Virtueller Kamin

Wo Feuer ist, ist noch lange kein Rauch

Von
Piero Di Stefano
(Kanada)

Ein sanft flackerndes und leise vor sich hin knisterndes Kaminfeuer ist eine Attraktion für Menschen und Haustiere. Aber nicht jeder kann sich ein reales Feuer im Wohnzimmer leisten. Eine Alternative stellt der voll-integrierte, mp3-unterstützte Virtuelle Kamin dar, der ohne Rauchgeruch auskommt, ohne fliegende Glut und die Sorge, dass er nicht rechtzeitig brennt. Und ohne Holzholen draußen in der Kälte…

Licht aus, Spot an: Hier ist ein Gerät, das einen brennenden Kamin simulieren kann. Es benötigt dazu 230-V-Glühlampen (alles, was dimmbar ist). Mit einer roten, nicht abgeblendeten Lampe erzeugen Sie einen tiefen roten Hintergrund, eine gelbe und eine weiße Lampe (oder jede andere Farbe, die Sie möchten) flackert im Rhythmus des Knistern und Knacken des Feuers. In der Tat verfügt der Virtuelle Kamin über einen kleinen, aber lautstarken Verstärker, der den Sound eines echten Kamins spielt. Die MP3-Sound-Datei kann von [1] heruntergeladen werden, und wenn Sie den Klang nicht mögen, gibt es eine Unzahl von Alternativen auf YouTube.

Wie es funktioniert

Ein Blick auf die Schaltung in **Bild 1** zeigt, dass die auf einer SD-Karte gespeicherte Audiodatei von einem MP3-Soundmodul (SoundModule1 oder 2) wiedergegeben wird. Der Autor verwendet einen Tenda-MP3-Player (als „SoundModule1“ verdrahtet). Das Elektor-Labor schwört auf den WTV020-SD-Mini-MP3-

Player (als „SoundModule2“ verbunden). Die beiden Module haben eine leicht unterschiedliche Anschlussbelegung.

Während des Hochfahrens weist der Mikrocontroller das MP3-Modul an, die Datei wiederzugeben. Die BUSY-Leitung (Pin 15) geht währenddessen auf LOW. Wenn die Wiedergabe beendet ist, kippt die BUSY-Leitung für einen Moment auf High. Der Microchip-Controller 12F683 (IC1) erkennt an GP0 diesen Pegel und sendet (wieder und wieder) einen Play-Befehl an das MP3-Modul (eigentlich nur einen Impuls). Und keine Angst: „Mikrocontroller“ bedeutet hier nicht eine Unzahl von beinahe unsichtbaren Beinchen, der 12F683 steckt in einem simplen und leicht handhabbaren DIP-8-Gehäuse.

Das Flackern der Lampen wird vom Mikrocontroller über zwei Triacs angeregt. Die rote Lampe ist direkt mit der Netzspannung verbunden, während die gelben und weißen Lampen durch ihre jeweiligen Triacs Tri1 und Tri2 gesteuert werden. Die Optokoppler IC5 und IC4 sorgen für eine galvanische Trennung zwischen Elektronik und Netzspannung.

Der gute alte Audio-Leistungsverstärker LM386 treibt einen 8-Ohm-Lautsprecher bis zu einer Leistung von etwa 500 mW. Ein Lautstärkeregler steht mit Trimmer P1 zur Verfügung. Das andere Steuerelement P2 mag hier vielleicht etwas seltsam erscheinen, erfüllt aber durchaus seine Funktion: Schauen Sie einmal in den Kommentar des microC-Quellcodes des Autors!

Die Spannungsversorgung ist wie üblich mit einem 3,3-V-Regler IC3 aufgebaut, der mit einer ungeregelten Spannung von etwa 8 V betrieben wird. Nur der Audio-Verstärker erhält die ungeregelte Spannung, alles andere läuft mit 3,3 V.

Der Mikrocontroller detektiert die Nulldurchgänge der Netzspannung mit Hilfe von T3. Diese Zeitinformation ist wesentlich, damit die Triacs zum richtigen Zeitpunkt geschaltet werden können, um die Spannung für die Lampen zu steuern, ohne dass massi-

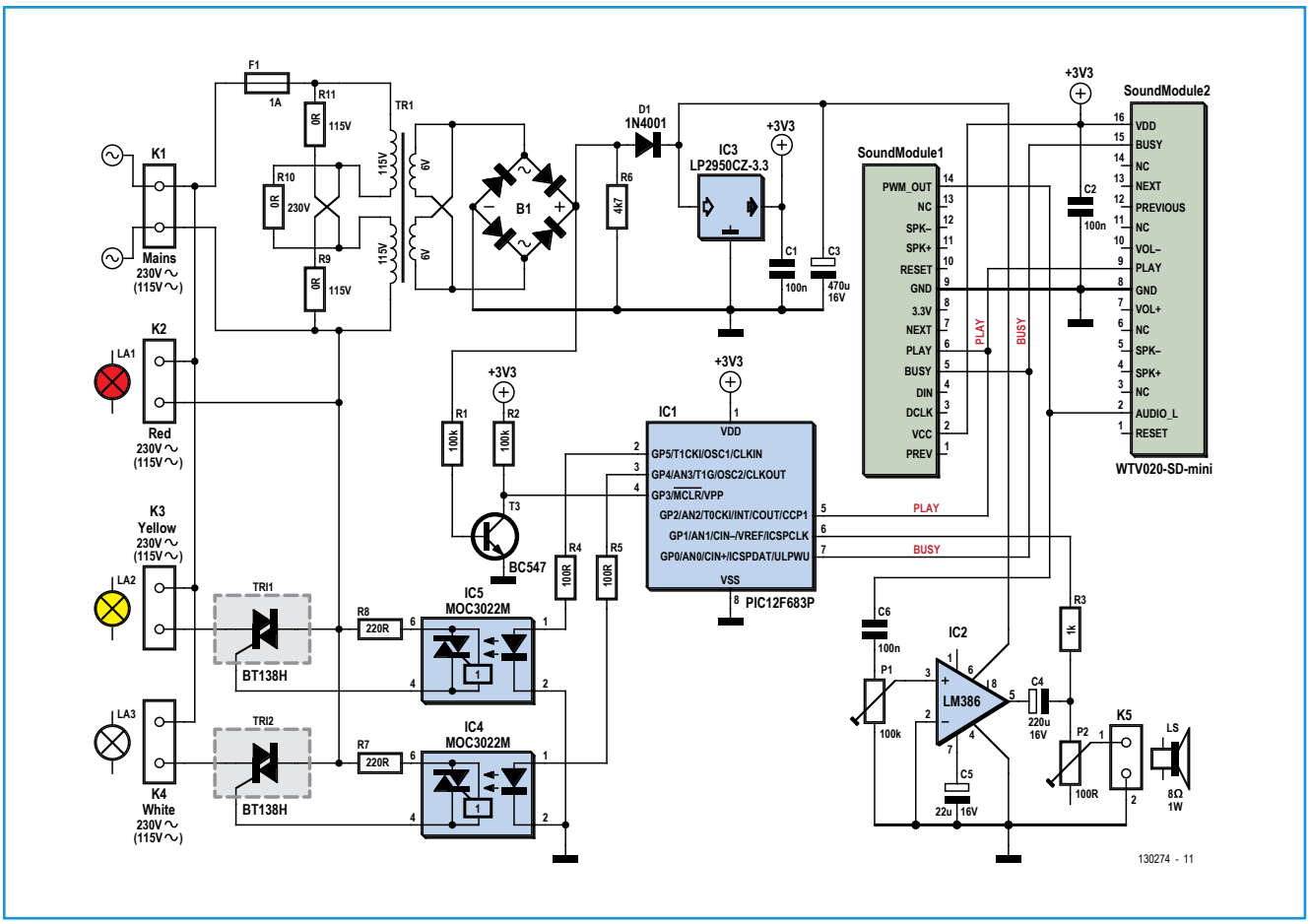

Bild 1.
Der Virtuelle Kamin ist die Antwort des Internets auf Holzhacken, Geruch von Anzündern, Qualm und quälender Wartezeit auf ein gutes Feuer. Es fehlt nur die Fähigkeit, viel Wärme zu erzeugen.

ver Elektrosmog auf die Netzleitungen gerät. Die Methode, solche Art Lampen zu dimmen, nennt man Phasenanschnittsteuerung.

Aufbau

Unser Elektor-Laborant Tim hat eine großzügig ausgelegte und elektrisch sichere Platine für das Projekt entworfen (**Bild 2**). Auf ihr befindet sich auch der 2x6-V-Transformator. Mit der Stückliste und den Fotos sollten die Bestückungsarbeiten keine Probleme bereiten, zumal keine SMD oder andere Kleinstteile beteiligt sind. Das Ergebnis ist in **Bild 3** und **Bild 4** dargestellt.

VORSICHT: Mehrere Leiterbahnen, Lötstellen, Schrauben und Bauteilanschlüsse auf der Platine führen Netzspannung. Daher dürfen – wenn die Schaltung in Betrieb ist – keine Bauteile oder die Verdrahtung berührt werden. Außerdem muss die Platine in einem berührsicheren Kunststoff-Gehäuse eingebaut werden, unter Einhaltung aller Vorschriften und Vorsichtsmaßnahmen für die elektrische Sicherheit.

Software: Ich bin so frei!

Die Firmware wurde mit der kostenlosen Version von microC von Mikroelektronika geschrieben und ist weniger als 2 KB groß. Den Code können wir hier in **Listing 1** in seiner ganzen Pracht präsentieren. Sie brauchen ihn aber nicht abzutippen, sondern können die Software bequem und kostenlos von [1]

Stückliste

Widerstände
(0,25 W, wenn nicht anders angegeben)
R1,R2 = 100 k
R3 = 1 k
R4,R5 = 100 Ω
R6 = 4k7
R7,R8 = 220 Ω
R9,R10,R11 = 0 Ω oder Drahtbrücke
P1 = Trimmpoti 100 k
P2 = Trimmpoti 100 Ω, 0,5 W

Kondensatoren
C1,C2,C6 = 100 n, 50 V
C3 = 470 µ, 16 V
C4 = 220 µ, 16 V
C5 = 22 µ, 16 V

Halbleiter
B1 = Brückengleichrichter (z.B. Vishay 2W005G-E4/51; Farnell 1497581)
D1 = 1N4001
IC1 = PIC12F683P-I/P, programmiert erhältlich 130274-41 [1]
IC2 = LM386N-1
IC3 = LP2950CZ-3.3/NOPB
IC4,IC5 = MOC3022M
T3 = BC547
TRI1,TRI2 = BT138-800

Außerdem
F1 = Sicherung, 1 A, träge
HS1, HS2 = Kühlkörper 21 K/W (Fischer FK 230 SA L1, Farnell 1892318)
K1,K2,K3,K4 = 2-polige Platinenanschlussklemme RM7,5
K5 = 2-polige Platinenanschlussklemme RM5
SoundModule1 = Tenda MP3-Player *
SoundModule2 = WTV020-SD-Mini-MP3-Player *
DIP6-Fassung für IC4 und IC5
DIP8-Fassung für IC1 und IC2
TR1 = Netztrafo für Platinenmontage 2x115 V prim./2x6 V sek. (Block AVB1.5/2/6, Farnell 1131474)
Platinensicherungshalter mit Kappe
Gehäuse (z.B. Hammond 1591USBK, Farnell 1426582)
LS = Miniaturlautsprecher 8 Ω, 1 W
Platine 130274-1 (V1.1), siehe [1]

* entweder SoundModule1 oder SoundModule2

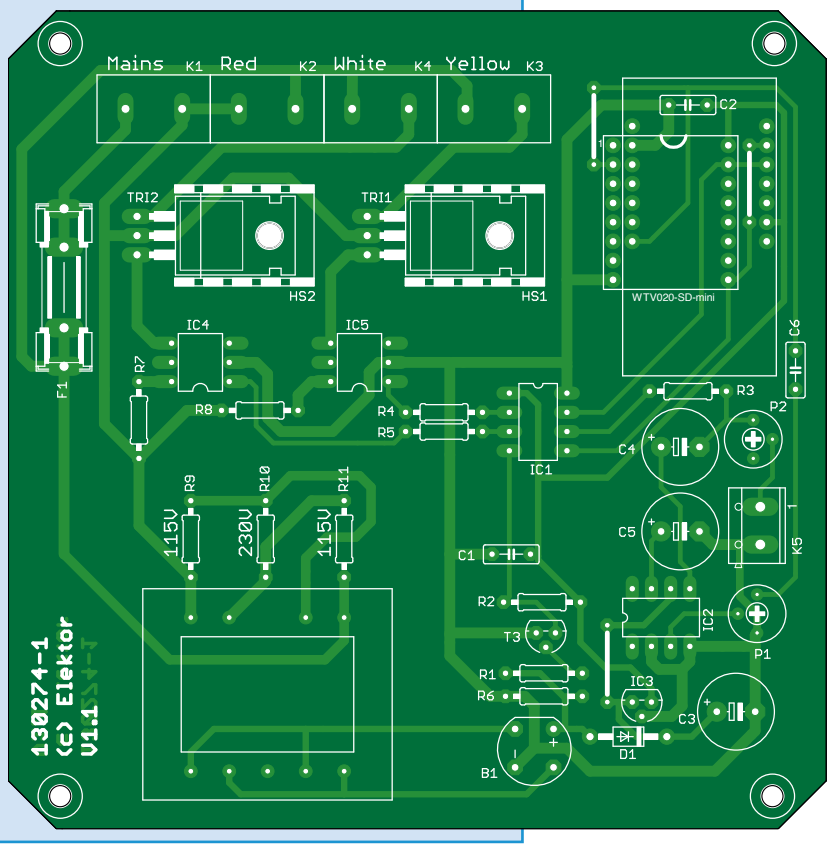

Bild 2.
Der Virtuelle Kamin wird auf dieser Leiterplatte aufgebaut.

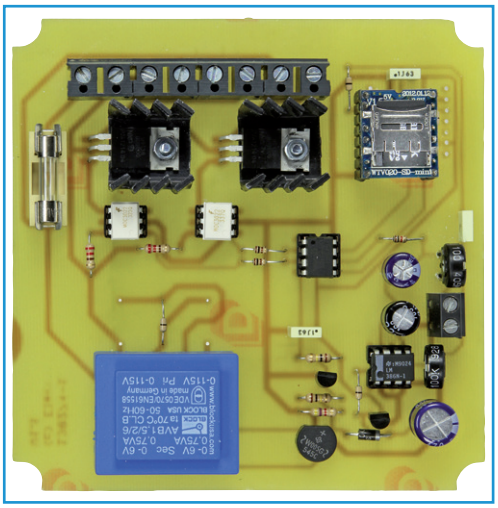

Bild 3. Die bestückte und getestete Platine.

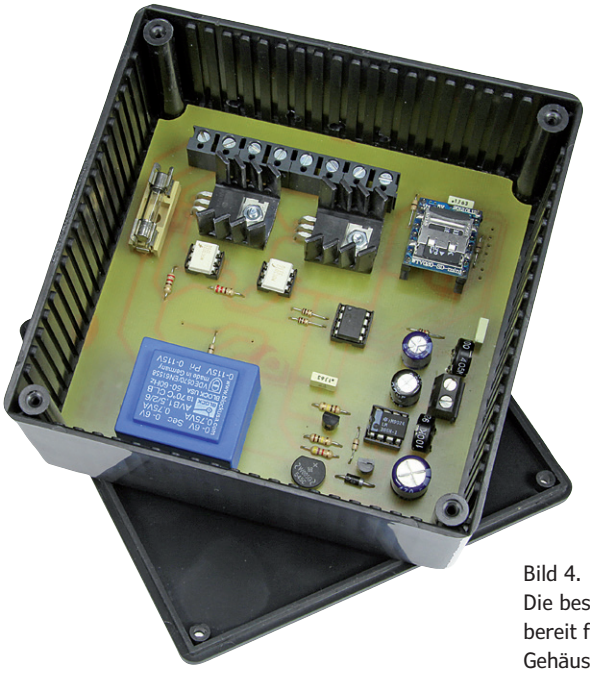

Bild 4.
Die bestückte Platine ist hier bereit für den Einbau in das Gehäuse.

herunterladen. Der Inhalt von TMR0 berücksichtigt die Netzfrequenz (50 Hz oder 60 Hz), lesen Sie bitte sorgfältig den Kommentar. So ist das halt, wenn ein Leser vom Kontinent der Füße, Höfe und Schoppen ein Projekt einschickt, das dann für die weltweite Verbreitung „engineered“ wird. So spielen die Widerstände R9...R11 nur in 115-V-Netzen eine Rolle, wir können sie einfach vergessen und Drahtbrücken einsetzen.

Wie immer muss man sich unweigerlich auch mit den Fuse-Bits des Controllers (**Bild 5**) beschäftigen.

In Action und auf YouTube

Der Virtuelle Kamin des Autors ist in mehr oder weniger flammender Aktion auf YouTube zu sehen [2]. Das Elektor-Labor hat auch eine Version aufgebaut und präsentiert das Ergebnis seiner Bemühungen in diesem Artikel. Es gibt auch ein Video des Virtuellen Kamins in Aktion, mit Tim, der erklärt, wie es funktioniert [3].

Obwohl die alten Kamine in mehreren Räumen des Elektor-Hauses (vermutlich aus der Mitte des 17. Jahrhunderts) vor ein paar Dekaden professionell renoviert wurden, sind die zugehörigen Schornsteine geschlossen und es ist nicht mehr möglich (oder erlaubt von unserem Vorstandsvorsitzenden, Architekten oder der Gesundheitsbehörde), ein echtes Feuer mit Holz, Rauch, starken Getränken und all diesem zu genießen. Die gute Weihnachtsstimmung konnte es uns dennoch nicht vermiesen, konnten wir doch diesen State-of-the-art-Ersatz ausbauen und statt Holzscheite den einen oder anderen PIC12F683 brennen.

(130274)

Weblinks

[1] Mp3-Datei, Software, Platinenlayout-Dateien: www.elektor-magazine.de/130274, 130274-11.zip

[2] Der Virtuelle Kamin des Autors auf YouTube: http://youtu.be/xK7Jj2aXOXI

[3] Der Virtuelle Kamin von Elektor auf YouTube: http://youtu.be/L_4yy8giTHk

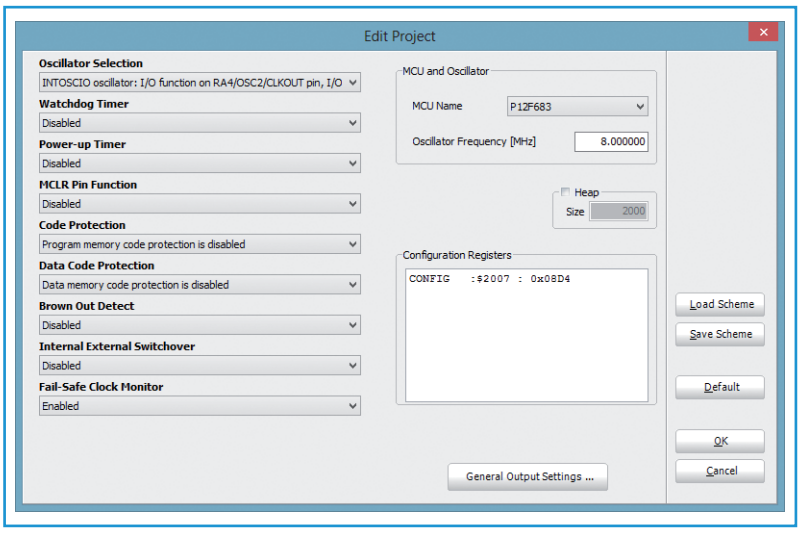

Bild 5.
Vor der Programmierung des PICs müssen die Fuse-Bits gesetzt werden.

Listing 1. MikroC-Quellcode Virtueller Kamin.

```
/*
 * Project name:
     PIC12F683 Electronic fireplace
 * Piero Di Stefano, MAY 8th 2013.
   The triacs are driven generating a delay that is inversely proportional to the sound level.
  a 60Hz half-wave takes 1/30= 16.66ms to complete, so we have to handle any time between zero and
      8333us.
  Here we go up to 8160, then above such a value, we keep the triac triggered. It's impossible to
      notice it.

  At 50Hz the timespan will be longer, 10ms, so we should set TMR0 prescaler to 128. That means
      that TMR0 interrupt will fire every
  128*255/2000000 = 16320us

  So first thing, in the main() remember to change the following statement:
  OPTION_REG = 0b10000101;  //TMR0 prescaler: 1/64

  with
  OPTION_REG = 0b10000110;  //TMR0 prescaler: 1/128

  In order to limit its range between 0 and 10000us, we must use an init value for TMR0 of:
      255*10000/16320 = 156.25
  Obviously we must use a smaller value, so no less than 156
  Countercheck:

  TMR0 = 156
  Tmr0 interrupt will fire every (156)*128/2000000 =  9984us

  Last important step to be taken:
  inside the TMROIF routine, we must change the following line:

  TMR0 = ADC_reading << 2;

  that prepares TMR0.
  Since we must use a minimum value of 156 for TMR0 to properly trigger the thyristor at 50Hz, we
      must limit our range to 255 -156 = 99.
  I suggest to limit the ADC reading to 127, shifting its value 3 times instead of just 2 and
      adding no more than 29 to it

  TMR0 = (ADC_reading << 3) + 29;

  A smaller value would give us more allowance

  So the problem should be solved.


 * Test configuration:
```

```
     MCU:             PIC12F683
     Dev.Board:       EasyPIC6
     Oscillator:      Internal, 8MHz
     Ext. Modules:    none

     SW:              mikroC PRO for PIC
                      http://www.mikroe.com/eng/products/view/7/mikroc-pro-for-pic/
 * NOTES:
 Tenda mp3 mplayback module:
http://www.google.ca/#hl=en&sclient=psy-ab&q=tenda+electronics+tdb380&oq=tenda+tdb&gs_
      l=hp.3.1.0j0i22i30.848.5801.0.8070.9.9.0.0.0.0.120.811.6j3.9.0...0.0...1c.1.14.psy-ab.Cd-
      q3yzoZcI&pbx=1&bav=on.2,or.r_qf.&bvm=bv.46751780,d.dmQ&fp=ca03a074a5a6f34d&biw=1017&bih=596
*/

     unsigned short dummy;
     unsigned int ADC_reading;

void interrupt()
{
   if(INTCON.T0IF)
   {
     GPIO.F5 = 1;          //past some time between 0 and 8160 us, brings LOW the output
     INTCON.T0IE = 0;      //disables TMR0 IRQ preventin it to fire until the next rising edge
      (GPIF)
     INTCON.T0IF = 0;      //clear bit
   }

   if(INTCON.GPIF)
   {
     dummy = GPIO;
     /*IMPORTANT, this is the core of the program
     First, we multiply ADC_reading by 4 to compensate for the low voltage level from the
      loudspeaker.
     Can't always amplify the mp3 file level, otherwise it will sound distorted. So keep the volume
      down and change the voltage threshold
     Note that the louder the volume, the higher the number we get from the ADC, the higher value
      TMR0 will be initialized at.
     So we get a smaller delay on triggering the triac and the light will appear brighter.
     */
     TMR0 = ADC_reading << 3 + 29;
     //If the sound level is all the way up to the max (at 245/255 to be precise), disable TMR0 and
      set GP5 HIGH permanently,
     //it will keep the triac triggered for the whole cycle
     if (TMR0 > 245)
     {
       INTCON.T0IE = 0;
       GPIO.F5 = 1;
     } else
       {
```

```
         delay_us(160);       //fine adjust zero-crossing synchronism, to compensate the early
      saturation of the bjt and trigger the triac precisely
         INTCON.T0IE = 1;
         GPIO.F5 = 0;         //triac gate LOW  NOTE: GP2 is LOW to begin with, then goes high.
       }

     INTCON.GPIF = 0;
   }

}


void main() {

OSCCON = 0b01110001;      //Enables 8MHz int osc  (we must SUPERSEDE Project prop. dialog box in
      order to get it to work)

OPTION_REG = 0b10000110;  //TMR0 prescaler: 1/128
INTCON = 0b10001000;      //general irq enabled  + GPIF (TMR0 enabled programmatically)
ANSEL = 0x02;             //AN1 enabled
CMCON0 = 0x07;            //disables comparators
IOC.IOC3 = 1;             //enables RBINT on GPIO.F0

trisIO = 0b00001011;      //GP0 digital input (~BUSY) to replay the file, GP1 = analog input from
      audio player, GP3 digital input from ZCD
GPIO = 0b00000100;        //GP2 must should start high.
TMR0 = 156;               //Init value


    // play mp3 file at startup
    GPIO.F2 = 1;
    delay_ms(100);
    GPIO.F2 = 0;
    delay_ms(100);



 while(1)
 {
    ADC_reading = ADC_Read(1);

    //White light bulb: when the sound level is higher, it goes on, and stays that way until next
      reading.
    if (ADC_reading > 1000) GPIO.F4 = 1; else GPIO.F4 = 0;


    if (GPIO.F0)     //F0 = HIGH only when device is in IDLE mode, goes LOW when playing
    {
      delay_ms(500);
```

```
      GPIO.F2 = 1;
      delay_ms(100);
      GPIO.F2 = 0;
      delay_ms(100);
    }

 }//end while

}// end main
```