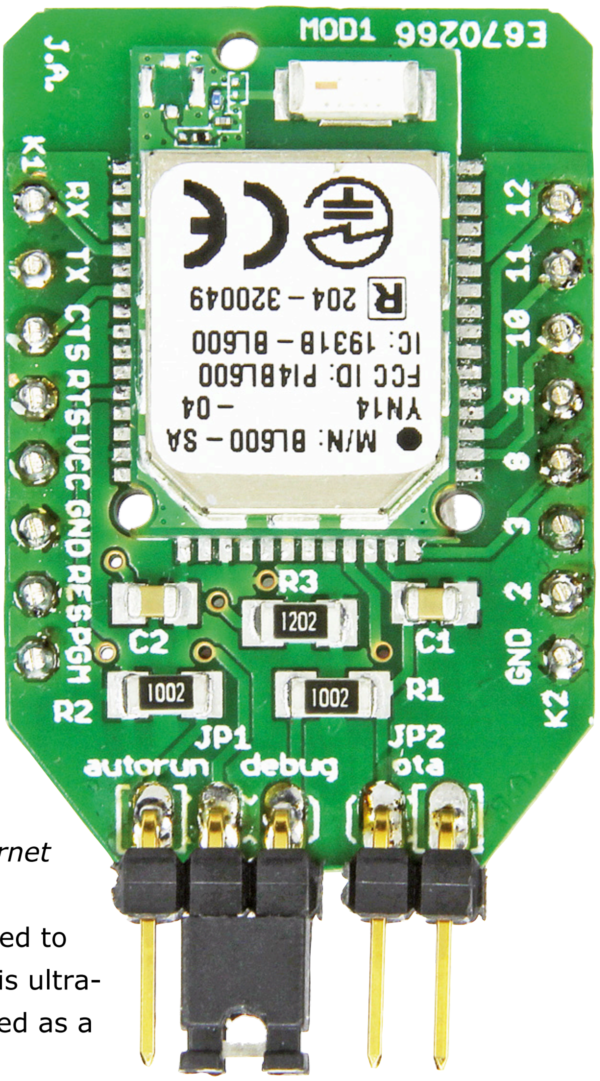
BL600-eBoB (2)

Editing, compiling, and transferring a program



By Jennifer Aubinais (France) elektor@aubinais.net

If we're to believe certain predictions, we'll soon no longer be talking about the *Internet of Things*, but rather about the *Internet of All Things*. This is going to imply a proliferation of wireless communication **and** low power consumption for the circuits used to connect up all these things. Which is exactly the purpose of this ultra-low consumption radio communication board, which can be used as a miniature computer, easily programmable in *smartBASIC*.

In the first of this series of articles devoted to the BL600-eBoB [1], we introduced the hardware: on the one hand the UART port, for communicating with a connected object — in our case a PC via the FT232 eBoB; and on the other, seven input/outputs we can use as we like: two ADC inputs, the I²C port, the SPI port. Then, using the OTA (over-the-air) function, we have seen how to transfer via our BL600-eBoB's radio link a first UART program, downloaded using an Android phone.

Here we're proposing a first application program, after familiarizing ourselves with the editing, compiling, and then transfer tools. This is going to be a simple light-chaser – but a Bluetooth Low Energy light-chaser! We'll be going through it step by step, and even one toe at a time, as even though the possibilities of this module are remarkable, you do have to master it. In the third article, we'll be discovering the principles of programming events in smartBASIC, which will enable you to create your own projects.

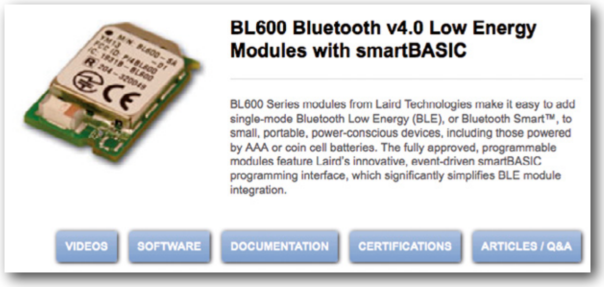
Editing

We're going to start by downloading the latest version of the BL600 firmware (Firmware Files version 1.5.70.0 – Revision 5 [2]) along with an editor; Laird Technologies recommends Notepad++ (Figure 1) [3] which recognizes smartBASIC syntax.

To obtain a login in the SOFTWARE DOWNLOADS section of the Laird Technologies site, click "If you need credentials, please click here" and wait a few days. Once you're logged in, download Firmware Files version 1.5.70.0 – Revision 5 which you'll need to unzip. This contains (**Figure 2**):

- program examples in the *smartBASIC_Sample_Apps* directory
- the UwTerminal.exe program in the *smartBASIC_Sample_Apps* directory
- the library in the *smartBASIC_Sample_Apps/lib* directory
- a number of examples (*UserManualExampleCode*)
- the special Notepad++ configuration for smartBASIC (*smartBASIC(notepad++).xml*)

Once the Notepad++ editor is installed, run it. In the menu, select *language* (**Figure 4**) then click *Define your language.* Click on *import*, select the *smartBASIC(notepad++).xml* file from the ZIP


BL600 Bluetooth v4.0 Low Energy Modules with smartBASIC

BL600 Series modules from Laird Technologies make it easy to add single-mode Bluetooth Low Energy (BLE), or Bluetooth Smart™, to small, portable, power-conscious devices, including those powered by AAA or coin cell batteries. The fully approved, programmable modules feature Laird's innovative, event-driven smartBASIC programming interface, which significantly simplifies BLE module integration.

Figure 2. The options on the Laird homepage, to which you're soon going to be a frequent visitor.

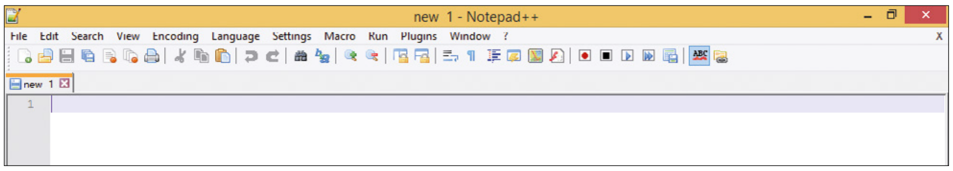
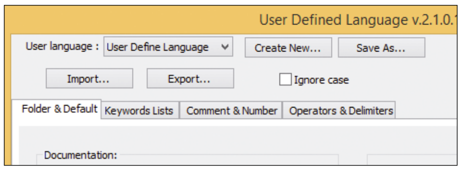

new 1 - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
new 1

Figure 3. The Notepad++ editor you'll be using to write your applications.



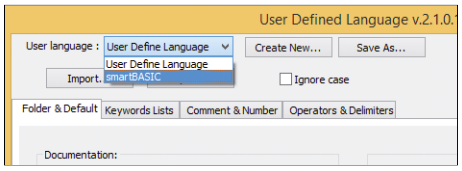
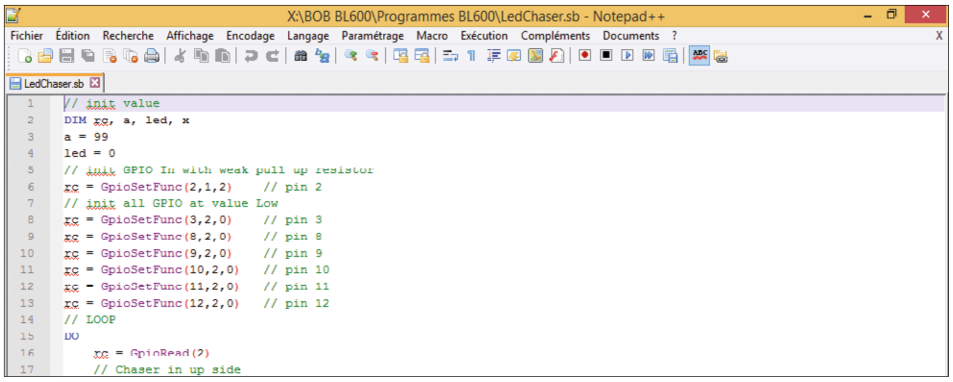


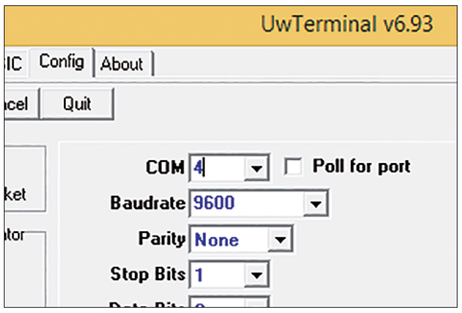
Figure 4. For Notepad++ to learn the smartBASIC syntax...

Figure 5. ... you just have to load the appropriate configuration file provided by Laird.

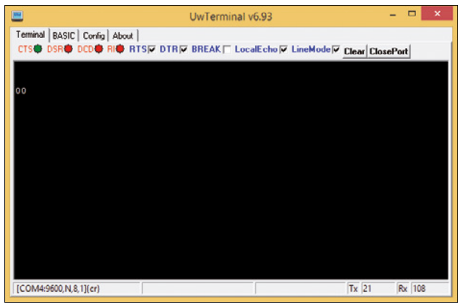

X:\BOB BL600\Programmes BL600\LedChaser.sb - Notepad++
Fichier Édition Recherche Affichage Encodage Langage Paramétrage Macro Exécution Compléments Documents ?
LedChaser.sb

```
// init value
DIM rc, a, led, x
a = 99
led = 0
// init GPIO In with weak pull up resistor
rc = GpioSetFunc(2,1,2)     // pin 2
// init all GPIO at value Low
rc = GpioSetFunc(3,2,0)     // pin 3
rc = GpioSetFunc(8,2,0)     // pin 8
rc = GpioSetFunc(9,2,0)     // pin 9
rc = GpioSetFunc(10,2,0)    // pin 10
rc = GpioSetFunc(11,2,0)    // pin 11
rc = GpioSetFunc(12,2,0)    // pin 12
// LOOP
DO
    rc = GpioRead(2)
    // Chaser in up side
```

Figure 6. The code editing window (full listing at the end of the article).



COM 4
Baudrate 9600
Parity None
Stop Bits 1


UwTerminal v6.93
Terminal BASIC Config About
CTS DSR DCD RI RTS DTR BREAK LocalEcho LineMode Clear ClosePort

```
00
```

[COM4:9600,N,8,1](cr) Tx 21 Rx 108

Figure 7. Configuring the *UwTerminal* communication program

Figure 8. The two 00 on the *UwTerminal*'s black screen – that's a good sign!

file and restart Notepad++. Lastly select *language* and click on *User language* (**Figure 5**), select *smartBasic*, and close. Notepad++ is now configured to display the syntax of the smartBASIC code in color. Try for example the LedChaser.sb program (**Figure 6**), which we'll be coming back to later in this article. But we're not going to be using this now for learning how to compile in the next paragraph; for this, we're going to use the upass program we've already used in the BL600 e-BOB article.

Compiling

Before we go any further, we ought to point out that the BL600 e-BoB must be connected to the PC via a UART/USB interface. For this, I recommend using the FT232 eBoB. It's the first in Elektor's eBoB series, a simple and very effective serial/USB bridge covered in the September 2011 issue [4]. The duo formed by our two eBoB's appears in **Figure 14** and on the adjacent photo. We'll come back to this arrangement later when we talk about our light-chaser application example. But first, the time has come to familiarize ourselves with compilation. To do so, we're also going to use the UwTerminal.exe program that's included in the software bundle downloaded from Laird. At the end of the previous article, our BL600 e-BoB was left in *AutoRun* mode, in which it runs the UART program previously loaded into the module. In order to compile (and transfer), jumper JP1 must be in the *cmd* position (called *debug* in the previous version of the BL600 e-BoB), while JP2 (*ota*) is not fitted. Reset the module either using the button, or by momentarily interrupting the power. Run *UwTerminal.exe* then select the correct COM port on the FT232 eBoB and a data rate of 9600 baud (**Figure 7**).

Press Enter when the black screen appears (**Figure 8**). Although it is possible to do everything at once, we're going to go through it step by step, starting with compiling. Right click on the black background. A window will open. Select *XCompile* (**Figure 9**) and then the file $autorun$.upass.vsp.sb. A compilation window opens, then the result of the compilation is displayed (**Figure 10**). The compilation creates the file $autorun$.upass.vsp.uwc supplied in the download with the previous installment [1]. Now here you are in a position to re-create it yourself. And soon, once you are familiar with the procedure described, you'll be able to modify it as you see fit.

Transferring

We're not going to use the BLE Over The Air service for downloading the program we've just compiled on our BL600 e-BoB. As it's connected to the PC, this is the quickest way to transfer the compiled program. What's more, it lets me show you in detail the steps that are going to help you write your own programs and… make your own mistakes — as that's the best way to learn! The configuration of the jumpers is JP1 in cmd mode and no jumper for JP2 (OTA). As the UwTerminal.exe software is already running on your PC, all you have to do is press ENTER to verify that the BL600 eBoB responds correctly with 00 on the UwTerminal's black screen. If necessary, you may have to reset the BL600 eBOB.
If you've already downloaded a program into the BL600, e.g. with the first article (Over The Air), you must now delete the program memory by entering the following command AT&F 1 which will also restart the module. Select Download (**Figure 9**). Then BASIC, then Load Precompiled BASIC and lastly the compiled program upass.vsp.uwc. The transfer of the compiled program ends with the message +++ DONE +++ (**Figure 11**).

To get a list of the programs downloaded, enter the command *AT+DIR* and to run your program, the command *AT+RUN "upass"* (**Figure 12**).

We've used UwTerminal.exe to compile, transfer, then run the program on your BL600 e-Bob. These three steps can be combined into a single one using the command *XCompile + Load + Run* (**Figure 9**).
Reminder of common commands:
- AT I 0: BL600 revision number
- AT I 3: BL600 firmware version
- AT+DIR: list of the programs in the BL600
- ATZ: reset BL600
- AT&F 1: clear memory and restart BL600
- AT+RUN "xxxx": run program xxxx

Wireless light-chaser command

This little 6-LED light-chaser controlled by the BL600 changes the chaser direction when you press the button. It makes simple use of the BL600's logic inputs/

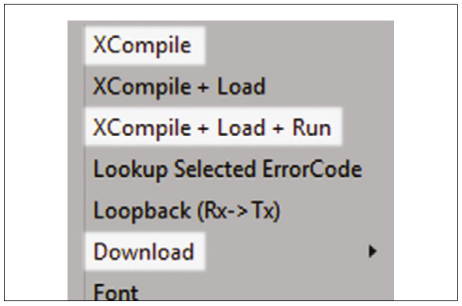
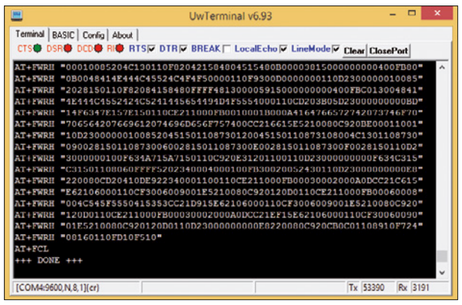


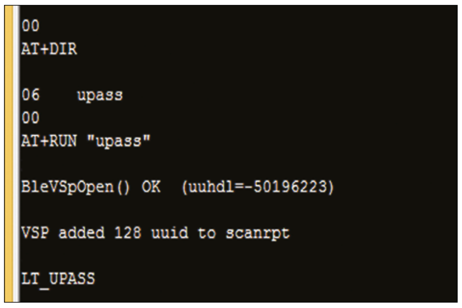
Figure 9. Xcompile menu | Download | *XCompile + Load + Run*


UwTerminal v6.93
Terminal BASIC Config About
CTS DSR DCD RI RTS DTR BREAK LocalEcho LineMode Clear ClosePort

```
AT+FWRH "00010085204C130110F820421584804515480B00003815000000000400FB80"
AT+FWRH "0B0048414E444C45524C4F4F50000110F9300D0000000110D23000000010085"
AT+FWRH "2028150110F82084158480FFFF4813000059150000000000400FBC013004841"
AT+FWRH "4E444C4552424C5241445654494D4F5554000110CD203B05D2300000000000BD"
AT+FWRH "14F6347E157E150110CE211000FB0010001B000A4164766572742073746F70"
AT+FWRH "706564207669612074696D656F757400CC216615E5210080C920BE00011001"
AT+FWRH "10D230000001008520451501108730120045150110873108004C1301108730"
AT+FWRH "090028150110873006002815011087300E002815011087300F0028150110D2"
AT+FWRH "3000000100F634A715A7150110C920E31201100110D2300000000000F634C315"
AT+FWRH "C31501108060FFFF5202340004000100FB30020052430110D2300000000000E8"
AT+FWRH "220080CD20410DE922340001100110CE211000FB00030002000A0DCC21C615"
AT+FWRH "E62106000110CF3006009001E5210080C920120D0110CE211000FB00060008"
AT+FWRH "004C545F5550415353CC21D915E62106000110CF3006009001E5210080C920"
AT+FWRH "120D0110CE211000FB00030002000A0DCC21EF15E62106000110CF30060090"
AT+FWRH "01E5210080C920120D0110D2300000000000E8220080C920CB0C01108910F724"
AT+FWRH "00160110FD10F510"
AT+FCL
+++ DONE +++
```

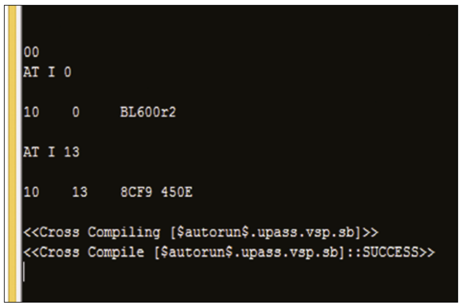
[COM4:9600,N,8,1](cr) Tx 53390 Rx 3191

Figure 11. The transfer of the compiled program ends with the message +++ DONE +++.



```
00
AT+DIR

06    upass
00
AT+RUN "upass"

BleVSpOpen() OK  (uuhdl=-50196223)

VSP added 128 uuid to scanrpt

LT_UPASS
```

Figure 10. The result of the compilation is displayed in the UwTerminal window.



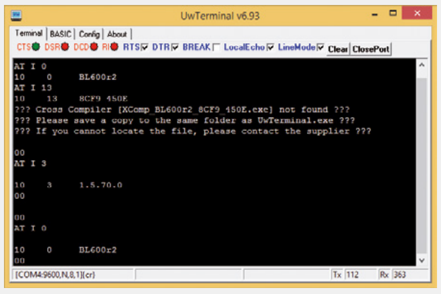
```
00
AT I 0

10    0     BL600r2

AT I 13

10    13    8CF9 450E

<<Cross Compiling [$autorun$.upass.vsp.sb]>>
<<Cross Compile [$autorun$.upass.vsp.sb]::SUCCESS>>
```

Figure 12. *AT+DIR* gives a list of the programs and *AT+RUN* runs your program.

This module offers amazing possibilities – you just need to master it.

Error: Cross Compiler [XComp_BL600…….exe] not found ???

This error occurs when the firmware in your BL600 doesn't match the BL600 bundle downloaded from the Laird Technologies website. The commands for identifying the firmware in your BL600:


UwTerminal v6.93
Terminal BASIC Config About
CTS DSR DCD RI RTS DTR BREAK LocalEcho LineMode Clear ClosePort

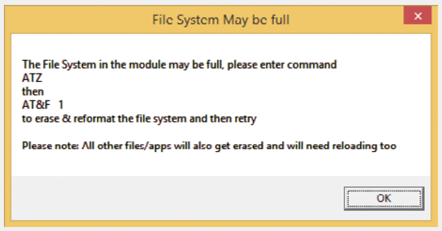
```
AT I 0
10    0     BL600r2
AT I 13
10    13    8CF9 450E
??? Cross Compiler [XComp_BL600r2_8CF9_450E.exe] not found ???
??? Please save a copy to the same folder as UwTerminal.exe ???
??? If you cannot locate the file, please contact the supplier ???

00
AT I 3

10    3     1.5.70.0
00

00
AT I 0

10    0     BL600r2
00
```

[COM4:9600,N,8,1](cr) Tx 112 Rx 363

AT I 3 for the firmware
AT I 0 for the revision

Once you have identified the firmware and revision of your BL600, download the corresponding bundle from the Laird Technologies site [2]. Perform the compilation described again.

Error: The File System in the module may be full



The File System in the module may be full, please enter command
ATZ
then
AT&F 1
to erase & reformat the file system and then retry

Please note: All other files/apps will also get erased and will need reloading too

This error is due to a lack of memory for the new program.
Clear the BL600 memory using the commands ATZ then AT&F 1 and download your smartBASIC program again.

Listing LedChaser.sb

```
// init value
DIM rc, a, led, x
a = 99
led = 0
// init GPIO In with weak pull up resistor
rc = GpioSetFunc(2,1,2)     // pin 2
// init all GPIO at value Low
rc = GpioSetFunc(3,2,0)     // pin 3
rc = GpioSetFunc(8,2,0)     // pin 8
rc = GpioSetFunc(9,2,0)     // pin 9
rc = GpioSetFunc(10,2,0)    // pin 10
rc = GpioSetFunc(11,2,0)    // pin 11
rc = GpioSetFunc(12,2,0)    // pin 12
// LOOP
DO
  rc = GpioRead(2)
  // Chaser in up side
    IF (rc == 1) THEN
    IF (led == 0) THEN : GpioWrite(12,0) : GpioWrite(3,1) : ENDIF
    IF (led == 1) THEN : GpioWrite(3,0) : GpioWrite(8,1) : ENDIF
    IF (led == 2) THEN : GpioWrite(8,0) : GpioWrite(9,1) : ENDIF
    IF (led == 3) THEN : GpioWrite(9,0) : GpioWrite(10,1) : ENDIF
    IF (led == 4) THEN : GpioWrite(10,0) : GpioWrite(11,1) : ENDIF
    IF (led == 5) THEN : GpioWrite(11,0) : GpioWrite(12,1) : ENDIF
    led = led + 1
    IF ( led == 6) THEN : led = 0 : ENDIF
  ELSE
  // Chaser in down side
    IF (led == 6) THEN : GpioWrite(3,0) : GpioWrite(12,1) : ENDIF
    IF (led == 5) THEN : GpioWrite(12,0) : GpioWrite(11,1) : ENDIF
    IF (led == 4) THEN : GpioWrite(11,0) : GpioWrite(10,1) : ENDIF
    IF (led == 3) THEN : GpioWrite(10,0) : GpioWrite(9,1) : ENDIF
    IF (led == 2) THEN : GpioWrite(9,0) : GpioWrite(8,1) : ENDIF
    IF (led == 1) THEN : GpioWrite(8,0) : GpioWrite(3,1) : ENDIF
    led = led - 1
    IF ( led == 0) THEN : led = 6 : ENDIF
  ENDIF
  // tempo : speed
  for x = 0 to 2000
  next
DOWHILE (a != 0)
```

Web Links

[1] BL600-eBoB, Elektor 2/2015, p. 42
www.elektor-magazine.com/140270

[2] https://laird-ews-support.desk.com/?b_id=1945

[3] http://notepad-plus-plus.org/

[4] www.elektor-magazine.com/110553

[5] https://www.youtube.com/watch?v=SxwaVI0Kkk8

[6] Bluetooth Low Energy Wireless Thermometer, Elektor 1/2015, p. 64
www.elektor-magazine.com/140190

outputs. This time, we're not going to look at either analog/digital conversion or pulse width modulation — both of which are possible with the BL600.
The serial link between the FT232 eBoB and our new BL600-eBoB was described in last edition's article. Remember that the operating voltage must be 3.3 V. There's nothing special about the BL600-eBoB's inputs/outputs: 6 LEDs, each with its 470-Ω dropping resistor (approximate value).
As for the BASIC programming, we're going to concentrate here on the general-purpose input/output (GPIO) functions by taking a closer look at the program for this light-chaser (**listing**).

GPIOSETFUNC(nSigNum, nFunction, nSubFunc)
The second argument for this function defines the role of the inputs/outputs on the GPIO pin identified by the first argument.

a. Configuration as a logic input
The push-button is connected to GPIO pin 2, pulled up to high via an internal resistor:
`rc = GpioSetFunc(2,1,2)`
- **nSigNum** = 2: GPIO pin 2
- **nFunction** = 1: port as input
- **nSubFunc** = 2: internal resistor

b. Configuration as a logic output
Each LED is connected via a dropping resistor to one of the outputs (3, 8–12). E.g. configuring output 12:
`rc = GpioSetFunc(12,2,0)`
- **nSigNum** = 12: GPIO pin 12
- **nFunction** = 2: port as output
- **nSubFunc** = 0: output low
  rc is the return code (= 0x0000 when the function has been successful).

GPIOREAD(nSigNum)
Read (logic input) on the GPIO pin identified by the argument.
The push-button is connected to GPIO pin 2 configured as an input with an internal pull-up resistor. This input is read as follows:
`rc = GpioRead(2)`
- **nSigNum** = 2: GPIO pin 2
- **rc** returns the value of the input signal on pin 2: 0 or 1
  Note: if the GPIO input is configured as an analog input, **rc** returns the voltage at the terminals of this input. We'll be using the A/D converter in a future article.

`GPIOWRITE(nSigNum, nNewValue)`
Write (logic output) to the GPIO pin identified by the first argument (if the pin number is invalid, nothing happens). Pins 3 and 8–12 are configured as outputs. In the program, we'll need to set the various outputs to 0 or 1:

`GpioWrite(12,0)`

- **`nSigNum`** = 12: GPIO pin 12
- `nNewValue` = 0: sets to low
  Note: GPIO pins can be configured for pulse width modulation. For the BL600, this function is limited to a choice of up to two pins.
  The value of nNewValue is in the range 0 to N, where N is the max. modulation index corresponding to a duty cycle of 100 %. In a future article, we'll be making use of this possibility of the BL600-SA module by way of an application example in a little vehicle remote controlled via Bluetooth [5].

At the end of this second article devoted to such a small module — in fact the third, if we count the Bluetooth wireless thermometer [6] — there's still a lot more I want to say about it. I've been having fun with it now for over 18 months, and I'm delighted to share my experience with you. In passing, I'd like to thank the whole team at Laird Technologies for their help, and I hope to see you again in the next edition for programming the BL600. ◄

(150014)

Selection of topics

to be covered in the future episodes of this series on the BL600-eBoB:

- handler, events
- the Red Green Blue program
- Low Energy, 5 µA
- the I²C & SPI ports
- Bluetooth communication
- explanation of the wireless remote thermometer program
- writing a program for Android
- writing a program for iOS (hmm… the Apple license isn't free)

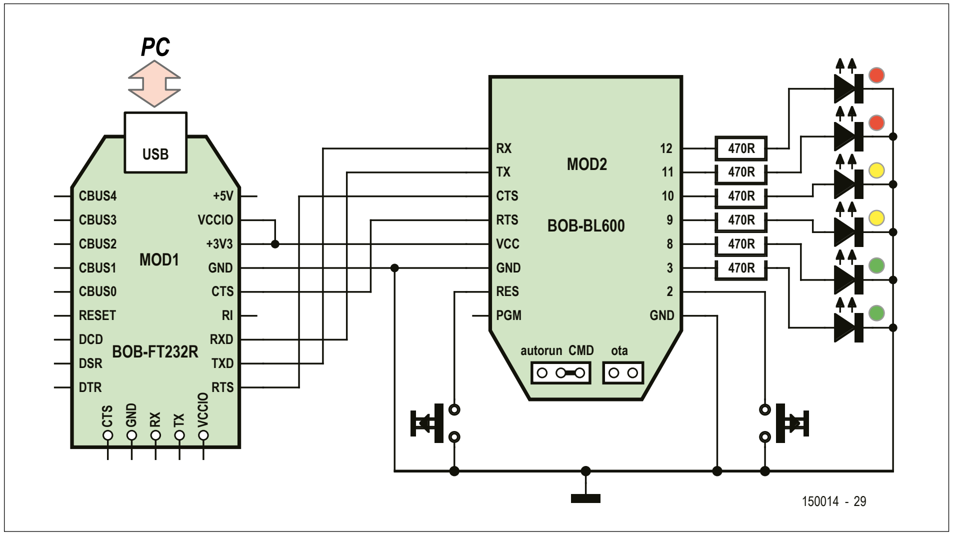


Figure 14. The BL600-eBoB is self-contained, but in order to program it from the PC, I use it in conjunction with the FT232 USB/serial eBoB bridge.

Component List

Resistors
R1–R6 = 470Ω

Semiconductors
D1–D6 = LED, 3mm (select color)

Miscellaneous
K1,K2 = pushbutton
MOD1 = FT232 eBoB, assembled, Elektor Store # 10553-91 (elektor.com)
MOD2 = BL600-eBoB, assembled, Elektor Store # 140270-91 (elektor.com)

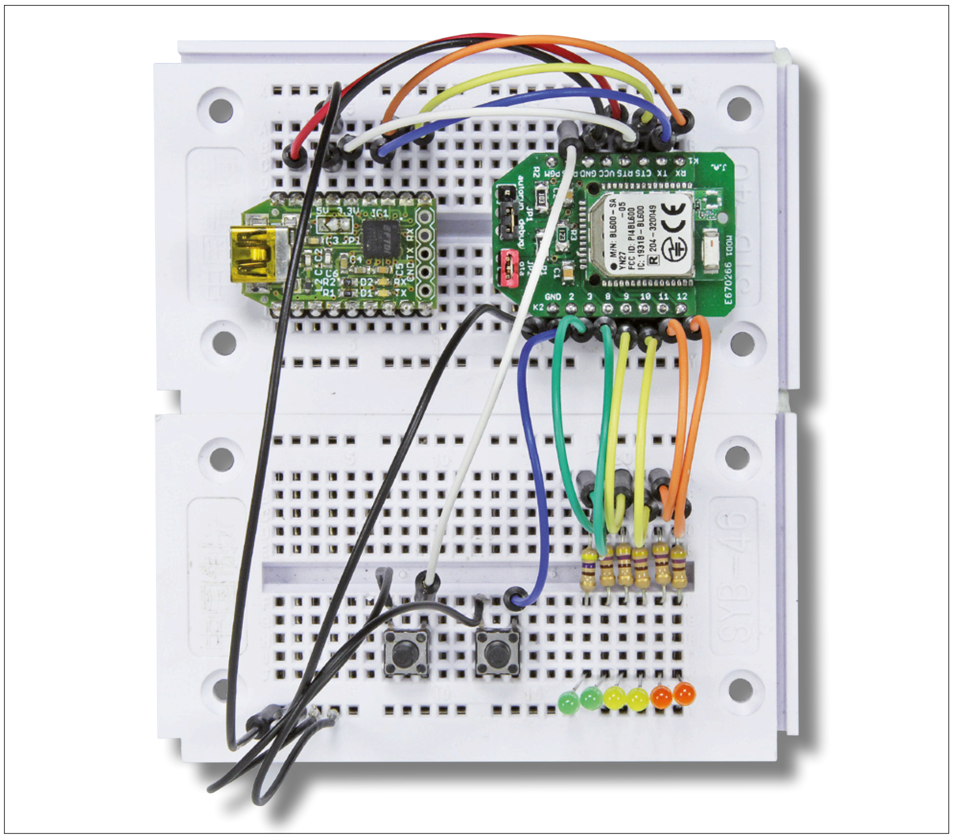


Figure 15. The light-chaser built on perforated prototyping board.