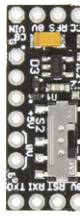
eRIC Nitro

By **Alex Robertson** (Scotland)

In 2015 I entered an "Internet of Things" design contest sponsored by LPRS, a UK manufacturer and supplier of low-power radio solutions, and RS Components' DesignShare. The prize: the winning idea taken all the way from concept through to design and manufacture, supported by LPRS and made a reality by Elektor Magazine. There were many submissions so I was delighted to learn I had the winning entry: eRIC Nitro – a wireless-enabled Arduino board.



My entry to the LPRS Design Contest [1] is based on experience gathered from my home electronics projects. I enjoy the practical aspects of taking my ideas and making them real with a physical board. However, instead of PCBs, these are built using proto boards, point to point wiring and a soldering iron. It can be a lot of fun, but sometimes building a board this way is a time consuming step I could do without. I'm sure other project makers have the same feeling and prefer spending their time on testing out their ideas. Radio communication fascinates me and I have patiently hand built many wireless sensor projects.

There is something magical in being able to transmit and receive data through the air ("aether") without wires and the LPRS eRIC module [2] makes it very easy. For me, having a pre-assembled wireless Arduino board ticks all my design boxes. It reduces the time to test out an idea and for IoT projects it's a perfect fit. The key features of the eRIC Nitro board are given in the inset.

eRIC Nitro Board Key Features

- LPRS eRIC module with a subset of eRIC signals brought out to user accessible pads on the PCB
- PCB form factor similar to Arduino Pro Mini
- Small size: 2.5 x 5.5 cm
- ATmega328 processor with Arduino bootloader with direct serial connection to eRIC module
- 8-MHz operation
- On-board 3.3-V regulator providing power to eRIC module and ATmega328 device
- Header for FTDI friend connector, to allow ATmega328 programming from Arduino IDE and also to provide serial communication interface
- Quickly deployable as a battery-powered low-power programmable RF node
- Total current consumption in low power mode <100 µA
- Open source, open hardware design

Powerful low-power radio

the module transmits it out on its RF interface. Data received on the module's RF interface gets transmitted on its SDO pin (Serial Data Out). By connecting a simple wire antenna, a power supply decoupling capacitor and power to a standalone module you can give any project the ability to communicate wirelessly. Any microcontroller with a hardware or software UART can easily communicate with eRIC.

The capabilities of the module don't end at being a serial bridge. What makes eRIC a rock star of the transceiver world is that the existing firmware in the device can be replaced with user MSP430 code (for free!). If you want to get into the internals of the transceiver and replace the firmware, you will need Texas Instruments Code Composer Studio (CCS) to compile your MSP430 application. On the LPRS website a demo CCS project is provided to get you started. The demo code is written in C.

The benefits of going down this road is that it can reduce the component count in your project since the GPIO pins on the transceiver are fully programmable, even supporting A-to-D functionality. LPRS also supply a PC utility to allow you to flash your code into the module. We won't delve further into programming the eRIC but suffice to say, the eRIC Nitro board has all of the connections needed to put the eRIC into bootloader mode to allow the MSP430 firmware to be replaced.

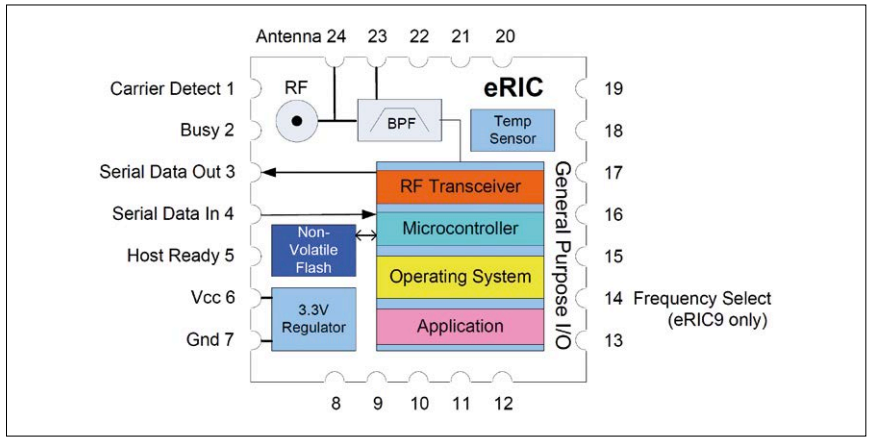


Figure 1. X-ray of eRIC revealing its guts. (Source: LPRS)

Circuit description

Arduino boards are easily programmable and have the major benefits of a free development environment (IDE), a huge development community and a vast library of user contributed open source software. If you want a software library for a sensor or a peripheral chip, it probably already exists for the Arduino. So basing the eRIC Nitro design on Arduino hardware was an easy choice.

The Arduino section of the eRIC Nitro is derived from the open-source Arduino Pro Mini. There are a few changes though. Looking at the schematic in **Figure 2**, you'll see the addition of inductive filter L1 at the ATmega328 (IC1) power supply, recommended by its manufacturer to improve the performance of the analog-to-digital converter. Another change is the clock source. The original 16-MHz resonator has been replaced by an 8-MHz version (X1). This keeps the MCU's clock speed within the datasheet specification for a 3.3-V supply.

It was a design decision to keep the pinout of the board as close as possible to the Pro Mini original. With so many peripheral ICs available with an I²C connection, it makes sense to bring Arduino's I²C pins A4 and A5 out on connector K2, replacing the A2 and A3 of the original Pro Mini design. The eRIC module (MOD1) reset pin (RST) is also made user acces-

Meet eRIC, the easy radio transceiver module

The eRIC radio transceiver module is the jewel in the eRIC Nitro design. eRIC stands for easy Radio Intelligent Controller and is a 24-pin module (**Figure 1**) containing a Texas Instruments CC430F5137 IC, a mixed-signal System-on-Chip (SoC) that combines an MSP430 microcontroller core with RF circuitry.

The eRIC modules are licence-exempt ISM band radios operating in the 433-MHz, 868-MHz or 915-MHz ISM frequency bands. Two versions of the module are available: eRIC4, operating in the 433-MHz band, and eRIC9 for both the 868-MHz (UK and Europe) and 915-MHz (USA) bands. The modules used in this project come configured as standard 'serial bridge' modules. Send serial data into the eRIC SDI (Serial Data In) pin and

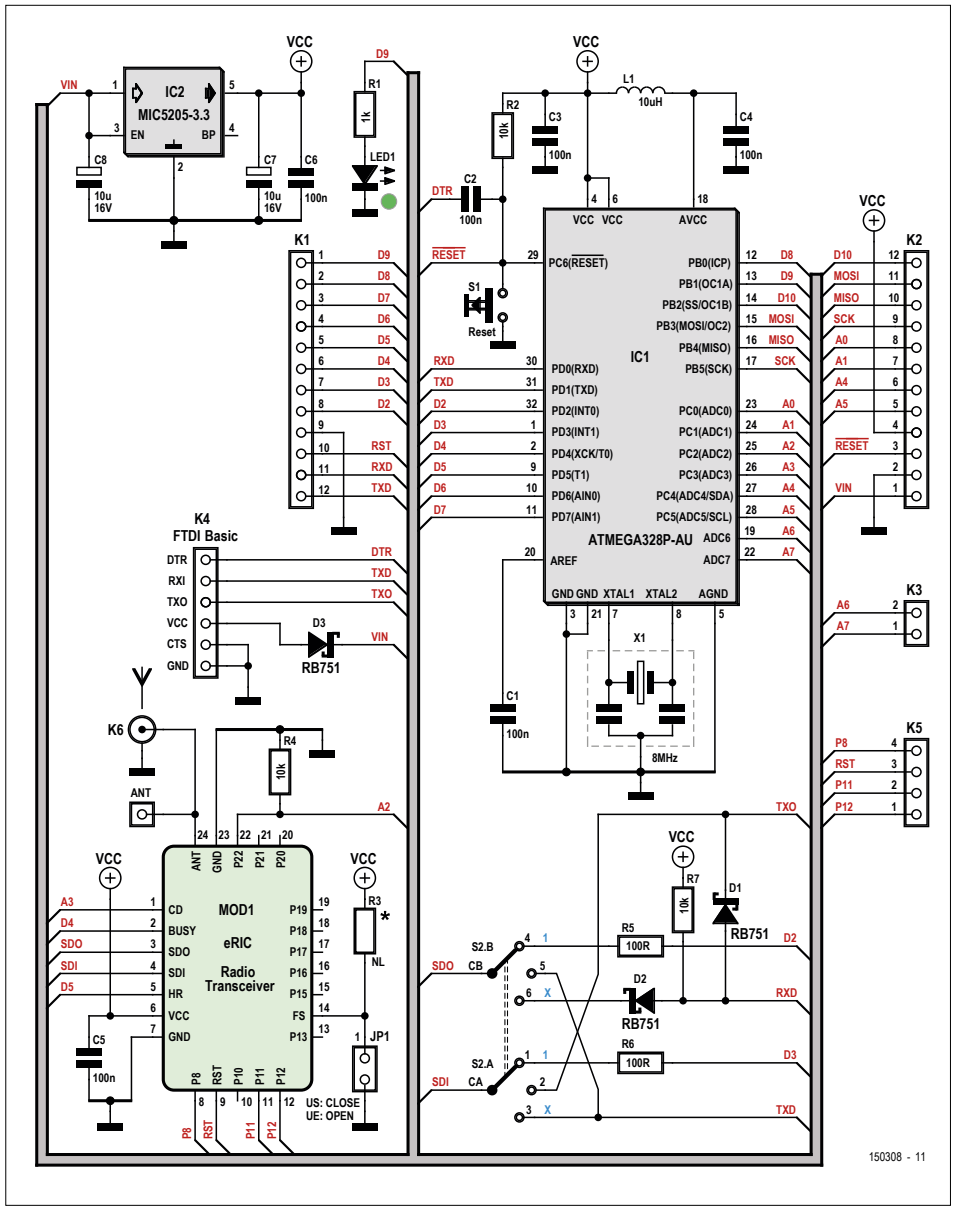


Figure 2. Schematic of the eRIC Nitro board. eRIC sits in the lower-left corner, the rest is Nitro.

sible and is brought out to pins on connectors K1 and K5 to replace one of the original Pro Mini RESET pins. Having the MCU and eRIC reset pins separate gives the user more control of the eRIC module. A user program can reset the eRIC module without having to reset the MCU. Voltage regulator IC2 provides a supply voltage V_{CC} of 3.3 V. IC2 is rated at 150 mA at 3.3 V output. However, the output current capacity is directly related to the input voltage V_{IN} on connectors K2 and K4. If you need this level of current then please refer to the MIC5205 datasheet for more information on calculating the maximum output current for any given V_{IN} since a high V_{IN} voltage and a high load current on the IC2 OUT pin could end up with IC2 overheating possibly causing permanent damage to IC2 and to the eRIC Nitro board.

The eRIC Nitro does not have the 'power present' LED of the Pro Mini. This was removed to reduce the board current consumption — an important requirement for low-power environments. It's always convenient to have an LED under program control and on the eRIC Nitro it's connected to pin D9 on connector K1.

Switch S2 and the passive components D1, D2, R5, R6 and R7 all play important roles in connecting the eRIC SDI and SDO pins to the MCU. These components permit SDI and SDO to be connected to the MCU hardware UART RXD and TXD pins, or software UART using pins D2 and D3 or to bypass the MCU completely while remaining accessible via the TXO and RXI pins of connector K4.

R3 and JP1 are only needed when an eRIC9 module is used. JP1 is needed to select the module's frequency (open for 868 MHz, closed for 951 MHz). R3 is not really needed as the module includes a pull-up resistor; it is on the board just in case.

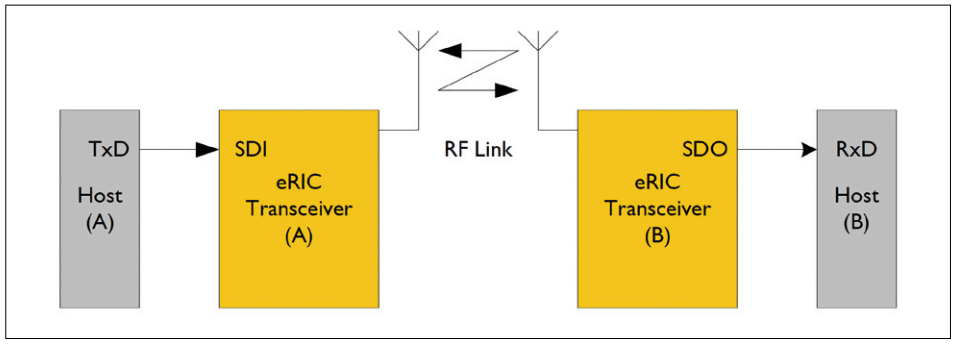


Figure 3. Serial bridge system overview. (Source: LPRS)

Sending data using eRIC

Under the bonnet of the eRIC module is where data received on SDI gets encoded before transmitting the SDI data over the RF interface. The eRIC firmware adds all necessary preamble, message length and message CRC information. Similarly, the firmware strips the same preamble, message length and CRC encoding data from RF data received by the module before transmitting it out serially on SDO.

The default serial SDI/SDO pin data rate is 19,200 baud, with an over-the-air (OTA) data rate of 38,400 baud. The SDI/SDO pin data format for one character is one start bit, eight data bits, no parity and one stop bit (19200n81). Data received on the SDI pin is automatically put into an internal data buffer (maximum 250 bytes). The module transmits the buffer contents over the RF inter-

face (emptying the buffer in the process) when it detects no data has was received on the SDI pin for a period equal to two characters (**Figures 3 and 4**).
Sending and receiving data is easily done with no need for a hardware handshake between eRIC and the MCU. Receive and transmit using the SDI and SDO pins is easily achievable just using software delays (for the two-character transmit delay on the SDI pin). If a hardware handshake is needed then the eRIC can support this using Clear to Send (CTS) and Request to Send (RTS) signals on the respective BUSY and HOST READY (HR) pins. The BUSY output pin goes low when the module is ready to receive data on the SDI pin. It will not send any RF data it has received if the HOST READY input pin is driven high by the MCU. By default the eRIC firmware disables the handshaking functions BUSY and HOST READY. If a user wants to enable the handshaking signals they can send the string data `ER_CMD#A51` to the module. It will trap this data string, recognizing it as a valid command and enable the hardware handshake signals. BUSY and HOST READY are connected to Arduino's D4 and D5 pins. If hardware handshaking is not required then D4 and D5 are free for Arduino program use.

eRIC modules have many neat features, one being the ability to detect if a radio frequency band is currently occupied by another source transmitting data. The module has a Carrier Detect (CD) pin which swings High to flag an RF carrier detected within the radio receiver bandwidth. In the eRIC Nitro the CD pin is connected to Arduino's A3 pin. eRIC command `ER_CMD#T8` is also useful for a similar purpose as it returns the Received Signal Strength Indication (RSSI) of the last packet received.
The eRIC module firmware permits changing many of the radio transceiver parameters such as transmit power, receive sensitivity, low power modes, SDI/SDO baud rate, and radio transmit and receive frequency. The module's datasheet describes these in more detail.

Antenna connection and radio range

As might be expected, the eRIC Nitro needs an antenna to transmit data using the RF interface. There are three antenna possibilities:

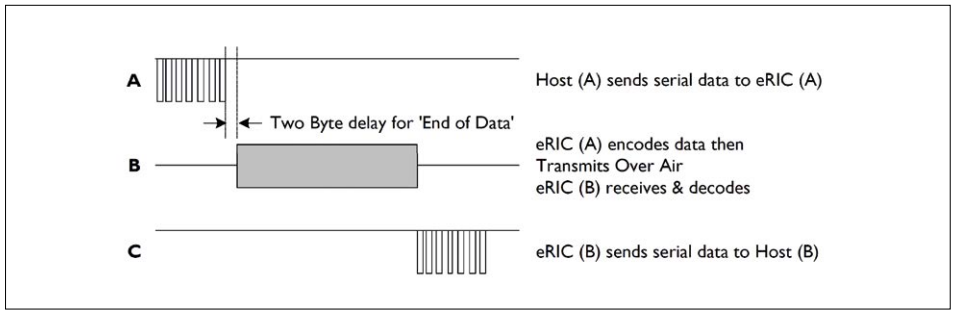


Figure 4. Serial bridge timing diagram. (Source: LPRS).

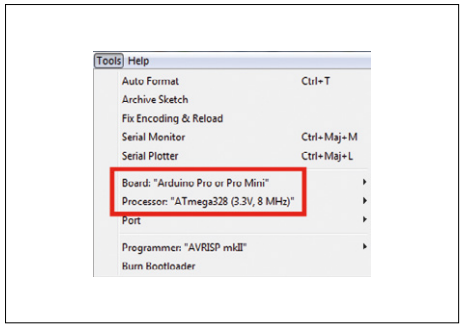


Figure 5. Choosing the right board in the Arduino IDE. It may look slightly different depending on the version of the IDE.

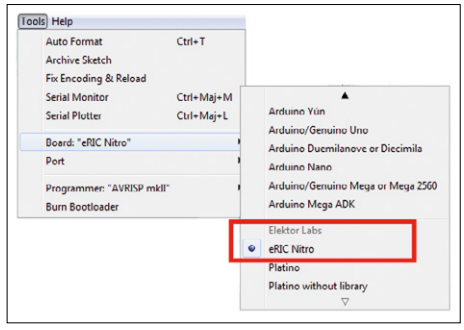


Figure 6 – It is even easier if you have the Elektor.Labs Arduino board package [4] installed.

- an SMA antenna on edge connector K6;
- an external antenna connected to the eRIC module UFL antenna connector;
- a minimum cost ¼-wavelength wire antenna.

When using a wire antenna, solder the relevant length of (stiff) wire to the ANT pad on the PCB (approximately 171 mm for the eRIC4 and 82 mm for the eRIC9 operating at 868 MHz). It goes without saying that when using a wire antenna, there is no need to have the SMA connector on the board and the SMA position on the PCB can be left unpopulated. If using the eRIC9, leave jumper JP1 open when operating at 868 MHz. JP1 needs to be closed to TX/RX at 915 MHz.
The transmit/receive range depends on many factors. These include the radio power (max 10 mW for eRIC4) and sensitivity, obstacles in the path of the radio signal, antenna type and also radio frequency. SMA and UFL antennas should give better performance than a wire antenna, but a wire antenna will still give good results. The physics of path losses say that the range at 433 MHz exceeds that at either 868 MHz or 915 MHz.
Keep in mind the legal requirements of using an ISM radio, both in terms of transmit power and transmit duty cycle [3].

Programming

Programming the eRIC Nitro couldn't be easier as the good folks at Elektor have pre-installed an Arduino bootloader into the board's ATmega328 chip. This is a derivative of the standard Arduino Uno Optiboot bootloader, conveniently flashing LED1 during programming. The board can be programmed using the Arduino IDE (version 1.0.6 or greater), and selecting board "Arduino Pro or Pro Mini (3.3 V, 8 MHz)" or "Arduino Pro or Pro Mini" plus a processor "ATmega328 (3.3 V, 8 MHz)" (**Figure 5**, this depends on the version of the IDE). An Arduino board file [4] is also available for extra ease of use. Add this into the IDE (v1.6.6 and up) and the eRIC Nitro appears in the list of supported boards (**Figure 6**).
Just as with Arduino Pro Mini boards, header K4 has the same pinout as the 6-pin end connector of FTDI basic breakout boards and FTDI TTL-232R cables. The FTDI chip converts USB signals to serial signals and allows programming the ATmega328 from the Arduino IDE. The

USB-to-serial function is also useful when you need to communicate to the eRIC Nitro from a PC. Always use a version of the FTDI chip having 3.3-V TTL signal levels and not the 5-V TTL version. The Nitro board will be damaged if you don't use the 3.3-V version. And always make sure you are connecting the FTDI pins to the matching pins of K4. The green and black wires match up with the "G" and "B" text on the Nitro PCB. Get this wrong and the Nitro board will be damaged.

When using the Arduino IDE to program the ATmega328, put switch S2 in the "upper" position (switch button closest to connector K1). While in this position eRIC's SDI-SDO pins are connected Arduino's D3-D2 pins. When switch S2 is in the 'lower' position (switch button closest to connector K2), eRIC's SDI-SDO pins are linked to Arduino's UART TXD-RXD pins and will interfere with the programming process.

When using the easyRadio Companion PC utility (available from the LPRS website) to talk directly to the eRIC via K4 (bypassing the ATmega328), put S2 in the 'middle' position. Before doing so, you maut force the ATmega328 into reset by connecting K2 pin /RESET to 0 V (with a jumper for instance).

Example software

Elektor.Labs have kindly released example Arduino hardware and software files [5] to get you started with the eRIC Nitro board. Software example `eRIC_blink.ino` is a good starting point to get you up and running with the Nitro board. With two boards you can begin two-way communication and example `eRIC_bridge.ino` is a great template for your future eRIC projects. Expect to see further software examples appear in the future.

You 2 can build your own

This project is open hardware and software and all the files needed to build and program your own eRIC Nitro board can be downloaded from GitHub [5]. The PCB design files are available as DesignSpark PCB and Eagle files. Note that assembling the board can be challenging as some components are rather hard to solder manually. The resonator is one of those; the Schottky diodes are so small that it's hard to see their pads. For this reason pre-assembled boards can be bought from the Elektor Store [6]. The eRIC module is not mounted and must be ordered separately because you must choose the right one. Anyone with lesser soldering skills should be able to mount the radio modules manually.

Component List

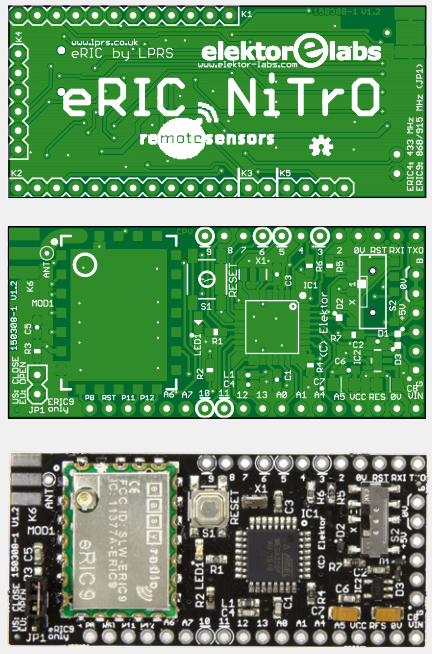
Resistors
All 5%, 50V, 0.1W, 0603
R5,R6 = 100Ω
R1 = 1kΩ
R2,R4,R7 = 10kΩ
R3 = 10kΩ

Capacitors
C1–C6 = 100nF, 0603
C7,C8 = 10µF 16V, tantalum

Inductor
L1 = 10µH, 0603

Semiconductors
IC1 = ATmega328P-AU
IC2 = MIC5205-3.3
D1,D2,D3 = RB751
LED1 = LED, green, 0603

Misc.
X1 = 8MHz resonator, SMT
K1,K2 = pinheader, 1x12, 0.1'' pitch
K3 = pinheader, 1x2, 0.1'' pitch
JP1 = pinheader, 1x2, 0.1'' pitch w. jumper
K5 = pinheader, 1x4, 0.1'' pitch
K4 = pinheader 1x6, 0.1'' pitch
K6 = SMA edge connector
S1 = tactile switch
S2 = DP3T slide switch
MOD1 = eRIC4/eRIC9 SoC Radio Transceiver
PCB 150308-1 v1.2



Design, Make, Use

The eRIC Nitro board is a fantastic platform for Arduino projects requiring wireless connectivity. The Internet is full of sensor, actuator and Wi-Fi breakout boards just waiting to be connected to this board. And if you want to progress further to programming in the native environment of the eRIC module, this is supported too. What project will you create? ⏮

(150308-I)

Web Links

[1] www.lprs.co.uk/smart-eric-iot-competition/
[2] www.lprs.co.uk/easy-radio/eric/
[3] www.ti.com/lit/an/swra048/swra048.pdf
[4] https://github.com/ElektorLabs/Arduino
[5] https://github.com/ElektorLabs/150308-eRIC-Nitro
[6] www.elektor.com/150308
[7] www.elektor-labs.com/eric-nitro