eLABS PROJECT

Bluetooth Low Energy Remote Control

your wish is your command

By **Jennifer Aubinais** (France)

Bluetooth is a protocol where a Master communicates with one or more Slaves; the Slaves cannot talk to each other. The Master is often a smartphone or computer, but this isn't always desirable. A Bluetooth Master module will work very well instead.

During 2015 we presented several Bluetooth Low Energy (BLE) projects in a series of articles [1] around the BL600, a BLE module produced by Laird Technologies. The BL600 is a Slave module, which is why the previous projects needed a Master device in the form of a smartphone or computer to work. The BL600 has a cousin, the BL620 [2], that is capable of taking on the role of Master and which therefore makes it possible to communicate with up to four Slave modules. In this article, we are suggesting using the BL620 as a remote control for a power switch. So there no longer any

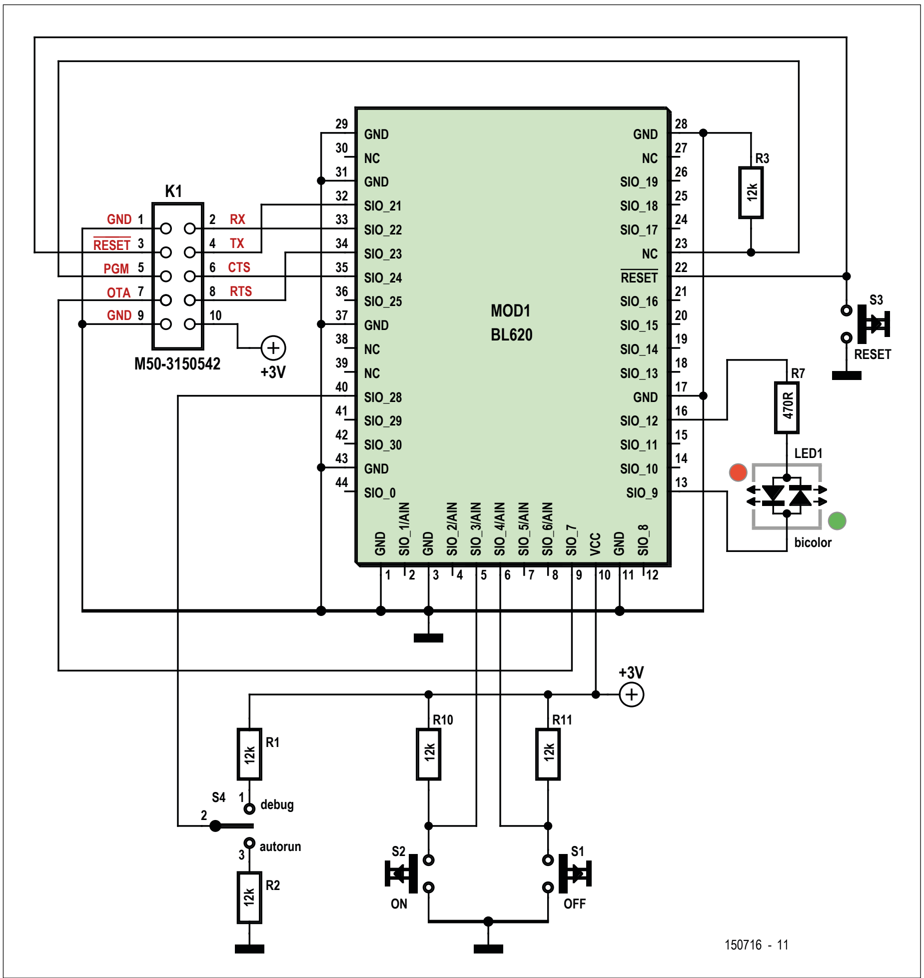


Figure 1. Basic circuit of the remote control.

PROJECT INFORMATION

Bluetooth Low Energy
BLE
BL600

entry level
➨ **intermediate level**
expert level

2 hours approx.

USB–serial 3.3 V adaptor, tools for soldering SMDs, J-Link Edu (optional)

€ 25 / £25 / $30 approx.

Specifications

- Low consumption
- Bluetooth Low Energy (BLE)
- Suitable for 12 V remote control cases

need for a smartphone with app to control the switch, it's easier with a dedicated remote control.

Electronics

The basic circuit of the remote control is fairly simple (see **Figure 1**). In fact, because the BL620 can do everything itself, there's no need for much extra besides. Two push-buttons and a two-color LED are enough; a reset button, programming connector, and switch for selecting the operation mode (debugging or run) complete the picture. The complications arise when we add the power supply.

Remote control implies battery. Remote controls are usually wireless and hence battery powered. Ours is the same, we'll even be using a 12 V battery. So why this voltage, when the BLE module operates at 3 volts? It's the result of a compromise. The problem was to find a small standard case suitable for a remote control with a battery holder, two push-buttons, and LED, and of course with room for the BL620 (which measures 19 × 12.5 mm). The only case that met our criteria was intended for 12-V batteries (in fact, it is a case for a garage

Putting the BLE module to sleep

At the outset, the idea had been to power the BLE module at around 3 V. In this scenario, the BL620 would spend the vast majority of its time in sleep mode, drawing only 0.5 µA. When the circuit supply changed to 12 V for reasons connected with the choice of case, the quiescent consumption was determined by IC1. It was too high, so it was decided to turn off the power when in standby, and the BLE module's sleep mode was set aside. Now it may happen that you are thinking about producing some other BLE project that is going to need the sleep mode. Here's how to go about it:

- `GpioSetFunction(3,1,0x12)`: pin 3 (first parameter) is configured as an input (by the value 1). The value 0x12 indicates that the module will be woken up by a 'low' (the push-button) on this input.
- `SystemStateSet(0)`: this function puts the BLE module into sleep mode. It's up to you to measure the consumption of your module.

Note that when the BL620 "wakes up", it starts its program from the beginning, as if you had pressed the reset button.

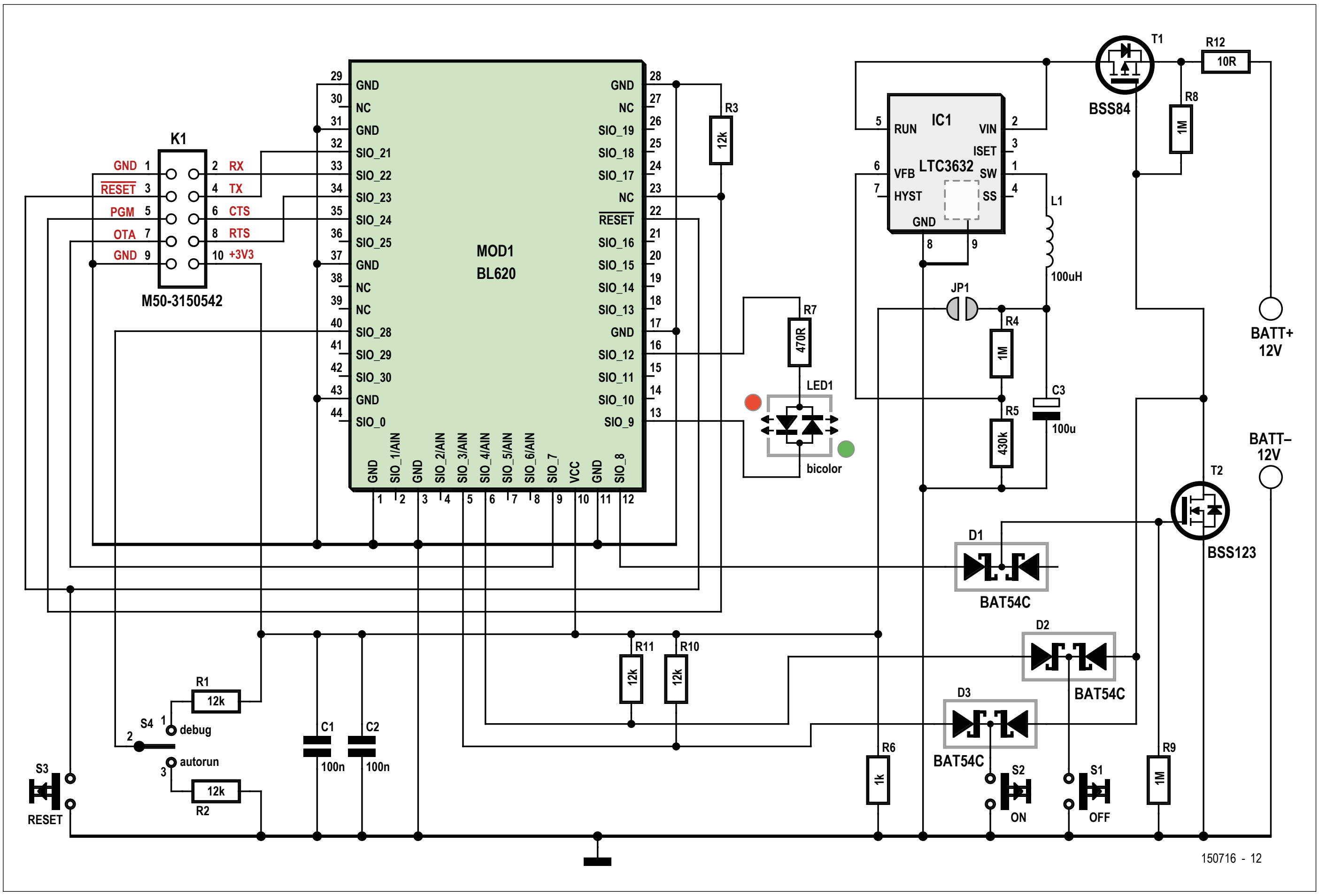


Figure 2. Full circuit with 12 V power supply.

door-style remote control) – that's why! We're going to drop the voltage using a little bit of electronics, and for this task we've chosen the LTC3632 from Linear Technology (now Analog Devices).

For the longest battery life, the circuit's consumption must remain as low as possible. There's no problem in terms of the BL620, it draws only 0.5 µA in sleep mode, but the converter/dropper also has to be fed. Now believe it or not, it's quite greedy: it draws 12 µA! This might not seem like very much to you, but in reality it's much too much, as the 12-V batteries (in 23A format) usually have a capacity of only 55 mAh. So it's out of the question to leave it powered continuously, which is why we've added an electronic on/off switch, with the help of a few transistors and diodes. **Figure 2** shows the final circuit.

The converter/dropper IC1 is powered if transistor T1 is conducting, which is the case when we press one of the two buttons. IC1 comes on and the BL620 is powered. The BL620 then takes care of enabling transistor T2 to ensure the grid of T1 is held low for as long as necessary for it to do its job. So when the button is released, the circuit remains powered.

The dual diodes D2 and D3 have two functions. On the one hand, they protect the BLE module against the 12 V battery voltage (even though R8 limits the current, prevention is better than a cure), on the other they form an AND gate to avoid the two buttons being short-circuited.

IC1 output voltage is adjusted by the voltage divider bridge R4/R5. There's nothing magical about the calculation:

$$V_{out} = 0.8 \times (1 + R4/R5)$$

Using the chosen values, V_{out} ought to be between 2.5 and 2.9 V (5% tolerance on the values of R4 and R5).

Resistor R12 has been added so as to be able to measure the circuit's current without having to break into the supply to insert a current meter. So all you have to do is connect a (milli)voltmeter across this resistor and divide the value

**Table 1. Meaning of LED1 indications.
If the colors are reversed, you've fitted the LED the wrong way round.**

Color	Flashing	Description
Green	No	Sending command to turn switch on after pressing S2.
Red	No	Sending command to turn switch off after pressing S1.
Green/red	Slow	Searching for switch after pressing S1 and S2 together.
Green/red	Fast	Connection error (switch not found).

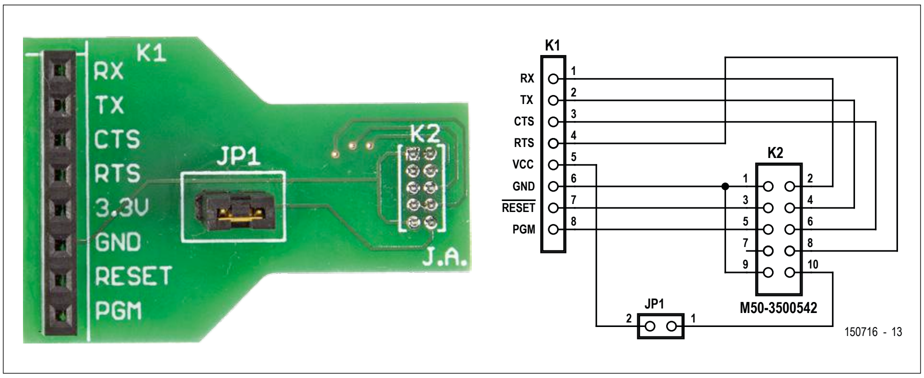


Figure 3. Provision has been made for an adaptor board to make the small connector K1 compatible with the connector on the eBoB-BL600. The jumper lets you disconnect the power if necessary.

measured by 10 to find out how much the remote control is drawing.
LED1 is a two-color LED. Its function of course depends on the application; here, it gives information about the command (switch on/off) and the state of the Bluetooth connection (**Table 1**).
The circuit is completed by a few so-called electromechanical components. First, there is K1, a 10-pin connector that lets you program the BL620 via a serial interface (3.3 V!) using the *UwTerminal* software. If necessary, this connector will also let you load other firmware into the BLE module if you have a J-Link probe from Segger. In this way, you will be able to convert a BL600 into a BL620 and vice versa. For reasons of space, K1 has a 1.27 mm [0.05"] pitch, which is why we've supplemented the project with a little adaptor card (**Figure 3**) that converts the 0.05" [1.27 mm] pitch to a 0.1" [2.54 mm] pitch, with the same pinout as connector K1 on the eBoB-BL600 [1].
S3 is a tiny button for resetting the BLE module – that can always come in handy. Switch S4 has the same function as JP1 on the eBoB-BL600. In *autorun* position, the BLE module automatically runs the main application at start-up — this is the normal operating mode. In *debug* position (*cmd* on the eBoB-BL600), the module accepts "AT"-type commands on the serial port, which is handy for experiments and perfecting a program.
Lastly, JP1 lets you separate the power supply from the rest of the circuit and check that IC1 is doing its job properly before completing assembly.

Software

Like the BL600, the BL620 is programmed in *smartBASIC*, the programming language developed by the manufacturer for their own modules. We're going to be using the functions covered in some detail in the previous articles, in particular events handling described in the 6th part of the series [1]. For more details, you are strongly advised to refer to the various articles [1],[3],[4],[5].
The program is built on *smartZ*, a sort of micro operating system provided by the manufacturer that, once loaded into the module, offers the various commands needed to use the BLE protocol: `hcitool lescan`, `hcitool lecc`, `gatttool --char-write`… (see [4] for more details).
Since the manufacturer has published the *smartZ* source code on GitHub [6], we've taken advantage of this to adapt it to the requirements of the remote control. **Figure 4** shows flow diagram for the program.
As soon as the BLE module is powered by pressing S1 or S2 (or both together), the first thing to do is to set the SIO_8 pin 'high' so as to maintain the power to the remote control when the buttons are released. There are then two possibilities:

- S1 and S2 pressed together: the BLE module starts looking for BLE

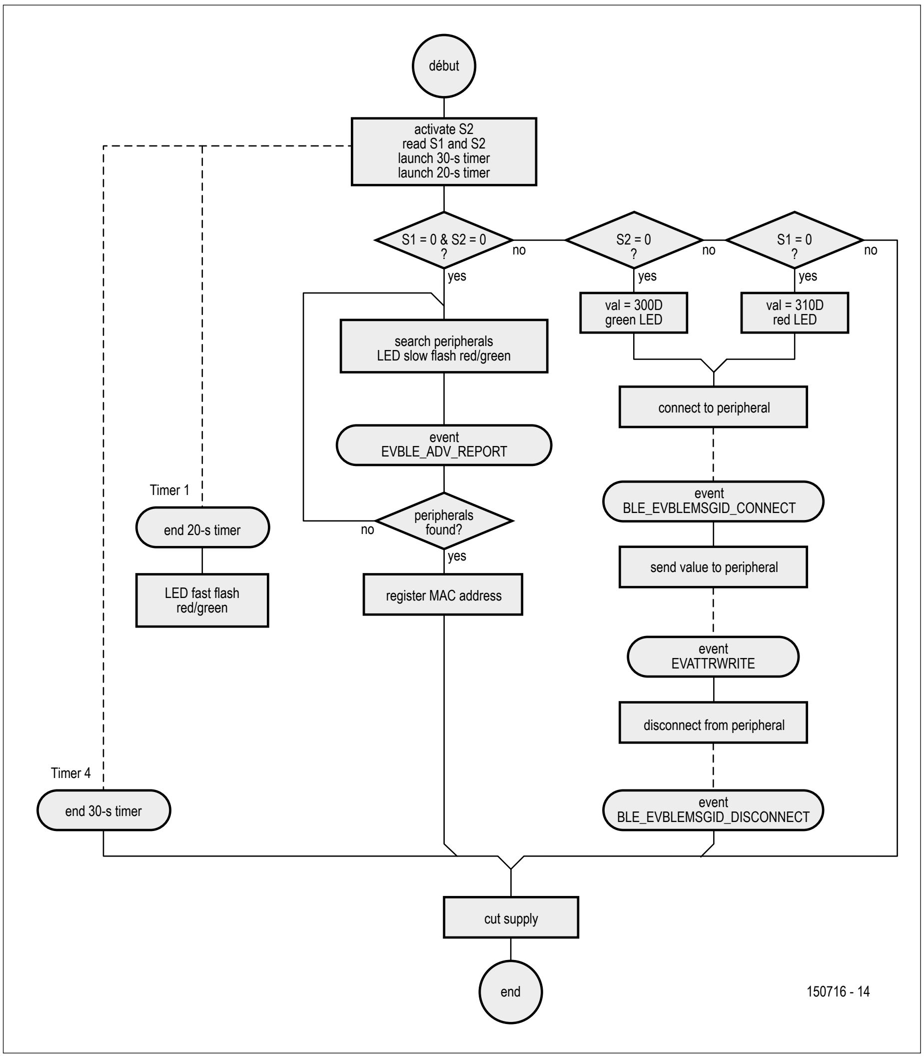


Figure 4. Flow chart for the remote control program. The "normal" path is on the right. The dotted lines indicate the moments when the program is waiting for an event.

peripherals and tries to find the one with the name of the power switch JA_SWITCH. If it finds this peripheral, its MAC address is saved into the BL620's EEPROM. The remote control is then ready to control the power switch. This action can be performed as many times as you like, if you have trouble finding the switch.

- Pressing S1 or S2: the BLE module tries to establish a connection with the power switch. If the connection is successful, it sends the command to either turn the switch on (S2 pressed) or turn it off (S1 pressed), then it interrupts the connection before shutting down its own power by taking the SIO_8 pin 'low'. Starting the remote control takes less than a second. This speed is due to recording the switch's MAC address in the module's EEPROM, which avoids having to search for the switch each time.

A 30 s timer is started when the program is run. If everything goes according to plan, the remote control will de-activate itself well before the end of this time. Otherwise, it will be turned off automatically after this time.

COMPONENT LIST

Resistors
Default: 5%, 0805
R1,R2,R3,R10,R11 = 12kΩ
R4,R8,R9 = 1MΩ
R5 = 430kΩ
R6 = 1kΩ
R7 = 470Ω
R12 = 10Ω

Capacitors
C1,C2 = 100nF, 0805
C3 = 100µF, 5650

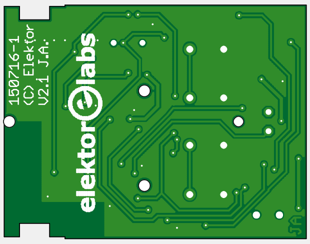
Inductor
L1 = 100µH, 1210 (e.g. Würth 744764920)

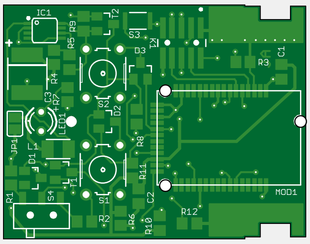
Semiconductors
D1, D2, D3 = BAT54C
IC1 = LTC3632
LED1 = 3 mm, two-color
T1 = BSS84
T2 = BSS123

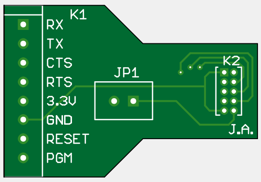
Miscellaneous
K1 = 10-way (2×5) board-to-board connector, 0.05" pitch, Harwin M50-3150542
MOD1 = BL620
S1,S2 = 6×6 mm tactile switch, e.g. B3F-1020
S3 = Panasonic EVPAWCD2A tactile switch
S4 = slide switch, C&K Components, AYZ0102AGRLC

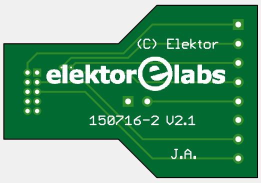
12V battery, size 23A
Case = CamdenBoss 2957-23 (includes battery connector)

Adaptor board
K1 = 6-pin pinheader, straight
K1 = 10-pin (2×5) pinheader, straight, 0.05" pitch, Harwin M50-3500542
JP1 = 2-pin pinheader, straight
1 jumper









References and Web Links

[1] Elektor issues 441–447; series of articles on the BL600
[2] www.lairdtech.com/products/bl620
[3] Bluetooth (Low Energy) controlled 230 V switch: www.elektormagazine.com/140115
[4] Bluetooth 2 to Bluetooth LE Bridge: www.elektormagazine.com/150753
[5] Bluetooth Low Energy wireless thermometer: www.elektormagazine.com/140190
[6] https://github.com/LairdCP/BL620-Applications
[7] www.elektormagazine.com/150716

Construction

Even though the board for this remote control includes a lot of SMDs, it's perfectly feasible to build it using a very fine soldering iron.

Remember to fit the two-color LED (LED1) approximately 1 cm [0.375"] above the board so it can be seen through a little hole in the case.

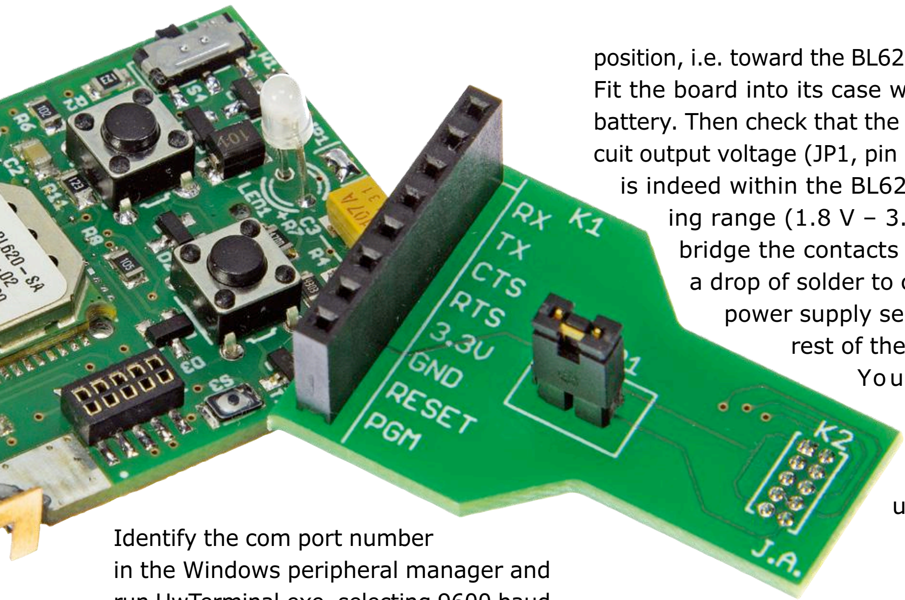
For the moment, leave JP1 on the remote control board open.

Adaptor board

As far as the adaptor board is concerned, connector K1 and jumper JP1 are fitted on the top ("component" side); the smaller connector K2 goes on the bottom ("soldering" side).

Programming

Start by downloading the program (including the UwTerminal programming utility and the libraries) [7]. Set S4 to the *debug* position, i.e. away from the BL620 (downward). Then plug the adaptor board onto K1 on the remote control, then connect up a USB-serial TTL (3.3 V!) interface to the adaptor board, before connecting it to your computer.



Identify the com port number in the Windows peripheral manager and run UwTerminal.exe, selecting 9600 baud and the correct com port.

- Right-click, then from the menu choose *Download* ➞ *BASIC* ➞ *Run*
- Choose the file *"$autorun$.Remote.smartZ.sb"*, the program will start to load.

Once the programming has finished correctly, exit the UwTerminal program and unplug everything. Set S4 to the *autorun* position, i.e. toward the BL620 (upward). Fit the board into its case with its 12 V battery. Then check that the reducer circuit output voltage (JP1, pin 2, IC1 side) is indeed within the BL620's operating range (1.8 V – 3.6 V). If so, bridge the contacts of JP1 with a drop of solder to connect the power supply section to the rest of the circuit.

Your remote control is now ready for use. The first action will be to press both buttons together in order to connect it to your BLE switch. The two-color LED flashes slowly then stops (if it starts flashing quickly, the switch has not been found). Press button S2 to turn the switch on; press button S1 to turn it off. ◄

(150716)

Converting a BL600 into a BL620

If you have a BL600 in stock and not a BL620, here's how to reprogram a BL600 (Slave) module to make it a BL620 (Master). To do this, you'll need a J-Link probe from Segger – the Edu version is good enough.

- Download "BL620 Firmware v12.4.10.0 Revision 1" (www.lairdtech.com/products/bl620 ® *software downloads*, you need to connect and set up an account using "Elektor" as the company).
- Download the "Software for Windows" driver (www.segger.com/downloads/jlink) for the J-Link probe and install it.
- Connect the J-Link probe's JTAG connector to a BL600, like this:

BL600	J-Link (Edu)	Color of wire
PGM	TCK	White
RES	TMS	Blue
GND	GND	Black
VCC	VCC	Red (this is not a supply rail)

- Make sure the BL600 is powered from an external supply – the J-LINK does not supply power.
- Run the command "_DownloadFirmwarev12_4_14_0.bat" contained in the download from step 1.

Congratulations, your BL600 module has become a BL620 module (note this on the module). The reverse, changing a BL620 into BL600, or updating a BL600 or BL620, is also possible by downloading the latest BL600 firmware from the manufacturer's website.

FROM THE STORE

- ➞ **150716-1 :** remote control PCB
- ➞ **150716-2 :** programming adaptor PCB
- ➞ **140270-91 :** eBoB-BL600 module
- ➞ **140190-91 :** BLE thermometer (BLE module mounted on a PCB)
- ➞ **140115-1 :** PCB for BLE 230 V switch
- ➞ **150168-91 :** BL600 module