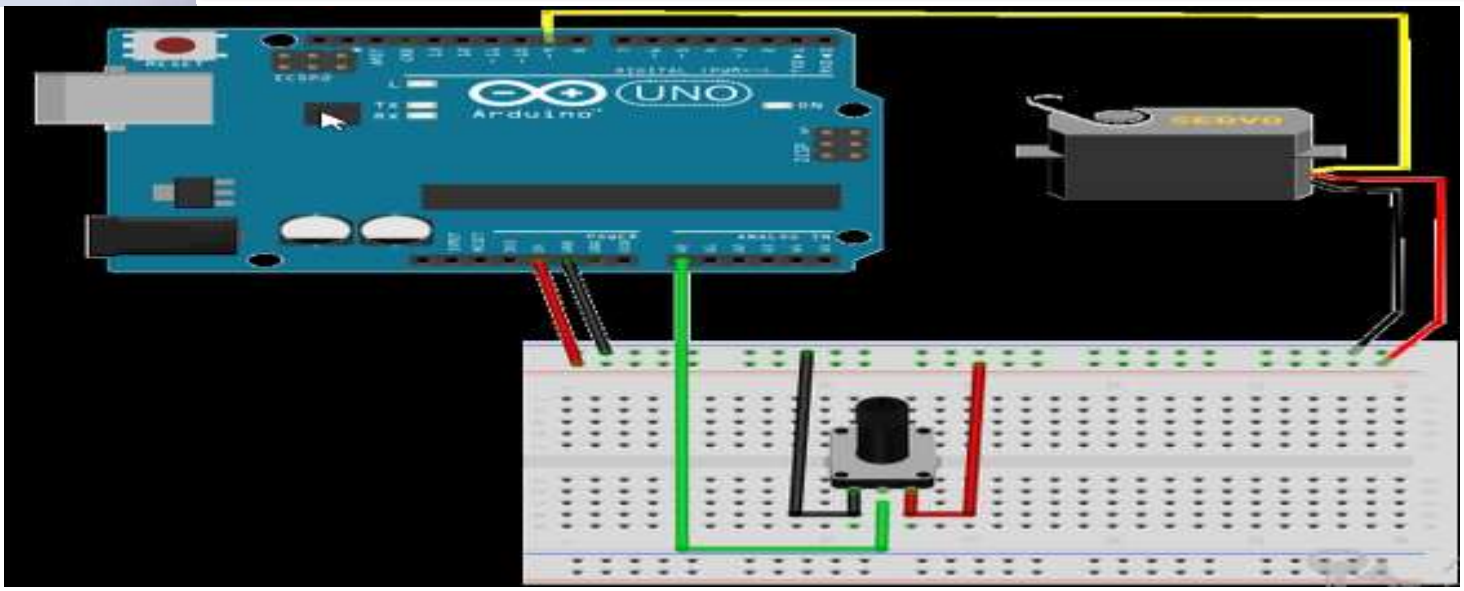
Schema :



code :

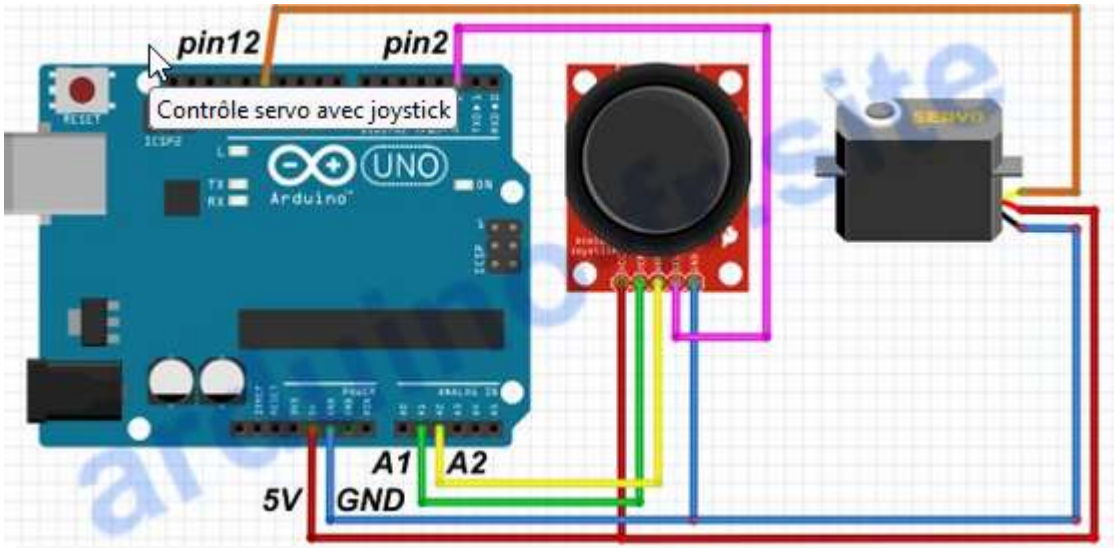
```
#include "Servo.h"

Servo servo; // création de l'objet "servo"

void setup() {
   servo.attach(10); // attache le servo au pin spécifié
   pinMode(A1, INPUT);
   Serial.begin(9600); // ouvre le port série
}

void loop() {
   int val = analogRead(A1); // lit la valeur actuelle du potentiomètre
   // mise à l'échelle pour renvoyer la position entre 0 et 180°
   val = map(val, 0, 1023, 0, 180);

   Serial.println(val);
   Serial.println();

   servo.write(val);
   delay(100); // attend 100 ms
}
```

Montage :



3) Servo-moteur with analog joystick

Montage :

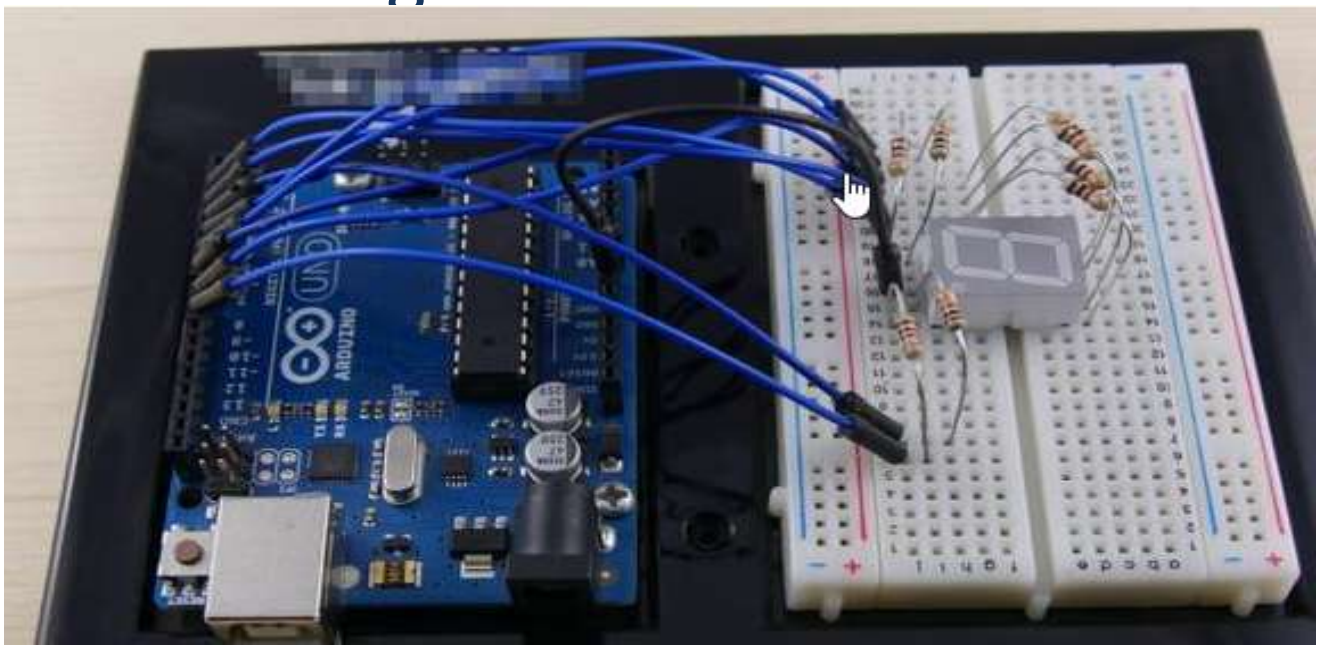


code :

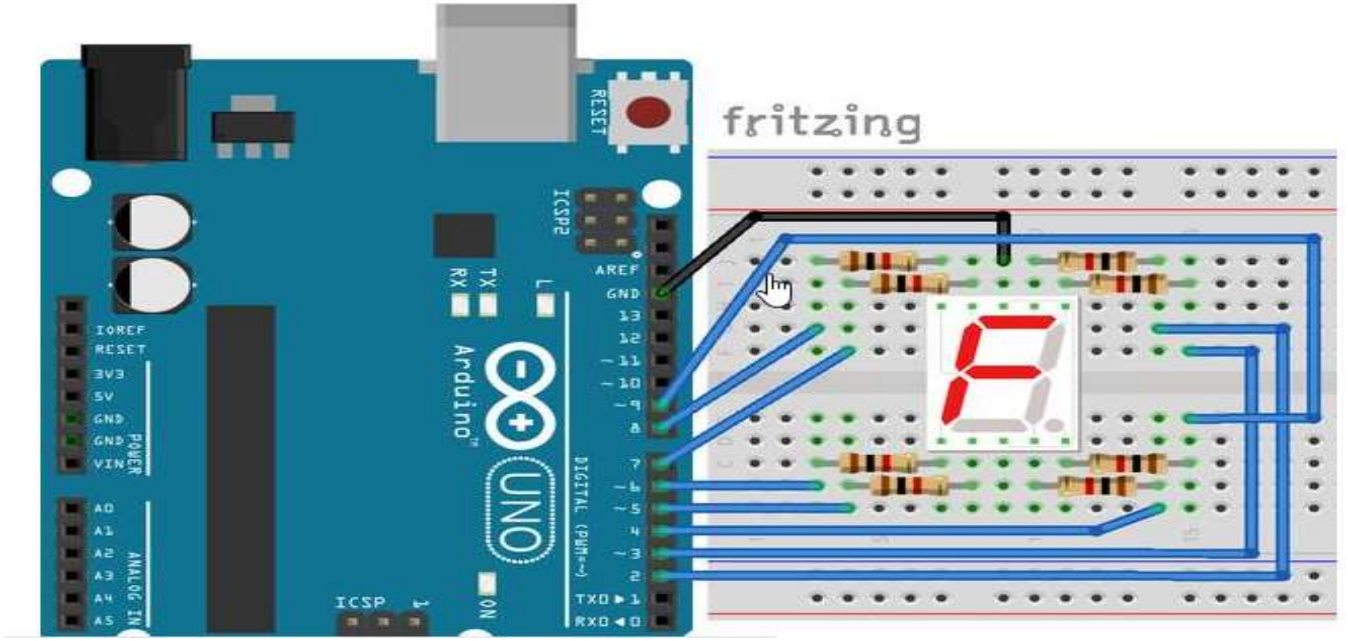
```
#define pinX A1

#include "Servo.h"
Servo servo1;

void setup() {
    pinMode(pinX, INPUT);

    servo1.attach(12);
}

void loop() {
    int X = analogRead(pinX);

    X = map(X, 0, 1023, 0, 180);

    servo1.write(X);
}
```

4) Afficheur 7-segments

Schema :



Montage :



Code :

```
/** Fonction permettant d'afficher un chiffre sur un afficheur 7 segments */
void affiche_chiffre_7seg(byte chiffre, byte dp) {

  /* Simple sécurité */
  if (chiffre > 15)
    return; // Accepte uniquement des valeurs de 0 à 15.

  /* Conversion chiffre -> états des segments */
  byte segments = LUT_ETATS_SEGMENTS[chiffre];

  /* Affichage */
  digitalWrite(PIN_SEGMENT_A, bitRead(segments, 0));
  digitalWrite(PIN_SEGMENT_B, bitRead(segments, 1));
  digitalWrite(PIN_SEGMENT_C, bitRead(segments, 2));
  digitalWrite(PIN_SEGMENT_D, bitRead(segments, 3));
  digitalWrite(PIN_SEGMENT_E, bitRead(segments, 4));
  digitalWrite(PIN_SEGMENT_F, bitRead(segments, 5));
  digitalWrite(PIN_SEGMENT_G, bitRead(segments, 6));
  digitalWrite(PIN_SEGMENT_DP, dp);
}
```

```
/** Fonction loop() */
void loop() {
  static byte chiffre = 0;
  static byte etat_dp = 0;

  /* Affiche le chiffre */
  affiche_chiffre_7seg(chiffre, etat_dp);

  /* Incrémente le chiffre de 0 à 15 */
  if (++chiffre == 16) {
    chiffre = 0;
  }

  /* Fait clignoter le point décimal (inverse l'état à chaque fois) */
  etat_dp = !etat_dp;

  /* Délai pour la démo */
  delay(1000);
}
```