Tutorial12. Control de servomotores con Arduino

Contenido

Material necesario

- Placa Arduino
- Cable micro USB
- Placa Protoboard
- Cables Jumper-Wire
- Componentes:

  - Resistencias
  - Pulsadores
  - Transistor BC547 o similar
  - Diodo 1N4001
  - Potenciómetro
  - Driver L293D
  - Motor DC-3V
  - Servomotor

Conceptos teóricos

- Conmutación del transistor
- Diodo de protección

Funciones propias de Arduino

- Librería Servo

IMPORTANTE:

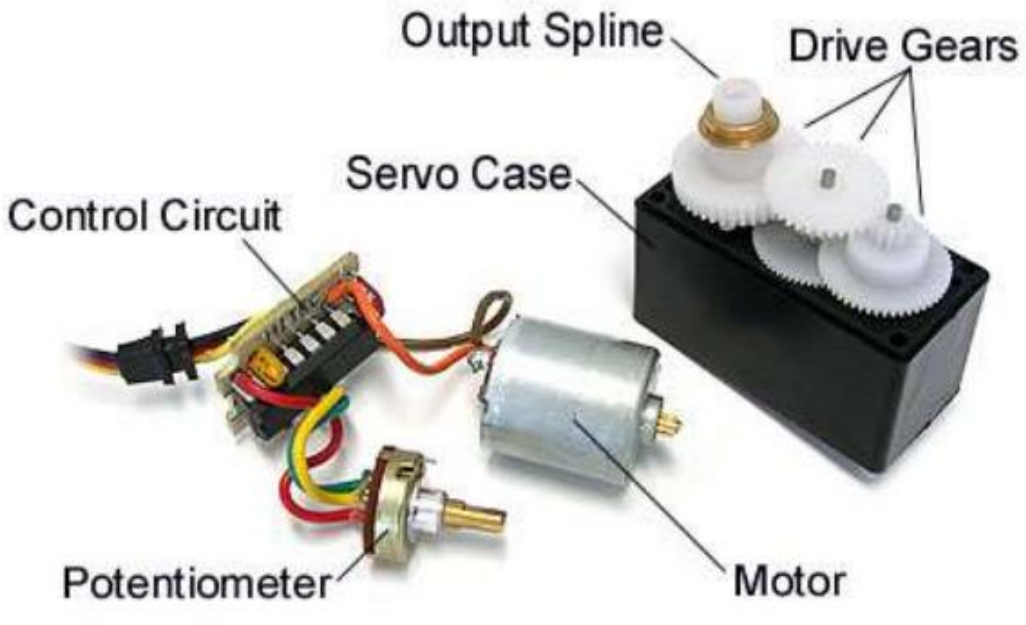
Comprobaremos que nuestra placa **Arduino** está **desconectada** y sin energía, puesto de no ser así podría dañarse tanto la placa, como el equipo. Una vez hemos realizado esta comprobación, pasaremos a realizar el montaje.

1. Servomotores

Qué es un servomotor

Un servomotor es un motor eléctrico que puede ser controlado tanto en velocidad como en posición. Consta de 4 piezas:

- Un motor DC
- Una reductora
- Un dispositivo para detectar su posición, normalmente un potenciómetro
- Un circuito de control electrónico



Actualmente existen dos tipos de servomotores:

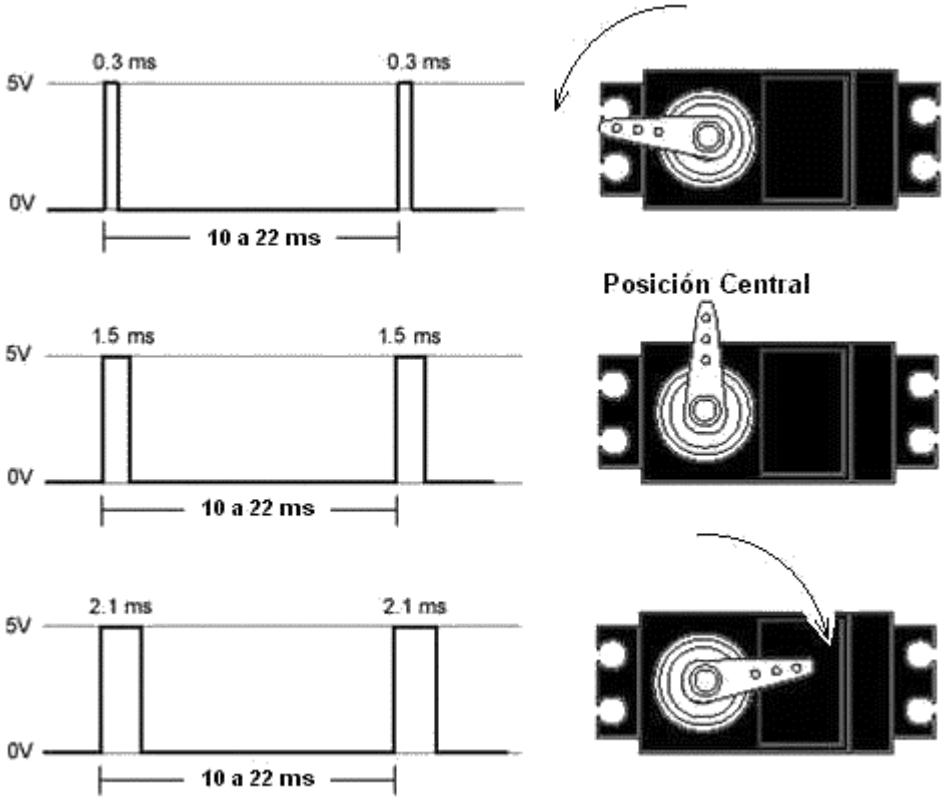
- De rotación continua, pueden girar 360º
- **Estándar, solo giran 180º**

Control de un servomotor

Los servomotores se controlan mediante pulsos de diferente duración que determinan la posición de su motor de corriente continua. La **posición** deseada se envía a través de la línea de control. La **velocidad** del motor es proporcional a la diferencia entre la posición actual y la deseada, es lo que se denomina CONTROL PROPORCIONAL.

El servo espera recibir un pulso cada 20ms (50HZ) de una duración determinada en base a la cual se moverá a una posición u otra y según el desplazamiento que deba hacer lo hará a mayor o menor velocidad.
Si el control del servomotor recibe una señal de duración entre 0.5 a 1.4 milisegundos, éste se moverá en sentido horario (CW). Entre 1.6 y 2 milisegundos moverá el servomotor en sentido antihorario (CCW). Por último, si el pulso dura 1.5 milisegundos pasará a un estado neutro para los servomotores estándares.

Mantienen la posición durante un tiempo, y no deben forzarse para sacarlo de esta posición. Sin embargo, la posición no la mantiene para siempre y habrá que refrescársela tras un tiempo. En esencia, cada 20ms hay que decirle al servo "colócate en X".



Este tipo de motor no es muy usado en las industrias ni en los trabajos mecánicos por tener baja potencia de trabajo.

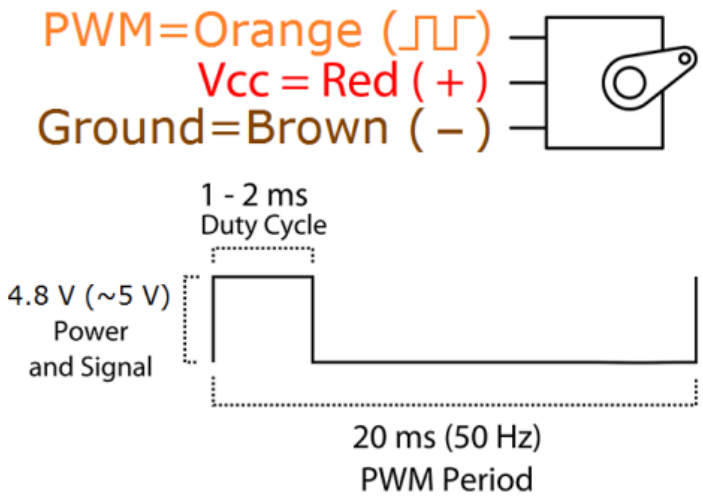
Proyecto12_01: Control de un servo motor de 180º con Arduino

Enunciado

En este proyecto conectaremos un servomotor estándar sin usar ningún otro componente. El servomotor tiene tres cables para Vcc (rojo), GND (negro o marrón) y control (naranja o amarillo). El cable de control lo conectaremos a la salida digital 9. Con el código propuesto controlaremos la posición del servomotor. Para ver el movimiento del servo más claramente, acopla al eje alguna pieza.

Material

1 Motor servo estándar (http://www.micropik.com/PDF/SG90Servo.pdf)



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.

Montaje

Son tres los cables de que dispone el servomotor, y debemos conectarlos de manera correcta:

- Cable **rojo:** se conectara a la tension de **5 V**.
- Cable **negro**: se conectara a tierra (**0 V**).
- Cable BLANCO o naranja: se conectara al pin de control (del 0 al 13).

Funciones Arduino

Arduino nos ofrece una librería para manejar servomotores que simplifican extraordinariamente esta tarea. Con las funciones incluidas en esta librería solo tendremos que preocuparnos de indicar la posición o la velocidad deseada según el tipo de servo con el que estemos trabajando.

http://arduino.cc/en/Reference/servo

En primer lugar, hay que indicar en cual pin se conectará el servo,

attach()

Description

Attach the Servo variable to a pin. Note that in Arduino 0016 and earlier, the Servo library supports only servos on only two pins: 9 and 10.

Syntax

servo.attach(pin)
servo.attach(pin, min, max)

Parameters

servo: a variable of type Servo

pin: the number of the pin that the servo is attached to

min (optional): the pulse width, in microseconds, corresponding to the minimum (0-degree) angle on the servo (defaults to 544)

max (optional): the pulse width, in microseconds, corresponding to the maximum (180-degree) angle on the servo (defaults to 2400)

Observa, que si usamos la primera opción de sintaxis, la librería considera por defecto un servomotor que requiere entre 544ms y 2400ms. Recuerde que nuestro servomotor requiere de otro rango, entre 1000ms y 2000ms, por tanto deberemos optar por la segunda sintaxis donde especificaremos esos valores máximos y minimos.

A continuación, la libreria Servo ofrece dos opciones para establecer la posición del servomotor:

write()

Description

Writes a value to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft (in degrees), moving the shaft to that orientation. On a continuous rotation servo, this will set the speed of the servo (with 0 being full-speed in one direction, 180 being full speed in the other, and a value near 90 being no movement).

Syntax

servo.write(angle)

Parameters

servo: a variable of type Servo

angle: the value to write to the servo, from 0 to 180

writeMicroseconds()

Description

Writes a value in microseconds (uS) to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft. On standard servos a parameter value of 1000 is fully counter-clockwise, 2000 is fully clockwise, and 1500 is in the middle.

Note that some manufactures do not follow this standard very closely so that servos often respond to values between 700 and 2300. Feel free to increase these endpoints until the servo no longer continues to increase its range. Note however that attempting to drive a servo past its endpoints (often indicated by a growling sound) is a high-current state, and should be avoided.

Continuous-rotation servos will respond to the writeMicrosecond function in an analogous manner to the write function.

Syntax

servo.writeMicroseconds(uS)

Parameters

servo: a variable of type Servo

uS: the value of the parameter in microseconds (*int*)

La segunda opción permite especificar el tiempo y la primera el ángulo. Es recomendable usar la segunda opción, suele ser más precisa.

Código

```
#include <Servo.h> // Servo's library

Servo miservo; //It declares servo object

int angulo=0;

void setup()
{
  miservo.attach(9,1000,2000); //Set the pin to attach the servo
}


void loop() {

 //Neutral position
  miservo.write(90);
  delay(3000);

  //Decrease the angle from 90
  for(angulo =90; angulo >0; angulo --)
    {
    miservo.write(angulo);
    delay(100);
    }
  delay(5000);
}
```

Ejercicios

1. Crea un nuevo código para el servomotor que repita la acción del proyecto 12_01, pero en esta ocasion usa la función writeMicroseconds.
2. Realiza un código que te permita mover un servomotor sin necesidad de usar la librería Servo.

   NOTA: Deberas generar la señal de periodo 20ms y establecer un pulso de ancho entre 1000ms y 2000ms. Crea una función cuyos argumentos sean el angulo deseado y el pin donde está conectado el servo .