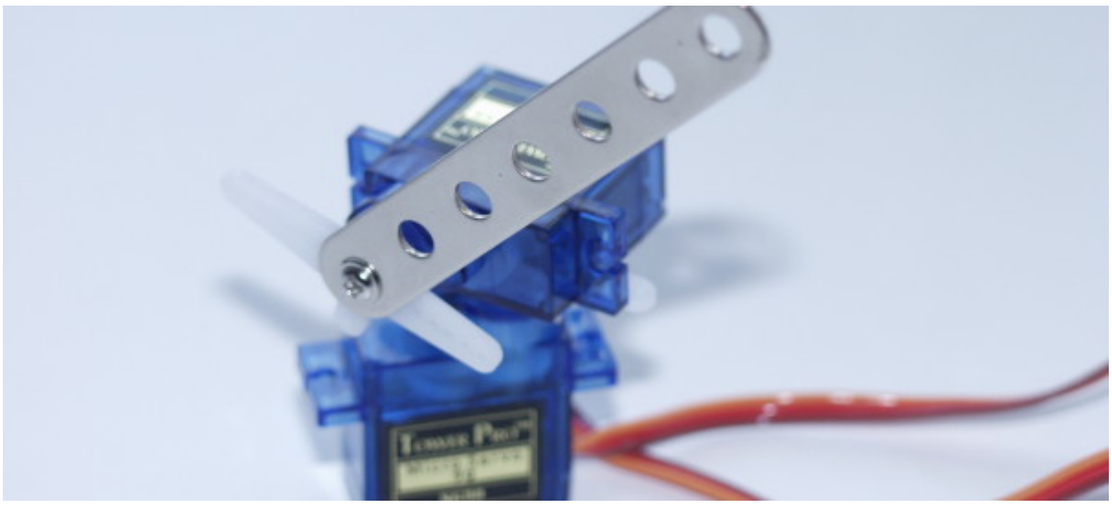
Brazo-robotizado con 2 servos y controlado con Arduino

Fuente: https://www.web-robotica.com Adaptación Prof: Bolaños DJ (*Versión 06-8-19*)



Un servomotor, también llamado servo, es un dispositivo similar a un motor de corriente continua que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición.
Hemos creado este sencillo «Brazo-robotizado» para explicar el funcionamiento básico de los servos y como podemos controlarlos con Arduino, en modo automático, controlado con pulsadores o por cualquier señal externa.

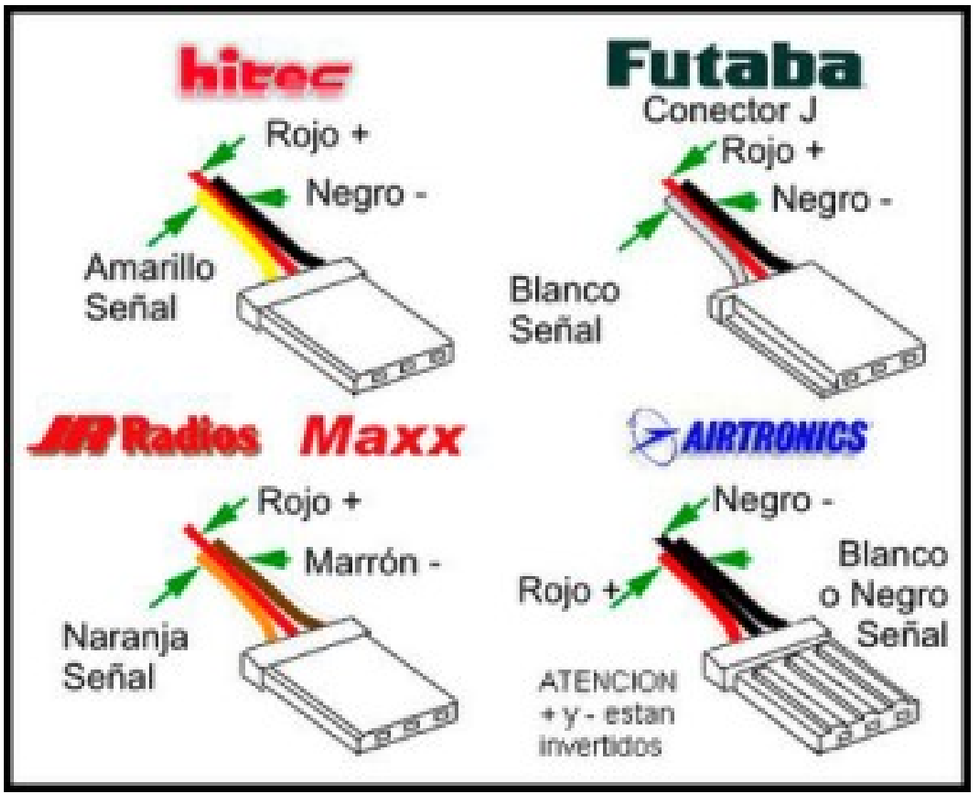
Un servomotor es un motor eléctrico que puede ser controlado tanto en velocidad como en posición.
Es posible modificar un servomotor para obtener un motor de corriente continua que, si bien ya no tiene la capacidad de control del servo, conserva la fuerza, velocidad y baja inercia que caracteriza a estos dispositivos.

Características de un servomotor

Está conformado por un motor, una caja reductora y un circuito de control. También potencia proporcional para cargas mecánicas. Un servo, por consiguiente, tiene un consumo de energía reducido.
La corriente que requiere depende del tamaño del servo. Normalmente el fabricante indica cuál es la corriente que consume. La corriente depende principalmente del par, y puede exceder un amperio si el servo está enclavado.

En otras palabras, un servomotor es un motor especial al que se ha añadido un sistema de control (tarjeta electrónica), un potenciómetro y un conjunto de engranajes.
Con anterioridad los servomotores no permitían que el motor girara 360 grados, solo aproximadamente 180, sin embargo, hoy en día existen servomotores en los que puede ser controlada su posición y velocidad en los 360 grados. Los servomotores son comúnmente usados en robots, y en modelismo, en aviones, barcos, helicópteros y trenes para controlar de manera eficaz los sistemas motores y los de dirección.

Como controlar un servomotor con Arduino



Los servomotores tienen tres cables: VCC, GND y Señal. El cable de alimentación VCC es típicamente rojo, y debe ser conectado al pin 5V de la placa Arduino. El cable de tierra, GND, suele ser negro o marrón y debe estar conectado a un conector de tierra de la placa Arduino. El cable de señal es normalmente de color amarillo, naranja o blanco y debe ser conectado a un pin digital en la placa Arduino. Tenga en cuenta que algunos servos pueden tener un consumo de energia considerable, por lo que si tienes que utilizar más de uno o dos, es recomendable una fuente de alimentación independiente, es decir, no conectados al pin + 5V en la placa Arduino. Asegúrese de conectar los GND del Arduino y de la fuente de alimentación externa juntos.

Los servomotores hacen uso de la modulación por ancho de pulsos (PWM) para controlar la dirección o posición de los motores de corriente continua.

ATENCION: Esto no implica que se deba usar estas salidas de Arduino en forma exclusiva.

La mayoría trabaja en la frecuencia de los cincuenta hertz, así las señales PWM tendrán un periodo de veinte milisegundos. La electrónica dentro del servomotor responderá al ancho de la señal modulada. Si los circuitos dentro del servomotor reciben una señal de entre 0,5 a 1,4 milisegundos, éste se moverá en sentido horario; entre 1,6 a 2 milisegundos moverá el servomotor en sentido antihorario; 1,5 milisegundos representa un estado neutro para los servomotores estándares.

Para controlar los servomotores con Arduino disponemos de la Biblioteca «Servo», que deberemos incluir en nuestros códigos. Esta biblioteca permite a una placa Arduino controlar servomotores. Los servos están formados por engranajes y un por eje que puede ser controlado con precisión. Los servos estándar permiten que el eje pueda ser posicionado en varios ángulos, por lo general entre 0 y 180 grados. Los servos de rotación continuas permiten la rotación del eje para ajustarse a diversas velocidades.

Biblioteca servo

Esta biblioteca permite que una placa Arduino controle los servomotores RC (hobby). Los servos tienen engranajes integrados y un eje que se puede controlar con precisión. Los servos estándar permiten que el eje se coloque en varios ángulos, generalmente entre 0 y 180 grados. Los servos de rotación continua permiten que la rotación del eje se ajuste a varias velocidades.
La biblioteca Servo admite hasta 12 motores en la mayoría de las placas Arduino y 48 en el Arduino Mega. En placas que no sean Mega, el uso de la biblioteca deshabilita la funcionalidad analogWrite () (PWM) en los pines 9 y 10, ya sea que haya o no un Servo en esos pines. En el Mega, se pueden usar hasta 12 servos sin interferir con la funcionalidad PWM; El uso de 12 a 23 motores deshabilitará PWM en los pines 11 y 12.

Circuito

Los servomotores tienen tres cables: alimentación, tierra y señal. El cable de alimentación suele ser rojo y debe conectarse al pin de 5 V de la placa Arduino. El cable de tierra suele ser negro o marrón y debe conectarse a un pin de tierra en la placa Arduino. El pin de señal es típicamente amarillo, naranja o blanco y debe conectarse a un pin digital en la placa Arduino. Tenga en cuenta que los servos consumen una potencia considerable, por lo que si necesita conducir más de uno o dos, probablemente necesite alimentarlos desde un suministro separado (es decir, no el pin de + 5V en su Arduino). Asegúrese de conectar los terrenos del Arduino y la fuente de alimentación externa.

Esta biblioteca se encarga de sintetizar los pulsos que espera el servo.

Los pines para controlar los motores puede ser cualquiera de las salidas digitales, la biblioteca servo se encarga de sintetizar los pulsos que necesita el motor

En nuestros prototipos incluimos la Biblioteca Servo a.

#include <Servo.h>

También tenemos que definir los servos que vamos a utilizar

```
Servo servo_1;  // Definimos los servos que vamos a utilizar
Servo servo_2;
Servo servo_n;
```

Despues de void setup() definimos las salidas de cada servo.

```
void setup()  // Comenzamos la configuración de los pines
{
  servo_1.attach(9) ; // Difinimos los pines de salida para cada servo
  servo_2.attach(10);
  servo_n.attach(11);
```

Para definir la posición del servomotor utilizaremos la siguiente expresión, donde el valor entre paréntesis será un valor entre 0 y 180. Este valor se puede ajustar por medio de alguna función

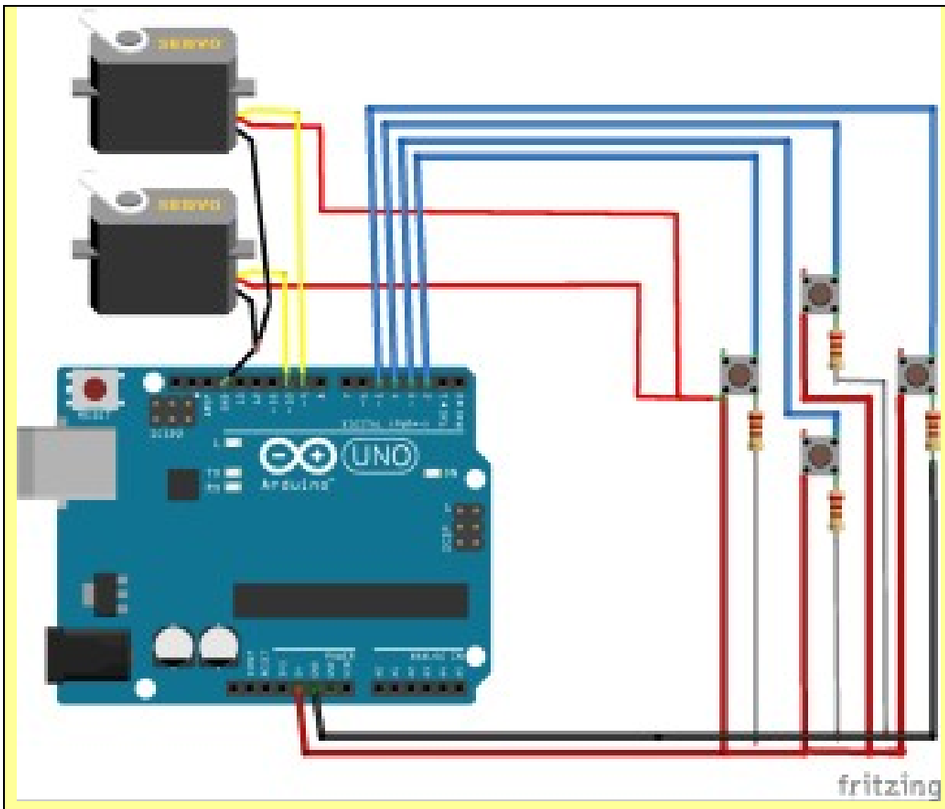
```
servo_1.write(posicion_servo_1);

servo_1.write(valor entre 0 y 180);
```

Nota Prof Bolaños. Este proyecto siguiente no lo he ensayado al día de hoy pero parece consistente.

Tutorial paso a paso para construir un sencillo «Brazo Robótico» con servos y controlado con Arduino.

Componentes y materiales

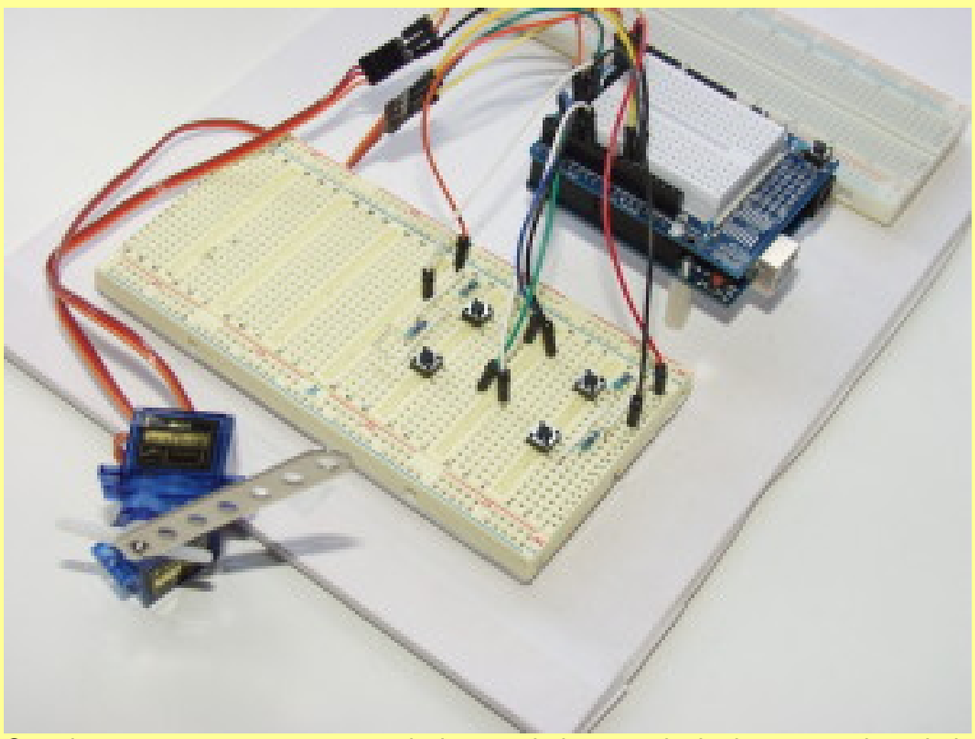
- Placa Arduino UNO
- 2 Micro-servos
- 4 Micro-pulsadores
- 4 Resistencias 220Ohms
- Protoboard
- Cables



Esquema y montaje

Como podemos ver en el esquema, conectamos los servos a +5V, a GND, y a los pines 9 y 10 respectivamente. Estas salidas están controladas por la señal recibida en los pines de entrada 2, 3, 4 y 5, por medio de los cuatro pulsadores.

Los cuatro pulsadores se conectan a las entradas de la placa Arduino, a VCC, y a GND se conectan por medio de una resistencia de 220 Ohms cada uno. Un pin del cada pulsador queda libre.
Cada pulsador permanece conectado a GND en estado normal, y al pulsarlo permite el paso de +5V.



Con dos servos vamos a conseguir dos movimientos principales, uno sobre el plano horizontal y otro en un plano vertical. Podremos mover el «Brazo robótico» hacia izquierda y derecha, y arriba y abajo.
Código Arduino para controlar servos con pulsadores

En la primera parte del código configuramos las salidas que controlarán los servos, las entradas que reciben la señal de los pulsadores y su estado lógico.

#include <Servo.h> // Incluimos la biblioteca Servo

```
  Servo servo_inferior;  // Definimos los servos que vamos a utilizar
  Servo servo_superior;

  int posicion_servo_inferior; // Estas variables definirán
  int posicion_servo_superior; // las posiciónes de los servos en cada momento

  const int pulsador_izquierda = 2; // Definimos el número de pin
  const int pulsador_derecha   = 3; // para cada uno de los pulsadores
  const int pulsador_arriba    = 4;
  const int pulsador_abajo     = 5;

  int estado_pulsador_izquierda = 0; // Variables para leer el estado del pulsador
  int estado_pulsador_derecha   = 0;
  int estado_pulsador_arriba    = 0;
  int estado_pulsador_abajo     = 0;
  int incremento_de_angulo      = 2; // Incremento que queremos cada vez que pulsamos

void setup()  // Comenzamos la configuración de los pines
{
  servo_inferior.attach(9) ; // Difinimos los pines de salida para cada servo
  servo_superior.attach(10);

  pinMode(pulsador_izquierda, INPUT); // Difinimos los pines para la señal de
  pinMode(pulsador_derecha,   INPUT); // los pulsadores como entadas
  pinMode(pulsador_arriba,    INPUT);
  pinMode(pulsador_abajo,     INPUT);
}....
```

Después configuramos las condiciones de funcionamiento, si se pulsa activa un pulsador reacciona de una manera, al pulsar otro, realiza otro movimiento, etc.

```
.....
void loop() {

  estado_pulsador_izquierda = digitalRead(pulsador_izquierda); // Leemos el estado
  estado_pulsador_derecha   = digitalRead(pulsador_derecha)  ; // de cada uno de los pulsadores
  estado_pulsador_arriba    = digitalRead(pulsador_arriba)   ;
  estado_pulsador_abajo     = digitalRead(pulsador_abajo)    ;

  if (estado_pulsador_izquierda == HIGH)

  // Si tenemos pulasdo el "Pulasdor izquierda"
  // El servo inferior se mueve a la nueva posición
  // La nueva posición será la posicion actual + el incremento angular
  // Esperamos 100ms hasta una nueva señal
{.....
```

Código completo

Este es codigo completo para nuestro «Brazo robótico«, copia, pega y modifica el código según tus necesidades.

```
/*  Brazo robótico utilizando dos servos. Como controlar servomotores Creado por www.web-robotica.com */
#include <Servo.h> // Incluimos la biblioteca Servo

  Servo servo_inferior;  // Definimos los servos que vamos a utilizar
  Servo servo_superior;

  int posicion_servo_inferior; // Estas variables definirán
  int posicion_servo_superior; // las posiciónes de los servos en cada momento

  const int pulsador_izquierda = 2; // Definimos el número de pin
  const int pulsador_derecha   = 3; // para cada uno de los pulsadores
  const int pulsador_arriba    = 4;
  const int pulsador_abajo     = 5;

  int estado_pulsador_izquierda = 0; // Variables para leer el estado del pulsador
  int estado_pulsador_derecha   = 0;
  int estado_pulsador_arriba    = 0;
  int estado_pulsador_abajo     = 0;
```

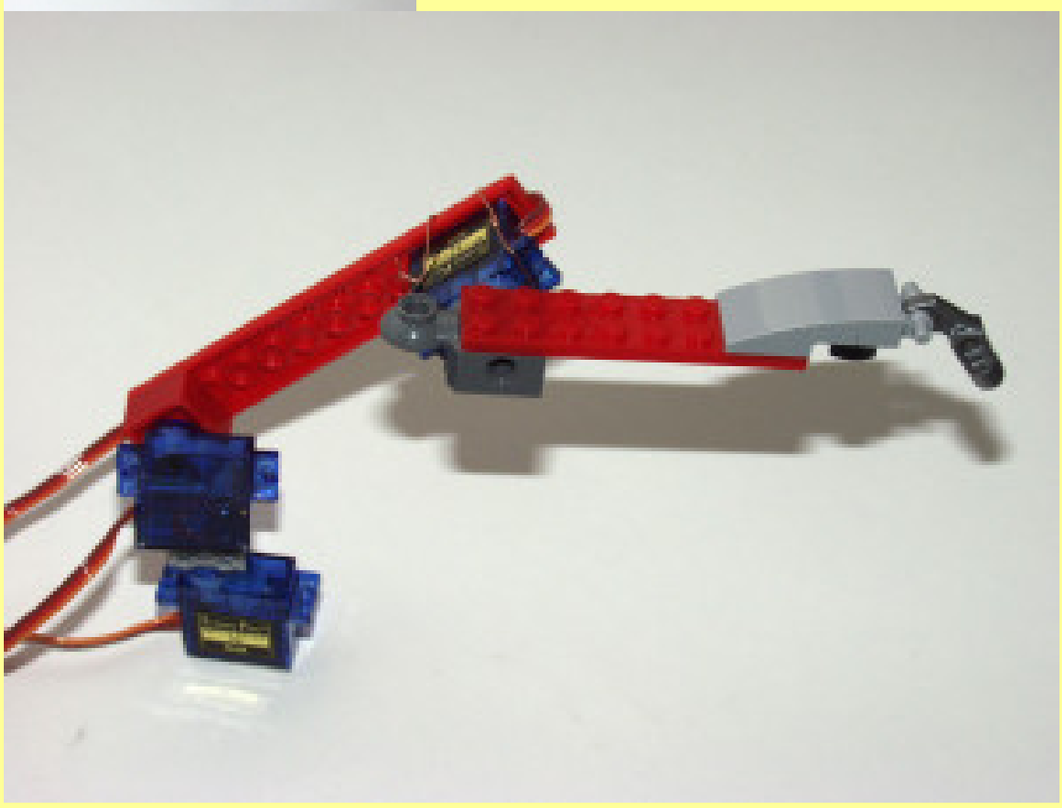
```
  int incremento_de_angulo      = 2; // Incremento que queremos cada vez que pulsamos

void setup()  // Comenzamos la configuración de los pines
{
  servo_inferior.attach(9) ; // Difinimos los pines de salida para cada servo
  servo_superior.attach(10);

  pinMode(pulsador_izquierda, INPUT); // Difinimos los pines para la señal de
  pinMode(pulsador_derecha,   INPUT); // los pulsadores como entadas
  pinMode(pulsador_arriba,    INPUT);
  pinMode(pulsador_abajo,     INPUT);
}

void loop() {

  estado_pulsador_izquierda = digitalRead(pulsador_izquierda); // Leemos el estado
  estado_pulsador_derecha   = digitalRead(pulsador_derecha)  ; // de cada uno de los pulsadores
  estado_pulsador_arriba    = digitalRead(pulsador_arriba)   ;
  estado_pulsador_abajo     = digitalRead(pulsador_abajo)    ;

  if (estado_pulsador_izquierda == HIGH)

  // Si tenemos pulasdo el "Pulasdor izquierda"
  // El servo inferior se mueve a la nueva posición
  // La nueva posición será la posicion actual + el incremento angular
  // Esperamos 100ms hasta una nueva señal
{
  posicion_servo_inferior = posicion_servo_inferior + incremento_de_angulo;
  servo_inferior.write(posicion_servo_inferior);
  delay(100);

  }
  else if (estado_pulsador_derecha == HIGH)

  // Si tenemos pulasdo el "Pulasdor derecha"
  // El servo inferior se mueve a la nueva posición
  // La nueva posición será la posicion actual - el incremento angular
  // Esperamos 100ms hasta una nueva señal
{
  posicion_servo_inferior = posicion_servo_inferior - incremento_de_angulo;
  servo_inferior.write(posicion_servo_inferior);
  delay(100);

  }
   else if (estado_pulsador_arriba == HIGH)

  // Si tenemos pulasdo el "Pulasdor arriba"
  // El servo superior se mueve a la nueva posición
  // La nueva posición será la posicion actual + el incremento angular
  // Esperamos 100ms hasta una nueva señal
{

  posicion_servo_superior = posicion_servo_superior + incremento_de_angulo;
  servo_superior.write(posicion_servo_superior);
  delay(100);

  }
  else if (estado_pulsador_abajo == HIGH)

  // Si tenemos pulasdo el "Pulasdor abajo"
  // El servo inferior se mueve a la nueva posición
  // La nueva posición será la posicion actual - el incremento angular
  // Esperamos 100ms hasta una nueva señal
{

  posicion_servo_superior = posicion_servo_superior - incremento_de_angulo;
  servo_superior.write(posicion_servo_superior);
  delay(100);
```

```
  }
  else {
  // Si no tenemos pulasdo ningún pulasdor
  // esperamos 100ms hasta una nueva señal
    delay(100);
  }
}
```

Esperamos que os guste y que os ayude en vuestros proyectos.

Prototipo para brazo con 3 servos

Este diseño está construido con 3 servomotores y piezas de Lego



Código Arduino

```
 #include servo

Servo nombre_1;  // Definimos los servos que vamos a utilizar
Servo nombre_2;
Servo nombre_3;

int posicion_nombre_1; // Estas variables definirán
int posicion_nombre_2; // las posiciónes de los servos en cada momento
```

```
int posicion_nombre_3

const int pulsador_1 = 1; // Definimos el número de pin
const int pulsador_2 = 2; // para cada uno de los pulsadores
const int pulsador_3 = 3;
const int pulsador_4 = 4;
const int pulsador_5 = 5;
const int pulsador_6 = 6;

int estado_pulsador_1 = 0; // Variables para leer el estado del pulsador
int estado_pulsador_2 = 0;
int estado_pulsador_3 = 0;
int estado_pulsador_4 = 0;
int estado_pulsador_5 = 0;
int estado_pulsador_6 = 0;

int incremento_de_angulo  = 2;  // Incremento que queremos cada vez que pulsamos
                          // Puede ser 2 u otro valor
```

Esta es la primera parte de código donde definimos los servos y el pin para cada pulsador.
Después introducimos esta parte

```
void setup()  // Comenzamos la configuración de los pines
{
  nombre_1.attach(9) ; // Difinimos los pines de salida para cada servo
  nombre_2.attach(10);
  nombre_3.attach(11);

  pinMode(pulsador_1, INPUT); // Difinimos los pines para la señal de
  pinMode(pulsador_2, INPUT); // los pulsadores como entadas
  pinMode(pulsador_3, INPUT);
  pinMode(pulsador_4, INPUT);
  pinMode(pulsador_5, INPUT);
  pinMode(pulsador_6, INPUT);
}
```