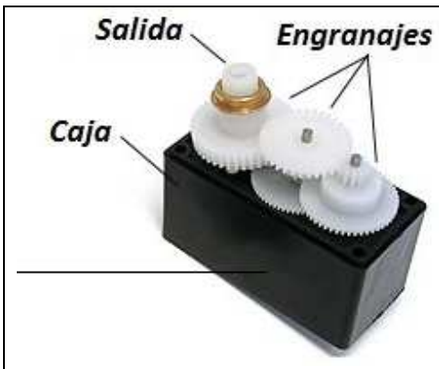
Programar posiciones en un Micro Servo Tower Pro SG90 9G

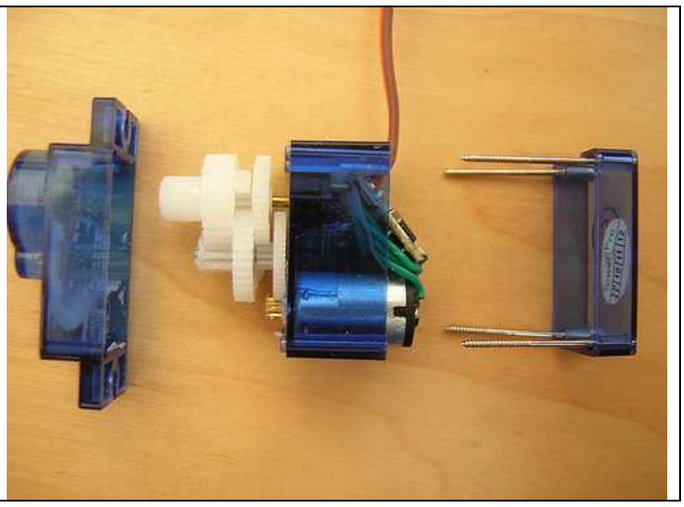
(Recopilado de Internet. Revision Prof: Bolaños DJB) Versión: 12-04-18

Los servos son motores de corriente continua (CC), pero en lugar tener un giro continuo que podamos aprovechar (para mover un molino por ejemplo), están preparados para moverse a un ángulo fijo en respuesta a una señal de control, y mantenerse fijos en dicha posición. Estos servomotores son muy frecuentes en Aero modelismo y en robótica, puesto que su funcionamiento y control son muy precisos.

Un servo principalmente está formado por un conjunto reductor (engranajes), un motor de CC y por último por un circuito de control, aunque en la práctica se comporta como un bloque funcional que posiciona su eje en un ángulo preciso en función de la señal de control.



Composición de un Servomotor



Habitualmente los servos tiene un margen de operación, es decir, pueden moverse entre 0º y ángulo dado, que suele ser de 180º. Normalmente estos pequeños servos funcionan sobre 5V y el control se realiza mediante una señal de control PWM, en la que el ancho el pulso indica el ángulo que deseamos que adopte el eje.

En este proyecto vamos a utilizar un Micro Servo SG90 Tower Pro, puesto que es ideal para las primeras experiencias de aprendizaje y prácticas con servos, ya que sus requerimientos de energía son bastante bajos y se permite alimentarlo con la misma fuente de alimentación que el circuito de control, es decir, si se conecta a un Arduino, se puede alimentar durante las pruebas desde el puerto USB del PC sin mayor problema, aunque **es recomendable alimentar el Arduino con una fuente de alimentación externa de más de 700 mA**, para evitar problemas de corriente.

NOTA: Se recomienda ver el video que acompaña esta teoría

A continuación los muestro las principales características de las que dispone:

- Dimensiones (L x W x H) = 22.0 x 11.5 x 27 mm (0.86 x 0.45 x 1.0 pulgadas)
- Peso: 9 gramos
- Longitud de cable de conector: 24.5cm
- Velocidad: 0.10 seg/60°@ 4.8V *(Sería 100 ms en recorrer 60 grados)*
- Torque: 1.8 Kg-cm @ 4.8V
- Voltaje de funcionamiento: 3.0-7.2V
- Temperatura de funcionamiento: -30□ ~ 60□
- Ángulo de rotación: 180°
- Ancho de pulso: 500-2400 µs

Cable CAFÉ: GND **Cable ROJO : 5 v** **Cable NARANJA : CONTROL** (en nuestro caso PIN 9)

Lista de Materiales:

-

- Arduino UNO Rev.3.
- Cable USB
- Micro Servo TowerPro SG90 9G (180º).
- Protoboard.
- Cables de conexión.

Código de programa:

Este programa barre constantemente en sentido horario y luego en sentido anti – horario.

```
//Programa SERVO1
//Programar posiciones en un Micro Servo Tower Pro SG90 9G.Z
// Incluir la librería Servo
#include <Servo.h>
Servo servo1;     // Declaramos la variable para controlar el servo

int pinServo = 9;    // Pin digital con PWM para el Servo
int pulsoMin = 650;  // Pulso en us para girar un ángulo de 0º
int pulsoMax = 2550; // Pulso en us para girar un ángulo de 180º

int angulo = 0; // Variable para guardar el ángulo del servo

void setup()
{
  // Señal del Servo1 conectado al pin digital 9 (PWM)--> pinServo
  // Calibración del servo para suavizar movimientos -->(pulsoMin, pulsoMax)
 // servo1.attach(pinServo, pulsoMin, pulsoMax);

}

void loop()
{
  // El Servo avanza de 0º a 180º
  for(angulo = 0; angulo <= 180; angulo++)
  {
    // Avanza 1 grado cada 10 ms
    servo1.write(angulo);
    delay(10);
  }

  // El Servo retrocede de 180º a 0º
  for(angulo = 180; angulo >= 0; angulo--)
  {
    // Retrocede 1 grado cada 10 ms
    servo1.write(angulo);
    delay(10);
  }
}
```

De la librería ***Servo.h*** hemos declarado un objeto o variable ***servo1*** (Servo servo1;) y hacemos uso de dos métodos. Por un lado el **attach**, que nos permite indicar en que **pin tenemos conectado** nuestro servo, y por otro lado el **write**, donde indicamos en qué ángulo queremos **posicionar nuestro servomotor**. Estos elementos que hemos mencionados son propios de la librería, asi lo ha definido el creador de la misma y se debe respetar.

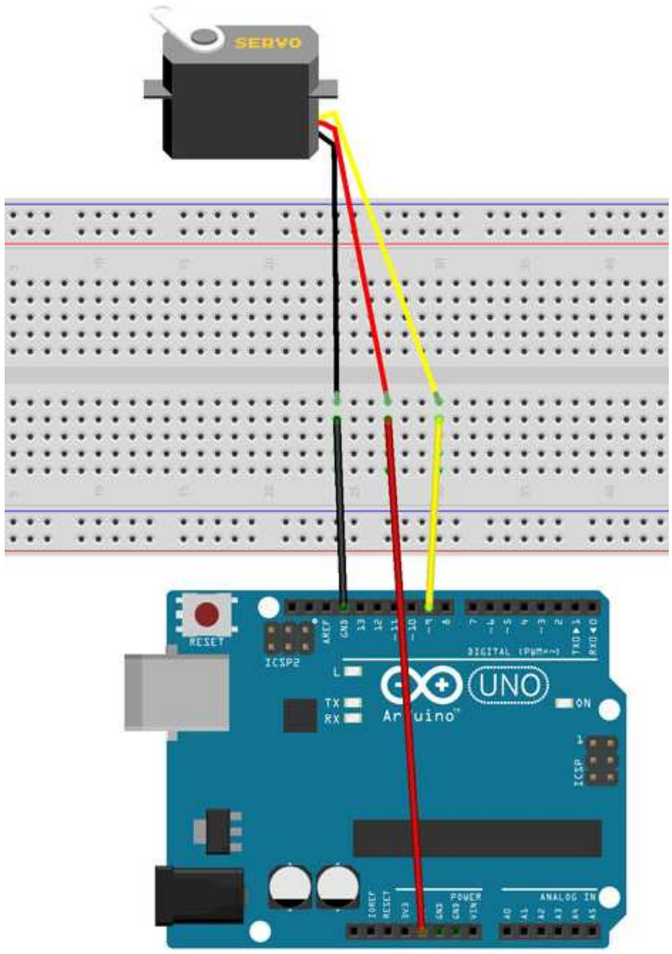
Conexionado con Arduino

Las conexiones dependerán del tipo de servomotor con Arduino que estemos utilizando.

El siguiente conexionado seria válido si usamos una fuente externa para alimentar nuestro Arduino.

Todos deben tener 3 cables.
Uno irá a tierra, otro a la alimentación de 5 Voltios y el tercero a un pin PWM.

También puedes utilizar un shield para controlar servomotores.

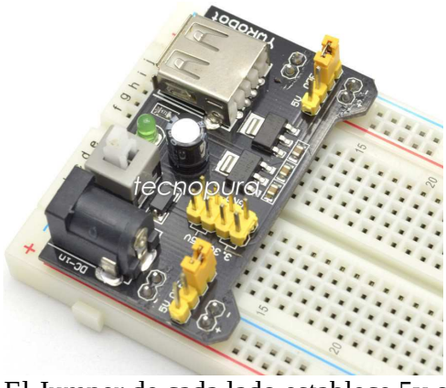


ADVERTENCIA: Es conveniente evitar alimentar el Arduino con el USB, estos puertos por lo general solo admiten hasta 500 mA y los ensayos hechos con el Micro Servo SG90 Tower Pro, dieron valores de 700 mA - 800 mA – 900 mA con picos de 1,1 A.

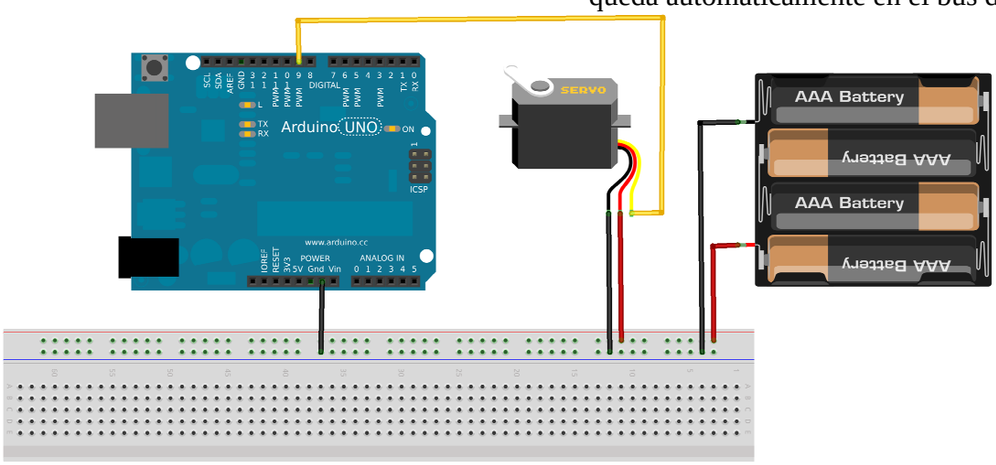
Lo mejor es alimentar con otra fuente de 5 V el servo.
Recordar desconectar el servo cuando programamos el Arduino.

Opciones de Fuente externa

Opción utilizada para el ensayo.



El Jumper de cada lado establece 5v o 3,5. Gnd queda automáticamente en el bus del proto.



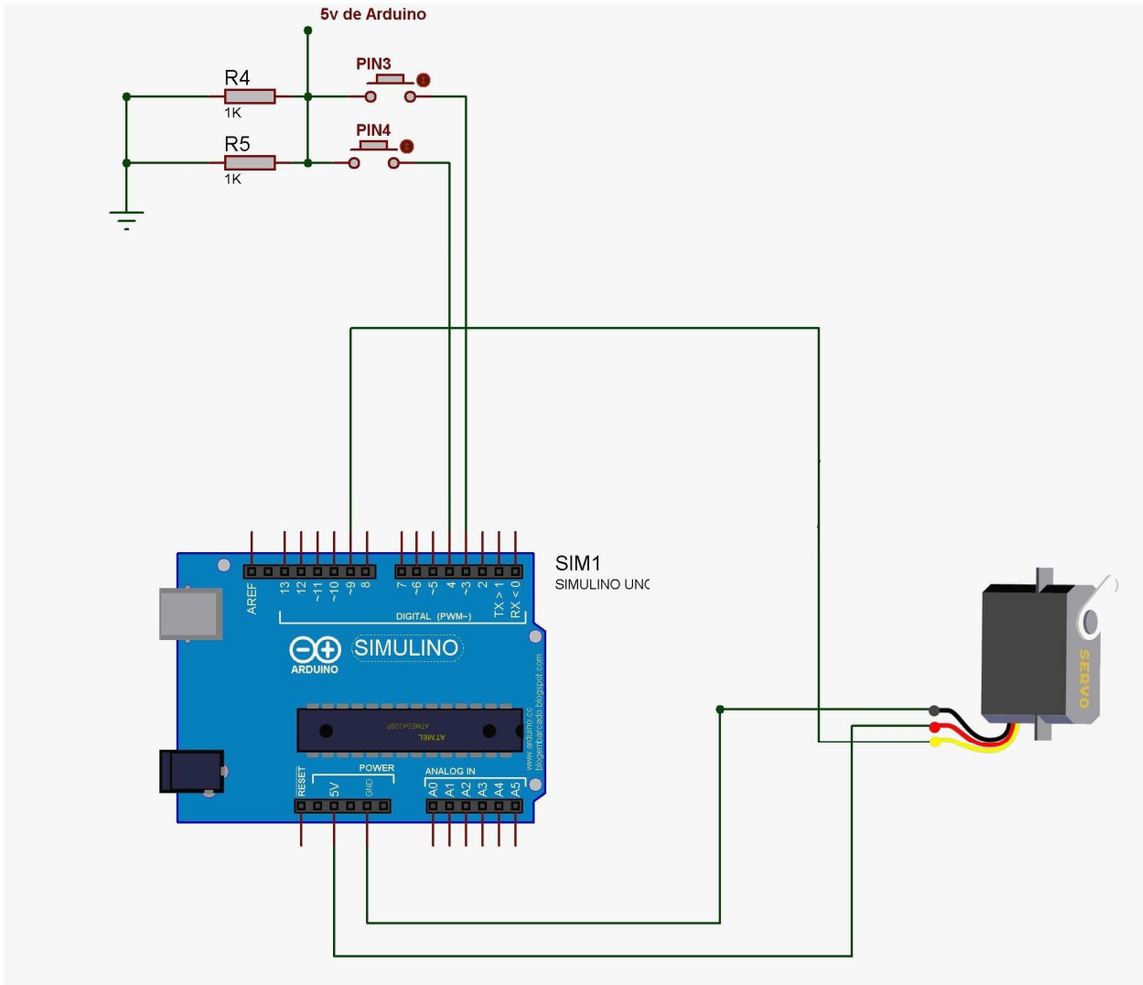
Otro ejemplo con el servo

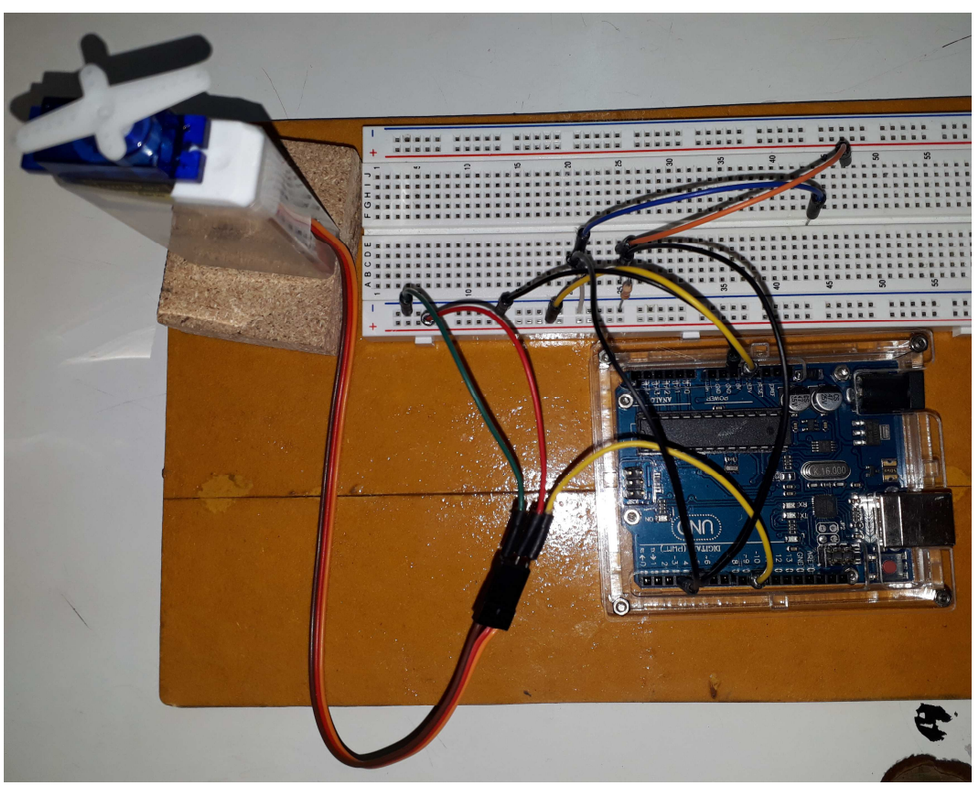
Programa ServoControl

Funcionamiento: El circuito tiene 2 pulsadores en PIN 3 y PIN 4 de Arduino, cada vez que se presiona el PIN3 (coloca un 1 en esa entrada) el servo gira 1 grado en sentido horario, mientras que si se presiona el PIN4 el servo gira 1 grado en sentido anti-horario.

```
//Programa ServoControl Control por pulsadores.
//Haciendo 1 Pin 3 gira en sentido horario
//Haciendo 1 Pin 4 gira en sentido anti- horario
//Usando Micro Servo Tower Pro SG90 9G.Z
// Incluir la librería Servo
#include <Servo.h>
Servo servo1;

int pinServo = 9;    // Pin digital con PWM para el Servo
int pulsoMin = 650;  // Pulso en us para girar un ángulo de 0º
int pulsoMax = 2550; // Pulso en us para girar un ángulo de 180º

int angulo = 0; // Variable para guardar el ángulo del servo

void setup()
{
pinMode(3, INPUT); // configura el 'pin' como entrada
pinMode(4, INPUT); // configura el 'pin' como entrada

  // Señal del Servo1 conectado al pin digital 9 (PWM)--> pinServo
  // Calibración del servo para suavizar movimientos -->(pulsoMin, pulsoMax)
  servo1.attach(pinServo, pulsoMin, pulsoMax);
}

void loop()
{
  // El Servo avanza de a un grado en sentido horario
 if ((digitalRead(3) == 1) && (angulo < 180))
  {
    // Avanza 1 grado

    servo1.write(angulo);
    angulo = angulo +1;
    delay(10);//Original 10 Al aumentar baja la velocidad
  }

  // El Servo avanza de a un grado en sentido anti-horario

 if ((digitalRead(4) == 1) && (angulo > 0))
  {
    // Retrocede 1 grado cada 10 ms

    servo1.write(angulo);
    angulo = angulo -1;
    delay(10);//Original 10 Al aumentar baja la velocidad
  }
}
```

ADVERTENCIA : Desconecte el motor para programar. No exija el puerto USB de la PC.


5v de Arduino
R4
1K
R5
1K
PIN3
PIN4
SIM1
SIMULINO UNO
SIMULINO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
SERVO



Otro ejemplo con el servo

Programa ServoBluet

Funcionamiento: El circuito tiene permite mediante una comunicación Bluetooth controlar el sentido de giro de un servo.
Se utiliza el modulo Bluetooth HC-05, cuya configuración y manejo lo puede ver en ejemplos del Tutor Arduino.

Con el objetivo de entender el funcionamiento se utilizara la APP SegundaBlue, ya ensayada anteriormente. Puede encontrar copia en RECURSOS del Tutor Arduino (SegundaBlue.apk)

Como siempre debe buscar el dispositivo al que desea conectarse, debiendo vincularse previamente, como es de costumbre la primera vez.

Cada vez que presiona ENVIAR CODIGO el servo gira 10 grados en sentido anti – horario, al menos que se encuentre al tope de esa dirección de giro.

Cada vez que presiona ENVIAR CIERRE el servo gira 10 grados en sentido horario, al menos que se encuentre al tope de esa dirección de giro.

El botón Enviar 2345678 no cumple función en este ejemplo.

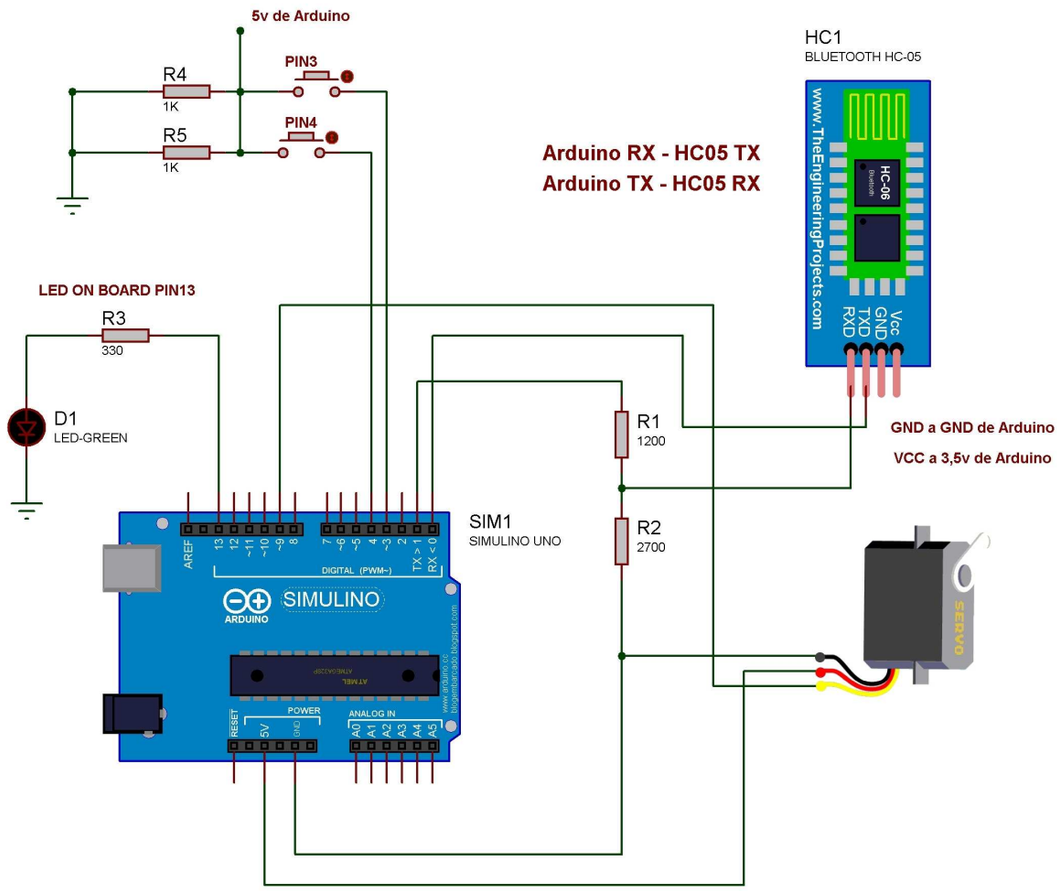
Se espera que el lector genere una APP con una mejor presentación visual. Solo utilizamos esta con fines didácticos.

El programa Arduino:

```
/* ServoBluet permite controlar el sentido de giro de un servomotor
 *  controlado desde un movil Android
 *  Se parte del ejemplo SEGUNBLU y se le incorpora el manejo
 *  del servo.
 */
//Configuracion PARA EL SERVO**********************
// Incluir la librería Servo
#include <Servo.h>
Servo servo1; // Declaramos la variable para controlar el servo (servo1
int pinServo = 9;    // Pin digital con PWM para el Servo
int pulsoMin = 650;  // Pulso en us para girar un ángulo de 0º
int pulsoMax = 2550; // Pulso en us para girar un ángulo de 180º

int angulo = 0; // Variable para guardar el ángulo del servo

//FIN SETUP para el servo*****************
```

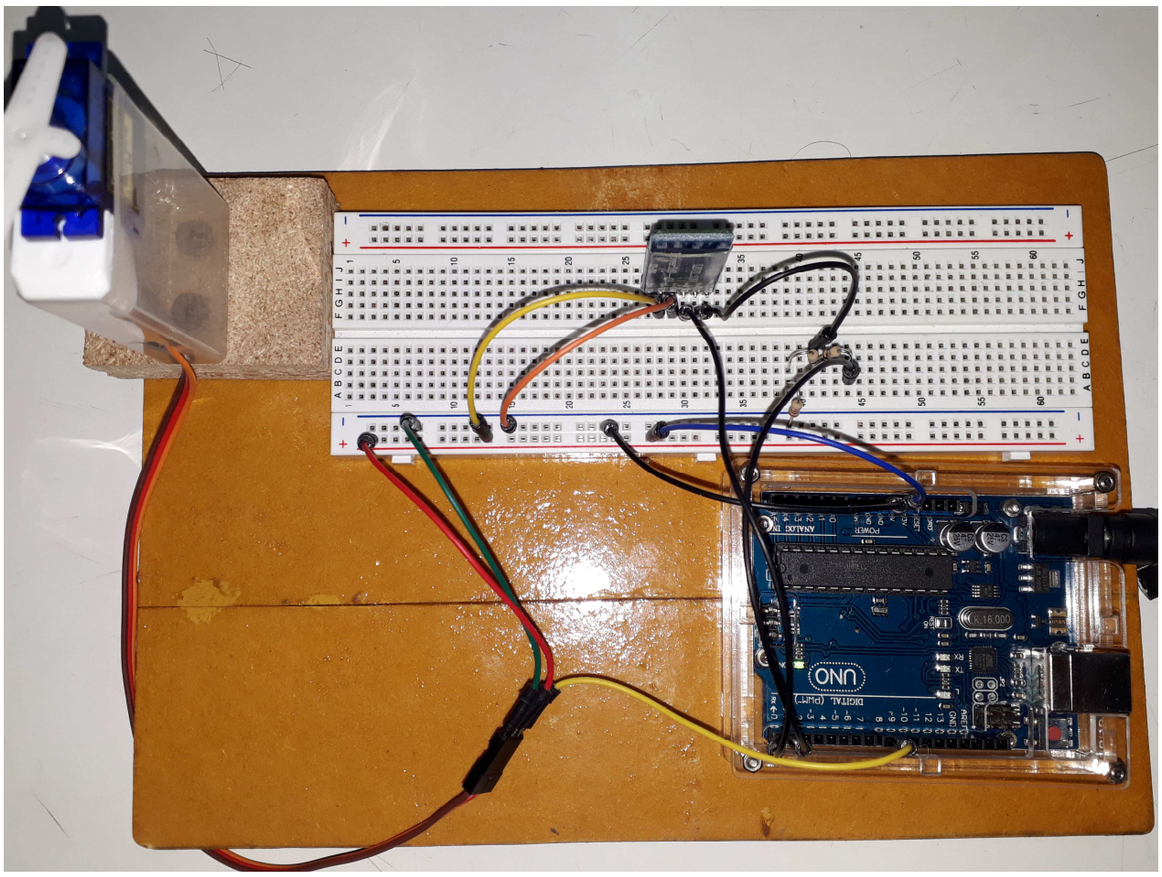
```
void setup()
{
Serial.begin(9600); //Iniciar el serial
pinMode(13, OUTPUT); // configura ‘pin’ como salida

//Setup para el servo**************************
 // Señal del Servo1 conectado al pin digital 9 (PWM)--> pinServo

  // Calibración del servo para suavizar movimientos -->(pulsoMin, pulsoMax)
  servo1.attach(pinServo, pulsoMin, pulsoMax);
//Fin setup para el servo**********************
}

void loop()
{
   if(Serial.available()>=1)
   {

      //Delay para favorecer la lectura de caracteres

    delay(22);

    //Se crea una variable que servirá como buffer
    String bufferString = "";

    /*
     * Se le indica a Arduino que mientras haya datos
     * disponibles para ser leídos en el puerto serie
     * se mantenga concatenando los caracteres en la
     * variable bufferString
     */

    while (Serial.available()>0) {
      bufferString += (char)Serial.read();
    }


    long entrada = bufferString.toInt(); //Se carga lo leido en la variable entrada


//Para el servo**************************************
  // El Servo avanza de a un grado en sentido horario
 if ((entrada == 12345) && (angulo < 180))
      {

       digitalWrite(13,HIGH );//LED testigo

     // Avanza 10 grados
    servo1.write(angulo);
    angulo = angulo +10; //gira 10 grados
    delay(10);//Original 10 .Al aumentar baja la velocidad


      }

     // El Servo avanza de a 10 grados en sentido anti-horario

        else   if ((entrada == 0) && (angulo > 0))

     {
       digitalWrite(13,LOW );//LED testigo
```

```
        // Retrocede 10 grados cada 10 ms

    servo1.write(angulo);
    angulo = angulo -10;
    delay(10);//Original 10 Al aumentar baja la velocidad

      }

//FIN para el servo ************************************
   }
}
```



MUY IMPORTANTE:

Debe desconectar el servo para programar y tambien el modulo Bluetooth si lo tiene conectado a Tx y Rx del Arduino. Si quiere evitar este inconveniente, es posible mediante una librería ya vista, mudar la conexión del modulo a otros pines.


UNO
