Some notes on the Dalek and the control software to date:

- Important files are in /home/pi/Desktop/Dalek and /home/pi/sketchbook
- It might be easiest to have the line-following and drive motor control completely separate from the sound, lights and head servo. However, there is a USB connection between the Pi and the Dspace board if you wish to use it, and have the whole thing co-ordinated from a master program running on the Pi.
- You can use the Pi's audio output as usual (via the Linux ALSA drivers). The Pi has mplayer installed, which is a nice command-line (i.e. scriptable!) media player. However, if you want, I've configured the Pi so it can use my TclJACK library for interacting with the JACK audio system from a Tcl program (see instructions below). There is example code already in the dalek-master-control.tcl program. However, you might prefer to use Python than learn a new language.
- There seems to be a problem with the lights interfering with the head servo. It might be possible to fix this by adding some big (electrolytic) capacitors (watch the polarity!) to filter the power supply and/or some diodes to divert the spikes. Adding a separate power supply just for the lights is another possibility. You might need to study the wiring of the shield for the head-control Arduino to figure out how to connect the LEDs (they won't work unless they are correctly polarised).
- The main power supply (from 2 parallel banks of 6 series batteries) is now 9 V. The NXT servos, audio amplifier, Dspace board and LEDs should all run happily from this supply (although the LEDs might not light if the batteries are going flat). The Pi has a regulated 5 V supply from a DC-to-DC converter (the black module with red LED).

When programming the Arduinos from the Pi, make sure you select the right board type:

- Dspace Robot for the Dspace Robot (duh!)
- Arduino Uno for the heat controller

To have the lights flash in sync with the audio:

```
# Tell the Pi to use the built-in audio even if HDMI is connected:
amixer cset numid=3 1

# Start the JACK audio server:
jackd -R -d alsa -n 3

# Play a sound file (open a new terminal, because jackd will hog its one):
mplayer -ao jack:name=mplayer /home/pi/Mirrormusic.mp3

# If you need to change the sound levels, run alsamixer (also in a new terminal)
alsamixer

# Open yet another terminal, and start the Tcl shell (pronounced “ticklish”!).
# (“rlwrap” gives the Tcl shell nice line-editing capabilities.)
rlwrap tclsh

# You should now be at a “%” prompt.  Code you enter now is actually Tcl
scripting commands, not shell commands.

# Load the TclJACK library:
load /usr/local/src/TclJACK/libtcljack.so

# Connect to the JACK server:
jack register

# Connect the audio stream from mplayer to TclJACK's sound meter input:
# (use “jack ports” to see what ports are available)
jack connect mplayer:out_0 tcljack:input
```

```
# Get various stats for the audio stream being received:
# (the second number (RMS level) should correlate to “volume”)
jack meter

# See the example .tcl scripts for connecting via serial to the Arduino and
constructing and streaming command bytes...
```

Having the Pi run the Dalek program automatically might be a bit tricky. You probably want to add a button so you can tell it when to start running. This should be added to the head-control Arduino (since the drive motor board might not be usefully connected to the Pi). Shutting down the Pi gracefully (using the Linux “shutdown” command), either automatically or in response to a button-press, would also be a nice refinement.