eLABS PROJECT

Weather Display

Visualize current weather data on an LC display

By **Markus Hirsch** (Germany)

Good weather data is more extensive than what is displayed by low-cost desktop devices that show the air temperature, air pressure and relative humidity on a monochrome LCD along with the time. A modern microcontroller with suitable peripherals can now quickly fetch the relevant and current data from the Internet and present the results in color, along with a realistic weather forecast.

PROJECT INFORMATION

Weather
Display WLAN
Home & Garden

entry level
➡ intermediate level
expert level

5 hours approx.

FTDI-USB / serial cable
Software downloads
(see text)

€50 / £50 / $55 approx.

Thanks to the Internet and advanced microcontrollers, it has never been easier to obtain the exact time of day, weather forecasts and a lot of other interesting information with your own electronics projects. This information can also be effectively processed for presentation on a low-cost color display. If you want to know the current weather and always have an up-to-date weather forecast, you can either hang a tablet computer with a suitable app on the wall or build the necessary hardware yourself. The latter option is clearly much more interesting, especially because it is not especially complicated with modern technology. The modular approach is the fastest way to the desired result. This means you get your hands on a suitable microcontroller board and a matching color display module, and for the rest of the necessary circuitry you build a shield or some other sort of add-on board in the appropriate format. Then you link these parts together and add the right firmware to the mix. That's exactly the recipe described below.

Weather data and more

There is a lot of data available that can be visualized on a display. In the current version of this project, it consists of weather conditions and a five-day forecast. The aim is to display all weather data, including temperatures, air pressure, wind direction and speed, using icons for some of this data. Some useful extras are the calendar week, zodiac sign and moon phase. The latter two items are not normally provided by weather services, so the moon phase and zodiac sign have to be calculated directly on the microcontroller board. It is also helpful if the display can additionally show technical data such as the SSID of the WLAN, the received signal strength,

and the relevant IP address. The user interface is basically simple: a button press opens a small menu, a short button press scrolls to the next menu item, and a longer button press selects the current menu item. The WLAN connection to a nearby hotspot can be configured conveniently with a smartphone or other device. Before plunging into the hardware details, you can see what the end result looks like in **Figure 1**.

Hardware

Now that we have defined what the overall system has to do, we have to consider which hardware is suitable for this purpose. Obviously the computation

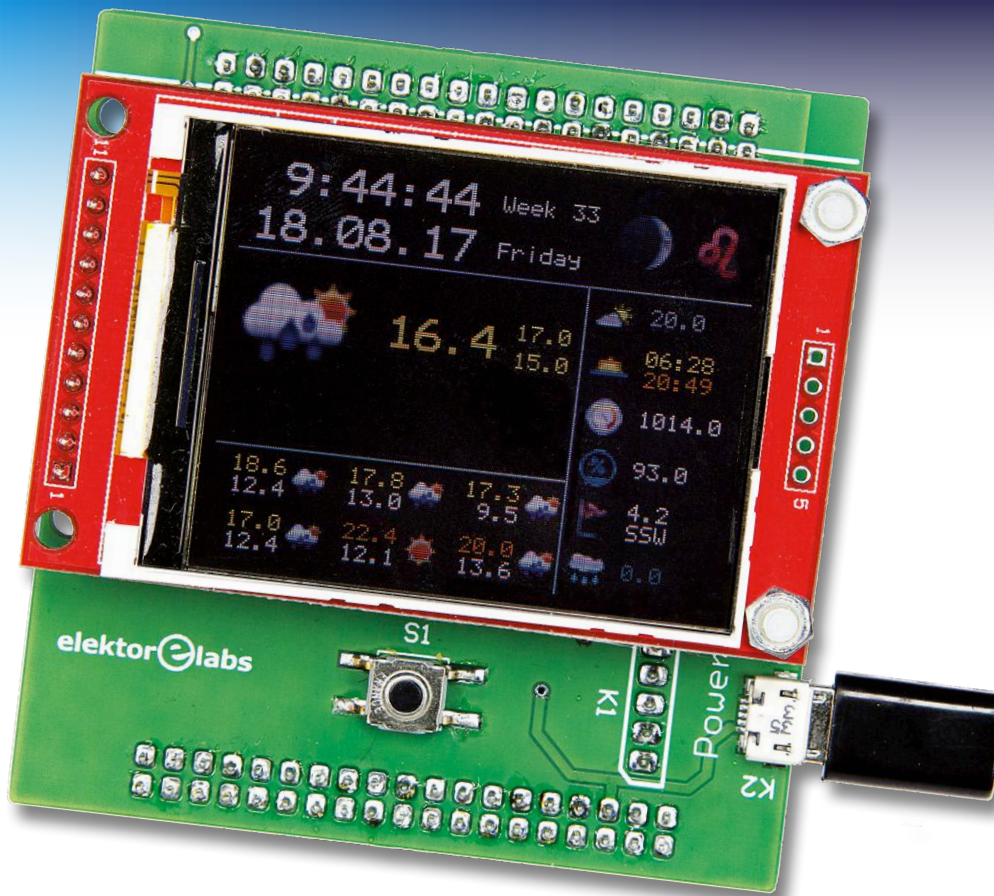


Figure 1. The prototype weather display with a screen full of weather and forecast data.

Graphic: shutterstock.com

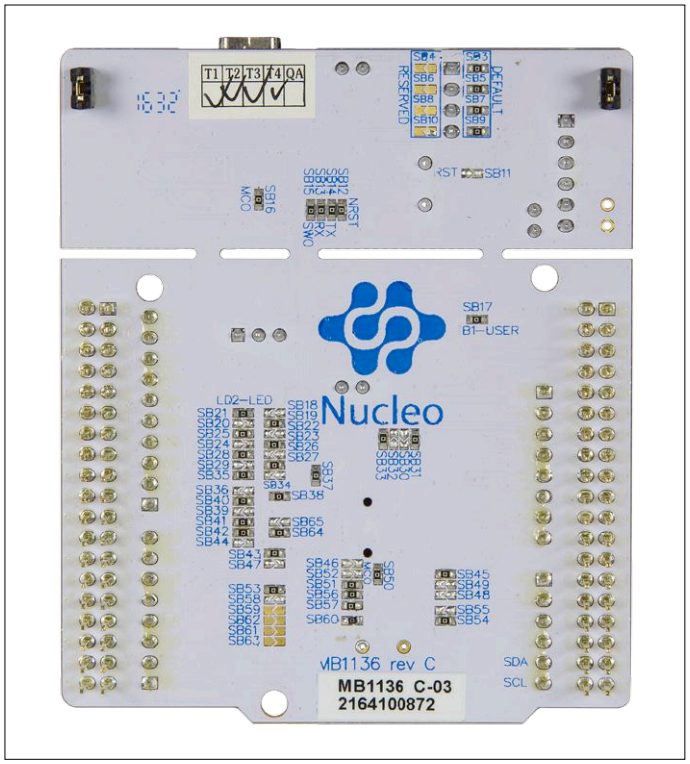


Figure 2. Bottom side of the Nucleo board. Here you can see the locations of the solder bridges and other components.

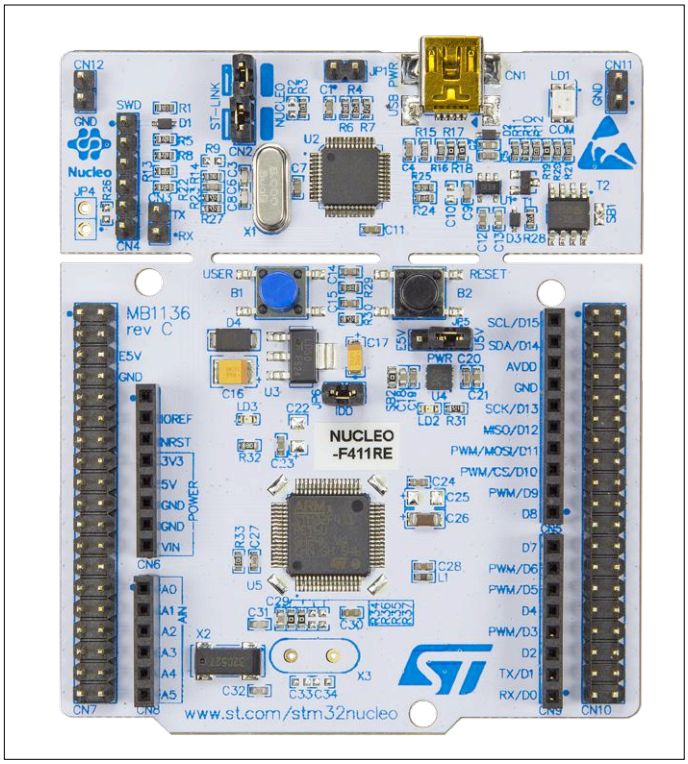


Figure 3. Top side of the Nucleo board. The upper part of the board is the ST Link section, which should be separated. The compatibility with the Arduino platform can be seen from the headers.

tasks for this sort of project can be handled by many current microcontrollers and perhaps even more ready-made boards. Fetching data from the Internet, processing it appropriately and displaying the results are straightforward tasks with the right firmware. On the other hand, embedded boards are now so cheap that it doesn't hurt to choose one with a bit more processing power than necessary, since the spare capacity can easily be used for your own extensions, or even entirely different projects, without creating resource issues. For this reason, the author chose a high-performance board from ST.

From the range of available boards, the author opted for the **Nucleo F411RE**.

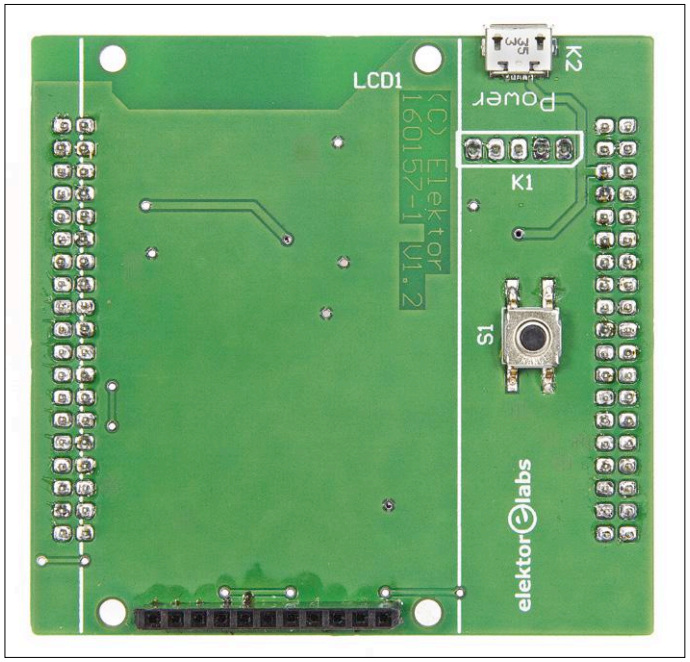


Figure 4. Except for the pushbutton and the USB connector, the display side of the shield is nearly empty.

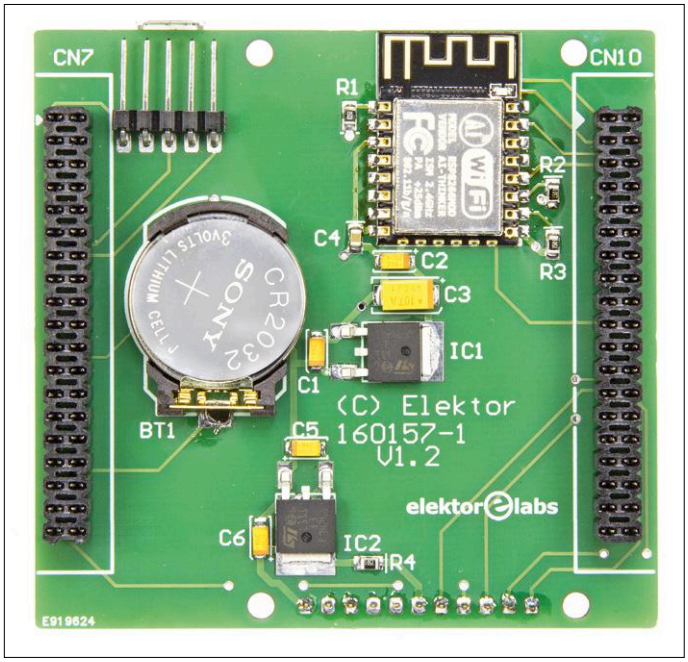


Figure 5. On the Nucleo side of the shield forming a bridge between the Nucleo board and the display, you can see the button cell, two voltage regulators and the WLAN module.

Along with a powerful STM32F411RE microcontroller, this development board comes with a debugger/programmer (ST Link), and the I/O pins of the MCU are available on suitable headers. An attractive feature of this board is that the connectors are compatible with the Arduino Uno layout. A wealth of information about the board is available on the ST website [1]. **Figure 2** shows the bottom of the board, and **Figure 3** the top.

The display is a ready-made module with a 2.2-inch (diagonal), 220 x 176 pixel screen based on the **ILI9225** display controller. If you search for this IC on eBay, you will find a multitude of modules with prices from 5 to 10 euros (or similar in pounds or dollars). The main advantages of this sort of module are that it can be driven in serial mode, its signal levels are compatible with 5 V and 3.3 V, and it even has a suitable voltage regulator on board.

The display module is not mechanically or electrically compatible with the Arduino or the Nucleo. The conventional way to resolve this problem is to design a simple **shield PCB** that serves as a bridge between the microcontroller board and the display, and which can also hold other necessary hardware in addition to the display. **Figure 4** shows the nearly empty side of the shield facing the display. All you see there is a micro USB connector for the power supply and a pushbutton for the user interface. **Figure 5** shows the side that plugs onto the Nucleo board. Along with some resistors and capacitors, the most noteworthy feature is a button cell. It is not intended for battery-powered operation, but instead as a backup power source for the integrated real-time clock on the Nucleo board in the event of a power outage. Aside from the two 3.3-V voltage regulators, the only other thing you see is a small breakout board with a serpentine PCB track. That looks like an RF component, and in fact it is an ESP8266 WLAN module, which has previously been used in a fair number of Elektor projects. The four holes in the PCB are for screw mounting the display module.

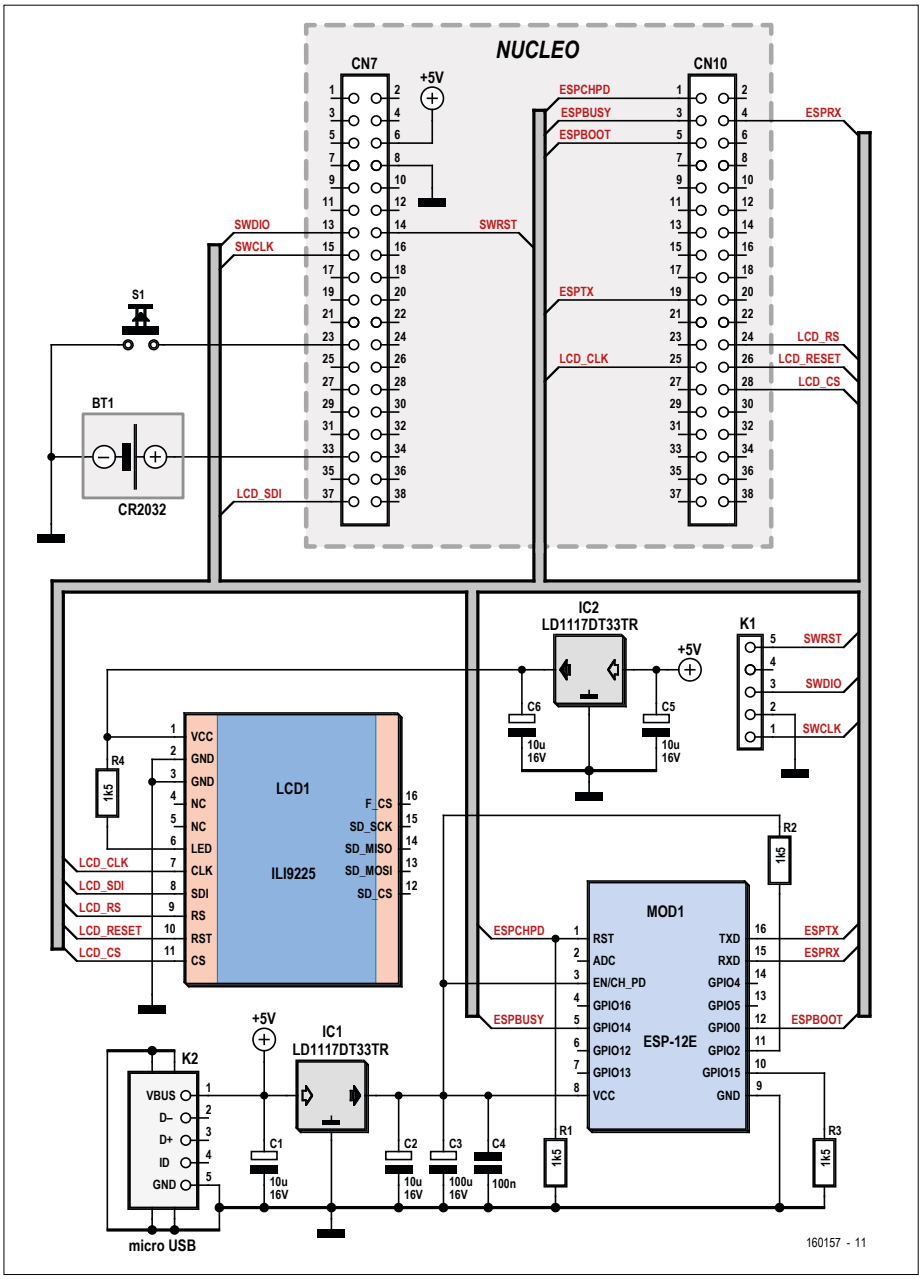


Figure 6. Circuit diagram of the shield. The component density on the board is low.

Shield

Along with the holding the necessary headers on both sides and some electronic components, the main role of the shield is to provide mechanical support for the display and a base for the pushbutton. The circuitry shown in **Figure 6** is correspondingly simple — something this shield has in common with many other shields. It essentially consists of two voltage regulators with associated block capacitors (for separate power supply to the display and the WLAN module) and four resistors, a USB connector (with no data line connections in the current version), a pushbutton, a button cell, the display, the WLAN module, and the two headers for plugging onto the Nucleo board. Connector K1 is an additional small header for a serial interface, which is used for programming with the ST Link programming interface included with the Nucleo. By the way, S1 is connected through the header in parallel with the blue button B1 on the Nucleo board

The WLAN module is connected to the Nucleo board through a serial interface (including control lines), which makes it possible to program the module (i.e. its integrated microcontroller) via the Nucleo board. We will do that with the Arduino IDE, as described later in this article. The simplicity of the circuitry results from the use of the Nucleo board, which already has a wealth of functions and components.

It's all so simple that it is scarcely necessary to describe the board layout

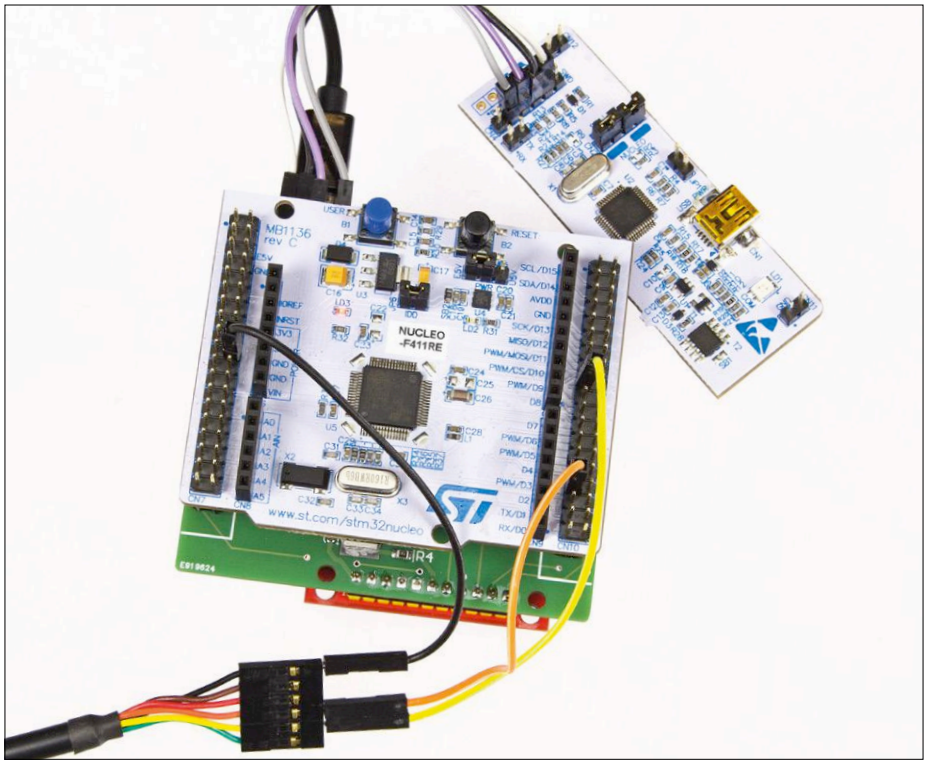


Figure 7. Here you can see how to connect the ST Link board to the Nucleo board (upper part) and how to connect an FDTI USB/serial cable for programming the WLAN module (lower part).

and assembly – provided that you have some prior experience with soldering SMDs. However, you don't need to be afraid of the SMDs used here, since nearly all of them are in solder-friendly 0805 packages. We should however point out that you should not try to use a 10-watt miniature soldering iron to heat the solder pads for the metal tabs of IC2 and IC3. The lower limit is 30 watts — more is better.

Table 1. Modifications on the Nucleo board.

Component	Action	Function
X3	Solder on 16 MHz crystal	X3
C33	Solder on 22 pF capacitor	X3
C34	Solder on 22 pF capacitor	X3
SB54	Open	X3
SB55	Open	X3
R35	Closed	X3
R37	Closed	X3
SB16	Open	MCO
SB45	Open	VBAT
SB50	Open	MCO
SB62	Closed	UART
SB63	Closed	UART
JP5	Set to E5V	Voltage

Development environment

The source code was written in C++. Both commercial and free development environments are available for STM32 MCUs. For this project the author selected TrueSTUDIO from Atollic [2]. This IDE is based on the well-known Eclipse environment. Although TrueSTUDIO is basically a commercial environment, there is also a free version with no code size restriction. Among other things, the versions differ with regard to some debug features that are probably only important for professional users.

TrueSTUDIO is tailored to development with ARM processors and provides a practical preconfiguration function, which in the original Eclipse version first had to be set up manually (and in addition some components still have to be installed retroactively). With TrueSTUDIO the connection to ST Link takes just a few clicks. The environment also includes a compiler. HAL libraries and many example project are available for many processors. For hobby use, that makes the free version of this development environment a simple and easily installed, but nevertheless high-performance solution for software development with the Nucleo board.

Nucleo configuration

Some small modifications to the Nucleo board are necessary before using it in this project. They are described in the following nine steps. For this it may be helpful to download the user manual [3] in the form of a PDF file.

Note: If you are programming the Nucleo board for the first time, you can skip steps 1 and 3.

1. Separate the ST Link part from the main Nucleo board (using a Dremel tool or a saw). Check that all PCB tracks have been cleanly parted and no shorts are present on the edges of both boards.
2. Carefully make the modifications to the hardware of the Nucleo board listed in **Table 1**. This includes closing or opening solder bridges and mounting a crystal and two capacitors. Tip: For closed solder bridges, you can reuse the 0-Ω resistors in 0603 format removed to open solder bridges at other places on the board.
3. Make the four connections between the ST Link board and the Nucleo board listed in **Table 2**. They are shown in the upper part of **Figure 7**.
4. Connect a 5 V power supply to CN7-6 (+5 V) and CN7-8 (GND). If you have not yet separated the ST Link board from the Nucleo board, you can leave JP5 in the default U5V position. In that case the Nucleo

Table 2 Connections between ST Link and Nucleo.

Designation	ST Link pin	Nucleo pin	Shield
SW CLK	CN4-2	CN7-15	K1-1
GND	CN4-3	CN7-8	K1-2
SW DIO	CN4-4	CN7-13	K1-3
SW Reset	CN4-5	CN7-14	K1-5

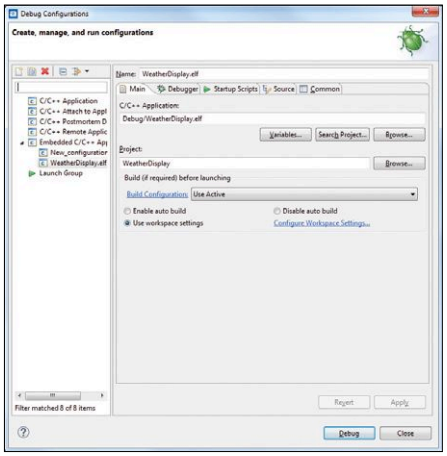


Figure 8. Nucleo configuration: the Main tab.

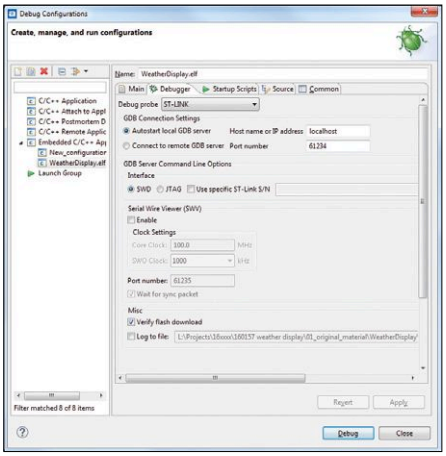


Figure 9. Nucleo configuration: the Debugger tab.

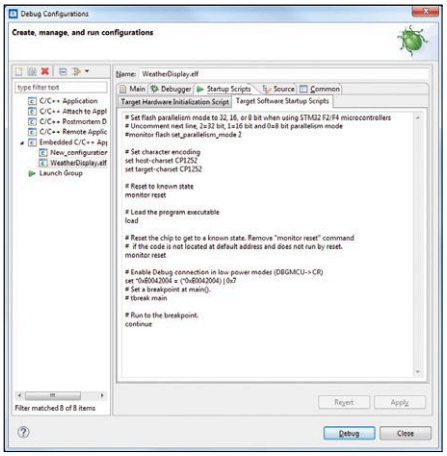


Figure 10. Nucleo configuration: the Startup Scripts tab.

board will be powered from the ST Link board via USB.

5. Download and install the evaluation version of Atollic TrueSTUDIO [2], and download the software necessary for the weather display.
6. Launch Atollic TrueSTUDIO and import the WeatherDisplay project: *File* → *Import* → *General* → *Existing Projects into Workspace.*
7. Configuration: go to *Run* → *Debug Configurations* and double-click on *Embedded C/C++ Application*. Then copy the configurations shown in the screenshots for the *Main* tab (**Figure 8**), the *Debugger* tab (**Figure 9**) and the *Startup Scripts* tab (**Figure 10**).
8. Connect the ST Link board via USB to your computer if you have not already done so. If you are using an external power supply instead of USB to power the Nucleo board, switch it on (see step 4).
9. Now program the Nucleo board by selecting *Debug Configuration* on the *Run* menu.

ESP8266 programming

The ESP8266 module is provided here for a WLAN connection to the Internet and must first be programmed for this purpose. We do this by writing the source code for this module in the Arduino IDE and then programming the module with the resulting machine code. You will also need one of the well-known FDTI USB/UART interface cables for this. With the Nucleo acting as a programming interface with 5 V tolerant I/O ports, you are not compelled us use a cable with 3.3 V signal levels.

For programming, plug the assembled shield with the display (after attaching the display to the board with suitable screws) onto the Nucleo board. The CN7 and CN10 headers of the two modules mate perfectly. To power the system through connector K2, use a conventional 5 V USB AC adapter and a cable with a micro USB plug. Note that jumper JP5 on the Nucleo board must be in the “E5V” position for this.

Next, make the connections listed in **Table 3** (see also the lower part of **Figure 7**).

Now you have to adapt the Arduino sketch to your own geographic location and your own APPID, so that the weather forecast data will match your location. You can obtain a free APPID from *openweathermap.org* by registering at [4]. The locations list is available at [5]. Then edit the following lines in the Arduino sketch *WeatherTimeget_el.ino*:

```
Const String APIID = "your_APPID";
Const String LOCATION = "your_location"; // Example: "london"
```

These are the first two declarations at the start of the sketch.

Now is a good time to enter the name (SSID) and password of your WLAN in the Arduino sketch. These parameters can also be modified later in the Nucleo menu, but it is more convenient to enter them directly in the sketch at this point.

Table 3. FDTI cable connections to Nucleo.

FDTI cable	Nucleo board
GND	CN7-20
TxD	CN7-31 (PB3)
RxD	CN10-17 (PB6)

In the sketch, go to the *Setup* function and replace the line

```
WiFi.begin ()

by

char cssid[] = "YOUR SSID";
char cpasswd[] = "YOUR PASSWORD";
WiFi.Begin(cssid, cpasswd);
```

Of course, you should enter the SSID and password of your own WLAN here.

If you have never previously used the Arduino IDE to program the ESP8266 module, you have to modify the IDE settings under *File* → *Preferences* as shown in **Figure 11**.

Then use the Board Manager to add the ESP module via *Tools* → *Board ...* (see **Figure 12**). More information about the ESP8266 library is available at [6].

Now select *Generic ESP Module* under *Tools* → *Board.* As previously mentioned,

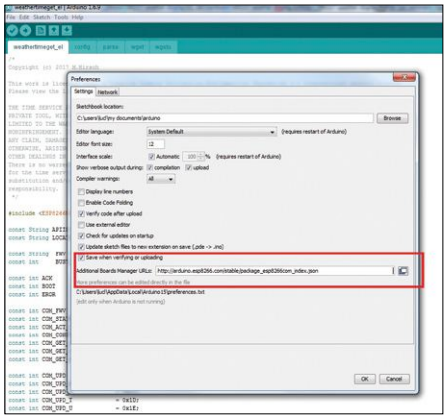


Figure 11. The Arduino IDE configuration for the ESP8266.

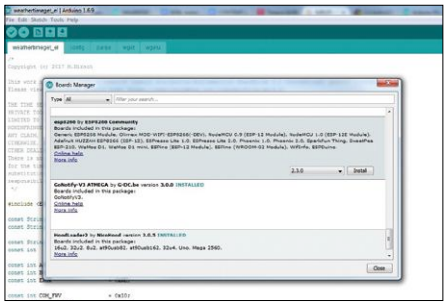


Figure 12. Adding the ESP8266 module in the Arduino IDE.

the Nucleo acts as a programming interface for the ESP module. For this you have to set the baud rate to 9600 under *Tools → Upload Speed*. Higher speeds can lead to errors. It goes without saying that you should also select the right COM port for programming under *Tools → Ports*.
Now the Arduino IDE is ready to put the Nucleo board in programming mode for the ESP8266. Press S1 below the display (or the blue button on the Nucleo board) to open the Nucleo main menu. Scroll down with short button presses until you reach the entry *ESP menu*. Then press somewhat longer to select this option. Next, select *Program loop serial* (see **Figure 13**).
Now everything is ready for programming the WLAN module with the appropriate sketch from the Arduino IDE. To start the programming process, select *Sketch → Upload*. It may take a while for this operation to complete. It is done when the status is shown as 100%. Following this you can exit Nucleo programming mode by briefly pressing S1 twice at *Exit*, by simply resetting the Nucleo board with the black button, or by brutally switching everything off and back on again.

Figure 13. Configuring the Nucleo as a programming interface for the ESP8266 module.

COMPONENT LIST

Resistors
R1–R4 = 1.5kΩ, SMD0805

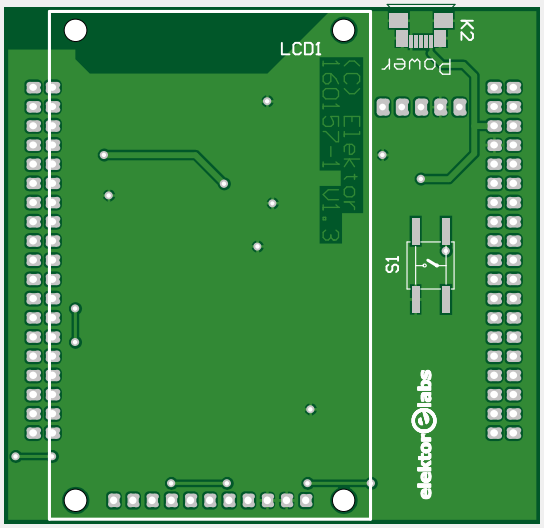
Capacitors
C1,C2,C5,C6 = 10µF 16V, SMD1206
C3 = 100µF 16V, SMD2312
C4 = 100nF 50V, X7R, SMD0805

Semiconductors
IC1,IC2 = LD1117DT33, LDO, 3.3V, 800mA

Miscellaneous
LCD1 = 2.2" TFT module, 220x176 pixel, with ILI9225 (eBay) *
MOD1 = ESP-12E (ESP8266 WLAN module) *
CN7,CN10 = 38-pin (2x19) stacking header, vertical
K1 = 5-pin (1x5) pinheader, right angle
K2 = micro USB type B female, PCB mount, horizontal
S1 = pushbutton, SPDT, 6x6mm pitch
BT1 = CR2032 lithium button cell, 3V, with 20mm battery holder
PCB 160157-1 V1.3 (Elektor Store)
ST Nucleo F411RE board

Components for Nucleo modification *
C33,C34 = 22pF 50V, SMD 0603
X3 = 16MHz crystal for 18pF load

* See text



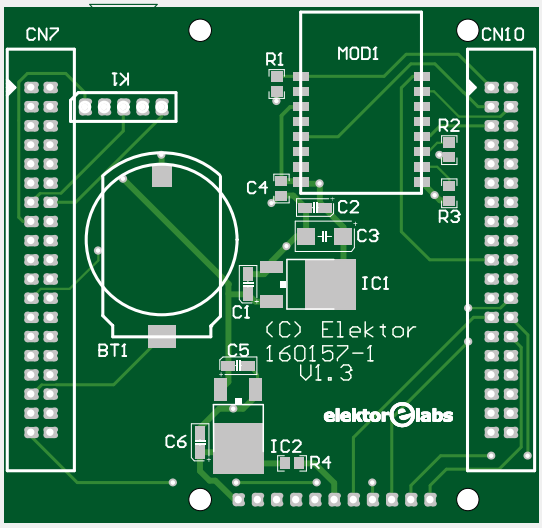


Table 4. Status bits.		
Bit no.	**Value = 0**	**Value = 1**
0	WLAN not connected	WLAN connected
1	Error in weather data update	Weather data update successful
2	Error in forecast update	Forecast update successful
3	Error in time data update	Time data update successful
4	No UV data	UV data loaded

That completes the configuration and programming of the weather display. After a short time, the LCD should show the weather forecast for your location as entered in the Arduino sketch.

WLAN configuration

If you did not previously enter your WLAN login data in the Arduino sketch or you want to change it afterwards, you can use the menu of the Nucleo board to enter or edit the data. To do so, go to the Nucleo main menu, select *Config WiFi*, and confirm with *YES* in the next window (see **Figure 14**).

Now you can use a smartphone or a tablet to establish a connection to the WLAN generated by the ESP8266. Bear in mind that this WLAN is only active when you are in *Config WiFi* mode or the weather display is not able to establish a connection to your own WLAN. The SSID is *WeatherNet* and the password is *WeatherPass*. Both terms can be changed in the Arduino sketch.

To configure the module, log in to this WLAN and open the ESP8266 page at IP address 10.0.0.1. There you will see a list of available wireless networks. Select the SSID that you want to use and enter the associated password. Then click on *Save* to save the configuration. After a few seconds, the WeatherNet WLAN is switched off and the Nucleo automatically returns to its main menu. If you now select *Exit*, the WLAN configuration process will be closed and the weather forecast will be shown again. The login data entered in the Arduino sketch is used after a restart.

To check the WLAN settings, select the *Status* entry on the menu. Then you will see the Nucleo and ESP firmware versions displayed at the top of the screen, followed by the IP address of the ESP, the SSID of the WLAN, and the WLAN signal strength. Of course, the status of the network connection is also shown. A value of 0x0F is good news because it means that the system is connected to the WLAN and all data has been loaded. The meanings of the individual status bits are listed in **Table 4**.

Miscellaneous

The routines for fetching UV data are unfortunately not included in the weather display firmware because this data is not available for free. If you happen to find a free source of UV data, the author would be very pleased to hear about it.

As mentioned at the start of this article, the STM32 is by no means fully loaded with the tasks described here. This means that there is plenty of opportunity to go hunting for other interesting data, such as ozone concentration etc., and modify the firmware so that this data can also be downloaded from the Web and displayed. Other conceivable options include several alternating views, animated weather icons, a feed reader, email notifications, or a WLAN boot loader. As usual, the Nucleo firmware for this project (along with the Arduino sketch for the WLAN module) is available on the Elektor web page for this article [7] for free download. We hope you enjoy this project! ⏮

(160157-I)

Figure 14. Configuring the WLAN settings via the Nucleo menu.

Web Links

[1] ST Nucleo F411RE board: ww.st.com/en/evaluation-tools/nucleo-f411re.html

[2] Atollic: https://atollic.com/resources/download/

[3] Nucleo manual: www.st.com/resource/en/user_manual/dm00105823.pdf

[4] APPID: https://openweathermap.org/appid

[5] Locations list: http://openweathermap.org/help/city_list.txt

[6] ESP8266 library info: https://esp8266.github.io/Arduino/versions/2.3.0/

[7] Downloads: www.elektormagazine.com/160157

About the Author

Markus Hirsch originally studied industrial engineering but ended up in R&D due to his many years as an electronics hobbyist. After spending several years in quality management, he now works as a hardware and software developer in a mid-size industrial firm. He also enjoys developing hardware and software projects in his spare time. In addition, he is active in 3D printing, app and Web programming, and other projects. Although these projects have practical or educational uses, the main consideration is that they are fun. Just like the project presented here.