Three-Dimensional Light

Programmable LED matrix with 125

Jerry Jacobs (The Netherlands)

Everyone will have encountered a 2D LED matrix at some time, but the version described here is of a completely different calibre: namely five matrices stacked together into a cube; a true 3D matrix therefore, every LED of which can be switched on and off individually.

Hardware specifications

- 125 LEDs in a special 3D matrix
- ATMEGA32 microcontroller running @ 1 MHz internal clock
- 10-way ISP-connector for reprogramming
- 5 transistors for switching the layers
- 25 transistors for switching the columns

Most people are fascinated by flashing LEDs. But these are usually limited to just a few LEDs or only a small display. This LED cube is something entirely different however, because there is an additional dimension for even more LEDs. Here we present a 3D display of LEDs, each of which can be controlled individually.
This magnificent cube has at its heart an Atmel AVR microcontroller.
These controllers are easy to obtain and superb open-source tools are available. Not only for Windows, but also for the Linux and Mac operating systems.

Operation

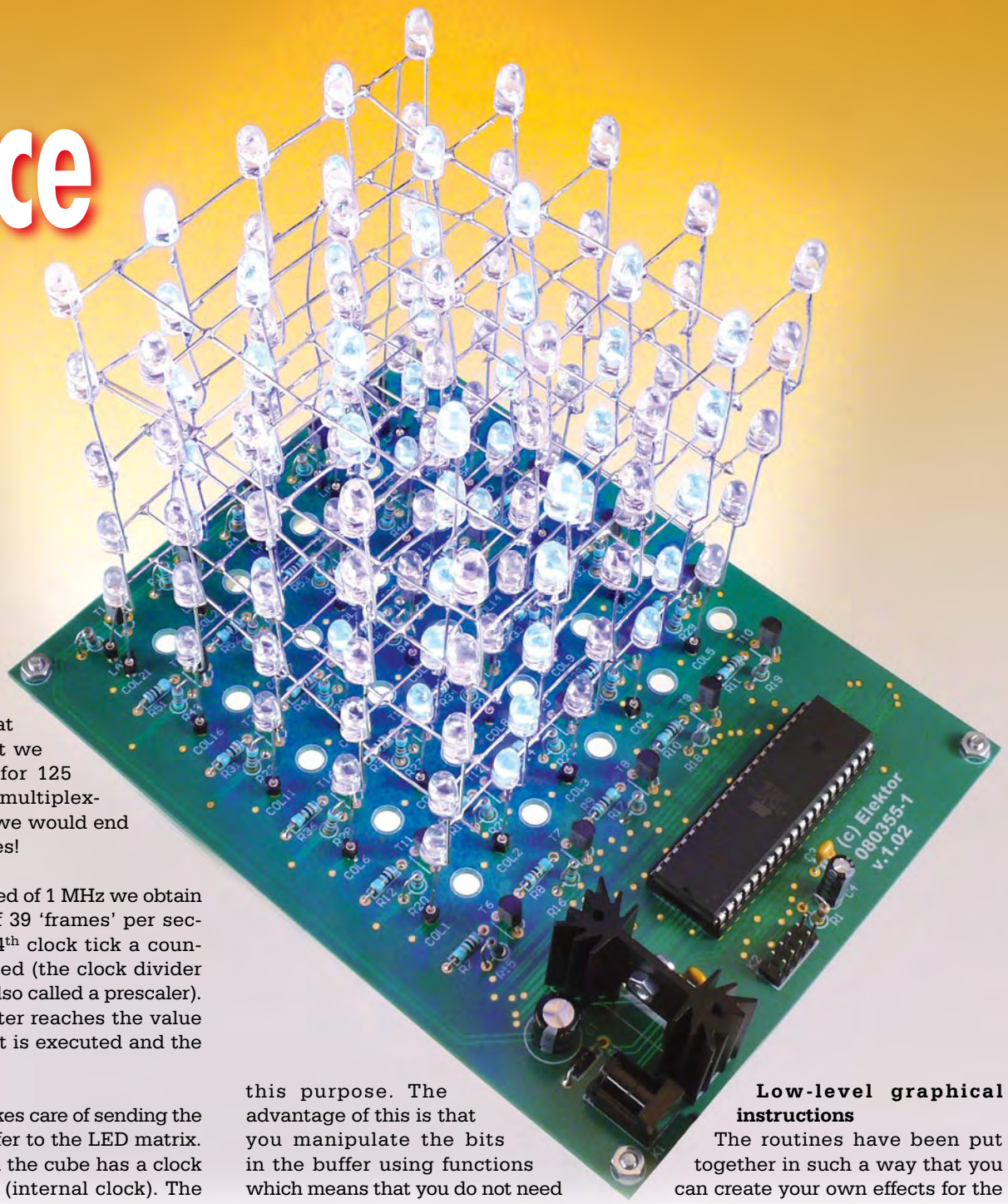
You would expect that with 125 LEDs in the cube you would need a large number of wires to be able to control them individually, but that is not so. A lot of wires can be saved because the signals are multiplexed. One 'layer', that is all 25 LEDs which are all at the same height in the columns, can be controlled with a single wire. This results in a total of 26 signal wires. If each LED were to be connected individually then 50 wires would be required.

To turn an LED on we switch the positive voltage to the desired layer on and select the appropriate column.

Table 1. Layer drivers & column drivers

PORT A							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
Column 8	Column 7	Column 6	Column 5	Column 4	Column 3	Column 2	Column 1
PORT B							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
Column 25	–	–	Layer 5	Layer 4	Layer 3	Layer 2	Layer 1
PORT C							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
Column 16	Column 15	Column 14	Column 13	Column 12	Column 11	Column 10	Column 9
PORT D							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
Column 24	Column 23	Column 22	Column 21	Column 20	Column 19	Column 18	Column 17

Source LEDs



Our cube has 5 layers and 25 columns. That means then that we have 30 wires for 125 LEDs. Without multiplexing the signals we would end up with 250 wires!

With a clock speed of 1 MHz we obtain a refresh rate of 39 'frames' per second. Every 1024th clock tick a counter is incremented (the clock divider used for this is also called a prescaler). When this counter reaches the value of 5, an interrupt is executed and the counter is reset.

This interrupt takes care of sending the value in the buffer to the LED matrix. The controller in the cube has a clock speed of 1 MHz (internal clock). The software updates every 5 counter clock ticks. The internal clock is divided by the prescaler to become the counter clock. This results in a refresh rate of 195 Hz for the entire cube. Since we have five layers we divide this rate by five and arrive at 39 Hz per layer.

Software

The software (firmware) is written in C and can be compiled with avr-gcc [1]. It is also documented in such a way that it can be viewed as a website. This is made possible because of Doxygen [2].

Buffer

Because it is quite complex to get the cube to display and arbitrary pattern in a simple way, a buffer is used for this purpose. The advantage of this is that you manipulate the bits in the buffer using functions which means that you do not need to write to outputs directly yourself. That's the job of the interrupt routine. The buffer, just like the cube, has multiple dimensions, so you can 'draw' a pattern in the buffer and the interrupt routine takes care of the rest.

Interrupt

The interrupt routine in the code, as already mentioned, ensures that the pattern on the cube is refreshed 39 times per second.

This interrupt routine writes the values, which you wrote to the buffer using another function, from the buffer to the appropriate ports and puts the bits in the correct place. We use bit masks which ensure that we only look at the corresponding bits which have to be low or high at the pins..

Low-level graphical instructions

The routines have been put together in such a way that you can create your own effects for the cube. In **Table 1** you can see exactly which pin connects to where on the cube. This makes it easier for beginners to get started quickly without the need to immediately understand how bit-masks, bit-shifts and other complicated functions work. These low-level instructions are defined in draw.h, this is the interface file for the instructions to control individual columns, layers, rows, etc. Below are a few examples which show how these functions can be used.

For controlling a row on a certain layer we use

```
set_row(ROW_1, LAYER_1);
clear_row(ROW_1, LAYER_1);
toggle_row(ROW_1, LAYER_1);
```

The following functions can be used

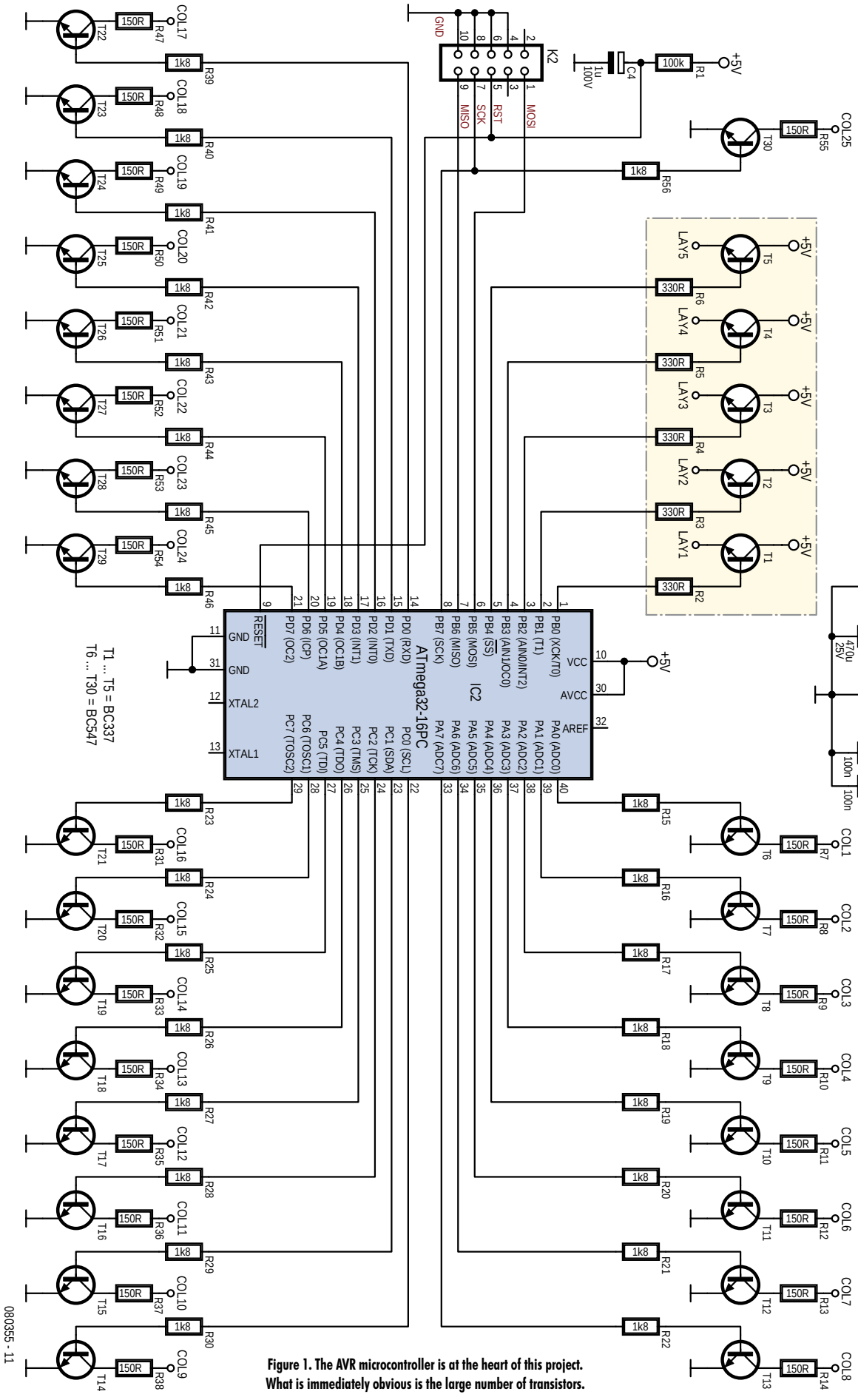


Figure 1. The AVR microcontroller is at the heart of this project. What is immediately obvious is the large number of transistors.

for switching a column on and off:

```
set_column(COLUMN_1, ON);
set_column(COLUMN_1, OFF);
```

We use handy names such as ON, OFF and COLUMN_1. These are defined names, also called macros, which have a fixed value. For example, ON has the value 1 and OFF the value 0.

All these functions can be used one after the other to draw the desired picture. Because we cannot show all the source code and examples here, you can download them from the Elektor website.

Hardware

Normal 'through-hole' components are used to build the cube. The PCB is still very compact nevertheless.

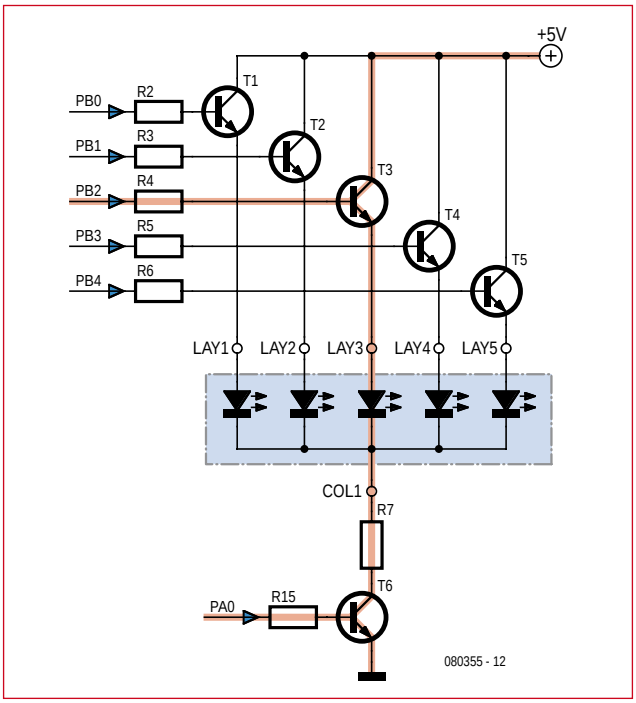


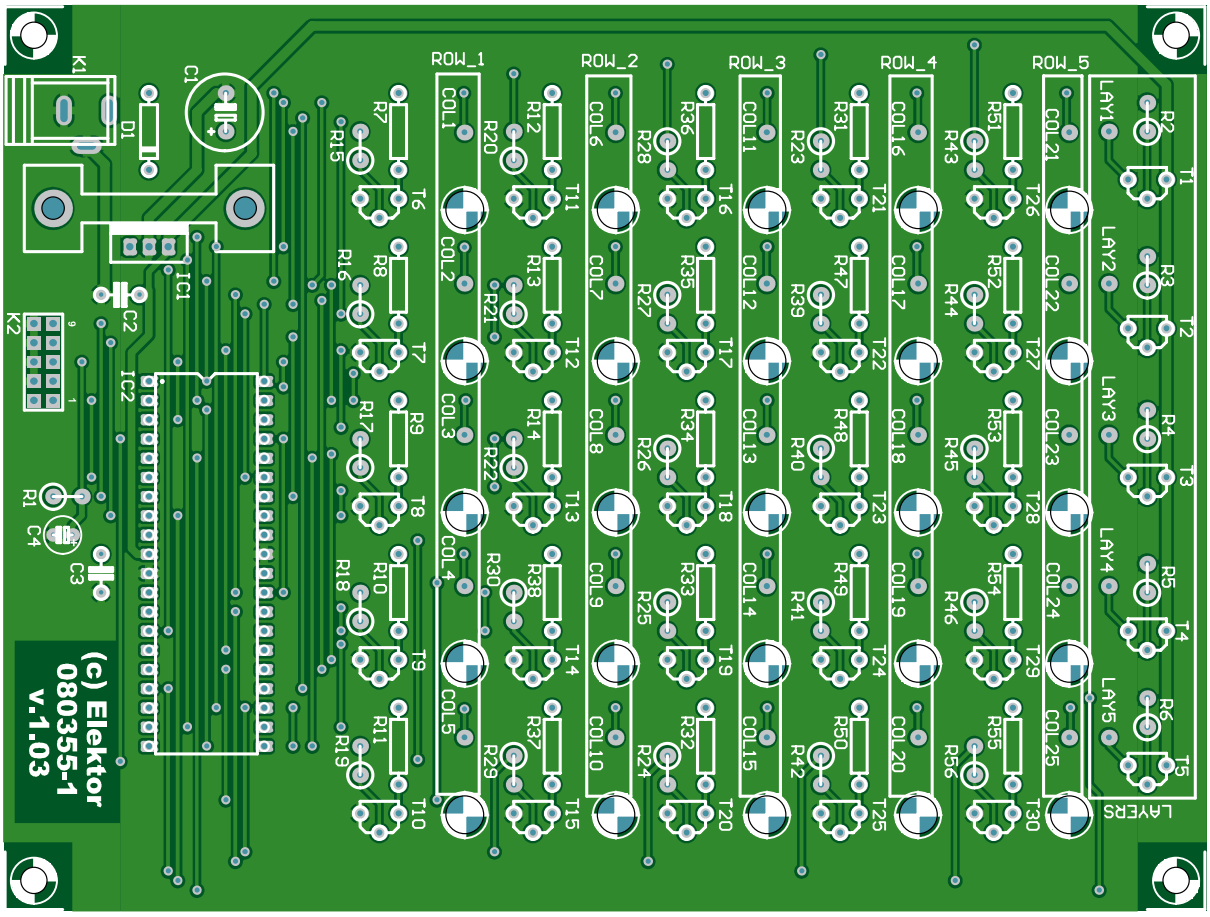
Figure 2. In this figure you can see the path that the current takes through the middle LED in the first column.

A mains adapter supplying at least 9 volts and rated 600 mA can be used as the power supply. IC1 (a 7805) provides the voltage regulation and protection diode D1 guards against accidental reverse polarity.

Transistors T1 through T5 are used to connect the 5-volt power supply voltage to the different layers. The columns are switched with T6 through to T30. The return current flows through these latter transistors to ground and completes the circuit through the LEDs (see **Figures 1** and **2**).

The value of the column resistors depends on the voltage drop across the LEDs. We make the assumption that each LED requires 20 mA. This is also the current that will flow through the entire column. The power supply for

Figure 3. The component overlay of the PCB shows the neatly arranged layout of the parts.



COMPONENTS LIST

Resistors
R1 = 100kΩ
R2-R6 = 330Ω
R7-R14, R31-R38, R47-R55 = see text
R15-R30, R39-R46, R56 = 1kΩ8

Capacitors
C1 = 470µF 25V
C2,C3 = 100nF
C4 = 1µF 100V

Semiconductors
IC1 = 7805
IC2 = Atmega32
D1 = 1N4001
T1-T5 = BC337
T6-T30 = BC547
125 LEDs for cube

Miscellaneous
10-way boxheader (2x5), 2.54mm lead pitch
10-way SIL pinheader (IC socket)
Heatsink, TO-220, 5°K/W for IC1
4 off M3x5 bolt with 10-mm long hex spacers
Mains adapter socket for 2.5mm pin diam.
PCB, order code 080355-1 from the Elektor SHOP.

each layer is 5 V. The calculation for the resistance is then approximately:

$R = (5\ V - U_{LED}) / 20\ mA$

Assuming the voltage drop across the transistors can be neglected.

For reprogramming of the cube the ISP interface is connected to K2. Table 1 shows which column and which row is connected to which bit.

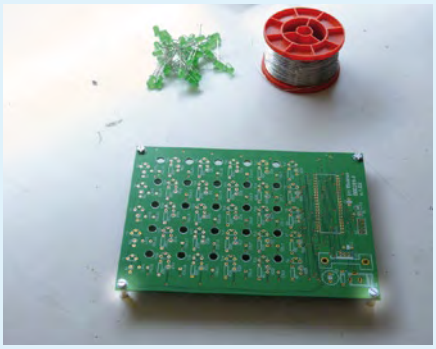
(080355)

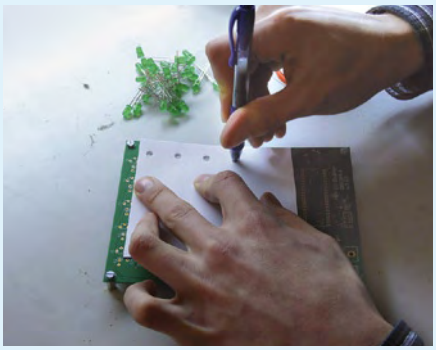
Internet Links

[1] AVR-GCC Tool chain:

- For Windows: http://winavr.sourceforge.net
- For Mac: www.obdev.at/products/avrmacpack
- For Linux: depends on the distribution

[2] Doxygen:
www.doxygen.org





Get cracking!

Step # 1

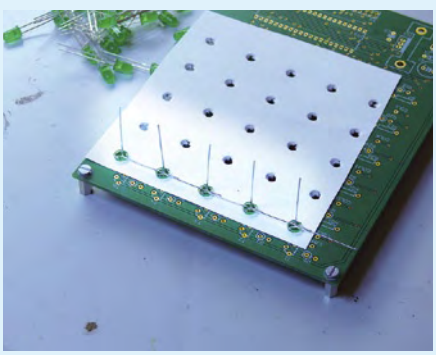
The 'pillars', made from PCB stand-offs, are mounted on the PCB first. The holes in the PCB serve as a guide so that each layer of LEDs can be soldered neatly. Use a sheet of paper so that the LEDs can be fitted tightly in the holes. It is best to pre-punch the holes first, using a ballpoint pen, for example.

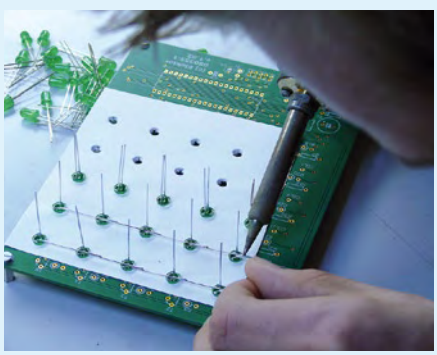
Step # 2

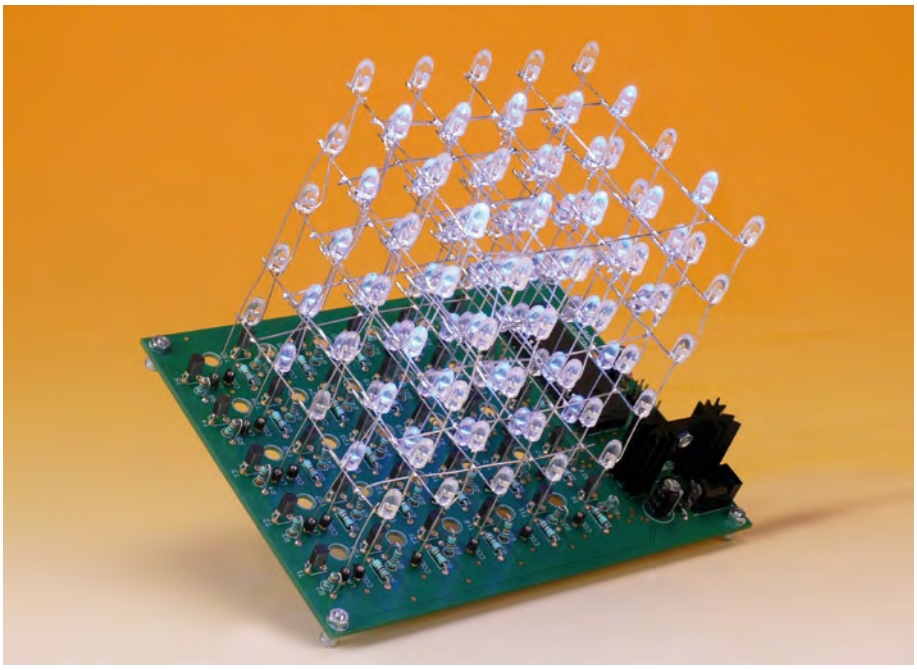
Fit five LEDs in the top row with the anodes (long pin) at the top and the cathodes (short pin) at the bottom. Next, bend the anode lead of the first LED to the left and move on to the second LED. The second LED is then soldered to the first LED. Repeat this until the entire row is complete. For each layer, five rows are required. We therefore repeat the whole story for the second to the fifth row. Once all five rows are complete, the bent anodes are all connected together with two perpendicular connections. After that the whole layer can be pushed out of the guide in one move using a flat plate.

Step # 3

Once all five layers are finished, they are soldered together until they form the final shape. You do this by placing a layer on the PCB. On the next layer, bend the ends

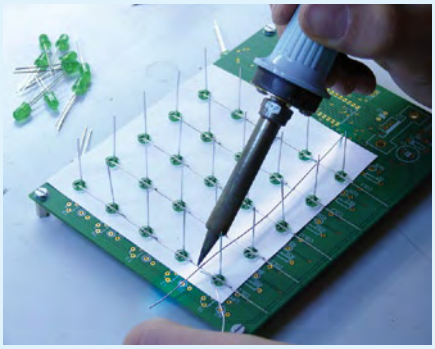


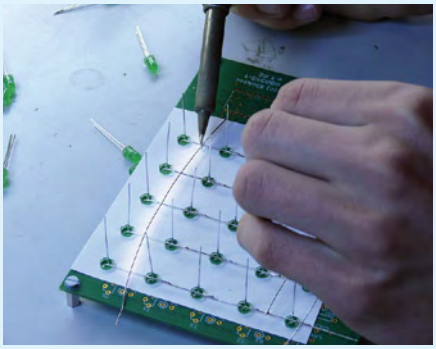


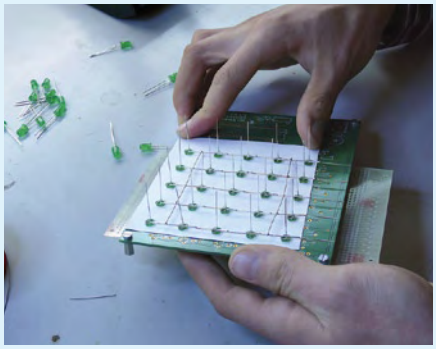


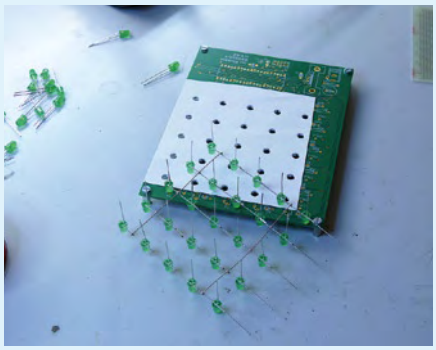
The author

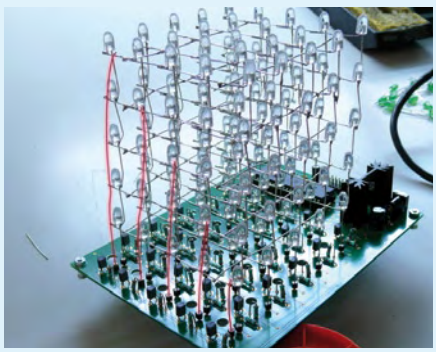
Jerry Jacobs (1989) is a third-year Telecommunications/ICT student at ROC Leeuwenborg College in Sittard, The Netherlands. From a young age Jerry's been fascinated by the inner workings of computers and electronics. He is also a big Linux fan. The project described in this article was designed and produced during Jerry's traineeship period at Elektor.











of the 25 column leads over by about 3 mm, to reach around the LEDs of the previous layer. The second layer is subsequently placed above the first layer and we solder this layer at every connection, with a uniform spacing between the layers and the LEDs neatly lined up.

Step # 4

All the other components are now mounted on the PCB. Make sure that the BC337 and BC547 transistors are not mixed up!
The voltage regulator with its heatsink is mounted last.

Step # 5

The final step is connecting each of the layers to their corresponding transistors. T1 is connected to the bottom layer and T5 to the top. Use tinned copper wire for this.

Step # 6

The example software can be programmed into the chip. This software, together with the source code, can be downloaded from the Elektor website. You can also order the bare PCB from the SHOP section.