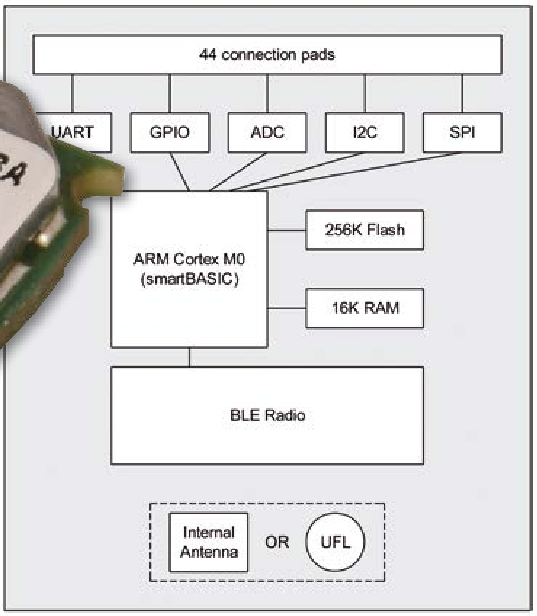
Bluetooth Low Energy Wireless Thermometer

remote temperature display on your smartphone

By **Jennifer Aubinais**
(France)
elektor@aubinais.net



This outdoor thermometer in a waterproof case uses Bluetooth Low Energy 4.0 (BLE) to communicate with a modern touch-screen phone: iPhone 4S (or later), Android 4.3 (or later), by means of a self-contained BL600-SA module designed by Laird. There are no (other) active components—everything is in the mini-module which is programmed in… BASIC!

I'm very keen on Bluetooth, and this thermometer gives me a good excuse to use the latest version of it, which is of interest because the current required is much less than for the previous Bluetooth standards (2.0 and 1.0) — with which the BLE mode is therefore not compatible.

Bluetooth in BASIC

The BL600-SA's smartBASIC programming language (event-oriented) simplifies the inclusion of Bluetooth by making it easier not only to manage sensors connected directly to the module, but also to transmit the measured values to any Bluetooth v4.0 receiver (touch phone or tablet, computer, bridge, etc.) So it's now almost child's play (for big kids!) to communicate by radio with small portable devices, powered by AAA or button cells. The 'A' in the module's reference number indicates that its antenna is built in.

Remarkable for its low consumption, the BL600-SA module is based on the nRFS1822 chipset from

Acknowledgements: Philippe and Laird Technologies support

Nordic Semiconductor and includes all the hardware and software required for radio communication with a raw data rate of 1 Mbps in the 2.402–2.480 GHz band. Main specifications:

- UART, I²C, SPI interfaces
- 28 general-purpose inputs/outputs (GPIO)
- 6 analog inputs (10-bit ADC)
- consumption:
  - 0.4 µA in deep sleep
  - 5 µA in stand-by
- 10 mA during transmission
- easy programming in smartBASIC
- compact: 19 × 12.5 × 3 mm
- free-field range: up to 20 m

Impressive, eh? It's also remarkable for its very modest price. The photo (**Figure 1**) shows the module on the evaluation board (SDK) offered by Laird Technologies. Yes, you have to look pretty hard to find it… The manufacturer seems to be aiming it principally at paramedical telemetry applications (blood pressure, heart rate, temperature, etc.) — but it's up to all of us to extend its fields of application.

The first time I used it, it took me less than an hour to send 1's and 0's to the BL600 from my iPhone and then make an LED flash. Following this encouraging start, I stopped counting the time it took to arrive at my little self-contained outdoor thermometer. Especially not the number of pages of the very comprehensive documentation, nor the time spent on the manufacturer's website [4]. And then there's the eternal question of miniaturization: the module is so densely-integrated that it's difficult but not impossible to solder it by hand (it's like watch-making!) Fortunately, Laird Technologies have found a simple trick for holding the module in position accurately — I'll be coming back to this later.

Flashing

Using this first – very simple – flashing LED program, I was able to verify that when the LED is off, the module only draws 5 µA. The same principle will be used for stopping the module's Bluetooth function at intervals to save the battery.

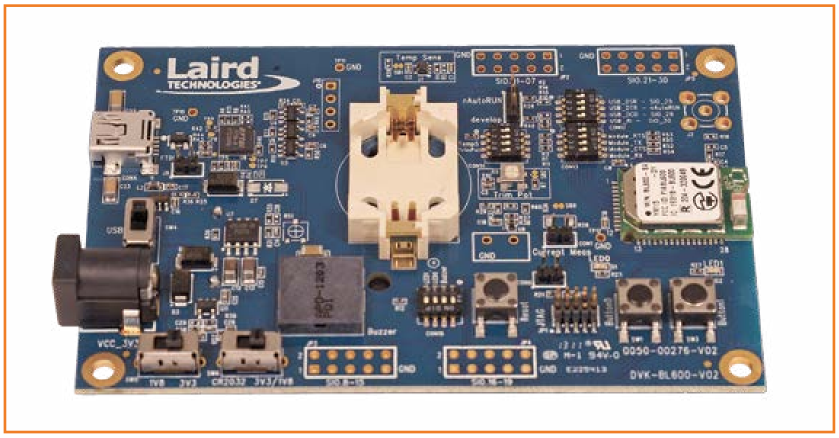


Figure 1.
Judging by the size and population density of the BL600 evaluation board, there must be a lot going on in the discreet radio module (the white rectangle on the right-hand edge).

```
'//**************************************
'// Jennifer AUBINAIS 2014
'// Test sleep with led
'//**************************************
'//**************************************
' LONG TIME : 2 seconds => led off
'//**************************************

FUNCTION Func0()
Dim rc
' led off
rc = GPIOSetFunc(17,2,0)
TIMERSTART(1,2000,0)
ENDFUNC 1

'//**************************************
' SHORT TIME : 100 milliseconds => led on
'//**************************************
FUNCTION Func1()
Dim rc
' led on
rc = GPIOSetFunc(17,2,1)
TIMERSTART(0,100,0)
ENDFUNC 1

'//**************************************
' event
'//**************************************
ONEVENT EVTMR0 CALL Func0
ONEVENT EVTMR1 CALL Func1

'****************************************
' main program
'****************************************
Dim rc
' output GPIO 17 and led on
rc = GPIOSetFunc(17,2,1)
TIMERSTART(0,100,0)
// SLEEP : close uart
```

```
uartclose()
rc = GPIOSetFunc(21,2,1) '// TX
rc = GPIOSetFunc(23,2,0) '// RTS
WAITEVENT
```

Circuit

It won't take us long to go through the circuit. To measure the temperature, I'm using the BL600-SA ADC (MOD1 on the diagram in **Figure 2**) with an NTC thermistor connected to K3. It can be remoted if necessary — the length of the lead is not critical. The measurement is made with the help of a potential divider using a known resistance: R1 = 10 kΩ 0.1 %. The equation for the temperature as a function of the value of the NTC is a logarithmic function that exceeds the BL600's calculation capacity. So this calculation, which requires a data table (temperatures, resistances, thermal coefficient alpha) provided by the thermistor manufacturer [10], will be performed on and by the smartphone and an iOS or Android application. I'll come back to this in a moment.

The supply is via the BL600's SIO_19 pin in order to reduce the consumption to 5 µA in stand-by mode. In deep-sleep mode, the consumption drops to 0.4 µA – but it can only be woken up again by resetting the module or by a change of state on an input. While in stand-by mode, software interrupts allow Bluetooth communication to be re-established briefly, just long enough to see if anyone is trying to connect. I arbitrarily chose a sleep period of 500 ms (configured in the `TIMERSTART` function). When the connection is established, the consumption during transmission is 10 mA.

Connector K2 is used for possible updating of the BL600's firmware via a special JTAG interface. In principle, this function is not needed for this project (but you never know…). Connector K1 lets you program the BL600 from a PC via a 3.3 V serial interface. The FT232R BOB [6] offered by Elektor in the e-shop is perfect for this purpose.

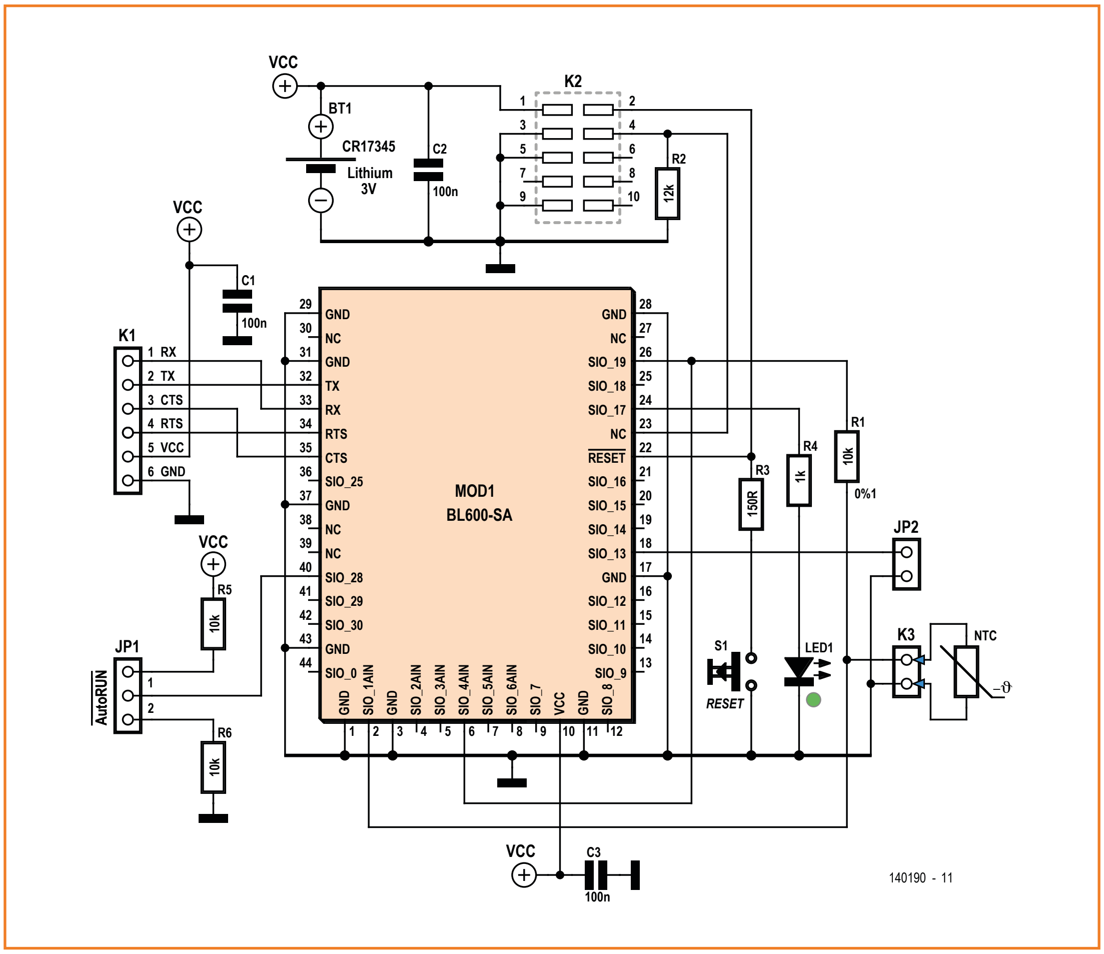


Figure 2.
Circuit diagram of the BT thermometer using the BL600-SA. In the Laird family, the BL600 only supports the LE (low energy) mode, while the BT900 handles both conventional BT and BLE — but it is even smaller and so more difficult to solder!

The LED serves two functions:

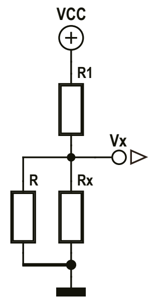
- debugging (with jumper): it flashes all the time and shows the Bluetooth stand-by and module connection modes
- normal (no jumper): it flashes briefly during initialization to indicate that the thermometer has started up correctly

In my initial tests, the data transmission was truncated: I had to choose between waking the module up for longer or transmitting less data. In order to save the battery, I chose the second solution. Example of data sequence communicated by the module via Bluetooth:

PW3012V853C433
PW: supply voltage (battery) to be displayed in the application
V: supply voltage to voltage divider
C: voltage at divider junction R1 / NTC
i.e. a corresponding temperature after calculation of 24.3 °C [see **Calculations** paragraph below]

Calculations

Without going into details, we shall note that the calculation of the thermistor value is achieved by measuring the voltage on a divider.



$$V_x = V_{cc} \times \left(\frac{\frac{RR_x}{R+R_x}}{R1 + \frac{RR_x}{R+R_x}} \right)$$

$$\vdots$$

$$R_x = \frac{V_x R1 R}{V_{cc} R - V_x (R1 + R)}$$

For the temperature, I'm not going to settle for a simple equation between thermistor and temperature by applying a beta coefficient as is usually done:

$$R_{ctn} = R_{25} \times e^{\beta \times \left(\frac{1}{T} - \frac{1}{298} \right)}$$

R_{25}: resistance value at 25 °C
T: calculated temperature in K
β: NTC beta coefficient
R_{ntc}: NTC value at the calculation temperature

I use an alpha α(%/K) coefficient that varies according to the temperature ranges laid down by the manufacturer [1] (**Table 1**), yielding the following equation:

$$R_T = R_{Tx} \cdot e^{\left[\frac{\alpha_x}{100} \cdot (Tx)^2 \cdot \left(\frac{1}{T} - \frac{1}{Tx} \right) \right]}$$

R_T: resistance at T
R_{Tx}: resistance at Tx in the table
T_x: temperature in K lower than the measurement temperature found in the table
T: measurement temperature in K ($T_x < T < T_{x+1}$)
α_x: thermal coefficient at T_x

From this same equation, the temperature calculation, applying the thermal coefficient α_x, will give:

$$T = \frac{1}{\frac{100}{\alpha_x \cdot (Tx)^2} \cdot \ln \left(\frac{R_T}{R_{Tx}} \right) + \frac{1}{Tx}}$$

Table 1.

R/T No	4901	
T (°C)	$B_{25/100}$ = 3950 K	
	R_T/R_{25}	**α (%/K)**
−30.0	16.915	6.1
−25.0	12.555	5.9
−20.0	9.4143	5.7
−15.0	7.1172	5.5
−10.0	5.4308	5.4
−5.0	4.1505	5.2
0.0	3.2014	5.0
5.0	2.5011	4.9
10.0	1.9691	4.7
15.0	1.5618	4.6
20.0	1.2474	4.5
25.0	1.0000	4.3
30.0	0.808	4.2
35.0	0.6569	4.1
40.0	0.5372	4.0

And lastly, an example of a calculation using this latter equation and the following parameters:
R_T = 10506.46 Ω (R_{NTC})
=> 12474 Ω > R_T/T_{25} > 10000 Ω
=> α_x = 4.5
=> T_x = 20 °C = 293.15 K
=> R_{Tx} = 12475 Ω

$$T = \frac{1}{\frac{100}{4.5 \cdot (293.15)^2} \cdot ln\left(\frac{10506.46}{12474}\right) + \frac{1}{293.15}}$$

$$T = 297.01\,K$$

$$T = 297.01 - 273.15 = 23.86\,°C$$

If we content ourselves with 3% accuracy, R1 can be an ordinary (and cheaper) 1% tolerance resistor (instead of 0.1%).

Construction

After finding a suitable waterproof case, I started designing a PCB (**Figure 3**) for building a temperature probe I wouldn't need to touch again for... let's say 10 years! I have myself created the Eagle library for the BL600 (available in the download of this article [2]). My case is a very sturdy, waterproof ABS one, but the internal configuration requires the PCB to have an irregular shape.

I was initially skeptical about the consumption of this new module. That's why I didn't power it from a CR2032 button cell, but a CR123. As a battery holder, I designed an adaptor (Figure 3) that can be plugged onto the main PCB via individual pin sockets. All you have to do is recover the eight pin sockets by extracting them from the plastic housing of a SIL socket header, then solder them onto the oblong PCB, and finally solder the CR123 battery holder onto the same board (paying attention to the polarity!)

Elektor is offering the Bluetooth thermometer in the form of a part-assembled module, with the BL600 **already soldered** [3]. Those who are tempted by reflow-soldering the BL600 module will make themselves a stencil for applying the paste. Three 1.6 mm screws fitted in the holes provided for the purpose let you position the stencil perfectly before applying the solder paste to the PCB. To position the module with the same accuracy, fit the three 1.6 mm screws around the position of the BL600 **from underneath**. Then fit the BL600 module; its three 1.6 mm slots will slide down the screws to position it correctly to within a tenth of a millimeter. I've made a little video to show you [5].

Then all that remains is to put the circuit board in the oven.

Separate eight more pin sockets from their plastic holder and solder them to the board, then cut off the thin end of the sockets so as to leave just the socket part: these will hold the oblong board onto which the battery holder is soldered. There's nothing more to say about the other components, all of them through-hole, except that R2 is only used for updating the firmware, and

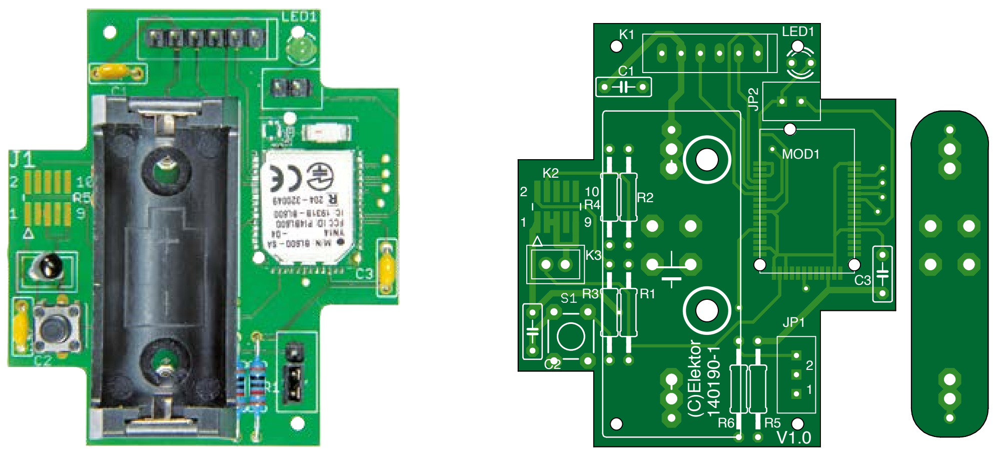
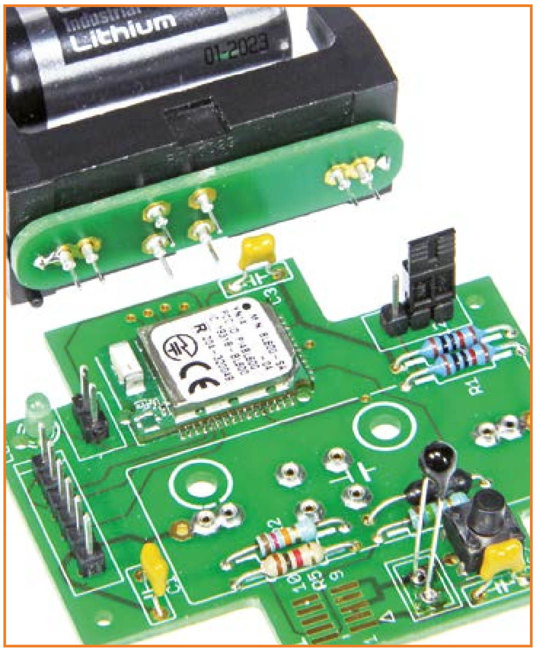


Figure 3.
For those who don't fancy the dainty work, Elektor is offering this PCB with the BL600-SA module already fitted. There are just a few through-hole components left to solder in.

BL600-SA radio module example application



hence may be omitted, as can K2.
Depending on the use you're going to be making of it, you can adopt my system with the removable battery holder, in the same case as me, or any other configuration to suit the requirements and the case of your choice.

Jumper JP1 lets you choose between two modes:

- autorun (position 1) – allows the program to be run automatically when the supply is connected or following a reset.
- command (position 2) – allows access to the module via the serial connection for running AT commands like for example "AT&f 1" which erases the program memory then restarts the BL600 and lets you load the program via the serial interface.

Jumper JP2 lets you go into debugging mode and send the program messages to the serial port. During normal operation, it is not fitted.

BL600 program

Programming the module is simple thanks to smartBASIC from Laird Technologies. It's BASIC just like at school, with simple arithmetic functions to facilitate processing of the data acquired. It lets you create sub-programs and functions and manages the BL600's I/O functions, along with all the more complex functions such as I²C, SPI, ADC, and UART. So this also puts it within reach of beginners. The program can be written in any text editor. I recommend Notepad+, which you can download from the Laird Technologies website. The program is compiled and transferred using the free UWTerminal program.

In the download associated with this article [2] is the source code of my `JATEMP` program which

Component List

Resistors
R1 = 10kΩ 0.1%*
R2 = 12kΩ*
R3 = 150Ω
R4 = 1kΩ
R5,R6 = 10kΩ
10 kΩ thermistor CTN B57891S103F8 (2112816)

Capacitors
C1,C2,C3 = 100nF, 0.2" pitch

Semiconductors
LED1 = LED, 3mm
MOD1 = BL600 module (Laird Technologies)

Miscellaneous
JP1 = 3-pin pinheader
JP2 = 2-way socket e.g. Molex (9731148)
2 jumpers
K1 = 6-pin pinheader
S1 = miniature pushbutton (1555985)
CR123A* battery holder (1650670)
16 PCB pin sockets
Enclosure, Multicomp type G302 (1094697)
PCB # 140190-1
Ready assembled Bluetooth 4.0 thermometer, Elektor Store # 140190-91

* see text
Numbers in round brackets are Farnell / Newark order codes

The author tells us about herself

My grandfather, whom I used to see collecting all sorts of things, is behind my love of electronics. Thanks to him, I am intrepid about the hobby. My journey started at secondary school with the school computer, leading me on to *Brevet de Technicien Supérieur* [Higher Technical Certificate] (with full marks in electronics), then to graduating as an IT engineer in 1992. As an operations engineer on major systems, Windows server support engineer, and vba and vbnet applications developer, I've done a bit of everything, except for Linux and networks. Around 2012, nostalgia for the smell of solder: return to electronics, with a small business associated above all with my projects with Elektor. I specialize in Bluetooth and Wi-Fi interfaces with iOS and Android smartphones.

shows how to establish the link between module and telephone, initializing the ADCs, then after half a second, the module transmitting the data to the telephone: battery voltage, divider supply voltage, divider midpoint voltage, in the form of a string like this: PW3012V853C433. The app on the phone performs the actual calculation.

iOS & Android programs

Laird Technologies offers a download of the source code for the `BL600 Serial` application for iOS, but as I am experienced with iOS, I preferred to write my own application [7] which displays two pieces of information: temperature and connection status. I then incorporated the sources of the `BL600 Serial` program (`UARTPeripherial.c` and `DataClass.c`). I haven't created a connection and disconnection button. I've replaced this action by a thread which looks for the thermometer JATEMP when it is disconnected (it's the thermometer that disconnects itself) and connects itself.

As I didn't have any experience in Java programming, the program under Android gave me quite a headache.

Thanks to the French developpez.com forum, where I found some help, above all for creating a homogeneous interface for all sizes of Android phones. I've created my own BL600 Serial application (code and interface) but under a new name and in a simplified form [8], just for connecting to my JATEMP thermometer. The "Scan and Connect" button made it possible to receive the raw thermometer data character string; I've replaced it with a thread as under iOS (see above).

For iOS and Android, the temperature calculation corresponds to the description above, with the addition of the details. The application's screen background changes according to the temperature measured. These two processing elements are coded in the `jatemp.c` source code. To avoid remaining connected indefinitely to the thermometer, the iOS application stops after 5 mins (or if you press the phone's Home key). Under Android, the application no longer looks for the thermometer after 5 mins or if you press the Home key; the application is not actually stopped.

To establish communication with the phone and connect the thermometer for the first time, you **don't have to do anything**. Everything is automatic.

The Fridge is not a Faraday cage

When I was doing my testing in early September 2014, we were enjoying an Indian Summer in Paris: difficult to obtain temperatures below 20 °C for any length of time. So I put the thermometer in the ice-bow for a quarter of an hour in order to make a low temperature reading the moment I took it out again. While I was waiting, I went back to putting the finishing touches to the Android application I was currently tweaking;

how surprised I was to see the temperature displayed by the Android phone quickly dropping! To check, I went to try the same test in my neighbor's ice-box, where I also watched live as the temperature dropped to 4 °C. I deduced from this that the door seal allows the Bluetooth signal to pass (a characteristic of refrigerators that domestic appliance manufacturers would do well to mention in their data sheets).

Conclusion

I hope I've convinced you of the interest of the BL600 and maybe even to try the adventure with the reflow oven, or even hand soldering, if you have dainty fingers.
For implementing an application based on the BL600 module, the drawback of the miniaturization of the contacts is more than made up for by the advantages of very easy construction and by the simplicity of its smartBASIC language and of the programming via the module's UART interface. And that's only the beginning… In the meantime, Laird Technologies is offering the BL620, which is the BL600 master: the same hardware but new firmware, capable of communicating with other BL600 modules. This opens up new prospects that Elektor and I will be sure to invite you to explore with us in the near future.

[140190]

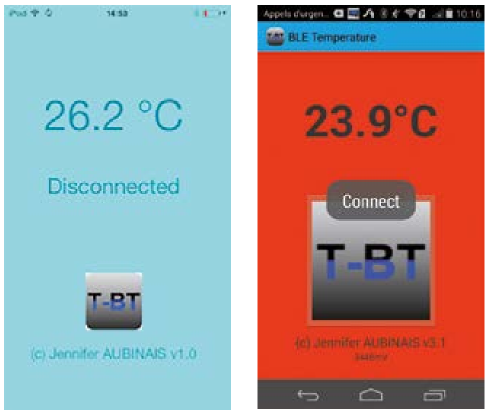


Figure 4.
The thermometer application on the screen of iOS and Android touch phones.

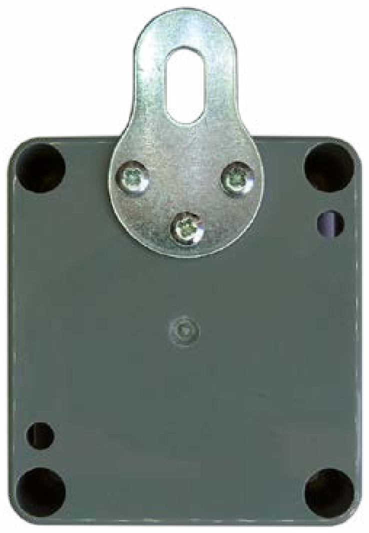


Figure 5.
The author's prototype, in a waterproof case and fitted with a sturdy screwed hook.

Web Links

[1] www.lairdtech.com/Products/Embedded-Solutions/Bluetooth-Radio-Modules/BL600-Series/

[2] www.elektor-magazine.com/140190

[3] www.elektor.com/bluetooth-thermometer

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

[5] www.youtube.com/watch?v=0YlKxtYwQiE

[6] www.elektor.com/ft232r-usb-serial-bridge-bob-110553-91

[7] https://appsto.re/fr/XTwnV.i

[8] https://play.google.com/store/apps/details?id=com.JA.bletemperature&hl=fr

[9] www.aubinais.net

[10] EPCOS NTC
www.epcos.com/inf/50/db/ntc_13/NTC_Leaded_disks_S891.pdf
www.epcos.com/blob/531152/download/2/pdf-standardizedrt.pdf

NTC Thermistors / General technical information:
www.physics.queensu.ca/~robbie/ENPH354/NTC-Thermistors-Technote.pdf