VxD LDR controller

1. Introduction

This is a high quality audio LDR attenuator and line input/output controller.

1.1. Features:

- customizable impedance, between 5kohm and 50kohm
- customizable number of attenuation steps, between 20 and 80
- you do not need matched LDRs
- LOG attenuation curve
- you can display your own welcome messages on the LCD screen
- the firmware is easy to update and modify
- remote controlled with any Apple remote
- big LCD screen, the volume level is displayed with 4-char high digits, easy to see from a distance
- the screen auto-dims to a customizable level after some inactivity time
- rotary encoder with push button
- controls up to 6 input and output stereo channels
- I/O switching is done with best quality latching relays with Silver-Palladium contacts, to avoid any degradation of the musical signal
- you can name each input and output channel
- the controller remembers the settings after power off
- can control a delay relay to soft-start a tube preamplifier
- achieves a large attenuation range by increasing series resistance at very high attenuation level and by increasing shunt resistance at very low attenuation level
- the calibration compensates for the load impedance effect
- the on-board calibration relays are best quality and they are powered only during calibration
- the LDR LEDs are working at low current (7 mA maximum), they will last a very long time
- the controller is isolated from the audio ground to avoid noise and loops
- separate linear analog and digital low noise power supplies
- power supplies on separate board, to keep the power transformer far from audio circuits
- easy to calibrate anytime from a menu – no need to plug jumpers or an external module
- better audio quality than R-2R relay attenuators (no multiple relay contacts and solder joints in the signal path, no noisy relay coils)

1.2. Composition:

The controller is composed of a number of modules:

- the main controller board
- a power supply board
- one or two audio input/output modules
- one 4x20 LCD backlit display
- one IR remote receiver sensor
- one rotary encoder

2. Assembly

Warnings:

- the MOSFETs are sensitive to static electricity. Touch a grounded metal before handling.
- the LDRs are sensitive to temperature. Do not solder for more than a few seconds.
- inverting the polarity on any power connexion will destroy the module
- once soldered, it is almost impossible to de-solder any component that has more than 3 pins (excepting the LDRs, which have long, flexible leads)

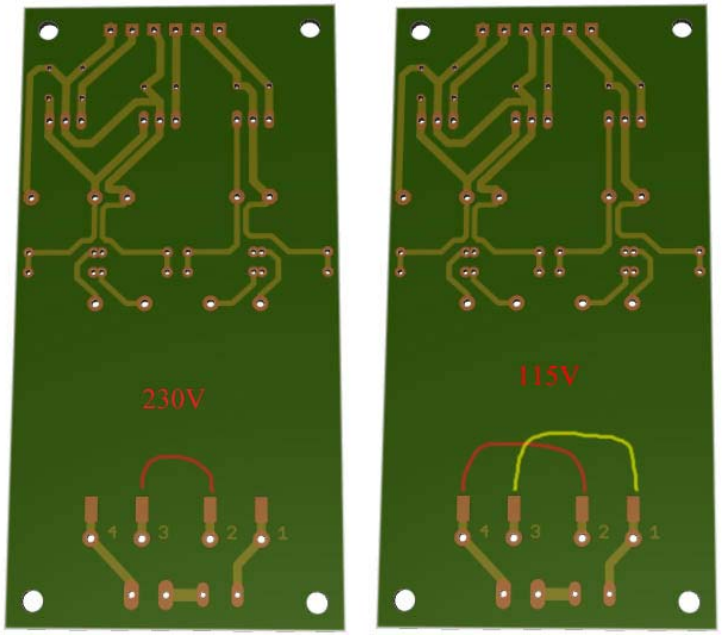
Using 0.5mm flux core solder is recommended for clean, good solder joints.
The component values are critical, so double check every resistor value before soldering.

The Arduino module should be connected using male and female 2.54mm / 0.1" pitch pin headers, so that it is easily unplugged from the controller.

Pay attention to the LDR polarity. The white dot on the body should match the white dot on the board.

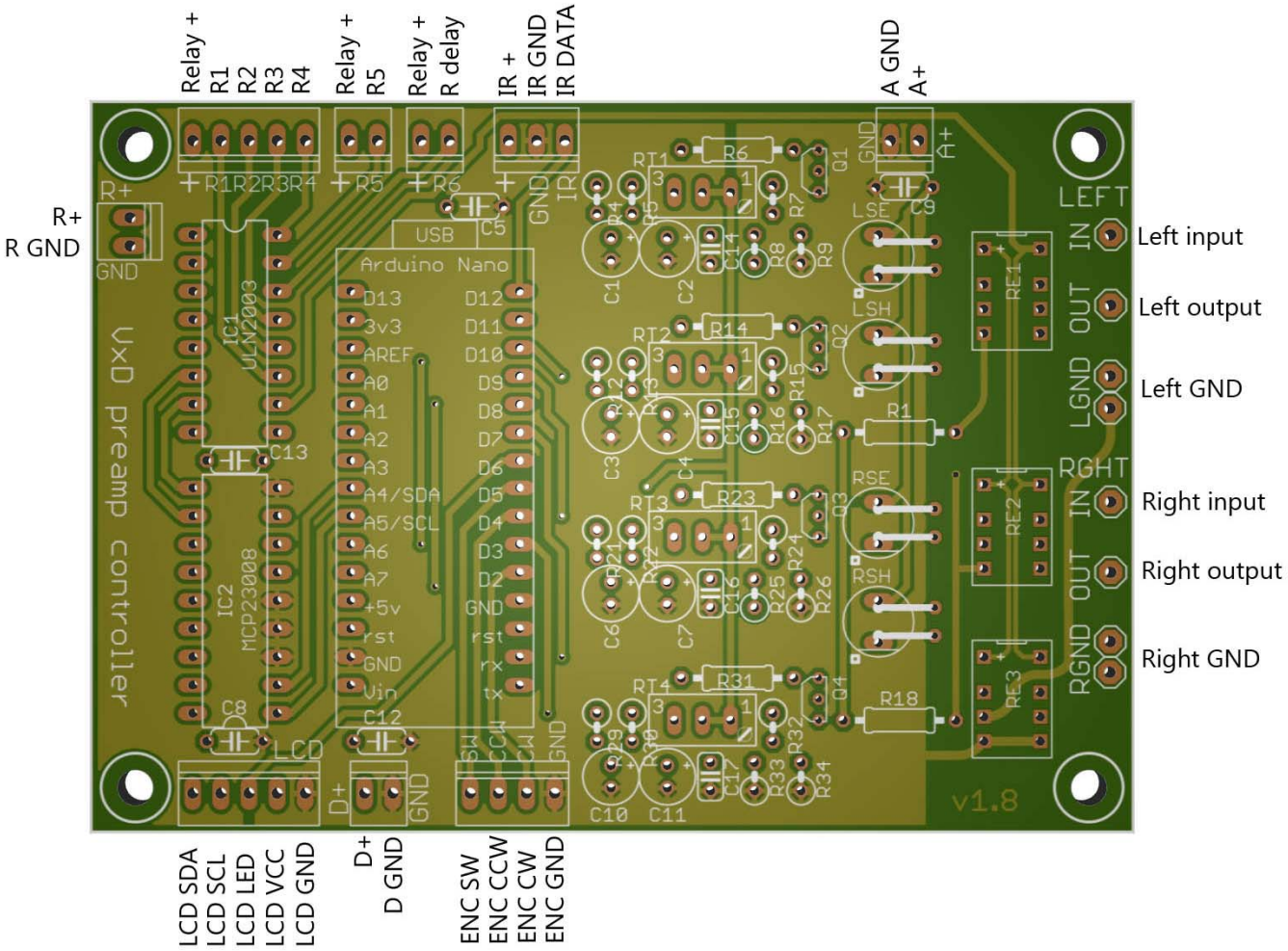
The power supply module uses a transformer with dual primary windings and it must be configured to match the mains voltage in your country: in series for 230V and in parallel for 115V.

For 230V mains, solder an insulated wire between pads 2 and 3 on the back of the PSU PCB.
For 115V mains, solder an insulated wire between pads 1 and 3, and another insulated wire between pads 2 and 4:



Use good quality insulated wire.
Warning - danger of deadly electric shock. Insulate and make sure you cannot touch any part connected to mains voltage.

3. Wiring



3.1. Power supplies

The module uses 3 separate power supplies: D+ (8v) for digital, R+ (12v) for relays and A+ (12v) for the LDRs.
The "GND" pin on the power supply module should be wired to the chassis or to ground/earth.
Use twisted or zip cord wire pairs for each power supply connexion.
Cut the wire pairs at needed length and then crimp each wire to a contact. Insert the contacts into the connector housings, paying attention to polarity.

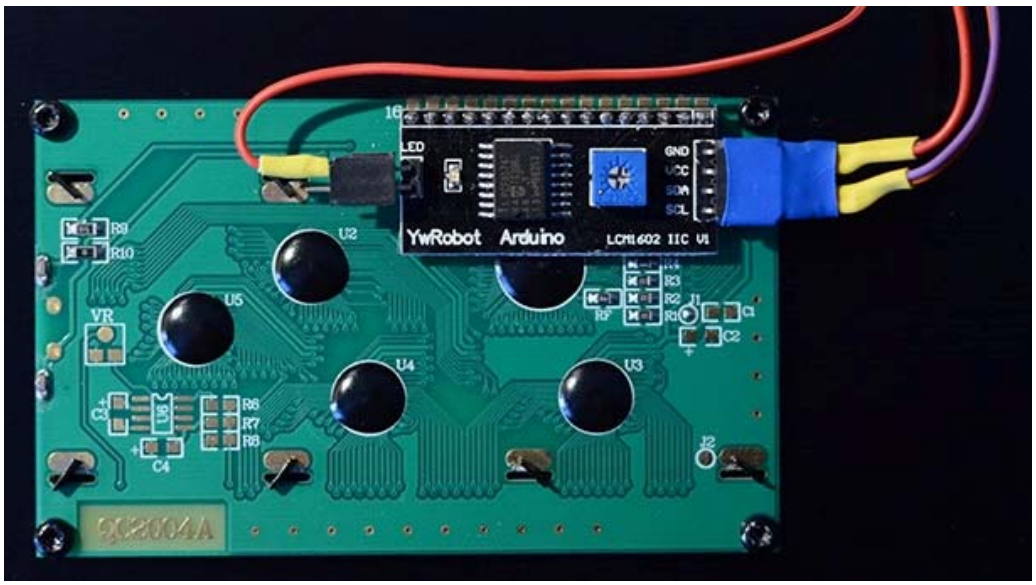
3.2. The LCD display

The display only needs 5 wires:

- GND
- VCC +5v
- SDA
- SCL
- LED backlight

The GND, +5v and backlight wires should be twisted or tied together.

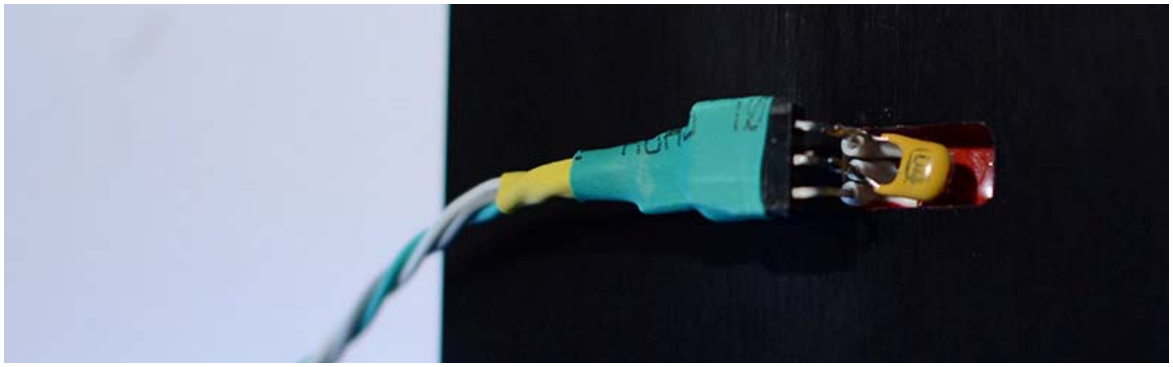
Solder the wires to 2.54mm / 0.1" pitch female pin header connectors, cut to length. See the following picture (the connector on the right is covered with a heat shrink sleeve).

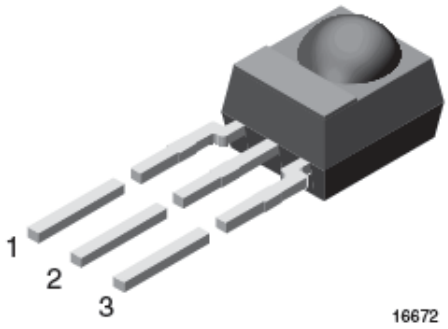


The backlight wire “LCD LED” goes to the upper pin of the “LED” header on the LCD module (remove any jumper, if present).
The GND, VCC, SDA and SCL pins of the LCD module are connected to the corresponding pins of the controller's LCD connector.
Adjust the display contrast using the trimmer on the back of the LCD module (if this adjustment is not done properly, the display may seem to remain blank).

3.3. The IR receiver

The IR receiver should have a 0.1µ ceramic or SMD capacitor soldered between its Vs (+3.3v) and GND pins, close to the receiver body.
In the following picture, the receiver is mounted in a recessed carving on a thick front panel and is connected with a female pin header connector (the wires can also be directly soldered to the IR receiver).





MECHNICAL DATA

Pinning for TSOP348.., TSOP344..:

1 = OUT, 2 = GND, 3 = V_S

- Pin 1 (OUT) goes to pin "IR DATA" on the controller board IR connector
- Pin 2 (GND) goes to pin "IR GND"
- Pin 3 (Vs) goes to pin "IR +"

The small capacitor goes between pins 2 and 3.

3.4. The rotary encoder

Please consult the datasheet of the rotary encoder for the pinout. The push button should be wired between ENC SW and ENC GND. The rotary contacts should be wired to ENC CW, ENC CCW and ENC GND. If the rotation direction is not coherent with the volume display, either invert CW and CCW pin numbers in software or invert the CW/CCW wires.

3.5. INPUT / OUTPUT control

The module can control up to five DPDT relay pairs - that means up to six stereo I/O channels. The signal ground is also switched.

High quality single coil latching relays are used, for best audio quality. Each latching relay needs a special driver circuit – see the sample schematic provided.

You have to define the number of input and output channels needed, by changing the `#define INPUTCOUNT` and `#define OUTPUTCOUNT` lines in the code.

If you set the input or output number to 0, no channel name will be displayed on the LCD (and the channel is wired directly, without relay switching).

If you set the input or output number to 1, the name of the channel will be displayed (and the channel is wired directly, without relay switching).

If set to 2 input or output channels, only one DPDT relay pair is used to switch between the two channels (one channel wired to the normally closed contacts, the other to the normally open).

If set to 3 to 4 input or output channels, each channel has its pair of DPDT relays.

The following I/O configurations can be used:

1) 0 to 2 input channels and 0 to 4 output channels
   If INPUTCOUNT = 2, the input channels use one relay pair wired on the R5 line.
   If INPUTCOUNT < 2, no input relay module is needed (wire directly to the RCA jacks)
   The outputs use lines R1 to R4.

2) 0 to 4 input channels and 0 to 2 output channels
   The input channels use lines R1 to R4.
   If OUTPUTCOUNT = 2, the output channels use one relay pair wired on the R5 line.
   If OUTPUTCOUNT < 2, no output relay module is needed (wire directly to the RCA jacks)

3) 0 to 2 input channels, 0 to 2 output channels
   If INPUTCOUNT = 2, the input channels use one relay pair wired on the R1 line.
   If OUTPUTCOUNT = 2, the output channels use one relay pair wired on the R5 line.
   If INPUTCOUNT < 2, no input relay module is needed (wire directly to the RCA jacks)
   If OUTPUTCOUNT < 2, no output relay module is needed (wire directly to the RCA jacks)

Note: you cannot declare 3 input and 3 output channels.

If you are using latching relays, there must be a small delay between channels switching. This is the `#define TIME_RELAYLATCH = 250` in the code.
If you are using normal relays, set `TIME_RELAYLATCH = 0`, to have a more responsive channel selection.

A number of sample I/O boards are provided as EAGLE files. You can, for example, use the "IO 3 in" board for three input channels and "IO 2 out" for two outputs, or combine them on a single PCB with "IO 3 in 2 out". Of course, the board named "IO 3 in" can also be used for 3 outputs - only the labels change from "in" to "out"…

For example, when using the "IO 2 in" board, wire as following:

- ATT R+ to the controller's RIGHT IN pin
- ATT R- to one of the controller's RGND pins
- IN1 R+ to input channel 1 right RCA jack's central pin
- IN1 R- to input channel 1 right RCA jack's ground tab
- IN2 R+ to input channel 2 right RCA jack's central pin
- IN2 R- to input channel 2 right RCA jack's ground tab
  (… and the same for the left channels)
- R+ and REL5 are connected with a twisted or zip cord pair to the controller's "+ R5" connector (R+ goes to "+", REL5 goes to "R5" on the connector)

3.6. Soft-start delay

If you have a tube preamp that needs to preheat the tubes before applying full B+ voltage, you can use the "+ R6" header on the board to connect a SPST low current 12v relay. No protection diode needed. Update the firmware with the needed delay, in seconds.

4. Starting up

Once the assembly of the power supply and controller modules is complete, follow these steps:

4.1. Upload the default source code

Connect the Arduino Nano module to the computer using a micro USB cable. See "**Uploading the code**" paragraph. Open the provided code file "`firmware_LDR_v1.0.ino`" with the Arduino IDE. You do not need to adjust any code parameters at this point. Upload the code and disconnect the Arduino from the USB cable.

4.2. Test the power supply

Power up the power supply module (warning: danger of deadly electric shock!). Measure the voltage of each of the 3 power outputs. You should have:

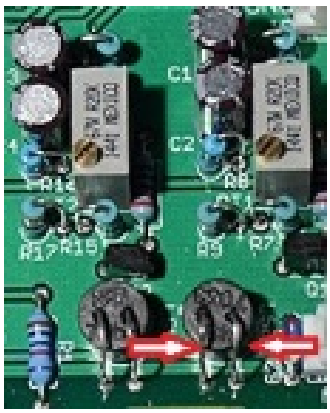
- 12v between A+ and G
- 8v between D+ and G
- 12v between R+ and G

4.3. First power up

Connect all three power supply connectors to the controller. Connect the LCD screen and the rotary encoder. Power up the power supply module.
The LCD screen should light up and display the calibration menu. The rotary encoder should allow you to navigate through the menu.

4.4. Adjust the bias

1. Rotate the encoder to select the "Adjust BIAS" menu line and push the encoder to select it.
2. Gently clip multimeter probes to each LDR's exposed two wires and measure its resistance.



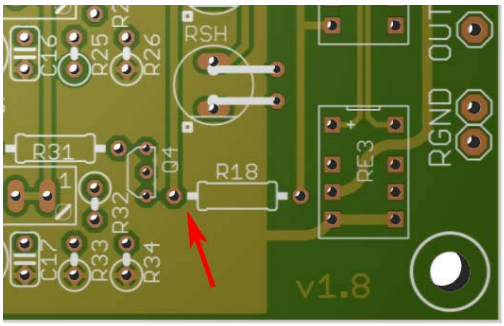
3. Adjust the trimmer corresponding to each LDR (trimmer "RT1" corresponds to LDR "LSE", "RT2" to "LSH", "RT3" to "RSE", "RT4" to "RSH") until you measure around 700Kohm. The precise value is not critical, but it should be between 500K and 1M.
4. Press on the encoder to end the procedure and go back to the menu.

4.5. Measure the LDR minimum values (optional)

1. Select " Measure LDRs" from the menu
2. Gently connect a multimeter to each LDR's exposed two wires and measure its resistance. It should be between 60 and 120 ohm.
3. Write down the values of each LDR: LSE, LSH, RSE, RSH
4. Press the encoder button to exit measuring mode and go back to the menu

4.6. Measure the +5v line (optional)

Measure the precise value of the 5 volts supply of the Arduino's onboard regulator. It can be measured between ground and the LEFT side of resistor R18. It must be measured with the module powered up and the USB cable disconnected. Write down the value, with one decimal (should be around 5.0v)



4.7. Measure R1 and R18 (optional)

Power down the module.
Measure and write down the value of resistors R1 and R18 (should be 10000 ohms if you used 0.1% parts).

4.8. Update and customize the code

Open the source code file "`firmware_LDR_v1.0.ino`" and follow the instructions in section "**5. Software configuration**".
Save and upload the code to the Arduino.

4.9. Calibrate

Power up the controller.
Select "Calibrate" from the menu. The calibration procedure will take around 10 minutes. Once complete, you can connect the IR receiver and the input/output modules or jacks.
You can now test the attenuation and I/O channel switching.
Directly rotating the encoder controls the attenuation.
Pressing the encoder button once and then rotating controls the input channel (if available).
Pressing the encoder button twice and then rotating controls the output channel (if available).
Pressing once more the button takes you to the Calibration menu.

You can repeat the calibration procedure at any time - the I/O connections can remain wired to the controller module during the procedure.

A new calibration is only needed if you change certain software parameters: nominal impedance `NOM_IMPEDANCE`, load impedance `LOAD_IMPEDANCE`, attenuation steps `VOL_MAX_STEP`, or if the ambient temperature is significantly different (is the module close to hot tubes or components?).

5. Software configuration

5.1. Editing the firmware code

You can customize multiple parameters of the attenuator by editing the `firmware_LDR_v1.0.ino` file.
To do this, you need to download and install the Arduino IDE version 1.6.1:
https://www.arduino.cc/en/Main/OldSoftwareReleases#previous
The newer Arduino IDE versions are not compatible.

Each parameter has a comment, explaining its use.
You can change the following lines:

Debug mode:

```
// Debug mode increases memory usage, the code may not compile. In this case, reduce attenuation steps number to 30.
// !! COMMENT OUT THE FOLLOWING LINE WHEN NOT DEBUGGING ("//#define DEBUG") !!
//#define DEBUG
```

Delete the comment "//" in front of "#define DEBUG" if you need to debug the code. In this mode, the number of attenuation steps must be limited to 30 (because of the available memory).

I/O configuration:

```
// Set I/O configuration here.
// There can be 0..2 inputs and 0..4 outputs, OR 0..4 inputs and 0..2 outputs.
// So for example we can have 2 IN and 4 OUT, or 3 IN and 2 OUT, or 4 IN and 1 OUT, but NOT 3 IN and 3 OUT.
#define INPUTCOUNT 2     //** number of inputs, 0 to 4. If 0, comment out the next line.
char* inputName[INPUTCOUNT] = { "DAC IN", "PHONO IN" }; //** each name maximum 9 characters. There must be exactly INPUTCOUNT names in the list.

#define OUTPUTCOUNT 3   //** number of outputs, 0 to 4. If 0, comment out the next line.
char* outputName[OUTPUTCOUNT] = { "LINE 1", "LINE 2", "HEADPHONE" }; //** each name maximum 9 characters. There must be exactly OUTPUTCOUNT names in the list.
```

See paragraph 3.5.

Delay:

```
// If you need a delayed relay contact (for tube filament preheating, for example), uncomment the following #define line.
#define DELAY 30  //** The value represents the delay in seconds. Maximum 100. Connects PIN_EXT_R6 to GND after DELAY seconds
```

Comment the line (type "//" before #define) if a delay is not needed. See paragraph 3.6

Text messages:

```
/******* Text messages *******/
char msgWelcome1[] = "";              //** <-- maximum 20 characters (line 1)
char msgWelcome2[] = "VxD Magister";  //** <-- maximum 20 characters (line 2)
char msgWelcome3[] = "";              //** <-- maximum 20 characters (line 3)
char msgWelcome4[] = "preamplifier";  //** <-- maximum 20 characters (line 4)

char msgCalib[] = "Calibrating";      //** <-- maximum 17 characters
char msgTest[] = "Self-testing";      //** <-- maximum 17 characters
```

You can edit these lines to display your own welcome message and the calibration menu entries. Mind the maximum length!

Attenuator control:

```
/******* Attenuator control *******/
#define NOM_IMPEDANCE 10000 //** target nominal attenuator impedance, between 5000 and 50000 Ohm. Recommended: 10000. Higher = less precision
#define LOAD_IMPEDANCE 220000 //** input impedance of the amplifier that follows the attenuator. Should not be less than 100K. Default: 220000
#define VOL_MAX_STEP 50     //** maximum volume steps; range: 20...80. Higher = more memory usage
#define VOL_DEFAULT 25      //** default volume step. Must be <= than MAX
```

NOM_IMPEDANCE is the nominal input impedance in ohms. For a 10K attenuator, use 10000. You can go up to 50000 or more, but the precision decreases. Recommended: 10000

LOAD_IMPEDANCE: this is the input impedance of the preamplifier following the attenuator. If not known, leave the default value.

VOL_MAX_STEP: the number of attenuation steps. Recommended: 50. You can go higher (up to 80), but due to limited Arduino memory, instability may occur.

LDR measured values:

```
/******* LDR measured values *******/
#define LDR_VOLTAGE 5.0     //** precisely measured value of the +5V supply (with decimal point. Default: 5.0)
#define LDR_R1 10010        //** precisely measured value of R1 resistor (default: 10000 ohm)
#define LDR_R18 10000       //** precisely measured value of R18 resistor (default: 10000 ohm)
```

Follow the measuring procedures in the section "**5. Starting up**".

LDR_VOLTAGE: this is the precise value of the 5 volts supply of the Arduino's onboard regulator, or you can use the default value.

LDR_R1 and LDR_R18: please measure the exact value of resistors R1 and R18, or leave the default value (10000).

LDR_LSE_MIN, LDR_LSH_MIN, LDR_RSE_MIN, LDR_RSH_MIN : these are the minimum resistances that the LDRs can go down to.
"LDR_LSE_MIN" is the measured value of the LDR labeled "LSE", "LDR_LSH_MIN" is the value of the LDR labeled "LSH", etc.

Screen

```
/******* SCREEN *******/
#define LCDBRI_MAX 90        //** LCD full brightness (1 - 255)
#define LCDBRI_MIN 18        //** LCD eco brightness
#define ROW_IN 0             //** LCD row to display input channel (0 - 3)
#define ROW_OUT 3            //** LCD row to display output channel (0 - 3)
```

After some time, the display backlight dims to LCDBRI_MIN value, to ease the load on the PSU. The value LCDBRI_MIN should be lower than LCDBRI_MAX.

ROW_IN, ROW_OUT: on which LCD line to display the I/O channel names

Timing

```
/******* TIMING *******/
#define TIME_LCDFADEIN 1        //** Time in seconds to bring the LCD to full brightness
#define TIME_LCDFADEOUT 10      //** Time in seconds to bring the LCD to dim brightness
#define TIME_LCDFADEAFTER 20    //** Time in seconds after last user input to start fade out
#define TIME_EXITSELECT 5       //** Time in seconds to exit I/O select mode when no activity
```

These values define how long it takes for the LCD backlight to go from full to dim, after how many seconds to go dim, and after how long to exit channel selection mode.

5.2. Uploading the code

Please use Arduino IDE version 1.6.1.
Copy the provided libraries into \My Documents\Arduino\libraries.
Depending on the Arduino module manufacture, you may need to download and install a special driver (consult the seller of the Arduino).
The Arduino module can be left plugged onto the main controller board during firmware upload, but the main power supply should be turned off.
If you don't have enough space to plug the USB cable into the Arduino module, you can unplug the module from the board.

6. Troubleshooting

The controller may display the following error codes:

1 : left series could not be calibrated
2 : right series could not be calibrated
3 : left shunt could not be calibrated
4 : right shunt could not be calibrated

In this case, make sure you have adjusted Bias and measured the LDR minimum values and updated the code with the values (chapter 4).
If it persists, replace the respective LDR (left series = LSE, left shunt= LSH, etc.).

10 : left series LDR too low max value. Adjust trimmer RT1
11 : right series LDR too low max value. Adjust trimmer RT3
12 : left shunt LDR too low max value. Adjust trimmer RT2
13 : right shunt LDR too low max value. Adjust trimmer RT4

Adjust Bias. See Chapter 4.

20 : cannot set LDRs or calibration relays. Analog and/or relay power supplies are not connected?

There is a problem with the A+ and/or R+ power supplies. Check the connectors, measure the voltage.

30 : configuration error: INPUTCOUNT and OUTPUTCOUNT out of range

This is a I/O configuration error. See 3.5