Arduino Sound-Level Protector

With three-color traffic light

By Clemens Valens (Elektor Labs)

The Arduino is a real jack-of-all-trades. You can use this inexpensive microcontroller board for all kinds of things thanks to the expansion connectors for which countless ‘shields’ are available. In this article we show you how you can realize a ‘sound-level traffic light’ using an Uno, together with an Elektor-developed multipurpose shield and a handful of common parts.

The book “Mastering Microcontrollers — helped By Arduino” [1] is an excellent starting point for anyone who would like to get started using microcontrollers and their programming, without the requirement for a lot of prior knowledge. The book also describes several different hardware examples, which you can usually build on a small piece of experimenting board. In the second edition of the book an extra chapter was added, which contains even more examples. A universal shield was developed for these examples (no. 129009-1), which is available from the Elektor Store, either by itself or as part of a kit. This can be used to quickly build the various circuits.

This circuit board offers space for an LCD, two pushbuttons, two power transistors, temperature, air pressure and humidity sensors, a microphone preamplifier, an LDR, an IR sensor and a remote control receiver. In addition there are two LED outputs and a buzzer, and there is a protected and filtered analog input. Furthermore, there are also two protected digital inputs and outputs, which can also function as a serial port.

Arduino is a veritable multitalent that fits a thousand and one applications complex and simple!

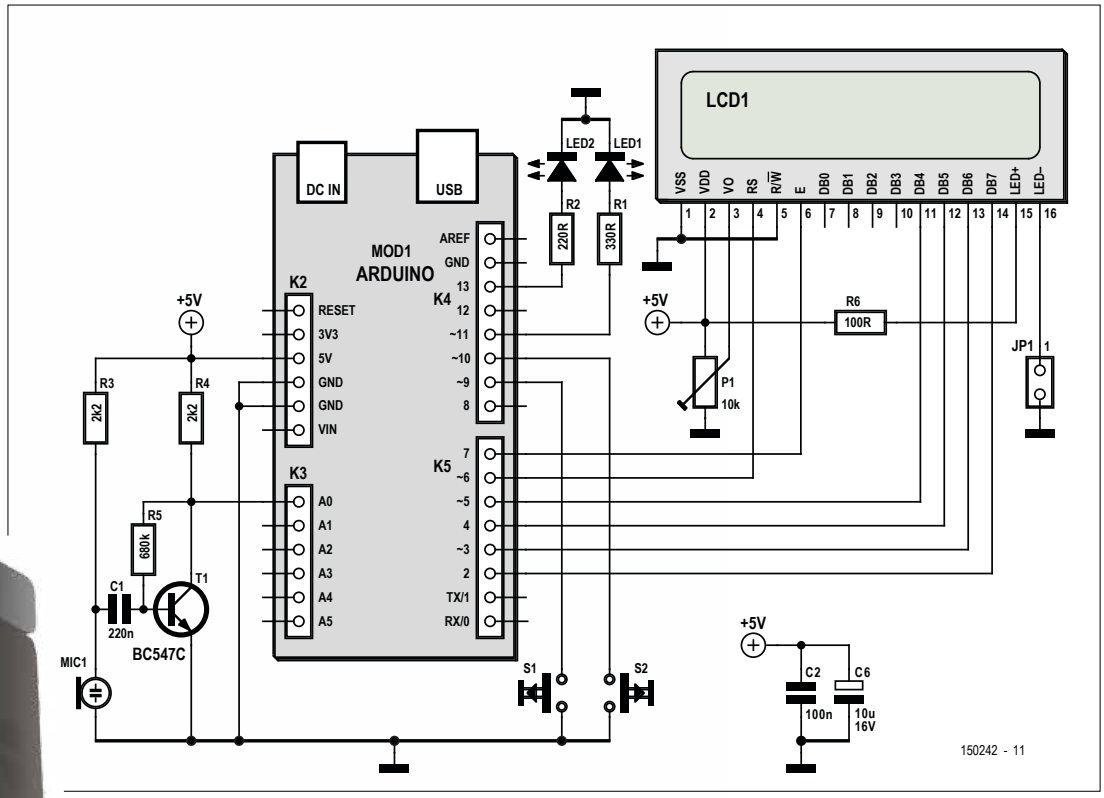


Figure 1. The entire circuit comprises only a few components which can all be fitted on the multipurpose shield.

Not everything can be fitted on the circuit board at the same time, but it still accommodates many of them simultaneously.

The idea behind this circuit was to use a few LEDs to make a kind of sound-level traffic light, where the colors of the traffic light indicate different sound levels. In this case red means 'too loud'. Handy for a concert or when the neighbors are having a loud party again.

It is, of course, possible to realize such a circuit entirely using only discrete components, but here we will show you how you could build one using an Arduino and a multipurpose shield. Besides the LEDs there is also an LCD that operates as a VU meter and with a few push-buttons it is easy to set the different thresholds. This is an excellent project to gain some programming experience with an Arduino!

The circuit

On the multipurpose shield 129009-1 there is space available for a microphone preamplifier. This can be used by the Arduino to react to sound. The microphone is an electret type and the amplifier is a classic single-transistor stage with a gain of about 1000 (see **Figure 1**). The output of the amplifier is connected to the Arduino's analog input A0.

Originally it was intended to use two LEDs for the indication, a green one and a red one. The shield also has space for these. However, as I was building this circuit I realized that I did not have a usable, 5-mm, green LED on hand. But I did happen to have a bi-color red/green LED with a common cathode, so I used that instead. Fitting it in the position of LED2, the third connection can then easily be connected to LED1, because the buzzer, which shares an output with LED1, is positioned close to LED2 on the circuit board. Thanks to the bi-color LED I now have a third color available, namely orange. So this ultimately became a 'real' traffic light.

Component List

Resistors
Default: 5%, 0.25W
R1 = 330Ω
R2 = 150Ω
R3,R4 = 2.2kΩ
R5 = 680kΩ
R6 = 100Ω
P1 = 10kΩ preset, horizontal

Capacitors
C1 = 220nF
C2 = 100nF
C6 = 10µF 50V

Semiconductors
LED1 = green, 5 mm
LED2 = red, 5 mm
Alternatively, LED2 = bicolor, common cathode
T1 = BC547C

Miscellaneous
S1,S2 = pushbutton, make contact, 6x6mm
MIC1 = electret microphone, 6mm
JP1 = 2-pin pinheader, 0.1" pitch
K2,K3 = 6-pin pinheader, 0.1" pitch
K4,K5 = 8-pin pinheader, 0.1" pitch mm
LCD1 = 2 x 16 characters, with backlight
K6 = 16-pin pinheader, 0.1" pitch
For LCD: 16-pin connector, 0.1" pitch
Jumper
PCB # 129009-1

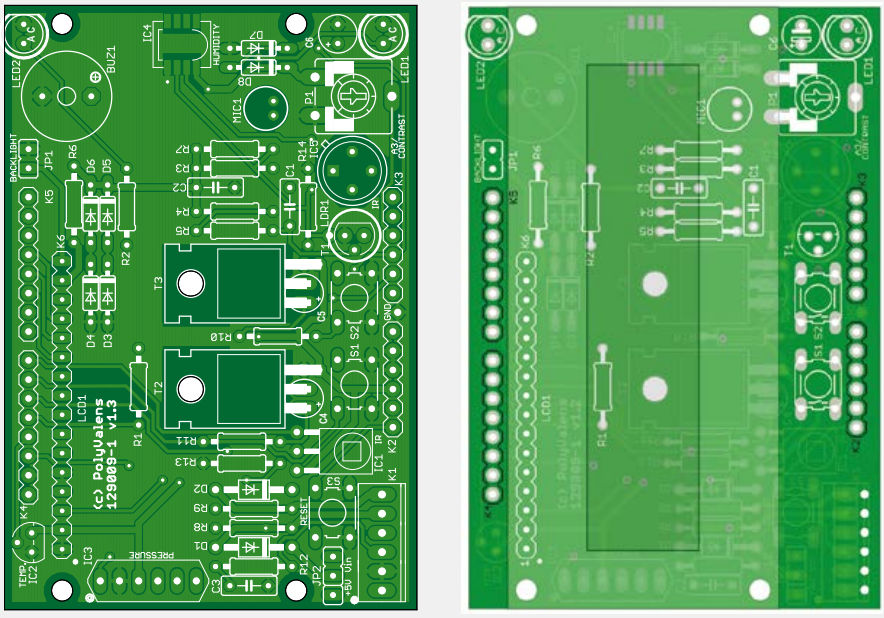
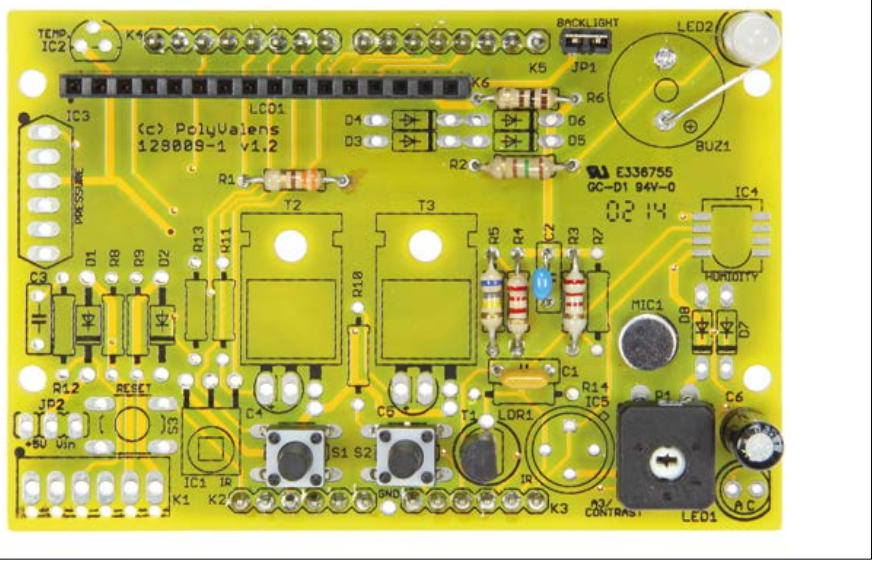


Figure 1. The entire circuit comprises only a few components which can all be fitted on the multipurpose shield.



Figuur 3. De opgebouwde print, het LCD is voor de duidelijkheid nog niet op de connector gestoken.

Because we don't have a sound-level meter at the Elektor lab any more, I was not able to calibrate the circuit in dB. To make the circuit as general-purpose as possible, it would be very convenient if the user could adjust the thresholds themselves. For this reason I have added an LCD in 4-bit mode to the circuit, as well as two pushbuttons. The display can show the raw values as well as the processed ones, and using the buttons a menu can be navigated. P1 is used for the contrast setting of the LCD and a jumper on JP1 can be used to turn on the backlight. All these components will also fit on the multipurpose shield without any problems.

The software

The program, an Arduino sketch, of course, ultimately became quite extensive in order to demonstrate a number of different tricks and techniques.
The sound level is measured with *analogRead*. This happens four times in a row, after which the values are averaged. The reason for this is not so much to filter the noise, but to slow down the program. As it happens, the function *analogRead* is very slow and four of these, one after the other, reduce the loop time of the main loop down to about 2 kHz. This is then also the sample rate of the sound signal.
The peak amplitude of the measured signal is determined with the aid of a leaky integrator, so that the peak value follows the sound level, but reduces to zero when the sound is gone.
The peak value is filtered by a first-order low-pass filter with a cut-off frequency of about 0.1 Hz, to build in some delay. For this a genuine floating-point IIR filter-algorithm has been used (with explanation), so that you can see how this is actually implemented in practice.
Once per second the actual filtered peak value is compared with the configured thresholds and the result is shown on the display and via the LEDs. During this update the sound level is not sampled, because it appeared that the switching of the LEDs had an impact on the analog input of the Arduino.

There are three thresholds for the sound level: green, orange and red. 'Off' is no or very quiet noise (background noise), green is a little bit louder (a quiet bit of background music, default value = 100), orange is a little louder still (singing along,

default value = 200) and red is very loud (yelling along with the music, deafeningly loud music, default value = 300).
The display shows the sound level as a numeric percentage of the highest (red) threshold. This value is also displayed as a horizontal bar on the second line. This makes it possible to read the display from a distance, but is also used as a demonstration of how you can program such a thing. The display also indicates the peak value as a number from 0 to 1023. This is very useful when selecting the thresholds.
Setting the thresholds is done by simultaneously pressing the two pushbuttons. There now appears the option of setting the green threshold (0 – 1023). Pressing both buttons again brings us to the orange threshold and once more gives the red threshold and finally the setup mode is exited by pressing both buttons for the last time. The selected threshold levels are stored in the EEPROM of the microcontroller.

Construction

Figure 2 shows the layout of the multipurpose shield. For a schematic of all the individual circuits that can be built on this board we refer to the book [1]. Only those components that have to be fitted for this particular application are shown in **Figure 2**. Note that connectors K2 through K5 have to be mounted on the solder side. The LCD is connected to the shield using a 16-way pin header and a corresponding connector. The manner in which the bi-color LED is connected can be clearly seen in **Figure 4**.
The built-up shield is plugged on top of an Arduino Uno, after this it can be loaded with the ino-file that is available as a free download from [2].

Conclusion

As you can see, this circuit demonstrates all kinds of things:

- Sound-level measurements with an Arduino
- Generating an alarm
- Digital filtering
- Driving an LCD
- Driving LEDs
- Reading switches
- Implementing a Setup menu
- Displaying a bar graph
- Using LCD custom characters
- Storing and then retrieving values from the EEPROM

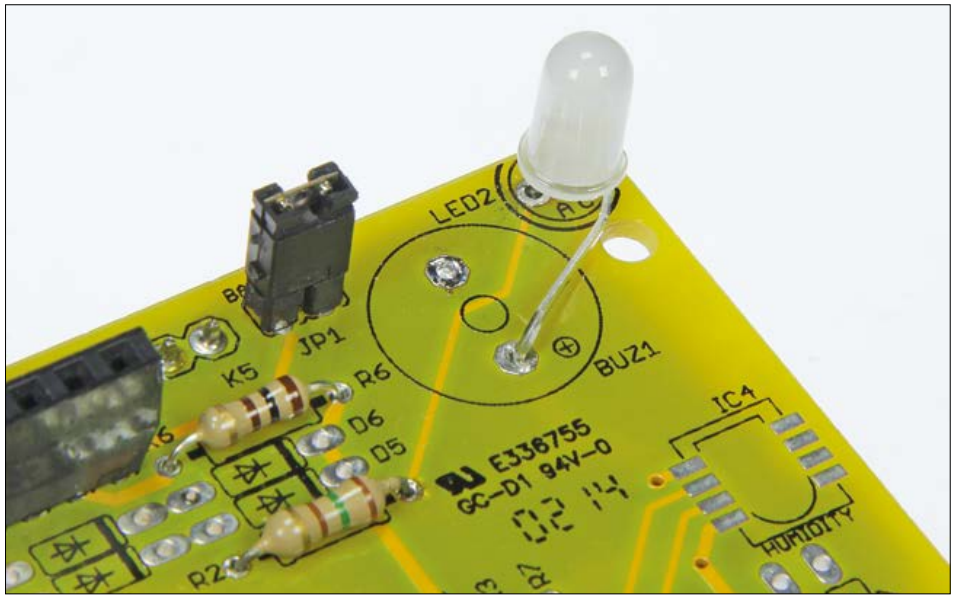


Figure 4. This is how the bicolor LED is mounted.
The lead for the green LED goes to the + connection for BUZ1.

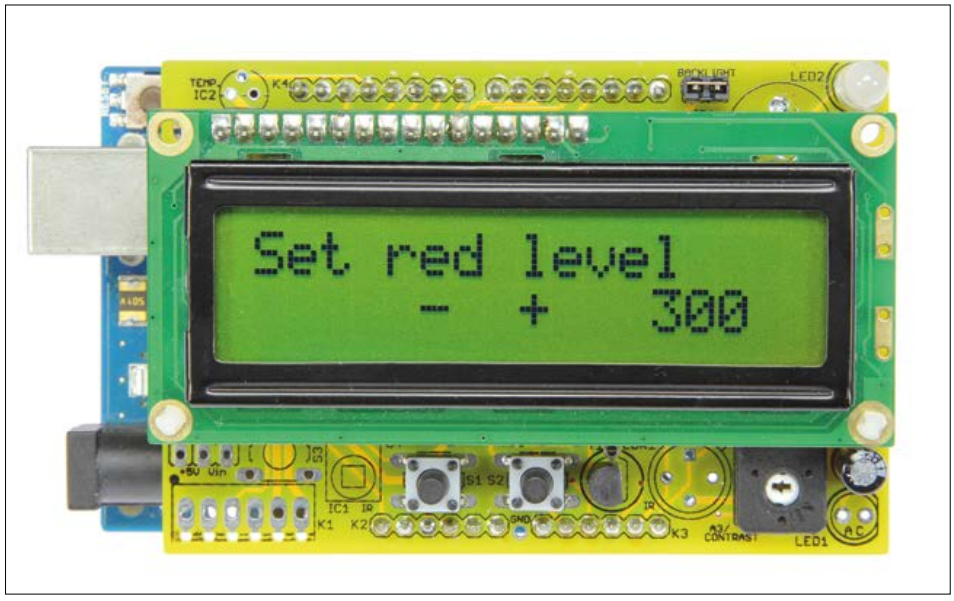


Figure 5. Setting the threshold level at which the red LED turns on.

As an exercise for the reader there is still the serial port. This can be used to send the measured values to a PC or so, where they can be logged. Handy as evidence in the event of a protracted disturbance. ⏮

(150242)

Web Links

[1] www.elektor.com/mastering-microcontrollers-helped-by-arduino

[2] www.elektormagazine.com/150242