'rx2at' debug and setup instructions

When you have all the hardware together and downloaded the 'sketch' into the Arduino, run a check to make sure 'rx2at' interprets the signals right.

Ground the pin labelled (10), connect a serial converter (5V signals!) to the Arduino (most likely you downloaded the 'sketch' with it) and power it up. You can NOT use the Arduino IDE Serial Monitor for this because it can NOT send a single control character with it... Use Hyperterminal or something else, set it up for 115200, 8 bits, 1 stop, no parity, no handshake.

Then type Ctrl-B (Control AND B), this is the 'magic' start character for 'rx2at'. It will show you something along the lines....

RX2AT	revis	10n	
AT2SO	locat	ion, cl	necksum
DRUPD	25.0 H	HZ 40	.0 ms
-RX-	f[ms]	p[us]	value
THRO	22.6	1500	0
AILE	22.6	1500	0
ELEV	22.6	1500	0
RUDD	22.6	1500	0
GEAR	22.6	1100	-1000
AUX1	22.6	1900	1000

. .

_ . . . _ _

'rx2at' measures the frametime (pulse to pulse) in [ms] and the pulse length in microseconds. The value column is the interpreted value. If you move the sticks or flip the switches on your TX the numbers should reflect that. Set your TX up for an airplane, meaning NO heli mixing of the channels, you want every stick on a separate channel. In a mode 2 TX the THRO channel is the left stick moving up and down, the RUDD channel is your left stick moving left and right, the ELEV channel is your right stick moving up and down, the AILE channel is your right stick moving up and down, the sticks are interchanged.

Make sure you got the sticks and switches going to the right channels, if not, fix the wireing or change the bit assignement in main(). Also make sure that each channel is only affected by a single actuator (switch or stick)! When all sticks are centered the value of RUDD, AILE, ELEV and RUDD should be 0. Check your trimms if they are not. This is important, because this will make the drone hover in place. You can also change the deadband in 'rx2at'.

Once this works, the polarities need to be checked... THRO up -> positive values (PCMD GAZ value) RUDD right -> positive values (PCMD YAW value) ELEV down -> positive values (PCMD PTC value) AILE right -> positive values (PCMD ROL value) Invert the channels on your TX until it behaves the way above...

Your TX/RX might have different switches and push buttons from what I use (DX7/AR6100), so you might have to become creative here:

'rx2at', as it is, looks at the GEAR channel to see if you want to launch or land. GEAR up -> positive values if GEAR > 0 -> launch drone if GEAR < 0 -> land drone

On mine the other switch is a three position switch which I made 'rx2at' interpret this way: AUX1 up -> negative values AUX1 center -> 0 AUX1 down -> positive values

if AUX1 = 0 -> normal, 'rx2at' launches/lands drone according to what GEAR says if AUX1 > 0 -> sends FTRIM/LED/COMWDG command to drone, return it to center once

you see the LED on the drone blink. 'rx2at' will not launch the drone if this switch is not in the center position.

if AUX1 < 0 -> EMERGENCY, sends ESTOP to drone, if you are in the air when you hit this it will kill the motors and drop out of the air. If you are on the ground, it merely resets a previously encountered EMERGENCY (drone LEDs go red and back to green when you switch back to center).

Once you got all this right, you need to make sure you know what is going to happen if your receiver looses contact with the transmitter. So, turn the TX off and check what the signals do. If at least one of them shows up in red you are ok, because 'rx2at' will land the drone if this happens. On my setup this did not work at first, because in this situation the AR6100 keeps on sending what it received last exept for the THRO channel, which will transmit the value the throttle stick had when it was bound to the transmitter. To fix this in my setup, I did rebind the AR6100 with a very low throttle stick value (-150%), which fixed this (after putting the gain back to 100%). Then turn the transmitter back on and make sure everything comes back to normal.

Once all this works, disconnect GND from pin (10) on the Arduino and hook it up to the drone. Power up TX, power up drone (standard practice) and wait...

- On my setup the following things happen:
- 1) Drone turns on RED lights and boots
- 2) Arduino GREEN light blinks first once every second while the drone is booting and it has not figured out the receiver yet. As soon as it has a grip on the receiver it goes to a double blink every second while it burns the rest of the drone boot time (25 seconds).
- 3) Drone turns off RED lights and turns on GREEN lights.
- 4) Arduino GREEN light blinks ON/OFF, .5 second rythm, while it waits for a 2 second 'quiet' time on the serial port, which means the drone is done with all the boot messages.
- 5) Arduino turns on GREEN light while it uploads and launches 'at2so' (quick).
- 6) 'rx2at' sends the init and configuration commands to 'at2so'. Since there are some LED commands embedded, the drone's LEDs will flash on begin and end of the configuration.
- 7) You are good to go! Do an FTRIM before you launch and have fun.