

DIMMER DIGITAL PARA LÁMPARA INCANDESCENTE CONTROLADO CON CONTROL REMOTO SONY.

Proyecto realizado por: Oscar Fernández Muñoz, estudiante de último semestre de electrónica. ITM

Revisado por: David Infante Sánchez

Instituto Tecnológico de Morelia

Nota. Para este proyecto se usaron el TRIAC y el receptor infrarrojo que se vende en la página, si ud. Consigue otro tenga en cuenta que su funcionamiento, terminales puede variar debiéndose hacer los ajustes necesarios

Introducción

En este proyecto se controla mediante un control remoto SONY la intensidad de una lámpara incandescente, lográndose bajar y subir la intensidad de un foco variando el ángulo de disparo de un tiristor, el control (los disparos del tiristores), así como la decodificación del código recibido del control remoto los hace un microcontrolador ATMEGA8.

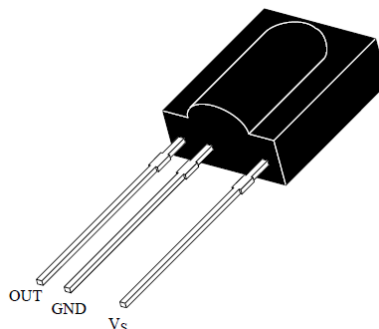
El receptor infrarrojo es el RPM-670CBR el cual puede adquirirse en la página de www.comunidadatmel.com, también pueden adquirirse los TRIACs y sus terminales se observan en las siguientes figuras.

Marco teórico

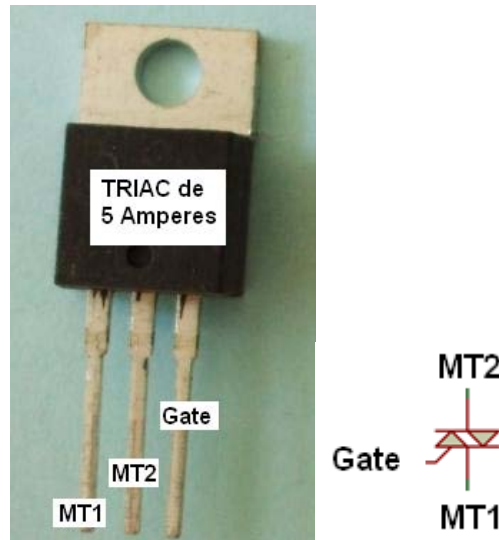
El protocolo de control que SONY utiliza es un protocolo llamado SIRC. El protocolo SIRC consta de un total de 13 bits. De los cuales, el primer bit, es un bit de inicio con duración de 2.4ms, 7 bits de comando, y 5 bits de dirección.

Para enviar un tren de pulsos, en el control se codifica la señal a 40kHz, y el receptor infrarrojo, se encarga de decodificar y filtrar la señal, de tal manera que su salida es en pulsos TTL para poder conectarse directamente a los pines del microcontrolador.

El detector infrarrojo usando es un RPM-670CBR que puede ser adquirido en www.comunidadatmel.com. En la siguiente imagen se muestra las terminales del receptor infrarrojo. El cual es vendido en el kit infrarrojo de la página.



En la siguiente imagen se muestra el TRIAC utilizado y las terminales del mismo.



Probando los comandos del control remoto

Conecte el receptor infrarrojo de la figura anterior de la siguiente manera: GND a tierra y VS a 5V, en OUT conecte un canal del osciloscopio, apunte el control remoto SONY al receptor infrarrojo y presione un botón del control remoto y verá una señal como se muestra en la figura 1:

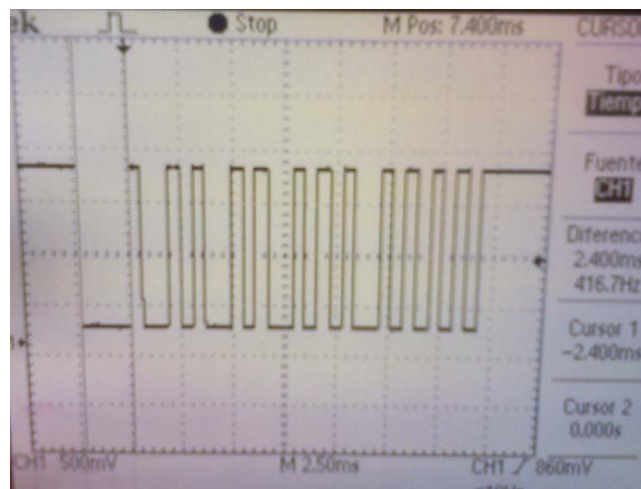


Figura1: Señal presionado el botón de POWER

Nota. Como comenté al inicio de este proyecto, otro receptor infrarrojo distinto al usado en este proyecto puede funcionar de manera distinta. Por ejemplo algunos tienen salida en colector abierto y requieren de resistencias de pull-up, o pueden tener salida invertida a este modelo en particular.

| Start | Command Code | | | | | | Device Code | | | | | |
|-------|--------------|----|----|----|----|----|--------------|----|----|----|----|----|
| S | D0 | D1 | D2 | D3 | D4 | D5 | D6 | C0 | C1 | C2 | C3 | C4 |
| 2.4mS | 1.2 or 0.6mS | | | | | | 1.2 or 0.6mS | | | | | |

Figura2: Protocolo SIRC de SONY

*NOTA: Observe que tanto los bits de comando (D0 a D6) y los bits de dirección (C0 a C4) están del bit menos significativo, al bit más significativo.

Para distinguir entre unos y ceros, en el comando y dirección se utiliza lo siguiente:

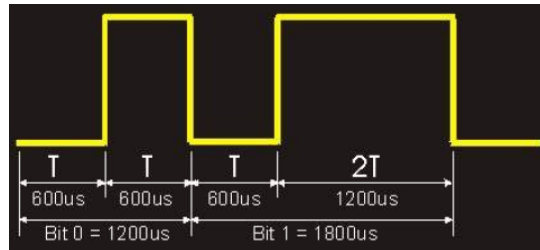


Figura3: Detección de bits

Después de un bit de inicio (2.4ms), siempre vendrá un pulso de banda de guarda de 600us, y la duración del siguiente pulso determinará si el bit es un cero, o es un uno. Si la duración del pulso es de 600us, es un cero, si la duración del pulso es de 1.2ms, entonces el bit es un 1.



Figura 4: Tren de pulsos de un control remoto SONY

Observando la figura 3:

Un bit 0 tiene duración de 1.2ms (600us de la banda de guarda+600us del bit0)



Un bit 1 tiene duración de 1.8ms (600us de la banda de guarda+1.2ms del bit1)



Dependiendo de la combinación de los bits de comando, y de los bits de dirección se forma una cadena de bits, y dependiendo del botón presionado en el control remoto, nos puede dar alguno de las siguientes tareas:

NOTA: los comandos pueden variar de acuerdo al control remoto utilizado y las funciones que tenga cada uno de ellos. Pero para las funciones básicas (cambio de volumen, cambio de canal, encendido, y números de tecla) son los mismos para todos los controles SONY.

| Comando | Función |
|---------|---------------|
| 0 | Digi key 1 |
| 1 | Digi key 2 |
| 2 | Digi key 3 |
| 3 | Digi key 4 |
| 4 | Digi key 5 |
| 5 | Digi key 6 |
| 6 | Digi key 7 |
| 7 | Digi key 8 |
| 8 | Digi key 9 |
| 9 | Digi key 0 |
| 16 | Channel + |
| 17 | Channel - |
| 18 | Volume + |
| 19 | Volume - |
| 20 | Mute |
| 21 | Power |
| 22 | Reset |
| 23 | Audio Mode |
| 24 | Contrast + |
| 25 | Contrast - |
| 26 | Coulor + |
| 27 | Coulor - |
| 30 | Brightness + |
| 31 | Brightness - |
| 38 | Balance Left |
| 39 | Balance Right |
| 47 | Standby |

| Address | Device |
|---------|-----------------------|
| 1 | TV |
| 2 | VCR 1 |
| 3 | VCR 2 |
| 6 | Laser Disc Unit |
| 12 | Surround Sound |
| 16 | Cassette deck / Tuner |
| 17 | CD Player |
| 18 | Equalizer |

Los bits de dirección son para cuando se requiere controlar uno o más dispositivos (TV, VCR, DVD) con un mismo control, es decir si quieres apagar un DVD, el comando siempre es igual, lo que cambia es la dirección, por ejemplo para TV la dirección es 1. Como se muestra en la siguiente tabla direcciones para controlar distintas unidades.

DESARROLLO

En cuanto al “variador” de intensidad para una lámpara incandescente, se necesita controlar el ángulo de disparo de un tiristor, para con esto, limitar la corriente que circula por la carga, en este caso el foco, y con esto aumentar o disminuir su intensidad.

En el diagrama que se muestra a continuación están todos los componentes que se deben utilizar para llevar a cabo este proyecto. En la terminal 4 del microcontrolador ATMEGA8 deberá conectar la salida del receptor infrarrojo (OUT) y recuerde alimentar el receptor infrarrojo con 5 Volts (Vs y GND).

DIAGRAMA

En la figura 5 se muestra el diagrama esquemático completo.

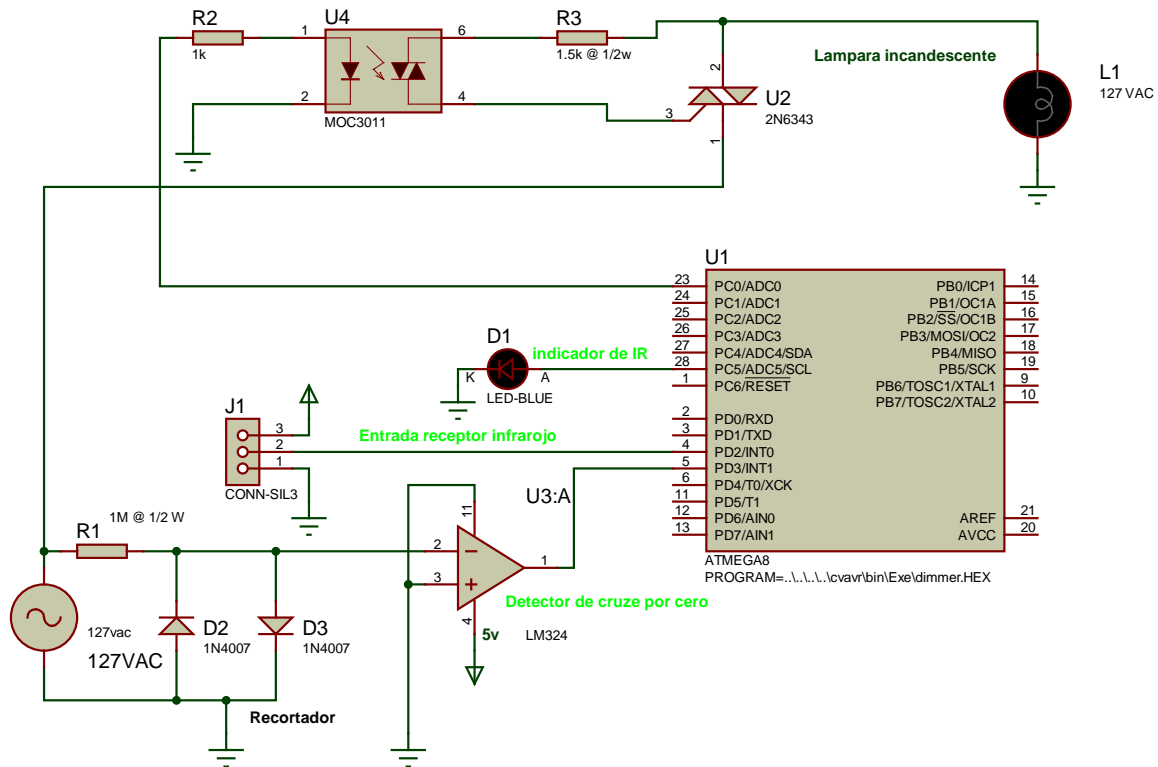


Figura 5: Diagrama completo del dimmer electrónico

NOTA: En la parte del *recortador* mostrada en la figura 5, observe que NO se cuenta con aislamiento de la línea eléctrica. En caso de algún descuido al tocar alguna de las terminales de los diodos, sufrirá una descarga eléctrica, por lo que se recomienda la utilización de un transformador reductor, para aislarnos de la red eléctrica. (recordar que no nos importa el voltaje de la línea, lo único que nos importa para fines del proyecto es la sincronización.)

NOTA: En caso de utilizar un transformador reductor, calcular la *RI* mostrada en la figura 5. Por ejemplo, si se utiliza un transformador reductor de 127VAC a 12VAC, se requiere una resistencia más baja que la mostrada; de aproximadamente 12 kOhm.

Para controlar el ángulo de disparo del tiristor, se necesita sincronizar con la línea eléctrica el tiempo que durará prendido el tiristor. Teniendo en cuenta que la frecuencia de la red eléctrica son 60Hz, en tiempo serían 16.66ms.

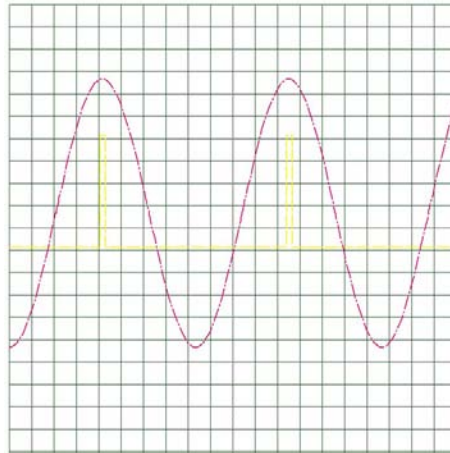
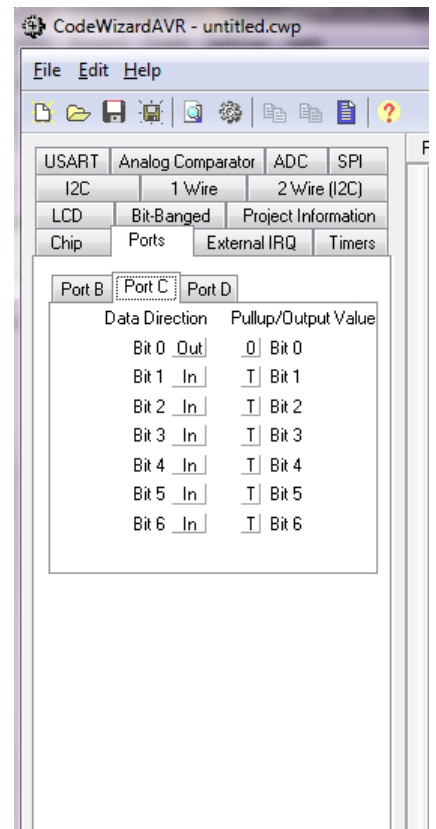
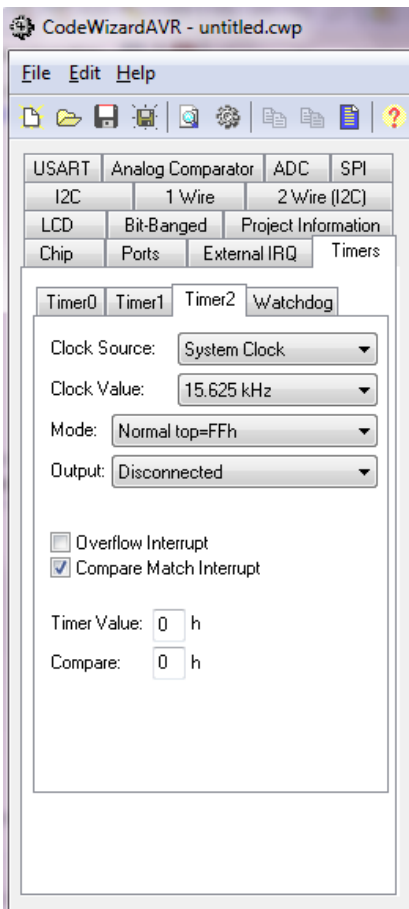
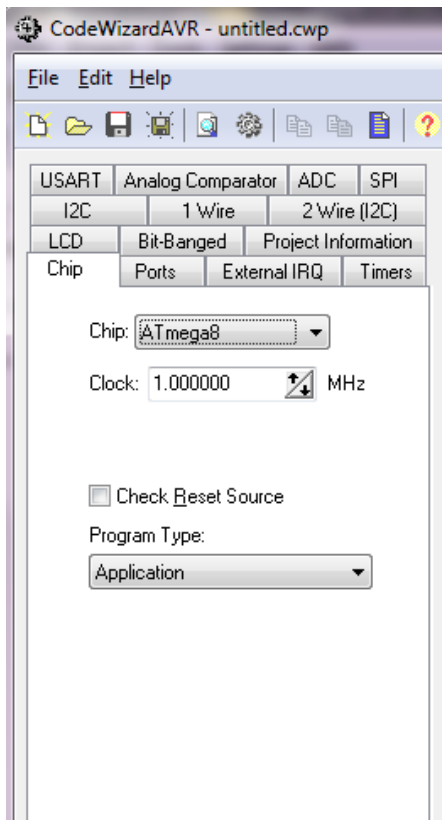
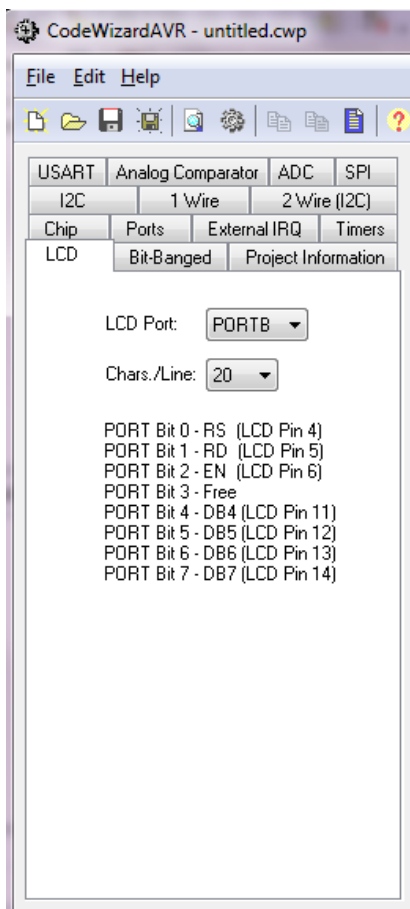
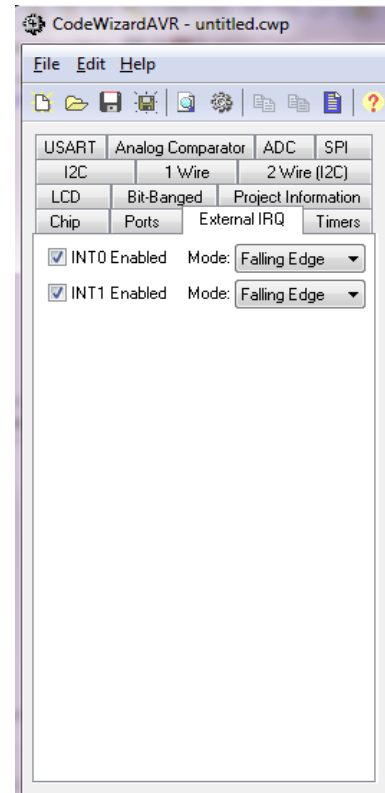
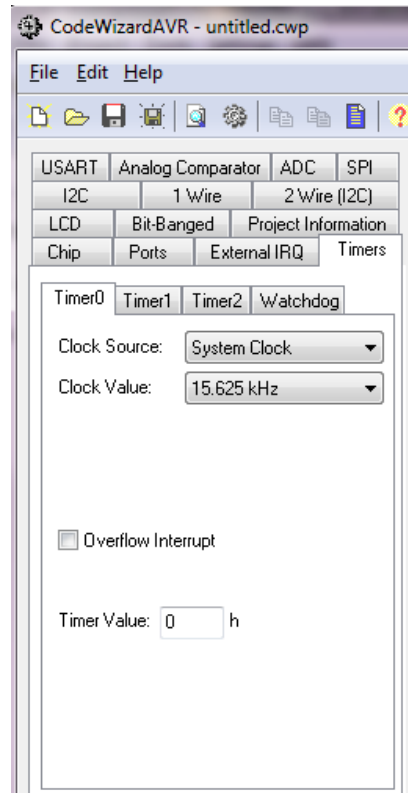
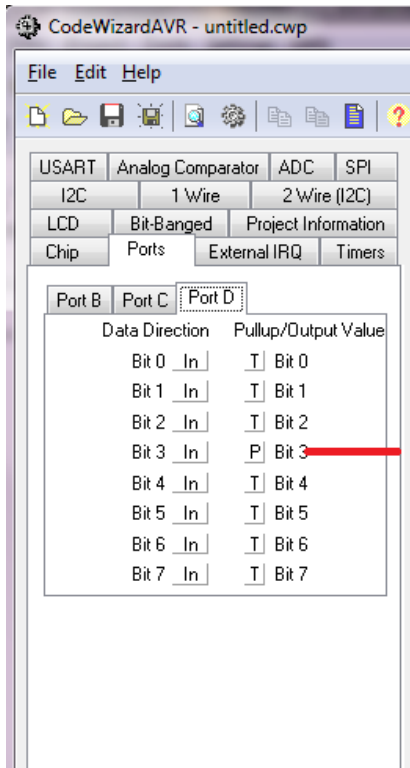


Figura 6: Disparo del tiristor al 50%

Según lo observado en la figura 6, la carga, en este caso el foco, solo recibirá el 50% de la corriente, con lo que se observaría la intensidad del foco a la mitad de lo que normalmente lo es.

En la figura 5 se observa en el diagrama que se tiene una pantalla de LCD, esta solo fue puesta para darse cuenta de que el código que era enviado por el control remoto era el correcto, pero esta se puede eliminar para hacer más eficiente el programa. Para lograr el disparo del tiristor en el instante adecuado, se utilizó un recortador, para no dañar el amplificador operacional utilizado como comparador (detector de cruce por cero), y el comparador a una interrupción del microcontrolador configurada como flanco de bajada para a partir de ahí hacer el disparo de un tiempo, ligeramente mayor a 0ms, pero menor de 8.66ms (el equivalente de 0 a 90 grados de control).





****NOTA:** Observe que se habilita resistencia de pull-up para la interrupción 1, esto es para que el micro-controlador funcione correctamente detectando los cruces por cero.

Programa

```
/******
```

```
Chip type      : ATmega8
```

```
Program type   : Application
```

```
Clock frequency : 1.000000 MHz
```

```
Memory model   : Small
```

```
External RAM size : 0
```

```
Data Stack size : 256
```

```
*****/
```

```
#include <mega8.h>
```

```
#include <stdio.h>
```

```
#include <delay.h>
```

```
#define COUNTER_LOWER_LIMIT 0x00 // para establecer el minimo del timer
```

```
#define COUNTER_UPPER_LIMIT 0x70 // para establecer el maximo del timer
```

```
#define STEP_SIZE 0x02 // para los incrementos en la intensidad
```

```
unsigned int read_IR (void); //funcion para leer el codigol del control remoto
```

```
void control_alfa (unsigned char code, unsigned char address); // funcion que controla el angulo de disparo
```

```
unsigned char code, address;
```

```
char lcd_buffer[33];
```

```
// Alphanumeric LCD Module functions
```

```
#include <lcd.h>
```

```
#asm
```

```
.equ __lcd_port=0x18 ;PORTB
```

```
#endasm
```

```
// External Interrupt 0 service routine
```

```
// PARA EL RECEPTOR INFRAROJO
```

```
interrupt [EXT_INT0] void ext_int0_isr(void)
```

```
{
```

```
    unsigned char count;
```

```
    unsigned int IR_input;
```

```
    TCNT0 = 0; // para poner el timer en 0
```

```
    while(!(PIND & 0x04)); //mientras PIND se mantenga en 1
```

```
    count = TCNT0;
```

```
    if(count < 30) //verifica pulso de inicio (2.4 ms)
```

```
    {
```

```
        delay_ms(20);
```

```
        return;
```

```
    }
```

```
    PORTC |= 0x20;
```

```
    IR_input = read_IR (); // brinca a la subrutina read_IR
```

```
    code = (unsigned char) ((IR_input & 0xff00) >> 8); //para acomodar el comando del control
```

```
    address = (unsigned char) (IR_input & 0x00ff); //para acomodar la dirección del control
```

```
    control_alfa (code,address); //brinca a la subrutina control_alfa (para el angulo de disparo)
```

```
    //lcd_gotoxy(4,0);
```

```
    //sprintf(lcd_buffer,"Com=%u Add=%u ",code,address);
```

```
    //lcd_puts(lcd_buffer);
```

```
    PORTC &= ~0x20;
```

```
    //delay_ms(500);
```

```
}
```

```
// External Interrupt 1 service routine
```

```
// PARA EL DETECTOR DE CRUZE POR CERO
```

```

interrupt [EXT_INT1] void ext_int1_isr(void)
{
    TCNT2=0x00;           //pone en cero el contador del timer2
}

// Timer 2 output compare interrupt service routine
//PARA LA VARIACIÓN DEL ÁNGULO DE DISPARO

interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
    PORTC.0=1;           // prende el bit 0 del puerto C
    delay_us(7);         // espera un poco para poder ver el led encendido
    PORTC.0=0;           // apaga el puerto c bit 0
}

//////////////////////////////////// para leer la señal del receptor
infrarojo////////////////////////////////////

unsigned int read_IR (void)
{
    unsigned char pulseCount=0, code = 0, address = 0, timerCount;
    unsigned int IR_input;

    while(pulseCount < 7) // para que haga 7 veces (LOS 7 BITS DEL COMANDO DEL
CONTROL)
    {
        while(PIND & 0x04); //mientras se mantenga el 1

        TCNT0 = 0; // contador del timer0 en cero

        while(!(PIND & 0x04)); //cuando ya no haya un 1

        pulseCount++; //aumenta pulseCount en 1, con lo que significa que ya encontro 1 bit de
la cadena

```

```
timerCount = TCNT0; // se almacena en timerCount lo que tiene TCNT0, es la duracion de 1 bit
```

```
if(timerCount > 14)
```

```
code = code | (1 << (pulseCount-1)); // si el bit fue un 1
```

```
else
```

```
code = code & ~(1 << (pulseCount-1)); // si el bit fue un 0
```

```
}
```

```
pulseCount = 0; //pone variable en 0
```

```
while(pulseCount < 5) // para saber la dirección del control remoto
```

```
{
```

```
while(PIND & 0x04); // mientras haya un 1 a la entrada
```

```
TCNT0 = 0; // contador del timer0 en cero
```

```
while(!(PIND & 0x04)); // cuando ya no haya un 1
```

```
pulseCount++; // aumenta pulseCount en 1
```

```
timerCount = TCNT0; // pone el valor que tiene el timer0 en timerCount
```

```
if(timerCount > 14)
```

```
address = address | (1 << (pulseCount-1)); // si fue un 1 el bit
```

```
else
```

```
address = address & ~(1 << (pulseCount-1)); // si fue un 0 el bit
```

```
}
```

```
IR_input = (((unsigned int)code) << 8) | address; // para guardar toda la cadena de bits
```

```
return(IR_input); // para que regrese a la función que mando llamar a esta rutina
```

```
}
```

```
/**  
****
```

```
//Funcion para controlar el angulo de disparo
```

```
/**  
****
```

```
void control_alfa (unsigned char code, unsigned char address)
```

```
{
```

```
static unsigned char counter;
```

```
if (address != 1) // para que solo funcione con señales de TV
```

```
return;
```

```
if(code == 19) // se presiono el boton Volume+
```

```
{
```

```
if(counter >= COUNTER_UPPER_LIMIT) // si ya esta en el limite, se queda ahi...
```

```
counter = COUNTER_UPPER_LIMIT;
```

```
else
```

```
counter += STEP_SIZE; //increase speed by a fixed step // incrementa el valor del contador
```

```
OCR2 = counter; // OCR2 genera una interrupción, cuando Timer2 sea igual a counter
```

```
}
```

```
if(code == 18) // Se presiono la tecla Volume-
```

```
{
```

```
if(counter <= COUNTER_LOWER_LIMIT) //si ya se esta en limite inferior, no hacer nada
```

```
counter = COUNTER_LOWER_LIMIT;
```

```

else
counter -= STEP_SIZE; //reduce el valor de counter, de acuerdo al step_size
OCR2 = counter; // OCR2 genera una interrupción, cuando Timer2 sea igual a counter

}
if(code == 21) //boton power presionado
{
OCR2=0; // poner el registro de comparacion en 0, el foco se apagara
}
//lcd_gotoxy(4,1);
//sprintf(lcd_buffer,"alfa=%u cuentas    ",counter);
//lcd_puts(lcd_buffer);
}
void main(void)
{
// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=Out
// State6=T State5=T State4=T State3=T State2=T State1=T State0=0
PORTC=0x00;
DDRC=0x01;
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=P State2=T State1=T State0=T
PORTD=0x08; // habilitar pull-up para INT1 si no, no funciona correctamente
DDRD=0x00;
// Timer/Counter 0 initialization
// Clock source: System Clock

```

```
// Clock value: 15.625 kHz
TCCR0=0x03;
TCNT0=0x00;
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 15.625 kHz
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x04;
TCNT2=0x00;
OCR2=0x00;
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: On
// INT1 Mode: Falling Edge
GICR|=0xC0;
MCUCR=0x0A;
GIFR=0xC0;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x80;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
```

```
// LCD module initialization  
lcd_init(20);  
  
// Global enable interrupts  
#asm("sei")  
  
while (1)  
{  
    // Place your code here  
  
};  
}
```

Los límites del contador, se establecen de acuerdo al control de intensidad que se quiera tener. En este caso de 0 a 90 grados de control. También se advierte que en este diseño, por limitaciones en los registros de comparación del timer0 y del timer2 (solo tiene un registro de comparación), por lo que sólo se utilizó un semiciclo de la onda senoidal. Pero lo óptimo sería utilizar toda la señal, es decir 2 disparos en un periodo.

El comentario más importante. **No envíen correo pidiendo asesoría o aclaración de dudas. Por cuestión de tiempo no contesto.** Recomiendo que armen y prueben el circuito, posteriormente analicen línea por línea lo que se hace para entender lo que hace y pueden modificarlo si así lo requieren.