ALIMENTACIÓN EXTERNA (Parte I)

Comenzamos con una de las cosas más interesantes de la robótica. La alimentación externa de nuestro dispositivo. Ella nos permitirá alejarnos de nuestra PC a la vez que nos independizarnos de la energía que pueda suministrarnos la Placa Arduino.
Y más aun, que nuestra placa Arduino, se independice de la nuestra PC o un cable para funcionar.

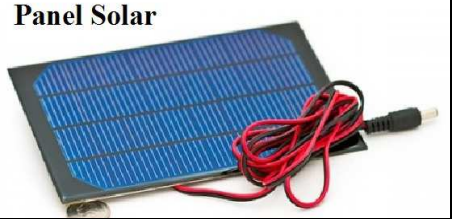
Esto nos permitirá controlar dispositivos independientes, con fuente de alimentación propia. No importa si se trata de un robot, un girasol, algún tipo de vehículo, o un sensor que dejemos conectados para monitorear un evento.

OPCIONES MÁS COMUNES

Como alimentar externamente un dispositivo, **lo dividiremos en dos partes**, ***en la primera,*** Arduino es alimentado por la PC, mientras controla algún elemento del dispositivo, alimentado externamente (Pilas, fuente, etc.), comenzaremos con un Servo Motor, y **en la segunda parte**, alimentaremos con una fuente o Pilas, a todo el dispositivo, incluyendo nuestra Placa Arduino. Desde este momento, seremos capases de construir dispositivos independientes, capases de deslazarse con relativa independencia.

Fuente ATX

Panel Solar



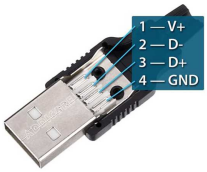
Hay muchas y variadas opciones para usar como fuentes de alimentación de nuestros dispositivos (Pilas, Baterías, Transformadores, etc.), pero por simplicidad, comenzaremos usando un Porta pilas con 4 Pilas comunes AA (1,5V - No Recargables), o si prefieres, 5 Pilas AA Recargables de 1,2V.

MUY IMPORTANTE: Cuando en nuestro dispositivo usemos más de una fuente de alimentación (Corriente Continua), debemos unir los Negativos de las fuentes, o lo que es igual, el o los negativos de las fuentes alternativas al GND de nuestra Placa Arduino. Esta conexión SIEMPRE debe ser hecha, para que las tierras (Masa o negativo) sean comunes en TODO el dispositivo, y la referencia de voltaje sea la misma.

Alimentar externamente nuestro proyecto es un gran salto hacia delante en el largo camino de la Robótica.

Comenzaremos Repitiendo algunos trabajos, pero este vez, los componentes de nuestro dispositivo (Servos, motores, etc.), serán alimentados por Pilas, y arduino (alimentado desde la PC) controlará las acciones que nosotros programemos, tal y como lo hemos venido haciendo hasta este momento. Siempre que podamos, veremos los dispositivos de dos formas, usando alimentación desde arduino y alimentación externa, de esta forma, podremos comparar y ver las diferencias.

DEBES SABER QUE: El estándar USB 1.1 original especificó una corriente máxima de **500 miliamperios (mA) a 5 voltios (2.5 vatios)**, y USB 2.0 permitió alcanzar este mismo valor máximo. Esto se modificó con la especificación USB 3.1, que permite alcanzar una corriente máxima de 900 mA.



La alimentación proviene de los pines 1 y 4, y los datos de los pines 2 y 3.

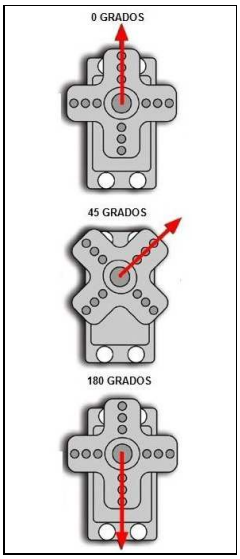
Cargadores USB: Usualmente los cargadores **USB** tienen una tensión de salida de 5 **voltios**; lo que varía es el amperaje que admiten. Deben estar entre 500 mA y 1.5 A (o 1500mA, que es lo mismo). Mientras mayor sea este valor, más rápido se cargará el equipo, si es compatible.

1) Movimientos básicos de un Servo Motor con alimentación externa.

Usaremos el código que usamos anteriormente, pero cambiaremos el circuito de nuestro dispositivo. La placa Arduino continuará siendo alimentada desde la PC por el cable USB, mientras controla el Servo Motor, que será alimentado con 4 Pilas comunes AA (1,5V - No Recargables).

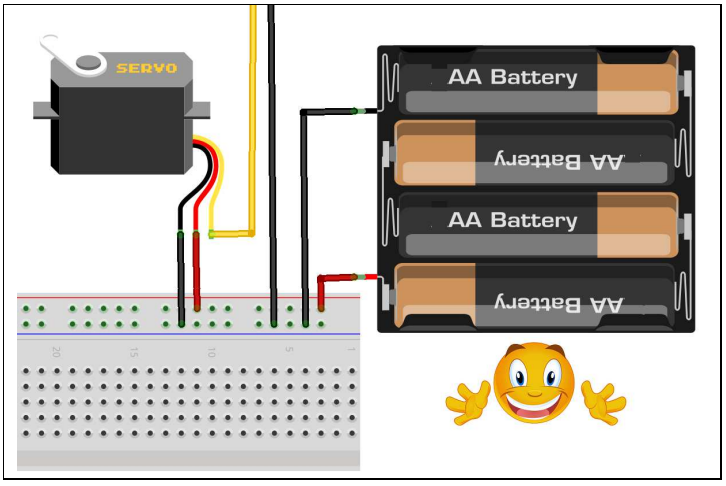
(Programa "008_Servo_01_MG90S_B")

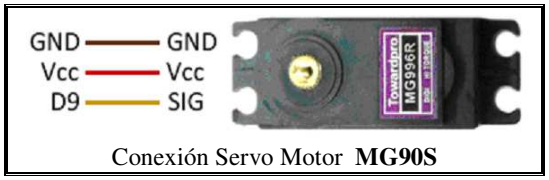
```
1   #include <Servo.h>
-
2   const int PinServo = 9;
3   int       Angulo,  // Angulo al que rotará el servo
4             ValorGiro;
-
5   Servo MiServo;   // Crea Objeto que Manejará Servo
-
6   void setup( ){
7     MiServo.attach(PinServo);  // Vincula el Servo al  Pin
8     Angulo= 90;     // Posiciono en el medio
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 Porta Pilas para 4 pilas comunes AA (1,5V) - 1 Servo Motor **MG90 o un MG90S** – 3 Cables Macho/Macho –1 Placa Protoboard - Placa Arduino y 1 Cable USB.





Conexión Servo Motor **MG90S**

Recordar que Arduino puede proporcionar alimentación suficiente para encender y manejar un (UNO) servo pequeño, como el SG90 o SG90S. **Sin embargo en este trabajo**, usamos pilas para que podamos analizar un ejemplo práctico muy simple de alimentación externa (No suministrada por Arduino).

Los cables deberán ser conectados: Amarillo, conectar al pin que configuramos en nuestro programa, para controlar el Servo Motor (PIN 9 de la Placa Arduino). El cable Negro, deberá ser conectado al **GND** de nuestra Placa para igualar las referencias de voltaje de las pilas y nuestra Placa Arduino.

MUY IMPORTANTE: Cuando en nuestro dispositivo usemos más de una fuente de alimentación, deberemos unir los Negativos de las fuentes, o lo que es igual, el o los negativos de las fuentes alternativas al GND de nuestra Placa Arduino. Esta conexión SIEMPRE debe ser hecha, para que las tierras (Masa o negativo) sean comunes en TODO el dispositivo, y la referencia de voltaje sea la misma.

2) Servo Motor controlado por un Potenciómetro con alimentación externa.

Con un potenciómetro, controlar el ángulo de giro de un Servo Motor. Al Angulo de giro debo oscilar entre 0 y 180. Usaremos un código que usamos anteriormente, pero cambiaremos el circuito de nuestro dispositivo. La placa Arduino continuará siendo alimentada desde la PC por el cable USB, mientras controla el Servo Motor y valores entregados por el Potenciómetro, quines serán alimentados con 4 Pilas comunes AA (1,5V - No Recargables).

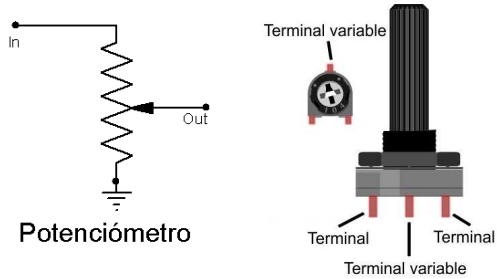
(Programa "008_Servo_02_MG90S_con_Potenciometro_01")

```
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 Porta Pilas para 4 pilas comunes AA (1,5V) - 1 Servo Motor **MG90 o un MG90S** – 8 Cables Macho/Macho –1 Placa Protoboard - Placa Arduino y 1 Cable USB.

Conexión Servo Motor **MG90S**

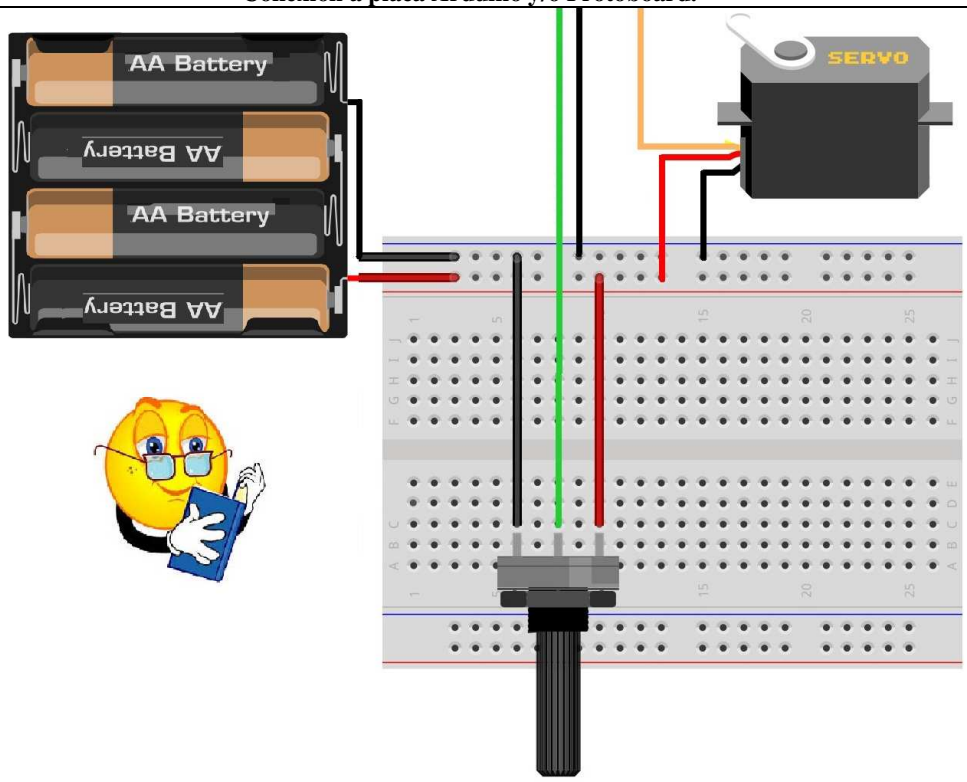


Conexión de un Potenciómetro – Terminal Variable A una entrada Analógica – Las Terminales se conectan a GND y 5V (Indistintamente).

<u>Los cables deberán ser conectados:</u> (Arriba de izquierda a derecha) Cable Verde, al PIN Analogico de nuestra Placa Arduino, que hayamos configurado en nuestro programa para leer los valores que nos entrega el Potenciómetro. Cable Negro al GND de la placa. Cable Amarillo, al Pin Digital que hayamos destinado desde nuestro programa para controlar el Servo.

Arduino puede proporcionar alimentación suficiente para encender y manejar un (UNO) servo pequeño, como el SG90 o SG90S. **Sin embargo en este trabajo**, usamos pilas para que podamos analizar un ejemplo práctico muy simple de alimentación externa (No suministrada por Arduino).

Conexión a placa Arduino y/o Protoboard.



MUY IMPORTANTE: Cuando en nuestro dispositivo usemos más de una fuente de alimentación, deberemos unir los Negativos de las fuentes, o lo que es igual, el o los negativos de las fuentes alternativas al GND de nuestra Placa Arduino. Esta conexión SIEMPRE debe ser hecha, para que las tierras (Masa o negativo) sean comunes en TODO el dispositivo, y la referencia de voltaje sea la misma.

3) Radar Ultrasónico con alimentación externa.– Posee una capacidad angular para detectar de 320 grados girando solo 180º.

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. Nuevamente usamos un código, que habíamos usamos, pero cambiamos el circuito de nuestro dispositivo. La placa Arduino continuará siendo alimentada desde la PC por el cable USB mientras controla el servo motor, los Led y los 2 Sensores de Distancia Ultrasónico HC-SR04, que serán alimentados con 4 Pilas comunes AA (1,5V - No Recargables).

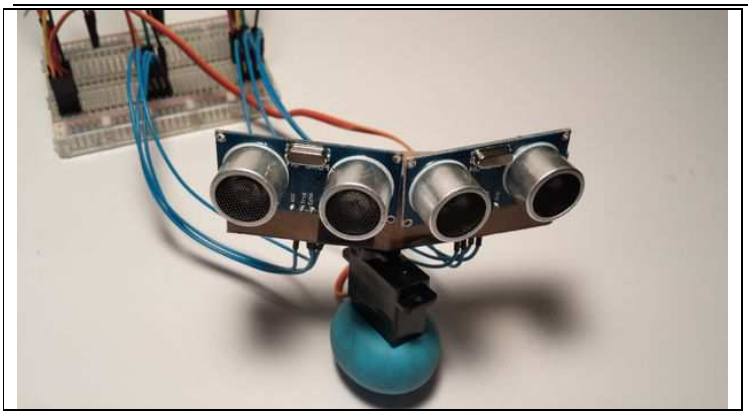
(Programa "008_Servo_05_MG90S_Radar_HC_SR04_Proyecto_B")

```
#include <Servo.h>

const int PinLedVerde = 3,
          PinLedRojo  = 4;
const int PinServo    = 6,
              PinS1_PulsoSaliente = 8,
              PinS1_PulsoEntrante = 9,
              PinS2_PulsoSaliente = 11,
              PinS2_PulsoEntrante = 12;

int      Angulo,  // Angulo al que rotará el servo
         ValorGiro,
         DistanciaSeguraCentimetros = 20, //Centimetros
         EstadoAlarma;

Servo MiServo;   // Crea Objeto que Manejará Servo

long TiempoTotal,
       Distancia01_Centimetros,
       Distancia02_Centimetros,
       TiempoAlarmaBeep,
       TiempoIniAlarma;

void setup( ){
  Serial.begin(9600);
  MiServo.attach(PinServo);  // Vincula el Servo al  Pin
  pinMode(PinS1_PulsoSaliente, OUTPUT);
  pinMode(PinS2_PulsoSaliente, OUTPUT);
  pinMode(PinS1_PulsoEntrante, INPUT);
  pinMode(PinS2_PulsoEntrante, INPUT);
  pinMode(PinLedVerde, OUTPUT);
  pinMode(PinLedRojo,  OUTPUT);
  EstadoAlarma = 0;
  TiempoAlarmaBeep = 250;
  TiempoIniAlarma = millis( );
  Angulo= 90;
  ValorGiro = 1; // Comoenza Sumando (Agranda ángulo)
}
void loop( ){
/****************** ROTACION DEL CERVO ********************/
 Angulo = Angulo + ValorGiro;
 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 Centimetros: ");
71    Serial.print(Distancia01_Centimetros);
72    Serial.print(" - ");
73    Serial.println(Distancia02_Centimetros);
74    delay(50);
75  /*************************************************************/
76  }
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Porta Pilas para 4 pilas comunes AA (1,5V) - 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.

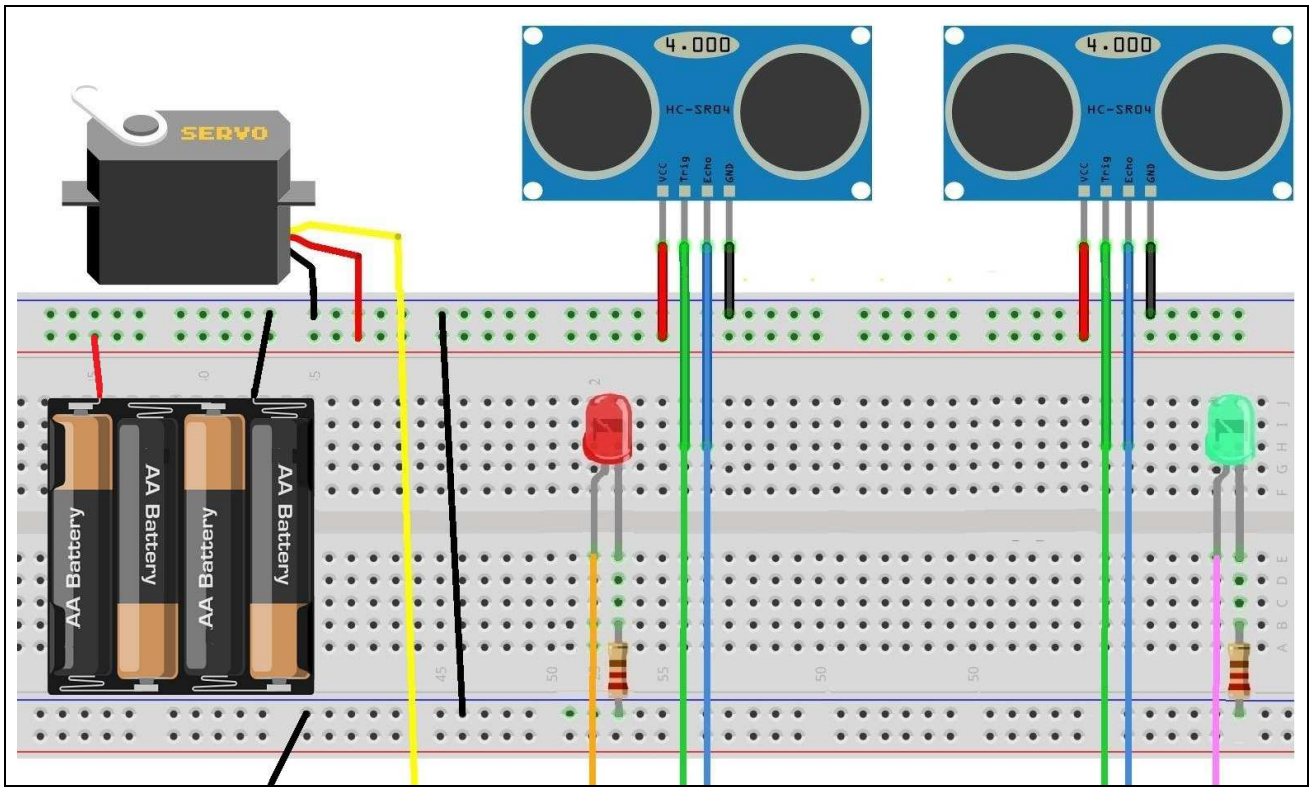


En el montaje del dispositivo (Servo **MG90 o MG90S** y Sensor **HC-SR04)** se usaron Brackets comerciales. Aunque usted mismo puede Fabricarlos con Madera Balsa o Cartón.

También puedes **fijar una linda tapa de frasco**, CD / DVD, con tornillos al Servo, y ya tendrás una funcional y elegante plataforma redonda capaz de girar.

Este dispositivo es muy fácil de construir, y podrás jugar un buen rato con el. Además podrás darle muy buena terminación, si le dedicas un mínimo de tiempo.

Los cables deberán ser conectados: Debajo de Izquierda a derecha. Cable Negro al GND de la placa Arduino (para que las Masas o negativos, sean comunes en TODO el dispositivo, y la referencia de voltaje sea la misma). 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.



MUY IMPORTANTE: Cuando en nuestro dispositivo usemos más de una fuente de alimentación, deberemos unir los Negativos de las fuentes, o lo que es igual, el o los negativos de las fuentes alternativas al GND de nuestra Placa Arduino. Esta conexión SIEMPRE debe ser hecha, para que las tierras (Masa o negativo) sean comunes en TODO el dispositivo, y la referencia de voltaje sea la misma.

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 que ya posee, podremos incorporar una alarma sonora (haga BEEP BEEP) que suene, al detectar un intruso.

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

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. Funcionamiento: mientras mantenemos apretado un botón, el servo gira hasta llegar al tope.

En este trabajo, usamos un código, que habíamos usamos, y cambiamos el circuito de nuestro dispositivo. La placa Arduino continuará siendo alimentada desde la PC por el cable USB mientras controla los 3 Led, el Speaker, los 2 botones y el servo motor, que será alimentados con 4 Pilas comunes AA (1,5V - No Recargables).

(Programa "008_Servo_03_MG90S_con_Dos_Botones_B")

```
#include <Servo.h>

const int PinParlante = 2,
          PinServo    = 3;

const int PinBotonIzquierdo = 7,
          PinBotonDerecho   = 8;

const int PinLedErrorIzquierda = 10, //Led Rojo Izquierdo
          PinLedOk             = 11,         //Led Verde Central
          PinLedErrorDerecha   = 12;    //Led Rojo Derecha

Servo MiServo;   // Crea variable que Manejara Servo

int BotonDerechoValor,  // Variable botón 01
    BotonIzquierdoValor;   // Variable botón 02

long TiempoInicial = millis( ), // Led Verd
     TiempoEspera  = 1000,     // Led Verde
     TISonidoTotal = millis( ),
     TIAlarmaDerecha = millis( ),
     TIAlarmaIzquierda = millis( ),
     TESonido = 500;

int  ValorFrecuencia = 1500,
     Frecuencia = 0;

int  EstadoAlarmaDerecha = LOW,
     EstadoAlarmaIzquierda = LOW;

int AnguloMaximo = 175,
    AnguloMinimo = 5,
    Angulo = (AnguloMaximo + AnguloMinimo)/2,
    AnguloAnterior = Angulo,
    Salto = 1,
    ErrorBotones;
```

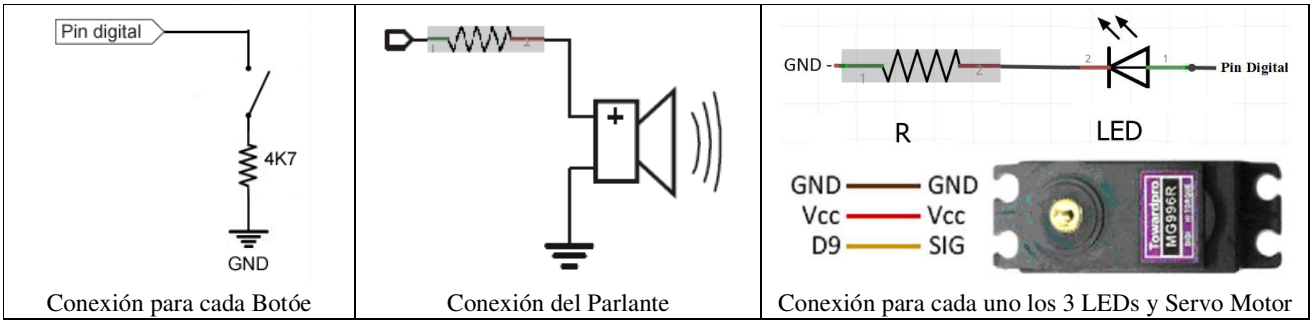
```
-
28  void setup( ){
29    Serial.begin (9600);
30    MiServo.attach(PinServo);  // Vincula el Servo al  Pin
31    pinMode(PinBotonIzquierdo, INPUT_PULLUP);
32    pinMode(PinBotonDerecho, INPUT_PULLUP);
32    pinMode(PinLedErrorIzquierda, OUTPUT);
33    pinMode(PinLedOk, OUTPUT);
34    pinMode(PinLedErrorDerecha, 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     }
```

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

```
-
74    if(ErrorBotones == 0 && Angulo != AnguloAnterior){
75       MiServo.write(Angulo);
76       AnguloAnterior = Angulo;
77    }
78  }  // Acá termina la Función loop
-
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   }
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 }
```

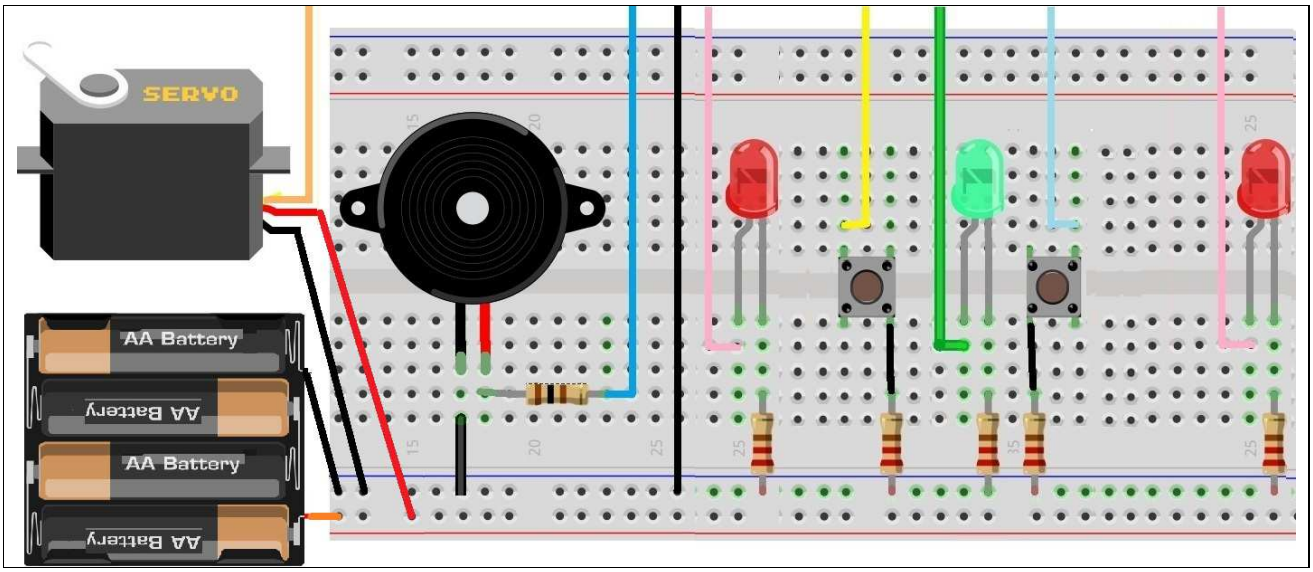
CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Porta Pilas para 4 pilas comunes AA (1,5V) - 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.



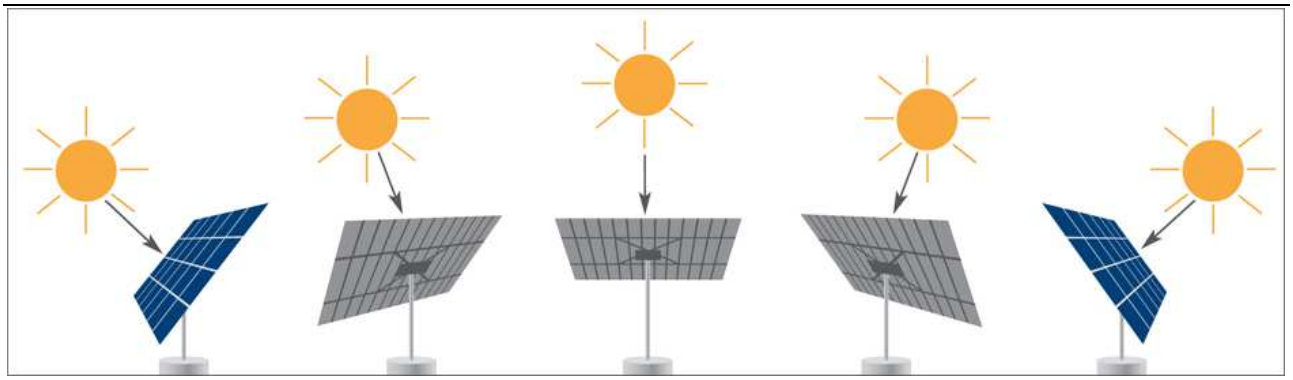
Conexión para cada Botóe | Conexión del Parlante | Conexión para cada uno los 3 LEDs y Servo Motor

Los cables deberán ser conectados: Arriba, de Izquierda a derecha: cable Marrón, al PIN Digital destinado a Manejar el Servomotor. Cable Azul, destinado al Parlante o Speaker. Cable Negro Conectar a GND de la placa Arduino (para que las Masas o negativos, sean comunes en TODO el dispositivo, y la referencia de voltaje sea la misma). Cable Rosa, Led rojo de 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 esta OK. Cable Celeste, al Pin destinado al botón de giro a la derecha, ultimo Cable de arriba, color Rosa, Led Rojo de alarma a la derecha.

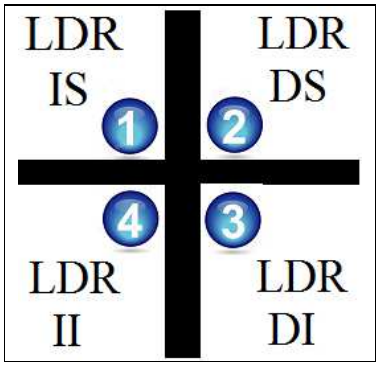


5) Seguidor de Luz (Girasol) Completo (en todas las direcciones - con dos servos). Alimentación Externa - Proyecto B.

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.
En este proyecto ahora podemos girar en cualquier dirección y buscar el mejor ángulo.



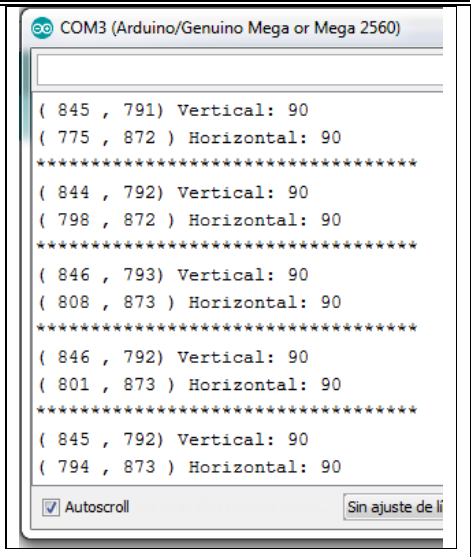
(Programa "008_Servo_04_MG90S_Girasol_B_Con_2_Servos")




```
1   #include<Servo.h>
-
2   #define PinIzquierdo_Superior A1 // Foto Resistencia LDR 01
3   #define PinDerecho_Superior A2   // Foto Resistencia LDR 02
4   #define PinDerecho_Inferior A3     // Foto Resistencia LDR 03
5   #define PinIzquierdo_Inferior A4   // Foto Resistencia LDR 04
6   #define PinServo_H 3    // Pin Para Control Servo Horizontal
7   #define PinServo_V 4    // Pin Para Control Servo Vertical
8   #define ZonaTolerancia 20
-
9   int PosAnguloLateral = 90,   // Posición Inicial
10      PosAnguloVertical = 90,  // Posición Inicial
11      LuzIzquierda_S,
12      LuzDerecha_S,
13      LuzDerecha_I,
14      LuzIzquierda_I;
-
15  Servo ServoHorizontal,
16        ServoVertical;
-
17  void setup( ) {
18    Serial.begin(9600);
19    while (!Serial) {
20     ; // Esperamos que el puerto serie este abierto.
21    }
22    pinMode(PinIzquierdo_Superior,INPUT);  // Foto Resistencia LDR 01
23    pinMode(PinDerecho_Superior, INPUT);   // Foto Resistencia LDR 02
24    pinMode(PinDerecho_Inferior, INPUT);   // Foto Resistencia LDR 03
25    pinMode(PinIzquierdo_Inferior,INPUT);  // Foto Resistencia LDR 04
26    ServoHorizontal.attach(PinServo_H);
27    ServoHorizontal.write(PosAnguloLateral); // Iniciamos servo en 90º
28    ServoVertical.attach(PinServo_V);
29    ServoVertical.write(PosAnguloVertical); // Iniciamos servo en 90º
30  }
-
31  void loop( ) {     //leemos la luz que recogen los Cuatro LDRs
32      LuzIzquierda_S =analogRead(PinIzquierdo_Superior); // Leo LDR IS
32      LuzDerecha_S   =analogRead(PinDerecho_Superior);   // Leo LDR DS
33      LuzDerecha_I   =analogRead(PinDerecho_Inferior);   // Leo LDR DI
34      LuzIzquierda_I  =analogRead(PinIzquierdo_Inferior); // Leo LDR II
35  /*************************************************************/
36  /*************** Izquierda - Derecha ***************************/
```

```
37      if(LuzIzquierda_S > LuzDerecha_S + ZonaTolerancia ||
38              LuzIzquierda_I > LuzDerecha_I + ZonaTolerancia){
39          girarIzda( );
40      }
41      if(LuzDerecha_S > LuzIzquierda_S + ZonaTolerancia ||
42              LuzDerecha_I > LuzIzquierda_I + ZonaTolerancia){
43          girarDcha( );
44      }
45  /*****************************************************************/
46  /********************** Arriba - Abajo ***************************/
47      if(LuzIzquierda_S > LuzIzquierda_I + ZonaTolerancia ||
48              LuzDerecha_S > LuzDerecha_I + ZonaTolerancia){
49          girarArriba( );
50      }
51      if(LuzIzquierda_I > LuzIzquierda_S + ZonaTolerancia ||
52              LuzDerecha_I > LuzDerecha_S + ZonaTolerancia){
53          girarAbajo( );
54      }
55  /*****************************************************************/
56      NuestraValores( ); // Muestra por Monitor valores
57      delay(800); // Esto permitirá ver más lento los resultados
58  }
 -
59  void girarDcha( ) {
60    if (PosAnguloLateral < 179) {
61      PosAnguloLateral++;
62      ServoHorizontal.write(PosAnguloLateral);
63    }
64  }
 -
65  void girarIzda( ) {
66    if (PosAnguloLateral > 1) {
67      PosAnguloLateral--;
68      ServoHorizontal.write(PosAnguloLateral);
69    }
70  }
 -
71  void girarAbajo( ) {
72    if (PosAnguloVertical > 1) {
73      PosAnguloVertical--;
74      ServoVertical.write(PosAnguloVertical);
75    }
76  }
 -
77  void girarArriba( ) {
78    if (PosAnguloVertical < 170) {
79      PosAnguloVertical++;
80      ServoVertical.write(PosAnguloVertical);
81    }
82  }
 -
83
84
85
```

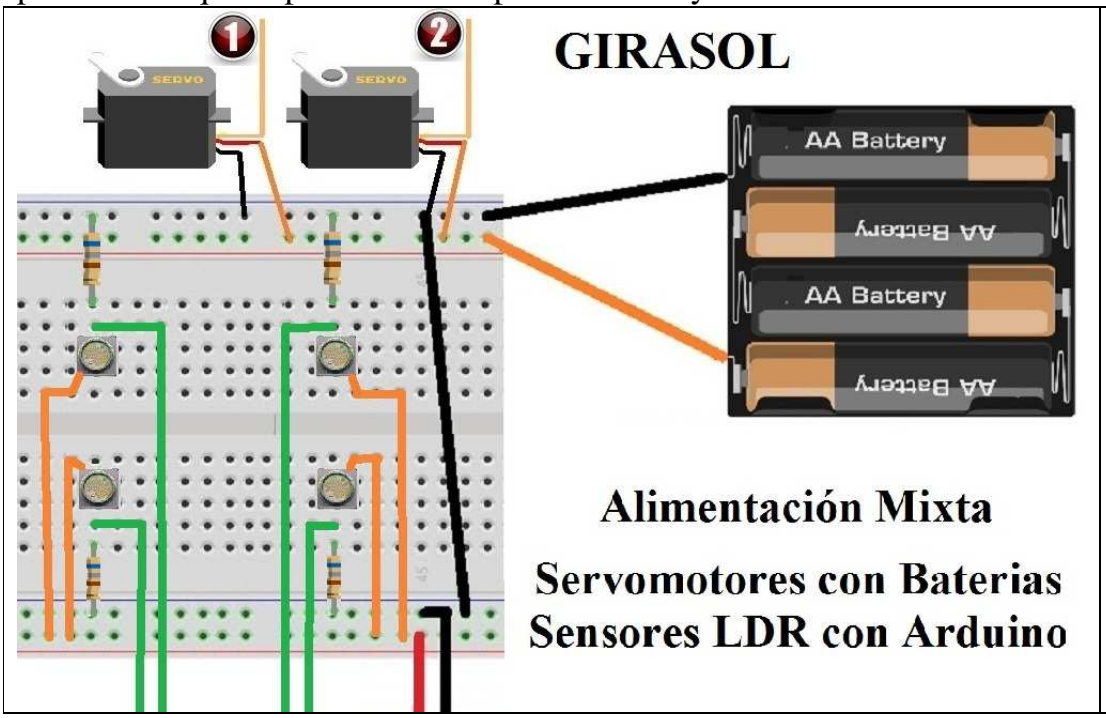
```
86
87  void NuestraValores( ){
88      Serial.print("( ");
89      Serial.print(LuzIzquierda_S);
90      Serial.print(" , ");
91      Serial.print(LuzDerecha_S);
92      Serial.print(")");
93      Serial.print(" Vertical: ");
94      Serial.println(PosAnguloVertical);
95      Serial.print("( ");
96      Serial.print(LuzIzquierda_I);
97      Serial.print(" , ");
98      Serial.print(LuzDerecha_I);
99      Serial.print(" )");
        Serial.print(" Horizontal: ");
        Serial.println(PosAnguloLateral);
       Serial.println("**************************");
  }
```



CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Porta Pilas para 4 pilas comunes AA (1,5V) – 2 Servo Motores - 4 Foto Resistencia LDR – 4 resistencia de 10 KΩ 1/4W - 18 Cables Macho/Macho – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

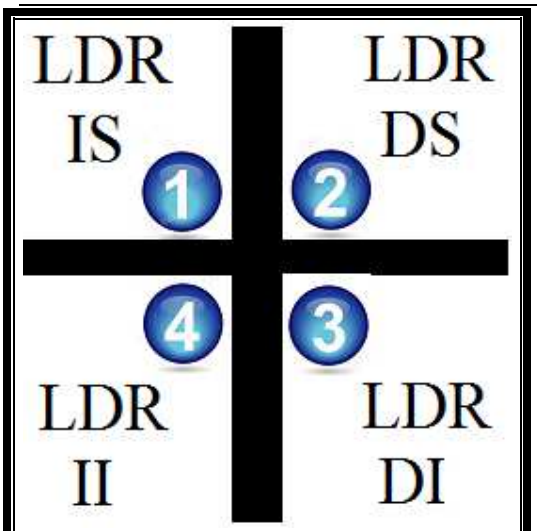
Piensa y comprende esto: Cuando pruebes la rotación de cada servo, acorde a la incidencia de luz (en cualquiera de los dispositivos), hay que tener en cuenta si los servos están fijos o ya se encuentran montados sobre las plataformas que le permiten al dispositivo rotar y buscar la luz.



RECORDAR

Conexión de Una Fotoresistencia LDR

Gráfica de conexiones LDR



1- LDR IS → LDR Izquierda Superior.
2- LDR DS → LDR Derecha Superior.
3- LDR DI → LDR Derecha Inferior.
4- LDR II → LDR Izquierda Inferior.

Los cables deberán ser conectados: Arriba, de derecha a izquierda, Cables Mostaza, correspondientes a los Servos 1 y 2. Servo 1, con movimiento Horizontal (Giratorio), Servo 2, con movimiento vertical (de abajo hacia arriba). Ahora los cables de abajo, de derecha a izquierda, En cuanto a la conexión de los pines de control de cada Foto Resistencia (Cables Verdes), nos guiaremos por la Gráfica de conexiones, y el segmento de condigo, tenemos que los Cables Verdes deben conectarse:

PinIzquierdo_Superior	en A1 → LDR IS o Foto Resistencia 01
PinDerecho_Superior	en A2 → LDR DS o Foto Resistencia 02
PinDerecho_Inferior	en A3 → LDR DI o Foto Resistencia 03
PinIzquierdo_Inferior	en A4 → LDR II o Foto Resistencia 04

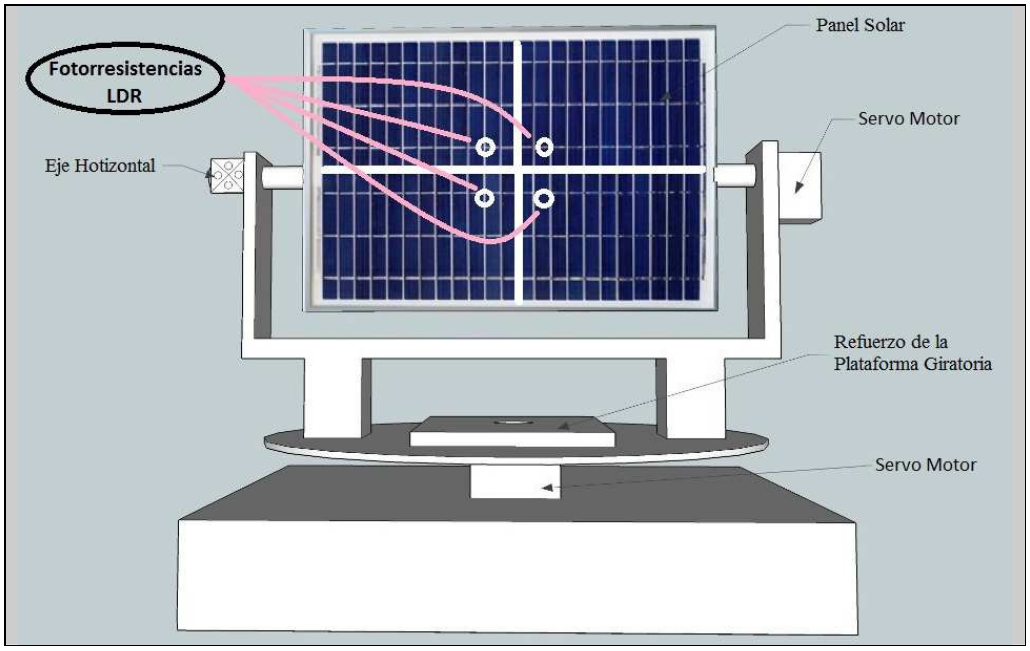
Y finalmente, los dos últimos cables de abajo, a la izquierda. Cable Rojo a 5V de placa Arduino, Cable negro a GND de arduino.

SIEMPRE Recuerda, debes unir Negativo de la Fuente externa con el Negativo (GND) de la Placa Arduino.

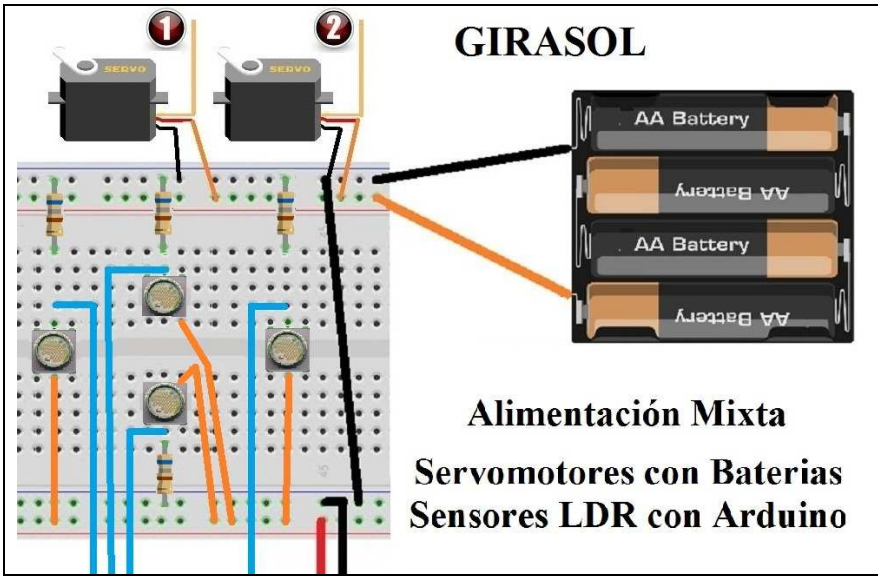
Montaje del Dispositivo: La Base puedes Fabricarla con Madera Balsa o Cartón, también puedes **fijar el Servo desde abajo a una linda tapa de un frasco o a un CD / DVD,** usando tornillos con la agarradera del Servo, y ya tendrás una plataforma redonda capaz de girar. Arriba fijas la estructura que usaste para la Parte A de este proyecto.

En las bocetos del girasol o buscador de luz ya terminado, se muestra una panel solar, es el que colectaría la mayor cantidad de rayos solares, para generar electricidad, por ejemplo cargar una batería..

Podríamos continuar, y aprender mucho más, sin embargo, por ahora, acá lo terminamos, más adelante, con más conocimiento, podremos retomarlo y trabajar realmente con el panel solar y la energía generada.



Otra Forma de acomodar las Foto Resistencias, seria la que a continuación les dejo, hay que mencionar que requiere una programación ligeramente distinta (Eso queda para ustedes), ya que la luz llegará a ellas en forma distinta. Cual de los dos formatos te parece mejor?



RECORDAR

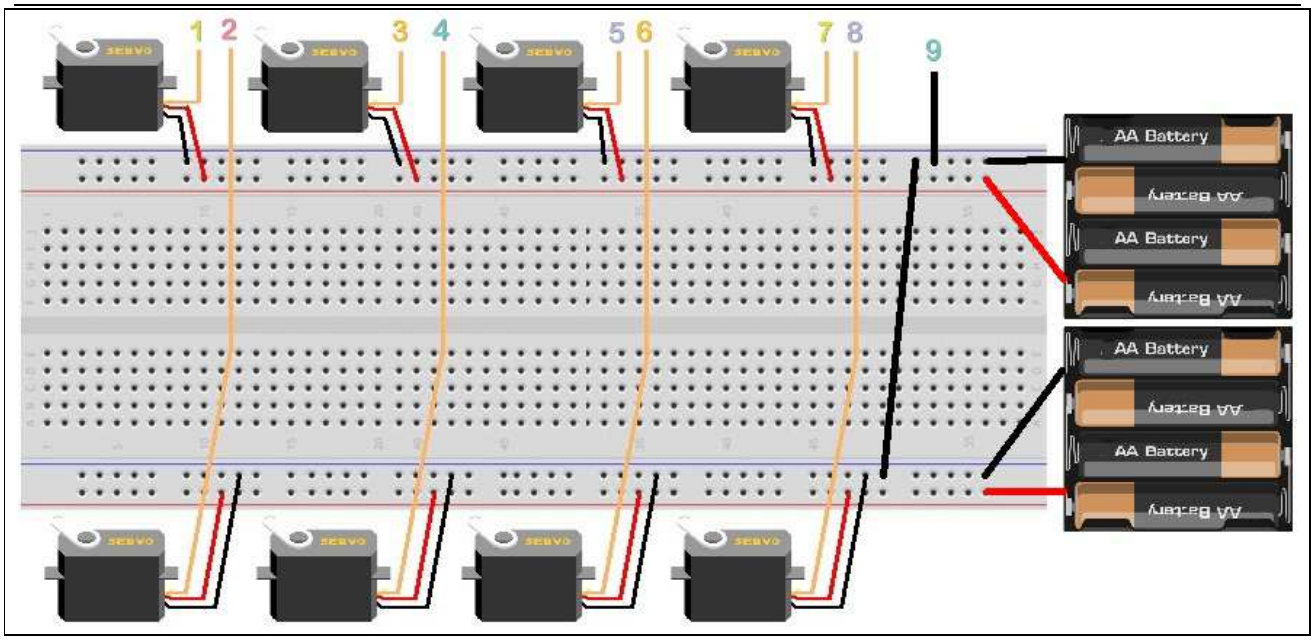
Conexión de Una Fotoresistencia LDR

IMPORTANTE

La cantidad de dispositivos que podremos conectar y controlar desde nuestra placa Arduino, solo esta limitada a la cantidad de pines que dispone, aunque hay circuitos que nos permitirán sobrepasar ampliamente este límite, tal es el caso del **MÓDULO CONTROLADOR DE SERVOS PCA9685,** pero esto lo veremos un poquito más adelante.

A continuación les dejo el circuito para controlar 8 Servos desde Arduino, pero esto pueden ampliarlo a 16, simplemente duplicando la cantidad de servos. También pueden cambiar la Fuente de alimentación externa, en vez de **dos porta Pilas**, puedes usar una Fuente de 5V (idealmente de 5A).

Presta Atención a la conexión de los cables. Los cables numerados del 1 al 8, deberás conectar cada uno al pin de control de Arduino, desde el Pin 2 en adelante, el cable 9 (color Negro) debes conectarlo al GND de Arduino, es la masa o GND común del Circuito. Por otro lado, tal como se muestra, el Porta Pilas de arriba, alimenta a la fila superior de Servos y el otro a la Fila inferior se Servos.



Si quieres equilibrar las Cargas, puedes conectar los dos Positivos (+) de los Porta Pilas (Solo positivos de Porta Pilas). De estar forma todas las Pilas se gastarán simultáneamente y equilibradamente (esta conexión sugerida no se muestra en el diagrama).

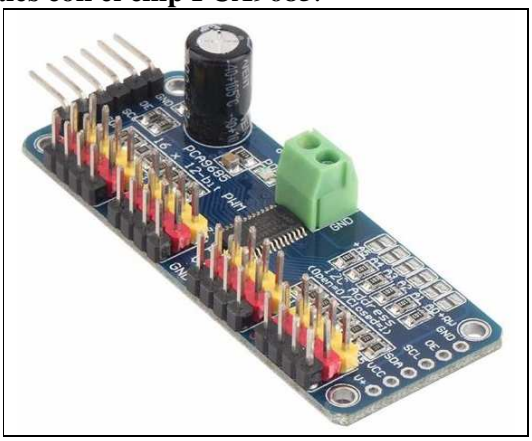
MÓDULO CONTROLADOR DE SERVOS PCA9685

Usaremos Librería Adafruit-PWM-Servo-Driver-Library-master

Originalmente, el controlador PCA9685, fue diseñado para controlar leds por PWM, pero también nos permite controlar servos, esto le ha dado gran difusión entre los entusiastas de la robótica.

Cuando desees realizar algún proyecto, seguramente en más de una ocasión necesitarás controlar varios servos, por ejemplo para un brazo, o el robot hexápodo, en fin cualquier proyecto en el que uses más de un servo. En realidad cualquier proyecto que requiera de una gran cantidad de señales PWM, encontraras una solución en este módulo **controlador de servos de 16 canales con el chip PCA9685**.

Adicionalmente, la placa fue diseñada para que al momento de controlar servos, encuentres los pines en el orden correcto para simplemente conectar los servomotores, además una bornera para la alimentación de los servos.



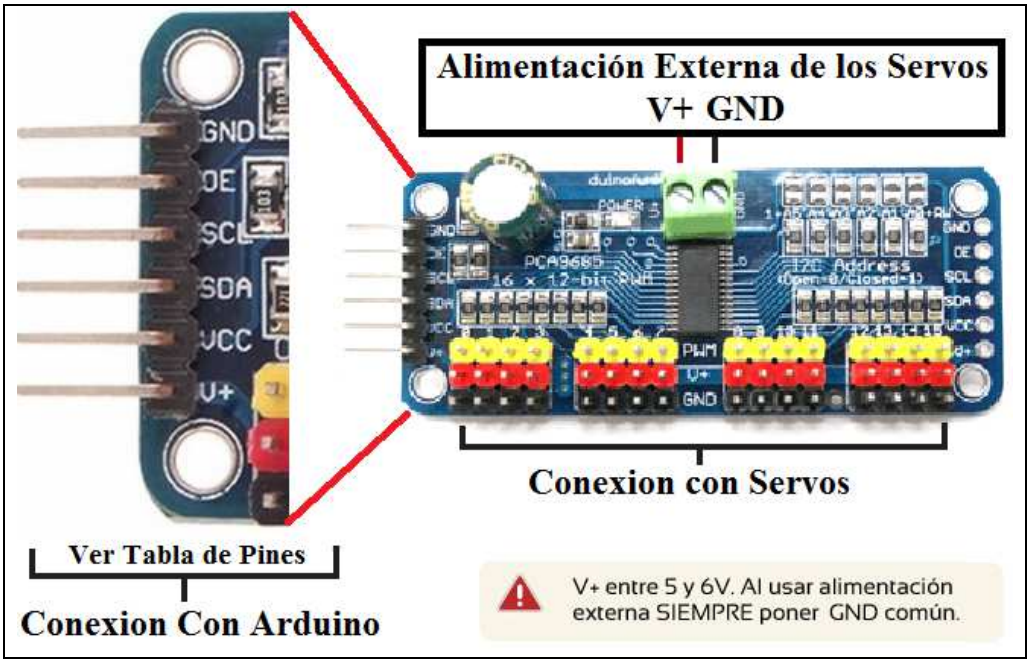
Características del Módulo Controlador (PCA9685)

- Controlador de PWM PCA9685 12 bits de resolución.
- El controlador es completamente independiente y no requiere señales de reloj externas.
- Control de 16 señales (canales) PWM a través de 2 hilos I2C.
- Salidas configurables “push-pull” o “drenador abierto”.
- Habilitación / Deshabilitación de salidas para anular rápidamente la señal en todas las salidas.
- Bloque de terminales para la alimentación de los servos.
- Resistencias de 220 Ohms en serie con cada salida PWM para protección y manejo directo de leds.
- Permite colocar varias tarjetas en el mismo bus para un máximo de 992 salidas PWM independientes.
- 6 Jumpers “soldables” para configurar la dirección de I2C de cada Tarjeta Controladora.
- Frecuencia 40-1000Hz.

- Voltaje DC 3-5 V
- Consumo 20mA (máximo)
- V max + 6 v (Alimentación externa)
- Tamaño del integrado 60mm x 25mm
- Peso completo (con 3x4 Pines, cabecera terminales): 10 gramos (Aproximados)

Conexión del Modulo PCA9685 con Arduino

Para conectar este modulo con Arduino, debemos ver la primer columna, luego buscar en el PinOut de nuestra placa, donde esta cada uno de estos pines requerido.



Para Facilitar un poco, acá les dejo una tabla con los pines que debemos usar en algunos modelos de Arduino. Recuerde que en muchos modelos Arduino, una funcionalidad determinada se encuentra en más de un PIN específico. Para encontrarlos, ver PinOut Completo de su modelo de Arduino en Nuestra Pagina.

Tabla de Conexiones			
Módulo PCA9685	**Pin Según Modelo Arduino**		
	Uno, Nano, Mini.	**Mega , DUE**	**Leonardo**
GND	GND	GND	GND
OE	GND/ No conectado	GND/ No conectado	GND/ No conectado
SCL	A5	21	3
SDA	A4	20	2
VCC	5V	5V	5V
+V	No conectado	No conectado	No conectado

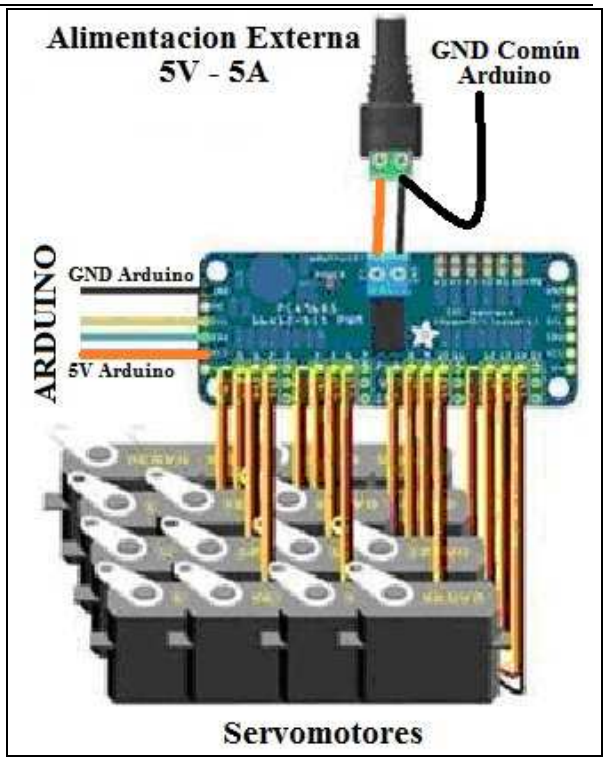
Si necesita manejar más de 16 servos, puede conectar varias Placas controladoras (máximo 64). Para esto consulte "**Apéndice B – Direccionamiento de Placa Controladora PCA9685**", que encontrara al final de la presente guía. Allí dispone de una extensa explicación y ejemplo de uso.

La mayoría de servos trabajan con voltajes de 4.5 y 6V. Acorde a los Servos que hemos usado en guías anteriores, Recomiendo usar una fuente externa de 5V / 5A. Aunque para realizar algunas pruebas preliminares, puede usar como alimentación externa 4 pilas comunes AA (1,5V). Igual que hemos venido usando hasta el momento.

La corriente mínima de la fuente externa, depende del tipo **de servomotores que se use** y de **la cantidad de servos que estarán conectados**, Si bien la corriente no es un dato constante en el servomotor es mejor sobredimensionar la fuente para que trabaje correctamente.

Si la fuente no es muy estable o genera ruido, es necesario soldar un condensador en el espacio de la placa del Módulo con un valor de 1000uF o el equivalente a 100uF por cada servomotor.

Para la conexión de los servos, disponemos de 16 columnas de 3 pines.



Siempre que Use Alimentación Externa, recuerde Unir todos los Negativos (GND Común)

Usaremos la Librería "**Adafruit-PWM-Servo-Driver-Library-master**" la cual podrá bajar desde la pagina, en la pestaña "**Librerías Adicionales**". Una vez instalada, encontrara que esta librería proporciona ejemplos de código, que resulta aconsejable revisar.

6) Mover 4 SevoMotores de los 16 posible que nos permite mover la Placa Controladora PCA9685.

Solo a modo de ejemplo, se toman 4 servos y se les da diferentes posiciones. Este proyecto esta pensado para tener alimentación de una Fuente de 5V 5A. Sin embargo para probar, puede sustituirse la fuente por 4 pilas comunes AA (1,5V c/u).

(Programa "009_Controla_Servos_PCA9685_Ej_03")

```
1   #include <Wire.h>
2   #include <Adafruit_PWMServoDriver.h>
-
3   Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver( 0x40 );
4   unsigned int pos0 = 172;     // ancho de pulso en cuentas para posición 0°
5   unsigned int pos180 = 565; // ancho de pulso en cuentas para la posición 180°
-
6   void setup( ) {
7     servos.begin( );
8     servos.setPWMFreq(60);  //Frecuencia PWM de 60Hz  o T=16,66ms
9   }
10  void setServo(uint8_t n_servo, int angulo) {
11    int duty;
12    duty=map(angulo, 0, 180, pos0, pos180);
13    servos.setPWM( n_servo, 0, duty);
14  }
15  void loop( ) {
16      setServo(0,10);
```

```
    delay(300);
    setServo(4,64);
    delay(300);
    setServo(8,117);
    delay(300);
    setServo(12,170);
    delay(300);
-
    setServo(0,64);
    delay(300);
    setServo(4,117);
    delay(300);
    setServo(8,170);
    delay(300);
    setServo(12,10);
    delay(300);
-
    setServo(0,117);
    delay(300);
    setServo(4,170);
    delay(300);
    setServo(8,10);
    delay(300);
    setServo(12,64);
    delay(300);
-
    setServo(0,170);
    delay(300);
    setServo(4,10);
    delay(300);
    setServo(8,64);
    delay(300);
    setServo(12,117);
    delay(300);
}
```

Los "delay( )" que usamos en el programa son simplemente para que todo funcione más despacio y podamos apreciar cada movimiento en particular. Aunque realmente no son necesarios en un dispositivo.

Recuerda que puedes manejar 16 servos con cada controladora que uses, y puedes usar varias uniéndolas en serie.

La fuente recomendada para usar esta controladora es una de 5V 5A, aunque en este ejemplo, con solo 4 servos puedas usar una de 5V 2,5A

1 y 2 Inclusión de Librerías

3

En esta línea declaramos o instanciamos el objeto, que nos permitirá manejar todos los servos que conectemos a la Placa Controladora. Pero me antes de continuar, quiero explicar un poco más de esta línea, a la que por claridad la dividiremos en 5 partes:

A	B	C	D	E
Adafruit_PWMServoDriver	**servos**	=	Adafruit_PWMServoDriver(0x40)	**;**

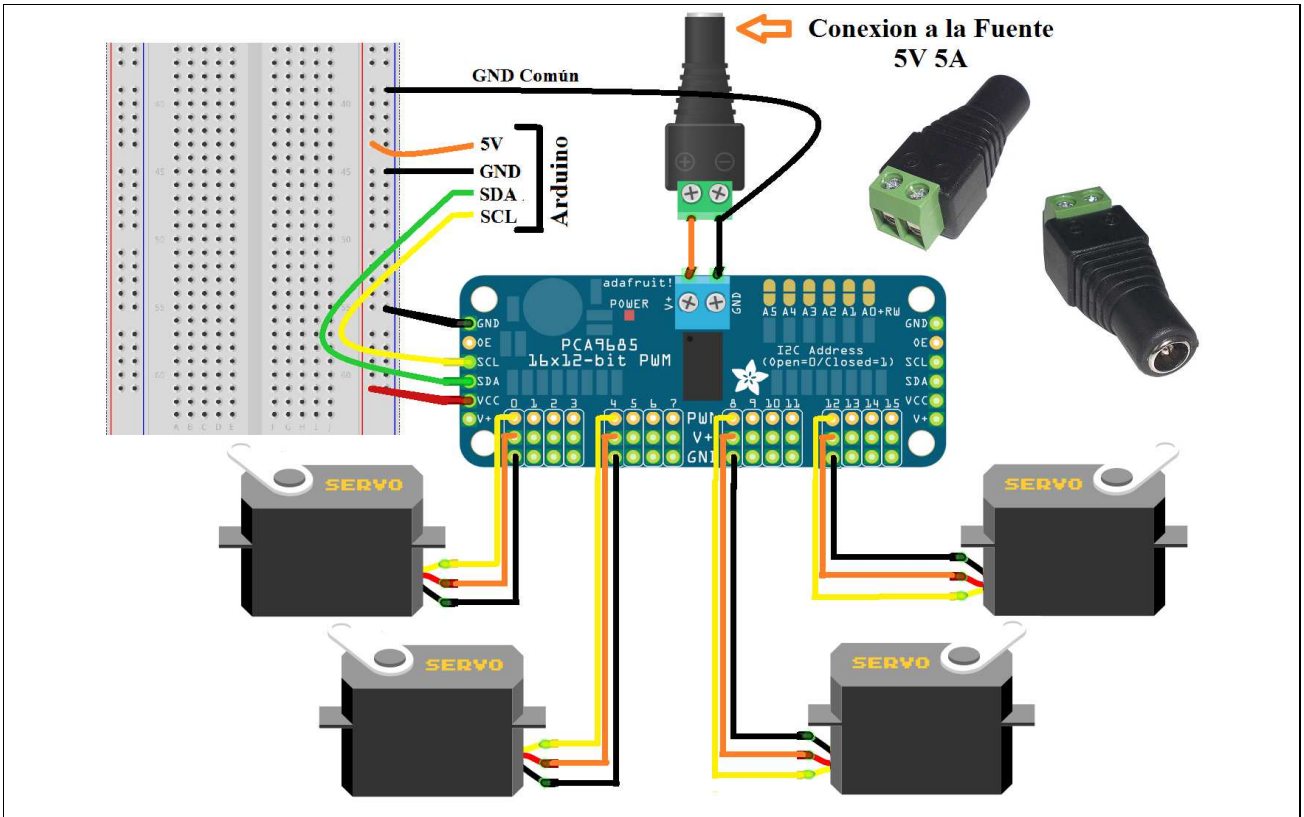
Para entender mejor, repito que, estamos declarando una variable (Instanciando un objeto), y ahora explico cada parte.

- A- Este es el tipo de dato (clase) provisto por la librería incluida en la línea 2 del programa
- B- Acá declaramos la Variable, a la que llamo "**servos**" (Porque me gusta el nombre)
- C- Signo igual, asignara la información de la placa y que dirección tiene dentro de nuestro dispositivo. Para más información ver "**Apéndice B – Direccionamiento de Placa Controladora PCA9685".**
- D- Llamo al constructor de la Clase, al que le doy la dirección de la Placa que deberá manejar (0x40), y que a su vez se la pasara a la Variable (Objeto instanciado) llamado "**servos**". Recordar que podemos poner Varias Placas controladoras en serie (una a continuación de la otra), y cada una deberá tener una dirección distinta a la anterior.

	Para más información ver "**Apéndice B – Direccionamiento de Placa Controladora PCA9685".** E- Y para finalizar la línea el punto y coma.
4 y 5	Establezco el ancho de pulso estándar para la posición 0° (cero) y posición 180° (ciento ochenta) de un servo motor como el que usamos en esta guía.
6	Cabecera de la función "**setup( )**"
7	Inicialización de la variable (objeto) "**servos**" mediante el comando **servos**.begin( );
8	Establecemos la frecuencia con la que trabajara la Placa controladora. En este caso la Frecuencia PWM será de 60Hz o T=16,66ms
9	Finaliza la función "**setup( )**"
10	Declaración de una función definida por nosotros, para clarificar el trabajo. "**void setServo(uint8_t n_servo, int angulo)**" Esta función, no retornara nada, y le entregaremos como dato, el numero de servo que debe mover, y el ángulo al que debe desplazarse (partiendo desde el ángulo cero)
11	Declaración de la la variable "duty" que usaremos para indicar cuanto deberá desplazarse el servo motor desde la posición inicial.
12	Transformo el valor expresado en Grado a valor que el servo reconozca.
13	Envió comando al servo para que gire la cantidad de grados establecida.
14	Finaliza nuestra función
15	A partir de esta línea, ya estamos en la función "**loop( )**" desde donde daremos las órdenes y manejaremos los Servo Motores.

Si bien en nuestro programa usamos solo 4 Servos, bastaría con utilizar la función que definimos "**setServo(NroServo, Angulo)**" para controlar todos; donde "NroServo" es el número de servo que se debe controlar (El cero, corresponde al primer servo), y "Angulo" es el ángulo al que debe desplazarse el servo.

CIRCUITO PARA NUESTRO PROYECTO



No desconecte el circuito, que lo usaremos nuevamente a continuación

Lista de Materiales: Como alimentación podremos usar 1 Fuente (de 5V-5A) - 1 Conector Plug Hembra Alimentación de 5.5 x 2.1 con Bornera **o** 1 Porta Pilas para 4 pilas comunes AA (1,5V). Para el resto del circuito usaremos 4 ServoMotores - Cables Macho/Macho Y Macho/Hembra – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

Si necesita manejar más de 16 servos, puede conectar varias Placas controladoras (máximo 64). Para esto consulte "**Apéndice B – Direccionamiento de Placa Controladora PCA9685**", que encontrara al final de la presente guía. Allí dispone de una extensa explicación y ejemplo de uso.

7) Mover 4 SevoMotores (distintos) de los 16 posible que nos permite mover la Placa Controladora PCA9685.

En el caso que usen diferentes tipos de servomotores con diferente rango del ancho de pulso; incluso siendo del mismo fabricante en sus modelos de servos los valores mínimo y máximo pueden variar, por ejemplo para algunos casos el ancho de pulso para la posición 0° puede ser 0.7ms, 1ms u otro valor, de igual forma para la posición 180°.

Para este caso se necesita tener en cuenta que si bien la señal PWM puede ser el mismo para todos los servos debemos de tener como referencia diferentes valores del rango del ancho de pulso. Esto lo hacemos haciendo las siguientes modificaciones en el código anterior.
Al declarar las posición 0 y 180, lo debemos hacer con valores independientes para los 16 servos, en nuestro caso, para los cuatro servos que usamos:

(Programa "009_Controla_Servos_PCA9685_Ej_04")

```
1   #include <Wire.h>
2   #include <Adafruit_PWMServoDriver.h>
-
3   Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver( 0x40 );
-
4   // ancho de pulso para posición 0°
5   unsigned int pos0[16]    = { 172, 160, 170, 165, 172, 160, 170, 165,
6                                172, 160, 170, 165, 172, 160, 170, 165};
7   // ancho de pulso para la posición 180°
8   unsigned int pos180[16] =  { 565, 560, 565, 570, 565, 560, 565, 570,
9                                565, 560, 565, 570, 565, 560, 565, 570};
-
10  void setup( ) {
11    servos.begin( );
12    servos.setPWMFreq(60);  //Frecuencia PWM de 60Hz  o T=16,66ms
13  }
-
14  void setServo(uint8_t n_servo, int angulo) {
15    int duty;
16    duty=map( angulo, 0, 180, pos0[n_servo], pos180[n_servo]);
17    servos.setPWM( n_servo, 0, duty);
18  }
-
19  void loop( ) {
20     setServo(0,10);
21     delay(300);
22     setServo(4,64);
23     delay(300);
24     setServo(8,117);
25     delay(300);
```

Los "delay( )" que usamos en el programa son simplemente para que todo funcione más despacio y podamos apreciar cada movimiento en particular. Aunque realmente no son necesarios en un dispositivo.

```
26      setServo(12,170);
27      delay(300);
28      setServo(0,170);
29      delay(300);
30      setServo(4,10);
31      delay(300);
32      setServo(8,64);
33      delay(300);
34      setServo(12,117);
35      delay(300);
36  }
```

Recuerda que puedes manejar 16 servos con cada controladora que uses, y puedes usar varias uniéndolas en serie.

La fuente recomendada para usar esta controladora es una de 5V 5A, aunque en este ejemplo, con solo 4 servos puedas usar una de 5V 2,5A

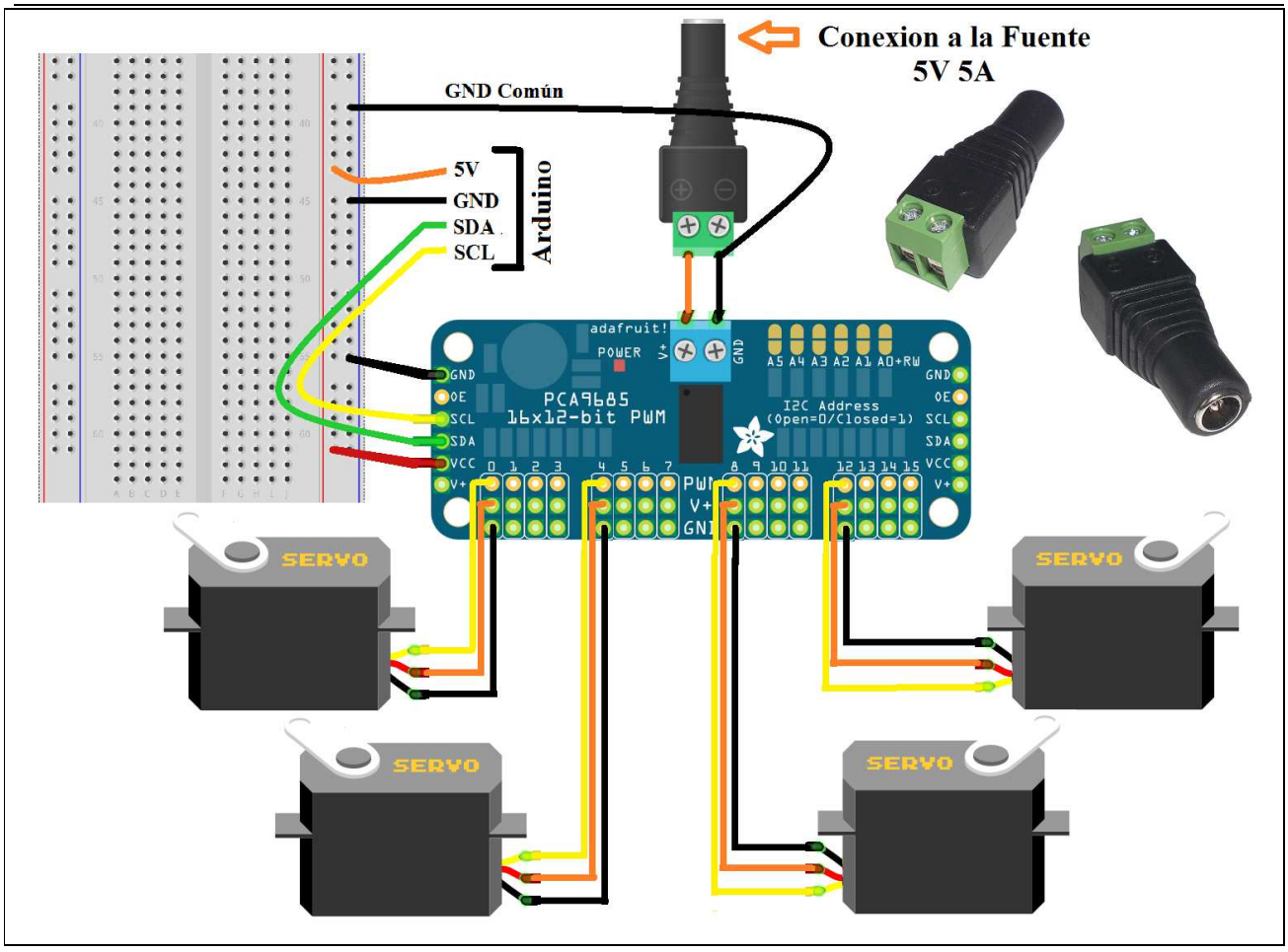
1 y 2	Inclusión de Librerías
3	En esta línea declaramos o instanciamos un objeto, que nos permitirá manejar todos los servos que conectemos a la Placa Controladora.
4 al 9	En el caso que usen diferentes tipos de servomotores con diferente rango del ancho de pulso; incluso siendo del mismo fabricante en sus modelos de servos los valores mínimo y máximo pueden variar, por ejemplo para algunos casos el ancho de pulso para la posición 0° puede ser 0.7ms, 1ms u otro valor, de igual forma para la posición 180°. Para este caso se necesita tener en cuenta diferentes valores del rango del ancho de pulso. Esto lo hacemos por medio de un vector, en el que almacenamos valores independientes para los 16 servos
10	Cabecera de la función "**setup( )**"
11	Inicialización de la variable (objeto) "**servos**" mediante el comando **servos**.begin( );
12	Establecemos la frecuencia con la que trabajara la Placa controladora. En este caso la Frecuencia PWM será de 60Hz o T=16,66ms
13	Finaliza la función "**setup( )**"
14	Declaración de una función definida por nosotros, para clarificar el trabajo. "**void setServo(uint8_t n_servo, int angulo)**" Esta función, no retornara nada, y le entregaremos como dato, el numero de servo que debe mover, y el ángulo al que debe desplazarse (partiendo desde el ángulo cero)
15	Declaración de la la variable "duty" que usaremos para indicar cuanto deberá desplazarse el servo motor desde la posición inicial.
16	Transformo el valor expresado en Grado a valor que el servo reconozca.
17	Envió comando al servo para que gire la cantidad de grados establecida.
18	Finaliza nuestra función
19	A partir de esta línea, ya estamos en la función "**loop( )**" desde donde daremos las órdenes y manejaremos los Servo Motores.

Solo a modo de ejemplo, se toman 4 servos y se les da diferentes posiciones. Este proyecto esta pensado para tener alimentación de una Fuente de 5V 5A. Sin embargo para probar, puede sustituirse la fuente por 4 pilas comunes AA (1,5V c/u).

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: Como alimentación podremos usar 1 Fuente (de 5V-5A) - 1 Conector Plug Hembra Alimentación de 5.5 x 2.1 con Bornera **o -** 1 Porta Pilas para 4 pilas comunes AA (1,5V). Para el resto del circuito usaremos 4 ServoMotores - Cables Macho/Macho Y Macho/Hembra – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

Si necesita manejar más de 16 servos, puede conectar varias Placas controladoras (máximo 64). Para esto consulte "**Apéndice B – Direccionamiento de Placa Controladora PCA9685**", que encontrara al final de la presente guía. Allí dispone de una extensa explicación y ejemplo de uso.



8) Hacer un Barrido de los 16 Servos que permite mover la Placa Controladora PCA9685.

Mover los 16 servos de izquierda a derecha y de derecha a izquierda. Repitiendo el ciclo permanentemente.

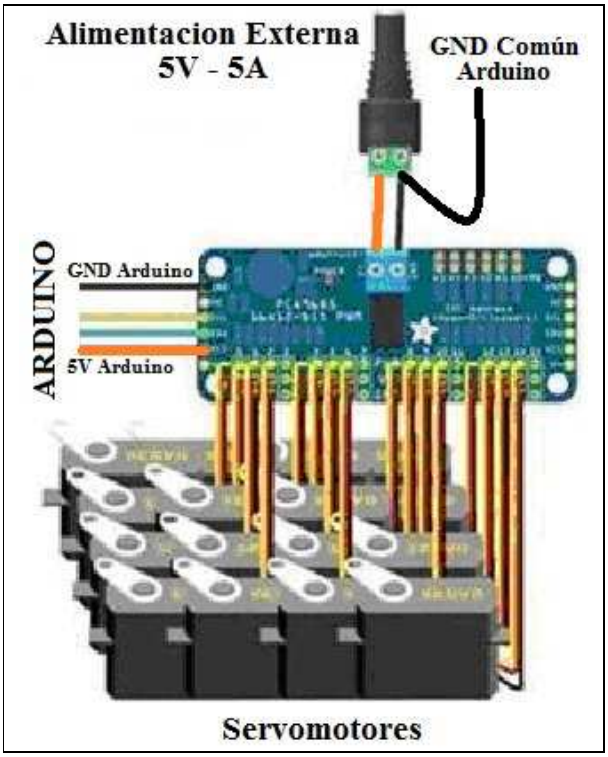
(Programa "009_Controla_Servos_PCA9685_Ej_02A")

```
1   #include <Wire.h>
2   #include <Adafruit_PWMServoDriver.h>
-
3   Adafruit_PWMServoDriver Controlador = Adafruit_PWMServoDriver(0x40);
-
4   const uint16_t frecuencia = 60,    // Frecuencia PWM de 50Hz o T=20ms
5                  pos0 = 172,         // ancho de pulso para posición 0°
6                  pos180 =  565;      // ancho de pulso para la posición 180°
7   uint8_t        NroServo;
8   uint16_t       duty;
9   int            angulo;
-
10  void setup( ){
11    Controlador.begin( );
12    Controlador.setPWMFreq( frecuencia );
13  }
-
14  void loop( ){
15    for (angulo = 0; angulo < 180; angulo++){
```

```
16      for (NroServo = 0; NroServo < 16; NroServo++){
17        duty=map(angulo,0,180,pos0, pos180);
18        Controlador.setPWM(NroServo, 0, duty);
19      }
20    }
21    delay(1000);
-
22    for (angulo = 180; angulo > 0; angulo--){
23      for (NroServo = 0; NroServo < 16; NroServo++){
24        duty=map(angulo,0,180,pos0, pos180);
25        Controlador.setPWM(NroServo, 0, duty);
26      }
27    }
28    delay(1000);
29  }
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: Como alimentación podremos usar 1 Fuente (de 5V-5A) - 1 Conector Plug Hembra Alimentación de 5.5 x 2.1 con Bornera - 16 ServoMotores - Cables Macho/Macho Y Macho/Hembra – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.



Para la conexión de cables deberá recurrir a la tabla de conexiones, de donde encontrará la que corresponda al modelo de su placa Arduino.

Solo preste atención a la secuencia, que esta perfectamente visible en la imagen. Y siempre recuerde conectar todos los negativos (GND).

En cuanto a la alimentación, si conecta los 18 servos, necesitara una fuente de 5V 5ª. Pero si solo realiza la prueba con 4 servos (uno en cada grupo de conectores) con Porta Pilas para 4 pilas comunes AA (1,5V), será más que suficiente

Si necesita manejar más de 16 servos, puede conectar varias Placas controladoras (máximo 64). Para esto consulte "**Apéndice B – Direccionamiento de Placa Controladora PCA9685**", que encontrara al final de la presente guía. Allí dispone de una extensa explicación y ejemplo de uso.

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

Suponiendo Que hemos declarado una variable (Instanciamos un Objeto) de la Clase "Adafruit_PWMServoDriver" (según vemos en el siguiente código):

```
Adafruit_PWMServoDriver Controla = Adafruit_PWMServoDriver( 0x40 );
```

En esta línea declaramos o instanciamos un objeto, que nos permitirá manejar todos los servos que conectemos a la Placa Controladora. Pero me antes de continuar, quiero explicar un poco más de esta línea, a la que por claridad la dividiremos en 5 partes:

A	B	C	D	E
Adafruit_PWMServoDriver	**Controla**	=	Adafruit_PWMServoDriver(0x40)	**;**

Para entender mejor, estamos declarando una variable (Instanciando un objeto) del tipo "**Adafruit_PWMServoDriver**" y que se llama "**Controla**". Ahora explico cada parte.

A- Este es el tipo de dato (clase) provisto por la librería, que usaremos para declarar la variable.
B- Declaramos la Variable, a la que llamo "**Controla**" (Porque me gusta el nombre)
C- Signo igual, asignara la información de la placa y que dirección tiene dentro de nuestro dispositivo. Para más información ver "**Apéndice B – Direccionamiento de Placa Controladora PCA9685".**
D- Llamo al constructor de la Clase, al que le doy la dirección de la Placa que deberá manejar (0x40), y que a su vez se la pasara a la Variable (Objeto instanciado) llamado "**Controla**". Recordar que podemos poner Varias Placas controladoras en serie (una a continuación de la otra), y cada una deberá tener una dirección distinta a la anterior. Para más información ver "**Apéndice B – Direccionamiento de Placa Controladora PCA9685".**
E- Y para finalizar la línea, el punto y coma.

Entonces, una vez declarada la Variable (instanciado el objeto) dispondremos de las siguientes funciones (métodos):

<table>
<tr><th colspan="2">FUNCIONES (Métodos) DE LA CLASE "Adafruit_PWMServoDriver"</th></tr>
<tr><td colspan="2"><u>Constructor de la Clase:</u> Adafruit_PWMServoDriver(uint8_t addr = 0x40);

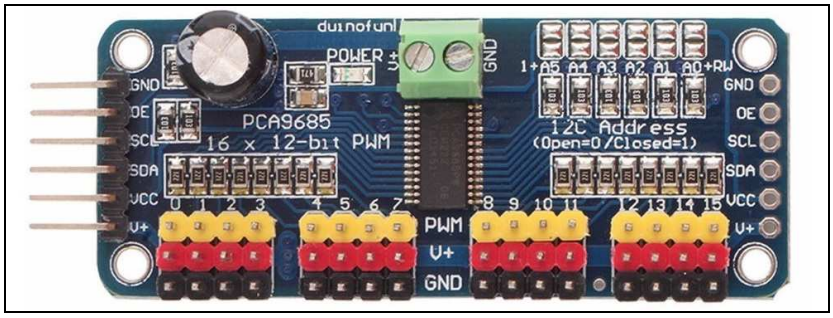
Al declarar una variable (instanciar un objeto de la clase), el constructor se ejecuta automáticamente. Si no se le pasara la direccion como parámetro, el metodo (funcion Constructor) asume que la dirección de la Placa que debe manejar, es por defecto "0x40". Ejemplos de uso:
Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver(0x40);
Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver( );
Adafruit_PWMServoDriver servos;
Las tres líneas harán exactamente lo mismo.
Sin embargo si se quisiera establecer una dirección distinta, deberá usarse la primera línea, por ejemplo (Uniendo el Puente A0). Para más información ver "Apéndice B – Direccionamiento de Placa Controladora PCA9685".

Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver(0x41);</td></tr>
<tr><td>begin( )</td><td>Definición de la función: void begin(void)
Inicializa el objeto. Ejemplo:
Controla.begin( );</td></tr>
<tr><td>reset( )</td><td>Definición de la función: void reset(void)
Reinicia la clase. Ejemplo:
Controla.reset( );</td></tr>
<tr><td>setPWMFreq( );</td><td>Definición de la función: void setPWMFreq(float freq)
Establece la frecuencia de trabajo de la Placa Controladora. Ejemplo:</td></tr>
</table>

	Controla.setPWMFreq(60); Establece Frecuencia PWM de 60Hz o T=16,66ms para esta placa controladora
setPWM( );	Definición de la función: void setPWM(uint8_t num, uint16_t on, uint16_t off) Envía un pulso PWM de un Valor determinado. Suponiendo que deberá mover un Servo, entonces: "**num**" identifica el se servo (entre cero y 15), partiendo de la posición "**on**" se desplace "**off**". Ejemplo: **Controla.setPWM(3, 0, 80);**
setPin( );	Definición de la función: void setPin(uint8_t num, uint16_t val, bool invert=false); Ejemplo: **Controla.**setPin(3, 50, True) **Controla.**setPin(3, 50) Ambos comando harán exactamente lo mismo

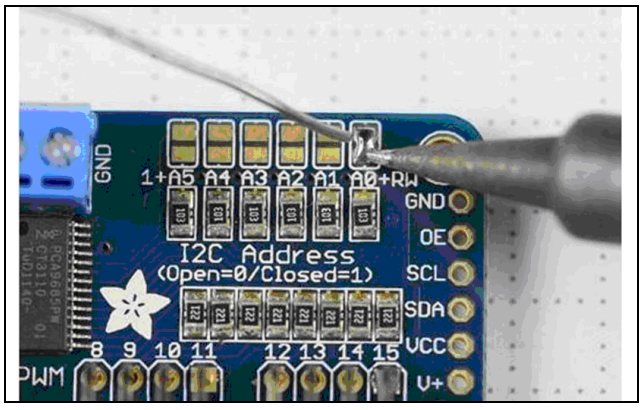
Siempre es resulta interesante analizar los ejemplos que trae cada librería. Aprenderá cosas muy útiles.

Apéndice B – Direccionamiento de Placa Controladora PCA9685.



Usaremos la Librería "**Adafruit_PWMServoDriver.h**", disponible en el paquete de instalación "**Adafruit-PWM-Servo-Driver-Library-master**", que encontrara y podrá bajar desde nuestra página, en la pestaña "**Librerías Adicionales**". Una vez instalada, junto con la librería, se encuentran ejemplos de código, que resulta aconsejable y muy interesante revisar.

Cuando conectamos una placa controladora **PCA9685** a nuestro dispositivo, esta necesita tener una dirección para poder ser encontrada y controlada por nuestro programa.



La dirección I2C inicial que toma por defecto cada placa controladora es de "**0×40**", pero usted podrá modificar esta dirección conectando cualquiera de los 6 puentes denominados A0 hasta el A5, ubicados en la esquina superior derecha de la placa controladora.

Cada Punte representa un cero o un uno (expresado en binario) del numero de la dirección de nuestra controladora (Puente sin Conectar: un Cero – Puente conectado: un uno).

Cuando compramos la placa, todos los puentes deben estar sin conectar. Entonces para formar la dirección, ponemos un numero 1 (binario) y a continuación los puentes (Puente sin Conectar: un Cero – Puente conectado: un uno). El conjunto formaran la dirección completa de nuestra placa.

Posiciones Binarias								Valor Hexa
1	A5	A4	A3	A2	A1	A0		

Inicialmente, cuando compramos nuestra placa controladora, todos los puentes están abiertos (Valor cero), por lo tanto la dirección será:

1	0	0	0	0	0	0		0x40

Si colocamos una segunda placa (en serie), a esta segunda le unimos (soldamos) el puente **A0** (último de la derecha), entonces la dirección de esta segunda placa será:

1	0	0	0	0	0	1		0x41

Si colocamos una terca placa controladora (en serie), a esta tercera placa, le unimos (soldamos) el puente A1. Entonces ahora, la dirección de esta placa será:

1	0	0	0	0	1	0		0x42

Note que estamos generando números binarios, que traducidos a Hexadecimal (Valor que vemos en la última columna de la derecha), son los que usaremos al momento de dar una dirección a cada placa.

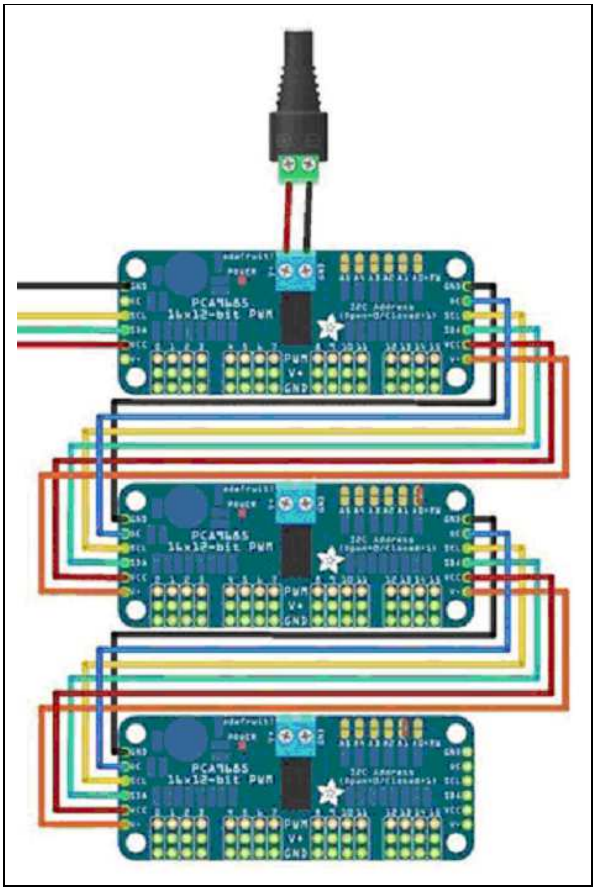
Y si nos adentramos un poco más en este tema, veremos que variando los 6 puentes (6 dígitos binarios = 2^6), podremos conectar 64 placas controladores en serie (cada placa con su dirección propia y única, y a su vez, cada una, nos permitirá controlar 16 canales PWM o lo que es igual, 16 dispositivos conectados a ella.

Esto nos arroja un impresionante total que calculamos 64 x 16 = 898 (que podemos extender a 992). Por ejemplo, podríamos controlar 898 Servo Motores.

Respecto a la fuente de alimentación, esta deberá ser calculada cuidadosamente en cada caso en particular, dependiendo de los dispositivos que se conecten.

Y finalmente el código que nos permitirá manejar cada placa con sus dispositivos.

Esto es muy simple. Declaramos una variable (objeto) para cada placa controladora con su dirección propia.



(Programa "009_Controla_Servos_PCA9685_Ej_05")

```
1   #include <Wire.h>
2   #include <Adafruit_PWMServoDriver.h>
-
3   Adafruit_PWMServoDriver Placa_01 = Adafruit_PWMServoDriver(0x40),
4                                      Placa_02 = Adafruit_PWMServoDriver(0x41),
5                                      Placa_03 = Adafruit_PWMServoDriver(0x42);
-
```

```
6   void setup( ) {
7     Serial. begin (9600);
8     while (!Serial) {
9       ; // Esperamos que el puerto serie este abierto.
10    }
11    Serial.println("Prueba de tres Controladoras con 16 canales PWM C/U");
-
12    Placa_01.begin( );
13    Placa_01.setPWMFreq(1600); // esta es la máxima frecuencia PWM
-
14    Placa_02.begin( );
15    Placa_02.setPWMFreq(1600); // esta es la máxima frecuencia PWM
-
16    Placa_03.begin( );
17    Placa_03.setPWMFreq(1600); // esta es la máxima frecuencia PWM
18  }
-
19  void loop( ) {
20      Placa_01.setPWM( n_servo, 0, duty);
21  // …… Otros Comandos
-
22      Placa_02.setPWM( n_servo, 0, duty);
23  // …… Otros Comandos
-
24      Placa_03.setPWM( n_servo, 0, duty);
25  // …… Otros Comandos
26  }
```

Para compilar y ejecutar este programa deben estar declaradas las variables:
"**uint8_t n_servo;**" que contiene el numero se servo que se quiere controlar y "**int duty;**" que indica cuanto debe rotar el servo desde la posición cero. Y Además tener un valor apropiado (eso ya es tu programa.! ☺).

01001100 01100001 00100000 01110000 01110010 11000011 10100001 01100011 01110100
01101001 01100011 01100001 00100000 01100101 01110011 00100000 01101100 01100001
00100000 01100010 01100001 01110011 01100101 00100000 01100100 01100101 01101100
00100000 01100001 01110000 01110010 01100101 01101110 01100100 01101001 01111010
01100001 01101010 01100101 00100000 00100001

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

El problema real no es si las maquinas piensan, sino si lo hacen los hombres.