RFX 328p dev/deployment board - Assembly instructions (long version, v1.3, Jan 12, 2014)

Author: Mark Pendrith (support@embeddedcoolness.com)

Kit overview

The RFX 328/nRF24l01+/Proto dev board is a Arduino Uno class 328p board running at 16MHz, designed as a minimalist, low-cost deployment board for wireless nRF24l01+ enabled projects. It is a 3v3 dev board with integrated nRF24L01+ module headers, and a small but usable prototyping area.

It uses all through-hole components, so nothing more than basic soldering skill and equipment is required to assemble it.

It supports the standard Arduino shield header layout, as well as a second breadboard compatible (i.e., proper 0.1" pitch grid-aligned) header layout, so you can design your own "shields" using common 50mm (or wider) proto board -- just check you've got at least 18 through holes at 0.1" pitch to span the second header layout.

For project flexibility, there is provision for two nRF24L01+ module connection headers on the dev board: One header connects to the hardware SPI pins D11-D13, while the other header connects to alternative set of pins (D4-D6) for a "bit-banging" software SPI implementation, if preferred. (For both headers, IRQ pin is set at D2, CE is D3 and CSN is D6.) (For headers connection reference, refer p.19.)

If you want to run the board as a 5V board, just use a 5V regulator in place of the 3v3, but then you need to use a nRF2401+ proto shield or equivalent with its own 3v3 regulator to connect a nRF24L01+ module. One advantage of this arrangement, however, is that you end up with *two* fairly beefy power supply rails, at both 3v3 and 5V, for projects requiring dual voltages (refer p.14 for detailed discussion).

Another small but important upgrade is the provision for the inclusion of a 10uH inductor in the AVcc power supply section. This results in significantly better analog input reading stability and precision.

To keep cost, complexity and size low, the board is without in-built USB connectivity, so it must be programmed using either an ISP programmer (e.g., a USBasp) via the ICSP header, or (assuming a serial bootloader is installed on your 328p chip), using a USB-to-TTL serial adaptor. (Refer pp.11-12, 16-18.)

Introduction

This is the long version of the assembly instructions, in step by step detail, with lots of photos, written with the less experienced builder in mind. (Indeed, a key design feature of the kit was to use only easy-to-use through-hole components, to make it accessible for those hobbyists without advanced soldering skills or special SMT soldering equipment.)

Although the detailed step by-step instructions may be skimmed by experienced builders, there is additional discussion of header configuration options (p.9) and power supply options (pp.11-15) or various deployment scenarios which should be of general interest.

Checking parts in kit

Parts list (kit with passive components):

1 RFX 328/nRF24La01+/Proto Brd PCB

1 Voltage Regulator, LD11173v3

1 crystal (16MHz)
2 caps, 22pF caps (ceramic, for crystal)
3 caps, 0.1 uF (ceramic, bypass)
1 cap, 10uF (electrolytic, filter)

1 LED

2 resistors:
R1 10K (Reset Pullup)
R2 1K (LED)

1 inductor:
L1 10uH (AVcc filter)

1 28-pin DIL socket (for Atmega328p)

nRF24L01+ connectors sockets:
2 female headers (2×4)

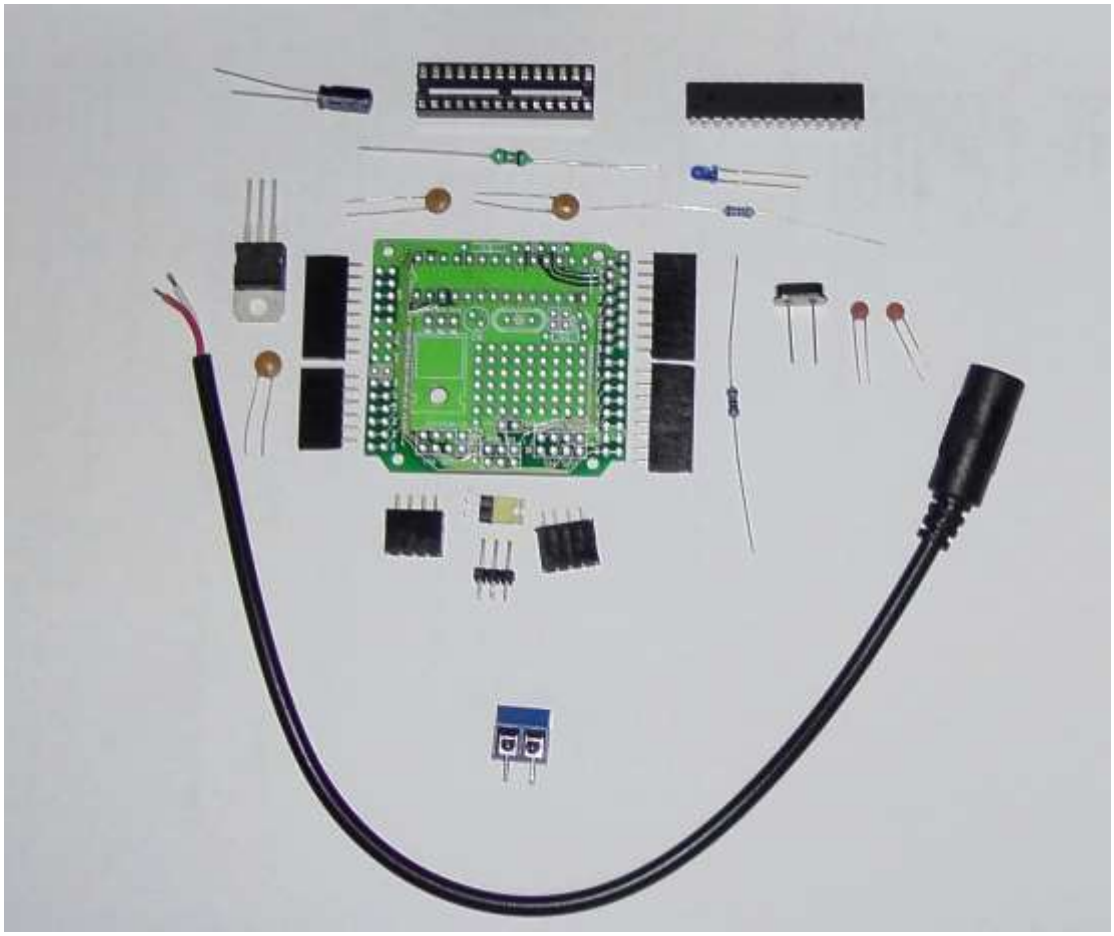
ICSP programming connector:
1 male header (2×3)

nRF24L01+ power supply jumper:
1 male header (1×2)
1 jumper

shield headers:
3 female headers (1×8)
1 female header (1×6)

Kit options:

a. if ordered optional Atmega328p, add to list
b. if ordered optional nRF24L01+ radio module (low or high power), add to list
c. if ordered optional screw terminal block and pigtail power lead, add those to list



(Note: nRF24L01+ radio module kit option not shown included above).

General assembly tips:

The tools required to assemble this kit are a temperature controlled soldering iron with small tip suitable for pcb work, and cutting pliers to trim component leads after soldering.

Rosin core solder (less than or equal to 1.0mm in thickness is recommended.)

A blob of poster putty can come in very handy holding components in place while soldering.

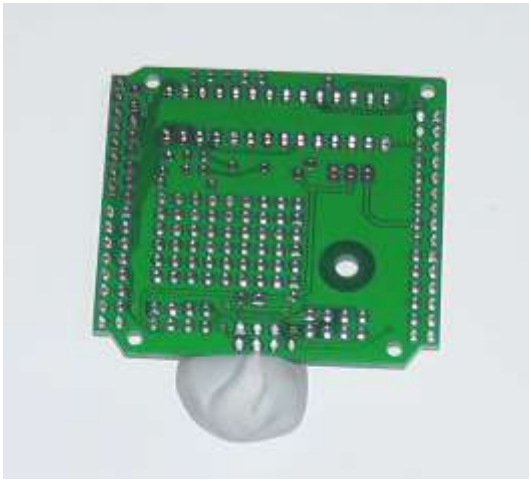
A well-lit workspace, and a magnifier of some sort is useful for inspecting the quality of the solder joints.

A basic rule with soldering is that less is more – the optimal amount of solder to use for a joint is the least amount that fully covers the joint, and no more. Working with too much solder on a joint can make things more difficult rather than easier.

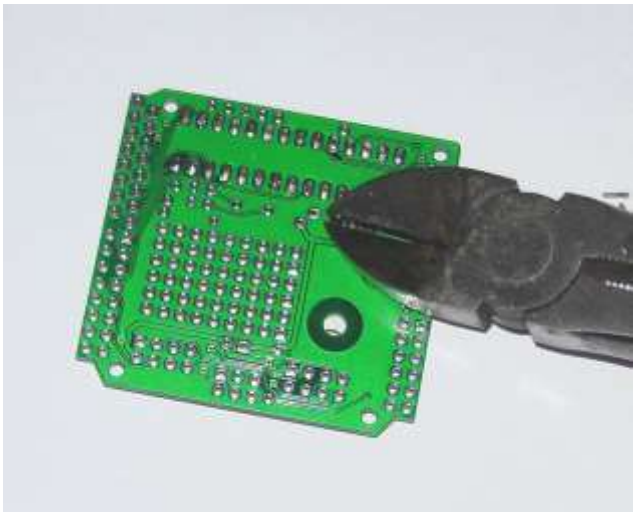
Another basic rule is to “heat the work” with the iron (briefly) before adding the solder to make the joint. Solder flows and adheres best to hot metal surfaces, not cold ones.

Assembly steps

1) Mount & solder 28pin DIP socket, orienting socket notch with silkscreen markings (note poster putty shown holding work in place so hands are free for soldering.)

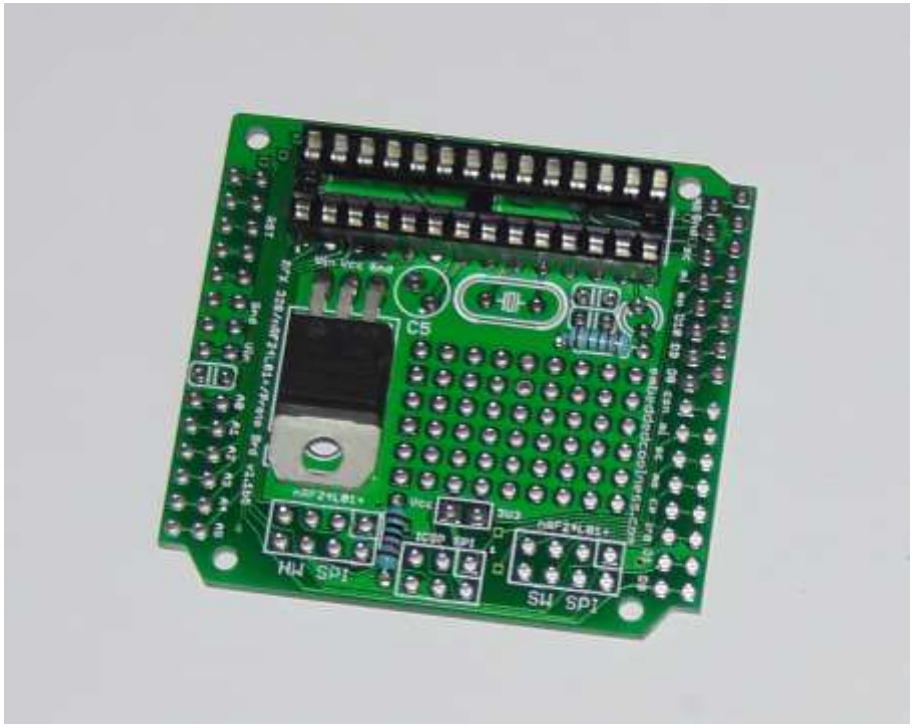


2) Mount & solder 3v3 voltage regulator. After soldering, trim excess leads on underside of board with cutting pliers or similar.

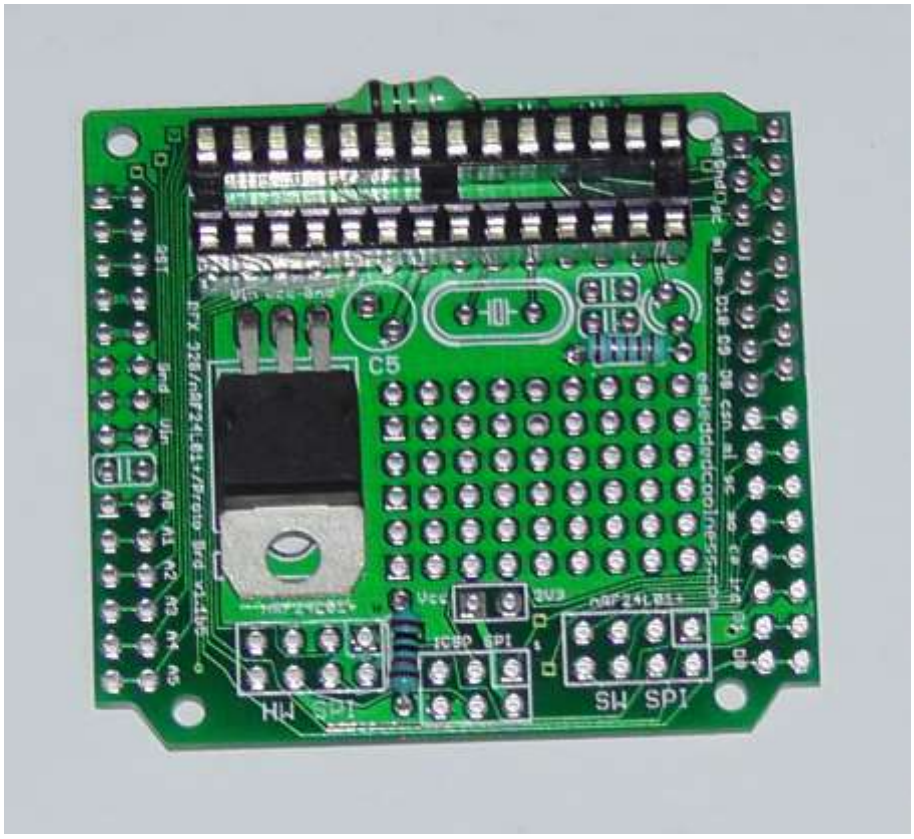


3) Mount, solder & trim resistors:

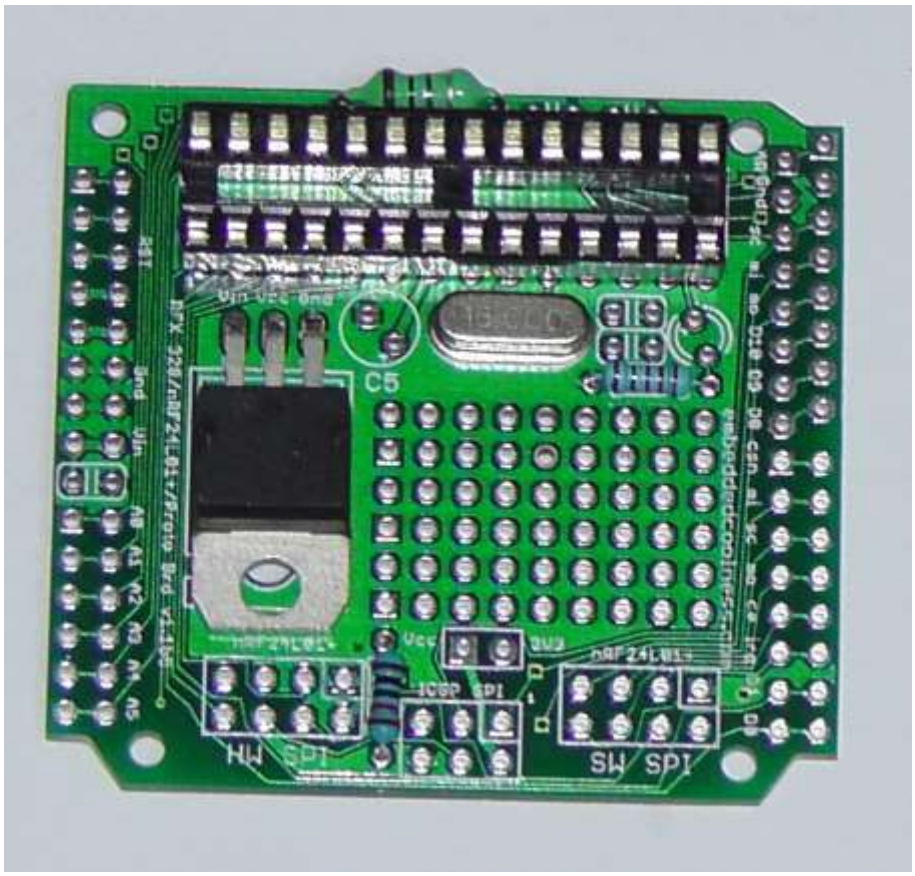
   a. R1 (10K, 1% – Br, Bl, Bl, Re, Br)
   b. R2 (1K, 1% - Br, Bl, Bl, Br, Br)



4) Mount, solder & trim inductor L1 (10mH, Br, Bl, Bl, Go)

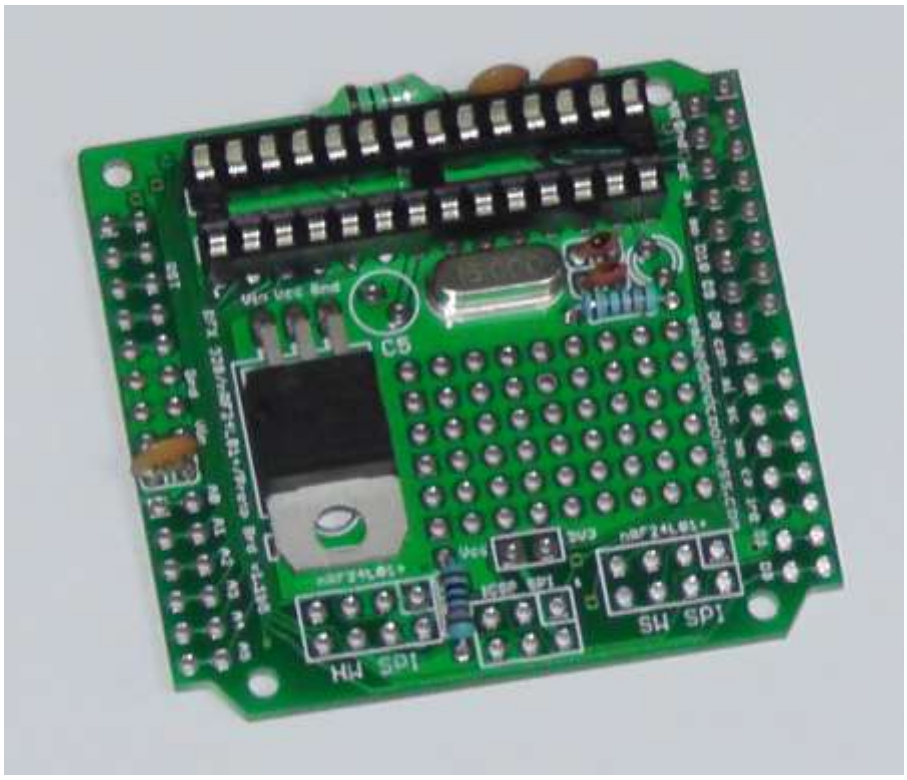


5) Mount, solder & trim 16MHz crystal

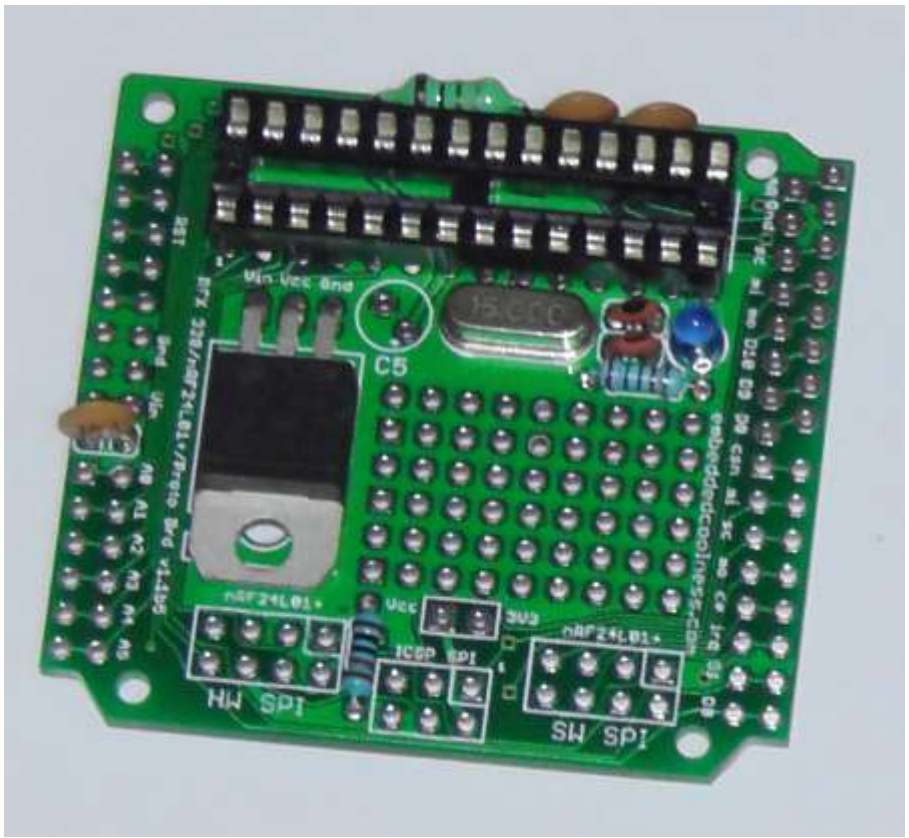


6) Mount, solder & trim 2 x crystal capacitors beside crystal (22pF, marked “22”)

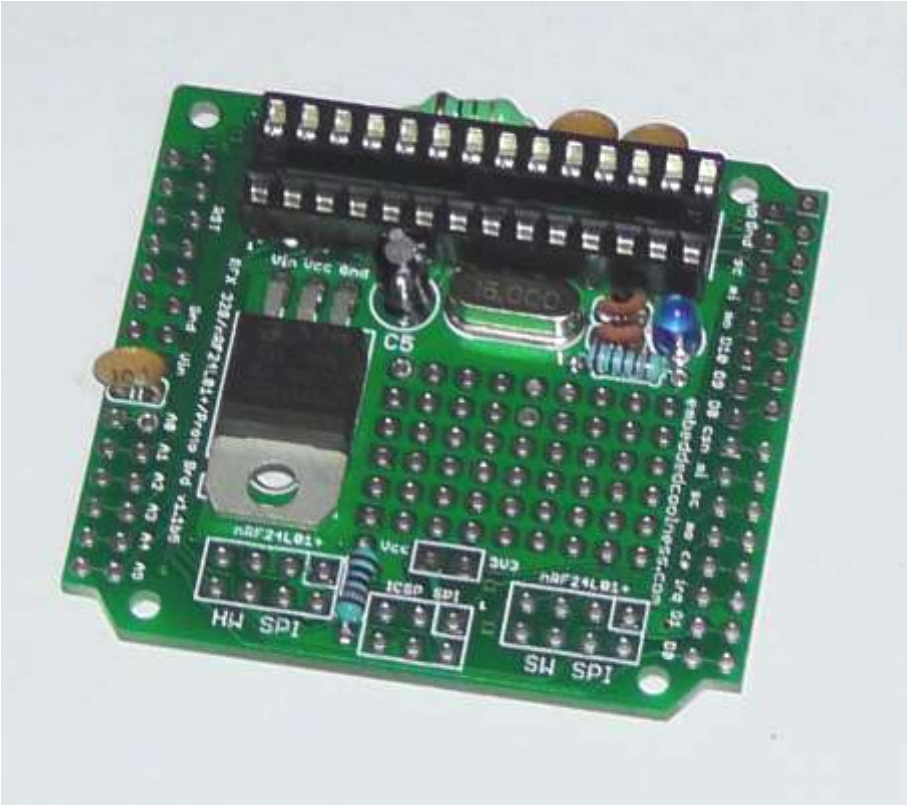
7) Mount solder & trim 3 x bypass capacitors (marked “104”, 1 between shield headers, 2 beside L1)



8) Mount, solder & trim LED, observe polarity (long lead is +, short lead is –ve)

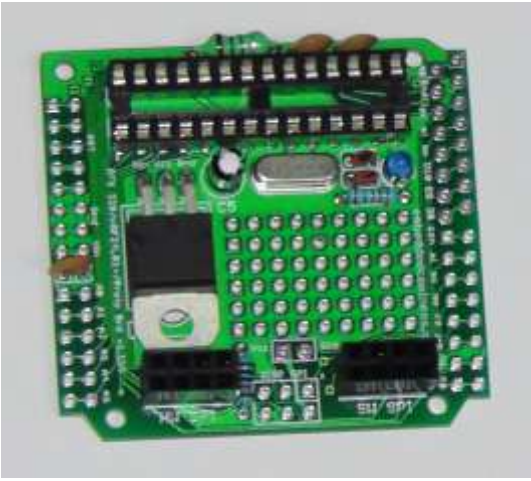


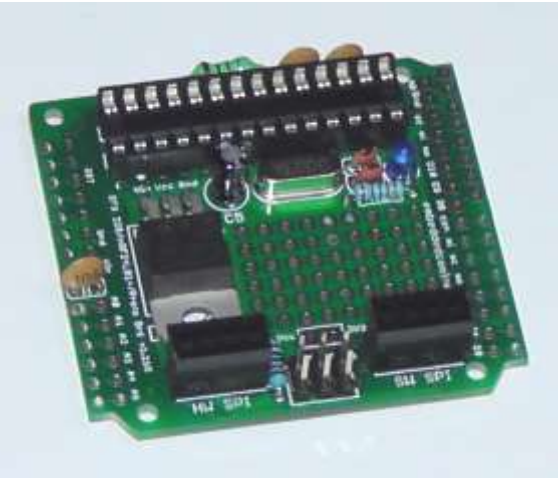
9) Mount, solder & trim electrolytic filter capacitor C5, observe polarity (10uF, long lead is +, short lead is –ve)

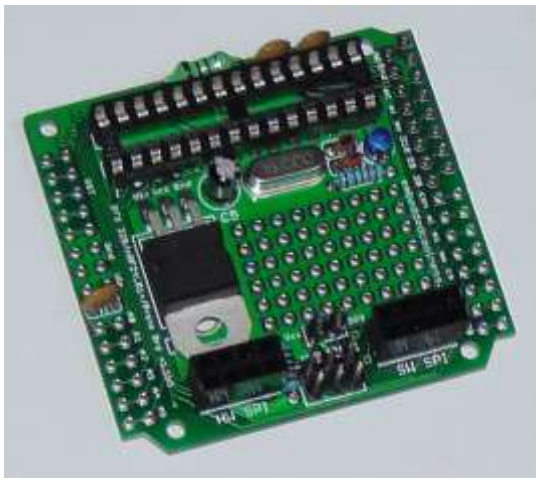


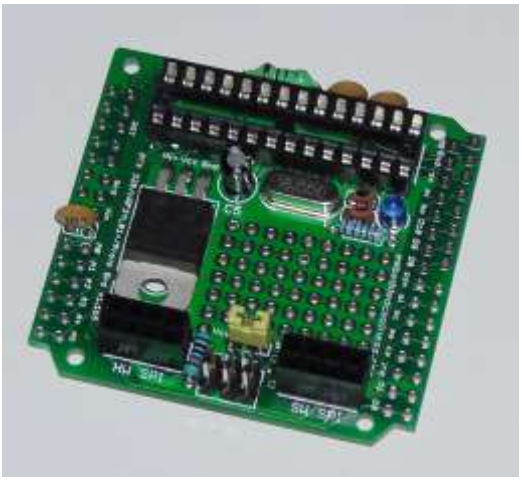
10) Mount & solder headers:

a. 2 x nRF24L01+ headers (female 8pin, 2x4)
b. ISCP header (male 6 pin, 2x3)
c. Vcc/3v3 header (male 2 pin jumper header, in front of ICSP header)









Place yellow jumper connector onto header pins once soldered.

11) (Optional step) Mount & solder shield connection headers (female 8pin x3, 6pin x1).

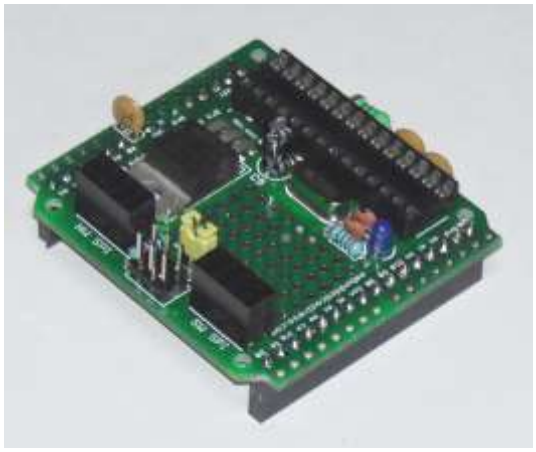
Either mount so that:

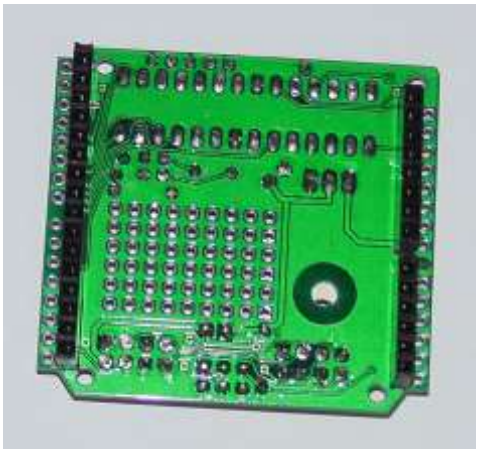
a. Shield headers are mounted on top of board, like traditional Arduino shield headers. Use rows of holes nearest edge of board to use standard shield header alignment and form factor. Or,

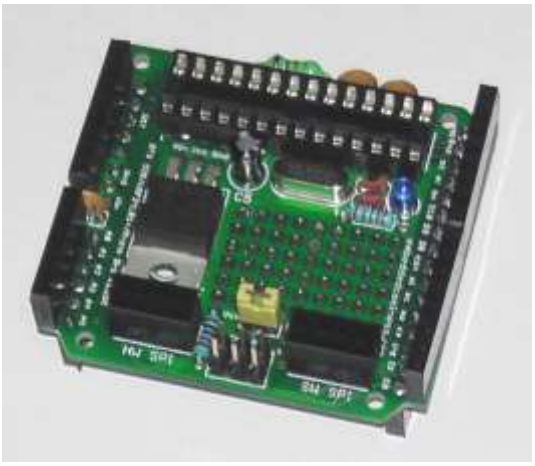
b. Shield headers are mounted on underside of board, pointing down, so that headers can be used to conveniently plug dev board onto prototyping board or breadboard with standard 01" pitch spacing (refer pp.12-13). Use secondary "inside" rows of holes from edge.

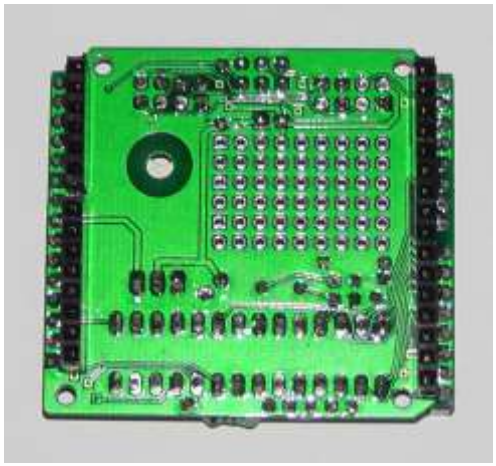
c. Also possible to have two sets of headers installed, one traditional, one 0.1" spacing aligned, if required (note: extra set of headers not included in kit).

On this demonstration build, we will opt for both standard Arduino shield header on top and prototyping/breadboard compatible header underneath. Note that if mounting both types of headers, it is easier for soldering access to mount the prototyping/breadboard compatible headers first.

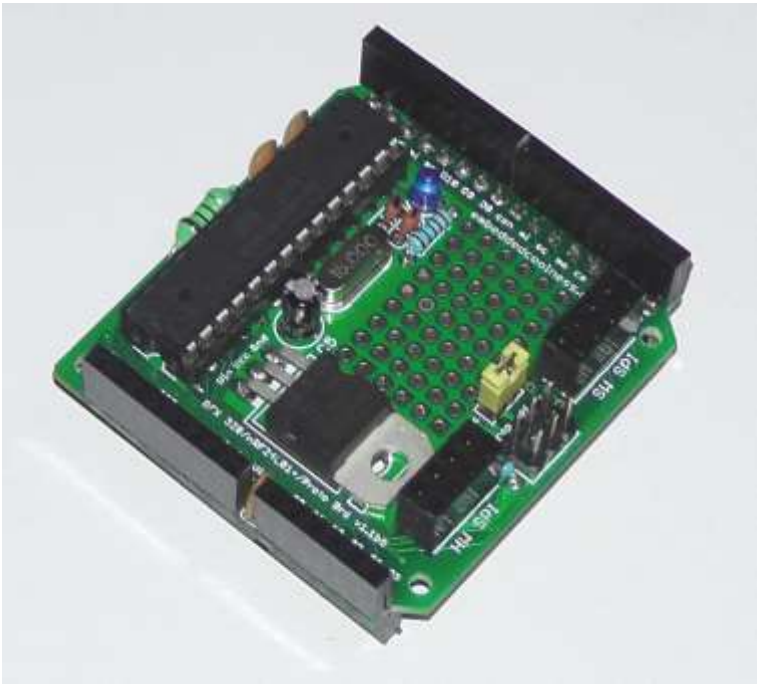








12) Plug in 328p chip, observing notch alignment with respect to DIP socket and silkscreen on board.

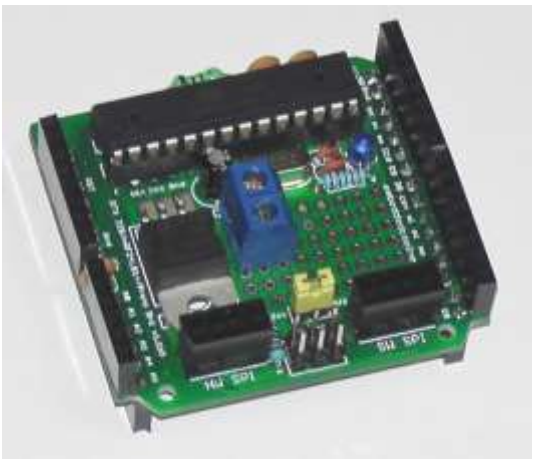


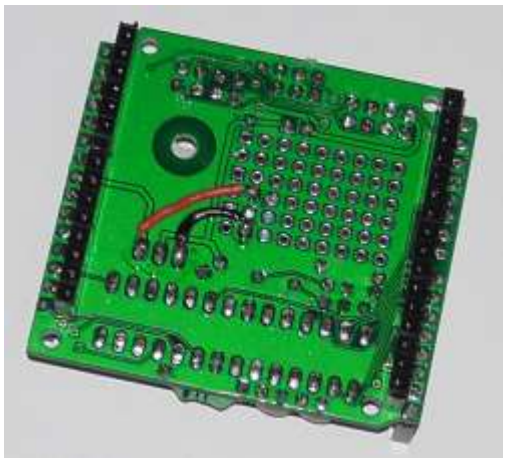
Power supply options

There are a number of possible ways to power the dev board. Here are some options:

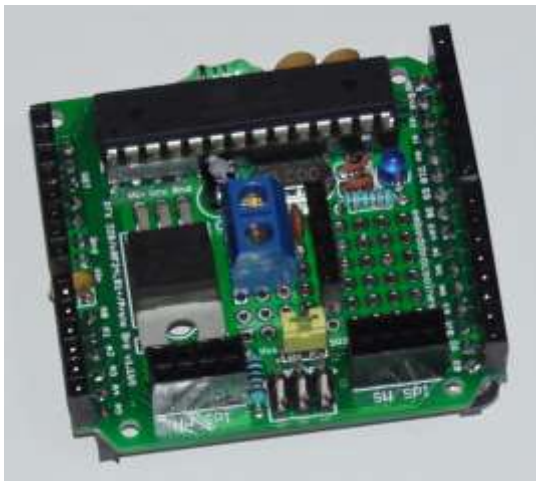
a. Power board by supplying 5V-16V DC to Vin pin on shield header (or equivalently, pin marked Vin on voltage regulator, or pin 2 on ICSP header).

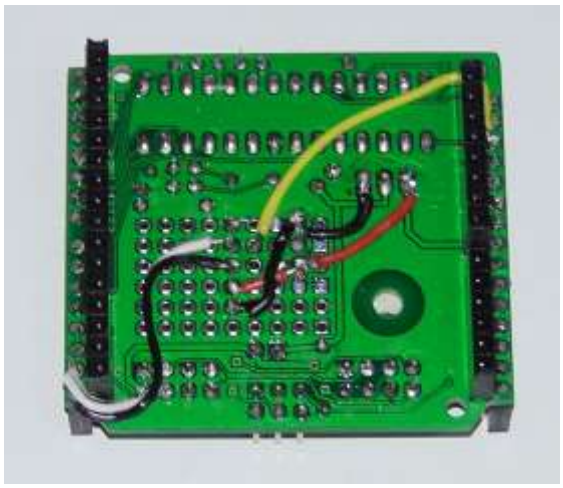
   Shown below is using the prototyping area to mount a screw terminal block (which is optionally a kit add-on, along with a 3.5mm connector pigtail power lead), connected to the voltage regulator using hook-up wire under the board:

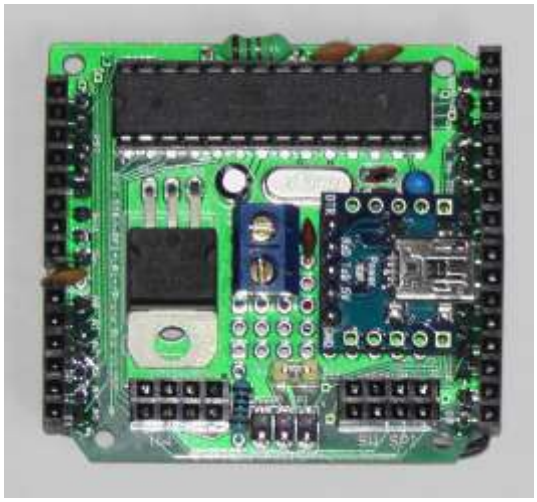




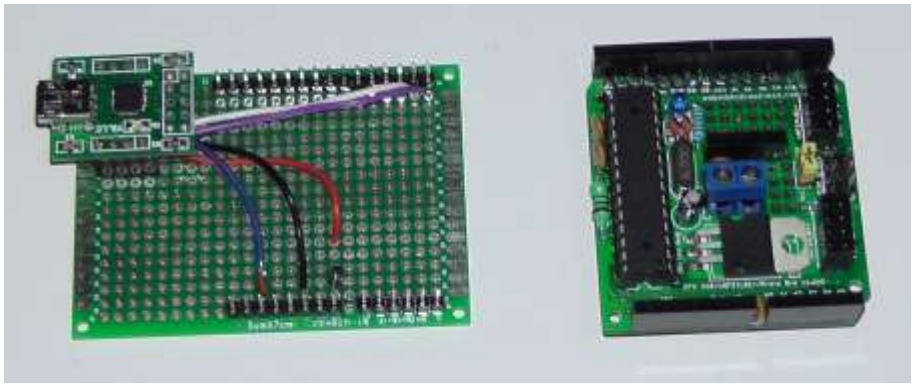
b. Another possible power configuration is supplying 5V to Vin from a USB-to-TTL adapter, if one is to be connected. The USB-to-TTL adapter could be mounted on the prototyping area of the RFX 328 dev/dep board itself, as shown below:

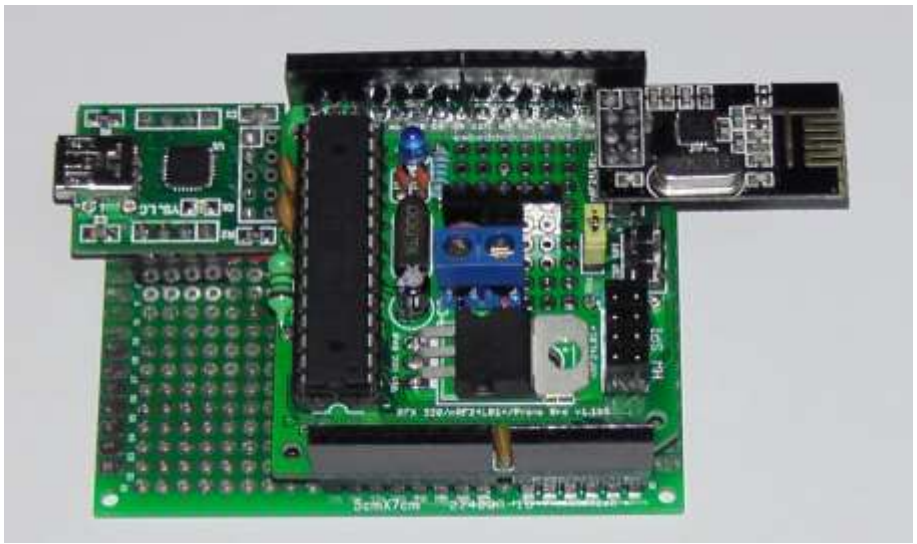




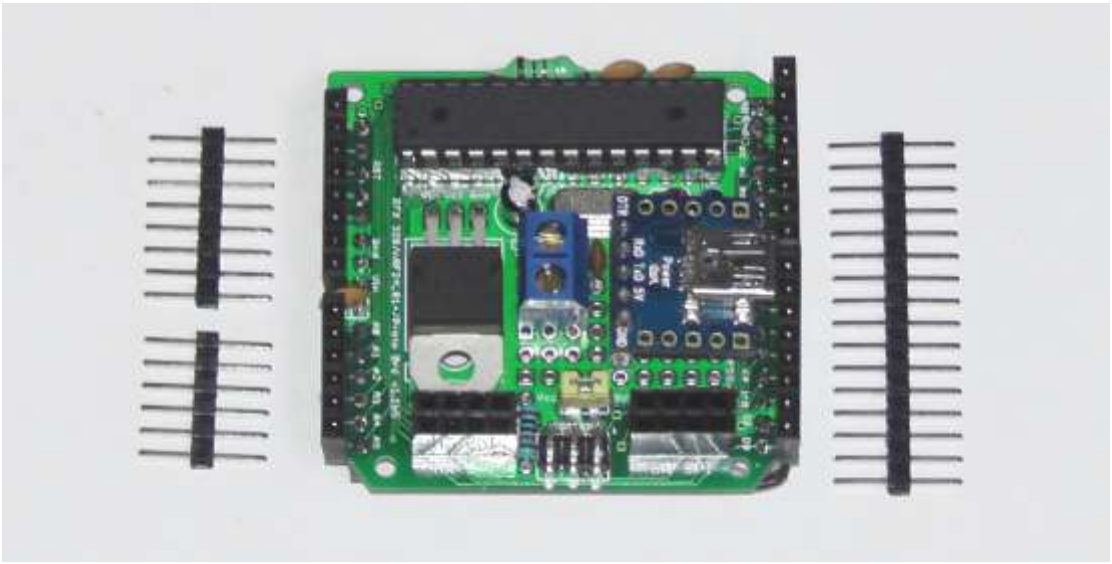


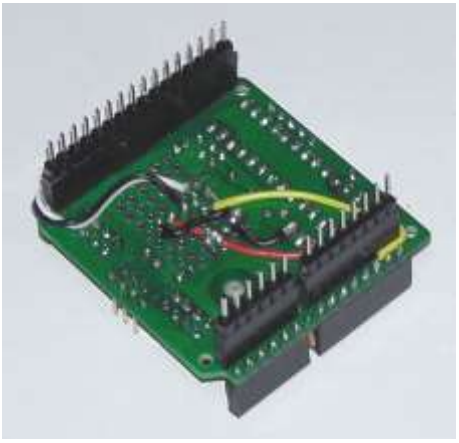
Alternatively, the USB-to-TTL adaptor could be mounted on a connected prototype board, and power to the dev board is provided via the header pins:

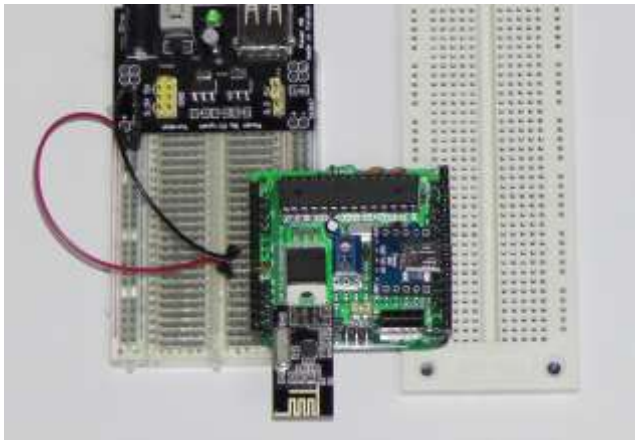




c. By using “gender changer” male-male pin headers, the RFX 328p dev board can be adapted to and powered by a breadboard set-up:







Some other power options include:

d. A battery pack of e.g. four AA or AAA cells (either 1.5V alkaline cells, or 1.2V rechargeables) could also be used as input to the voltage regulator (Vin). For reliable results, aim for a Vin supply voltage of at least 4.8V. (Although the LD1117V33 voltage regulator can theoretically work with a Vin voltage as low as 4.3V, in practice getting too close to the drop out voltage limit can lead to instabilities in the power supply regulation.)

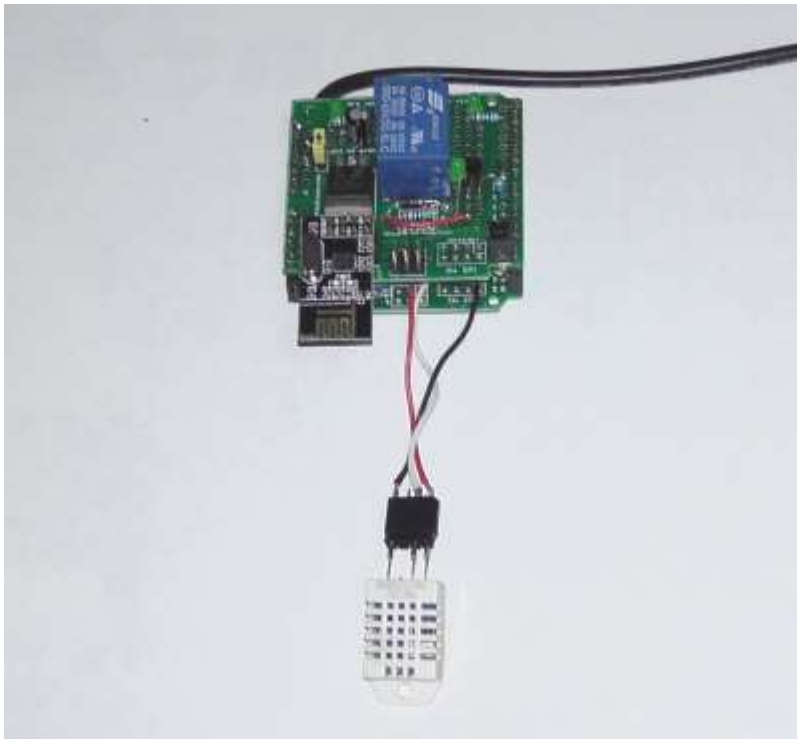
e. It is also possible to supply board power with already regulated 3v3 DC directly to the Vcc rail (e.g., at the middle pin marked Vcc on voltage regulator.) Note that in this case the regulator is not actually used, and could be omitted from the build, if desired.

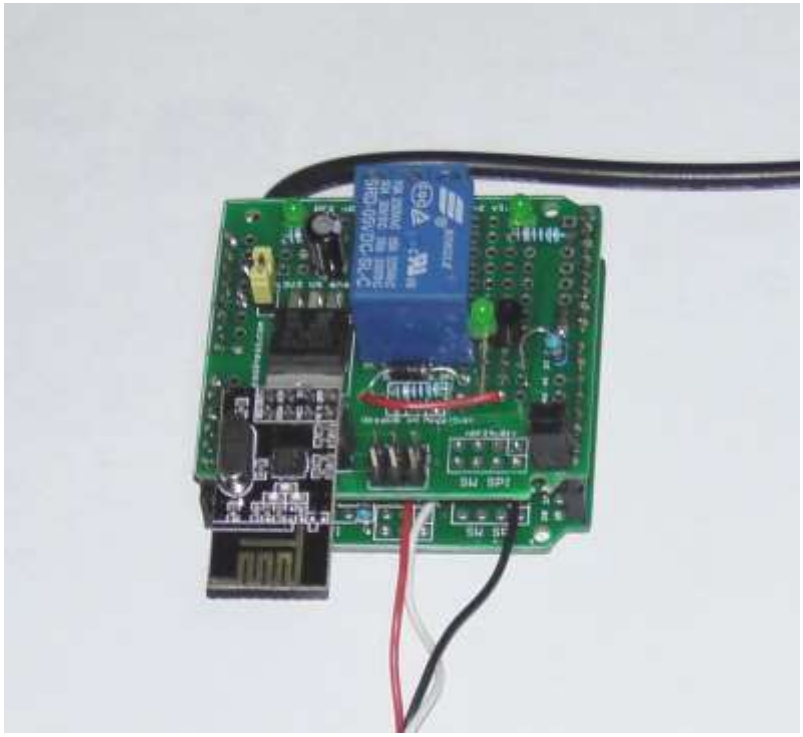
f. If a nRF24L01+ module is not to be connected to the dev board, then the Vcc rail can optionally be supplied with a regulated DC voltage > 3v3, up to 5.5V. A 5V regulator can be directly substituted for the 3v3 regulator if so desired (in which case supply Vin with at least 6V DC, to allow for ~1V regulator drop out voltage.)

   One reason a 5V regulator on the dev board might be used is if a dual rail 5V/3v3 set-up was required. The next section describes such an arrangement.

5v/3v3 dual rail power option using RFX nRF24L01+ proto shield

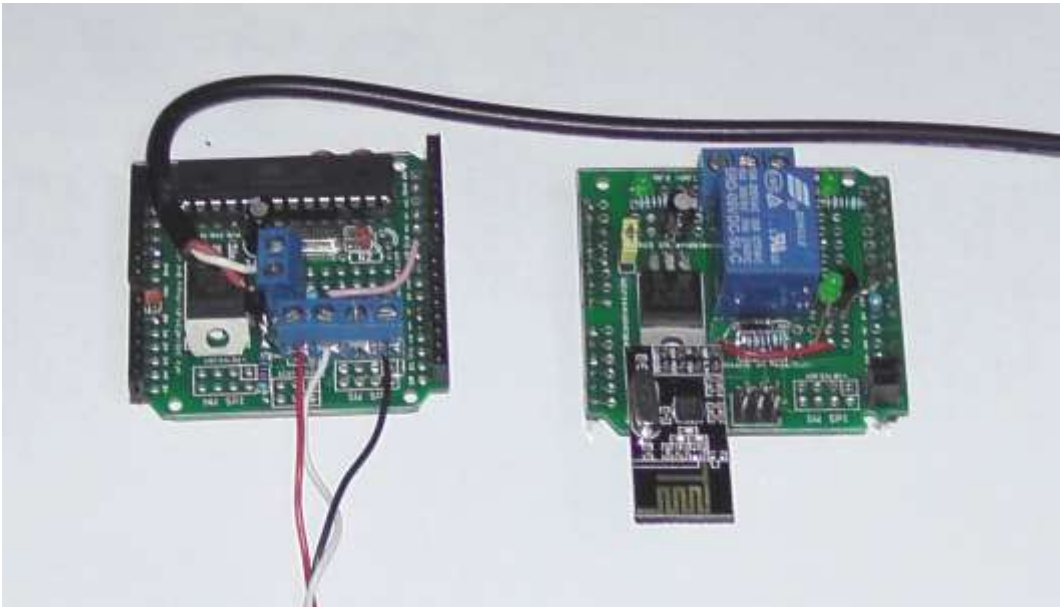
An interesting and useful power supply configuration can be set-up for projects requiring dual power supply voltages at both 5V and 3.3V.





Above is a photo of a 328 dev board set up with a LD1117V50 voltage regulator powering its 5V rail, coupled with a RFX nRF24L01+ proto shield providing a 3v3 rail powered by a LD1117V33 regulator (RFX nRF24L01+ proto shield available as a separate kit).

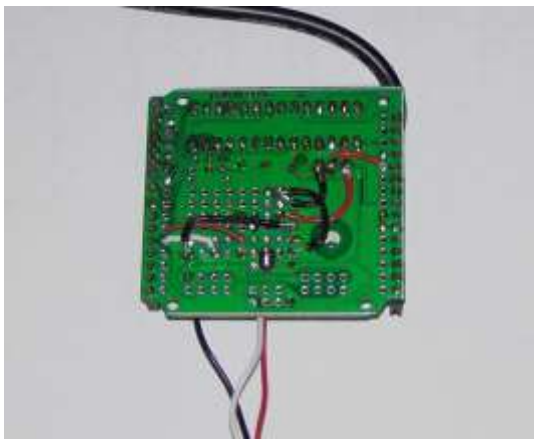
The dev board is shown with the shield detached below:

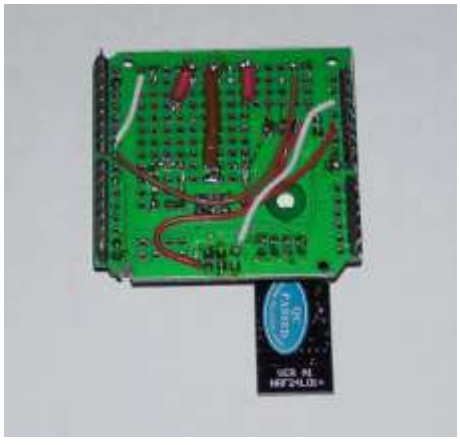


The dev board is set up as a nRF24L01+ enabled controller for an air exchange fan that controls humidity in an under-house subspace. The controller compares the readings from two DHT22 temperature-humidity sensors (one shown, the other mounted outdoors) that run on a 5V supply. It also operates a 5V relay (shown mounted on the prototyping area of the shield) to control the switching of the exhaust fan.

Since the nRF24L10+ module requires a 3v3 supply, supplying dual voltage rails was required for this mix of components. The LD1117V50 used for the dev board and the LD1117V33 used for the shield are both rated at 800mA, thus providing a generous capacity dual rail configuration for many projects requirements -- far better in fact than the relatively feeble dual rail setup (50mA max 3v3 supply rating) on the standard 8-bit Arduino boards or clones.

The prototyping area wiring underneath is shown for both the dev board and shield:





A connection from the 5V regulator output to the 5V header pin on the dev board (short red wire in photo on left) provides the shield with access to the 5V rail. By default, both the 5V and 3v3 header pins on the 328 dev board and nRF24L01+ shield are unconnected, so as to provide just this sort of flexibility on regulator set-up. (This is worth noting in case you were expecting to see voltage on those header pins without explicitly providing the connections.)

Programming options

Preliminaries

If using a new "factory fresh" 328p chip for the first time, it will need to have its fuses set from the default factory settings in order to be programmed at full speed using an AVR ISP compatible programmer, and further must have an Arduino bootloader installed if it is to be programming via virtual com port serial link option.

The fuses can be set, and the bootloader installed, in one operation from the Arduino IDE via the command "Tools|Burn Bootloader".

Before the "Burn Bootloader" operation is executed, the "Tools|Board Selection" must be set to "Arduino Uno", and the "Tools|Programmer" selection must correspond to the model of ISP programmer to be used (e.g., USBasp). Also, the "slow clock" option must be selected on the ISP programmer when programming a new chip; the programmer clock speed must be set < 250KHz. (Don't forget to set the programmer clock speed back to full speed once the fuses have been set – this will allow for *much* faster programming!)

If you don't have access to an ISP programmer, at a pinch you can use another Arduino board running the "Arduino as ISP" sketch to act as one. Basic instructions of how to do this can be found at the arduino.cc website (http://arduino.cc/en/Tutorial/ArduinoISP). Googling may also be worthwhile to get additional tips and tricks to get this working.

However, it is highly recommended to invest in a dedicated ISP programmer with a 6 pin socket header compatible with the ICSP header. They don't have to be expensive; variants of the USBasp and USBtinyISP can be found on eBay, for example, in the $5-15 price range, and the simplicity, reliability and convenience they offer over the "Arduino as ISP" set-up makes the modest investment well worthwhile.

Once you have the fuses set and the bootloader installed, you then have the choice of:

a) programming via the ICSP header using an AVR ISP compatible programmer, or

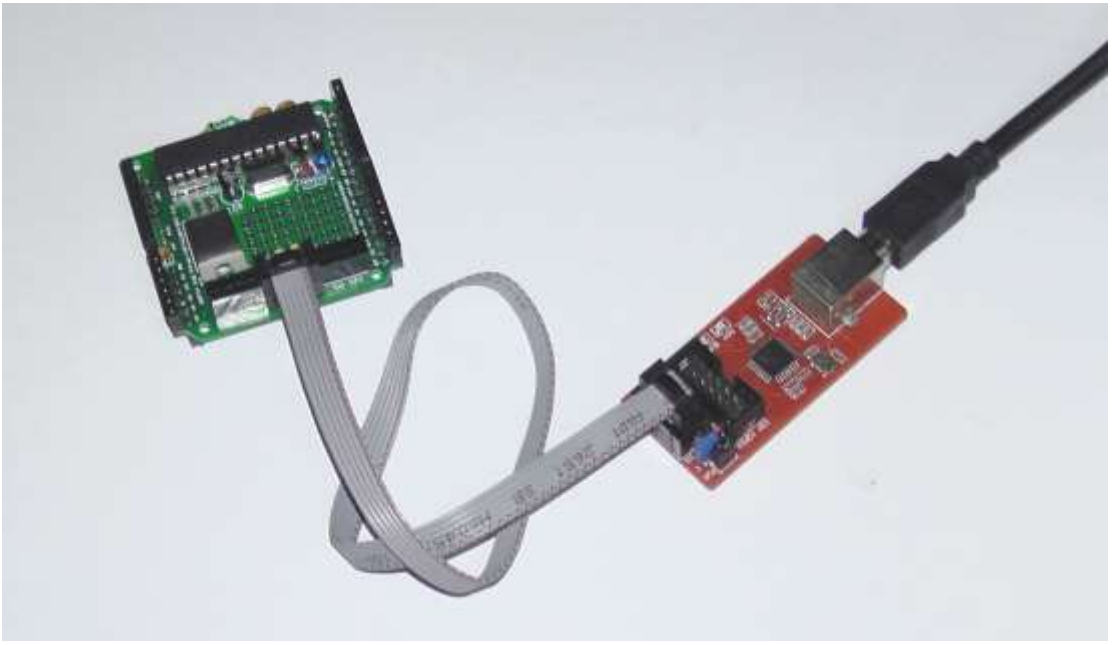
b) programming via the bootloader using a USB-to-TTL converter.

Note that programming a sketch using the ISP programmer via the ICSP header will overwrite the bootloader flash area (but the fuse settings will not be disturbed). This is not a problem if programming by the ISP programmer will be your standard method of programming, but keep in mind the "burn bootloader" step will have to be repeated if you want to program the 328p via the bootloader at some later stage.

a) Programming dev board via ICSP header

Program as Arduino Uno (Tools|Board Selection) using AVR ISP compatible programmer (USBasp, etc.) from Arduino IDE. **Make sure the jumper for pins Vcc/3v3 header is removed if a nRF24L01+ module is plugged in, or, even better, make sure the nRF24L01+ modules is removed before connecting programmer to ICSP header.** The danger here is that the programmer (e.g., USBasp) could supply 5V to Vcc rail, which is fine for 328p chip, but could damage nRF24L01+ modules if they were connected.

The photo below shows a ISP programmer (USBasp) connected to the dev board with the nRF24L01+ module removed.



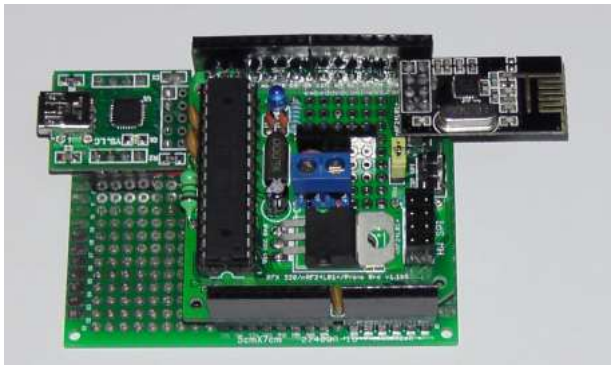
After programming, disconnect programmer from ICSP headers, replace Vcc/3v3 jumper, and plug in nRF24L01+ module (if nRF24L01+ module is used for this project.)

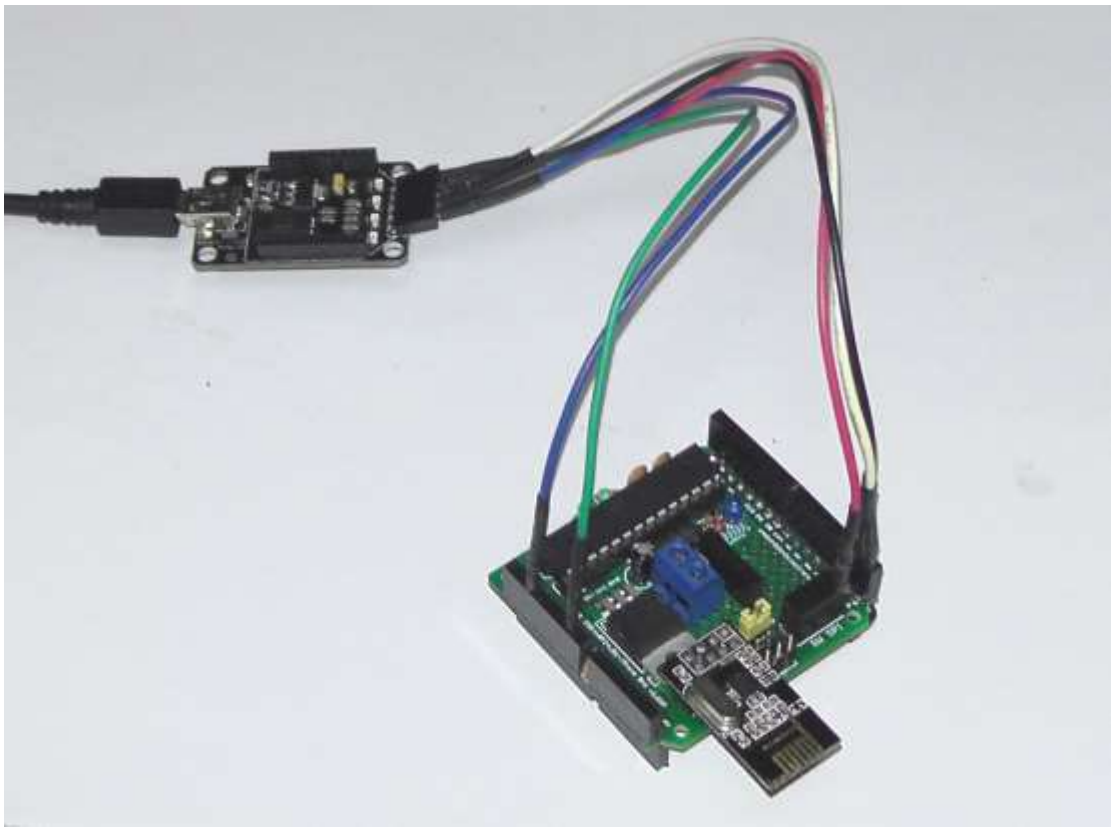
b) Programming via bootloader and USB-to-TTL converter

While programming via an ISP programmer is conceptually the simplest and most direct method, the more familiar approach (at least to Arduino users) of programming using the serial IO pins via an installed Arduino bootloader program is also an option, if you have a USB-to-TTL converter module.

The advantage of programming this way is that immediately after you upload your program, you can run the sketch and communicate with it via the USB serial port connection within the IDE, in the style of a regular Arduino – power, programming and serial communications are all catered for via one USB connection. Arguably, it is this single convenience more than any other that has made the Arduino platform so popular with newcomers to microcontroller programming.

A USB-to-TTL module is not expensive, and can be useful for powering and communicating with a dev board via serial IO during sketch development, even if not used for programming or ultimately needed for deployment of the completed project. Below we show three different USB-to-TTL modules, the first two are designed to be mounted on a PCB or protoboard via headers, while the third is the external style.





Note the hook-up using the external USB-to-TTL adapter module (3rd photo): The power supplied from the module is 3v3, so instead of connecting to Vin, it was connected directly to the 3v3 Vcc rail via pin 2 of one of the the nRF24L01+ header sockets (red wire). The power header jumper (yellow in the photo) must be in place for this to work. (If instead the module supplied 5V as output, it would be correct to connect to Vin. Note, however, connecting to the Vcc rail with 5V would also work -- as long as the nRF24L01+ module was disconnected!)

The other connection worth noting is the module DTR output to the reset pin on the shield header (blue wire). This direct connection arrangement happens to work for this particular module, however it is usually specified that a 0.1uF ceramic capacitor is connected in series to provide a better “pulse” shape to the reset signal connection. (The prototyping area could be convenient to accommodate this arrangement if necessary.)

The remaining connections are simply ground (green wire), and serial Tx/Rx (black and white wires) to D0 and D1.

Header connections reference

For project flexibility, there is provision for two nRF24L01+ module connection headers on the dev board: One header connects to the hardware SPI pins D11-D13, while the other header connects to alternative set of pins (D4-D6) for a “bit-banging” software SPI implementation, if preferred. For both headers, IRQ pin is set at D2, CE is D3 and CSN is D7.

To support the choice of running a board at either 3v3 or 5V, the 5V and 3v3 pins on the power supply header are both unconnected. Simply connect the appropriate pin to Vcc via hook-up wire as required for supply power to as shield, etc.

HW SPI header:

MOSI D11
MISO D12
SCK D13

(HW SPI header also connects to ICSP SPI pins.)

SW SPI header:

MOSI D4
SCK D5
MISO D6

Both HW SPI and SW SPI headers:

IRQ D2
CE D3
CSN D7

ICSP header:

Vcc connects to dev board Vcc (i.e., output pin 2 of LD1117 voltage regulator)
Gnd to dev board ground plane
Reset to 328p reset pin, power header reset, 10K pull-up resistor
MOSI, MISO, SCK to HW SPI header, and to D11-D13

Power header:

Vin (pin 0) connects to input of LD1117 voltage regulator (decoupled 10uF cap.)
Gnd (pins1,2) to dev board ground plane
5V and 3v3 pins (pins3,4) unconnected
Reset (pin 5) to 328p reset pin, ICSP header reset, 10K pull-up resistor
Pins 6,7 unconnected

Analog/Digital IO headers:

A0-A5 connected as Arduino Uno
D0-D15 connected as Arduino Uno

Board schematic: