



Aguijón

Notas de aplicación

Nota de aplicación 23:

Unipolar Stepper Motor

Descripción:

Controlar un motor a pasos unipolar a través del puerto de salida Open Colector.

Herramientas:

1. Aguijón 3.0, Aguijón 4.0 ó Aguijón 4.1
2. MPLAB X®
3. Aguijón HID bootloader
4. Cable USB 'A' to 'A'
5. Librerías HammerHead.
6. Motor a pasos unipolar (12v Max).

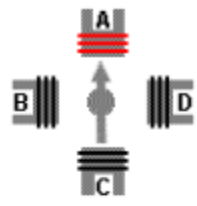
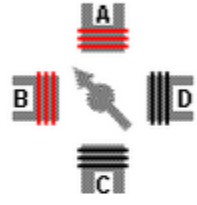
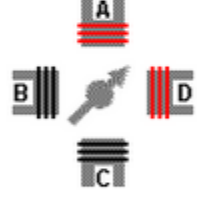
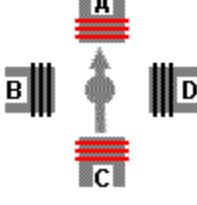
Pasos:

1. Motores a pasos

Los motores paso a paso son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos.

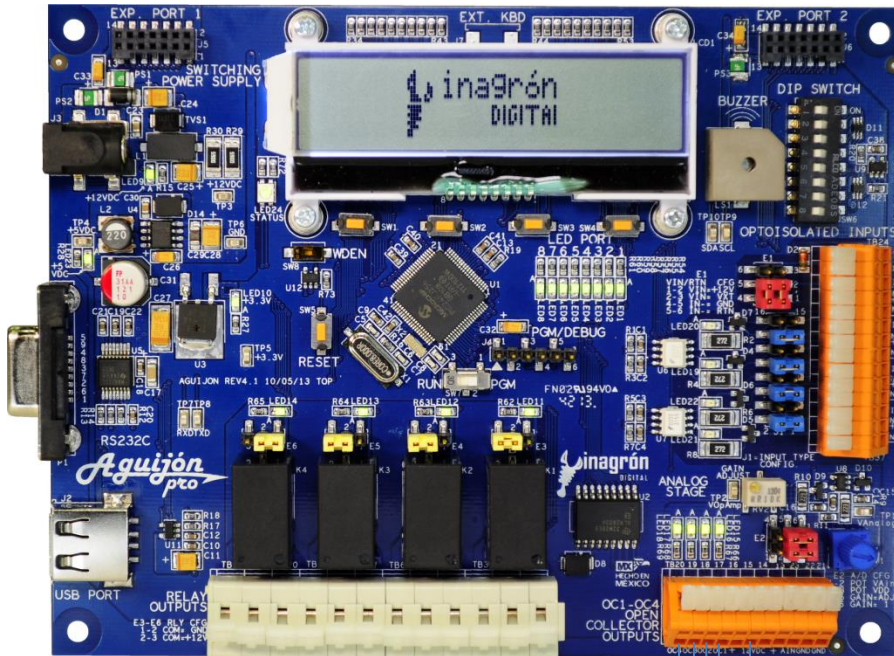
La característica principal de estos motores es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8° .

- **Unipolar:** Estos motores suelen tener 6 o 5 cables de salida, dependiendo de su conexión interna
- **Identificar bobinas (A, B, C y D)**

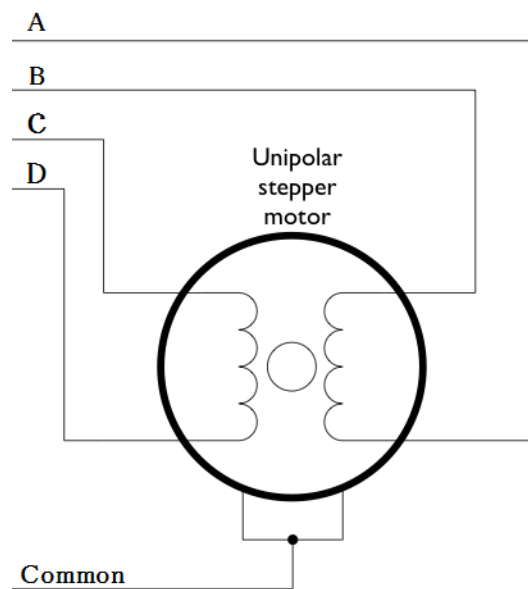
Seleccionar un cable y conectarlo a masa. Ese será llamado cable A .	
Manteniendo el cable A conectado a masa, probar cuál de los tres cables restantes provoca un paso en sentido antihorario al ser conectado también a masa. Ese será el cable B .	
Manteniendo el cable A conectado a masa, probar cuál de los dos cables restantes provoca un paso en sentido horario al ser conectado a masa. Ese será el cable D .	
El último cable debería ser el cable C . Para comprobarlo, basta con conectarlo a masa, lo que no debería generar movimiento alguno debido a que es la bobina opuesta a la A .	

NOTA: La nomenclatura de los cables (A, B, C y D) es totalmente arbitraria.

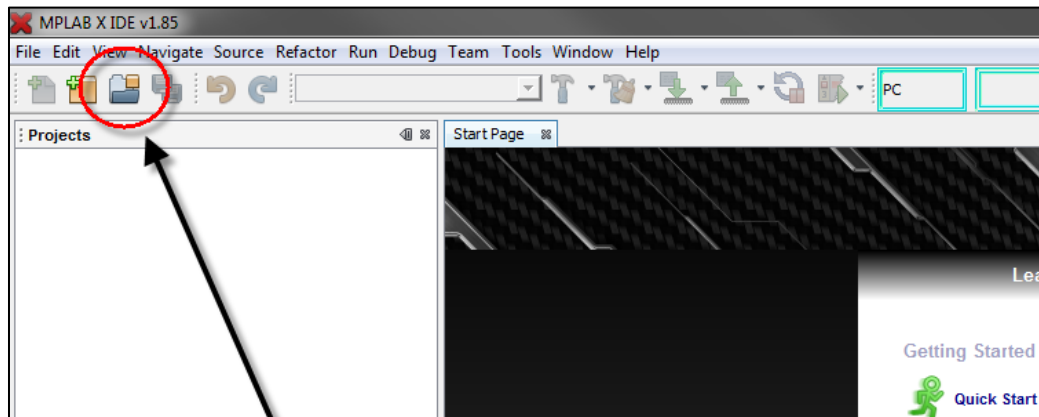
2. Diagrama de Conexión



OC4	OC3	OC2	OC1	+12V
A	B	C	D	Common

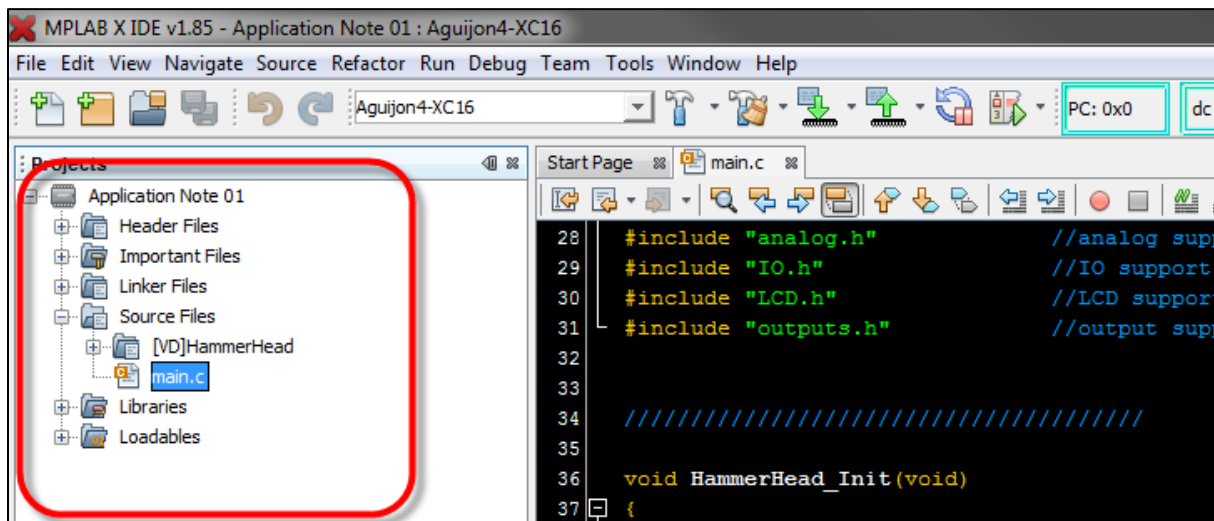


3. Abrir MPLAB X® y cargar el archivo del proyecto: **Application Note 23.X**



Haz 'clic' aquí y
abre el proyecto

4. Abrir el archivo **main.c**



5. Ir a la línea #73.

Utilizaremos la siguiente función:

```
66     HammerHead_Init(); //initialize [VD]HammerHead
67     LCD_IntroAnimation();
68     LCD_PutStr(1,0,"Vinagron Digital",TRUE);
69     LCD_PutStr(2,0,"Application Note 23",FALSE);
70
71     for(;;){
72
73         keyboard=SW_Read(); //Read keyboard
74
75         switch (keyboard)
76         {
77             case 1:
78                 //Create a strings for the LCD//
79                 LCD_PutStr(1,0,"15 Half step",TRUE);
80
```

SW_Read ();

- Esta función lee el puerto de PUSH BUTTONS;
Regresa un valor equivalente al Push-Button presionado (carácter del 1 al 4)

6. Ir a la línea #79

Utilizaremos la siguiente función:

```
72
73     keyboard=SW_Read();           //Read keyboard
74
75     switch (keyboard)
76     {
77         case 1:
78             //Create a strings for the LCD//
79             LCD_PutStr(1,0,"15 Half step",TRUE);
80
81             Unipolar_Set(MODE_HALF_STEP,48,SPEED_MOTOR_4); /*Setting mo
82             Steps(15);           //Move 15 Steps
83             break;
84         case 2:
85             //Create a strings for the LCD//
86             LCD_PutStr(1,0,"-15 Simple step",TRUE);
```

LCD_PutStr (int y, int x, char *msg, BOOL clear);

- Función que muestra una cadena de caracteres en la pantalla LCD; donde:
Int y = Coordenada en y (Valor entero del 1 al 2.)
Int X = Coordenada en x(Valor entero del 0 al 20.)
Char *msg = Cadena de caracteres (De 0 a 20 caracteres)
BOOL clear = Determina si se borra la pantalla antes de escribir
(TRUE =Borrar, FALSE =No borrar).

7. Ir a la línea #81

Utilizaremos la siguiente función:

```
74
75     switch (keyboard)
76     {
77         case 1:
78             //Create a strings for the LCD//
79             LCD_PutStr(1,0,"15 Half step",TRUE);
80
81             Unipolar_Set(MODE_HALF_STEP,48,SPEED_MOTOR_4); /*Setting mo
82             Steps(15);      //Move 15 Steps
83             break;
84         case 2:
85             //Create a strings for the LCD//
86             LCD_PutStr(1,0,"-15 Simple step",TRUE);
87
88             Unipolar_Set(MODE_SIMPLE_STEP,48,SPEED_MOTOR_4); /*Settin
```

Unipolar_Set (BOOL Mode, int Steps, int Speed);

- Función que configura, tanto modo pasos y velocidad de nuestro motor; donde:
BOOL Mode = Modo de paso que deseamos utilizar (MODE_HALF_STEP ó MODE_SIMPLE_STEP)
Int Steps = Cantidad de pasos por vuelta del motor (Numero entero del 1 al 200).
Int Speed=Velocidad de conmutación del motor (SPEED_MOTOR_MIN – SPEED_MOTOR_1 – SPEED_MOTOR_2 - SPEED_MOTOR_3 - SPEED_MOTOR_4 - SPEED_MOTOR_5 - SPEED_MOTOR_MAX)
- **Int Steps**= Cantidad de pasos por vuelta del motor.
Para determinar el número de pasos requeridos para completar una vuelta del motor, es necesario disponer de la hoja de datos del fabricante, dicho número dependerá directamente de los grados por paso del motor.

8. Ir a la línea #82

Utilizaremos la siguiente función:

```
75     switch (keyboard)
76     {
77         case 1:
78             //Create a strings for the LCD//
79             LCD_PutStr(1,0,"15 Half step",TRUE);
80
81             Unipolar_Set(MODE_HALF_STEP,48,SPEED_MOTOR_4); /*Setting mo
82             Steps(15); //Move 15 Steps
83             break;
84         case 2:
85             //Create a strings for the LCD//
86             LCD_PutStr(1,0,"-15 Simple step",TRUE);
87
88             Unipolar_Set(MODE_SIMPLE_STEP,48,SPEED_MOTOR_4); /*Settin
89             Steps(-15); //Move -15 Steps
```

Steps (int latets_steps);

- Función que mueve nuestro motor una determinada cantidad de pasos, a una velocidad y modo determinado por la llamada más reciente a la función Unipolar_Set() :
donde;

Int latets_steps = Número de pasos por recorrer. (Valor entero positivo para pasos en sentido horario, valor entero negativo para pasos en sentido anti horario)

- **Nota**

Las función **Steps** es una funciones bloqueante; esto es, esperará a finalizar el movimiento para devolver el control a la siguiente línea de tu código.

9. Ir a la línea #102

Utilizaremos la siguiente función:

```
93         LCD_PutStr(1,0,"180\xDF Speed max",TRUE);
94
95         Unipolar_Set(MODE_SIMPLE_STEP,48,SPEED_MOTOR_MAX); /*Settin
96         Degrees(180); //Move 180 degrees
97         break;
98     case 4:
99         //Create a strings for the LCD//
100        LCD_PutStr(1,0,"-90\xDF ",TRUE);
101
102        Degrees(-90); //Move -180 degrees
103        break;
104    }
105 }
106 return 0;
107 }
```

Degrees (int degrees);

- Función que mueve nuestro motor una determinada cantidad de grados, a una velocidad y modo determinado por la llamada más reciente a la función Unipolar_Set() ; donde:

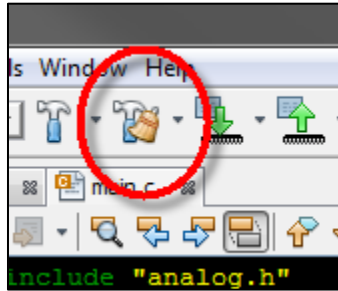
Int degrees = Número de grados por recorrer. (Valor entero positivo para grados en sentido horario, valor entero negativo para grados en sentido anti horario)

- **Nota**

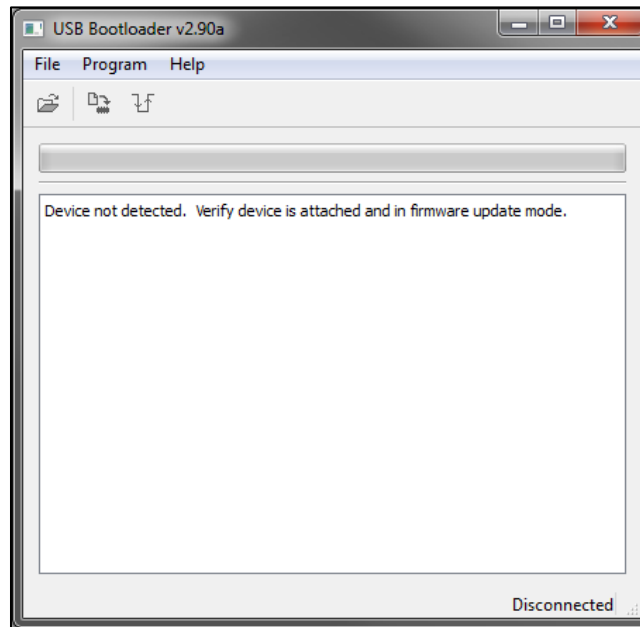
Las función **Degrees** es una funciones bloqueante; esto es, esperará a finalizar el movimiento para devolver el control a la siguiente línea de tu código.

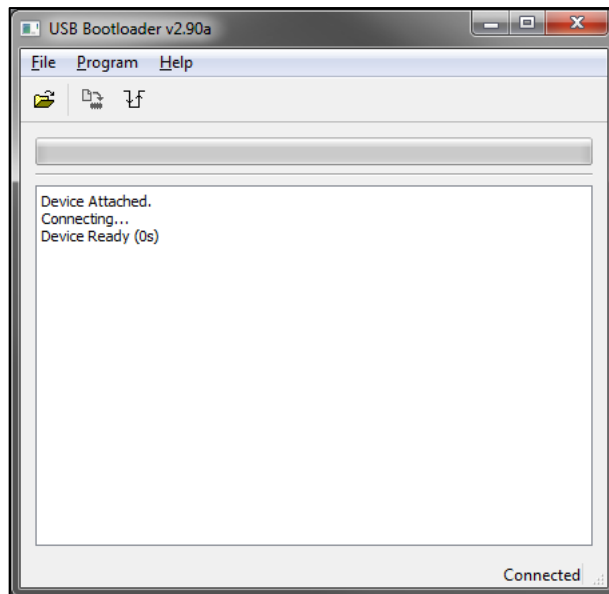
10. Compilar y programar

Al hacer clic en el ícono de compilar, y si no hay errores de compilación, el bootloader será cargado automáticamente.



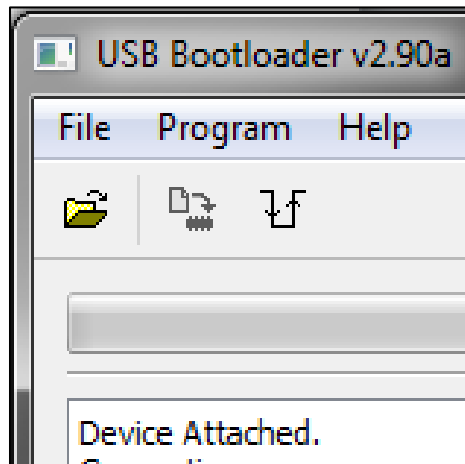
Cuando aparezca la ventana del bootloader, presione el Push-Button número 1 y conecte la fuente de voltaje o encienda el Aguijón y mantenga el PB1 presionado hasta que los LEDs empiecen a parpadear.





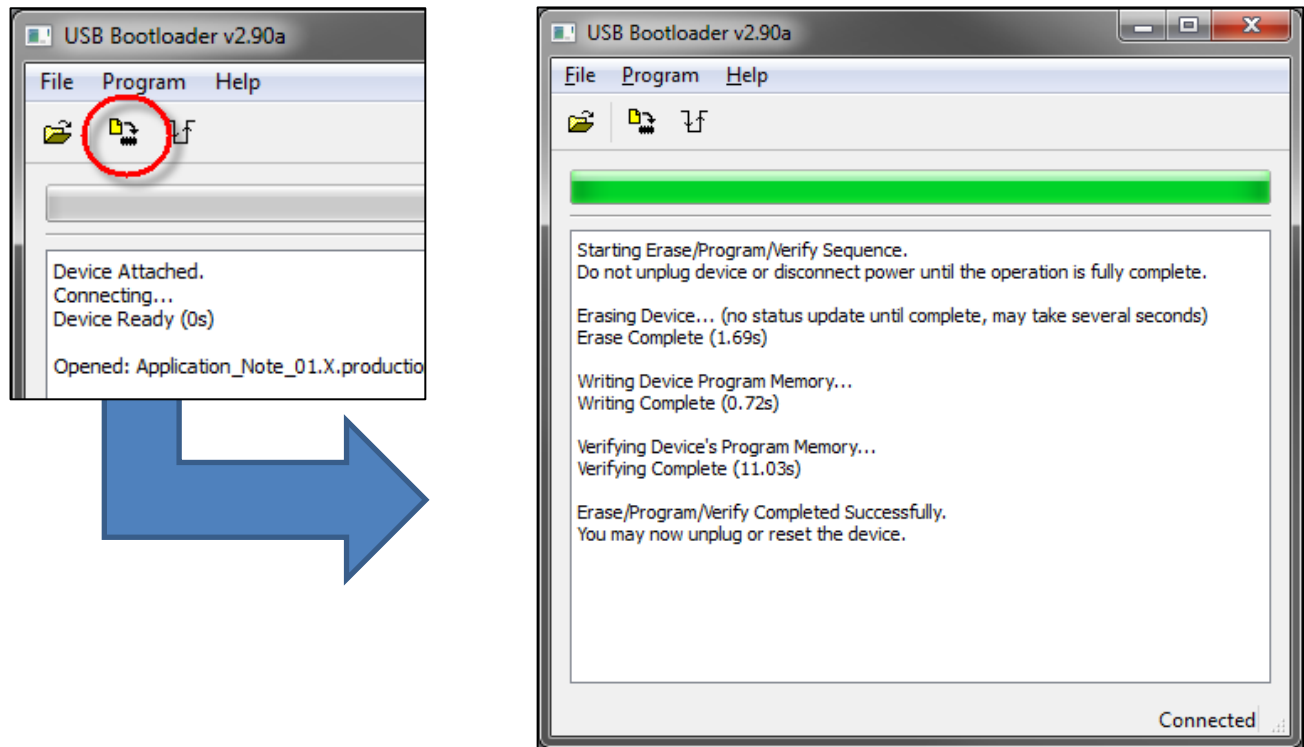
La ventana del Bootloader indicará la conexión establecida con el aguijón:

11. Hacer Clic en Abrir y Cargar el archivo **Application Note 23.X.production.hex**

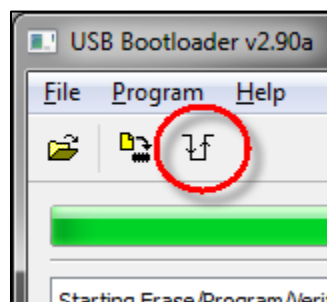


El archivo, depende de la plataforma de hardware.

Una vez cargado el archivo, hacer clic en el ícono de programar.



Hacer clic en el ícono de Reset cuando en la ventana del Bootloader se indique que se terminó de programar con éxito.



Una vez programado podemos verificar el programa corriendo en la tarjeta.

12. Para verificar el funcionamiento del programa verifique que el Motor Unipolar Conectado al puerto de Open Collector gire en diferentes sentidos, modos, y velocidades dependiendo esto del puerto de entradas Push Buttons.