servo motor

Not all types of actuators are equally suitable to become a robot component. Alternating current engines are too big, DC motion solutions are too fast and low-torque. On the other hand, steppers and servo motors seem to have the qualities essential to build a robot—consistently high mechanical work output, precision positioning, and compact size The engines are similar in that they both ensure flexible regulation of speed, position, and force output. However, differences in their capabilities, control logic, and prices are significant. Below are the key ones, from the perspective of robot applications.

Servos

- Motion accuracy: Closed-loop mechanism, with an encoder or another device giving accurate feedback on the actual motor path.

- Torque and speed: Consistent torque output over a wide speed range; copes well with short-term power bursts (e.g., due to rapid acceleration).

- Integration and operation issues: Costly; simple integration, but extra time needed to set the drives properly.

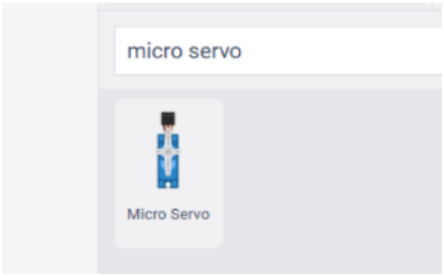
Why robots need a servo motor ?

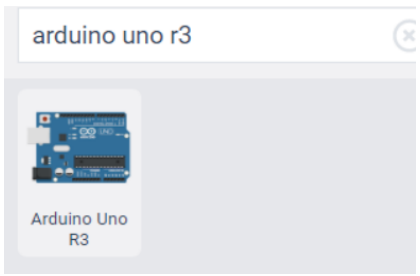
- Similar to a human body, robotic links and robotic joints form the skeleton of a robot. Servo motors are the muscles that move and rotate the robotic links and robotic joints to alter the machine's position and movement.

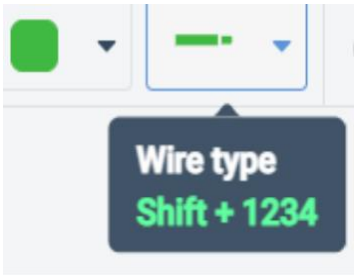
- In contrast to a regular motor that rotates continuously and halts only when we turn it off, a servomotor can stop at any required position and hold it until it receives another command. With an integrated feedback mechanism, servos are fully self-contained, which allows for implementing speed and angle control in a robot arm or a robotic hand.

- Boasting a high repeatability rate, the robotic actuator maintains the precision of a robot's movements throughout numerous work cycles. Energy-efficient and easy-to-service, servomotors make it possible to reduce the cost of operating and maintaining a robot.

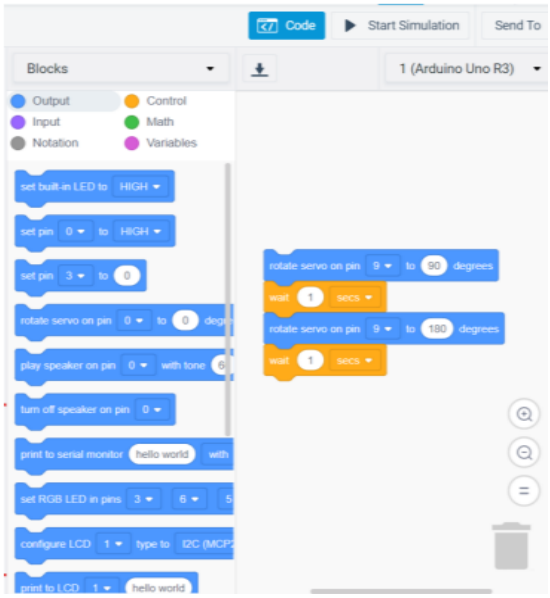
simple servo motor tools :

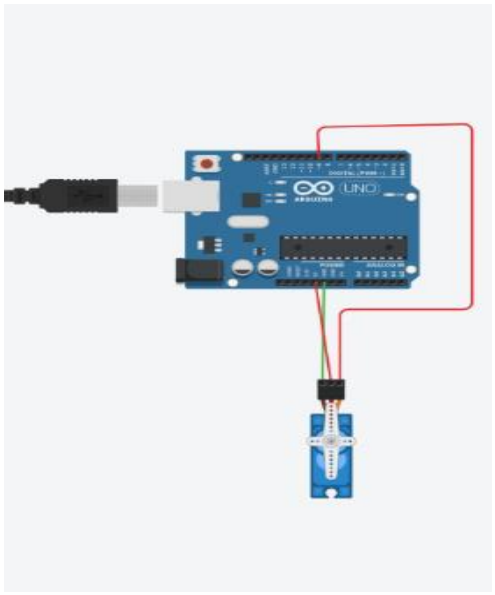




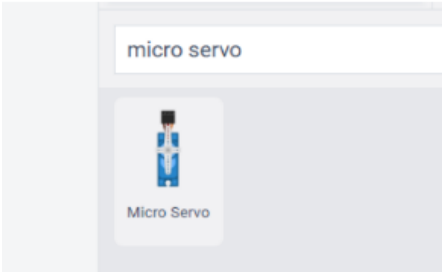


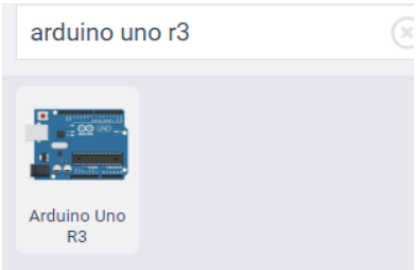
The code:

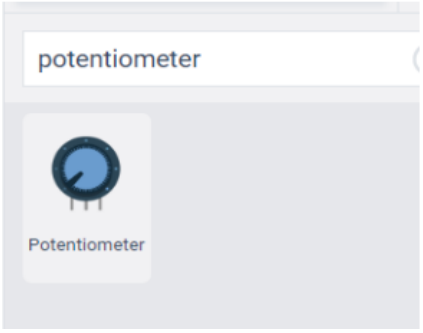


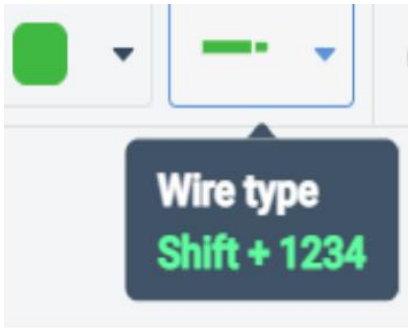


complex servo motor for Robotic Arm









The code:

```
#include <Servo.h>
Servo servo0;
Servo servo1;
Servo servo2;
Servo servo3;
int sensorvalue0;
int sensorvalue1;
int sensorvalue2;
int sensorvalue3;
```

Definition of variables

```
void setup()
{
  pinMode(A0,INPUT);
  pinMode(3,OUTPUT);
  servo0.attach(3);

  pinMode(A1,INPUT);
  pinMode(5,OUTPUT);
  servo1.attach(5);

  pinMode(A2,INPUT);
  pinMode(6,OUTPUT);
  servo2.attach(6);

  pinMode(A3,INPUT);
  pinMode(9,OUTPUT);
  servo3.attach(9);
}
```

Definition of outputs and inputs

```
void loop()
{
  sensorvalue0 = analogRead(A0);
  sensorvalue0 = map(sensorvalue0, 0, 1023, 0, 180);
  servo0.write(sensorvalue0);
  sensorvalue1 = analogRead(A1);
  sensorvalue1 = map(sensorvalue1, 0, 1023, 0, 180);
  servo1.write(sensorvalue1);
  sensorvalue2 = analogRead(A2);
  sensorvalue2 = map(sensorvalue2, 0, 1023, 0, 180);
  servo2.write(sensorvalue2);
  sensorvalue3 = analogRead(A3);
  sensorvalue3 = map(sensorvalue3, 0, 1023, 0, 180);
  servo3.write(sensorvalue3);
}
```

For operation a loop