

Conmutador de cambio de polarización por ordenador

Por Jordi Quintero, EA3GCV



Cuando monté mi instalación para satélites LEO, adquirí los relés de conmutación de polarización para VHF y UHF de la marca Wimo, para poder sacar el máximo partido a mi instalación con las antenas X-Quad de la misma marca. Estos relés no incluyen ninguna caja de conmutación para la estación e “instan” al cliente a construirse uno. No es algo complicado para los que estamos acostumbrados al “cacharreo”, pero considero deberían incluirla dado el elevado coste de estos relés (aunque fuera pequeña y sencilla). Para fortuna mía, mi amigo Mariano EA3EDU me obsequió un conmutador excelente realizado por EA3KP y así evité que tuviera que construirme uno. Este conmutador (bautizado como “SwitchBox” por el autor) tiene un bonito diseño, con interruptor principal de encendido, conmutadores independientes para VHF y UHF, leds indicadores de cada polarización y dispuesto en una caja amplia con generosos botones.



Conmutador realizado por EA3KP

Desde el año 2015, el 90% de mi operativa en radio suele ser en remoto. En las revistas de URE de octubre a noviembre de 2015 más un anexo que salió en junio de 2016, publiqué una serie de artículos de mi control remoto “low cost” o de bajo coste. En la actualidad, he simplificado muchísimo la parte de encendido remoto de la estación. Ya no es necesaria la complicada configuración a través de WOL (Wake on LAN) y DDNS dinámicas que expliqué con detalle en las dos primeras partes de mi artículo. La tecnología actual, muy avanzada y económica, permite hacerlo de forma muy sencilla usando enchufes WIFI. Utilizo sus propias aplicaciones para encender y apagar todo, donde puedo crear escenas para actuar en diversos enchufes a la vez e incluso pongo toda la estación en marcha a través de la voz con Alexa. Un “pijada” pero muy cómodo cuando llego a casa ;-)

Obviamente tengo toda la estación controlada por ordenador, tanto rotores como equipos. Para trabajar satélites utilizo el programa SATPC32 que es una auténtica maravilla ya que el control del Doppler es prácticamente automático. Explico todo esto porque para poder trabajar satélites en remoto como si estuviera delante de la estación, tenía que encontrar una solución para poder conmutar los relés de cambio de polarización. Sino tendría que dejarlos en una posición fija y eso sería un desperdicio, no sólo por la inversión realizada sino también para sacar el máximo partido a cada pase.

Para ello, he diseñado un sistema por Arduino y he desarrollado un programa específico para Windows que permite controlarlo por ordenador. En mi caso, he integrado la placa de Arduino y módulo de relés dentro del "SwitchBox" de EA3KP de forma que la caja sirva tanto para operativa manual como de control por ordenador a través del programa. En el esquema general de conexiones detallo el montaje sólo para para controlarlo por ordenador, sin la parte manual de conmutadores, ya que saldrá todo más económico y sencillo de montar.

Antes de empezar a describir el proyecto, y para tranquilizar al lector, quiero indicar que no necesita saber programación ni ser un experto en montajes ni tan sólo saber cómo funciona Arduino. Siguiendo las instrucciones que indico, cualquier persona con unos mínimos conocimientos puede hacerlo. El código para programar el Arduino se lista más adelante y puedo facilitarlo electrónicamente a quien me lo pida. También explicaré cómo programar el Arduino con el código facilitado. El programa que he creado se puede bajar gratuitamente desde internet.

Qué necesitamos

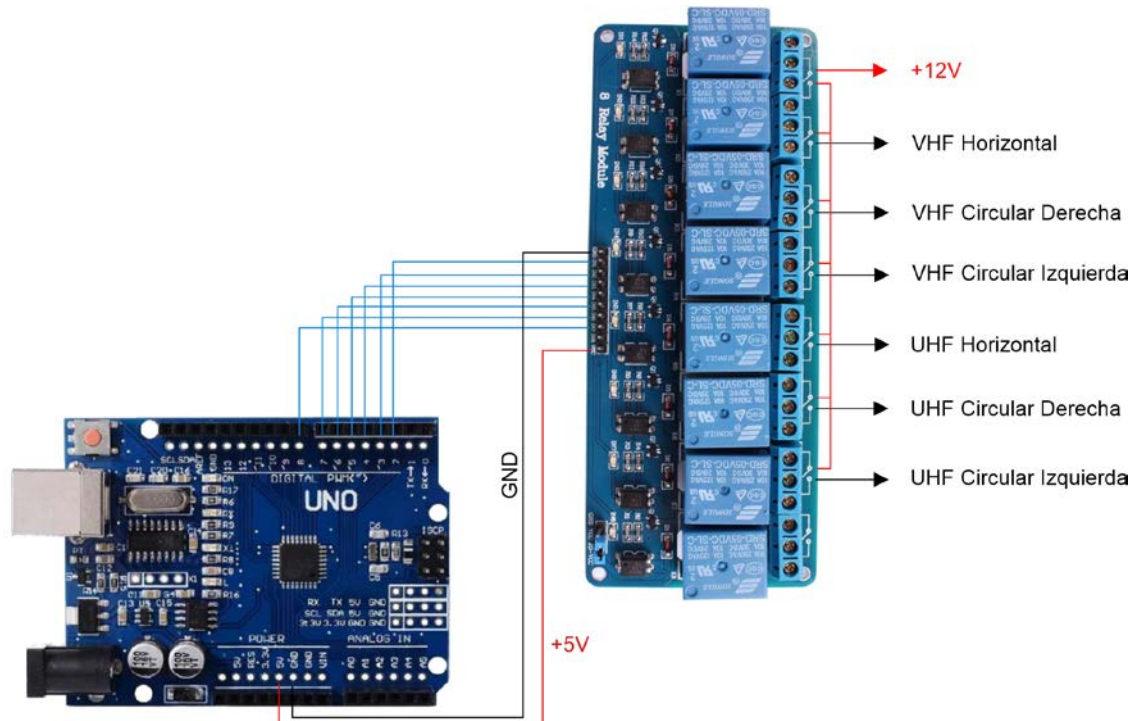
- Placa Arduino Uno
- Módulo de relés de 8 canales de 5V

No hace falta comprar marcas genuinas, cualquier Arduino Uno genérico compatible sirve. El coste de estas placas varía dependiendo la marca y dónde se compre. Si se compran placas compatibles en Ali Express o similar las dos placas pueden salir por unos 10€ o incluso menos. En Amazon sobre 20-25€, ya que sólo venden ELEGOO que es una marca conocida. Pero compre la que compre, este proyecto funcionará igual a un coste realmente bajo. Sale mucho más barato que el coste de realizar una caja de conmutación manual. Aconsejo igualmente poner las placas dentro de alguna caja.

Funcionamiento de los relés de polarización Wimo

La polarización será en vertical si no se aplica tensión a ningún relé. Para el resto de polarizaciones hay que aplicar 12 voltios a cada respectivo relé. Y la premisa es que no puede aplicarse tensión a dos relés a la vez. Este proyecto puede aplicarse a otros relés de cambio de polarización del mercado que funcionen del mismo modo.

Esquema general de conexiones



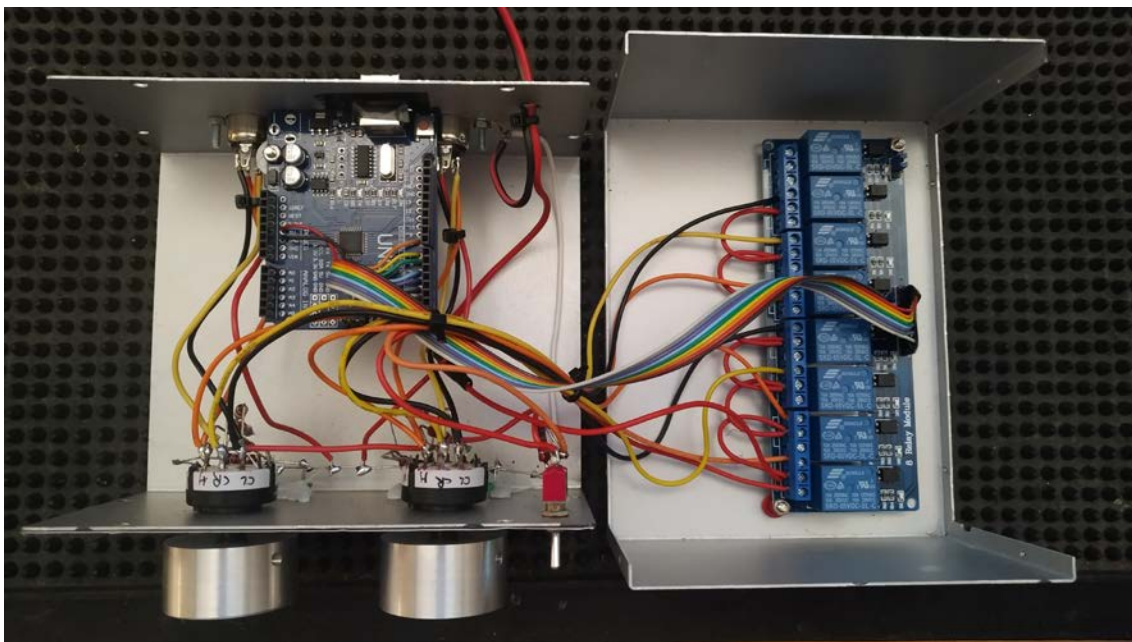
El conexionado entre los pines internos de ambas placas es el siguiente:

ARDUINO	Módulo Relés
2	IN1
3	IN2
4	IN3
5	IN4
6	IN5
7	IN6
8	IN7
5V	VCC
GND	GND

Depende el modelo del módulo de relés, los pines pueden estar marcados como CH1, CH2, etc. en lugar de IN1, IN2... Hay que conectar 12 V de una fuente de alimentación externa a la patilla común del primer relé que actuará de interruptor ON/OFF y dará alimentación común al resto de relés. Las conexiones de salida de cada relé indican el conexionado con los relés de polarización. El octavo relé no se utiliza.

El módulo de relés lleva un pequeño puente que por defecto está entre la posición JD-VCC y VCC. Esto indica que la alimentación de las bobinas de los relés se realizará

por la propia alimentación del Arduino a través del puerto USB (no se utilizará el conector de alimentación que lleva el Arduino). Hay que saber que, como máximo, sólo habrá 3 relés alimentados a la vez: el interruptor ON, uno para la polarización de VHF y otro para UHF. Al ir cambiando de polarización, se van activando y desactivando los relés. El sistema ya está pensado para que sea imposible activar 2 relés de polarización a la vez. Dependiendo del módulo, el consumo rondará sobre 200-250ma, que es la mitad o menos de la corriente que ofrece el puerto USB (500ma). Así que los relés pueden alimentarse directamente del puerto USB sin necesidad de utilizar otro alimentador específico para ello. Hay que saber que este tipo de conexión directa tiene un pequeño inconveniente y es que se pierde el aislamiento que ofrecen los optoacopladores del circuito. Si hubiese algún problema en las cargas de los relés, el circuito Arduino podría llegar a dañarse. Dada que la tensión de carga de los relés es de 12V y de poca intensidad, considero altamente improbable que pueda dañarse el Arduino y he prescindido de utilizar otra fuente, simplificando el conexionado. Pero si el lector lo prefiere, entonces debería quitar el puente y conectar una tensión de 5V desde otra fuente de alimentación a la patilla JD-VCC.



Resultado final del Arduino y módulo de relés dentro del conmutador de EA3KP

Descripción del código para el Arduino

El código se basa en unos sencillos comandos que envía el programa de Windows al Arduino cuando se pulsa el botón de encendido o cada uno de los botones de cambio de polarización. Cuando el Arduino reconoce el código, activa y desactiva los relés correspondientes y envía una respuesta al programa para que éste sepa que la acción se ha completado con éxito.

Los lectores que dominen el código, observarán un detalle sobre el estado de los pines de salida de los relés que quizá les pueda parecer extraño. Y es que, en los módulos de relés, la lógica está invertida. Por lo tanto, el estado bajo en los pines lo que hace realmente es activar los relés. Y el estado alto lo desactiva. Por eso el estado inicial de los pines en el código es alto, para dejar los relés en estado de reposo. Y se ponen en bajo cuando se necesita activar un relé determinado.

El código para el Arduino

```
/*
 * Conmutador de polarización por ordenador
 * por Jordi Quintero, EA3GCV
 *
 * Lista de comandos de la aplicación:
 *
 * 0 = Power OFF
 * 1 = Power ON
 * 2 = VHF Vertical
 * 3 = VHF Horizontal
 * 4 = VHF Circular Derecha
 * 5 = VHF Circular Izquierda
 * 6 = UHF Vertical
 * 7 = UHF Horizontal
 * 8 = UHF Circular Derecha
 * 9 = UHF Circular Izquierda
 *
 */

// Pins
const byte PWRSw = 2;
const byte VHFH = 3;
const byte VHF CR = 4;
const byte VHFCL = 5;
const byte UHFH = 6;
const byte UHF CR = 7;
const byte UHFCL = 8;

// Configuración
void setup() {
// Asignar pines
pinMode(PWRSw, OUTPUT);
pinMode(VHFH, OUTPUT);
pinMode(VHF CR, OUTPUT);
pinMode(VHFCL, OUTPUT);
pinMode(UHFH, OUTPUT);
pinMode(UHF CR, OUTPUT);
pinMode(UHFCL, OUTPUT);

// Asignar estado inicial de relés
digitalWrite(PWRSw, HIGH);
digitalWrite(VHFH, HIGH);
digitalWrite(VHF CR, HIGH);
digitalWrite(VHFCL, HIGH);
digitalWrite(UHFH, HIGH);
```

```

digitalWrite(UHFCCR, HIGH);
digitalWrite(UHFCL, HIGH);

// Inicializar Puerto serie
Serial.begin(9600);

}

void loop() {
  handleSerial();
}

void handleSerial() {
  if (Serial.available() > 0) {
    switch (Serial.read()) {
      case '*': // Conectado
        Serial.println("OK;");
        break;
      case '0': // Power OFF
        digitalWrite(PWRSw, HIGH);
        digitalWrite(VHFH, HIGH);
        digitalWrite(VHFCCR, HIGH);
        digitalWrite(VHFCL, HIGH);
        digitalWrite(UHFH, HIGH);
        digitalWrite(UHFCCR, HIGH);
        digitalWrite(UHFCL, HIGH);
        Serial.println("PWR_OFF;");
        break;
      case '1': // Power ON y asigna VHF y UHF Circular Derecha como selecci3n
        inicial
        digitalWrite(PWRSw, LOW);
        digitalWrite(VHFCCR, LOW);
        digitalWrite(UHFCCR, LOW);
        Serial.println("PWR_ON;");
        break;
      case '2': // VHF Vertical
        if (digitalRead(PWRSw) == LOW){
          digitalWrite(VHFH, HIGH);
          digitalWrite(VHFCCR, HIGH);
          digitalWrite(VHFCL, HIGH);
          Serial.println("VV;");
        }
        break;
      case '3': // VHF Horizontal
        if (digitalRead(PWRSw) == LOW){
          digitalWrite(VHFH, LOW);
          digitalWrite(VHFCCR, HIGH);
          digitalWrite(VHFCL, HIGH);
          Serial.println("VH;");
        }
        break;
      case '4': // VHF Circular Derecha
        if (digitalRead(PWRSw) == LOW){
          digitalWrite(VHFH, HIGH);
          digitalWrite(VHFCCR, LOW);

```

```

digitalWrite(VHFCL, HIGH);
Serial.println("VCR;");
}
break;
case '5': // VHF Circular Izquierda
if (digitalRead(PWRSw) == LOW){
digitalWrite(VHFH, HIGH);
digitalWrite(VHFCL, HIGH);
digitalWrite(VHFCL, LOW);
Serial.println("VCL;");
}
break;
case '6': // UHF Vertical
if (digitalRead(PWRSw) == LOW){
digitalWrite(UHFH, HIGH);
digitalWrite(UHFCL, HIGH);
digitalWrite(UHFCL, HIGH);
Serial.println("UV;");
}
break;
case '7': // UHF Horizontal
if (digitalRead(PWRSw) == LOW){
digitalWrite(UHFH, LOW);
digitalWrite(UHFCL, HIGH);
digitalWrite(UHFCL, HIGH);
Serial.println("UH;");
}
break;
case '8': // UHF Circular Derecha
if (digitalRead(PWRSw) == LOW){
digitalWrite(UHFH, HIGH);
digitalWrite(UHFCL, LOW);
digitalWrite(UHFCL, HIGH);
Serial.println("UCR;");
}
break;
case '9': // UHF Circular Izquierda
if (digitalRead(PWRSw) == LOW){
digitalWrite(UHFH, HIGH);
digitalWrite(UHFCL, HIGH);
digitalWrite(UHFCL, LOW);
Serial.println("UCL;");
}
break;
}
}
}
}

```


Programación del código en el Arduino

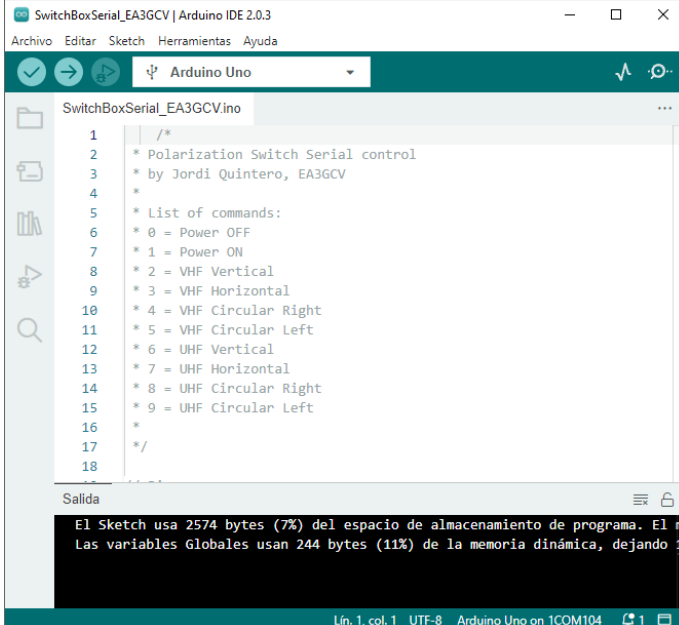
Primero de todo hay que bajar e instalar el Arduino IDE para Windows que se encuentra en esta página:

<https://wiki-content.arduino.cc/en/software>

Una vez instalado, conectar el cable USB a la placa Arduino y conectarlo en una ranura USB del ordenador. Si ha comprado una placa de marca oficial (como ELEGOO), el sistema lo detectará automáticamente y le pedirá para instalar los controladores. Si ha comprado un Arduino genérico, primero tendrá que buscar e instalar los controladores para esas placas genéricas ya que usan otro chip. Escriba en Google **driver CH340** y le saldrán enlaces para bajar los controladores. Una vez instalados, abra el administrador de dispositivos de Windows y vaya al apartado Puertos COM y LPT. Allí debe ver “Arduino Uno” (o “USB CH340” si es una placa genérica) más el número de puerto COM asignado. Anote el número de puerto.

Ahora abra el programa Arduino y en el menú Herramientas seleccione la placa “Arduino Uno”. Y en Puerto seleccione el puerto COM que ha anotado anteriormente. En la parte inferior del programa debe aparecer “Arduino Uno en COM x”.

Ahora borre las líneas de código que aparecen y ya puede escribir el código del apartado anterior. O si ha recibido el código en un fichero, vaya a **Fichero > Abrir** y seleccione el fichero recibido. Cuando haya finalizado, en la parte superior, pulse el primer botón empezando por la izquierda (el que tiene el signo de visto) para compilar el código. Si está todo correcto al final aparecerá Compilado. De lo contrario, revise bien el código porque seguramente habrá cometido algún fallo al teclearlo. Cuando ya esté compilado correctamente, pulse el segundo botón (el que tiene una flecha a la derecha) para subir el código al Arduino. Si todo ha ido bien saldrá la palabra *Subido*. ¡Ya está el Arduino programado y puede cerrar el programa!



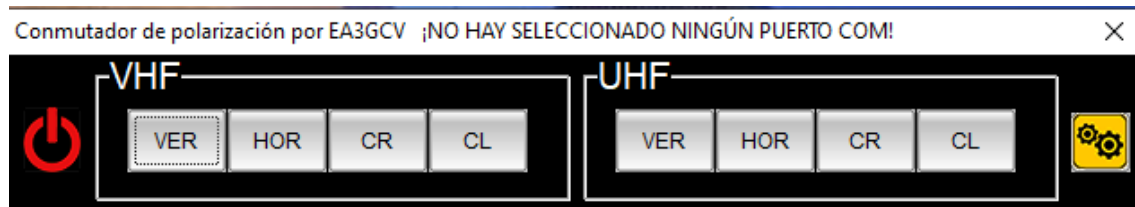
```
SwitchBoxSerial_EA3GCV | Arduino IDE 2.0.3
Archivo  Editar  Sketch  Herramientas  Ayuda
Arduino Uno
SwitchBoxSerial_EA3GCV.ino
1  /*
2  * Polarization Switch Serial control
3  * by Jordi Quintero, EA3GCV
4  *
5  * List of commands:
6  * 0 = Power OFF
7  * 1 = Power ON
8  * 2 = VHF Vertical
9  * 3 = VHF Horizontal
10 * 4 = VHF Circular Right
11 * 5 = VHF Circular Left
12 * 6 = UHF Vertical
13 * 7 = UHF Horizontal
14 * 8 = UHF Circular Right
15 * 9 = UHF Circular Left
16 *
17 */
18
Salida
El Sketch usa 2574 bytes (7%) del espacio de almacenamiento de programa. El m
Las variables Globales usan 244 bytes (11%) de la memoria dinámica, dejando 1
Lín. 1, col. 1  UTF-8  Arduino Uno on 1COM104
```

Programa Arduino con el código cargado y compilado

El programa de control para Windows

Este programa no requiere instalación. Sólo hay que bajar este fichero, guardarlo en una carpeta y ejecutarlo:

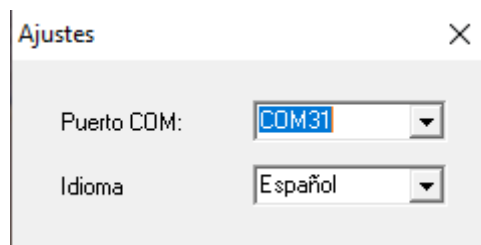
<https://www.swisslogforwindows.com/Download/SwitchBox.exe>



Programa de control para Windows

Para facilitar la operativa, este programa permanece por encima del resto de otros programas abiertos. En la parte del título de la ventana se muestra el estado de conexión con el puerto serie. Importante saber que cuando indique que está conectado al puerto X, no quiere decir que esté en marcha. Conectado significa que hay comunicación con el Arduino y puede empezar a operar con el programa.

La primera vez que se ejecute indicará que no se ha seleccionado ningún puerto COM (ver imagen). Hay que entrar en los ajustes (el botón amarillo) y allí hay que seleccionar el puerto COM que corresponda al Arduino y el idioma del programa (por defecto viene en inglés). El campo de puerto COM mostrará los puertos COM disponibles.



Ajustes del programa

Al pulsar el botón de encendido, se pondrá de color verde y, por defecto, se activa la polarización circular derecha en ambas bandas.



Programa una vez puesto en marcha

Ahora sólo es cuestión de ir pulsando los botones de polarización que desee para cada momento en cada banda.

Al cerrar el programa se crea un pequeño fichero de configuración donde se guarda el número de puerto COM, el idioma seleccionado y la última posición del programa en la pantalla.

Futuras mejoras del proyecto

Estoy pensando que sería interesante añadir un sistema de gestión automática del conmutador. Para ello quiero realizar un enlace con el programa Swisslog (el cual soy el desarrollador actual). Swisslog implementa la opción de lectura de la señal de S-Meter en el control del transceptor de Swisslog y puedo enviar esta información internamente al programa de conmutación. La idea sería añadir un botón AUTO para cada banda y un campo de intervalo de “consulta” en los ajustes. Cuando el usuario pulse el botón AUTO en la banda de recepción del satélite, el programa iría conmutando las diferentes polarizaciones, en los intervalos definidos por el usuario, comparando la señal. Si la señal es mayor en la nueva polarización, entonces dejaría seleccionada esta polarización hasta que pase el intervalo de consulta y vuelva a comprobar. Habría que poner también un campo en los ajustes para definir un valor de señal a partir del cual se empiecen a hacer las comprobaciones. Así se evitaría que estén los relés cambiando todo el tiempo sin que haya señales. A esto se añade el problema del QSB de la señal que, si es muy pronunciado, dificultará el proceso de detección de saber qué polaridad es la mejor. No sé, pienso que quizá podría ser una mejora a este sistema de conmutación por ordenador que, al menos, podría ser muy interesante experimentar. Cualquier sugerencia al respecto será bienvenida.

Quien esté interesado, puedo facilitar el código del Arduino en un fichero que puede cargarse directamente y así evitar posibles errores de escritura. Podéis pedírmelo a ea3qcv@castelldefels.net, así como hacerme llegar vuestras dudas o sugerencias de mejora de este proyecto.

Este artículo salió publicado originalmente en la revista de URE de Agosto / Septiembre de 2021. Lo he actualizado para publicarlo en AMSAT EA en febrero de 2023.