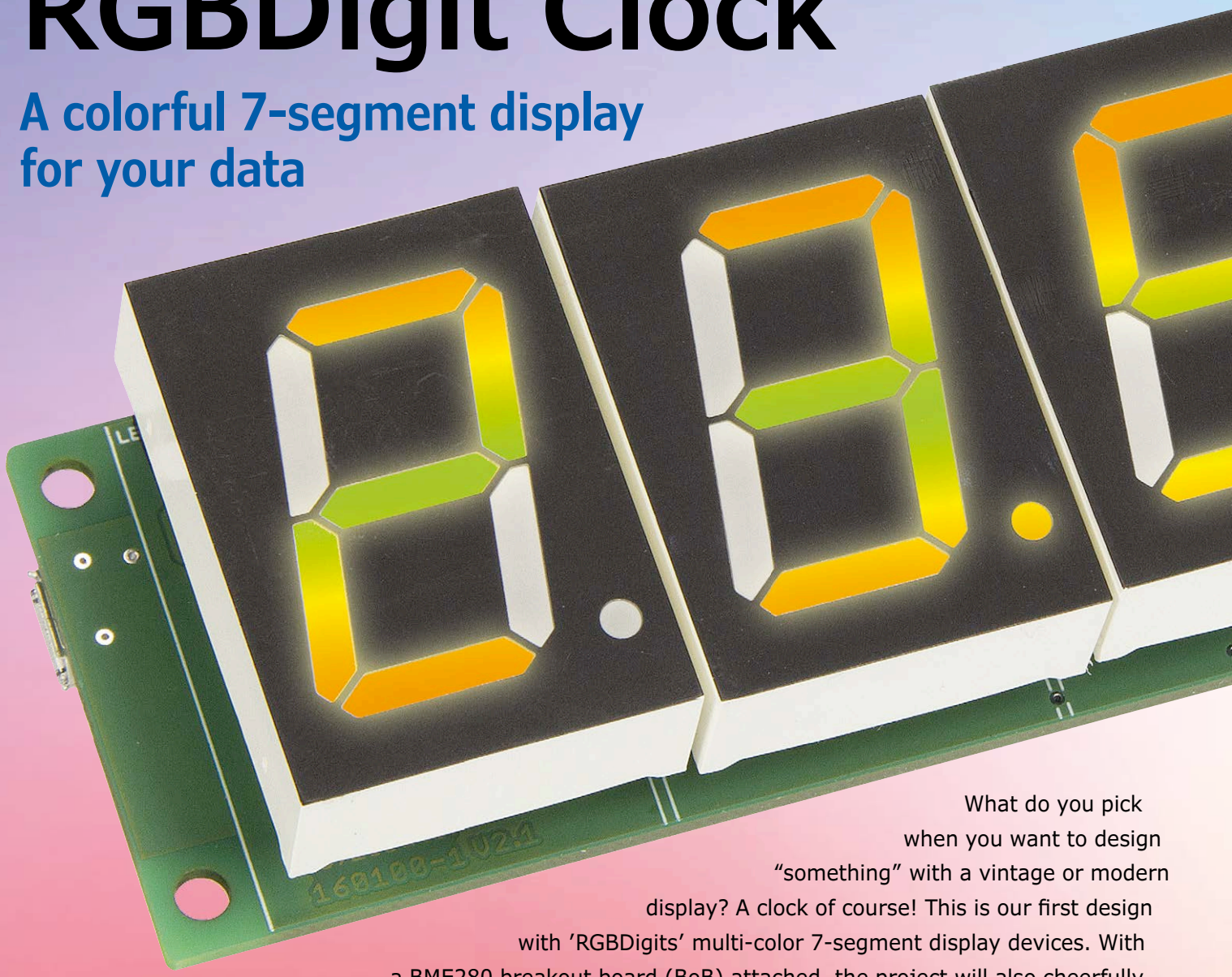
eLABS PROJECT

RGBDigit Clock

A colorful 7-segment display for your data



What do you pick when you want to design "something" with a vintage or modern display? A clock of course! This is our first design with 'RGBDigits' multi-color 7-segment display devices. With a BME280 breakout board (BoB) attached, the project will also cheerfully display temperature, humidity and air pressure.

By **Coen de Bruijn** (The Netherlands)

The clock is controlled by an ESP12 module, which enables synchronizing the clock with an Internet time server, change the clock settings from any mobile device or computer in the network, or transmit sensor data via Wi-Fi.

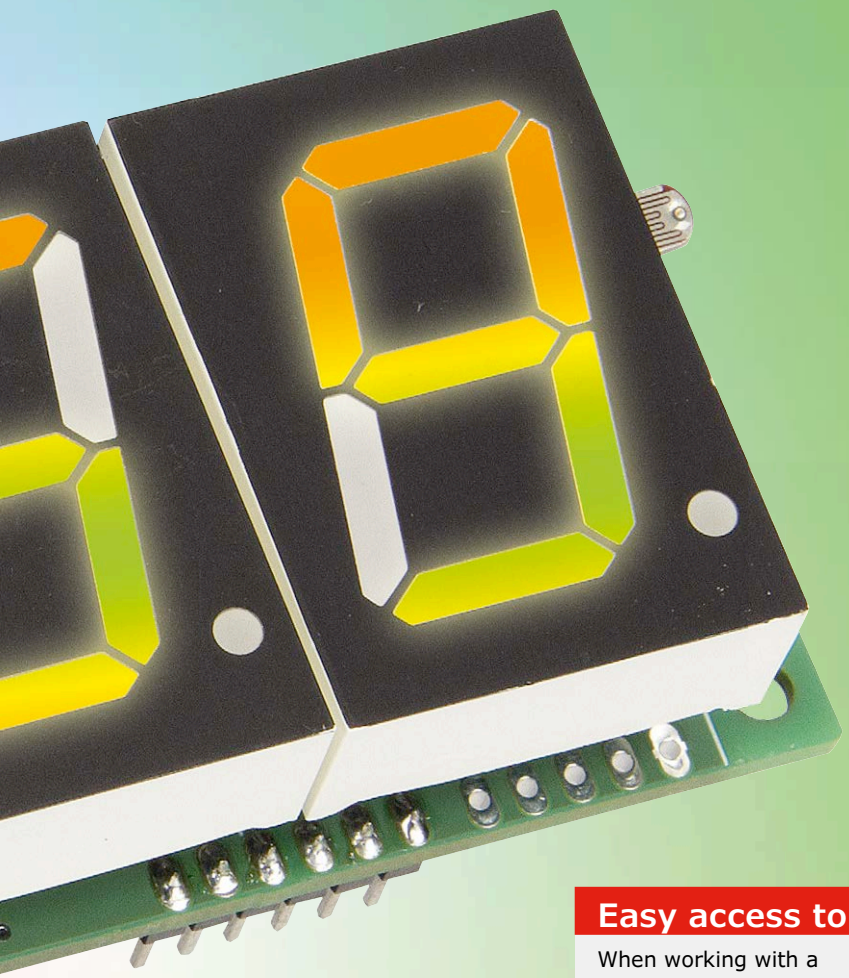
What are they?

RGBDigits are manufactured by a Dutch company with the same name. Every display contains eight 5050 RGB NeoPixel LEDs (seven segments plus decimal point) with integrated driver chips, which allow the user to control color and brightness of every segment/DP individually via a 3-wire bus (VCC, GND an DATA). Up to 10 displays can be daisy chained via their DATA IN and DATA OUT pins.

Each primary color LED of a NeoPixel can be set to 256 levels of brightness, resulting in a whopping 256 x 256 x 256 = 16,777,216 colors for every segment/DP.

How the circuit works

Figure 1 depicts the full circuit diagram for this project. The clock is 5-V powered via micro-USB connector K1. The power supply is protected by 2-A PTC resettable fuse F1, with Schottky diode D1 acting as a reverse polarity protection. IC1 is the main 3.3-V voltage regulator, and IC6 is the 3.3-V supply for the Qtouch touch sensors IC4 and IC5 (note separated supply to prevent interference). The two touch sensors Button0 and Button1 are used to control the display



PROJECT INFORMATION

Test & Measurement

RGB LED Clock Sensors

entry level
➡ intermediate level
expert level

3 hours approx.

Computer with Arduino IDE

€100 / $105 / £85 approx.

modes of the clock. S1 and S2 are only used for resetting the clock and for flashing the firmware. IC2 serves as an unidirectional level shifter between the 3.3 V at the ESP-12E side and the V+ (approximately 4.5 V) power supply of the displays. EEPROM IC3 is used to retain the clock settings after power off, while LDR R1 is used to dim the displays in the dark.

A 3.3-V FTDIstyle cable (or another 3.3-V USB UART) can be connected to K2 to flash the ESP-12E module; it can also be used for debugging applications.

K3 makes the 3.3-V power and three general purpose I/O pins of the ESP12E accessible for your own developments and/or future expansions.

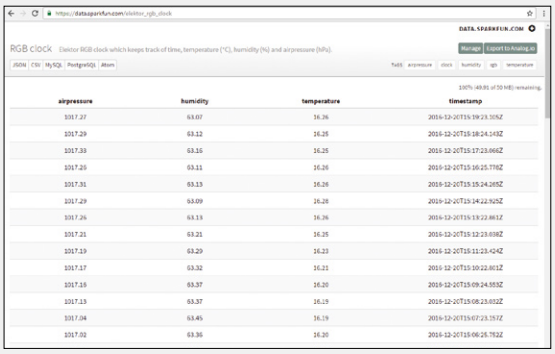
A BME280 BoB (Store #160109-2, **Figure 2**) can be connected to K4. The clock will run without it, but of course there will be no sensor data available. An onboard BME280 was no option — heat from the displays will affect the data.

Flashing the sketch

The ESP12E Wi-Fi module (which contains an ESP8266) is supported by the Arduino IDE. So in order to flash the

Easy access to your sensor data

When working with a project like this, or with any IoT-ish project for that matter, it is convenient at times to have access to a data storage server. The techies at Sparkfun recognized this and set up a service that is free to use for everyone. Accessible through *data.sparkfun.com* is a server that will store (and share!) your data in a convenient way. It works like a kind of database and allows you to store and view your data (and that of others) in a table (see **screenshot**).



First, create a datastream by clicking *create*. Fill in the fields and you'll receive a public key, a private key and a delete key. These keys are used to send data to the stream or to delete data. Our clock will need the public as well as the private key to send data to the Sparkfun data server.

From here you can export your data with the click of a button (top right) to *analog.io*, an online service for displaying (IoT) data. At analog.io you get to see a nice graph of your data and/or measurements.

software into the module, first select the ESP8266 (NodeMCU) in the Arduino IDE. Instructions on how to add the ESP8266 (NodeMCU) can be found at [1]. Follow the steps described carefully before you proceed.

Download the software archive for this project (160100-11.zip) from our website [2]. Unpack the zip file into an Arduino sketch folder on your hard drive, while ensuring subfolder DATA (containing the ESP12's webpage) is preserved in this folder.

Now follow these steps:

- connect a 3V3 (!) FTDI cable between your computer and K2 of the RGB Clock;
- connect a 5-V power supply to micro USB connector K1;

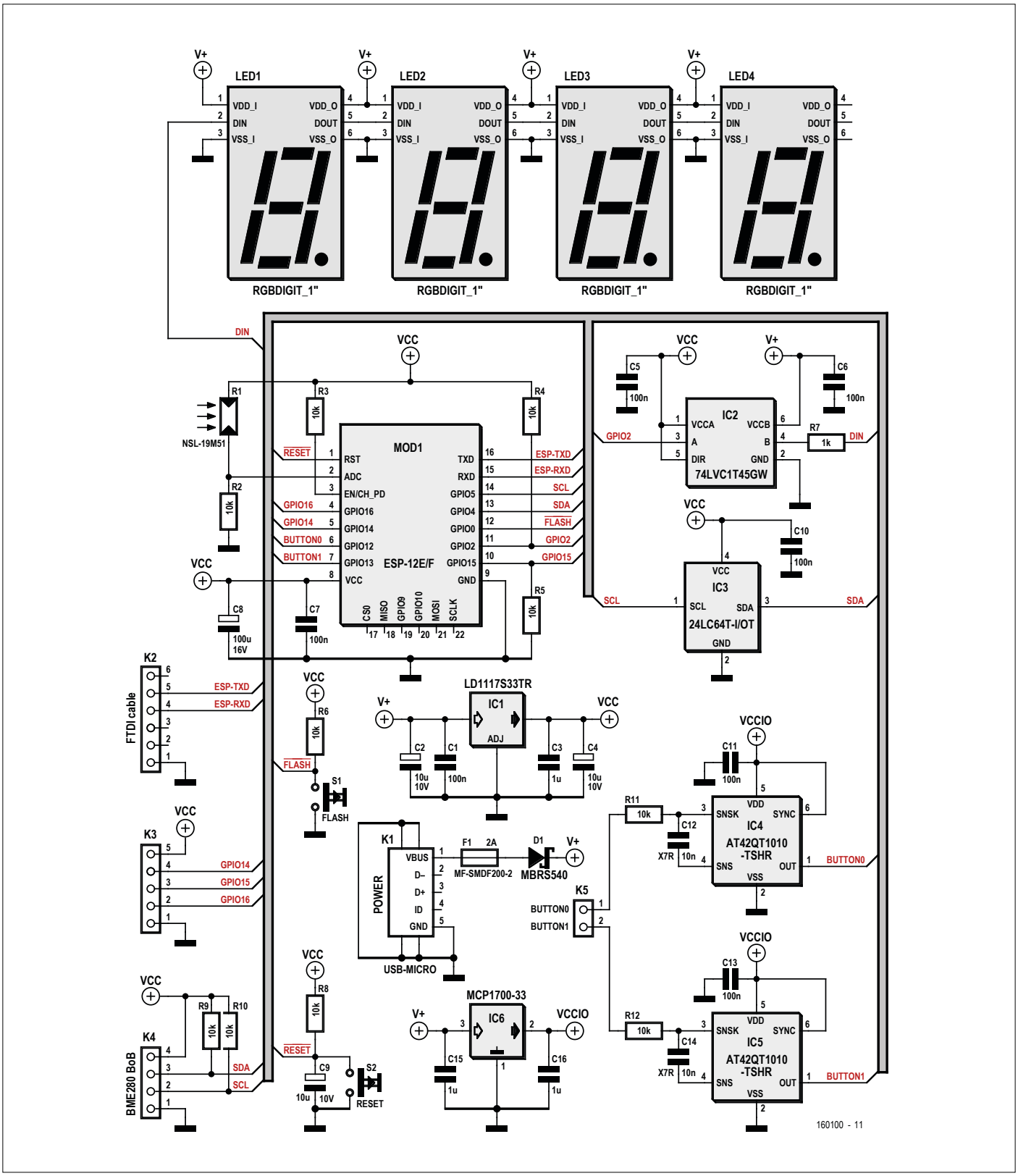


Figure 1. A fair number of connections need to be made. To keep the schematic readable a lot of those connections are bundled in a 'bus'.

- open the Arduino IDE;
- in the menu Tools → Board, select Board NodeMCU 1.0 (ESP-12E Module);
- in the menu Tools → Port, select the COM port associated with the FTDI cable.
- in the menu File → Open, select sketch 'esp12_rgb_clock.INO', included in the download from our website;
- on the RGB cClock board, press and hold reset button S2, press and hold flash button S1, release S2 and then release S1 to enable flashing of the firmware;
- compile and upload the sketch (CTRL+U).

Note that if the compiler exits with an error, this is probably caused by one or more missing Arduino libraries. The compiler will report the name of the first missing library, but you should also check whether all libraries used in the INCLUDE directives in the 'esp8266_rgb_clock' sketch are installed in your Arduino IDE before you start the compiler.

Uploading the clock's webpages

All settings of the clock can be configured via Wi-Fi. The initial connection should be made via the ESP8266's Access Point (AP). Connect your smartphone/tablet/laptop (or PC, if it has a direct Wi-Fi connection) to the open network called 'RGB Clock' that will appear in the list of available Wi-Fi networks after flashing the clock's firmware. However, the clock's webpages must first be uploaded to the ESP8266 before you will be able to get easy access to the settings of the clock. On [3] you'll find instructions how to do this under 'Uploading files to the file system':

- download the tool "Arduino ESP8266 filesystem uploader" from [4];
- unpack the tool into the tools directory (the path will be similar to <home_dir>/Arduino/tools/ESP8266FS/tool/esp8266fs.jar);
- restart the Arduino IDE;
- re-open the "esp12_rgb_clock.INO" sketch (if it didn't open automatically).
- make sure you have selected the correct board and COM port and closed the Serial Monitor;
- press and hold reset button S2, then press and hold flash button

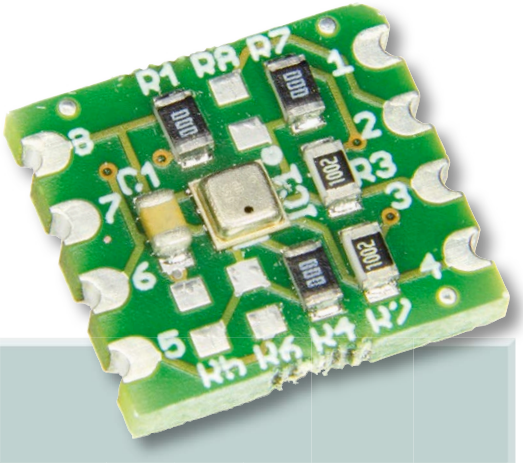


Figure 2. Our BME280 Break-out Board adds temperature, pressure and humidity sensors to the RGBDigit Clock.

COMPONENT LIST

Resistors

R1 = LDR NSL-19M51*
R2,R3,R4,R5,R6,R8,R9,R10,R11,R12 = 10kΩ
R7 = 1kΩ

Capacitors

C1,C5,C6,C7,C10,C11,C13 = 100nF, 50V, X7R, 0805
C2,C4,C9 = 10µF, 10V, tantalum, 1206
C3,C15,C16 = 1µF, 50 V, X5R, 0805
C8 = 100µF, 16V, 2312
C12,C14 = 10nF, 50V, X7R, 0805

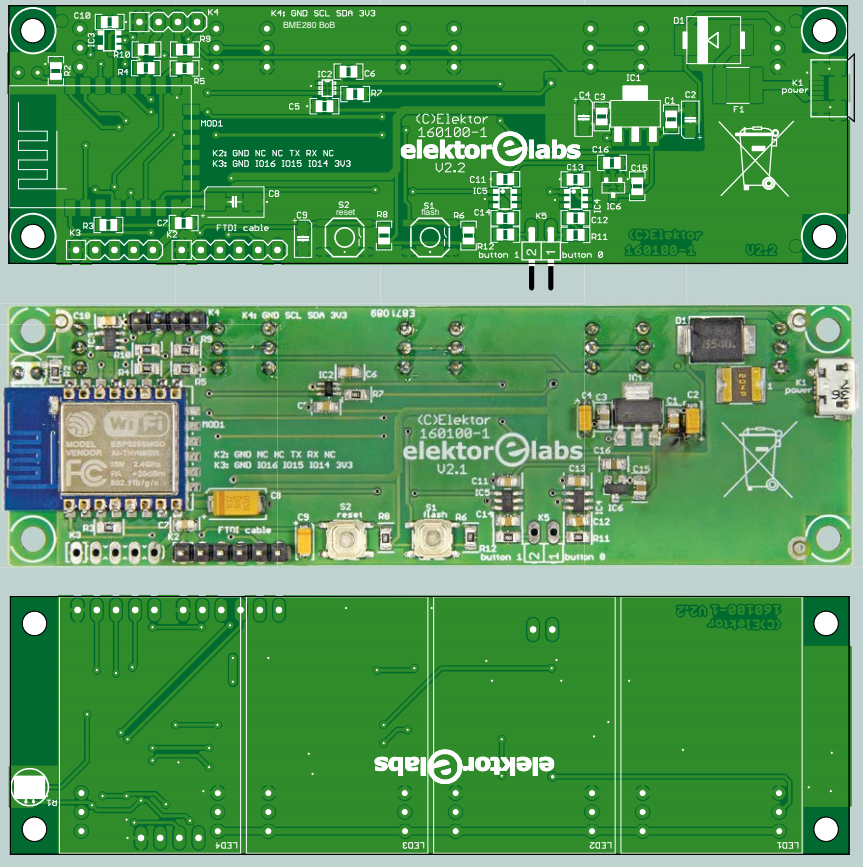
Semiconductors

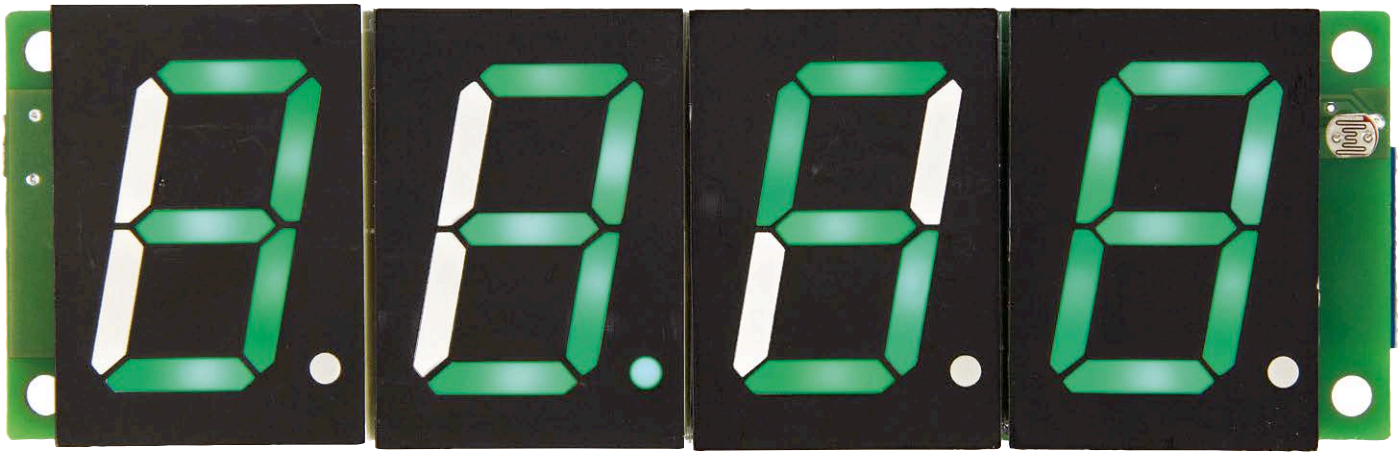
D1 = MBRS540
LED1,LED2,LED3,LED4 = 7-segment RGB display*
IC1 = LD1117S33TR
IC2 = 74LVC1T45GW
IC3 = I2C EEPROM 8K × 8 bit, type 24LC64T-I/OT
IC4,IC5 = Qtouch Touch Sensor type AT42QT1010-TSHR
IC6 = MCP1700T-3302E/TT

Miscellaneous

F1 = 2A PTC resettable fuse, type MF-SMDF200-2 (Bourns)
K1 = Micro-USB type B receptacle, underside mount
K2 = 6-pin SIL pinheader, 0.1" pitch, right angled
K3 = 5-pin SIL pinheader, 0.1" pitch, right angled
K4 = 4-pin SIL pinheader, 0.1" pitch, straight
K5 = 2-pin SIL pinheader, 0.1" pitch, right angled
MOD1 = ESP8266-12F
S1 = Tactile switch
PCB # 160100-1 v2.2

*mount at bottom side of PCB





S1, now release S2 and then finally release S1 to enable flashing of the webpages;
- select Tools → ESP8266 Sketch Data Upload. This should start uploading the files stored in the "data" folder that is included in the download from our website into the ESP8266 flash file system. When done, the IDE status bar will display the message 'SPIFFS Image Uploaded'.

If you fancy redesigning the clock's web page, that's also within reach. See the inset **How to create/edit a webpage for the ESP12.**

Now the fun starts!

On your PC, tablet or Smartphone, connect to the 'RGB Clock' Wi-Fi network, open a browser and connect to 192.168.4.1:81. In the menu (top left corner) select 'Wi-Fi settings' and enter the SSID and password of your wireless network. The clock will use this network to connect to the time server (NTP) to synchronize its time. In the sketch the server 'time.nist.gov' is defined as the NTP server for our clock. The clock will also send its sensor data to a server at Sparkfun (see inset **Easy access to your sensor data**).

A few remarks apply:

- the ESP12's access point will still be active after you have changed the Wi-Fi settings, so you can always modify them (and other settings) with your mobile device connected to the "RGB Clock" network;
- firewalls — like the one used at the Elektor offices — may block the NTP-protocol of an Internet time server, making automatic synchronization of the RGB clock impossible. We used a smartphone configured as a Wi-Fi hotspot as a quick bypass for retrieval of the time data for the clock. Once it is synchronized it can run without NTP data. Alternatively you can use the manual settings on the clock's webpage to set the correct time;
- if you would like to connect the clock to an unprotected network (i.e. no password), check the 'Unprotected Wi-Fi network' check mark in the Wi-Fi settings of the clock.

Every device with access to your Wi-Fi router can now access the clock to change its settings. Simply type 'clock.local:81' in the address bar of a browser to get access to the clock's webpages.

(160100)

How to create/edit a webpage for the ESP12

We built the internal website for the ESP12 using the HTML-Kit292 (www.htmlkit.com/download). This free program allows you to create and design your own website. Out web page comprises three files: *index.html*, *style.css* and *script.js*. Index.html is the main file and is programmed in HTML. It determines the structure of the website. In style.css we define how our page looks. Script.js determines what the page does.

The ESP12 webpage is built from several elements. Using these elements the layout and functionality are defined in style.css and script.js.

Web Links

[1] www.arduinesp.com/getting-started

[2] www.elektormagazine.com/160100

[3] https://github.com/esp8266/Arduino/blob/master/doc/filesystem.md

[4] https://github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.2.0/ESP8266FS-0.2.0.zip