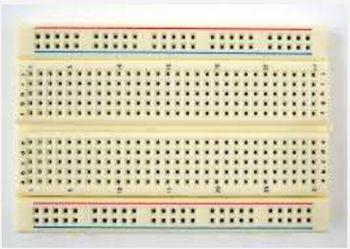
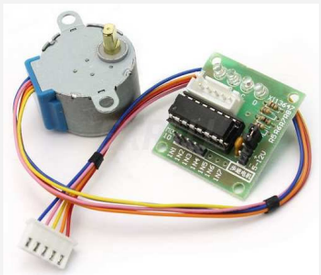
MOTORES PASO A PASO: 28BYJ-48

Primer contacto con un motor paso a paso - Prometec. *(Versión 10-4-18)*

OBJETIVOS

- Conocer mas de los **motores paso a paso**.
- Montar un pequeño motor **28BYJ-48** con su adaptador.
- Escribir un primer programa de control muy sencillo.

MATERIAL REQUERIDO.

	**Arduino UNO**
	**Una Protoboard**
	**4 x Cables** **Dupont Macho Hembra**
	**Un pequeño motor** 28BYJ-48 **y su breakboard**

VOLVIENDO CON LOS MOTORES PASO A PASO

En la sesión anterior dimos un pequeño repaso a la teoría que hay detrás de los motores paso a paso e intentamos sentar las cuatro ideas importantes que debéis conocer para elegir y usar con éxito, uno de esto motores.

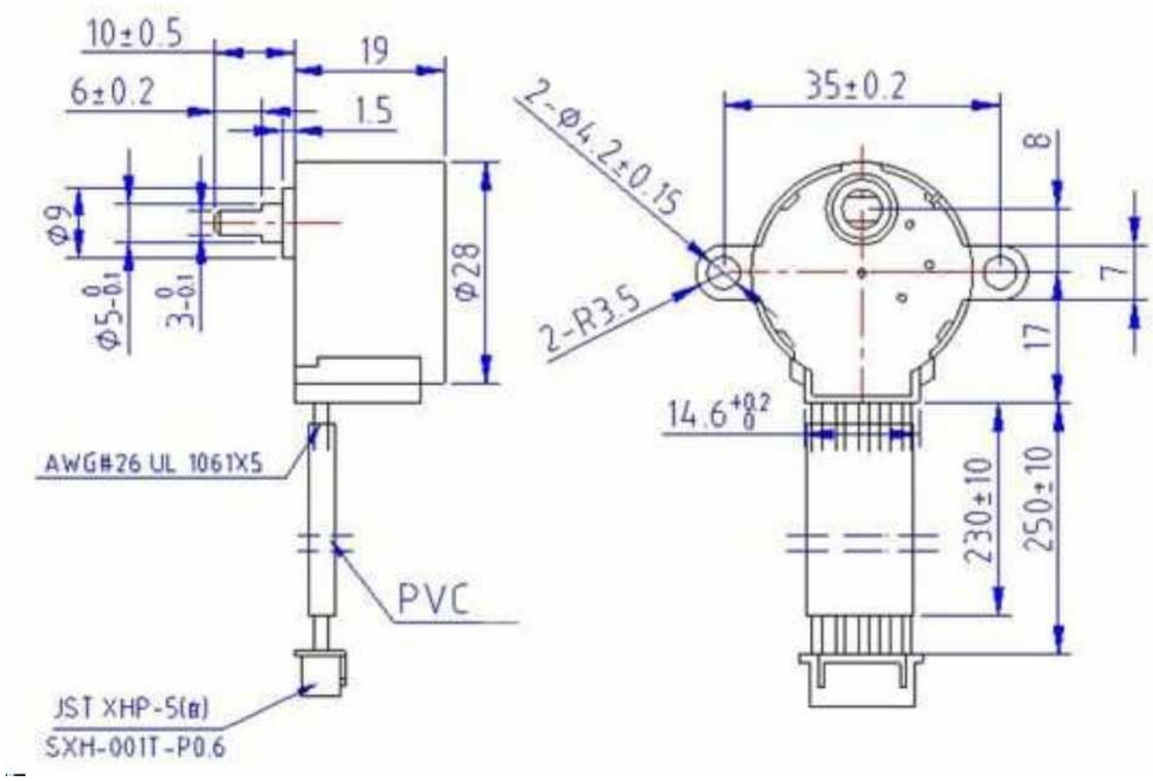
En esta sesión, vamos a usar un pequeño motor paso a paso unipolar, muy común en el mundo **Arduino** por su pequeño tamaño y bajo coste, el **28BYJ-48** y el adaptador que suele venir con él, basado en el chip **ULN2003A**.

Veremos en primer lugar las características que presenta y después montaremos un pequeño circuito básico, para mover el motor. Como siempre intentaremos que el ejemplo sea lo más simple posible para que en esta primera ocasión ver con claridad la forma de manejarlos.

Y poco más que decir en esta pequeña introducción. Vamos pues, a meternos directamente en harina.

EL MOTOR PASO A PASO 28BYJ-48

Este pequeño motor es muy habitual en los proyectos con pequeños robots y posicionadores caseros sencillos, porque aunque no es demasiado potente, ni rápido, tiene varias características más que simpáticas, además de ser muy barato. Y como no podía ser de otra manera, empezaremos por buscar su manual.



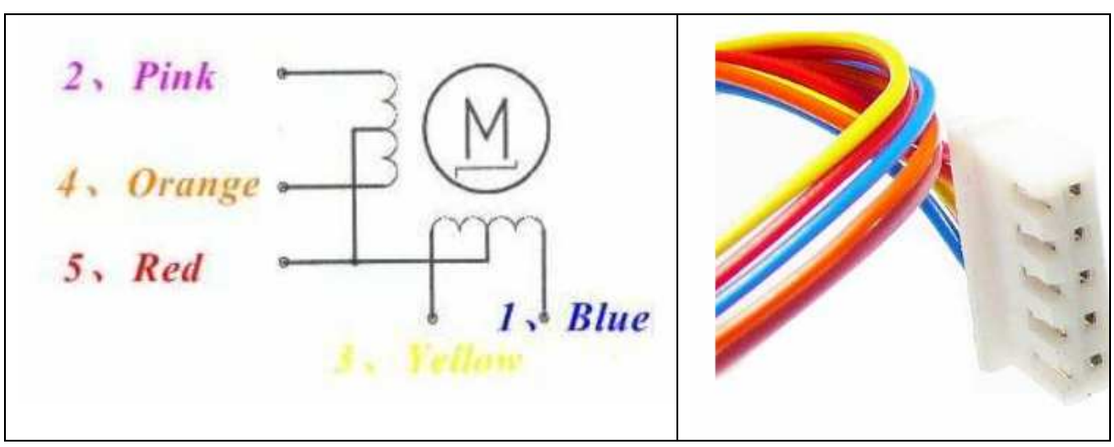
Es un motorcito unipolar con las siguientes características:

- Tensión nominal de entre 5V y 12 V.
- 4 Fases.
- Resistencia 50 Ω.
- Par motor de 34 Newton / metro más o menos 0,34 Kg por cm.
- Consumo de unos 55 mA.
- 4 pasos por vuelta.
- Reductora de 1 / 64.

Traduciendo esto, quiere decir que como es de 4 pasos (Steps), u 8 medios pasos (O half Steps) por vuelta y usa una reductora de 1 /64, por lo que necesitamos dar 8 * 64 = 512 impulsos para completar un giro completo a medios pasos.

Repasemos. Hay 4 bobinas, si las excitamos de una en una tenemos 4 pasos x 64=256 pasos por vuelta. Pero también podemos excitar la bobina mediante medios pasos, como veíamos en las tablas de la sesión previa (que es el ejemplo que vamos a hacer) y por eso a medios pasos una vuelta son 8 * 64 = 512 impulsos

Además, un par de 0,34 Kg por cm, significa que con una polea de un 1cm de diámetro colocado en el eje, este pequeño motor levantaría un peso de 350 gramos contra la gravedad, lo que no está mal.

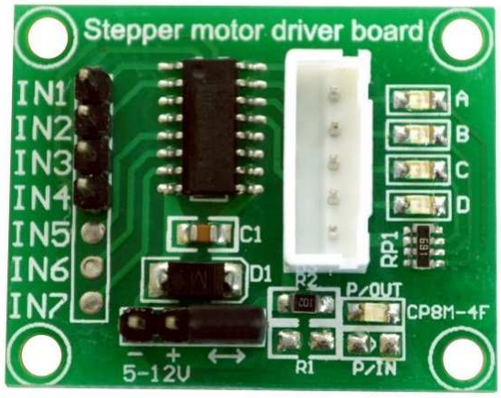


Este es su diagrama de conexión de bobinas, y además marca los colores del cable en función de su conexión interna. El motor presenta un conector al extremo en el que podemos pinchar cables de protoboard si hay que moverlo directamente, o bien para conectarlo a su adaptador.

Fijémonos que su consumo es muy bajo, de unos 55 mA, dentro del rango que nuestro USB proporciona a **Arduino** (*Siempre que no intentemos alimentarlo con un pin)* y lo alimentaremos mediante la breakboard típica que le acompaña, que suele usar un integrado del tipo **ULN2003A** que es un array de transistores Darlington, que soporta hasta 500 mA y que ya dispone de un conector para el motor y de unos pines (IN1 – IN4) para conectar a nuestro **Arduino**.

- *Sin entrar en muchos detalles, un Darlington suele ser un par de transistores bipolares colocados juntos y que se manejan como uno único.*

- *La ventaja de este esquema es que aumenta mucho la ganancia del transistor resultante y además permite la conducción de grandes corrientes y tensiones.*



CONECTANDO EL MOTOR 28BYJ-48

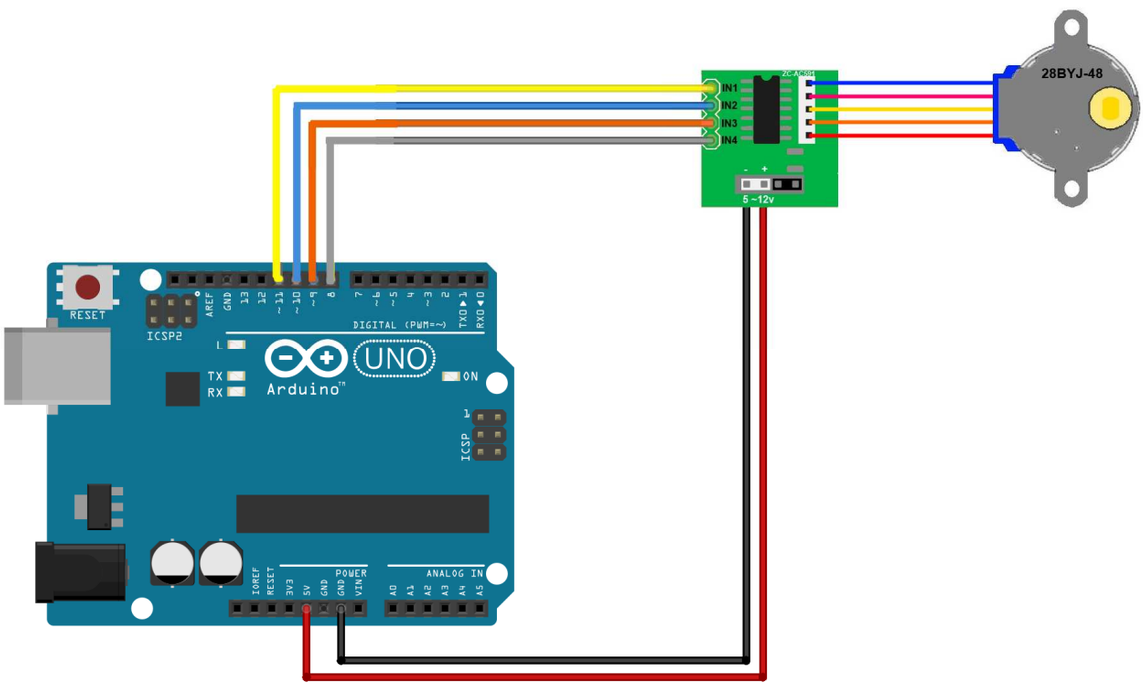
Conectar el motor primero a su conector, que tiene una posición para impedir que lo coloquemos al revés y después vamos a colocar las 4 fases a nuestro Arduino de modo que podamos excitarlas de forma independiente.

Es importante que entender la idea de que vamos a ir excitando cada una de las fases (O varias simultáneamente) en secuencia levantando a HIGH el pin de **Arduino** correspondiente.

Para conectar vuestro **Arduino** usar la siguiente tabla:

ARDUINO	11	10	9	8
BREAKOUT	IN1	IN2	IN3	IN4

Conectar GND y Vcc a los pines correspondientes del adaptador y eso es todo (*Gran ventaja cuando usamos módulos a medida*) vamos con el programa.



EL PROGRAMA DE CONTROL

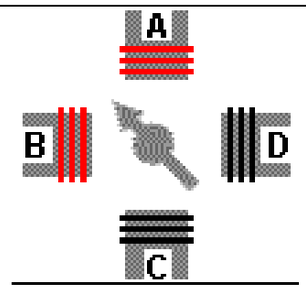
Mientras que los **motores CC** normales, están diseñados para que al alimentarlos giren de forma continua, un motor paso a paso está diseñado para girar un pasito cada vez que alimentas una de las fases.

Por eso nuestro programa tendrá que gestionar la secuencia en la que excitamos las bobinas para que el motor vaya avanzando de forma continua.

En un motor de 4 fases como este que vamos a usar hay tres maneras de hacer esta secuencia como veíamos en la sesión anterior.

Excitando dos bobinas cada vez (***Suele ser lo que recomienda el fabricante***)

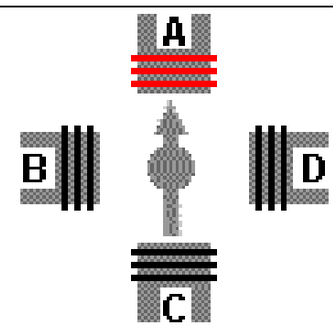
PASO	BOBINA A	BOBINA B	BOBINA C	BOBINA D
1	**ON**	**ON**	OFF	OFF
2	OFF	**ON**	**ON**	OFF
3	OFF	OFF	**ON**	**ON**
4	**ON**	OFF	OFF	**ON**



Tendríamos máximo par, buena velocidad y alto consumo.

Excitando solo una bobina cada vez (***Se le llama wave drive***):

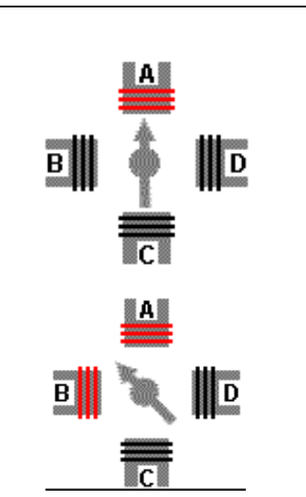
PASO	BOBINA A	BOBINA B	BOBINA C	BOBINA D
1	**ON**	OFF	OFF	OFF
2	OFF	**ON**	OFF	OFF
3	OFF	OFF	**ON**	OFF
4	OFF	OFF	OFF	**ON**



Que produciría un par menor (*Porque solo se activa una bobina en lugar de dos*) y consumo bajo.

O podríamos dar medios pasos así:

PASO	BOBINA A	BOBINA B	BOBINA C	BOBINA D
1	**ON**	OFF	OFF	OFF
2	**ON**	**ON**	OFF	OFF
3	OFF	**ON**	OFF	OFF
4	OFF	**ON**	**ON**	OFF
5	OFF	OFF	**ON**	OFF
6	OFF	OFF	**ON**	**ON**
7	OFF	OFF	OFF	**ON**
8	**ON**	OFF	OFF	**ON**



El movimiento es más suave y lento que con los métodos anteriores, y el consumo y el par es también intermedio.

Un programa elegante para controlar el motor

(NOTA: luego veremos otras opciones menos elegante pero más fácil de entender)

Como trabajaremos en Arduino?. Vamos a definir unos arrays con estas tablas para secuenciar el movimiento. Por ejemplo en el caso de usar medios pasos el array sería algo así:

```
int Paso [ 8 ][ 4 ] =
   {  {1, 0, 0, 0},
      {1, 1, 0, 0},
      {0, 1, 0, 0},
      {0, 1, 1, 0},
      {0, 0, 1, 0},
      {0, 0, 1, 1},
      {0, 0, 0, 1},
      {1, 0, 0, 1}
   };
```

Nuestro programa recorrerá el array y alimentara las bobinas de acuerdo a los valores que presenta. Empecemos con algunas definiciones:

```
#define IN1  11
#define IN2  10
#define IN3  9
#define IN4  8
int steps_left=4095;
boolean Direction = true;
int Steps = 0;           // Define el paso actual de la secuencia
```

Los ***defines*** indican a que pines de nuestro **Arduino** vamos a conectar cada una de los terminales de las bobinas del motor. Después algunas variables para control y un array que representa la secuencia. El setup es bastante sencillo:

```
void setup()
   { pinMode(IN1, OUTPUT);
     pinMode(IN2, OUTPUT);
     pinMode(IN3, OUTPUT);
     pinMode(IN4, OUTPUT);
   }
```

Vamos a definir una función llamada ***stepper()*** que avanza un paso cada vez que la invocamos, en realidad medio paso dada la matriz de excitación y que controle en que punto de la secuencia estamos. Así, el programa principal quedaría:

```
void loop()
   { while(steps_left>0)
        {
           stepper() ;     // Avanza un paso
           steps_left-- ;  // Un paso menos
```

```
            delay (1) ;
          }
      delay(300);
      Direction =! Direction;  // Invertimos la direceccion de giro
      steps_left = 4095;
    }
```

Usamos ***Steps_left*** para definir el número de pasos que queremos girar, y por eso mientras queden pasos pendientes seguimos en el **while**, que lo que hace es avanzar un paso, disminuir el número de pasos pendientes y hacer un **delay** que controla la velocidad de giro.

Cuando acaba el **while**, hacemos un **delay** para poder apreciar el final e invertimos el valor de ***direction*** para cambiar la dirección de giro. Vamos con la **funcion Stepper** que parece más complicada:

```
void stepper()            //Avanza un paso
   {  digitalWrite( IN1, Paso[Steps][ 0] );
      digitalWrite( IN2, Paso[Steps][ 1] );
      digitalWrite( IN3, Paso[Steps][ 2] );
      digitalWrite( IN4, Paso[Steps][ 3] );
      SetDirection();
   }
```

Usamos la variable **Steps** para saber en cuál de los 8 estados posibles de la matriz estamos y escribimos en las bobinas los valores que corresponden a esa situación.

Es **SetDirection()** quien va a controlar la dirección de giro y el valor de **Steps**, vamos a ver como:

```
void SetDirection()
   {
      if(Direction)
         Steps++;
      else
         Steps--;
      Steps = ( Steps + 8 ) % 8 ;
   }
```

Es muy fácil ver que si giramos en la dirección digamos positiva, hay que ir incrementando **Steps** para mantener el giro. Y tampoco es complicado ver que en el giro contrario hay que irlo decrementando para que lea el array al revés. Pero quizás la última línea merece una explicación aparte.

A medida que vamos incrementando **Steps**, queremos que cuando pase de 7 vuelva a 0 y esto se consigue muy fácil haciendo

```
Steps = Steps % 8
```

Es decir tomando el modulo con respecto a 8, pero si lo vamos decrementando, alcanzaremos valores negativos, y el módulo de algo negativo sigue siendo negativo, lo que no nos vale, porque necesitamos que el siguiente valor a 0 sea 7 y no -1.

Esto es lo que conseguimos haciendo

```
Steps = ( Steps + 8 ) % 8 ;
```

Nota: El operador módulo, % en lenguaje C++, es un operador binario y tiene sentido solo para números enteros. Se utiliza de la siguiente manera; sean A y B dos números enteros, si realizamos la operación A%B el resultado que obtenemos será el resto de la división entre A y B.

He visto en Internet que esto lo hacen con un programita así:

```
if (Steps>7)
    Steps=0 ;
if (Steps<0)
    Steps=7 ;
```

Son completamente equivalentes (Y además es un código mas compacto el primero).

El programa quedaría asi:

```
int TP=100; //Tiempo entre pasos
void setup() {
pinMode(8, OUTPUT); // configura 'pin' como salida
pinMode(9, OUTPUT); // configura 'pin' como salida
pinMode(10, OUTPUT); // configura 'pin' como salida
pinMode(11, OUTPUT); // configura 'pin' como salida
}
void loop() {
for (int i=0; i<513; i++) //Una vuelta
{
digitalWrite(8, HIGH); // enciende el pin
delay(TP);            // espera
digitalWrite(8, LOW); // apaga el pin
digitalWrite(9, HIGH);// enciende el pin
delay(TP);            // espera
digitalWrite(9, LOW);  // apaga el pin
digitalWrite(10, HIGH);  // enciende el pin
delay(TP);             // espera
digitalWrite(10, LOW);  // apaga el pin
digitalWrite(11, HIGH);  // Enciende el pin
delay(TP);            // espera
digitalWrite(11, LOW);  // apaga el pin
}
 }
```

El resultado es un motorcito girando primero en una dirección y después en la contraria.

Otra opción para manejar el motor (Wave Drive - por Ola)

El programa anterior es solución elegante para manejar un motor. Ahora usando los mismos conceptos físicos del motor, utilizaremos un programa más sencillo.

PASO	BOBINA A	BOBINA B	BOBINA C	BOBINA D
1	ON	OFF	OFF	OFF
2	OFF	ON	OFF	OFF
3	OFF	OFF	ON	OFF
4	OFF	OFF	OFF	ON

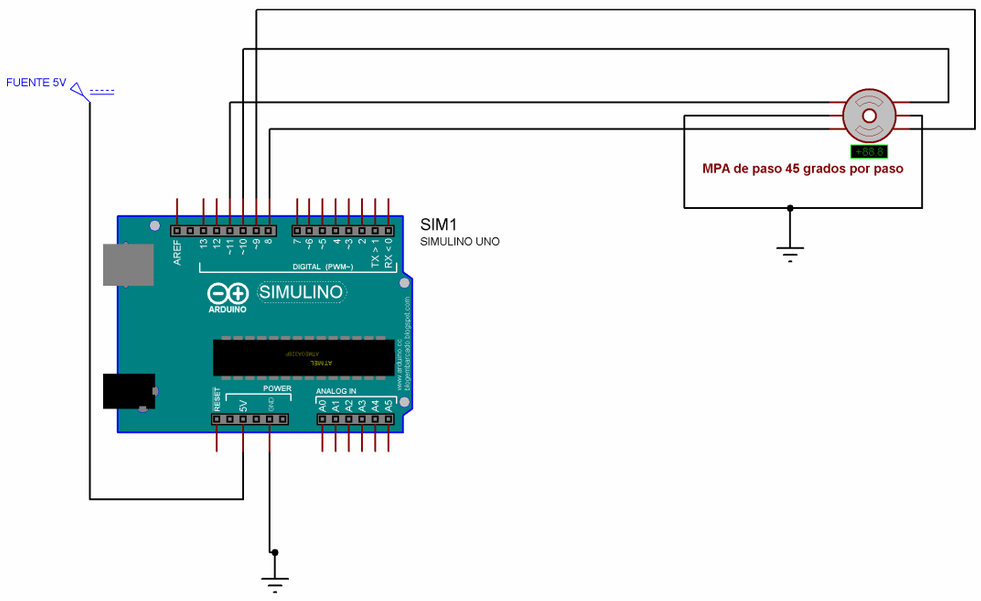
Según creo la manera mas sencilla para programar el movimiento de un MPA (motor paso a paso) es mediante:

manejo por ola o Wave Drive

El siguiente programa realiza ese manejo:

```
//Manejo de MAP por OLA

int TP=100; //Tiempo entre pasos para proteus

void setup() {
pinMode(8, OUTPUT); // configura ‘pin’ como salida
pinMode(9, OUTPUT); // configura ‘pin’ como salida
pinMode(10, OUTPUT); // configura ‘pin’ como salida
pinMode(11, OUTPUT); // configura ‘pin’ como salida

}
void loop() {
for (int i=0; i<513; i++) //Una vuelta

{
digitalWrite(8, HIGH); // enciende el pin
delay(TP);            // espera
digitalWrite(8, LOW); // apaga el pin

digitalWrite(9, HIGH);// enciende el pin
delay(TP);            // espera
digitalWrite(9, LOW);  // apaga el pin

digitalWrite(10, HIGH);  // enciende el pin
delay(TP);             // espera
digitalWrite(10, LOW);  // apaga el pin

digitalWrite(11, HIGH);  // Enciende el pin
delay(TP);            // espera
digitalWrite(11, LOW);  // apaga el pin

}
 }
```

Motores PASO A PASO UNIPOLARES (manejo por OLA)



Otra opción para manejar el motor (Por 2 fases))

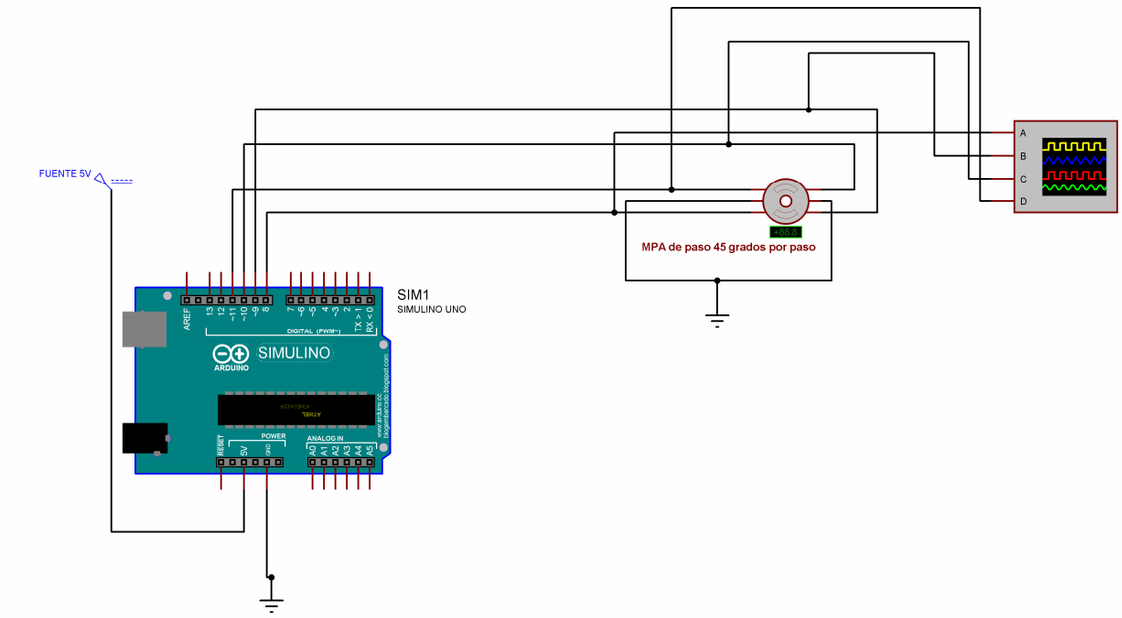
Es la secuencia que recomienda el fabricante, porque al haber siempre dos bobinas activas tenemos un par motor (Torque) importante para avanzar y para mantener la posición.

PASO	BOBINA A	BOBINA B	BOBINA C	BOBINA D
1	**ON**	**ON**	OFF	OFF
2	OFF	**ON**	**ON**	OFF
3	OFF	OFF	**ON**	**ON**
4	**ON**	OFF	OFF	**ON**

El programa sería:

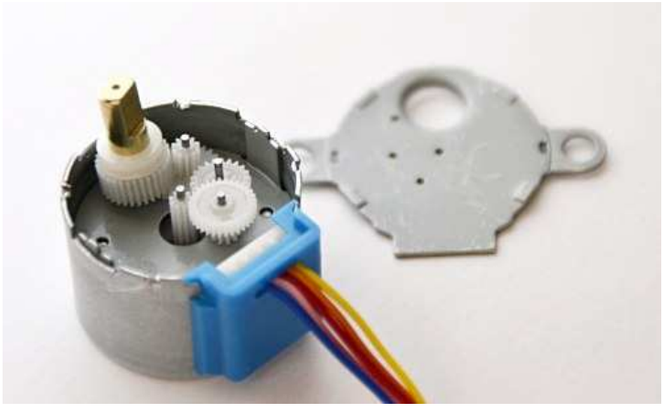
```
//Motor PAP alimentado por 2 fases

//continua haciendo esto infinitamente

int TP=100; //Tiempo entre pasos

void setup() {
pinMode(8, OUTPUT); // configura ‘pin’ como salida
pinMode(9, OUTPUT); // configura ‘pin’ como salida
pinMode(10, OUTPUT); // configura ‘pin’ como salida
pinMode(11, OUTPUT); // configura ‘pin’ como salida

}

void loop() {


digitalWrite(8, HIGH); // 1
digitalWrite(9, HIGH); // 1
digitalWrite(10, LOW);  // 0
digitalWrite(11, LOW);  // 0

delay(TP);          //espera


digitalWrite(8, LOW); // 0
digitalWrite(9, HIGH); // 1
digitalWrite(10, HIGH);  // 1
digitalWrite(11, LOW);  // 0

delay(TP);          //espera


digitalWrite(8, LOW); // 0
digitalWrite(9, LOW); // 0
digitalWrite(10, HIGH);  // 1
digitalWrite(11, HIGH);  // 1

delay(TP);          //espera

digitalWrite(10, LOW);  // 0

digitalWrite(8, HIGH); // 1
digitalWrite(9, LOW); // 0

digitalWrite(11, HIGH);  // 1

delay(TP);          //espera


}
```

Motores PASO A PASO UNIPOLARES (manejo por 2 fases)



NOTA IMPORTANTE: El motor del Proteus esta seteado en pasos de 45 grados. Se debe tener en cuenta que el motor que trabajemos en la practica puede tener características distintas a estas, lo que seguramente provocara que gire mas lento o mas rápido, dependiendo del ángulo por paso y si tiene caja reductora.

En el KIT de Arduino utilizado viene el 28BYJ-48.



RESUMEN DE LA SESIÓN

- Montamos un primer circuito de control de nuestro motor paso a paso 28BYJ-48.
- Vimos las varias maneras de controlarlo según como excitemos las fases del motor.