



Aguijón

Notas de aplicación

Nota de aplicación 27:

irDA Receiver

Descripción:

Mediante el uso de un módulo de recepción infrarroja, decodificar la señal enviada por algún control remoto.

Herramientas:

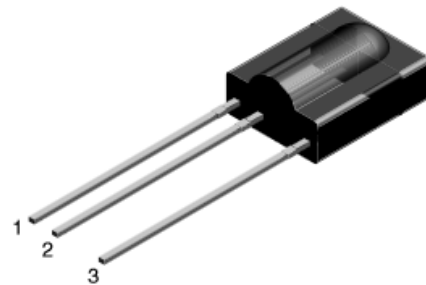
1. Aguijón 3.0, Aguijón 4.0 ó Aguijón 4.1
2. MPLAB X®
3. Aguijón HID bootloader
4. Cable USB 'A' to 'A'
5. Librerías HammerHead.
6. Módulo de recepción infrarroja.
7. Control remoto universal.

Pasos:

1. Modulo receptor de infrarrojos

Este es un dispositivo electrónico capaz de medir la radiación electromagnética infrarroja de los cuerpos en su campo de visión. Todos los cuerpos reflejan una cierta cantidad de radiación, esta resulta invisible para nuestros ojos pero no para estos aparatos electrónicos, ya que se encuentran en el rango del espectro justo por debajo de la luz visible.

Ejemplo(TSOP31238)



MECHANICAL DATA

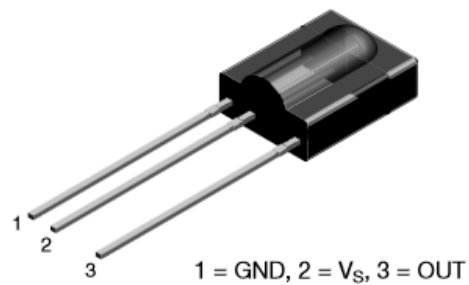
Pinning:

1 = GND, 2 = V_s , 3 = OUT

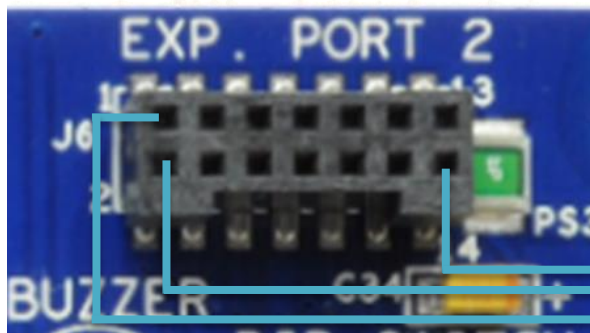
2. Control remoto universal



3. Diagrama de conexión

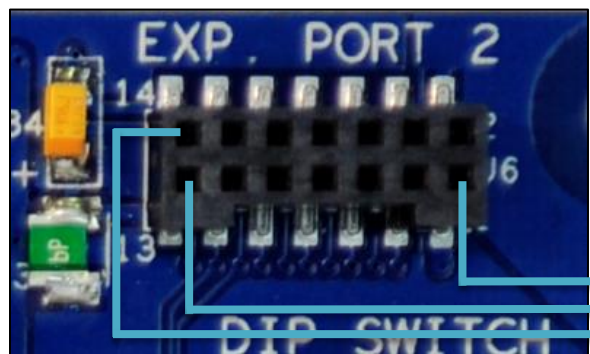


Puerto de expansión 2, Aguijon 4.0



Pin 1	Pin 2	Pin 14
GND	RP14	+3.3V

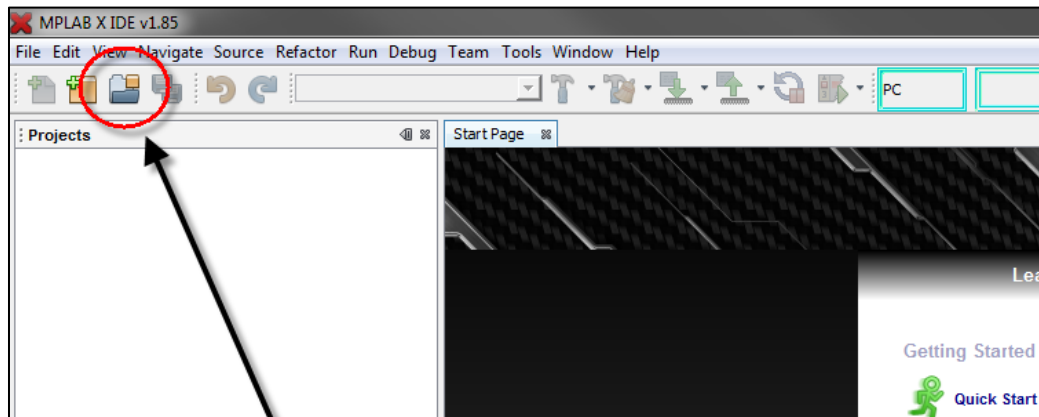
Puerto de expansión 2, Aguijon 4.1



Pin 14	Pin 13	Pin 12
GND	+3.3V	RP14

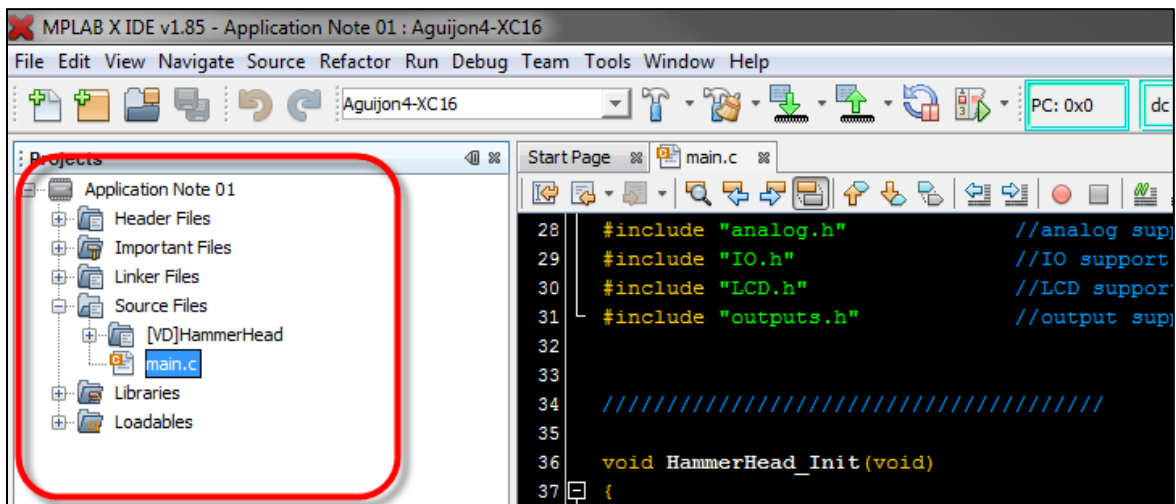
* Ver archivo **Cambios REV4.1**

4. Abrir MPLAB X® y cargar el archivo del proyecto: **Application Note 27.X**



Haz 'clic' aquí y
abre el proyecto

5. Abrir el archivo **main.c**



6. Ir a la línea #50

Utilizaremos la siguiente función:

```
43      /*Inputs*/
44      ADC_Init();
45
46      /*LCD*/
47      LCD_Init(LCD_MODE_1);
48
49      /*irDA*/
50      IR_init();
51
52  #if defined(USE_LCD_EXTRA_FEATURES)
53      LCD_BacklightFadeIn();
54  #else
55      LCD_BacklightSet(BLIGHT_LVL_4);
56  #endif
57
```

IR_init ();

- *Función que inicializa el puerto de entrada irDA, este configuran tanto la velocidad de transmisión (1200 Baudios), como el pin de entrada (RP14, Puerto de expansión 2); Dicha función debe ser llamada antes que cualquier otra relacionada con el uso de irDA*

7. Ir a la línea #79

Utilizaremos la siguiente función:

```
72 LCD_PutStr(2,0,"Application Note 27",FALSE);
73
74 for(;;){
75     for(i=1;i<=5;i++)        //Configure Buttons from 1 to 5
76     {
77         //Create a strings for the LCD//
79         sprintf(lcdMSG,"Press Button %i ",i);
80         LCD_PutStr(1,0,lcdMSG,TRUE);
81
82         while (SW_Read() !=1) {} //Just sit here until the button is pressed
83         Button_Config(i);        //Configure Button
84     }
85
86     for(;;)
```

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.

8. Ir a la línea #80

Utilizaremos la siguiente función:

```
73
74     for(;;){
75
76         for(i=1;i<=5;i++)      //Configure Buttons from 1 to 5
77         {
78             //Create a strings for the LCD//
79             sprintf(lcdMSG,"Press Button %i ",i);
80             LCD_PutStr(1,0,lcdMSG,TRUE);
81
82             while (SW_Read() !=1) {} //Just sit here until the button is pressed
83             Button_Config(i);        //Configure Button
84         }
85
86     for(;;)
87     {
```

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).

9. Ir a la línea #82.

Vamos a utilizar la siguiente función:

```
75
76     for(i=1;i<=5;i++)          //Configure Buttons from 1 to 5
77     {
78         //Create a strings for the LCD//
79         sprintf(lcdMSG,"Press Button %i ",i);
80         LCD_PutStr(1,0,lcdMSG,TRUE);
81
82         while (SW_Read() !=1) {} //Just sit here until the button is pressed
83         Button_Config(i);        //Configure Button
84     }
85
86     for(;;)
87     {
88         if(Button_compare()==1) //If button 1 is pressed, Then
89             LEDport_Set(3,ON);  //Show 3 in LEDport
```

SW_Read ();

- Función que lee el puerto de PUSH BUTTONS;
Devuelve un valor equivalente al Push-Button presionado (carácter del 1 al 4)

10. Ir a la línea #83

Utilizaremos la siguiente función:

```
76     for(i=1;i<=5;i++)          //Configure Buttons from 1 to 5
77     {
78         //Create a strings for the LCD//
79         sprintf(lcdMSG,"Press Button %i ",i);
80         LCD_PutStr(1,0,lcdMSG,TRUE);
81
82         while (SW_Read() !=1) {} //Just sit here until the button is pressed
83         Button_Config(i);        //Configure Button
84     }
85
86     for(;;)
87     {
88         if(Button_compare()==1) //If button 1 is pressed, Then
89             LEDport_Set(3,ON);  //Show 3 in LEDport
90         if(Button_compare()==2) //If button 2 is pressed, Then..
```

Button_Config (int num);

- Función que almacena en una localidad de memoria, el código enviado por cualquier tecla de nuestro control; donde:
Int num= Localidad donde se desea almacenar el código enviado. (Numero entero del 1 al 10.)

- **Button_Config**

Esta función almacena el último arreglo de bytes cargado en el buffer de entrada, es decir almacenara el código enviado por el último botón del control que haya sido presionado, en la localidad indicada.

11. Ir a la línea #88

Utilizaremos la siguiente función:

```
81
82     while (SW_Read() !=1) {} //Just sit here until the button is pr
83     Button_Config(i);        //Configure Button
84 }
85
86 for(;;)
87 {
88     if(Button_compare()==1) //If button 1 is pressed, Then
89         LEDport_Set(3,ON);   //Show 3 in LEDport
90     if(Button_compare()==2) //If button 2 is pressed, Then..
91         LEDport_Set(12,ON);
92     if(Button_compare()==3) //If button 3 is pressed, Then..
93         LEDport_Set(48,ON);
94     if(Button_compare()==4) //If button 4 is pressed, Then..
95         LEDport_Set(192,ON);
```

Button_Compare ();

- Función que compara el arreglo del buffer de entrada del puerto irDA, con los arreglos previamente almacenados mediante la función **Button_config**; donde:
Devuelve un valor proporcional al número de la localidad, donde anteriormente se almaceno el código de entrada (Valor entero del 1 al 10)

12. Ir a la línea #89

Utilizaremos la siguiente función:

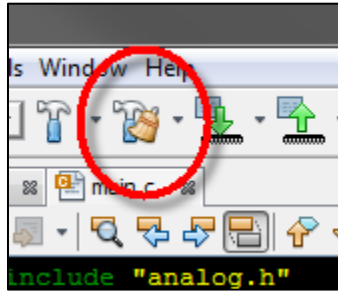
```
82         while (SW_Read() !=1) {} //Just sit here until the button is pressed
83         Button_Config(i);         //Configure Button
84     }
85
86     for(;;)
87     {
88         if(Button_compare()==1)    //If button 1 is pressed, Then
89             LEDport_Set(3,ON);      //Show 3 in LEDport
90         if(Button_compare()==2)    //If button 2 is pressed, Then..
91             LEDport_Set(12,ON);
92         if(Button_compare()==3)    //If button 3 is pressed, Then..
93             LEDport_Set(48,ON);
94         if(Button_compare()==4)    //If button 4 is pressed, Then..
95             LEDport_Set(192,ON);
96         if(Button_compare()==5)    //If button 5 is pressed, Then..
```

LEDport_Set (char value, BOOL invert);

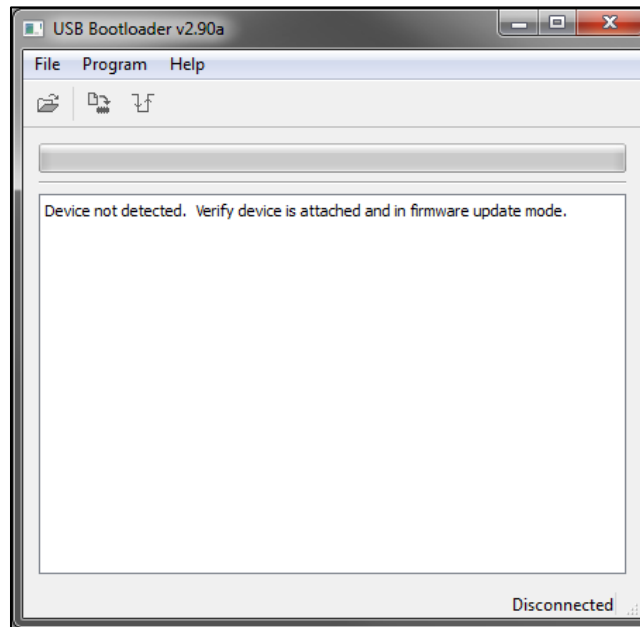
- Función que enciende y apaga el puerto de LEDs; donde:
Char value = Número que queremos Encender (Valor entero del 1 al 255.)
BOOL invert = Estado del puerto de LEDs (ON = Encendido, OFF = apagado).

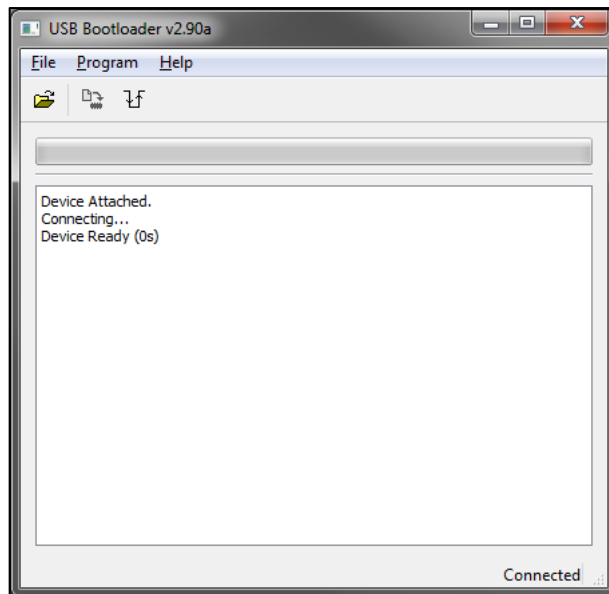
13. Compilar y programar

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



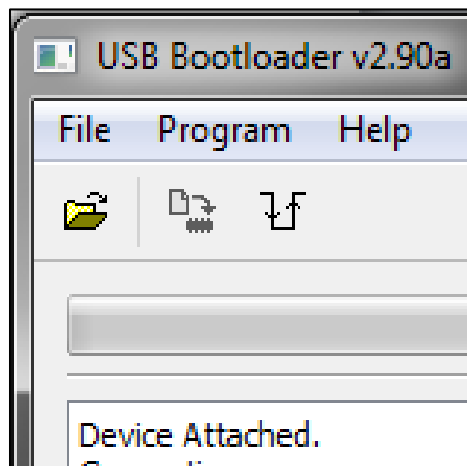
Cuando aparezca la ventana del bootloader, presione el Push-Button número 1 y conecte la fuente de voltaje o encienda el Aguijón y mantenga el PB1 presionado hasta que los LEDs empiecen a parpadear.





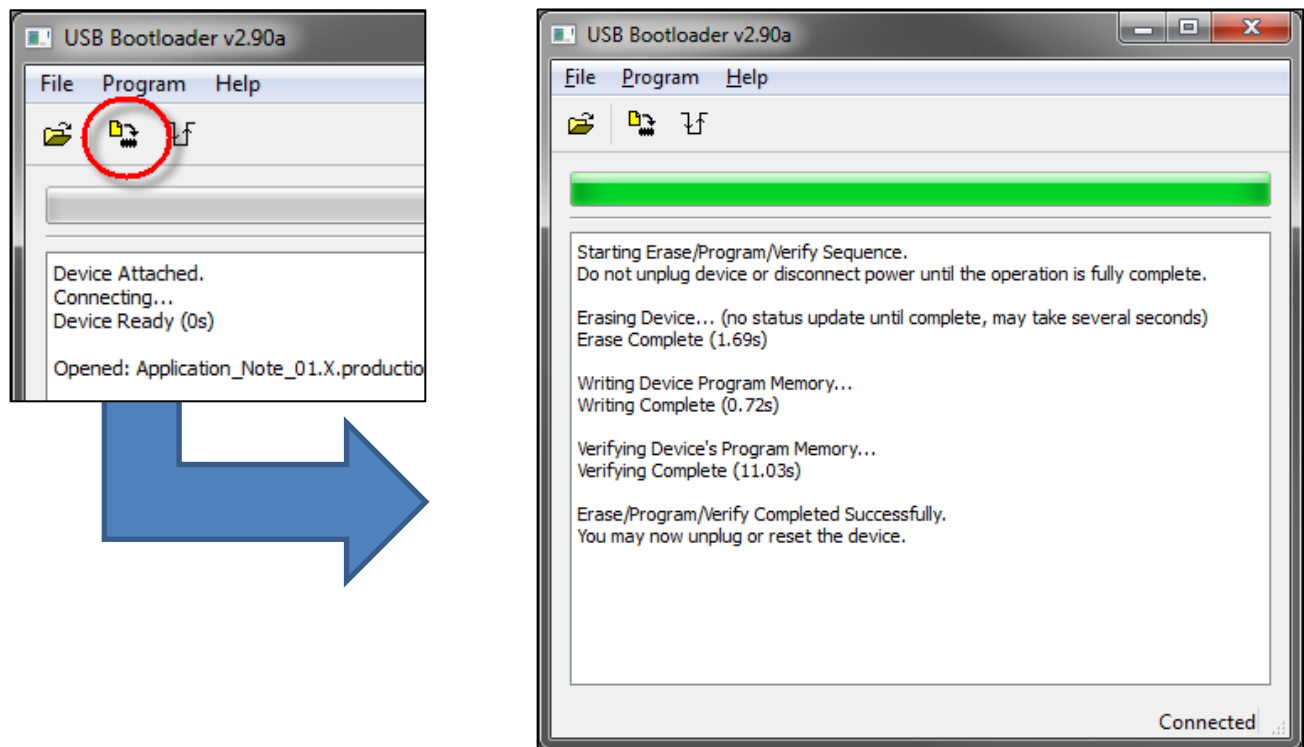
La ventana del Bootloader indicará la conexión establecida con el aguijón:

14. Hacer Clic en Abrir y Cargar el archivo **Application Note 27.X.production.hex**

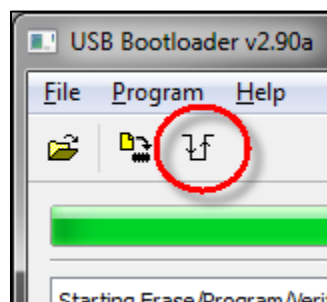


El archivo, depende de la plataforma de hardware.

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.

15. Para verificar el funcionamiento del programa verifique al iniciar el programa se almacenen 5 diferentes botones de nuestro control remoto, para esto hay que presionar cada botón uno a uno, y seguido de cada uno de estos, presionar el PUSH-BUTTON 1 de nuestro Aguijon, seguido de esto y teniendo ya almacenado los 5 botones, presionar cualquiera de estos, para realizar cambios en el puerto de leds.