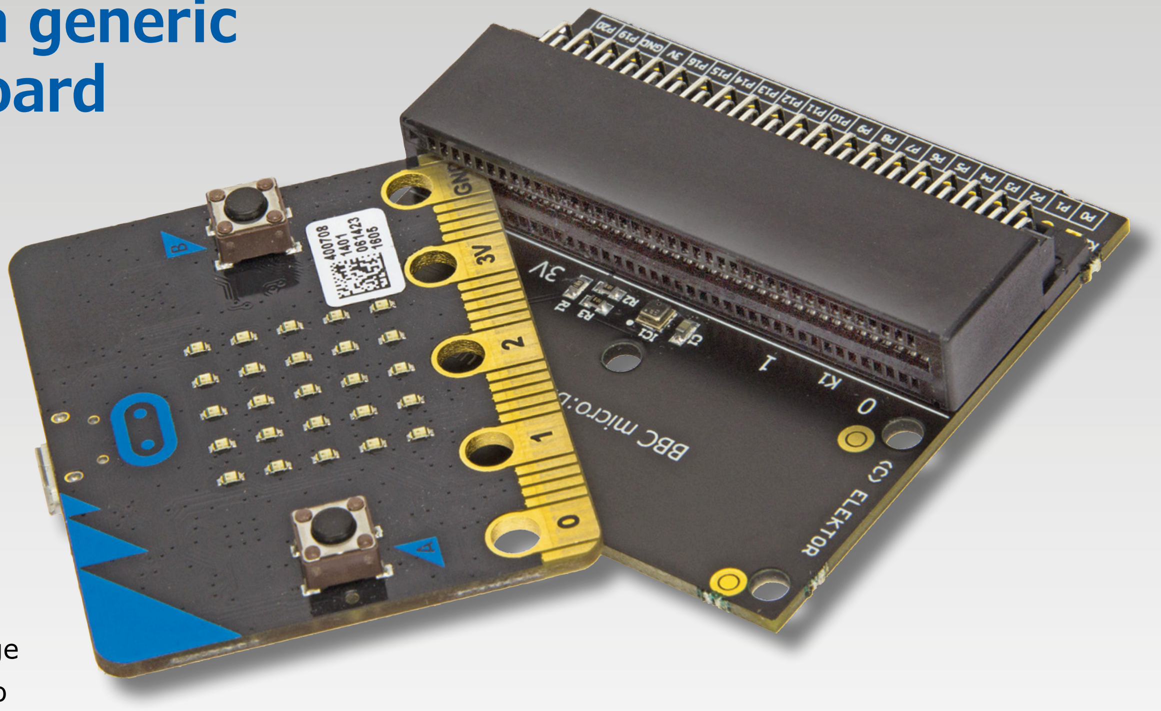
BBC micro:bit Weather Station

Doubles as a generic extension board

By **Clemens Valens**
(Elektor Labs)



The BBC micro:bit is a small powerhouse loaded with cool functions and supported by a suite of excellent development tools and libraries. Its edge connector allows access to goodies like GPIO ports, analog inputs, an I²C bus and an SPI port. To show how to use it we build a small weather station for it.

People love to measure. Give a child a tape ruler and it will immediately start to measure everything in sight. The BBC micro:bit, initially intended for children, is equipped with two sensors, an MMA8653 3D accelerometer that measures acceleration on three axes, and a MAG3110 3D magnetometer that measure magnetic field strengths on three axes. The latter also measures its die temperature and outputs it, giving yet another measurement result. These sensors are intended for orientation, movement and gesture detection. To complete this set of sensors, we added a BME280 combined humidity, pressure and temperature sensor from Bosch Sensortec. Typical applications of this sensor are context awareness, skin detection and vertical navigation, but it also happens to be an excellent weather sensor. It provides very precise humidity, atmospheric pressure and temperature data on an I²C or SPI bus. The joint force of the three sensors allows you to turn the micro:bit into an accurate inertial measurement unit (IMU). Or a wireless weather station if you add micro:bit's Bluetooth to the mix.

Hardware

The schematic of our add-on board is pretty straightforward (**Figure 1**): a sensor (IC1, **Figure 2**), an LED, three resistors and a capacitor. The hard part this time is the edge connector K1 because it is difficult to find. For the rest, there is not much to tell about such a simple design.

We connected the CSB signal of IC1 (pin 2) to V_{IO} (which is connected to V_{CC}). This selects the device's I²C interface. IC1 can also do SPI in three- or four-wire mode, but for that the CSB pin must be pulled Low. Once CSB is sampled Low, the device remains in SPI mode. If, for some reason, you want to select the I²C interface by using an external signal, this signal must have a level of V_{IO} before the chip is reset.
The SPI interface is compatible with SPI mode '00' (CPOL = CPHA = '0') and mode '11' (CPOL = CPHA = '1'). Mode selection is automatic, and determined by the value of SCK after the CSB falling edge. Three- or four-wire mode is controlled in software by the SPI3W_EN bit. SDI is the data line in three-wire mode, SDO is not used in this case.
Because we use the device in I²C mode, we provided the possibility of adding pull-up resistors to the bus (R2 & R3). If you only use the extension board with the micro:bit you don't need these, because they are already present on the micro:bit. When the BME280 is in I²C mode, its SDO pin (pin 5) determines its Slave address. Connecting SDO to GND sets the Slave address to 0x76; connecting it to V_{IO} like we did results in 0x77. The SDO pin cannot be left floating.
LED1 is a power indicator — it's helpful because the one on the micro:bit itself becomes invisible when the board is inserted in connector K1. LED1 remains visible at all times.
K2 is a pin header that provides easy connections of the

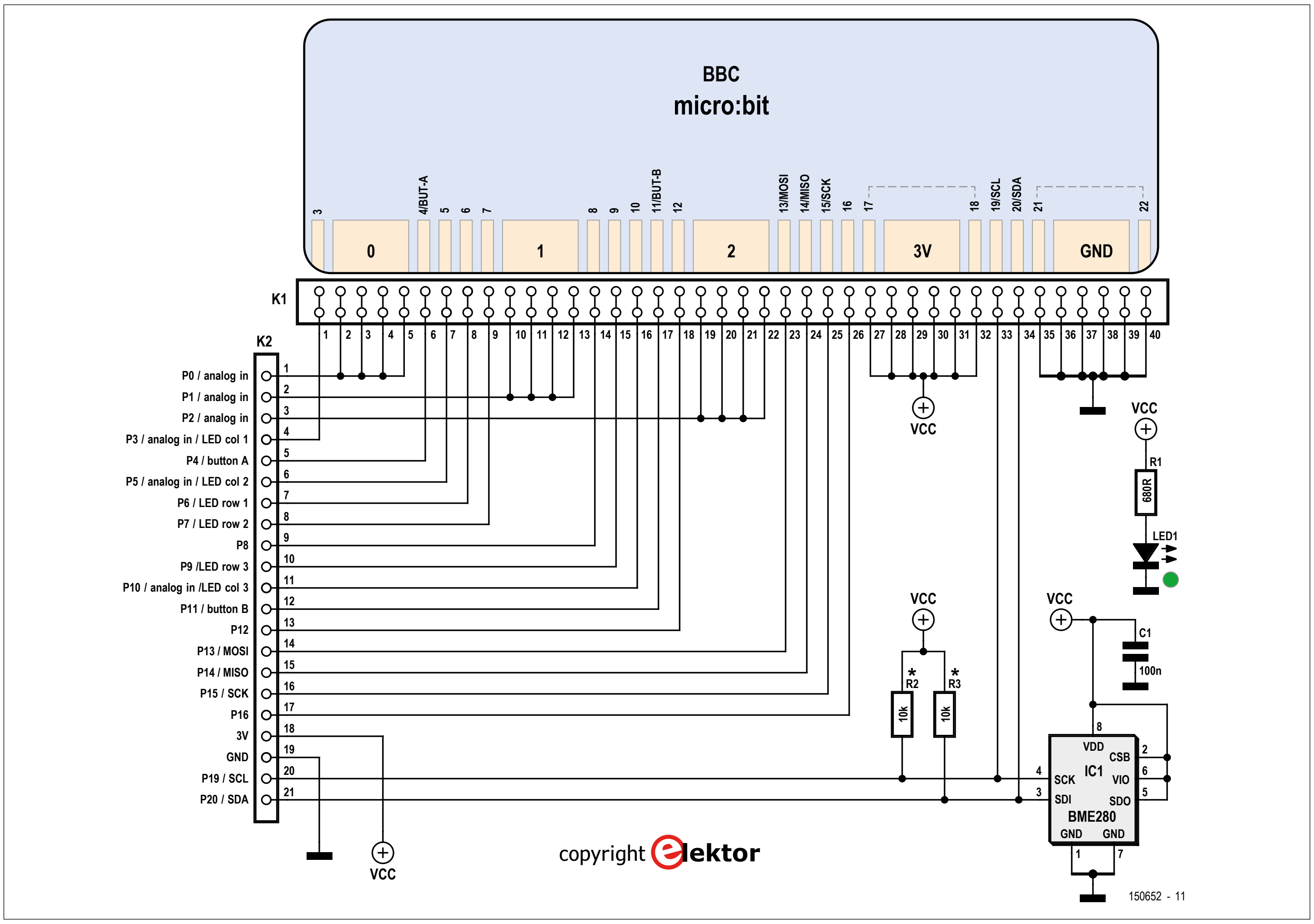


Figure 1. Schematic of the weather station extension board for the BBC micro:bit.

micro:bit edge connector to, for instance, a breadboard. The order of the pins on K2 is slightly different from the micro:bit because we chose to put them in increasing order from 0 to 20. On the micro:bit pins 0, 1 and 2 correspond to the large banana-plug-compatible holes that each take up several contacts of K1. The other contacts count from 3 to 21 from left to right (LED matrix side visible, edge connector at the bottom).

Software...

... is needed for the microcontroller to be 'at speaking terms' with the BME280 and so we wrote a driver for it. While we did this the micro:bit was still unreleased and the nice libraries for it were not available yet. However, the board was already a known platform in mbed [1] and so we wrote our driver as an mbed project. We also did an Arduino Sketch for it, you can get it at [2] or [3], so you can use the sensor with an Arduino compatible board.

Bosch Sensortec provides a driver for the BME280 on GitHub, but it is rather complicated, not only because it is supposed to work on all sorts of platforms, 64-bit, 32-bit, 16-bit, with or without floating point units (FPUs), but also because it supports every detail of the chip. For our simple weather station application we don't need all this, and since it is rather instructive to write your own driver, that is what we did. Our driver [2],[3] also lets you control every bit in every corner, but you will need the datasheet to figure out what to write to, and read from, specific locations. And since C++ permitted us to write an Arduino-style API, we wrote our driver in C++.

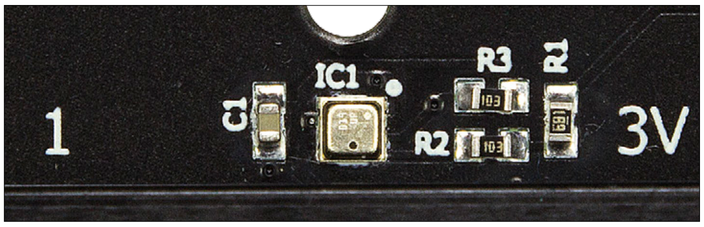


Figure 2. IC1, the tiny metal can in the middle of the board, contains three high-precision sensors.

Using our driver is easy enough. It starts by creating a BME280 object and then calling its `begin` function, as has become customary since Arduino rules the world:

```
BME280 bme280;
bme280.begin(0x77);
```

The function (or *method* if you prefer) `begin` takes the I²C address as argument, on our board it is hardwired to 0x77. If you choose not to provide an address, the driver will default to an SPI bus. The function returns 0 if the sensor was found on the bus and if its calibration data could be read. The next step is to configure the sensor:

```
bme280.writeConfigRegister(BME280_STANDBY_500_
US,BME280_FILTER_OFF,0);
bme280.writeControlRegisters(BME280_
OVERSAMPLING_1X,BME280_OVERSAMPLING_1X,
BME280_OVERSAMPLING_1X,BME280_MODE_NORMAL);
```

We used basic settings here — no fancy stuff, no high speed, no filtering. Now you're ready to collect data. You do this by first calling the function `read` to get the data from the sensor into the driver and then by calling one or more of the member functions `temperature`, `pressure` and `humidity`. Typically you would do this in a loop, but you don't have to.

```
bme280.read();
printf("T=%d degrees C, ", bme280.temperature()/100);
printf("P=%d mbar, ", bme280.pressure()/100);
printf("RH=%d%%\n", bme280.humidity()>>10);
```

Note how the values obtained are being scaled. `Temperature` must be divided by 100 to obtain degrees Celsius. `Pressure` is expressed in Pascal and to be practical must be divided by 100 to obtain milli-bars (mb). `Humidity` has to be divided by 1024 (which is the same as shifting it 10 bits to the right) to obtain a percentage. It is possible to enable floating point arithmetic in the driver by making `BME280_ALLOW_FLOAT` unequal to 0 (at the top of the file bme280.h):

```
#define BME280_ALLOW_FLOAT  (1)
```

This will probably make the executable bigger and slower, but it depends on your application if that presents a problem or not. The floating point versions of the member functions `temperature` and `humidity` do not require scaling, `pressure` is again in Pascal.

Our driver can handle both SPI and I²C busses (although we didn't test SPI) and it is the user's responsibility to provide the functions for it. The driver will call

```
i2cWrite(…) & i2cRead(…)
spiWrite(…) & spiRead(…)
```

as needed, depending on how it was set up. All four functions must be provided, but may remain empty. If you use I²C leave the SPI stubs empty, if you use SPI then leave the I²C stubs empty.
Note that in mbed I²C addresses are 8-bit because the read/write bit is included at bit 0, meaning that 0x77 becomes 0xee. In Arduino the address is 0x77.

Expanding further

Connector K2 is supposed to be a pinheader, but you are free to make anything that suits your application. With a pinheader or socket it is easy to connect the board to a breadboard, especially if you have some of those very practical breadboard wires. If you mount a pinheader, horizontal or vertical, you will be ready for our upcoming micro:bit project: the micro:bit Dock. Stay tuned…

(150652)

Web Links

[1] https://developer.mbed.org/platforms/Microbit/

[2] https://github.com/ElektorLabs/bme280-driver

[3] www.elektormagazine.com/150652

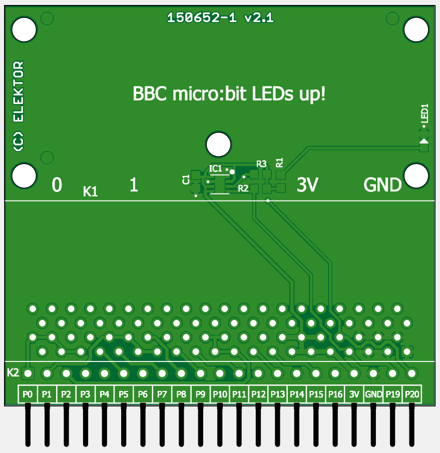
Component List

Resistors
5%, 50V, 0.1W, 0603
R1 = 680Ω
R2, R3 = 10kΩ

Capacitor
C1 = 100nF, 0603

Semiconductor
IC1 = BME280
LED1 = LED, green, 0603

Miscellaneous
K1 = socket, PCB-to-PCB, 2x40 links, 0.05'' (1.27mm) pitch
K2 = pinheader, 1x21 pins, 0.1" pitch
PCB # 150652-1 v2.1


150652-1 v2.1
(C) ELEKTOR
BBC micro:bit LEDs up!
0
K1
1
3V
GND
K2
P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16 3V GND P19 P20