



*Aguijón*

---

*Notas de aplicación*

## **Nota de aplicación 28:**

### **Read file from the usb drive & send for RS232**

#### **Descripción:**

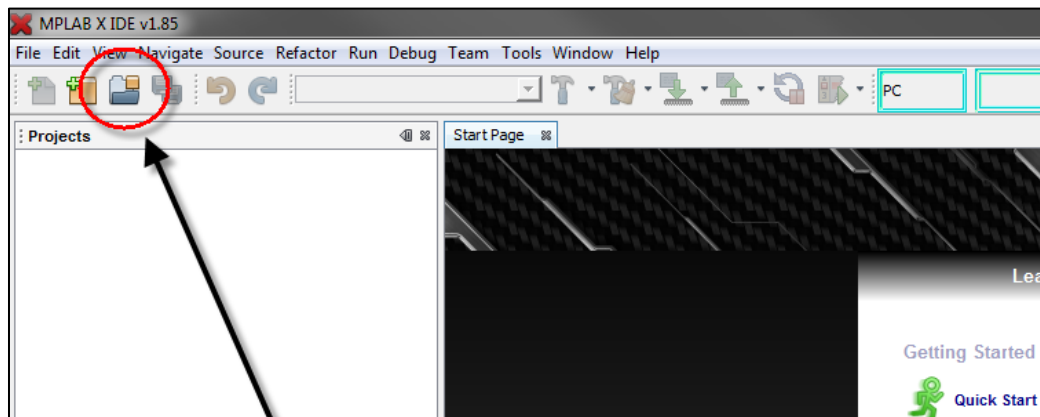
Leer un mensaje contenido en un archivo de texto ubicado en una memoria USB, para luego ser enviado por el puerto serial RS232.

#### **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. Memoria usb 2.0

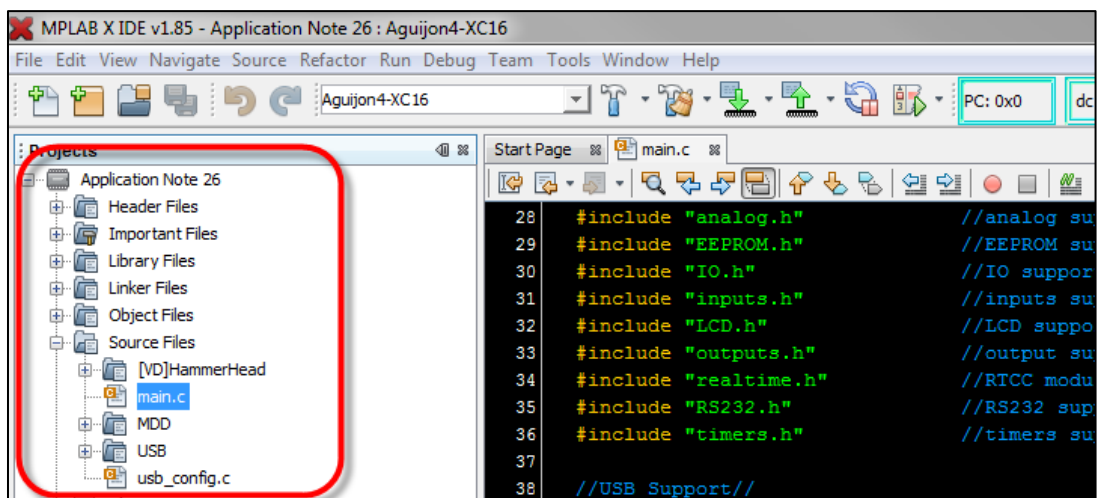
## Pasos:

1. Abrir MPLAB X® y cargar el archivo del proyecto: **Application Note 28.X**



Haz 'clic' aquí y  
abre el proyecto

2. Abrir el archivo *main.c*



3. Ir a la línea #150

Utilizaremos la siguiente función:

```
143 LCD_PutStr(2,0,"Application Note 25",FALSE);
144 delayms(1000);
145 LCD_PutStr(1,0,"Aguijon USB MSD",TRUE);
146 LCD_PutStr(2,0,"Insert flash drive",FALSE);
147
148 for(;;){
149     //USB stack process function
150     USBTasks();
151
152     //if thumbdrive is plugged in
153     if(USBHostMSDSCSIMediaDetect()){
154         InsertChime();
155         //now a device is attached
156
157         //See if the device is attached and in the right format
```

### **USBTasks ();**

- Esta función es la herramienta necesaria para configurar e inicializar nuestro dispositivo USB, esta debe ser llamada antes que cualquier otra función relacionada con el puerto USB

4. Ir a la línea #153

Utilizaremos la siguiente función:

```
146 LCD_PutStr(2,0,"Insert flash drive",FALSE);
147
148 for(;;){
149     //USB stack process function
150     USBTasks();
151
152     //if thumbdrive is plugged in
153     if(USBHostMSDSCSIMediaDetect()){
154         InsertChime();
155         //now a device is attached
156
157         //See if the device is attached and in the right format
158         if(FSInit()){
159             //Opening a file in mode "r"
160             //The device must have the file with the same name
```

**USBHostMSDSCSIMediaDetect ( );**

- Función que determina si un dispositivo de almacenamiento masivo está conectado y listo para utilizar; donde:  
Devuelve un valor booleano dependiendo del estado del dispositivo  
(TRUE= MSD presente y listo, FALSE = MSD no está presente o no está listo).

5. Ir a la línea #158.

Utilizaremos la siguiente función:

```
151
152     //if thumbdrive is plugged in
153     if(USBHostMSDSCSIMediaDetect()){
154         InsertChime();
155         //now a device is attached
156
157         //See if the device is attached and in the right format
158         if(FSInit()){
159             //Opening a file in mode "r"
160             //The device must have the file with the same name
161             myFile = FSfopen("msg.txt","r");
162
163             //Get file size
164             file_size=File_GetSize(myFile);
165
```

### **FSInit ();**

- Esta función inicializa el dispositivo físico que debe ser conectado al microcontrolador ;  
donde:  
Devuelve un valor booleano dependiendo del estado de la inicialización  
(TRUE= Inicialización exitosa, FALSE = Inicialización no exitosa).

6. Ir a la línea #161

Utilizaremos la siguiente función:

```
154         InsertChime();
155         //now a device is attached
156
157         //See if the device is attached and in the right format
158         if(FSInit()){
159             //Opening a file in mode "r"
160             //The device must have the file with the same name
161             myFile = FSfopen("msg.txt", "r");
162
163             //Get file size
164             file_size=File_GetSize(myFile);
165
166             for(i=0;i<file_size;i+=CHUNK_SIZE)
167             {
168                 //Change the current position in a file
```

**FSfopen (char \* fileName, char \*mode);**

- Función que edita un archivo ya sea este nuevo o existente; donde:  
**Char \*FileName** = El nombre del archivo a abrir o crear. (Cadena de caracteres, incluyendo extensión del archivo.)  
**Char \*mode** = Modo del uso que se le desea dar al archivo; donde:
  - W:** Crea un archivo nuevo o reemplaza uno existente
  - R:** Lee datos de un archivo existente
  - A:** Agrega datos a un archivo existente
  - W+:** Crea un archivo nuevo o reemplaza uno existente (También habilita lectura)
  - R+:** Lee datos de un archivo existente (También habilita escritura)
  - A+:** Agrega datos a un archivo existente (También habilita lectura)

7. Ir a la línea #164

Utilizaremos la siguiente función:

```
157 //See if the device is attached and in the right format
158 if(FSInit()){
159     //Opening a file in mode "r"
160     //The device must have the file with the same name
161     myFile = FSfopen("msg.txt","r");
162
163     //Get file size
164     file_size=File_GetSize(myFile);
165
166     for(i=0;i<file_size;i+=CHUNK_SIZE)
167     {
168         //Change the current position in a file
169         FSfseek(myFile,i,SEEK_SET);
170
171         //Read some data from the file
```

**File\_GetSize (FSFILE \*inFile);**

- Función que obtiene la longitud del contenido del archivo ; donde:  
**FSFILE \*inFile** = Apuntador a la estructura del archivo.  
Devuelve un valor entero proporcional a la longitud del contenido, de un archivo específico.



8. Ir a la línea #169.

Utilizaremos la siguiente función:

```
162
163         //Get file size
164         file_size=File_GetSize(myFile);
165
166         for(i=0;i<file_size;i+=CHUNK_SIZE)
167         {
168             //Change the current position in a file
169             FSfseek(myFile,i,SEEK_SET);
170
171             //Read some data from the file
172             units_read = FSfread(rs232MSG,1,CHUNK_SIZE,myFile);
173
174             for(unit=0;unit<units_read;unit++)
175             {
176                 //wait if the buffer is full
```

**FSfseek (FSFILE\*stream,long offset, int whence);**

- Esta función cambia la posición actual dentro de un archivo ; Donde:

**FSFILE\*stream** = Apuntador a la estructura del archivo.

**Long offset** = Offset de la ubicación de la base.

**Int whence** =

**SEEK\_SET**= Desde el inicio del archivo

**SEEK\_CUR**=Desde la posición actual

**SEEK\_END**=Desde el final del archivo

9. Ir a la línea #172

Utilizaremos la siguiente función:

```
165
166     for(i=0;i<file_size;i+=CHUNK_SIZE)
167     {
168         //Change the current position in a file
169         Fseek(myFile,i,SEEK_SET);
170
171         //Read some data from the file
172         units_read = FSfread(rs232MSG,1,CHUNK_SIZE,myFile);
173
174         for(unit=0;unit<units_read;unit++)
175         {
176             //wait if the buffer is full
177             while(TxBufferFull());
178             //Create a strings for the RS232//
179             RS232_Send(rs232MSG[unit]);
```

**FSfread (void \*ptr, unsigned long size, unsigned long n, FSFILE \*stream);**

- Función que lee datos de algún archivo específico; donde:  
**\*ptr** = Buffer de destino para los bytes leídos.  
**Size** = Tamaño de las unidades en bytes  
**n** = Número de unidades a transferir.  
**Stream** = Apuntador a la estructura del archivo  
Devuelve un valor proporcional al número de bytes leídos.

10. Ir a la línea #177

Utilizaremos la siguiente función:

```
170
171      //Read some data from the file
172      units_read = FSfread(rs232MSG,1,CHUNK_SIZE,myFile);
173
174      for(unit=0;unit<units_read;unit++)
175      {
176          //wait if the buffer is full
177          while(TxBufferFull());
178          //Create a strings for the RS232//
179          RS232_Send(rs232MSG[unit]);
180      }
181
182
183      //Always make sure to close the file.
184      FSfclose(myFile);
```

### **TxBufferFull ( );**

- Función que revisa el estado del buffer de transmisión; donde:  
Devuelve un valor entero dependiendo del estado del buffer. (1=el buffer está lleno, 0=el buffer está despejado.)

11. Ir a la línea #179

Utilizaremos la siguiente función:

```
172         units_read = FSfread(rs232MSG,1,CHUNK_SIZE,myFile);
173
174         for(unit=0;unit<units_read;unit++)
175         {
176             //wait if the buffer is full
177             while(TxBufferFull());
178             //Create a strings for the RS232//
179             RS232_Send(rs232MSG[unit]);
180         }
181     }
182
183     //Always make sure to close the file.
184     FSfclose(myFile);
185
186     //Just sit here until the device is removed.
```

**RS232\_Send (int data);**

- Función que envía datos al buffer de transmisión del puerto RS232; donde:  
**Int data** = Datos a transmitir.

12. Ir a la línea #184

Utilizaremos la siguiente función:

```
177         while(TxBufferFull());
178         //Create a strings for the RS232//
179         RS232_Send(rs232MSG[unit]);
180     }
181 }
182
183 //Always make sure to close the file.
184 FSfclose(myFile);
185
186 //Just sit here until the device is removed.
187 LCD_PutStr(1,0,"DONE!",TRUE);
188 LCD_PutStr(2,0,"Safe to remove USB",FALSE);
189 while(USBHostMSDSCSIMediaDetect()){
190     USBTasks();
191 }
```

**FSfclose (FSFILE \*fo);**

- Función que Actualiza la información y cierra el archivo seleccionado; donde:  
**FSFILE \*fo** = Apuntador a la estructura del archivo a cerrar.

13. Ir a la línea #187

Utilizaremos la siguiente función:

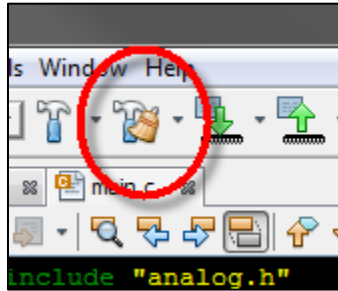
```
180         }
181     }
182
183     //Always make sure to close the file.
184     FSfclose(myFile);
185
186     //Just sit here until the device is removed.
187     LCD_PutStr(1,0,"DONE!",TRUE);
188     LCD_PutStr(2,0,"Safe to remove USB",FALSE);
189     while(USBHostMSDSCSIMediaDetect()){
190         USBTasks();
191     }
192     RemoveChime();
193     //Create a strings for the LCD//
194     LCD_PutStr(1,0,"Aguijon USB MSD",TRUE);
```

**LCD\_PutStr (int y, int x, char \*msg, BOOL clear);**

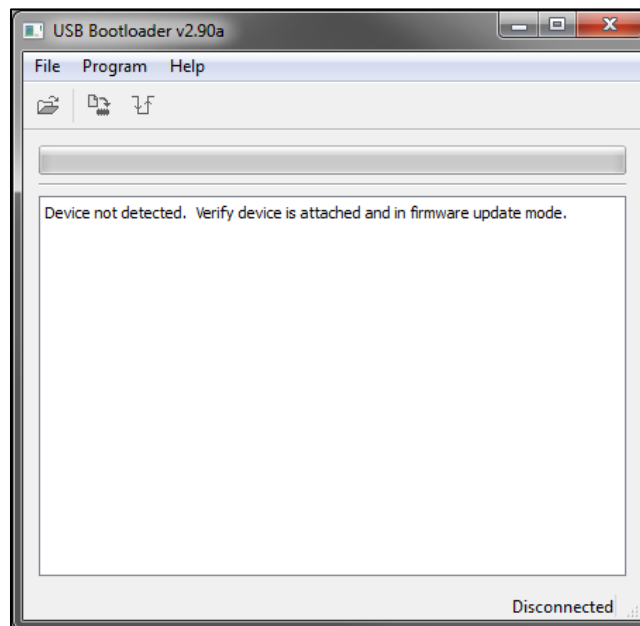
- Función que muestra una cadena de caracteres en la pantalla LCD; donde:  
**Int y** = Coordenada en y (Valor entero del 1 al 2.)  
**Int X** = Coordenada en x(Valor entero del 0 al 20.)  
**Char \*msg** = Cadena de caracteres (De 0 a 20 caracteres)  
**BOOL clear** = Determina si se borra la pantalla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

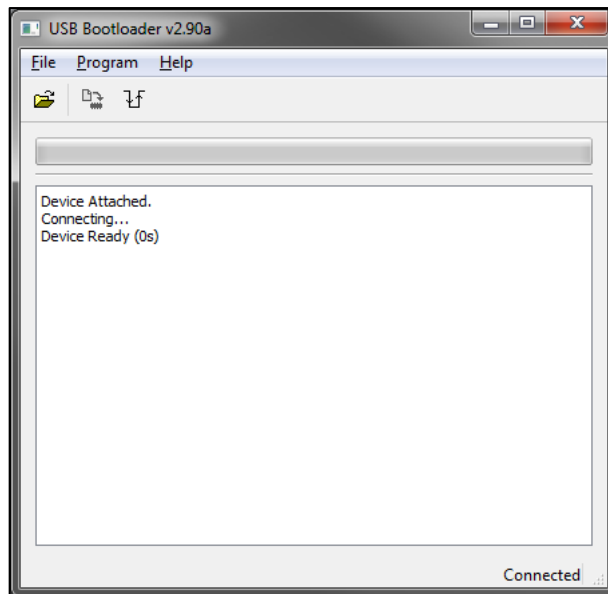
#### 14. Compilar y programar

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



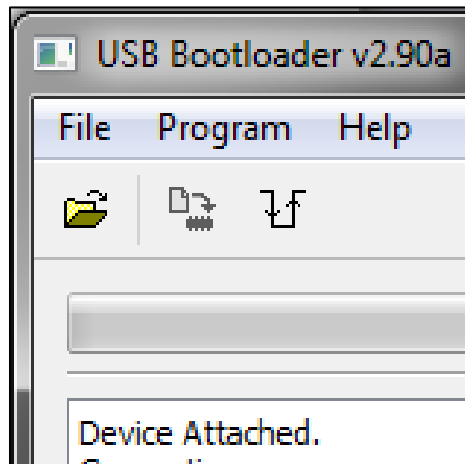
Cuando aparezca la ventana del bootloader, presione el Push-Button número 1 y conecte la fuente de voltaje o encienda el Aguijón y mantenga el PB1 presionado hasta que los LEDs empiecen a parpadear.





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

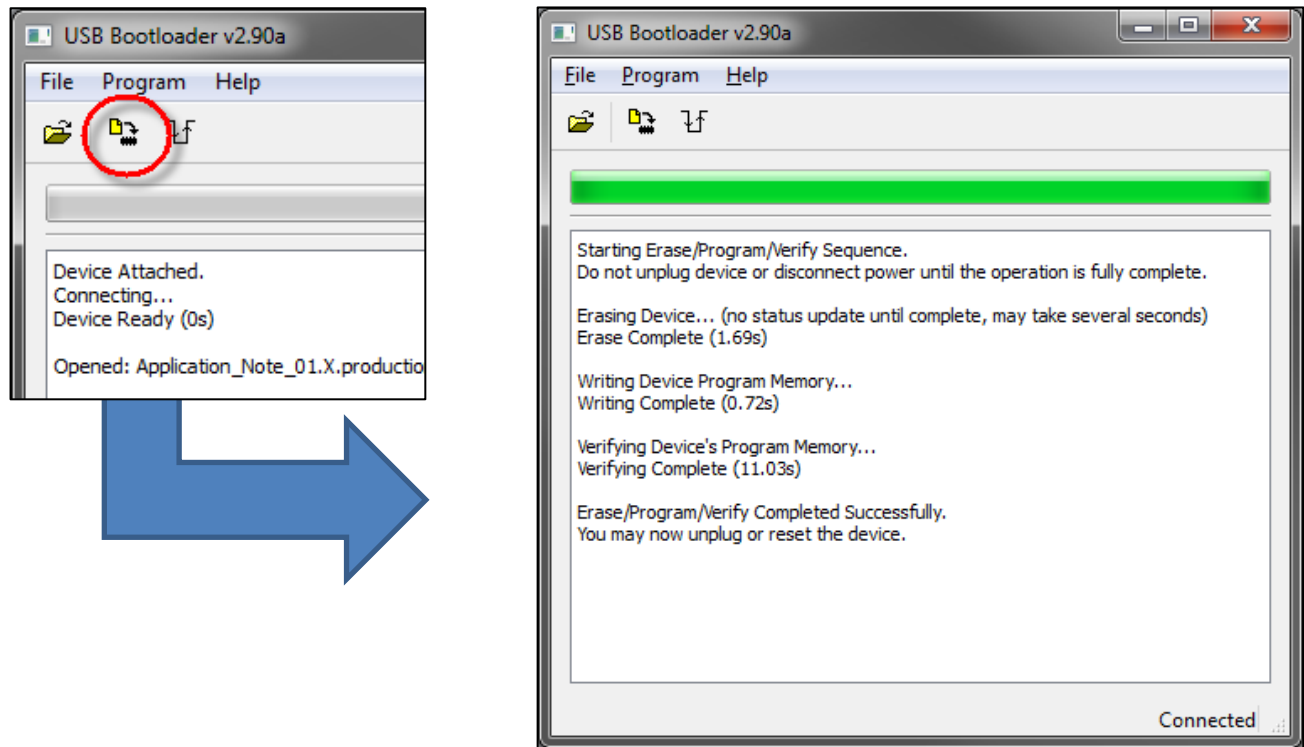
15. Hacer Clic en Abrir y Cargar el archivo **Application Note 28.X.production.hex**



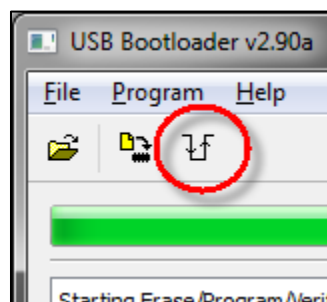
El archivo, depende de la plataforma de hardware.



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.

16. Para verificar el funcionamiento del programa verifique que al conectar un dispositivo de almacenamiento USB, el cual contenga un archivo de texto llamado "MSG.TXT", el texto de este sea enviado por el puerto RS232.