SERVO MOTORES

Un servo es un motor con determinadas características que los diferencian del resto de los motores. Podemos decir que un Servo, (a diferencia de otros tipos de motores en los que controlamos la velocidad de giro), le indicamos directamente el ángulo deseado y el servo se encarga de posicionares en este ángulo.

Servo MG90S – Características

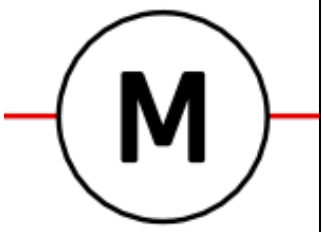
Peso: 13.4g (aprox)
Dimensión: 22.8 x 12.2 x 28.5 mm
Stall Torque: 1.8 kg / cm (4.8V), 2.2kg / cm (6V)
Velocidad de funcionamiento: 0.1seg / 60graods (4.8V), 0.08seg / 60grados (6V)
Voltaje de funcionamiento: 4.8-6.0V
Tipo de motor: Motor sin núcleo
Longitud de cable de conector: 175mm (Aprox.)
Rango de temperatura: 0℃_ 55℃
Pulso del Ciclo 20 ms

https://youtu.be/ZZhuD78BLDk

Típicamente, los servos disponen de un rango de movimiento que oscila entre 0 a 180º. Es decir, no son capaces de dar la vuelta por completo (disponen de topes internos que limitan el rango de movimiento - Aunque recientemente se incorporan al mercado, servos que permiten girar como los motores normales.

Con los servos podemos crear toda clase movimientos de una forma controlada, por ejemplo en robótica, para el control del movimiento del brazo de un robot o en los sistemas de radio control,etc.

La mayoría de los servomotores que se utilizan, son de corriente continua, pero también existen en corriente alterna.



Simbología Usada
(Imagen orientativa)

Internamente un servo, frecuentemente consta de un mecanismo reductor. Por tanto proporcionan fuerza y un alto grado de precisión (incluso décimas de grado). Por otro lado las velocidades de giro son pequeñas frente a los motores de corriente continua. Sin embargo esto puede ser una ventaja al momento de controlar robots o móviles de alta velocidad, como el caso de los aviones radio controlados o brazos robóticos, evitando saltos inesperados.

Los servos soportan una tensión de alimentación entre 4,8V a 7,2V, siendo el valor más adecuado es 6V. Con tensiones inferiores el motor tiene menos fuerza y velocidad. Por otro lado, con tensiones superiores a 6,5V es común que comiencen a oscilar demasiado, lo cual los hace poco estables.

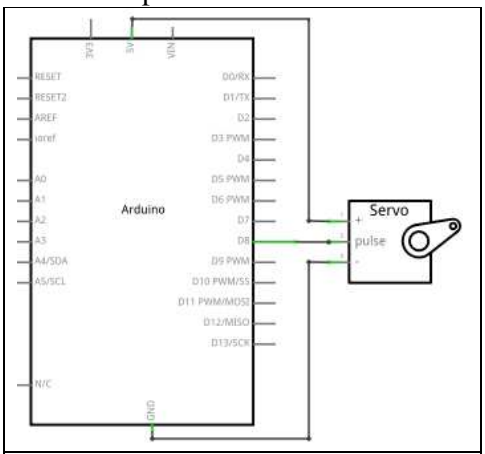
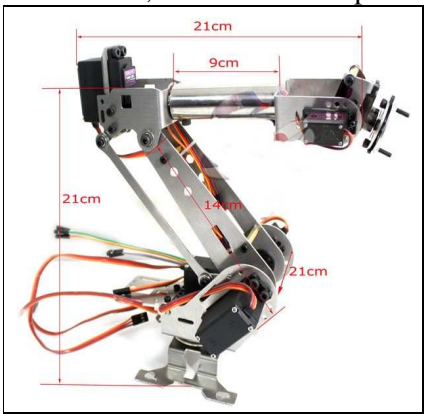


Diagrama de Conexiones



Medidas de un Brazo Robot Educativo

Los servos son cómodos de utilizar, ya que ellos mismos realizan el control de posición, que con otro tipo de motores debe hacerse de forma externa.

Arduino es capas de controlar (mediante librería propia) varios modelos, quizás, los más comunes en el mercado. A continuación una breve descripción de los mismos:

SG90: Es el servo de tamaño "pequeño" estándar dentro de los proyectos de electrónica. Es un servo pequeño, ligero, y barato, que dispone de engranajes de plástico. Muchos dispositivos, como torretas y partes de robots, están pensados para usar estos servos. Tiene un peso de 13.4g (aprox) y sus dimensiones son: 22.8 x 12.2 x 28.5 mm. Llegando a ejercer un Torque (fuerza): 1.8 kg/cm con 4.8V, llegando a 2.2kg/cm cuando se lo alimenta con 6V. Recuerda siempre que hay otros muchos modelos de servos, que poseen mayor fuerza, de los que

a continuación dejo algunos ejemplos. Investiga cual te resultaría más conveniente, según tus necesidades.

MG90S: El MG90S es similar al SG90, pero dispone de engranajes y el acoplamiento metálicos. A cambio pesa un poco más y es un poco más fuerte. Físicamente es compatible con los accesorios del SG90. Usaremos este servo como sustituto del SG90 cuando tengamos una aplicación en la que realmente necesitemos ejercer un poco más de fuerza, y las partes de plástico podrían ceder y/o gastarse (engranajes internos).

M996R: Este es el servo de tamaño "grande" y tiene FUERZA (Torque Aprox: 15kg·cm con 6V). Este tipo de servos es ampliamente utilizado en proyectos de robótica. Igual que con el SG90, muchos dispositivos y kits como brazos robóticos, hexápodos, están diseñados para instalar este tamaño de servo.
Siempre hay que tener en cuenta, que esta es la gama se servos económicos, son aptos para aprender y modestos en sus características. Pero existe una gama de servos más caros, que son más rápidos y precisos. Siempre hay que recordar que hacer robots es sencillo, lo difícil es hacerlos con poca plata.

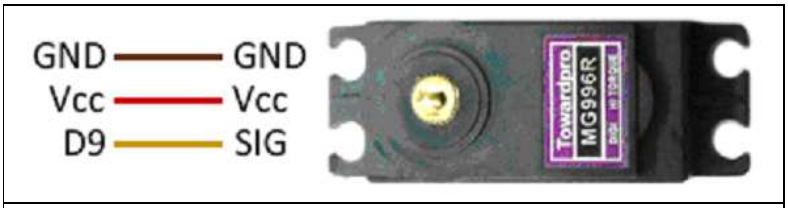
GX32070BLS (Ip68 Metal): Modelo: GX32070BLS **70KG** 5-8.4V 360GR IP68 METAL. Voltaje mínimo - Voltaje máximo: 5V - 7.4V - Largo del servomotor : 4 cm - Ancho del servomotor: 2,1 cm - Altura del servomotor: 4,1 cm - Peso del servomotor 88 g - Largo del cable: 30 cm
Material del engranaje: Metal - Desarrolla una asombrosa fuerza de 70Kg y si esta es tu elección, que te den el que puede rotar hasta 270°.

Esquema de Conexión

La conexión de un servo, es simple. El servo dispone de tres cables, dos de alimentación (GND y Vcc) y uno de señal "**Sig**", cable que conectamos a la placa arduino, desde donde enviaremos las ordenes al servo.
Los servos, sin importar el modelo, tienen tres cables de conexión, y estos cables suelen tener dos combinaciones de colores:

- Marrón (GND), Rojo (Vcc) y Naranja (Sig).
- Negro (GND), Rojo (Vcc) y Blanco (Sig)

Vista del cable (Servo SG90)



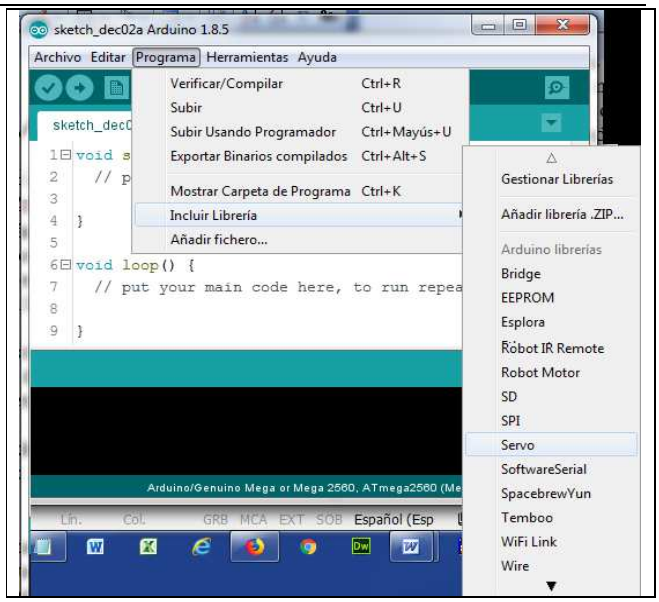
Como conectar el servo – D9 Puede ser cualquier PIN Digital de Arduino

Por un lado, alimentamos el servo mediante el Terminal GND (Marrón o Negro) y Vcc (Rojo). Finalmente, el Naranja o Blanco al Pin de la Placa Arduino que usaremos para controlar el servo.

IMPORTANTE: la alimentación a los servos se debería realizar desde una fuente externa como una batería o fuente de alimentación a una tensión de 5V-6.5V, siendo 6V la tensión idónea. La Alimentación externa lo veremos un poco más adelante.

Inclusión de librería con ayuda del IDE Arduino

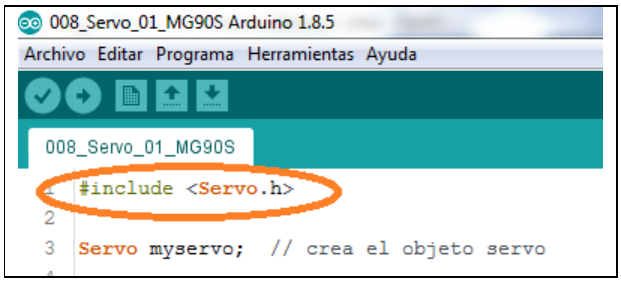
Por otro lado, Arduino puede llegar a proporcionar corriente suficiente para encender y manejar un (UNO) servo pequeño, como el SG90 o SG90S, lo que nos permitirá hacer unos cuantos proyectos de prueba.
Es importante recordar, que no hay que conectar más de uno servo pequeño. Arduino No dispone de corriente suficiente para alimentar un servo grande como el MG996R.



Incluso, no podría alimentar eficientemente varios servos pequeños, y hacer fuerza con ellos, puede exceder la capacidad de corriente de Arduino, provocando un sobrecalentamiento, reinicio, o que deje de funcionar definitivamente.

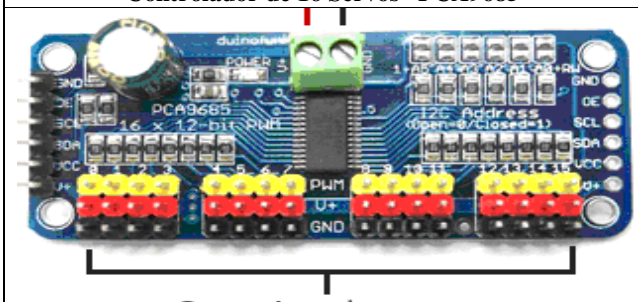
CONTROLANDO UN SERVO MOTOR

Ahora vamos lo interesante, la programación. Para controlar un Servo, tenemos que hacer uso de una librería, de las que Arduino nos provee desde el momento que instalamos el IDE. Esta librería se llama "**Servo.h**". Y deberemos recordarlo cada vez que hagamos un programa que controle un servo.
Ver "**Apéndice A - Librería "Servo.h" Clases y Métodos disponibles**".



Para incorporar esta librería, tenemos dos opciones, incluirla con ayuda del IDE o hacerlo manualmente. El resultado será el mismo.

Controlador de 16 Servos "PCA9685"

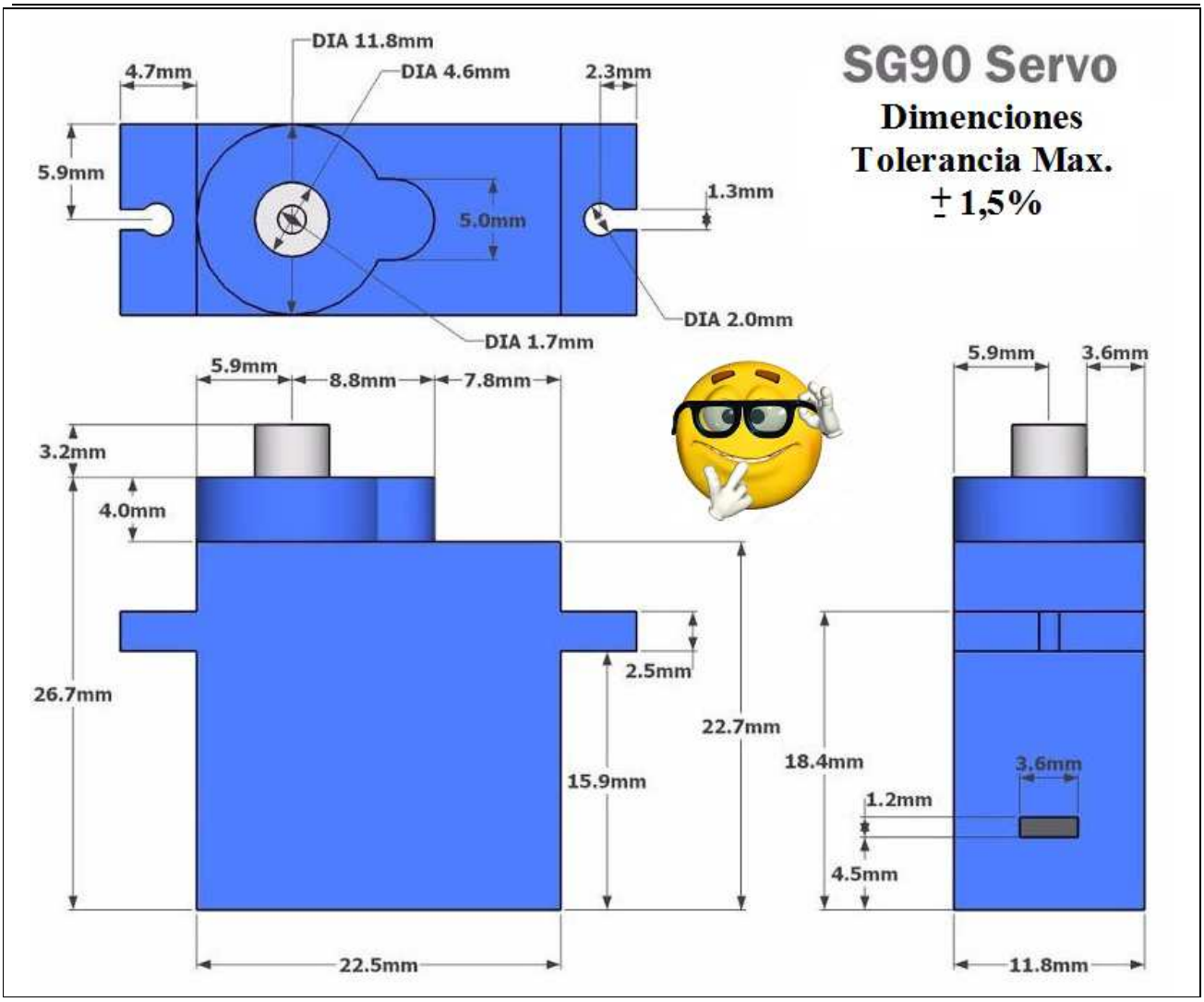


En Nuestros primeros ejemplos, usaremos el Servo Motor "MG90S" debido a que es de los más pequeños con menor costo, menos consumo energético (podremos manejarlo directamente desde Arduino), y a la vez tiene fuerza para completar los ejercicios programados.

Luego, podremos comenzar a utilizar el Controlador de Servos "PCA9685". Que nos permitirá manejar hasta 16 Servos Motores simultáneamente con alimentación externa, de una forma muy simple y rápida. Pero antes hay que aprender otras cosas. Así que esperemos un poquito más.

Medidas del Servo Motor SG90 y su compañero SG90S

Recuerde que las medidas están acá, por que las necesitará cuando deba usar este servo.



1- Reconocimiento de un Servo Motor. Proyecto A - Usando For.

Hacemos girar en toda su amplitud a un servo motor (180). Para este ejemplo usamos el Servo Motor **MG90S**. Debido a su bajo consumo y gran fuerza.

(Programa "008_Servo_01_MG90S_A")

```
1   #include <Servo.h>
-
2   Servo MiServo;   // Crea Objeto que Manejara Servo
3   int PinServo = 9,
4       Angulo;      // Angulo al que rotará el servo
-
5   void setup( ) {
6     MiServo.attach(PinServo);  // Vincula el Servo al  Pin
7   }
8   void loop( ) {
9-    for (Angulo = 0; Angulo <= 180; Angulo += 1){
10    //varía el Angulo  de 0 a 180, con esperas de 10ms
11       MiServo.write(Angulo);
12       delay(10);
13    }
14-   for (Angulo = 180; Angulo >= 0; Angulo -= 1){
15    //varía el Angulo de 180 a 0 , con esperas de 10ms
16       MiServo.write(Angulo);
17       delay(10);
18    }
19  }
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Servo Motor **MG90 o un MG90S** – 3 Cables Macho/Macho –1 Placa Protoboard - Placa Arduino y 1 Cable USB.

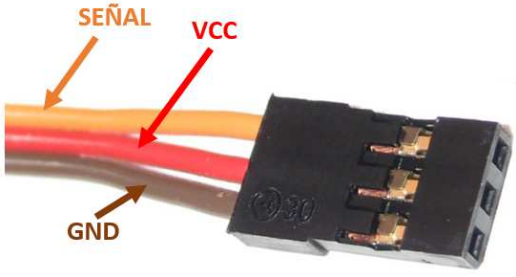
Los cables deberán ser conectados según modelo:

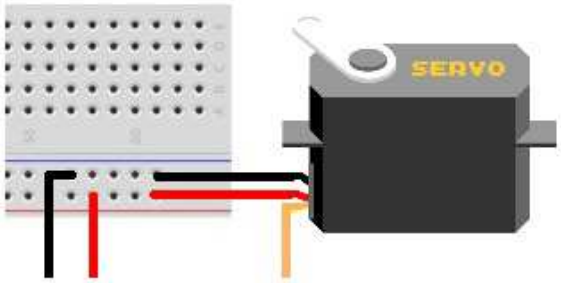
- Marrón (GND), Rojo (5V) y Naranja – (SIG o Señal)
- Negro (GND), Rojo (5V) y Blanco – (SIG o Señal).

Aclaración: Naranja o Blanco al Pin de la Placa Arduino que usaremos para controlar el servo.

Conexión Servo Motor **MG90S**

Conexión a placa Arduino y/o Protoboard.





Recordar que Arduino puede proporcionar alimentación suficiente para encender y manejar un (UNO) servo pequeño, como el SG90 o SG90S. Próximamente usaremos el Controlador de Servos "PCA9685" que nos permitirá manejar hasta 16 Servos Motores simultáneamente (Por cada placa controladora que usemos).

2- Reconocimiento de un Servo Motor. Proyecto B – Programación Multitarea (Esta es la forma de programar Recomendada).

Hacemos girar en toda su amplitud a un servo motor (180). Para este ejemplo usamos el Servo Motor **MG90S**. Debido a su bajo consumo y gran fuerza.

(Programa "008_Servo_01_MG90S_B")

```
1   #include <Servo.h>
-
2   const int PinServo = 9;
3   int       Angulo,  // Angulo al que rotará el servo
4             ValorGiro;
5   Servo MiServo;   // Crea Objeto que Manejará Servo
-
6   void setup( ){
7    MiServo.attach(PinServo);  // Vincula el Servo al  Pin
8    Angulo= 90;
9-   ValorGiro = 1; // Comienza Sumando (Agranda ángulo)
10  }
11  void loop( ) {
12   Angulo = Angulo + ValorGiro;
13   if( Angulo < 1 || Angulo > 179 ){
14     ValorGiro *= -1;
15   }
16   MiServo.write(Angulo);
17   delay(10); // Solo para hacer que gire más lento
18  }
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Servo Motor **MG90 o un MG90S** – 3 Cables Macho/Macho –1 Placa Protoboard - Placa Arduino y 1 Cable USB.

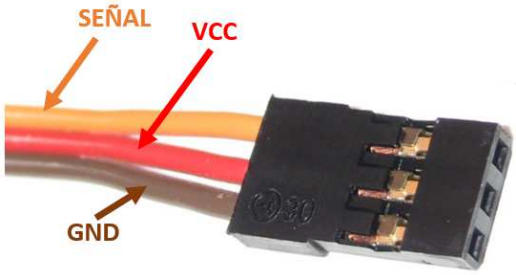
Los cables deberán ser conectados según modelo:

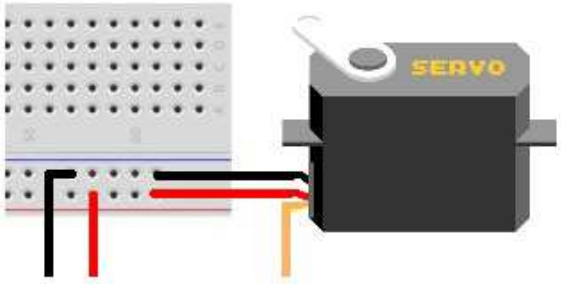
- Marrón (GND), Rojo (5V) y Naranja – (SIG o Señal)
- Negro (GND), Rojo (5V) y Blanco – (SIG o Señal).

Aclaración: Naranja o Blanco al Pin de la Placa Arduino que usaremos para controlar el servo.

Conexión Servo Motor **MG90S**

Conexión a placa Arduino y/o Protoboard.





Recordar que Arduino puede proporcionar alimentación suficiente para encender y manejar un (UNO) servo pequeño, como el SG90 o SG90S. Próximamente usaremos el Controlador de Servos "PCA9685" que nos permitirá manejar hasta 12 Servos Motores simultáneamente (Por cada placa controladora que usemos).

3- Servo Motor controlado por un Potenciómetro.

Con un potenciómetro, controlar el ángulo de giro de un Servo Motor. Al Angulo de giro debo oscilar entre 0 y 180.

(Programa "008_Servo_02_MG90S_con_Potenciometro_01")

```
1   #include <Servo.h>
```

```
-
2   Servo MiServo;  // crea el objeto servo
-
3   int   PinAnalogico = A0;
4   int   PinServo    = 9;
5   double ValorOriginal,
6          Angulo;
-
7   void setup( ) {
8     Serial.begin(9600);
9-    pinMode( PinAnalogico, INPUT_PULLUP );
10    MiServo.attach(PinServo);
11  }
12  void loop( ) {
13    ValorOriginal = analogRead(PinAnalogico); // Lectura analógica del Potenciómetro
14    Angulo = map( ValorOriginal, 0, 1023, 0, 180); // convertir valor Angular del Servo
15    MiServo.write(Angulo);  // Gira Servo
16    VerPuertoSerie( );
17  }
-
18  void VerPuertoSerie(void){
19    Serial.print("V Original: ");
20    Serial.print(ValorOriginal);
21    Serial.print(" - V Convertido: ");
22    Serial.println(Angulo);
23    //delay(250);  // Probar con y sin el "delay( )"
24  }
```

CIRCUITO PARA NUESTRO PROYECTO

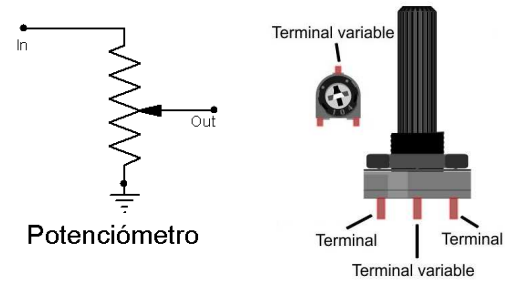
Lista de Materiales: 1 Servo Motor **MG90 o un MG90S** – Potenciómetro **50K Rotativo (Para usar con Protoboard) -** 8 Cables Macho/Macho –1 Placa Protoboard - Placa Arduino y 1 Cable USB.

Conexión Servo Motor **MG90S**

Los cables deberán ser conectados: a la Izquierda de arriba hacia abajo: cable Naranja, al PIN Digital destinado a Manejar el Servomotor. Cable Verde, al Pin Analógico destinado al Potenciómetro. Cable Negro a GND. Cable Marrón a 5V.

Recordar: Los colores del los cables de conexión del Servo Motor Pueden cambiar según modelo. Y estoas son las opciones de conexión:

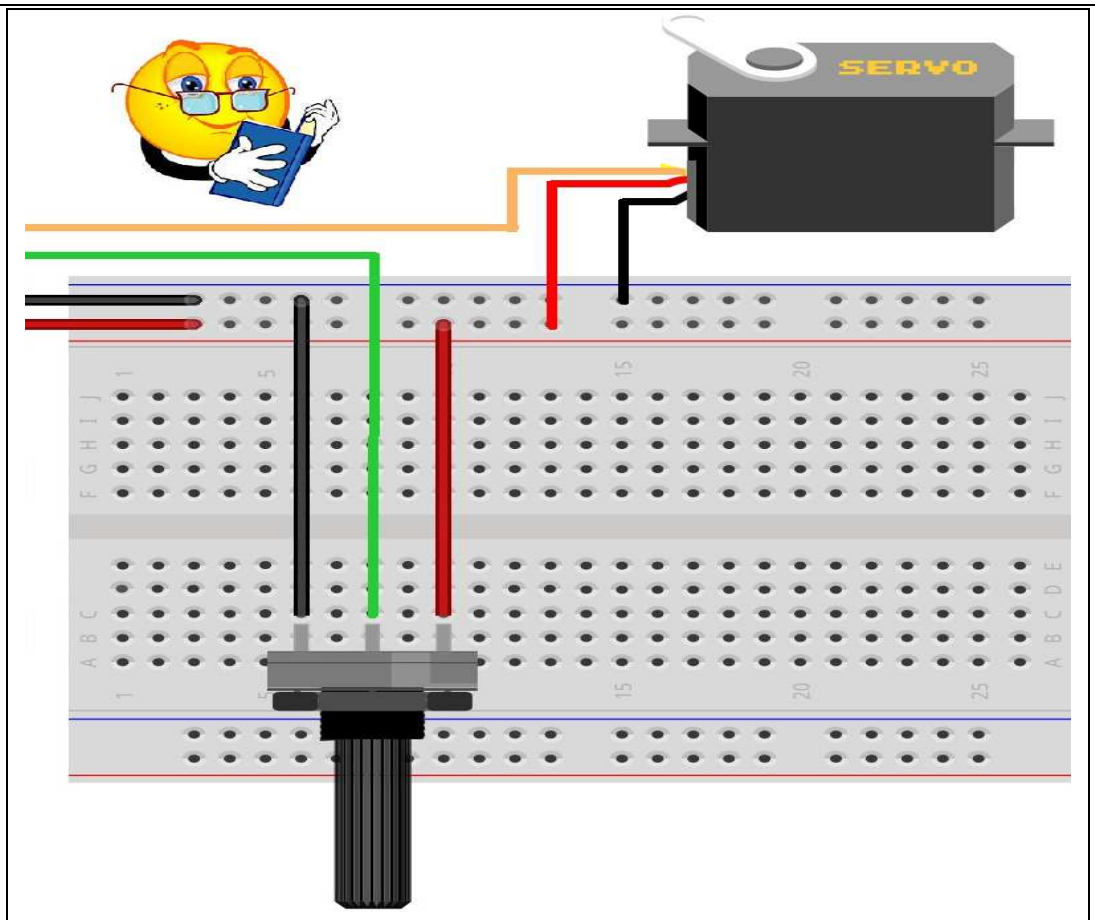
- Marrón (GND), Rojo (5V) y Naranja – (SIG o Señal)
- Negro (GND), Rojo (5V) y Blanco – (SIG o Señal).



Conexión de un Potenciómetro – Terminal Variable A una entrada Analógica – Las Terminales se conectan a GND y 5V (Indistintamente).

Aclaración: Naranja o Blanco al Pin de la Placa Arduino que usaremos para controlar el servo.

Conexión a placa Arduino y/o Protoboard.



Recordar que Arduino puede proporcionar alimentación suficiente para encender y manejar un (UNO) servo pequeño, como el SG90 o SG90S. Próximamente usaremos el Controlador de Servos "PCA9685" que nos permitirá manejar 12 Servos Motores simultáneamente.

4- Hacer Girar Servo Motor controlado por un dos Botones (Izquierdo y derecho). Prever límite izquierdo y derecho del servo y usar una alarma. Proyecto A.

Con dos botones hacer girar un Servo Motor, avisando con Una alarma de luz y sonido cuando se presione ambos botones al mismo tiempo, y alarma Izquierda o derecha cuando se intente hacer girar más allá del imite al servo. Recordar que un Servo Motor tiene un Angulo de giro que oscilar entre 0^{o} y 180^{o}.

Este proyecto lo haremos dos veces, en este, el proyecto "A", mientras mantenemos apretado un botón, el servo gira hasta llegar al tope (este resulta muy practico pero muy inexacto al momento de necesitar un giro preciso). Y el otro Proyecto, el B, en el que cada vez que presionamos un botón realiza un pequeño giro (valor de cada salto es configurable) que resulta muy preciso, a la hora de hacer un pequeño giro.

(Programa "008_Servo_03_MG90S_con_Dos_Botones_B")

PROYECTO A

```
1   #include <Servo.h>
-
2   const int PinParlante = 2,
3             PinServo    = 3;
-
4   const int PinBotonIzquierdo = 7,
5             PinBotonDerecho   = 8;
-
6   const int PinLedErrorIzquierda = 10, //Led Rojo Izquierdo
7             PinLedOk             = 11,       //Led Verde Central
```

```
8           PinLedErrorDerecha   = 12; //Led Rojo Derecha
-
9   Servo MiServo;   // Crea variable que Manejara Servo
-
10  int BotonDerechoValor,  // Variable botón 01
11      BotonIzquierdoValor;   // Variable botón 02
-
12  long TiempoInicial = millis( ), // Led Verde
13       TiempoEspera  = 1000,      // Led Verde
14       TISonidoTotal = millis( ),
15       TIAlarmaDerecha = millis( ),
16       TIAlarmaIzquierda = millis( ),
17       TESonido = 500;
-
18  int  ValorFrecuencia = 1500,
19       Frecuencia = 0;
-
20  int  EstadoAlarmaDerecha = LOW,
21       EstadoAlarmaIzquierda = LOW;
-
22  int AnguloMaximo = 175,
23      AnguloMinimo = 5,
24      Angulo = (AnguloMaximo + AnguloMinimo)/2,
25      AnguloAnterior = Angulo,
26      Salto = 1,
27      ErrorBotones;
-
28  void setup( ){
29    Serial.begin (9600);
30    MiServo.attach(PinServo);  // Vincula el Servo al  Pin
31    pinMode(PinBotonIzquierdo, INPUT_PULLUP);
32    pinMode(PinBotonDerecho, INPUT_PULLUP);
32    pinMode(PinLedErrorIzquierda, OUTPUT);
33    pinMode(PinLedOk, OUTPUT);
34    pinMode(PinLedErrorDerecha, OUTPUT);
-
35    Serial.print("Angulo Inicial: ");
36    Serial.println(Angulo);
37  }
-
38  void loop ( ){
39     BotonDerechoValor   = digitalRead(PinBotonDerecho);   // Leemos PinBoton_01.
40     BotonIzquierdoValor = digitalRead(PinBotonIzquierdo); // Leemos PinBoton_02.
41     ErrorBotones = 0;
42     if( BotonDerechoValor == LOW && BotonIzquierdoValor == LOW){
43        Serial.println("------> ERROR");
44        AlarmaDerecha(HIGH);
45        AlarmaIzquierda(HIGH);
46        AlarmaTotal(HIGH);
47        ErrorBotones = 1;
48     }else if( BotonDerechoValor == LOW){
49        digitalWrite( PinLedOk, HIGH );
50        TiempoInicial = millis( );
-
```

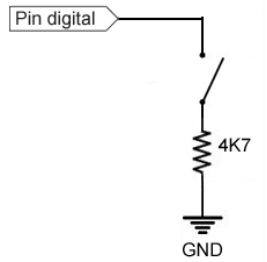
```
51      Angulo += Salto;
52      if( Angulo > AnguloMaximo ){
53        Angulo = AnguloMaximo;
54        AlarmaDerecha(HIGH);
55        AlarmaTotal(HIGH);
56      }
57      Serial.print("------> Preciono Boton Derecho: ");
58      Serial.println(Angulo);
-
59    }else if( BotonIzquierdoValor == LOW ){
60      digitalWrite( PinLedOk, HIGH );
61      TiempoInicial = millis( );
-
62      Angulo -= Salto;
63      if( Angulo < AnguloMinimo ){
64        Angulo = AnguloMinimo;
65        AlarmaIzquierda(HIGH);
66        AlarmaTotal(HIGH);
67      }
68      Serial.print("------> Preciono Boton Iquierdo: ");
69      Serial.println(Angulo);
-
70    }else if( BotonDerechoValor == HIGH && BotonIzquierdoValor == HIGH){
71      //Serial.println("------> Apaga Todo");
72      ApagaTodo( );
73    }
-
74    if(ErrorBotones == 0 && Angulo != AnguloAnterior){
75       MiServo.write(Angulo);
76       AnguloAnterior = Angulo;
77    }
78  }
-
79  void AlarmaDerecha(int Activa){
80   if( millis( ) > TIAlarmaDerecha + TESonido ){
81    if( Activa == HIGH ){
82      EstadoAlarmaDerecha   = HIGH - EstadoAlarmaDerecha;
83      TIAlarmaDerecha = millis( );
84     }else{ // Apagando Alarma - Deja listo para Próxima
85      EstadoAlarmaDerecha   = LOW;
-     }
86   }
87   digitalWrite( PinLedErrorDerecha, EstadoAlarmaDerecha );
88  }
-
89  void AlarmaIzquierda(int Activa){
90   if( millis( ) > TIAlarmaIzquierda + TESonido ){
91    if( Activa == HIGH ){
92      EstadoAlarmaIzquierda   = HIGH - EstadoAlarmaIzquierda;
93      TIAlarmaIzquierda = millis( );
94     }else{ // Apagando Alarma - Deja listo para Próxima
95      EstadoAlarmaIzquierda   = LOW;
96     }
97   }
```

En este Bloque, se controla que no haya erres al presionar Botones y que el ángulo haya cambiado desde la ultima vez que se movió el servo, antes de intentar moverlo nuevamente.
Note que se actualiza el Angulo anterior.

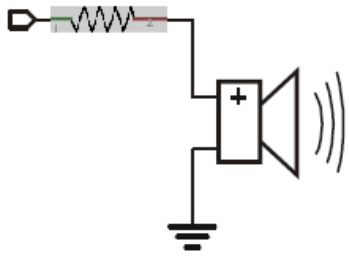
```
98    digitalWrite( PinLedErrorIzquierda, EstadoAlarmaIzquierda );
99   }
 -
100  void AlarmaTotal(int Activa){
101   if( millis( ) > TISonidoTotal + TESonido ){
102    noTone( PinParlante);
103    if( Activa == HIGH ){
104     Frecuencia   = ValorFrecuencia - Frecuencia;
105     TISonidoTotal = millis( );
106    }else{ // Apagando Alarma - Deja listo para Próxima
107     Frecuencia = ValorFrecuencia;
108    }
109   }else{
110    tone( PinParlante, Frecuencia);
111   }
112  }
 -
113  void ApagaTodo(void){
114   if( millis( ) > TiempoInicial + TiempoEspera ){
115    digitalWrite( PinLedOk,LOW);
116   }
117   AlarmaDerecha(LOW);
118   AlarmaIzquierda(LOW);
119   AlarmaTotal(LOW);
120   noTone( PinParlante);
121  }
```

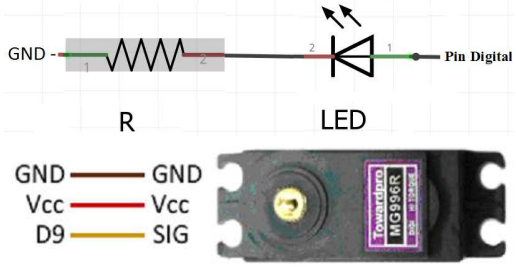
CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Servo Motor **MG90 o MG90S** – 1 Parlante – 1 resistencia de 470Ω (Para el parlante, pero si usa uno reciclado puede omitir la resistencia) - 3 LEDs - 3 resistencias de 470Ω (para los Leds) – 2 Botones Pulsadores – 2 resistencias de 4K7 Ω (para Botones) - 10 Cables Macho/Macho – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.



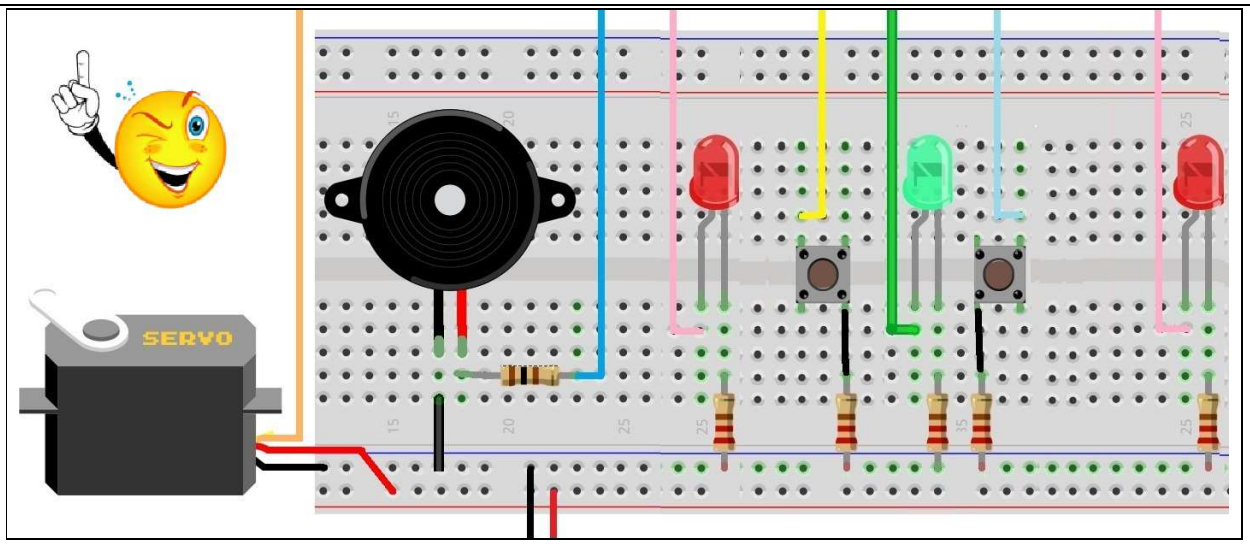
Conexión para cada Botóe


Conexión del Parlante



Conexión para cada uno los 3 LEDs y Servo Motor

Los cables deberán ser conectados: Arriba, de Izquierda a derecha: cable Marrón, al PIN Digital destinado a Manejar el Servo Motor. Cable Azul, destinado al Parlante o Speaker, cable Rosa, alarma, cable Amarillo al Pin destinado al botón de giro a la izquierda, cale verde, Led Verde que indica que el Servomotor esta en movimiento y OK, cable Celeste al Pin destinado al botón de giro a la derecha, ultimo Cable de arriba, color Rosa, alarma a la derecha. Cables de Abajo: el Negro a GND y cable Rojo a 5V.



Este Proyecto, es muy similar al Proyecto B. La programación solo cambia en la Función "**setup( )**", y en el circuito, cambia la conexión de los Botones.

5- Hacer Girar Servo Motor controlado por un dos Botones (Izquierdo y derecho). Prever límite izquierdo y derecho del servo y usar una alarma. Proyecto B.

Con dos botones hacer girar un Servo Motor, avisando con Una alarma de luz y sonido cuando se presione ambos botones al mismo tiempo, y alarma Izquierda o derecha cuando se intente hacer girar más allá del imite al servo. Recordar que un Servo Motor tiene un Angulo de giro que oscilar entre 0^o y 180^o. En este Proyecto, el B, cada vez que presionamos un botón realiza un pequeño giro (valor de cada salto es configurable) que resulta muy preciso, a la hora de hacer un pequeño giro.

(Programa "008_Servo_03_MG90S_con_Dos_Botones_A")

PROYECTO B

```
1   #include <Servo.h>
-
2   const int PinParlante = 2,
3             PinServo    = 3;
-
4   const int PinBotonIzquierdo = 7,
5             PinBotonDerecho   = 8;
-
6   const int PinLedErrorIzquierda = 10, //Led Rojo Izquierdo
7             PinLedOk             = 11, //Led Verde Central
8             PinLedErrorDerecha   = 12; //Led Rojo Derecha
-
9   Servo MiServo;   // Crea variable que Manejara Servo
-
10  int BotonDerechoValor,  // Variable botón 01
11      BotonIzquierdoValor,   // Variable botón 02
12      PresionadoDerecho,     // Para saber si Este Botón Fue Presionado
13      PresionadoIzquierdo;     // Para saber si Este Botón Fue Presionado
-
14  long TiempoInicial = millis( ), // Led Verd
15       TiempoEspera  = 1000,     // Led Verde
16       TISonidoTotal = millis( ),
17       TIAlarmaDerecha = millis( ),
```

```
18      TIAlarmaIzquierda = millis( ),
19      TESonido = 500;
-
20  int  ValorFrecuencia = 1500,
21      Frecuencia = 0;
-
22  int  EstadoAlarmaDerecha = LOW,
23      EstadoAlarmaIzquierda = LOW;
-
24  int AnguloMaximo = 175,
25     AnguloMinimo = 5,
26     Angulo = (AnguloMaximo + AnguloMinimo)/2,
27     AnguloAnterior = Angulo,
28     Salto = 5,
29     ErrorBotones;
-
30  void setup( ){
31   Serial.begin (9600);
32   MiServo.attach(PinServo);  // Vincula el Servo al  Pin
32   pinMode(PinBotonIzquierdo, INPUT_PULLUP);
33   pinMode(PinBotonDerecho, INPUT_PULLUP);
34   pinMode(PinLedErrorIzquierda, OUTPUT);
35   pinMode(PinLedOk, OUTPUT);
36   pinMode(PinLedErrorDerecha, OUTPUT);
37   PresionadoDerecho   = 0;
38   PresionadoIzquierdo = 0;
39   Serial.print("Angulo Inicial: ");
40   Serial.println(Angulo);
41  }
42  void loop ( ){
43    BotonDerechoValor   = digitalRead(PinBotonDerecho);   // Leemos PinBoton_01.
44    BotonIzquierdoValor = digitalRead(PinBotonIzquierdo); // Leemos PinBoton_02.
45    if(BotonDerechoValor == LOW){
46      PresionadoDerecho = 1;     //Activo Botón
47    }
48    if(BotonIzquierdoValor == LOW){
49      PresionadoIzquierdo = 1;     //Activo Botón
50    }
-
51    BotonDerechoValor   = digitalRead(PinBotonDerecho);   // Leemos PinBoton_01.
52    BotonIzquierdoValor = digitalRead(PinBotonIzquierdo); // Leemos PinBoton_02.
53    ErrorBotones = 0;
54    if( BotonDerechoValor == LOW && BotonIzquierdoValor == LOW){
55      Serial.println("------> ERROR");
56      PresionadoDerecho   = 0;
57      PresionadoIzquierdo = 0;
58      AlarmaDerecha(HIGH);
59      AlarmaIzquierda(HIGH);
60      AlarmaTotal(HIGH);
61      ErrorBotones = 1;
62    }else if( BotonDerechoValor == HIGH && PresionadoDerecho == 1){
63      PresionadoDerecho   = 0;   //La variable vuelve a su valor original
64      digitalWrite( PinLedOk, HIGH );
65      TiempoInicial = millis( );
```

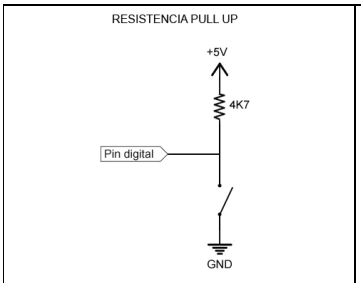
```
-
66        Angulo += Salto;
67        if( Angulo > AnguloMaximo ){
68          Angulo = AnguloMaximo;
69          AlarmaDerecha(HIGH);
70          AlarmaTotal(HIGH);
71        }
72        Serial.print("------> Preciono Boton Derecho: ");
73        Serial.println(Angulo);
-
74      }else if( BotonIzquierdoValor == HIGH && PresionadoIzquierdo == 1){
75        PresionadoIzquierdo   = 0;    //La variable vuelve a su valor original
76        digitalWrite( PinLedOk, HIGH );
77        TiempoInicial = millis( );
-
78        Angulo -= Salto;
79        if( Angulo < AnguloMinimo ){
80          Angulo = AnguloMinimo;
81          AlarmaIzquierda(HIGH);
82          AlarmaTotal(HIGH);
83        }
84        Serial.print("------> Preciono Boton Iquierdo: ");
85        Serial.println(Angulo);
-
86      }else if( BotonDerechoValor == HIGH && BotonIzquierdoValor == HIGH){
87        //Serial.println("------> Apaga Todo");
88        ApagaTodo( );
89      }
-
90      if(ErrorBotones == 0 && Angulo != AnguloAnterior){
91          MiServo.write(Angulo);
92          AnguloAnterior = Angulo;
93      }
94    }
-
95    void AlarmaDerecha(int Activa){
96     if( millis( ) > TIAlarmaDerecha + TESonido ){
97       if( Activa == HIGH ){
98         EstadoAlarmaDerecha   = HIGH - EstadoAlarmaDerecha;
99         TIAlarmaDerecha = millis( );
100      }else{ // Apagando Alarma - Deja listo para Próxima
101        EstadoAlarmaDerecha   = LOW;
102      }
103    }
104    digitalWrite( PinLedErrorDerecha, EstadoAlarmaDerecha );
105   }
-
106   void AlarmaIzquierda(int Activa){
107    if( millis( ) > TIAlarmaIzquierda + TESonido ){
108      if( Activa == HIGH ){
109        EstadoAlarmaIzquierda   = HIGH - EstadoAlarmaIzquierda;
110        TIAlarmaIzquierda = millis( );
111      }else{ // Apagando Alarma - Deja listo para Próxima
112        EstadoAlarmaIzquierda   = LOW;
```

En este Bloque, se controla que no haya erres al presionar Botones y que el ángulo haya cambiado desde la ultima vez que se movido el servo, antes de mover el servo. Note que es acá donde se actualiza el Angulo anterior.

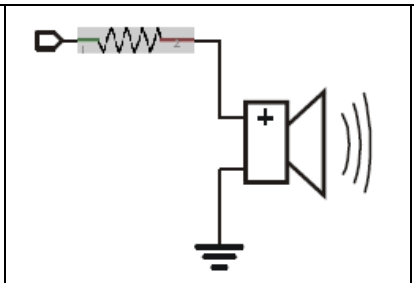
```
113     }
114   }
115   digitalWrite( PinLedErrorIzquierda, EstadoAlarmaIzquierda );
116 }
 -
117 void AlarmaTotal(int Activa){
118   if( millis( ) > TISonidoTotal + TESonido ){
119     noTone( PinParlante);
120     if( Activa == HIGH ){
121       Frecuencia   = ValorFrecuencia - Frecuencia;
122       TISonidoTotal = millis( );
123     }else{ // Apagando Alarma - Deja listo para Próxima
124       Frecuencia = ValorFrecuencia;
125     }
126   }else{
127     tone( PinParlante, Frecuencia);
128   }
129 }
 -
130 void ApagaTodo(void){
131   if( millis( ) > TiempoInicial + TiempoEspera ){
132     digitalWrite( PinLedOk,LOW);
133   }
134   AlarmaDerecha(LOW);
135   AlarmaIzquierda(LOW);
136   AlarmaTotal(LOW);
137   noTone( PinParlante);
138 }
```

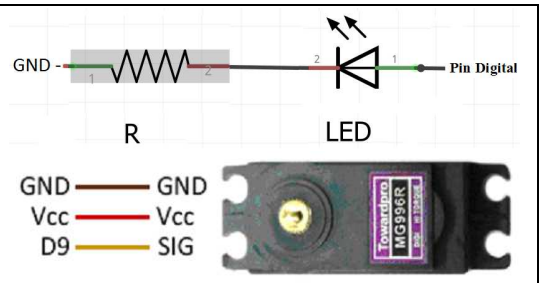
CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Servo Motor **MG90 o MG90S** – 1 Parlante – 1 resistencia de 470Ω (Para el parlante, pero si usa uno reciclado puede omitir la resistencia) - 3 LEDs - 3 resistencias de 470Ω (para los Leds) – 2 Botones Pulsadores – 2 resistencias de 4K7 Ω (para Botones) - 20 Cables Macho/Macho – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.



Conexión para cada Botón


Conexión del Parlante

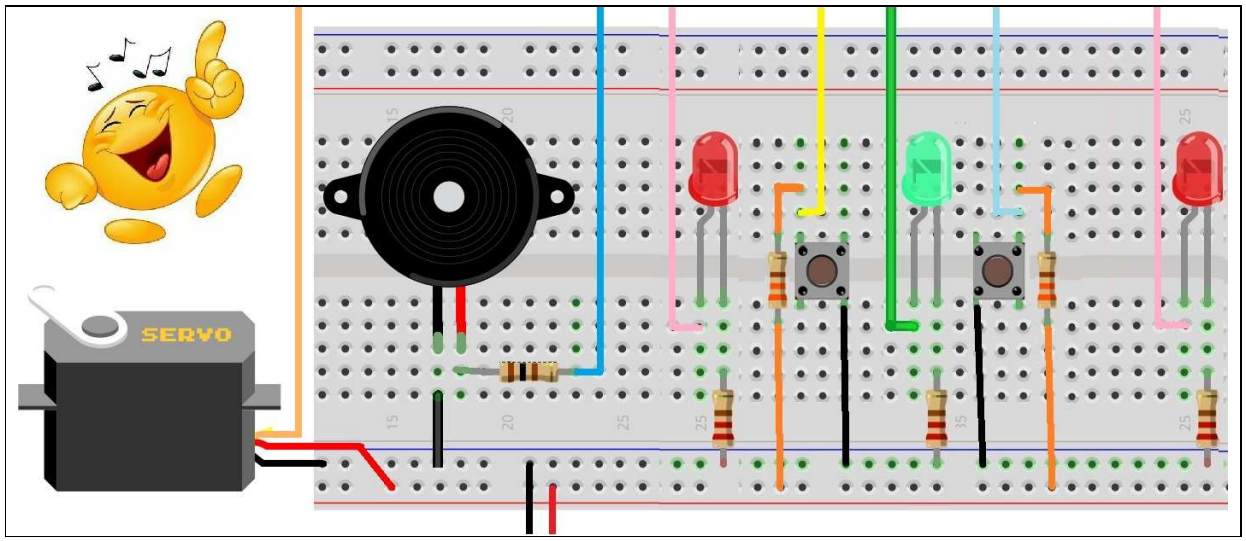


Conexión para cada LEDs y Servo Motor

Este Proyecto, es muy similar al A. La programación solo cambia en la Función "**setup( )**", y en el circuito, cambia la conexión de los Botones.

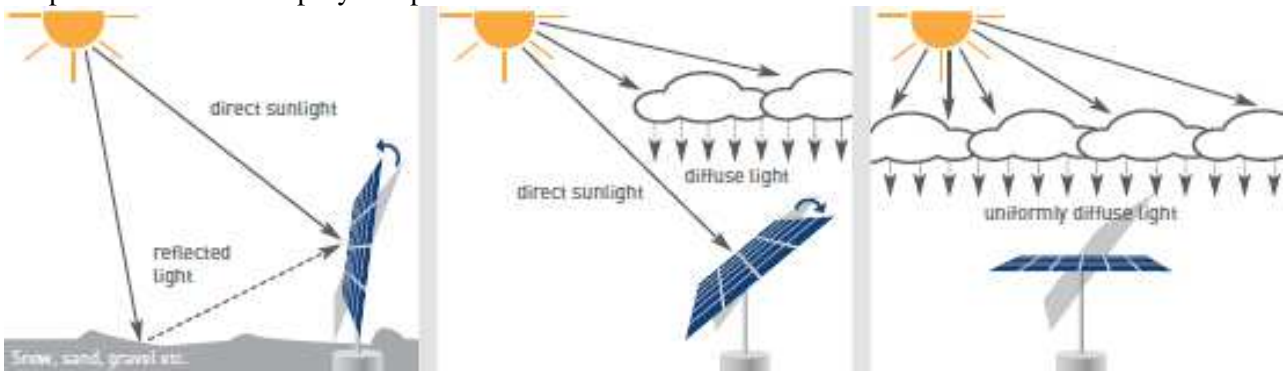
Los cables deberán ser conectados: Arriba, de Izquierda a derecha: cable Naranja, al PIN Digital destinado a Manejar el Servomotor. Cable Azul, destinado al Parlante o Speaker, cable Rosa, alarma exceso a la izquierda, cable Amarillo al Pin destinado al botón de giro a la izquierda, cale ver, luz que indica que el Servomotor esta en movimiento y OK, cable Celeste al Pin destinado al botón de giro a la derecha, ultimo Cable de arriba, color Rosa, alarma exceso a la derecha. Cables de Abajo: el Negro a

GND y cable Rojo a 5V. **IMPORTANTE:** Notar que ambos botones, tienen una conexión del tipo "**Pull_Up**" (Secuencia: 5V-Resistencia-PindeControl-Boton-GND).



6- Seguidor de Luz (Girasol) simple - Busca a la derecha o a la izquierda – Usa un único servo. Proyecto A.

Este Dispositivo tiene la capacidad de buscar el ángulo, en el que recibe la mayor cantidad de luz, sin importar si proviene de un reflejo, o directamente del sol. Tiene una gran desventaja, y es que no puede girar en todas las direcciones - Problema que solucionaremos usando dos servos en la parte B de este Proyecto, que desarrollaremos próximamente. Este proyecto puede concretarse usando dos Paneles solares reales.

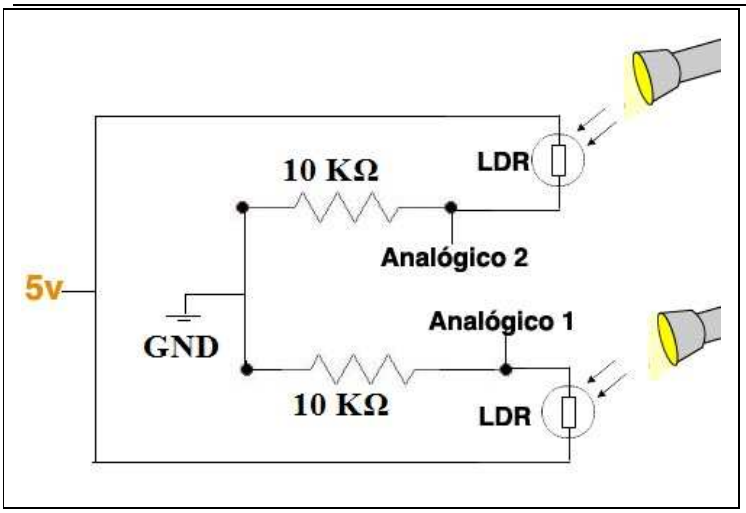


(Programa "008_Servo_04_MG90S_Girasol_A")

```
1   #include<Servo.h>
-
2   #define PinDerecho   A1
3   #define PinIzquierdo A2
4   #define PinServo 3
5   #define ZonaTolerancia 20
-
6   int PosicionAngular = 90,  // Posición Inicial
7       LuzDerecha,
8       LuzIzquierda;
-
9   Servo MiServo;
-
10  void setup( ) {
```

```
11    Serial.begin(9600);
12    pinMode(PinDerecho, INPUT);
13    pinMode(PinIzquierdo,INPUT);
14    MiServo.attach(PinServo);
15    MiServo.write(PosicionAngular); // Iniciamos servo en 90º
16  }
-
17  void loop( ) {
-
18      LuzDerecha =analogRead(PinDerecho);     // Leo LDR DERECHO
19      LuzIzquierda =analogRead(PinIzquierdo); // Leo LDR IZQUIERDO
-
20      Serial.print("( ");
21      Serial.print(LuzDerecha);
22      Serial.print(" , ");
23      Serial.print(LuzIzquierda);
24      Serial.print(" , ");
25      Serial.print(LuzIzquierda-LuzDerecha);
26      Serial.print(" , ");
27      Serial.print(PosicionAngular);
28      Serial.println(" )");
-
29      if (LuzDerecha > LuzIzquierda + ZonaTolerancia) {
30          girarDcha( );
31      }
32      if (LuzIzquierda > LuzDerecha + ZonaTolerancia) {
32          girarIzda( );
33      }
34      delay(300);
35  }
-
36  void girarDcha( ) {
37    if (PosicionAngular < 179) {
38      PosicionAngular++;
39      MiServo.write(PosicionAngular);
40    }
41  }
-
42  void girarIzda( ) {
43    if (PosicionAngular > 1) {
44      PosicionAngular--;
45      MiServo.write(PosicionAngular);
46    }
47  }
```

CIRCUITO PARA NUESTRO PROYECTO

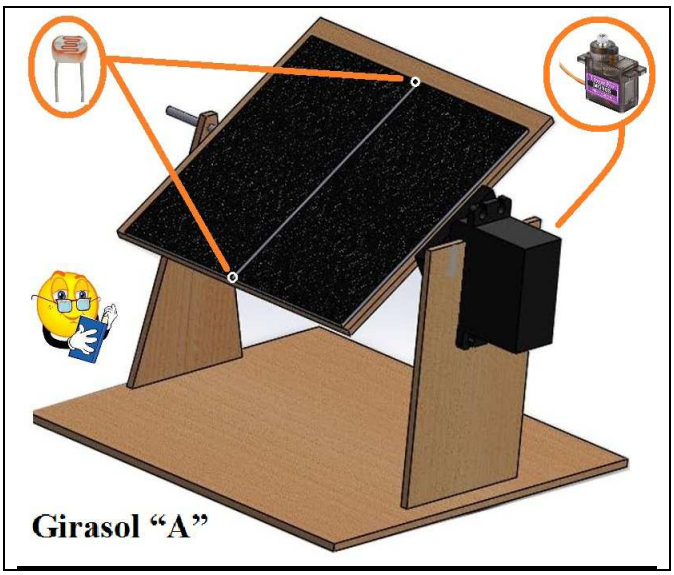


Lista de Materiales: 1 Servo Motor **MG90 o MG90S** – 2 Foto Resistencias LDR – 2 resistencias de 10 KΩ 1/4W – Cables conectores - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

Preste atención a la forma de conectar los LDR. La secuencia A Mayor Luz, Mayor Voltaje (5V-LDR-PIN-Resistencia-GND).

Como debemos conectar el Servo Motor

Los cables deberán ser conectados: En la imagen esta perfectamente explicado a donde conectar cada uno de los cables.



Montaje del dispositivo:

El armado del seguidor de luz o girasol, es muy simple, acá dejo una imagen para que tomen la idea. Los materiales pueden se cartón, o madera de poco espesor (4 milímetros o menos). Si les gusta trabajar con madera, recomienzo "Madera Balsa" muy liviana, fácil de trabajar para usar en maquetas

Lo Importante es que:

a- Las **Foto Resistencias Ldr** se encuentren simétricamente y opuestas (Ver imagen).

b- La placa sobre las que aplicamos las Foto Resistencias, debe estar pegada al Eje y en equilibrio. De esta forma el Servo Motor hace el menos fuerza, minimizando el consumo y evitando posibles calentamientos del equipo.

7- Radar Ultrasónico, posee una capacidad de giro de 180 grados, usa un único Servo Motor y un Sensor HC-SR04. Proyecto A.

Este dispositivo es capaz de detectar un objeto e indicar a que distancia se encuentra. Solo esta limitado por la capacidad de giro del servo motor, y por la distancia que nuestra servo es capas de soportar.

(Programa "008_Servo_05_MG90S_Radar_HC_SR04_Proyecto_A")

```
1   #include <Servo.h>
-
2   const int PinLedVerde = 3,
3             PinLedRojo  = 5;
-
4   const int PinServo    = 7,
5             PinPulsoSaliente = 9,
6             PinPulsoEntrante = 11;
-
7   int     Angulo,  // Angulo al que rotará el servo
8           ValorGiro,
```

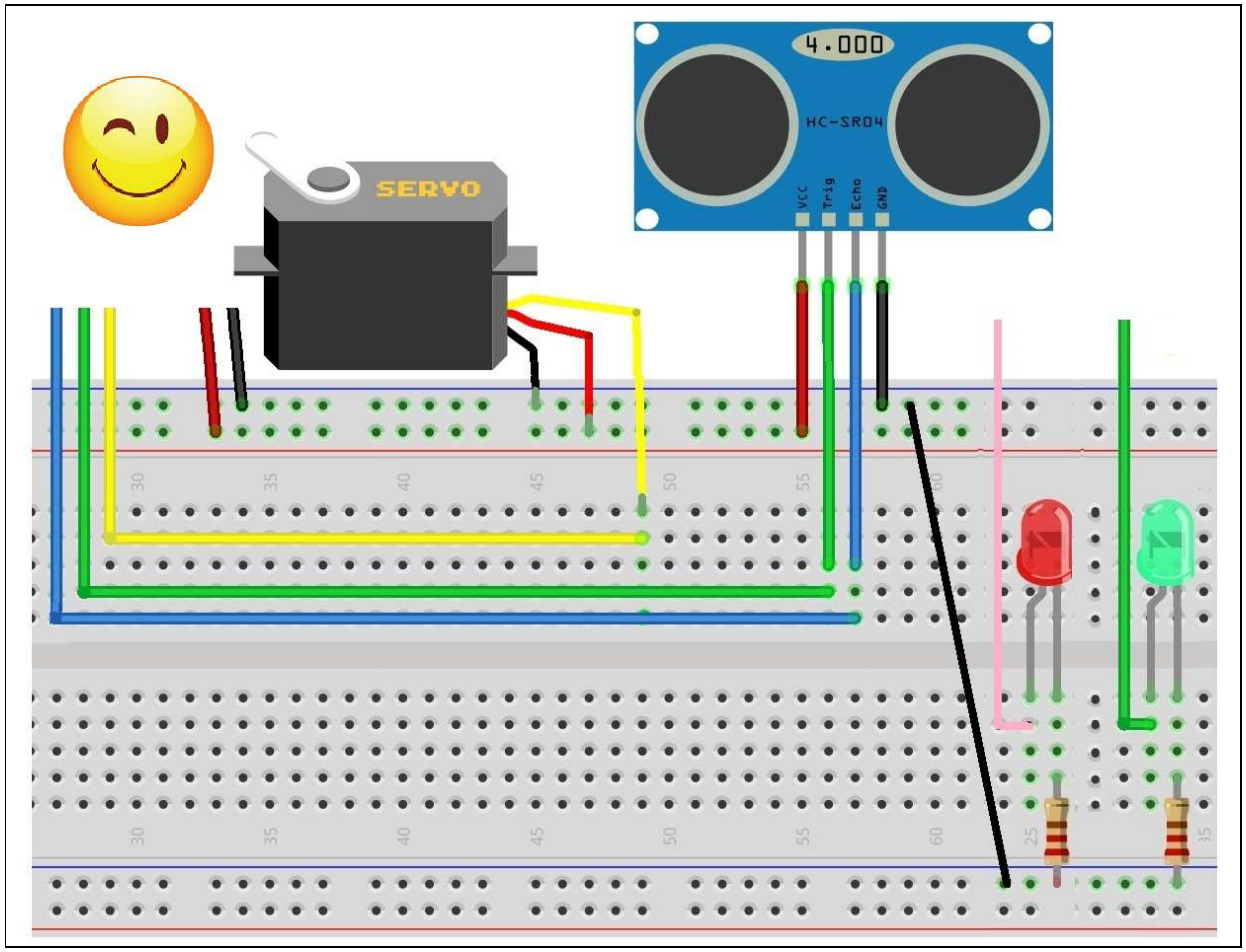
```
9           DistanciaSeguraCentimetros = 20,
10          EstadoAlarma;
-
11  Servo MiServo;   // Crea Objeto que Manejara Servo
-
12  long TiempoTotal,
13      DistanciaCentimetros,
14      TiempoAlarmaBeep,
15      TiempoIniAlarma;
-
16  void setup( ){
17   Serial.begin(9600);
18   MiServo.attach(PinServo);  // Vincula el Servo al  Pin
19   pinMode(PinPulsoSaliente, OUTPUT);
20   pinMode(PinPulsoEntrante, INPUT);
21   pinMode(PinLedVerde, OUTPUT);
22   pinMode(PinLedRojo,  OUTPUT);
23   EstadoAlarma = 0;
24   TiempoAlarmaBeep = 250;
25   TiempoIniAlarma = millis( );
26   Angulo= 90;
27   ValorGiro = 1; // Comienza Sumando (Agranda ángulo)
28  }
-
29  void loop( ){
30  /****************** ROTACION DEL CERVO ********************/
31   Angulo = Angulo + ValorGiro;
32   if( Angulo < 1 || Angulo > 179 ){
32     ValorGiro *= -1;
33   }
34   MiServo.write(Angulo);
35  /**************************************************************/
36  /************* MANEJO SENSOR SENSOR HC-SR04 ***************/
37   digitalWrite(PinPulsoSaliente,LOW);
38   delayMicroseconds(3); // entre 2 y 5 Milisegundos
39   digitalWrite(PinPulsoSaliente, HIGH); /*envío pulso ultrasónico*/
40   delayMicroseconds(10);
-
41   TiempoTotal = pulseIn(PinPulsoEntrante, HIGH);
42   DistanciaCentimetros = int(0.017*TiempoTotal);
43  /************************************************************/
44  /******************* LEDs DE ALARMA ************************/
45  if(DistanciaCentimetros <= DistanciaSeguraCentimetros){
46      digitalWrite(PinLedVerde, LOW);
47      if(millis( ) > TiempoIniAlarma + TiempoAlarmaBeep ){
48        EstadoAlarma = 1 - EstadoAlarma;
49        TiempoIniAlarma = millis( );
50      }
51      digitalWrite(PinLedRojo, EstadoAlarma);
52    }else{
53      digitalWrite(PinLedVerde, HIGH);
54      digitalWrite(PinLedRojo, LOW);
55      TiempoIniAlarma = millis( );
56      EstadoAlarma = 0;
```

En este Bloque, se apaga el LED Verde, que indica que todo esta OK y se hace titilar el LED Rojo de Alarma

```
57    }
58  /*************************************************************/
59  /***************** MENSAJES PUERTO SERIE *****************/
60    Serial.print("Distancia en Centimetros: ");
61    Serial.print(DistanciaCentimetros);
62    Serial.println(" cm");
63   // delay(50);  // Puede activarlo si quiere que todo funcione más rápido
64  /*************************************************************/
65  }
```

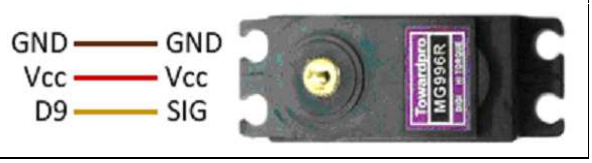
CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Servo Motor **MG90 o MG90S** – 1 Sensor de Distancia Ultrasónico HC-SR04 – 2 Leds 5mm – 2 resistencias de 470Ω (para los Leds) – 12 cables conectores - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.



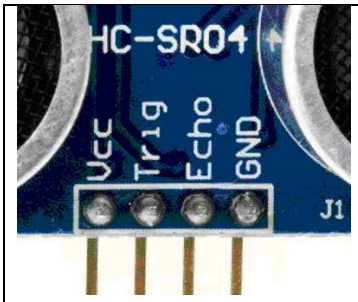
<u>Los cables deberán ser conectados:</u> (Arriba de izquierda a derecha) Cable Azul, al Pin de entrada o recepción de Pulso Sensor **HC-SR04.** Cable Verde, al Pin de Salida o emisión de Pulso sensor **HC-SR04.** Cable Amarillo, al Pin de la Placa Arduino que usaremos para controlar el Servo Motor. Cable Rojo Oscuro, 5V de placa Arduino. Cable Negro, GND de la Placa arduino. Cable Rosado, Al pin de alarma configurado en Programa. Cable Verde, (el ultimo de arriba, a la derecha) lo conectamos al pin del LED que nos marca que todo esta OK.

Como debemos conectar el Servo Motor



<u>Los cables deberán ser conectados según modelo:</u>
Marrón (GND), Rojo (5V) y Naranja – (SIG o Señal)
Negro (GND), Rojo (5V) y Blanco – (SIG o Señal).
Aclaración: Naranja o Blanco al Pin de la Placa Arduino que usaremos para controlar el servo.

Pines de Conexión



VCC	5V
Trig	Pin de Salida
Echo	Pin de Entrada
GND	GND

En el montaje del dispositivo (Servo **MG90 o MG90S** y Sensor **HC-SR04)** se usaron Brackets comerciales. Aunque usted mismo puede Fabricarlos con Madera Balsa o Cartón. Puedes **fijar una linda tapa de frasco** con tornillos al Servo, y ya tendrás una bonita plataforma redonda capaz de girar. Este dispositivo es muy fácil de construir, y puedes darle muy buena terminación.

8- Radar Ultrasónico – Posee una capacidad angular para detectar de 320 grados girando solo 180º. <u>Proyecto B.</u>

Para esto usaremos 2 sensores Ultrasónicos rotados como se muestra en la figura y un Servo Motor. Este dispositivo es capaz de detectar un objeto e indicar a que distancia se encuentra. Solo esta limitado por la capacidad de giro del servo motor, y la distancia que nuestros sensores sean capaces de soportar.

(Programa "008_Servo_05_MG90S_Radar_HC_SR04_Proyecto_B")

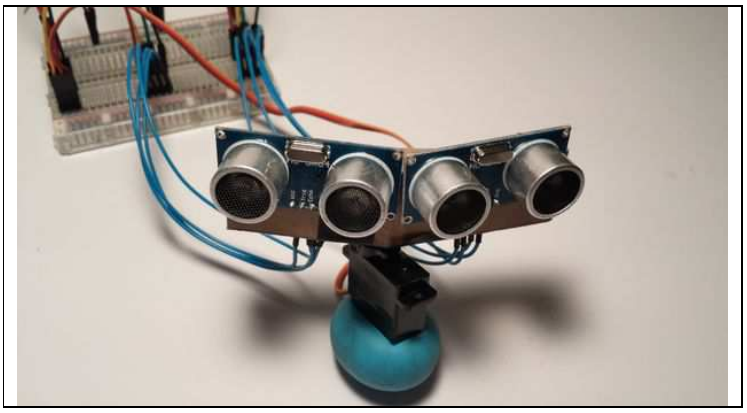
```
1   #include <Servo.h>
-
2   const int PinLedVerde = 3,
3         PinLedRojo  = 4;
```

```
-
4   const int PinServo    = 6,
5           PinS1_PulsoSaliente = 8,
6           PinS1_PulsoEntrante = 9,
7           PinS2_PulsoSaliente = 11,
8           PinS2_PulsoEntrante = 12;
-
9   int      Angulo,  // Angulo al que rotará el servo
10          ValorGiro,
11          DistanciaSeguraCentimetros = 20, //Centimetros
12          EstadoAlarma;
-
13  Servo MiServo;   // Crea Objeto que Manejará Servo
-
14  long TiempoTotal,
15      Distancia01_Centimetros,
16      Distancia02_Centimetros,
17      TiempoAlarmaBeep,
18      TiempoIniAlarma;
-
19  void setup( ){
20   Serial.begin(9600);
21   MiServo.attach(PinServo);  // Vincula el Servo al  Pin
22   pinMode(PinS1_PulsoSaliente, OUTPUT);
23   pinMode(PinS2_PulsoSaliente, OUTPUT);
24   pinMode(PinS1_PulsoEntrante, INPUT);
25   pinMode(PinS2_PulsoEntrante, INPUT);
26   pinMode(PinLedVerde, OUTPUT);
27   pinMode(PinLedRojo,  OUTPUT);
28   EstadoAlarma = 0;
29   TiempoAlarmaBeep = 250;
30   TiempoIniAlarma = millis( );
31   Angulo= 90;
32   ValorGiro = 1; // Comienza Sumando (Agranda ángulo)
32  }
33  void loop( ){
34  /***************** ROTACION DEL CERVO ********************/
35   Angulo = Angulo + ValorGiro;
36   if( Angulo < 1 || Angulo > 179 ){
37     ValorGiro *= -1;
38   }
39   MiServo.write(Angulo);
40  /*************************************************************/
41  /************ MANEJO SENSOR SENSOR HC-SR04 ***********/
42    digitalWrite(PinS1_PulsoSaliente,LOW); // Sensor 01 - Apagado
43    digitalWrite(PinS2_PulsoSaliente,LOW); // Sensor 02 - Apagado
44    delayMicroseconds(3); // entre 2 y 5 Milisegundos
45    digitalWrite(PinS1_PulsoSaliente, HIGH); /*Sensor 01 - envío pulso ultrasónico*/
46    digitalWrite(PinS2_PulsoSaliente, HIGH); /*Sensor 02 - envío pulso ultrasónico*/
47    delayMicroseconds(10);
48    TiempoTotal = pulseIn(PinS1_PulsoEntrante, HIGH); // Leo Pulso Entrante Sensor 01
49    Distancia01_Centimetros = int(0.017*TiempoTotal);  // Calculo Distancia Sensor 01
-
50    TiempoTotal = pulseIn(PinS2_PulsoEntrante, HIGH); // Leo Pulso Entrante Sensor 02
```

```
51    Distancia02_Centimetros = int(0.017*TiempoTotal); // Calculo Distancia Sensor 02
52  /*********************************************************/
53  /****************** LEDs DE ALARMA *********************/
54    if(Distancia01_Centimetros <= DistanciaSeguraCentimetros ||
55              Distancia02_Centimetros <= DistanciaSeguraCentimetros){
56      digitalWrite(PinLedVerde, LOW);
57      if(millis( ) > TiempoIniAlarma + TiempoAlarmaBeep ){
58        EstadoAlarma = 1 - EstadoAlarma;
59        TiempoIniAlarma = millis( );
60      }
61      digitalWrite(PinLedRojo, EstadoAlarma);
62    }else{
63      digitalWrite(PinLedVerde, HIGH);
64      digitalWrite(PinLedRojo, LOW);
65      TiempoIniAlarma = millis( );
66      EstadoAlarma = 0;
67    }
68  /*********************************************************/
69  /****************** MENSAJES PUERTO SERIE ******************/
70    Serial.print("Distancia en Centímetros: ");
71    Serial.print(Distancia01_Centímetros);
72    Serial.print(" - ");
73    Serial.println(Distancia02_Centímetros);
74    delay(50);
75  /*********************************************************/
76  }
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Servo Motor **MG90 o MG90S** – 2 Sensores de Distancia Ultrasónico HC-SR04 – 2 Leds 5mm – 2 resistencias de 470Ω (para los Leds) – 16 cables conectores - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.



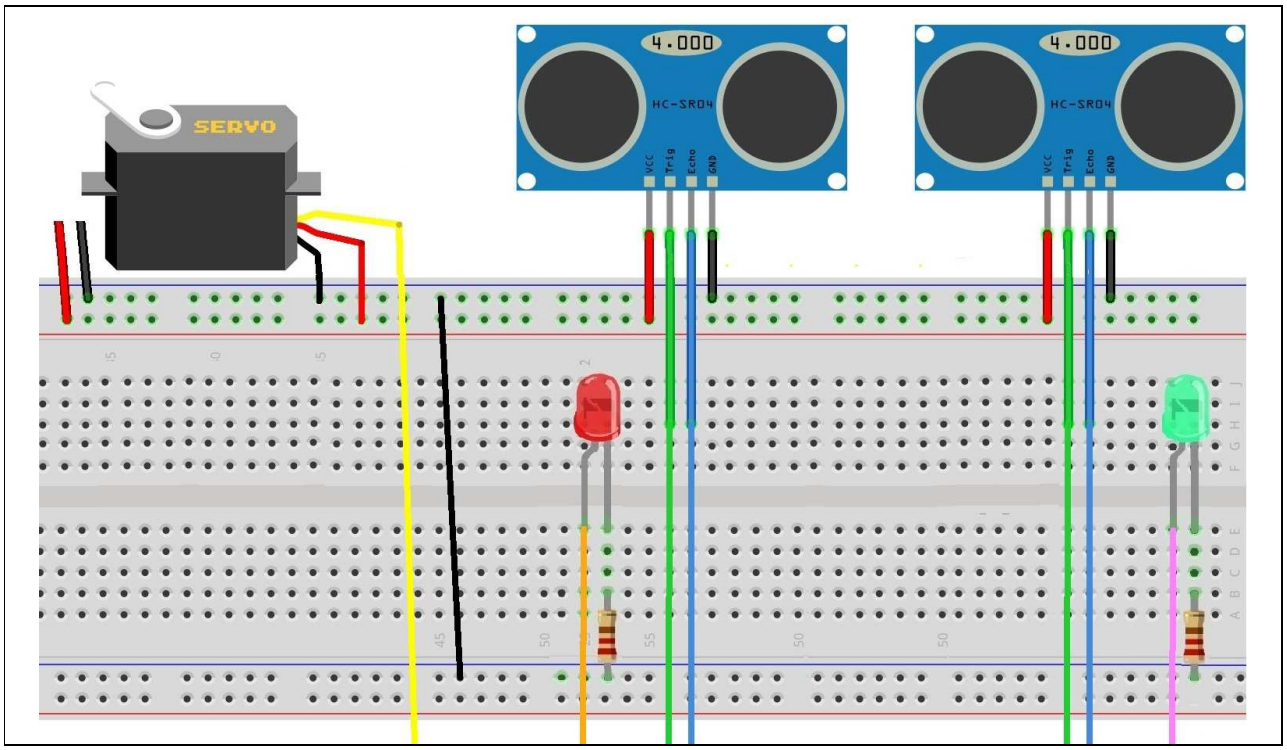
En el montaje del dispositivo (Servo **MG90 o MG90S** y Sensor **HC-SR04)** se usaron Brackets comerciales. Aunque usted mismo puede Fabricarlos con Madera Balsa o Cartón.

También puedes **fijar una linda tapa de frasco** con tornillos al Servo, y ya tendrás una bonita plataforma redonda capaz de girar.

Este dispositivo es muy fácil de construir, y podras jugar un buen rato con el. Además puedes darle muy buena terminación.

Los cables deberán ser conectados: Arriba, de izquierda a derecha. Cable Rojo a 5V y Cable negro a GND. Debajo de Izquierda a derecha. Cable amarillo, conectar al Pin Digital destinado al Servo Motor. Cable Marrón, debe conectarse al Pin Destinado al LED rojo de Alarma. Cable Verde, al Pin de Salida o emisión de Pulso sensor **HC-SR04.** Cable Azul, al Pin de entrada o recepción de Pulso Sensor **HC-SR04.** Nuevamente el otro cable Verde, al Pin de Salida o emisión de Pulso sensor **HC-SR04.** Cable Azul, al

Pin de entrada o recepción de Pulso Sensor **HC-SR04.** Finalmente el cable Rosa debe conectarse al Pin Destinado al LED Verde, que indicara que todo esta Bien.



Con un poco de ingenio, podemos modificar este Proyecto, para que puedas configurar mediante un potenciómetro la distancia a controlar con el radar, y además del LED rojo ya posee, podremos incorporar una alarma sonora (haga BEEP BEEP) que suene al detectar un intruso.

9- Manejar un Servo desde la PC por el puerto Serie. Proyecto A.

El programa leerá un número entero (Angulo de rotación) por el puerto serie y posicionara el servo. En este caso no podemos usar la función "**Serial.read( )**" ya que interpretará la entrada por Serial como una secuencia de caracteres y no como un número entero. Para ello utilizaremos la función "**Serial.parseInt( )**" que interpreta la entrada por serial como un número entero. Recordatorio: para mover el servo utilizaremos la librería "**Servo.h**" ya provista por el IDE Arduino. y que los servos **sólo pueden moverse de 0 a 180 grados** (excepto los servos de giro continuo).

(Programa "008_Servo_06_MG90S_Manejar_Un_Servo_Puerto_Serie")

```
1   #include <Servo.h>
-
2   #define PinServo 5
3   int Valor_Angulo,
4       Valor_Anterior;
5   Servo MiServo; //Creamos objeto Servo de nombre... MiServo
-
6   void setup( ){
7     Serial.begin(9600); //Iniciamos Puerto Serie
8     MiServo.attach(PinServo); //Conectamos servo a pin digital
9     Valor_Angulo = 0;            // Inicio Valores de Control
10    Valor_Anterior = 0;          // Inicio Valores de Control
11    MiServo.write(Valor_Angulo); // Posiciono Servo
12  }
13  void loop( ){
```

```
14    if(Serial.available( ) > 0){ //Detecta entrada de datos
15      Valor_Angulo = Serial.parseInt( );
16      if(Valor_Angulo != Valor_Anterior &&
17                          Valor_Angulo >=180 0 &&
18                                              Valor_Angulo <= 180){
19          MiServo.write(Valor_Angulo); //Mueve servo a posición
20          Valor_Anterior = Valor_Angulo;
21      }else{
22        // Hacer sonar alarma sin detener programa
23      }
24    }
25    //delay(500);
26  }
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Servo Motor **MG90 o un MG90S** – 3 Cables conectores –1 Placa Protoboard - Placa Arduino y 1 Cable USB.

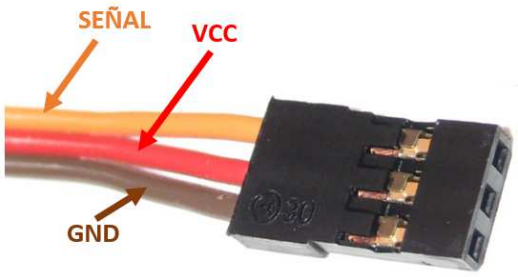
Los cables deberán ser conectados según modelo:

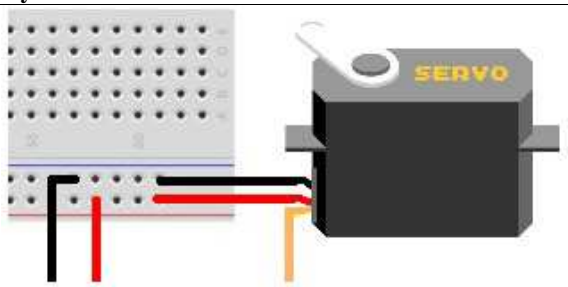
- Marrón (GND), Rojo (5V) y Naranja – (SIG o Señal)
- Negro (GND), Rojo (5V) y Blanco – (SIG o Señal).

Aclaración: Naranja o Blanco al Pin de la Placa Arduino que usaremos para controlar el servo.

Conexión Servo Motor **MG90S**

Conexión a placa Arduino y/o Protoboard.





Recordar que Arduino puede proporcionar alimentación suficiente para encender y manejar un (UNO) servo pequeño, como el SG90 o SG90S. Próximamente usaremos el Controlador de Servos "PCA9685" que nos permitirá manejar hasta 12 Servos Motores simultáneamente (Por cada placa controladora que usemos).

Apéndice A - Librería "Servo.h" Clases y Métodos disponibles

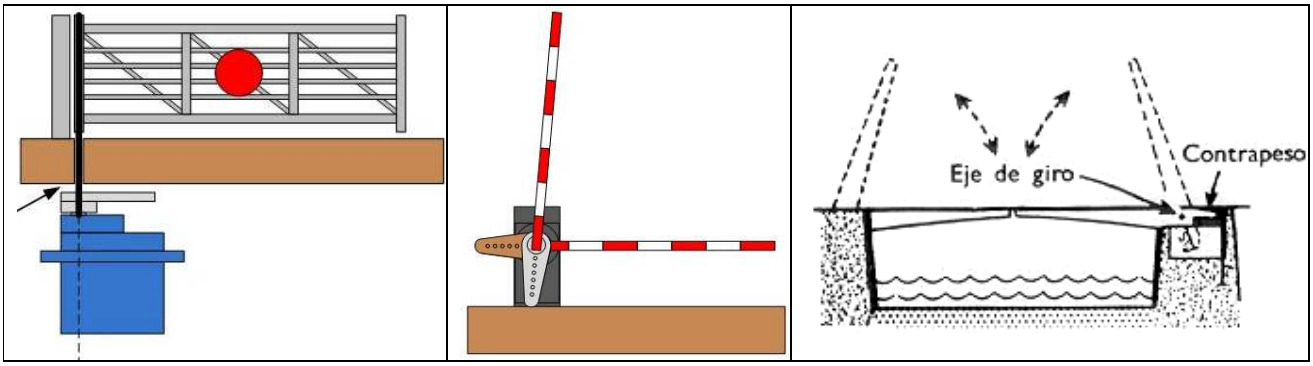
La librería Servo esta incluida en el IDE de arduino, y no es necesario instalarla, contiene la clase del mismo nombre, y fue diseñada para facilitar la comunicación de Arduino con los servomotores. La clase Servo, y por tanto, cualquier objeto que instanciemos a partir de la misma. Un Objeto del tipo de dato **Servo**, se declara o instancia de la siguiente forma:

```
Servo MiServoMotor;
```

Es a partir de este momento, que ya hemos declarado o instanciado una variable (objeto) que podremos usar los siguientes métodos:

attach( )	Recibe como argumento un número de pin (digital), y establece dicho pin para manejar el servomotor, de modo que, conectando el cable de control (generalmente blanco o amarillo) a dicho pin, podremos controlar el servomotor. Ejemplo: **MiServoMotor.attach(9);** Establezco el Pin digital 9, para control del servo motor
write( )	Recibe como argumento un valor entre 0 y 180, que envía a través del cable de control, haciendo que el servomotor gire su eje hasta el ángulo indicado.
writeMicroseconds( )	El nombre de este comando (método) es poco intuitivo, ya que no tiene nada que ver con tiempo. Realmente se refiere a transmitir un valor entre 1000 y 2000, para colocar el eje del servomotor en una posición específica, dentro de los 180 grados que puede girar. Por ejemplo **MiServoMotor.writeMicroseconds(1500);** colocaría el eje a la mitad de su recorrido de giro. Con el valor 1000 se colocaría totalmente a la izquierda y con el valor 2000 totalmente a la derecha. Sin embargo, algunos fabricantes desarrollan servomotores que aceptan otros valores límites, por ejemplo valores entre 700 y 2300, por lo que no es muy fiable usar este método, salvo que tengamos la hoja de especificaciones de nuestro servomotor concreto
read( )	Permite determinar el ángulo en el que se encuentra el eje de un servomotor en un momento dado. No recibe argumentos y devuelve un valor entre 0 y 179. Se usa como en la siguiente línea de ejemplo: **angulo = MiServoMotor.read( );** En la variable angulo se almacena un valor que corresponde con la posición del eje.
attached( )	Recibe como argumento un número de pin, y devuelve un valor booleano (true o false), según el motor esté o no asociado a dicho pin. Se usa como en la siguiente línea de ejemplo: **asociado = MiServoMotor.attached(9);** Si el objeto **MiServoMotor** está asociado al pin 9, la variable adquiere un valor **true**; en caso contrario, la variable recibe un valor **false**. La variable empleada (en este caso, asociado) debe haber sido definida previamente. Por ejemplo: **boolean asociado = false;**
detach( )	Este comando (método) es lo contrario del comando (método) attach( ). Desasocia el servomotor de su pin. No recibe argumento y no devuelve valor alguno. Su uso sería así: MiMotorServo.detach( ); Una vez se ejecuta este comando, el servo no puede ser usado hasta que se se asocie nuevamente a un Pin, esta vez mediante el comando "**attach( )**"

Puedes construir Increíbles Maquetas. Acá te dejo IDEAS..!!!



Puedes usar también un sensor de Proximidad, para abrir el portón, o una alarma sonora y unos led cuando la barrera este baja, y puedes pasar horas de gran diversión. En el puente puedes usar un potenciómetro para levantar y bajar el puente, o un sensor de proximidad que haga que se levante automáticamente….Otra Alternativa es controlar todo desde la PC.
Usa tu imaginación!

01000011 01110101 01100001 01101100 01110001 01110101 01101001 01100101 01110010
00100000 01110100 01100101 01100011 01101110 01101111 01101100 01101111 01100111
11000011 10101101 01100001 00100000 01110011 01110101 01100110 01101001 01100011
01101001 01100101 01101110 01110100 01100101 01101101 01100101 01101110 01110100
01100101 00100000 01100001 01110110 01100001 01101110 01111010 01100001 01100100
01100001 00100000 01100101 01110011 00100000 01100101 01110001 01110101 01101001
01110110 01100001 01101100 01100101 01101110 01110100 01100101 00100000 01100001
00100000 01101100 01100001 00100000 01101101 01100001 01100111 01101001 01100001
00101110

Si tienes algunas Correcciones y/o Sugerencias, por favor contáctame.