



*Aguijón*

---

*Notas de aplicación*

## **Nota de aplicación 33:**

### Genérate sine wave

#### **Descripción:**

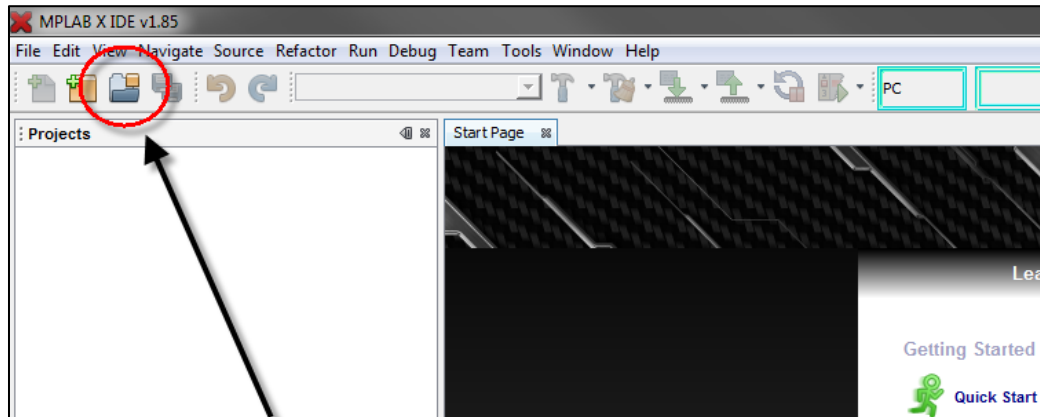
Generar una onda Senoidal desde el puerto de salida Open-Collector, teniendo la posibilidad de variar la frecuencia de esta.

#### **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. Resistencias (varias)
7. Capacitores(Varios)
8. Osciloscopio.

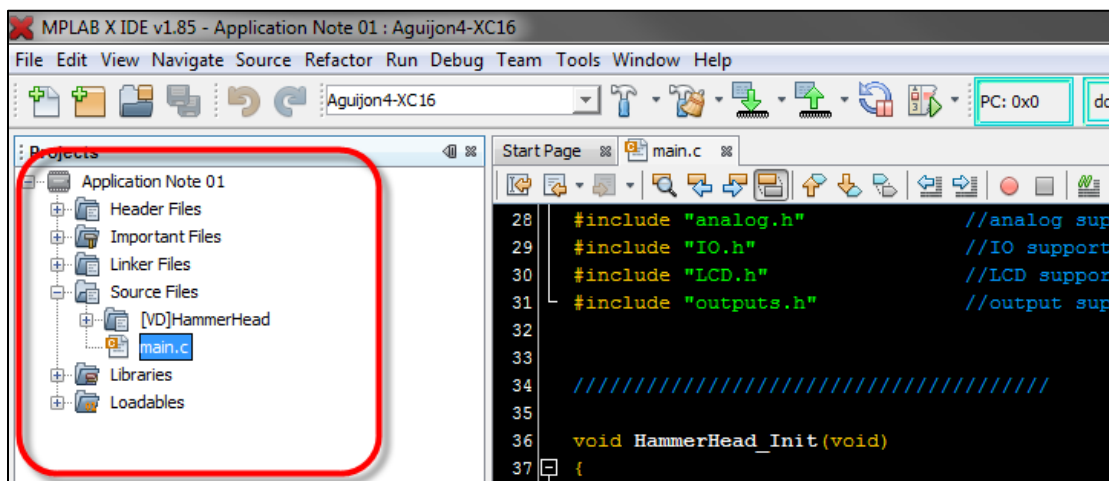
## Pasos:

1. Abrir MPLAB X® y cargar el archivo del proyecto: **Application Note 33.X**



Haz 'clic' aquí y  
abre el proyecto

2. Abrir el archivo **main.c**



3. Ir a la línea #42.

Utilizaremos la siguiente función:

```
35
36 //////////////////////////////////////////////////
37
38 unsigned int duty_value=0;
39
40 /* Duty value for 0° 10° 20° 30° 40° 50° 60° 70° 80° 90° 100 110 120 130 140 150 160 170*/
41 /*      180 190 200 210 220 230 240 250 260 270 280 290 300 310 320 330 340 350 */
42 int Duty [36]={50, 58, 67, 75, 82, 88, 93, 97, 98, 99, 98, 97, 93, 88, 82, 75, 67, 58,
43              50, 41, 32, 25, 18, 11, 6, 3, 2, 1, 2, 3, 6, 11, 18, 25, 32, 41 };
44
45 void __attribute__((__interrupt__, __shadow__, __no_auto_psv__)) _T1Interrupt(void)
46 {
47
48     /*Configure PWM*/
49     CloseOC3();
```

- En este arreglo se almacenan los valores calculados a través de una función seno, dichos valores serán asignados al ciclo de trabajo de un PWM, el cual será actualizado cada desbordamiento del Timer 1.

4. Ir a la línea #72

Utilizaremos la siguiente función:

```
65      /*Inputs*/
66      ADC_Init();
67
68      /*LCD*/
69      LCD_Init(LCD_MODE_1);
70
71      /*PWM*/
72      OC_PWMChannelConfig(4); //assigns PWM to open collector output #1
73
74      /*Timer*/
75      Timer1_Init();
76
77
78  #if defined(USE_LCD_EXTRA_FEATURES)
79      LCD_BacklightFadeIn();
```

### OC\_PWMChannelConfig (int num);

- Función que configura un PWM a la salida Open Collector seleccionada; donde:  
**Int num** = Número de OPEN COLLECTOR que queremos utilizar (Valor entero del 1 al 4.)

5. Ir a la línea #75

Utilizaremos la siguiente función:

```
68      /*LCD*/
69      LCD_Init(LCD_MODE_1);
70
71      /*PWM*/
72      OC_PWMChannelConfig(4);      //assigns PWM to open collector output #1
73
74      /*Timer*/
75      Timer1_Init();
76
77
78  #if defined(USE_LCD_EXTRA_FEATURES)
79      LCD_BacklightFadeIn();
80  #else
81      LCD_BacklightSet(BLIGHT_LVL_4);
82  #endif
```

**Timer1\_Init ();**

*Función que Inicializa los registros de control del Timer 1 y establece un Prescaler de 1:64*

6. Ir a la línea #101.

Utilizaremos la siguiente función:

```
94     int    timer_tick=0;
95
96     HammerHead_Init(); //initialize [VD]HammerHead
97     LCD_IntroAnimation();
98     LCD_PutStr(1,0,"Vinagron Digital",TRUE);
99     LCD_PutStr(2,0,"Application Note 33",FALSE);
100
101     Timer1_Enable();
102
103     for(;;){
104
105         timer_tick=ADC_Range(45,500);           //Read ADC range of 45 to 500
106
107         Timer1_SetTick(timer_tick);             //SetTick of Timer 1
108     }
```

**Timer1\_Eneable ();**

- Función que habilita el módulo Timer 1;

**Timer1\_Disable ();**

- Función que deshabilita el módulo Timer 1;

7. Ir a la línea #105.

Utilizaremos la siguiente función:

```
98     LCD_PutStr(1,0,"Vinagron Digital",TRUE);
99     LCD_PutStr(2,0,"Application Note 33",FALSE);
100
101     Timer1_Enable();
102
103     for(;;){
104
105         timer_tick=ADC_Range(45,500);    //Read ADC range of 45 to 500
106
107         Timer1_SetTick(timer_tick);      //SetTick of Timer 1
108
109         Frequency=1/(timer_tick*.000144); //Calculate Frequency
110
111         /*Create a string for the LCD*/
112         sprintf(lcdMSG,"Frequency= %.2fHZ",Frequency);
```

**ADC\_Range (int min\_val, int max\_val);**

- Esta función lee el puerto ADC y devuelve un valor proporcional al Rango establecido;  
Donde:  
**Int min\_val** = valor mínimo del rango (Entero de 0 a 1023)  
**Int max\_val** = valor máximo del rango (Entero de 0 a 1023)  
Devuelve un valor proporcional al rango (Entero de min\_val a max\_val).



Ir a la línea #107

Utilizaremos la siguiente función:

```
100
101     Timer1_Enable();
102
103     for(;;){
104
105         timer_tick=ADC_Range(45,500);           //Read ADC range of 45 to 500
106
107         Timer1_SetTick(timer_tick);             //SetTick of Timer 1
108
109         Frequency=1/(timer_tick*.000144);       //Calculate Frequency
110
111         /*Create a string for the LCD*/
112         sprintf(lcdMSG,"Frequency= %.2fHZ",Frequency);
113         LCD_PutStr(1,6,"Sine wave",TRUE);
114         LCD_PutStr(2,0,lcdMSG,FALSE);
```

**Timer1\_SetTick (int tick\_ms);**

- Función que ajusta el temporizador 1; donde:

**Int tick\_ms =** Valor del preset (

- TIMER\_TICK\_100M
  - TIMER\_TICK\_50MS
  - TIMER\_TICK\_10MS
  - TIMER\_TICK\_5MS
  - TIMER\_TICK\_1MS
  - TIMER\_TICK\_500uS
- ).

8. Ir a la línea #112

Utilizaremos la siguiente función:

```
105     timer_tick=ADC_Range(45,500);           //Read ADC range of 45 to 500
106
107     Timer1_SetTick(timer_tick);               //SetTick of Timer 1
108
109     Frequency=1/(timer_tick*.000144);        //Calculate Frequency
110
111     /*Create a string for the LCD*/
112     sprintf(lcdMSG,"Frequency= %.2fHZ",Frequency);
113     LCD_PutStr(1,6,"Sine wave",TRUE);
114     LCD_PutStr(2,0,lcdMSG,FALSE);
115
116     delayms(100);                            //Delay 100 milliseconds
117 }
118 return 0;
119 }
```

**sprintf (char \*, const char \*, ...);**

- Función que Genera una cadena de caracteres y lo asigna a una variable; Donde:  
**Char \*** = Variable a asignar cadena de caracteres (Variable de tipo char)  
**const char \***, = Cadena de caracteres a asignar a la variable.

9. Ir a la línea #113

Utilizaremos la siguiente función:

```
106     Timer1_SetTick(timer_tick);           //SetTick of Timer 1
107
108     Frequency=1/(timer_tick*.000144);     //Calculate Frequency
109
110     /*Create a string for the LCD*/
111     sprintf(lcdMSG,"Frequency= %.2fHZ",Frequency);
112     LCD_PutStr(1,6,"Sine wave",TRUE);
113     LCD_PutStr(2,0,lcdMSG,FALSE);
114
115
116     delays(100);           //Delay 100 milliseconds
117 }
118 return 0;
119 }
```

**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).

10. Ir a la línea #116.

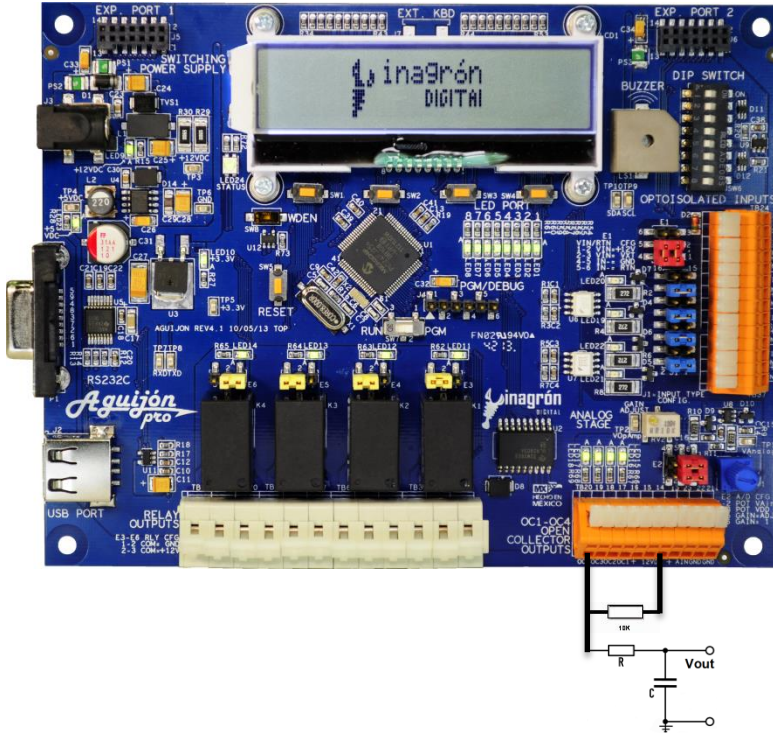
Utilizaremos la siguiente función:

```
105     timer_tick=ADC_Range(45,500);           //Read ADC range of 45 to 500
106
107     Timer1_SetTick(timer_tick);               //SetTick of Timer 1
108
109     Frequency=1/(timer_tick*.000144);         //Calculate Frequency
110
111     /*Create a string for the LCD*/
112     sprintf(lcdMSG,"Frequency= %.2fHZ",Frequency);
113     LCD_PutStr(1,6,"Sine wave",TRUE);
114     LCD_PutStr(2,0,lcdMSG,FALSE);
115
116     delayms(100);                             //Delay 100 milliseconds
117 }
118 return 0;
119 }
```

### **Delayms (ms);**

- Función que Pausa el programa por un tiempo determinado (en milisegundos); donde: **ms** = el número de milisegundos que se desea pausar el programa.

## 11. Diagrama de conexión.



- Para calcular los valores del circuito RC utilice las siguientes formulas.

Para calcular el capacitor, con una resistencia fija utilice:

$$C = \frac{1}{(F)(R)(18)}$$

Para calcular la resistencia, con un capacitor fijo utilice:

$$R = \frac{1}{(F)(C)(18)}$$

Dónde:

C=Valor del capacitor en Faradios

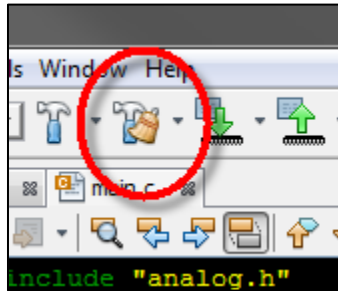
R=Valor de resistencia en ohm's

F=Valor de frecuencia en Hertz

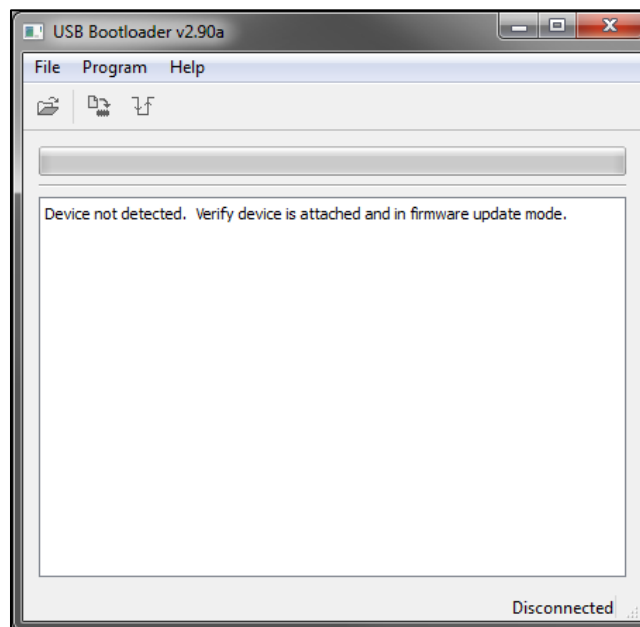
18=Valor constante

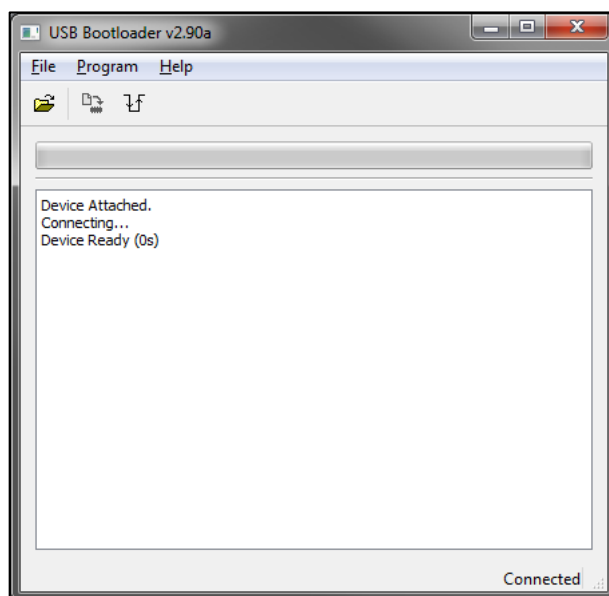
## 12. Compilar y programar

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



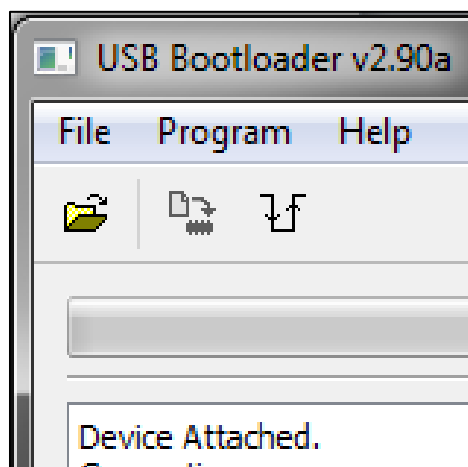
Cuando aparezca la ventana del bootloader, presione el Push-Button número 1 y conecte la fuente de voltaje o encienda el Aguijón y mantenga el PB1 presionado hasta que los LEDs empiecen a parpadear.





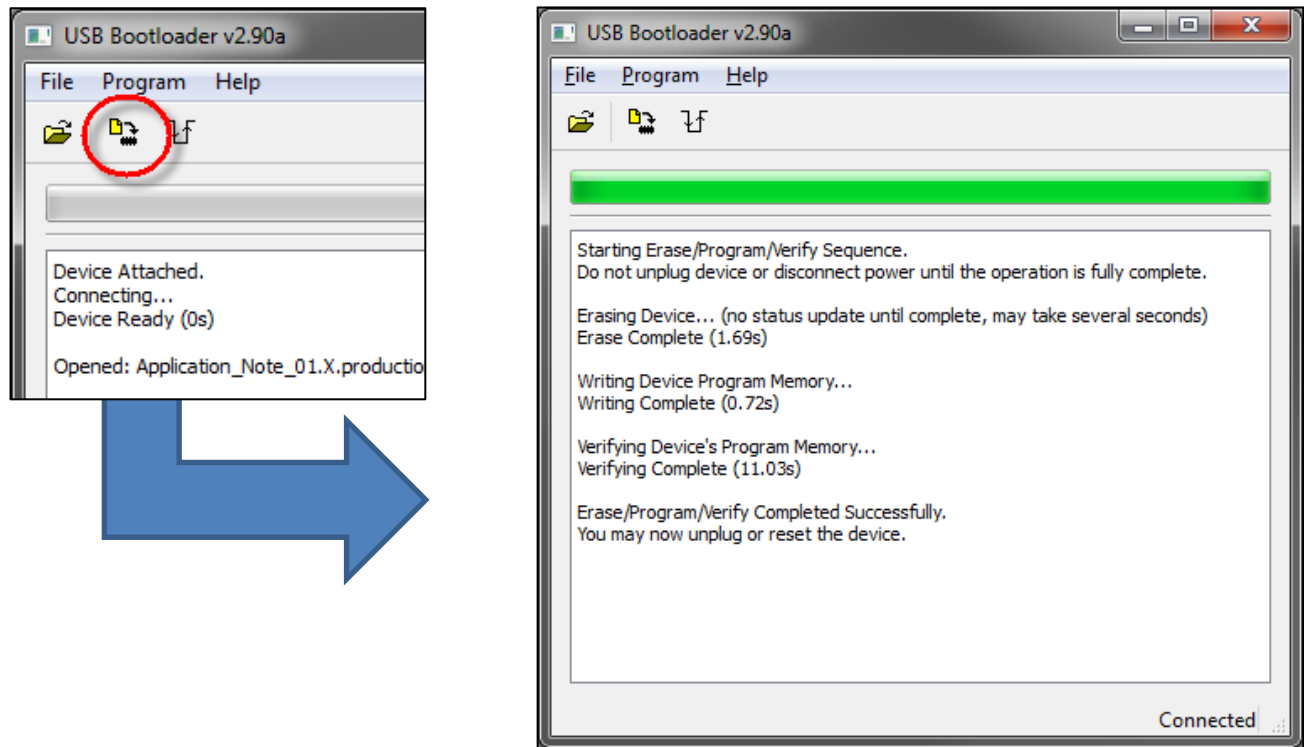
La ventana del Bootloader indicará la conexión establecida con el aguijón:

13. Hacer Clic en Abrir y Cargar el archivo **Application Note 33.X.production.hex**

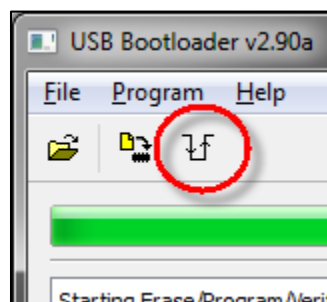


El archivo, depende de la plataforma de hardware.

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.

14. Para verificar el funcionamiento del programa verifique al implementar el circuito RC previamente calculado, en la salida de este se genere una onda Senoidal de frecuencia variable a través del puerto de entrada ADC, esto mientras continuamente se monitorea el valor de la frecuencia en la pantalla LCD.