Phil's homebrew NRF24L01+ FHSS R/C set - lockdown project, May 2020. Part 1 - The transmitter:

This is a simple expandable four channel RC outfit based on the NRF24L01+ using a true Frequency Hopping Spread Spectrum system. Its development was documented in real time on the mode-zero forum, and runs to over 200 posts, hence the need for a more concise “how-to” document.

The system uses a unique 'protocol' which sends one packet every 5ms, each packet containing all the control positions, so 200 updates per second. It uses sixteen RF channels well spread through the band, and has full failsafe on all channels. The system works very well, has logged hundreds of hours with dozens of sets in use.

Features:
Mixers for V-tail, Elevons, and Single-channel compound escapement emulation with two S/C keying speeds
A minute-minder, failsafe set from the tx or rx, soft and hard throttle lock
Expo, rates, +-400 range + trim, dual (+/-) stick mapping, optional non-linear throttle

Please note that the calibration procedure must be followed exactly.

Connections:
Stick pots are wired between ground and regulated 5v from Arduino using servo leads. Wipers connected as follows:
A0=aileron, A1=elevator, A2=rudder, A3=throttle, A6=expo, A7=rates.
Always use this wiring order regardless of channel order.

Ground A6 and A7 with links or Spektrum bind-plugs if rates & expo are not required. Dont leave them floating.

Unwanted digital functions can be omitted except for S/C button on D8 which is needed for calibrating the sticks, this can be done with a bind-plug link if no button is fitted.

The board used is the DIY More ProMini Strong, this is a typical 'Arduino style' board and substitutions are fine.

Pin allocations:
D0=spare, D1=spare, D2=buzzer, D3=increment minute timer, D4=set failsafe from tx button,
D5=50:50 mixer, D6=75:25 mixer, D5&D6 together=60:40 mixer
D7=throttle lock, D8=compound s/c button (and stick calibration)
D9 & D10=NRF24, D11, D12 & D13=NRF24 SPI.

Operation - Rates, expo, mixers & reversing:
Variable expo & rates on Futaba ch1, 2 & 4, ditto JR ch2, 3 & 4.
Elevon mixer on Futaba ch1 & ch2, JR ch2 & 3, 75% aileron, 25% elevator or 50:50 for V-tails
Mixer option can only be changed on power-up (avoids accidentally switching during flight).
Single-channel emulation mix is compound only, though sequential would be easy to add.
Servo reversing by holding sticks over on power up and is saved to eeprom.
Soft throttle lock holds the throttle shut until the throttle stick is pulled back, whilst this hold is active a trill will sound.
A manual throttle-lock is also provided.

Follow the story on the single-channel forum http://mode-zero.uk/viewtopic.php?f=27&t=844
Diagram and documentation on http://www.singlechannel.co.uk see Archive page

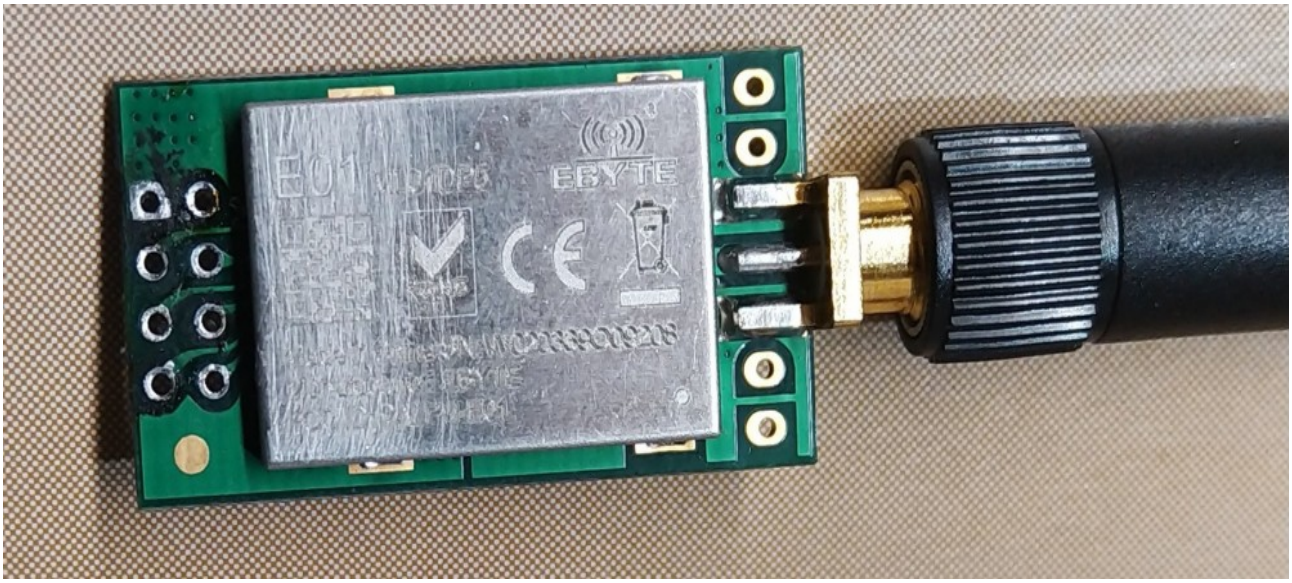
Choice of donor Transmitter:
For a first project, a transmitter having mechanical trims is much easier. These are the type where the trim control mechanically rotates the same pot as the stick. Separate trim pots can be accommodated but its slightly more involved and best avoided for a first project. The old Skyleader Clubman, Digifleet, Horizons, Waltrons, etc are typical examples, as is anything with Kraft-Hayes stick units (Staveley Deluxe, some Heathkits, and of course Kraft sets).

Components:

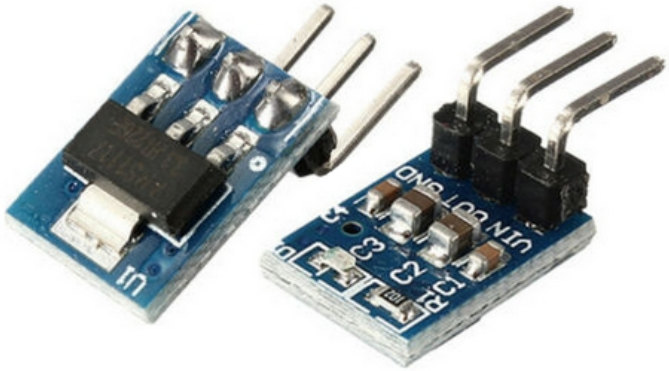
The RF is a generic NRF24L01+ board widely available at very low cost. Choose the screened version:

The pins should be carefully removed, I hold the board flat in a vice, then clamp a pair of fine-nosed pliers to the pin. When desoldering, the weight of the pliers will gently pull the pin out:

The 'de-pinned' NRF should look like this with nice clean holes.
If not, you might struggle to get the connection wires through:

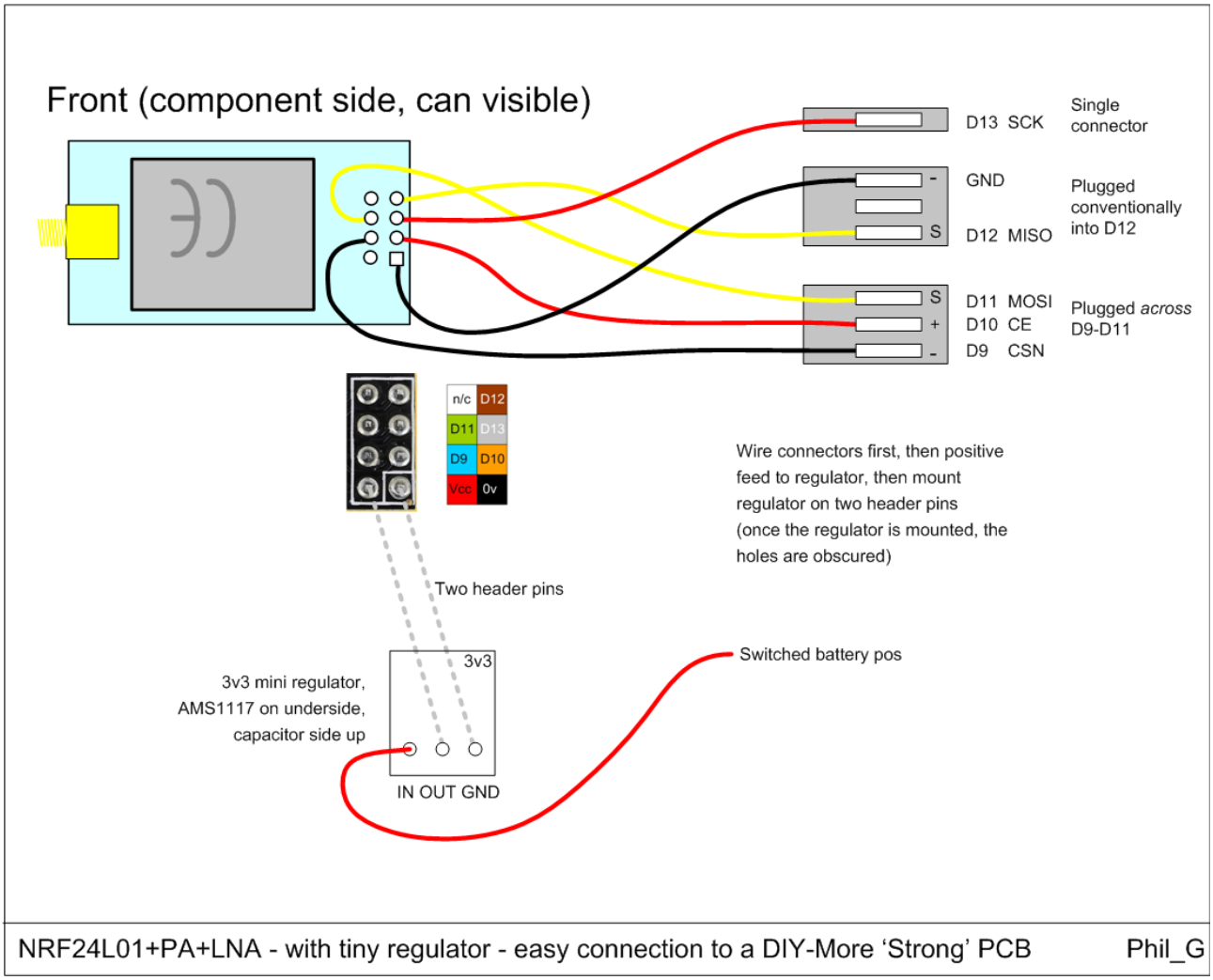


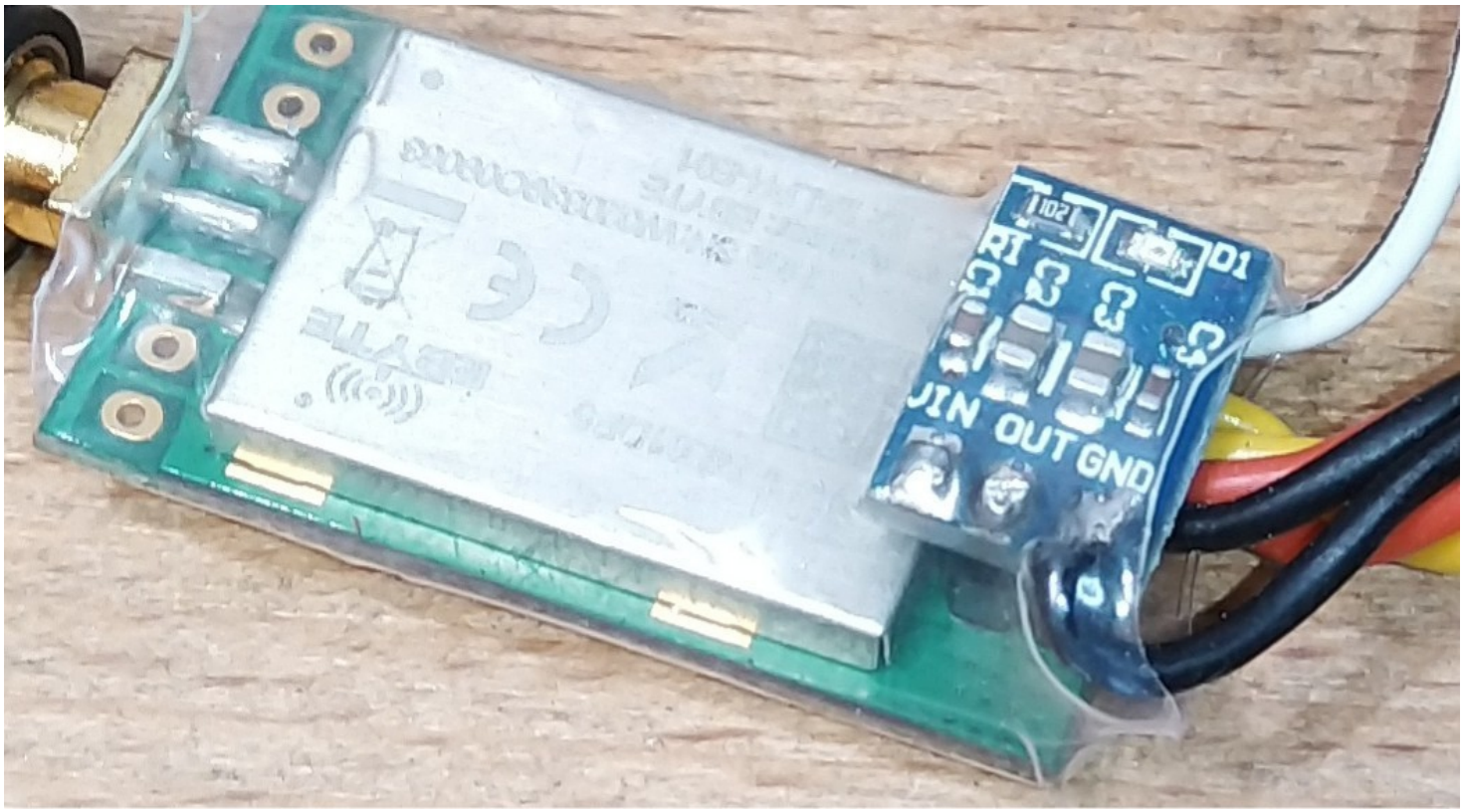
The 3.3 volt regulator is a cheap ebay item. As with the NRF, desolder and remove the pins:



Note that it mounts chipside-down, with the capacitor side visible. It is physically supported by two header-pins through the GND and OUT holes. The IN hole has the positive feed from the encoder. Mount the regulator after all the NRF wires except the negative are soldered up, otherwise they will be obscured under the regulator.

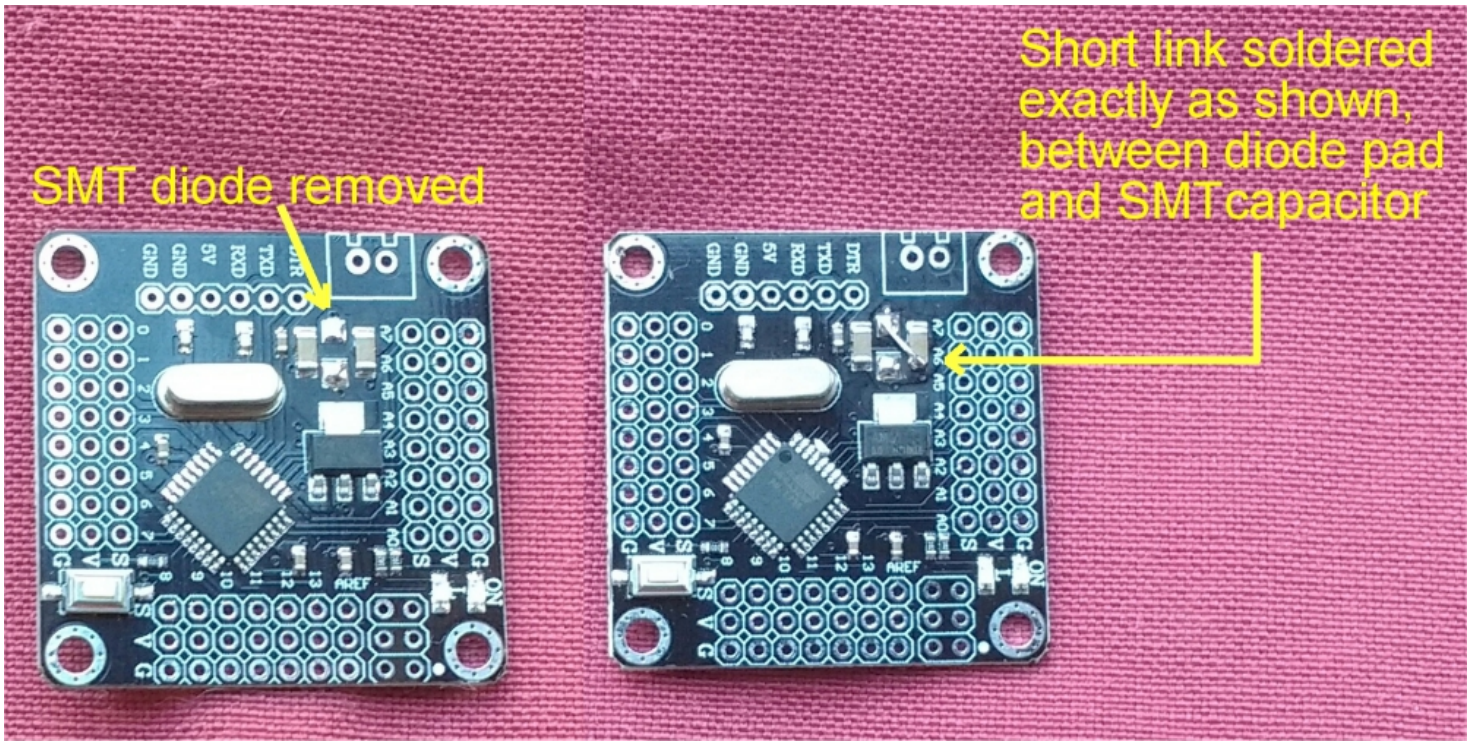
Take two thin servo leads, and wire these connections. Leave mounting the 3.3 volt regulator until last as it obscures the connection holes. T* Note * that the neg from the D12 plug should be soldered last, after mounting the regulator. Its soldered to the stub of the regulator mounting pin, as can be seen in the lower photo:





The encoder board
These notes assume a DIY-More 'Pro-mini Strong' which is widely available on ebay , Aliexpress, Banggood etc
It must be programmed via ICSP using a USBASP programmer. Please do not use the bootloader and serial load.
The software link is attached to the first post of the mode-zero forum thread :
http://mode-zero.uk/viewtopic.php?f=27&t=844
If you modify this software in whole or in part please acknowledge its source - Phil_G on most forums, philg@talk21.com
http://www.singlechannel.co.uk

DIY-More 'Strong' board preparation:
Remove the power input schottky and add the 45° wire strap from the incoming positive pad to the adjacent capacitor:

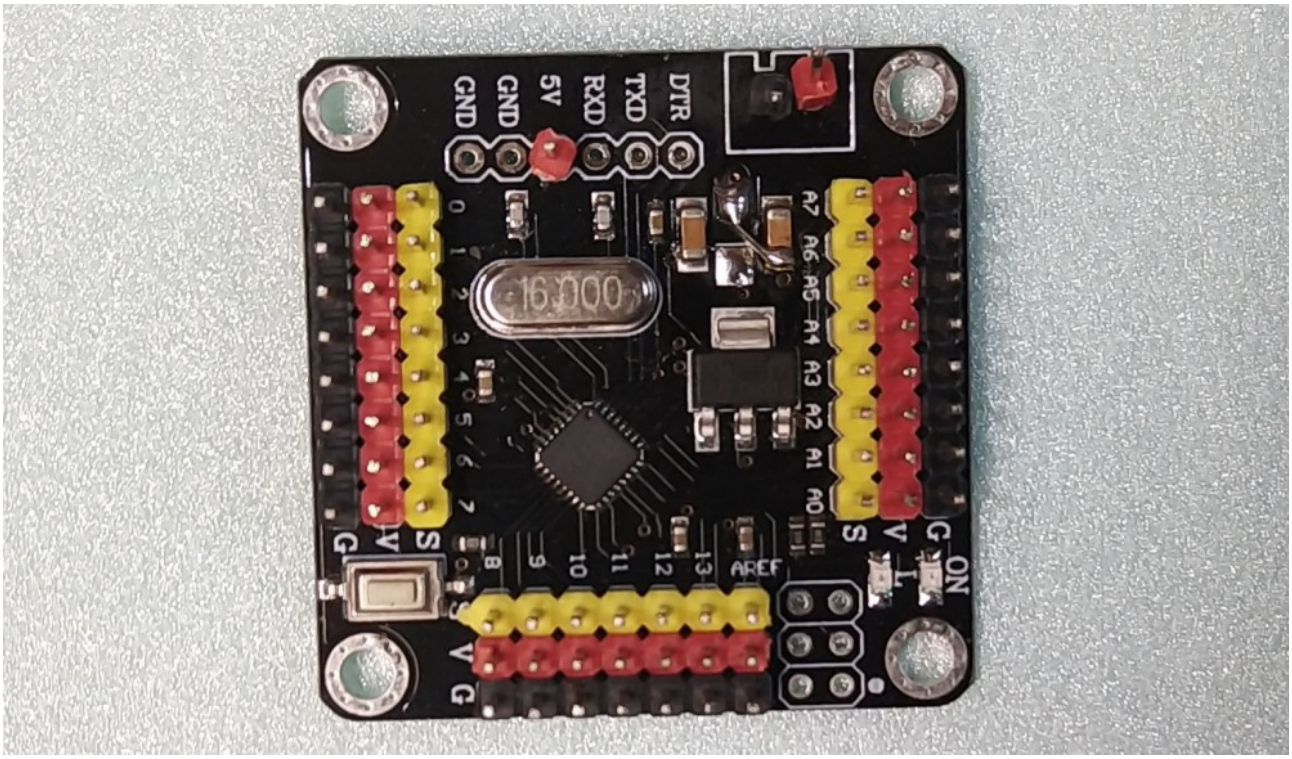


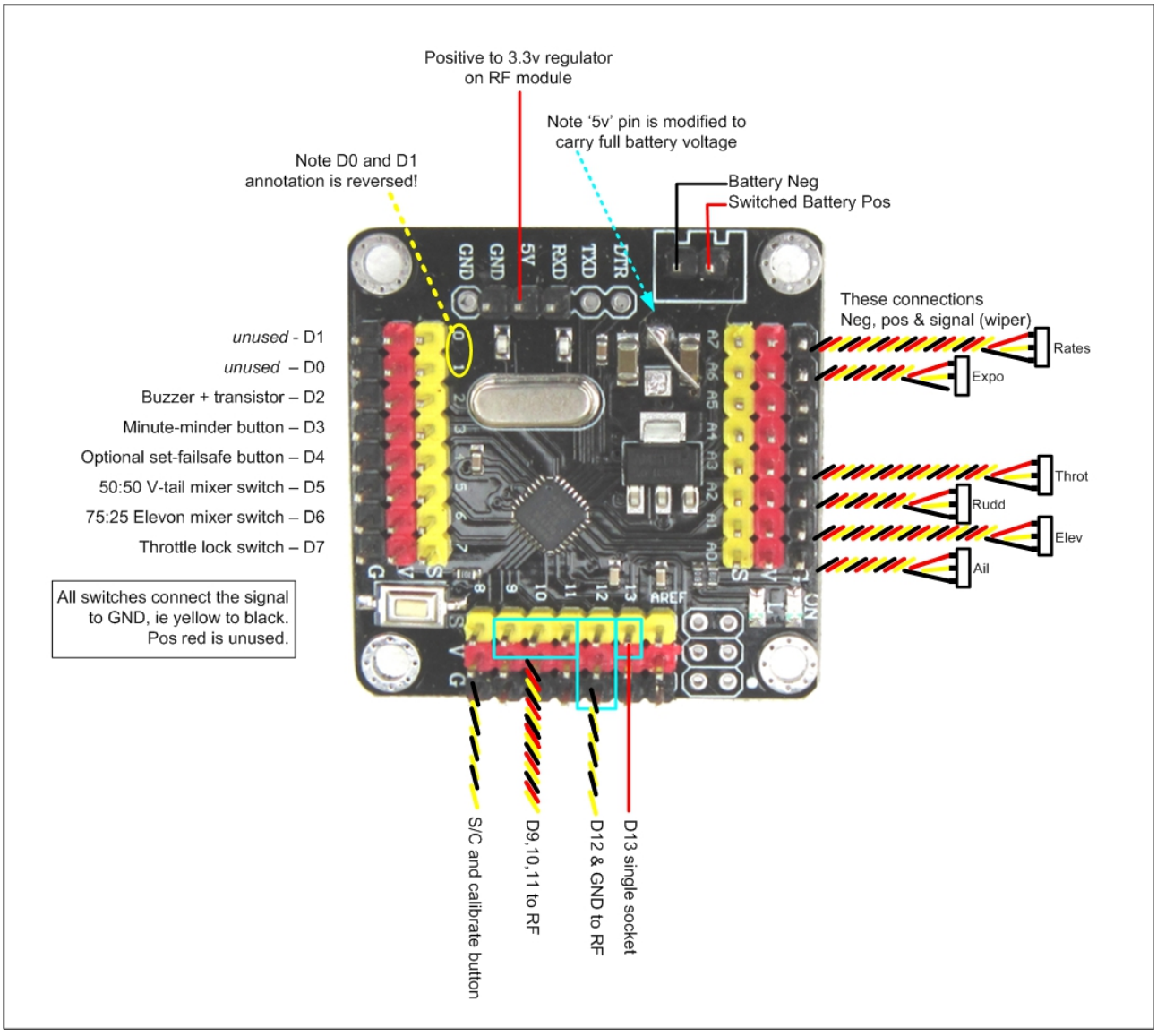
Add all the header pins, or those that remain after inserting an optional 4-way DIP-Switch into D4-D7
Note that black pins are outermost, reds in the middle and the yellow signal pins innermost.
Add a single red pin to the 5v hole seen in the middle of the top row in the photos above.
The schottky mod supplies full battery voltage to this pin which is used to power the NRF module.

Buzzer:
The external buzzer is used by the minute-minder, the throttle-warning, and the inactivity alarm.
The Atmega328P processor is capable of driving a DC buzzer directly however I much prefer to buffer it with a 2N3914 transistor as this avoids electrical noise getting back into the processor port:
The circuit used is an emitter-follower which can be made on a servo extension lead that plugs onto the D2 header.
The 2N3904 is laid flat-face-down and the emitter soldered to the buzzer long leg (buzzer pos).
The 3-wire servo cable is then soldered to the buzzer neg, collector & base, then insulated with heatshrink or hot-glue.

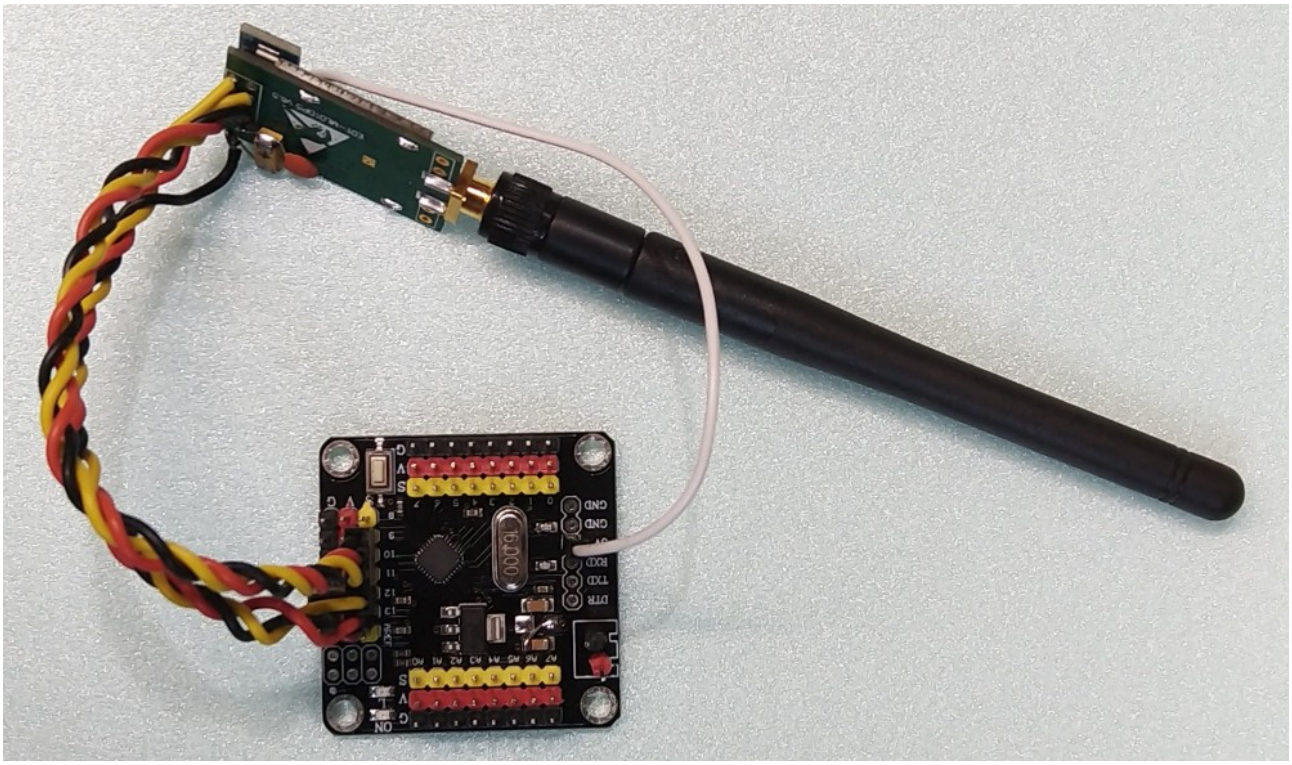
The completed buzzer assembly is mounted on the D2 header.
Any ebay-board encoders I supply will have the buzzer assembly ready-made & tested.

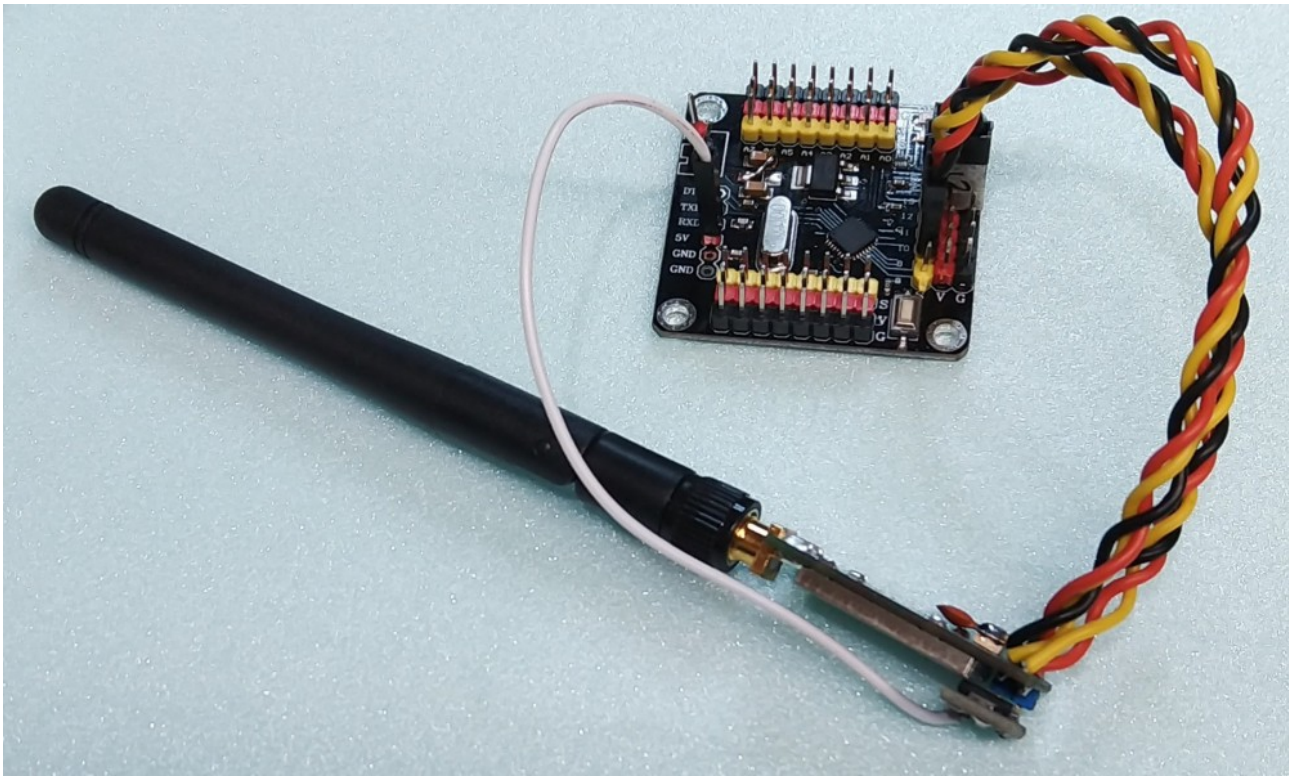
Some use an unbuffered buzzer directly powered by a AtMega328P pin. Its personal choice, I would advise against it as the buzzer is a highly inductive device which introduces a lot of hash into the processor pin, which is plainly visible on a scope. Buffering the buzzer almost completely isolates the hash.





The NRF assembly plugs onto D12, across D9, 10 & 11, and the single socket onto D13.
The positive feed connects to the single pin as mentioned above.


GND
GND
RXD
TXD
DTR
NO


D1
TX1
RX1
5V
GND
GND

Calibration:
This must be done exactly to the instructions – not following the process is the single most common problem.
To calibrate the sticks, centre all trims, and dont touch them throughout the calibration process.
Hold the D8 Single-Channel button down, or fit a bind-plug on the D8 header.
Do not release the button or remove the link until calibration is complete.
Still holding the button (or with the D8 link in place), switch on. Keep the encoder switched on until told otherwise!
Do not switch off!
Still holding the button (or with the D8 link in place), move each of the sticks into all four corners a few times.
Leave the trims alone during this step.
Still holding the button (or with the D8 link in place), centre all the controls including the throttle.
When done, you let go of the button or remove the D8 link, and everything should spring to life.
Because the throttle is central you should hear the throttle warning. The encoder saves the maximum, minimum and neutral values from each pot to flash. Every time you power-up it reads the calibrated values from flash and uses these for the stick maps. You can repeat the calibration as often as you like but it only needs to be done once unless you change the stick configurations. Now, you may switch off when you're ready :)

Thats all, its really easy – but failing to do the calibration exactly as above is the main source of problems.

Optionally, and at risk of complicating a simple but inexplicably baffling process - the throttle response can optionally be made non-linear during calibration by 'centering' the throttle stick to say 2/3 rather than to neutral stick.

Calibration remembers extreme up and extreme down stick and also wherever the stick is when you let go of the button, so if your throttle is at say 2/3 when you come out of calibration, then the overall range of low throttle to high throttle will be normal, but 1.5ms will be at 2/3 stick. This means that half the throttle servo travel, mid-to-low is expanded into the bottom 2/3 of the stick movement and the other half servo travel, mid to high is compressed into the top 1/3 of the stick movement - where normally it would be half and half equally. This gives finer control at low throttle and courser at high. Of course throttle stick at centre no longer gives throttle servo at centre, though overall travel remains the same. Most IC engines are non-linear.

Its one of those things thats much easier to demonstrate than to explain!

Part 2 – the receiver (or rather, the many receiver options...)

*** See appendix for updated receiver software & operation ***

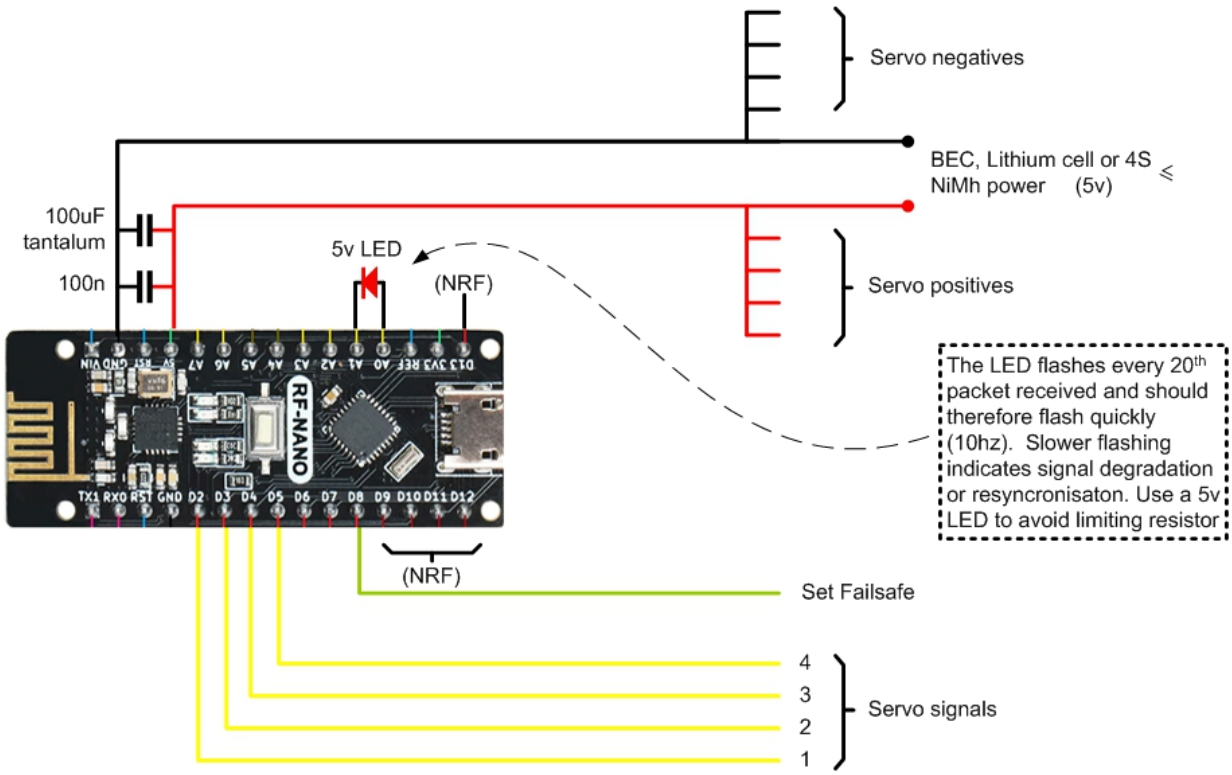
Several hardware options have been used for the receiver as the project developed:
A Keywish RF-Nano (from Aliexpress)
An Arduino 3.3 volt Pro-Mini with a standard NRF24L01+ board
Tobes excellent receiver PCB with a standard NRF24L01+ board
Tobes all-in-one, single PCB combining the processor and NRF sections. Check with Tobe for availability.

All work equally well and are functionally identical.

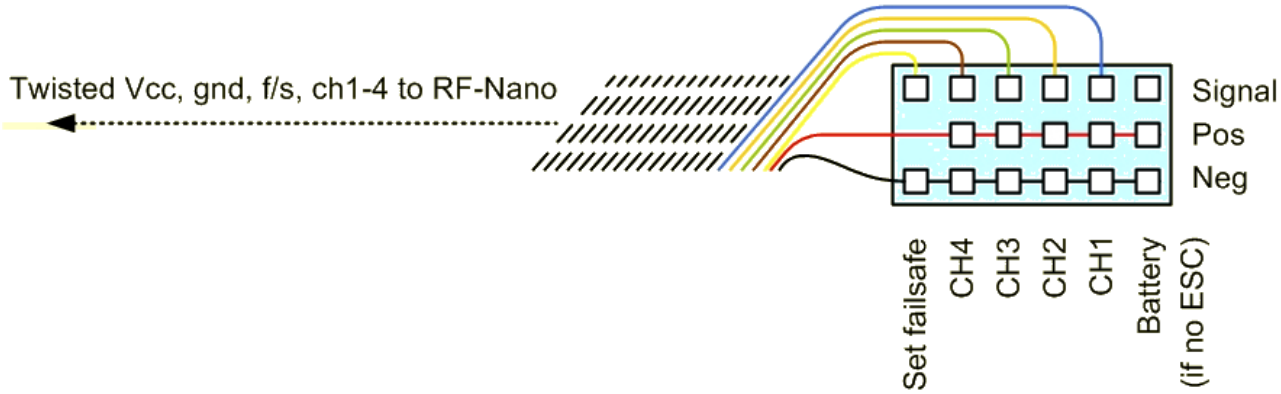
Here for example is the Keywish RF-Nano receiver.
Note that all the negatives are commoned together, as are all the positives.
The servo connections can be individual servo leads, or a common block, your preference.
Try to avoid heavy servo leads as they are difficult to fit through the PCB holes, and bulky to common.

The LED is either a 5v type, or if a conventional LD is used, it must have a 330Ω series resistor. A 5v type is much easier and more convenient as it can be mounted flush to the board. Please dont omit this LED – the system will work without it, but its a valuable indicator of good reception or of failsafe.

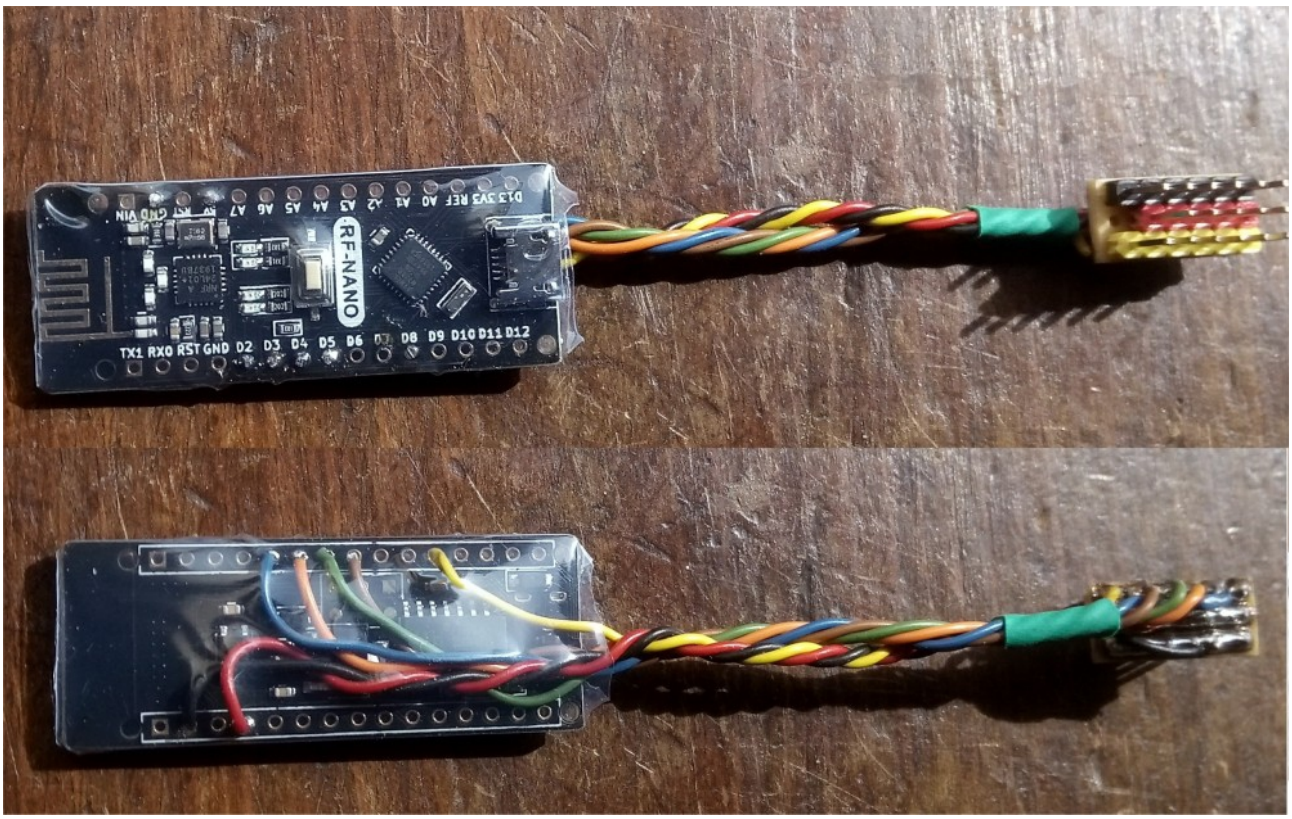
Optionally the 'set failsafe' pin can be brought out to the block connector (if used), such that a bind-plug momentarily applied to this connection will set the failsafe to the current control positions. When this is used, the servos give a quick 'waggle' to confirm failsafe has been set.



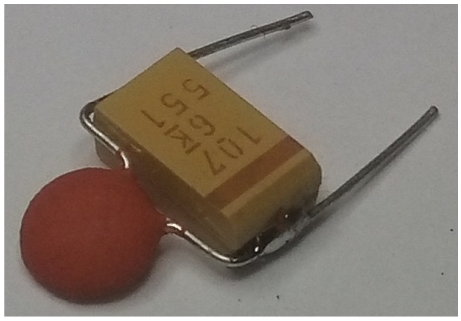
The type of 0.1” perfboard with separate, square pads can be used to make a servo block with red, black & yellow header pins. Twist the collection of wires for neatness.



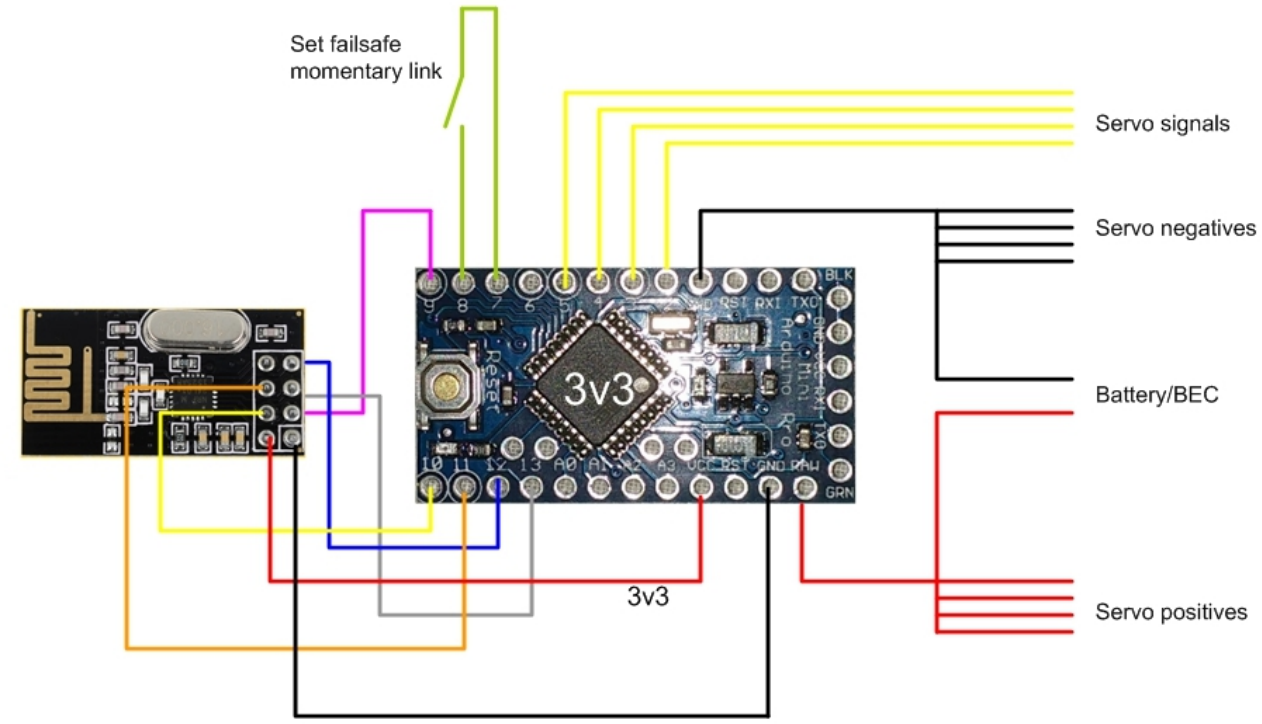
In the photo immediately below, the failsafe option is nearest the RF-Nano (note no positive pin), followed by the servo outputs, and power input on the very right:

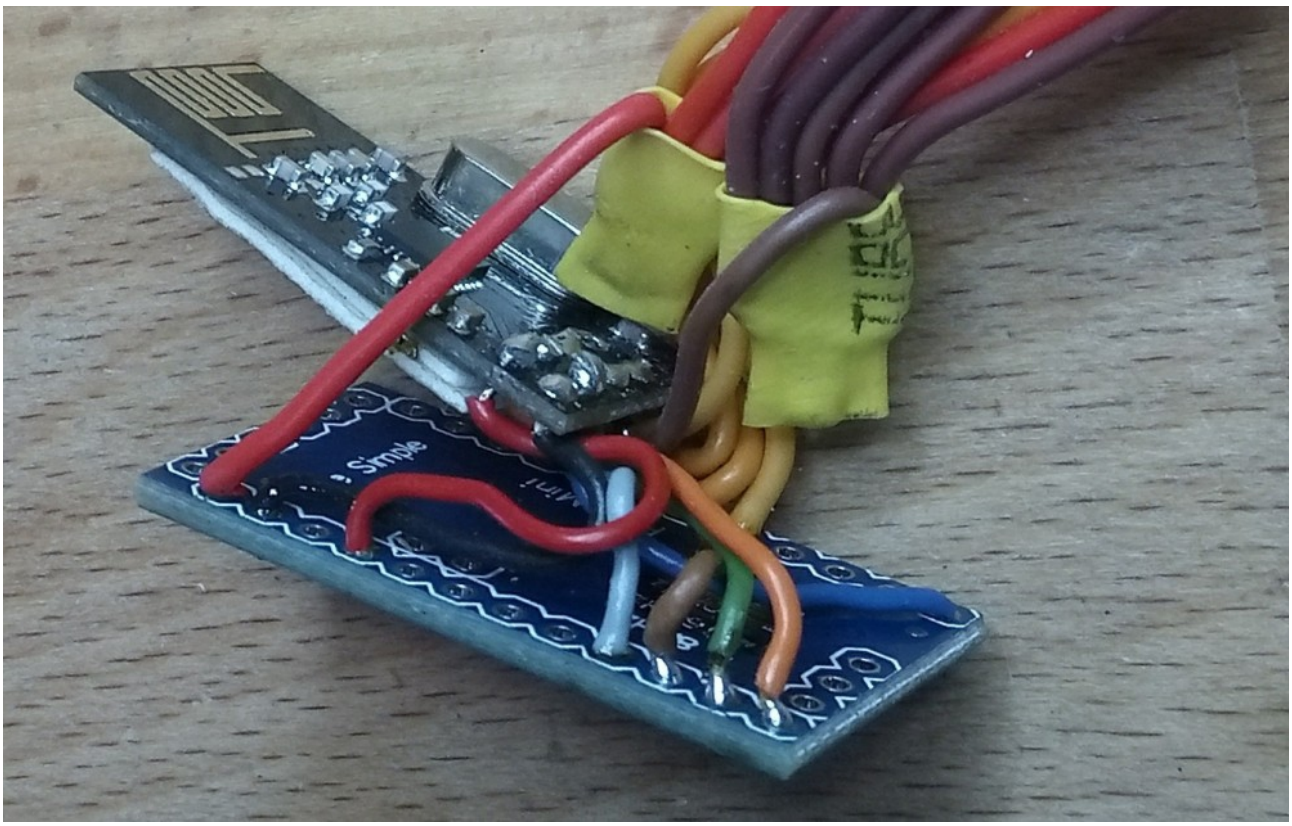


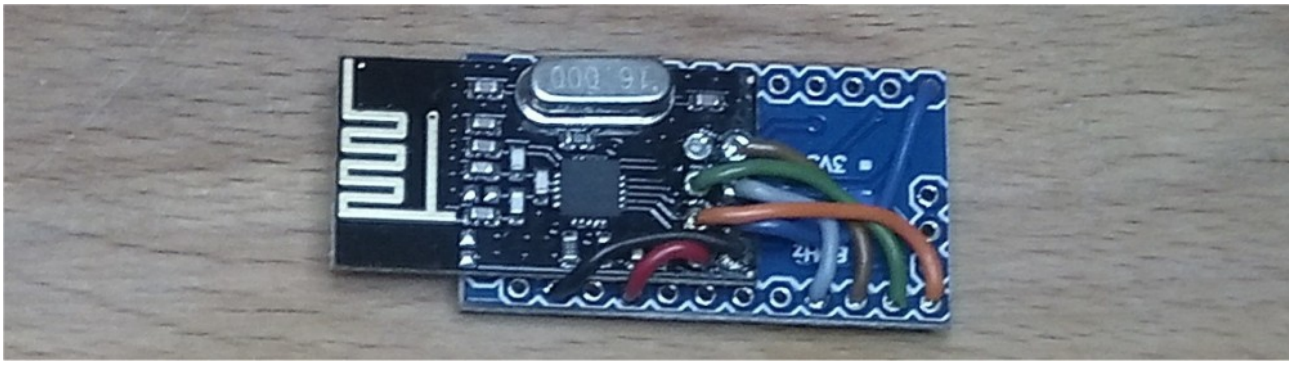
Its advisable to solder a pair of paralleled capacitors across the power input of the RF-Nano, I use a 220µF and a 100n which are first soldered together like this, then tagged onto the ground and 5v pins of the RF-Nano:

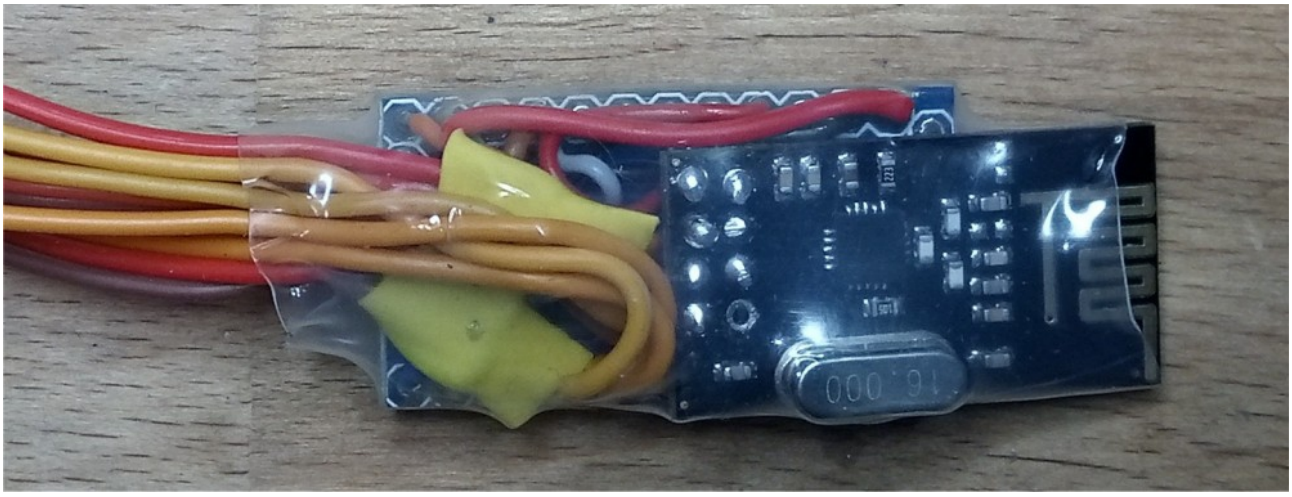


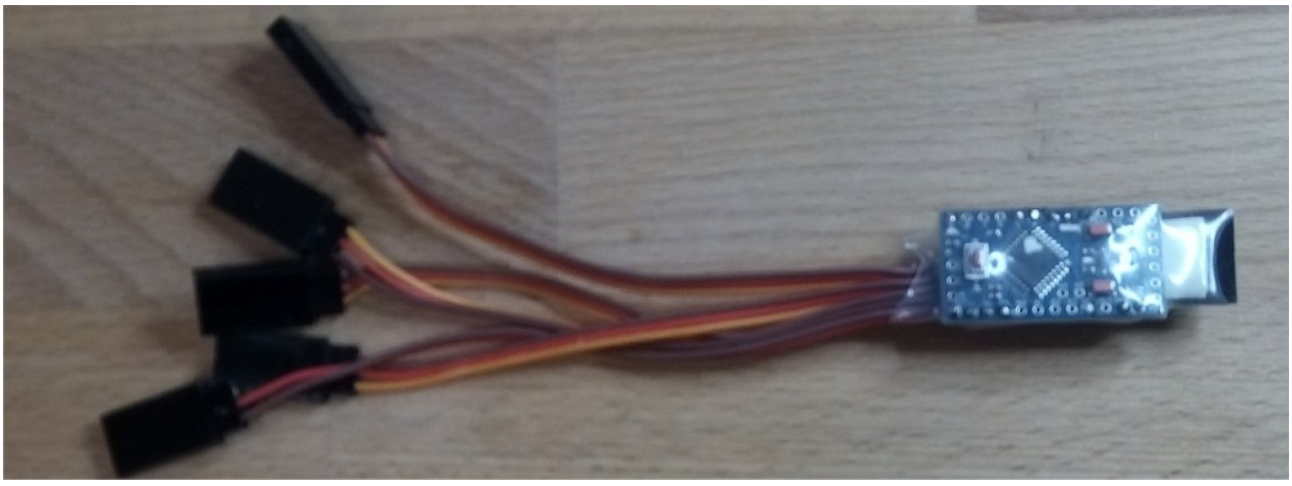
The 3.3v 8Mhz Promini and NRF board receiver:
This is a bit more fiddly to wire up, but can make a smaller receiver. Its also more tolerant of higher battery voltages.











As with the RF-Nano receiver, remember to add the parallel capacitors between GND & Vcc.

Appendix:

Receiver code update, posted Nov 2024.

Since late 2022 I've been flying an updated receiver software, totally compatible with the existing transmitters but with enhancements. There are no changes to the receiver hardware.

Mixing and reversing functions now happen within each individual receiver rather than in the transmitter.
In the context of this simple project, moving these functions into the receiver makes transmitter model memories redundant.

If a model needs a surface reversing, the reversal in stored and performed in that models individual receiver, which to me is far more logical than selecting a model memory in the transmitter. This means I can fly a variety of model types, full-house, rudder-elevator or an elevon flying wing, regardless of servo directions - *with any of my transmitters*, propo, reed or s/c - without changing or selecting anything on the transmitter. You can even swap from a S/C to a propo transmitter mid-flight! Here's a brief demo showing a mixer-wing and a rudder elevator model, both operating without any transmitter changes or selections: https://www.youtube.com/watch?v=6NA8ovWJu0U

Servo reversing:

This is used in place of the previous 'transmitter-based' reversing method. Its similar but different.
With the transmitter powered on and operating normally, hold the transmitter stick hard over whilst powering on the receiver to reverse a function. The setting is retained in that one particular receiver's flash, and of course only affects that receiver. Nothing changes in the transmitter.

The elevon mixer is also in the receiver and is configured at programming time by un-commenting the mixer section, to suit the model.

Also theres a #define for the reversed traffic LED on the early PCBs produced by Tobe, if your traffic LED doesnt flash, try changing this #define. Its value should be 1 or 2, nothing else.
This saves all the messing about swapping the allocations of A0 & A1 which was a bit of a faff.

This receiver code has actually been in use for about 2 years but I've only just got around to posting it :-)
The update is only to the receiver code, there are no changes to the transmitter and existing transmitters will work fine with this new receiver code. I haven't even taken out the (now obsolete) transmitter reversing!

Cheers
Phil

Tobes receiver PCBs will be added with Tobes permission