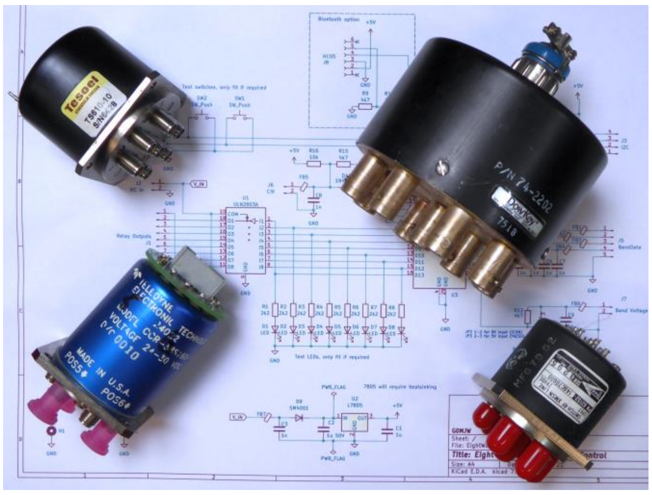
An Arduino based antenna switch controller

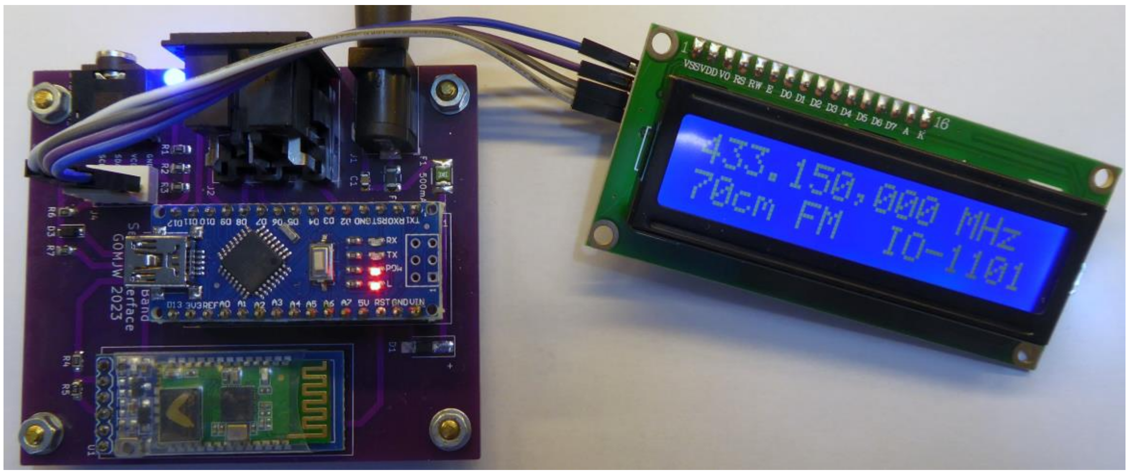
I have been frustrated with the lack of antenna outputs on my Icom IC-7300 for a while. I have a relay based antenna switch but it needed to be manually controlled and I wanted to automatically select an antenna depending on the band in use. Ideally changing the antenna configuration should not require re-programming. I also needed a band select for a linear amplifier, which is essentially the same function when you take away the relays. I built a 6-relay design with an Arduino connecting to the IC-7300 CI-V bus, which provides data on the frequency which can in turn be used to select antennas using inexpensive multi-pole relays which occasionally appear surplus at good prices.



Multi-way Antenna switches

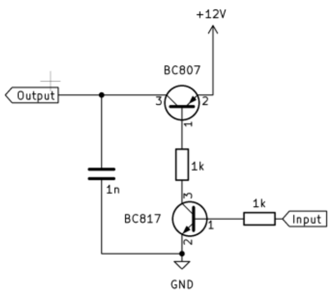
HF antenna switches based on standard relays are also widely available. I bought and made up one of these to use with my IC-7300. This used a positive voltage to switch the relay coil rather than the more usual method of grounding the coil. Hence my first controller used six sets of BC817/BC807s configured as high side switches. I developed the firmware using ideas from Nick Gammon, ON7EQ, DM2RM, VE1ZAC and OK1CD and code I had developed for my Icom IC-7900 PTT distribution board.

When I got an Icom IC-705, I needed to control the filter bank in an HF linear amplifier I was building. The IC-705 does not have a CI-V port but does provide CI-V data over Bluetooth. Compatible Bluetooth serial data modules are readily available at low cost, so a new PCB was developed.

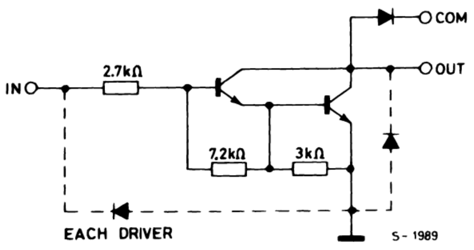


IC-705 Band outputs via HC05 Bluetooth module

I replaced the many discrete SMD transistors with a ULN2803A Darlington array which combines all the transistors, resistors and protection diodes in an 18 pin DIP IC package. It uses low side switching, but that's fine for most relay switches and those that use high side can often be re-wired to work.

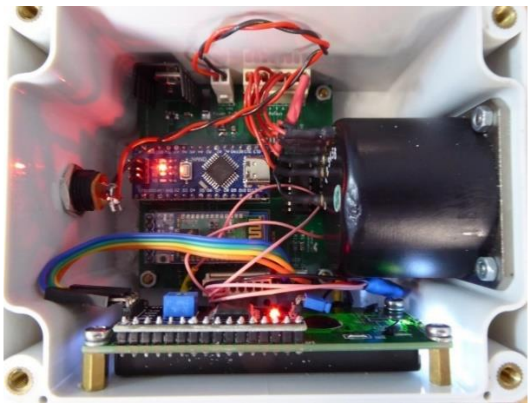


High Side Switch



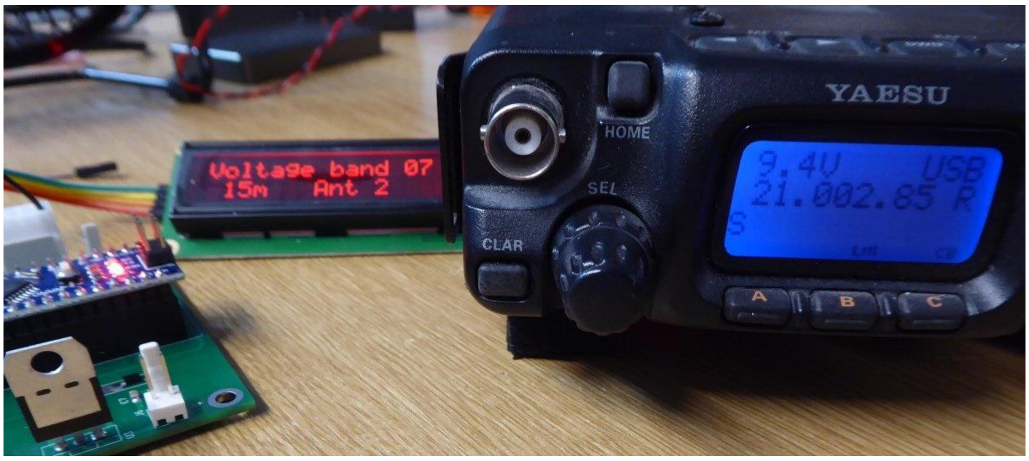
Functional schematic of a ULN2803A switch cell

This worked well driving a 6-way SMA relay so I developed a second board, essentially the same but without relay drivers to provide band data to my linear amplifier. Only Icom radios use the CI-V bus and there are other radios in use with similar requirements to control antenna switches.



Bluetooth Switch and proof the box you chose will turn out to be too small

There are various options for these other radios. Many have RS232 or USB connectivity. The Yaesu FT-817 and some other radios have a voltage output depending on the band and the Arduino has an analogue to digital converter which can be used to detect that band.

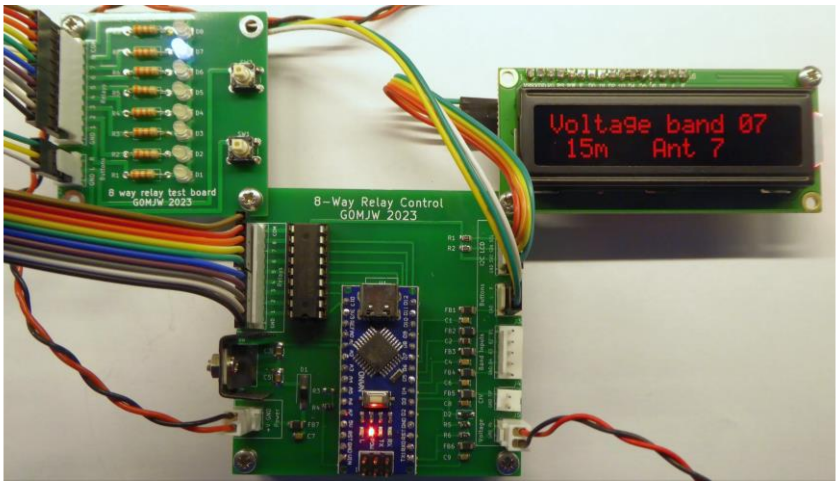


FT-817 control via band voltage input

Other radios, including the Elecraft K3 provide band data as four TTL level signals on an accessory port. The BATC Portsdown and Langstone projects also use this method. This allows the selection of up to 16 bands, though these input methods only indicate the band, not the specific frequency or mode in use.

Rather than having a different board for each of the band indication standards and applications, I developed a more universal board and firmware. A header was added to allow an HC05 Bluetooth module to be used with the Arduino's built in serial port

I developed a test PCB with buttons and LEDs to indicate what the board was doing. The test board functionality proved useful and has been integrated onto the main board.



Prototype 8-way multi-input controller

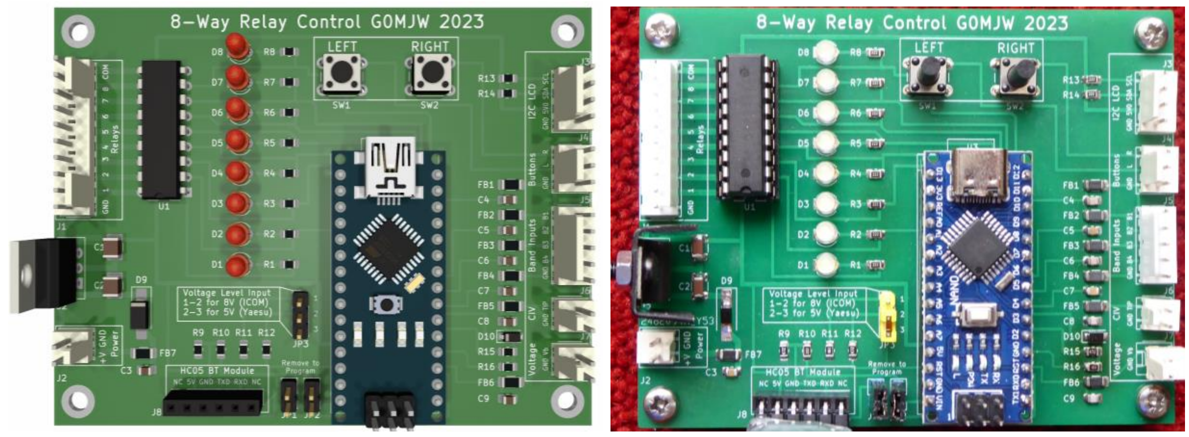
Firmware

The firmware is written in the Arduino IDE in C++ with configuration in a header file "definitions.h". It uses a two-button menu system. A short press on the left button changes to the next antenna relay down and a press on the right button select the next antenna relay up. The selection is stored in EEPROM so the switch will remember to it when that band next is selected, even after power cycling. At power on, a long press of both buttons enters a configuration mode. The left button cycles through menus, the right button selects. This allows setting the detection mode, the CI-V address, if used, and several other options, including a reset to defaults.

The CI-V input expects 9600 baud. It is possible to set this higher, but much higher and the Arduino may not be able to keep up. To avoid saturating the bus, the radio is polled once per second. That means band changes are not instant. It should not interfere with logging software. The CI-V address needs to be configured, the default is 94 Hexadecimal for the IC-7300. This can be changed and stored in EEPROM via the Menu.

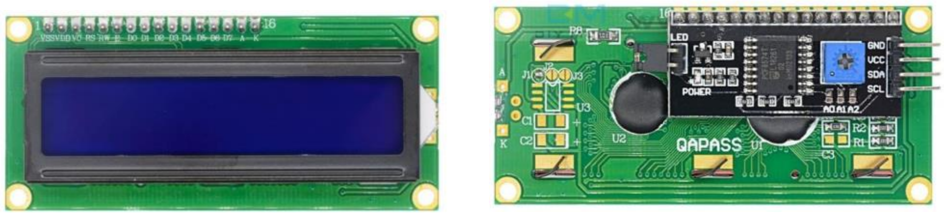
Design

I developed the PCB in KiCAD, it is free, easy to use and has a good 3D rendering capability that is especially useful in visualising how parts fit. The PCB uses a mixture of through hole and surface mount parts, with relatively large surface mount components that are widely spaced. There are test LEDs and control buttons for setting up on the PCB. I used an Arduino Nano V3 in this design. The newer Nano Every will also work but not the more recent 3.3V only Arduinos.


8-Way Relay Control G0MJW 2023
LEFT
RIGHT
Relays
I2C LCD
Buttons
Band Inputs
CIV
Voltage
Power
Voltage Level Input 1-2 for 8V (ICOM) 2-3 for 5V (Yaesu)
HC05 BT Module
Remove to Program
8-Way Relay Control G0MJW 2023

KiCAD 3D render and final PCB

The display is a standard 16x02 LCD with an I2C backpack. These greatly simplify connecting an LCD. The backpacks have a configurable base address which need to be set in the header file, and is typically 27 or 3F. Another I2C display type, e.g., 04x20 LCD or an OLED module could be used, with appropriate changes to the software to drive it.



LCD Module and I2C backpack

The Bluetooth module is optional. RF engineers will know not to mount the project in a screened metal box if using Bluetooth.

HC-05 Bluetooth module

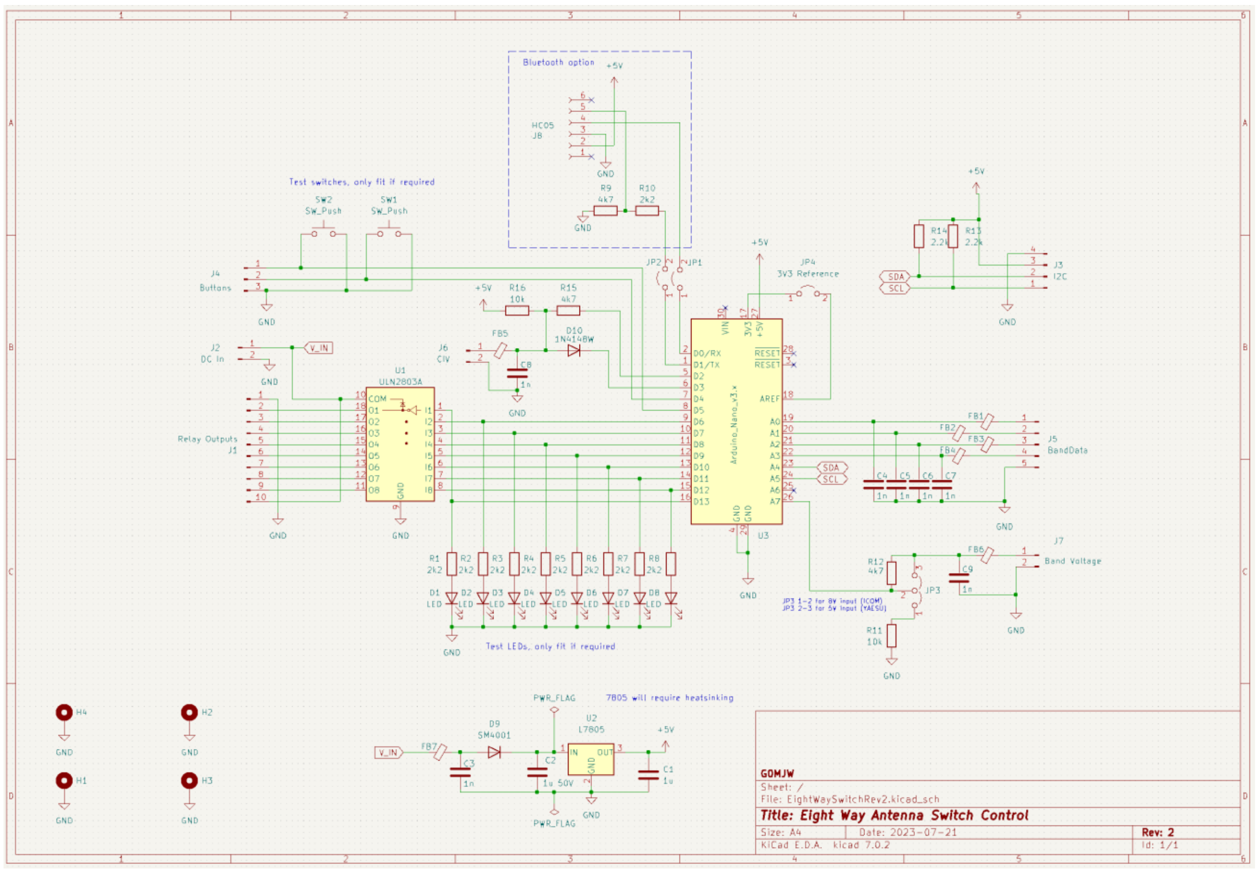
The module uses the Nano’s serial interface which is also used for uploading the firmware. It is necessary to remove the jumpers JP1 and JP2 when updating the firmware. Theoretically you could program it via Bluetooth, or with appropriate changes to the code support other standards

The TTL band data option is based on the Elecraft standard, 0 = no band, 1 = 160m, 2 = 80m etc. This table is also stored in the header file and can be adjusted as required. The pins are configured as inputs without internal pull ups. They expect TTL levels, no more than 5V.

The voltage input option uses a jumper (JP3) to select either direct, up to 5V input, for the FT-817, or a voltage divider for 8V for some Icom radios as the Arduino ADC is limited to 5V. With other radios it will be necessary to modify the lookup table in the header file. The FT-817 has quite a high source impedance, around 7k. Other radios may be similar, so pay attention to this when calibrating. As a bonus, when using the CI-V or band voltage inputs, it is possible to configure the 4 band lines as TTL level outputs for controlling other hardware. This functionality must be set up before flashing the firmware. This is not a menu option as accidentally setting the Nano pins to outputs while they are connected to a radio's outputs could cause damage. As they use the same pins, band data input option is not unavailable when using the band data output and vice versa.

The relay supply, input is also used to power the board using a LM7805 5V regulator. A header (J1) carries the relay outputs. One side of each coil should be connected to the positive supply common (COM) line. The other side connects to an ULN2803A output. When its input line is set to 5V, the ULN2803A grounds the corresponding output, in our case switching the relay. There is no need for diodes across the coils as these are included in the chip. J1 has a ground connection to facilitate powering other equipment from the relay supply but be careful not to take more than an amp or so. It would be wise to fit a fuse on the DC input if there is any risk of short circuits.

Schematic

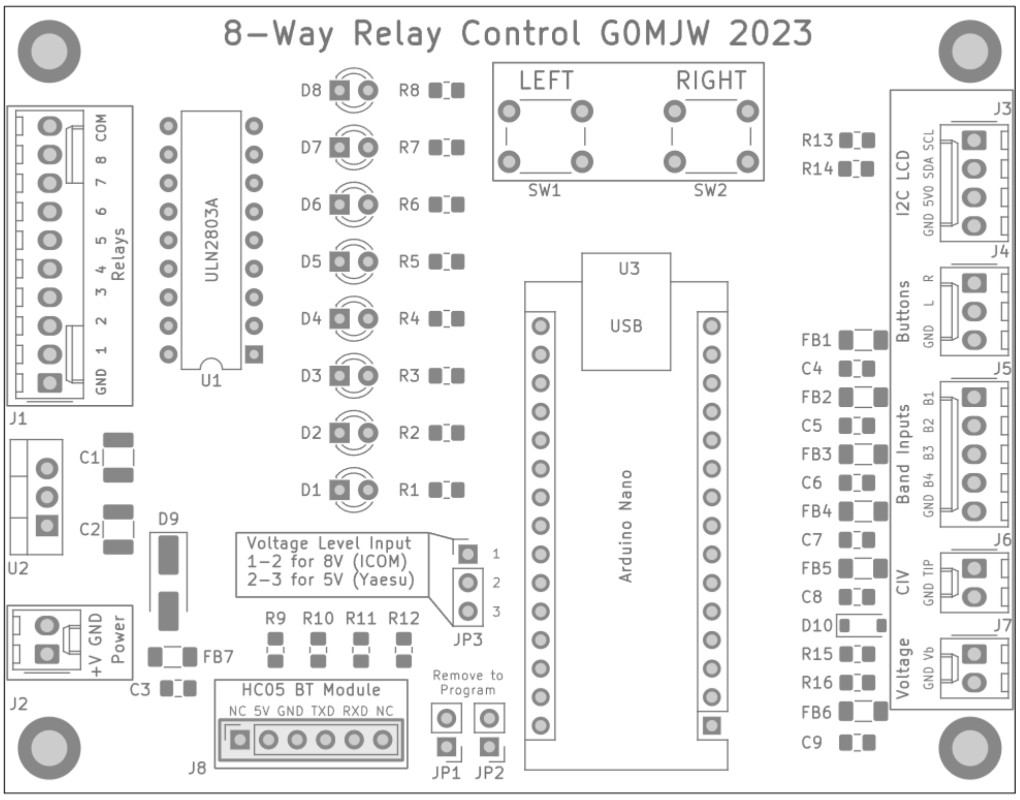


Parts list and construction

None of the passives are especially critical. Pay attention to the voltage rating of the input capacitor though and do not exceed 28V. I find it best to install parts by component height, starting with the surface mount parts. The LM7805 needs a heatsink, especially if used above 12V. A drop-in switching equivalent for example the RECOM R-785.0-0.5 could be used instead, but these usually cost more and tend to generate RF noise. Pretty much any ferrite bead will do, something that has a few hundred ohms impedance at the frequencies being used.

I prefer to use sockets for the modules. You can solder them in, but it is then very hard to remove them later. I also socket the ULN2003A for easy replacement in case of accidents.

Reference	Part	Package	Qty
C1, C2	1uF 50V Ceramic	1210	2
C3 – C9	1n Ceramic	0805	7
R1 – R8, R10, R13, R14	2k2	0805	11
R9, R12, R15	4k7	0805	3
R11, R16	10k	0805	2
D1 – D8	LED	3mm LED	8
D9	SM4001	SMA	1
D10	1N4148W	SOD-123	1
U1	ULN2803A	DIP-18	1
U2	L7805	TO-220	1
U3	Arduino Nano		1
U3 Headers	15 way pin socket	2.54mm spacing	2
SW1, SW2	Push Switch	6mm push switch,	2
FB1 – FB7	Ferrite Bead	1206	7
JP1, JP2	2 pin jumper	1x02 pinheader 2.54mm	2
JP3	3 pin jumper	1x03 pinheader 2.54mm	1
J1	10 way pin header	Molex KK-254	1
J2, J6, J7	2 way pin header	Molex KK-254	3
J3	4 way pin header	Molex KK-254	1
J4	3 way pin header	Molex KK-254	1
J5	5 way pin header	Molex KK-254	1
J8	8 way pin socket	2.54mm spacing	1
BT Module	HC05 Module		1
LCD Module	Display	16x02 LCD with I2C backpack	1


8-Way Relay Control G0MJW 2023
LEFT
RIGHT
SW1
SW2
COM 8 7 6 5 4 3 2 1 GND
Relays
J1
ULN2803A
U1
D8 D7 D6 D5 D4 D3 D2 D1
R8 R7 R6 R5 R4 R3 R2 R1
U3
USB
Arduino Nano
R13
R14
I2C LCD
GND 5V0 SDA SCL
J3
J4
Buttons
GND L R
J5
Band Inputs
GND B4 B3 B2 B1
J6
CIV
GND TIP
J7
Voltage
GND Vb
FB1 C4 FB2 C5 FB3 C6 FB4 C7 FB5 C8 D10 R15 R16 FB6 C9
C1
C2
U2
D9
+V GND
Power
J2
FB7
C3
Voltage Level Input
1–2 for 8V (ICOM)
2–3 for 5V (Yaesu)
1 2 3
JP3
R9 R10 R11 R12
HC05 BT Module
NC 5V GND TXD RXD NC
J8
Remove to Program
JP1 JP2