

Experimenters' Corner

Back to radio basics: the diode ring mixer

If you have looked at QRP rig designs or simple superheterodyne receivers, you will have undoubtedly come across a diode ring mixer. This is a component found in many published radio designs, and I was recently reminding how it functions. A circuit diagram of a diode ring mixer is presented in **Figure 1**. Note that the orientation of the trifilar transformer windings is critical and the blobs on the circuit diagram represent the same side or orientation of the windings. The job of the diode ring mixer is to take the RF and local oscillator (LO) signals, and mix them to make the intermediate frequency (IF). Typically, this will be the sum or the difference of the two inputs. Let's build one and see what's actually going on.

To construct our diode ring mixer, we need four diodes, ideally 'matched' from a selection. We also need two trifilar-wound transformers. I wind my trifilar transformers using three different colours of enamelled copper wire. I twist the three wires together using the 'door handle and drill' method described previously when we made some bifilar wound baluns during previous antenna musings.

Once I have my wire twisted together, my trifilar transformers consist of ten turns on an FT37-43 toroid. To match my diodes, I simply used the diode test setting on my digital multimeter (DMM), connecting the diode to the device and then letting the PN junction warm up a little and the reading settle before noting the value presented. I tried to find four diodes that gave very close or identical readings. In reality, this is only matching the DC characteristics of the diodes, good enough for a bench test but at RF frequencies, junction capacitance balance matters just as much for carrier suppression so there is a potential trap here. Experimenter beware!

In my example I used a small signal diode, the 1N4148, mainly because I had a load lying around. A Schottky diode like the 1N5711 may be a better choice but I didn't have anything to hand. In theory Schottky diodes, with lower forward voltages and faster switching times, should significantly improve the noise figure and conversion loss of the mixer.

My bodger's diode ring mixer has been constructed ugly style and is shown in **Figure 2**. A mixer has one basic job; it takes two signals at different frequencies and then combines them using a non-linear process. In the real world, there are all sorts of harmonic components in the mix too, so we end up with the sum, the difference, and a host of other signals. Normally it is only

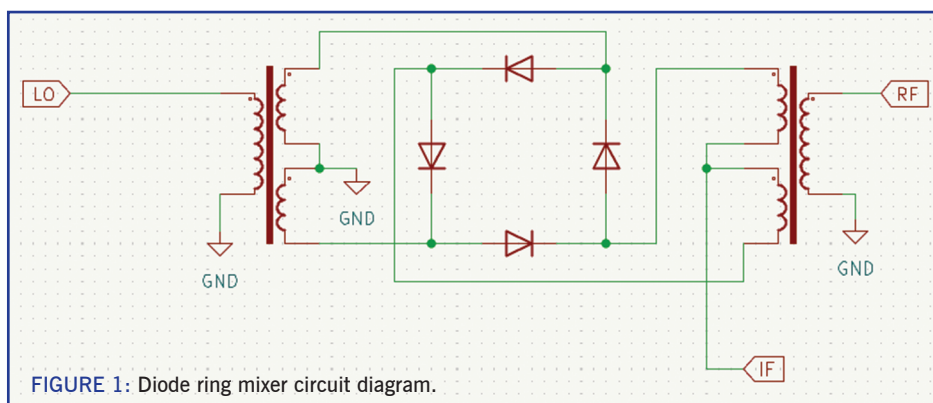


FIGURE 1: Diode ring mixer circuit diagram.

one of the output products that we are interested in and the others get filtered away further down our radio design stages. We now need to use our imagination for a while. In my example, we'll have 4MHz into the LO port and 7MHz into the RF port. In these mixers the LO drive level is important as it must be large enough to hard-switch the diodes into saturation. A level of +7dBm is typical with Schottky diodes but, with my bodger's build using 1N4148s, +13dBm may be needed.

Try to picture the signal that's being injected into the LO port of our mixer. It is an AC signal swinging in polarity. At the peak of the signal there is one pair of diodes that will conduct and at the trough of the signal the other diodes will conduct. Our diodes are effectively switching between these two conduction states at the LO frequency. As a consequence, the RF port input at 7MHz is either replicated or inverted at the IF port. This is the same as a multiplication by 1 or -1 at the rate of the LO. I could baffle you with some fancy maths about multiplication of sine waves at this point but I think it's much easier just to look at a visualisation of the output so we can see what's going on. I have taken the output of our mixer, connected it to my spectrum analyser and annotated the output as **Figure 3**.

We expect to see the sum of my inputs which is $4\text{MHz} + 7\text{MHz} = 11\text{MHz}$ and the difference which is $7\text{MHz} - 4\text{MHz} = 3\text{MHz}$. In reality, we have all sorts of other products as well. In a typical superheterodyne design, the mixer might be followed by a crystal filter which would remove everything except the wanted signal from the output. It is noteworthy that the 1N4148s work well at HF when driven with a beefy local oscillator. However, 1N4148s fail as mixers above HF frequencies because of their reverse recovery time, which is around 4ns. If you were to try

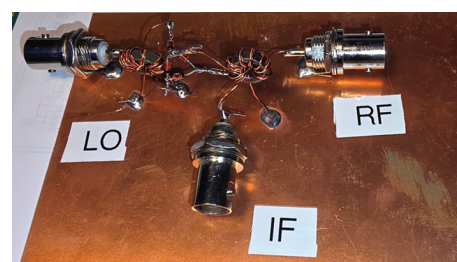


FIGURE 2: Ring mixer bodger's build.

my bodger's mixer on the two-metre band (144MHz, where the period is under 7ns), the diodes never fully turn off, destroying the switching action.

You also often see a small attenuator pad at the LO port of a mixer like this in published designs. This is because a diode ring presents a highly non-linear dynamically changing load to the LO port. Driving it with +13dBm from a high-impedance source will cause severe waveform clipping and harmonic generation. Adding a small (say 3dB) attenuator pad on the LO port is a cheap and easy way to ensure the oscillator sees a stable 50Ω load. Add a small pad yourself and see what difference it makes to the output.

I hope that these remarks give an insight into the diode ring mixer. I would encourage you to make one yourself and see the signals. It's a basic building block that's used in many home-brew designs and great fun to play with on the bench.

Mark Jones, G0MGX
g0mgx@rsgb.org.uk

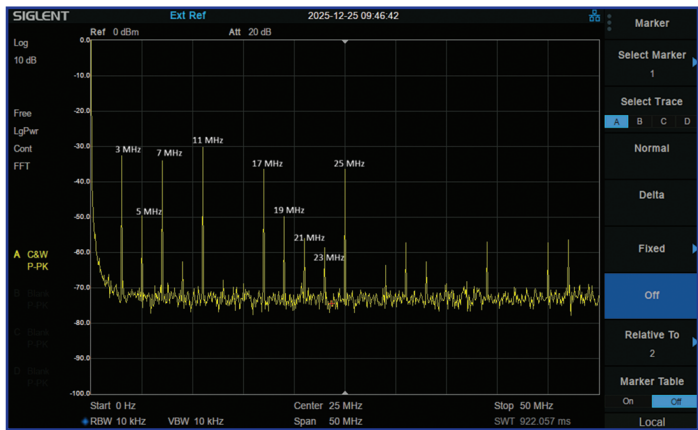


FIGURE 3: The RF spectrum of the output from ring mixer.

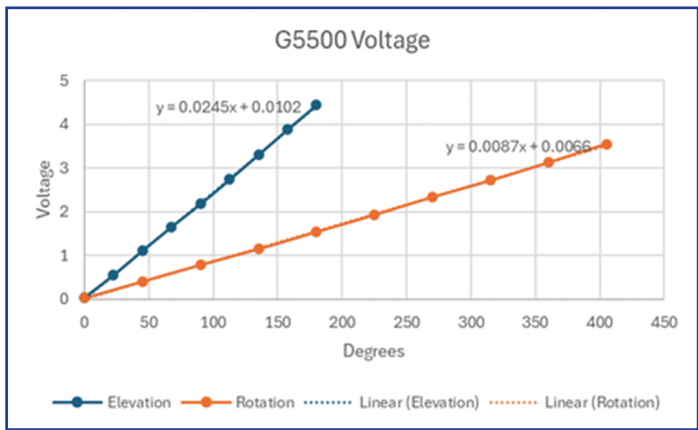


FIGURE 4: A plot of the DC voltages measured on the rotator controller.

Rotator control

For some time, I have wanted to be able to track satellites; the International Space Station, low Earth-orbit cubesats and other amateur satellites. To facilitate this, I've finally bought myself a Yaesu G-5500 DC azimuth/elevation rotator. Of course, this purchase immediately initiated a project for computer control of the rotator. Having looked around the internet for ready-made candidates, I found K3NG's Radio Artisan website [1]. There are several projects on this website,

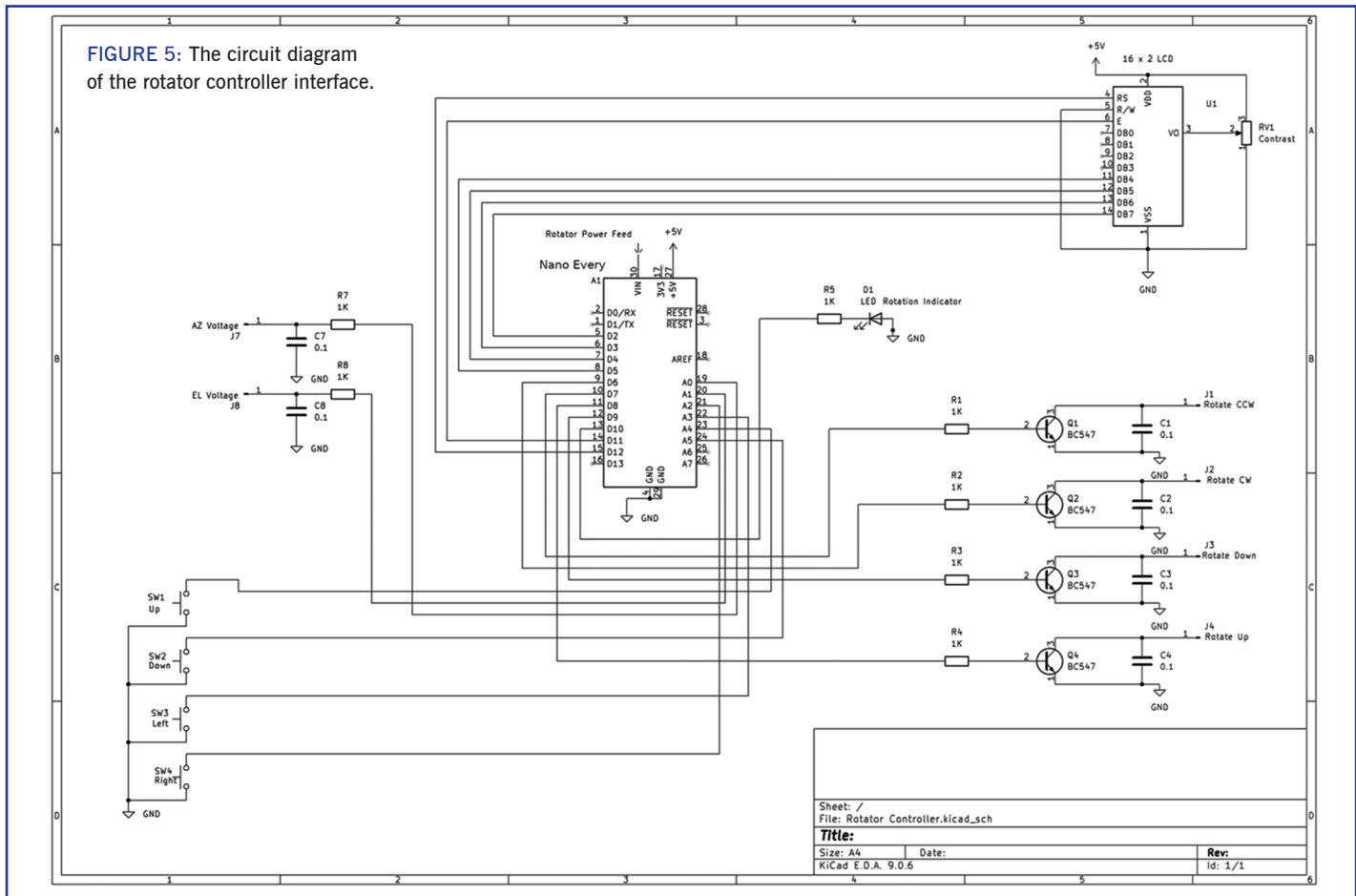
including a versatile and configurable rotator controller. I have implemented a basic version of the project.

The first thing I did was to test the connections made available on the back of the G-5500 DC Yaesu control box. There are eight pins in total as shown in Table 1. I measured voltages on pins 1 (elevation) and 6 (rotation in azimuth) at different rotator positions and plotted them in Excel. If you create a scatter plot of the values, you can add a linear trend line to the data set. One option for a linear

trend line is to display the equation on the chart. My chart is shown in Figure 4.

Showing the equation on the trend line allows us to see how to calculate the voltage for any given angle, very handy for testing any automated controller I might develop. I also thought that at some point I would need to calibrate my software against the voltages at different angles and I could now do this with a bench power supply much faster than rotating the motors repeatedly.

The circuit diagram of the interface I



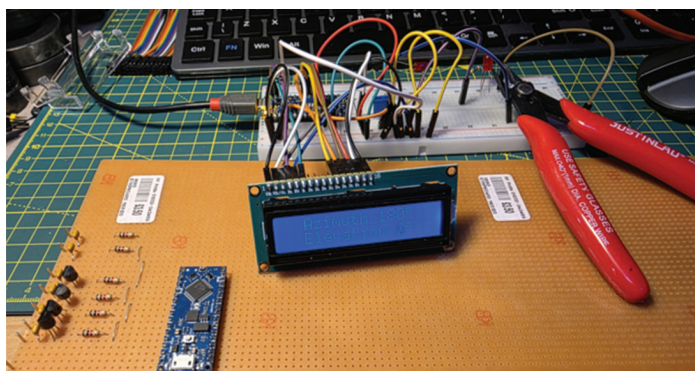


FIGURE 6: The project at an early stage in its development.



FIGURE 7: The completed project with a 3D-printed front panel.



FIGURE 8: GPSDO from the Far East.

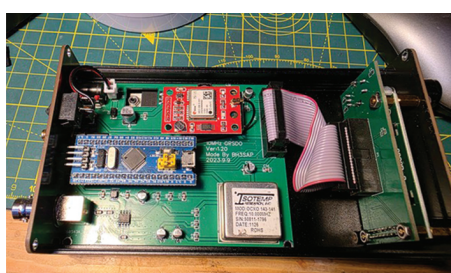


FIGURE 9: Inside of the GPSDO.

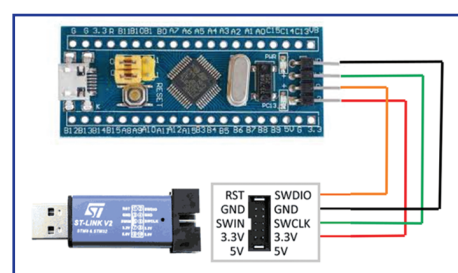


FIGURE 10: STM32 to STLink interface connections.

have built is shown in **Figure 5**. I elected to use an Arduino Nano Every as the microcontroller, primarily because I had one lying around but more importantly because it is a 5V device and the analogue inputs of the microcontroller will support the full output range of the G-5500 DC controller lines. You would need to recalibrate the Yaesu control box using the trim pots on the rear panel if you elected to use a 3.3V microcontroller. The Arduino Nano Every is an evolution of the classic Arduino Nano, sharing the exact same pin connections but featuring the more powerful ATmega4809, as well as 48kB of CPU flash memory and a higher clock speed of 20MHz. A genuine board from a reputable supplier currently costs around £13.

Please note that standard engineering practice dictates the use of diodes (eg 1N4148) across inductive switching loads to prevent transient spikes and a non-bodger's build would include these.

You could also consider other microcontrollers for this project, especially if you wanted to look at remote or web interfacing. The ESP32 may be a better choice in that case. It is a 3.3V device, so the rotator box would need to be recalibrated, but may well be worth considering in the long term.

There is some excellent support available for the K3NG projects at their groups, io message board and there is online documentation available for the project at [2].

An early stage of my implementation is shown at **Figure 6**. I built my interface and microcontroller on a piece of VeroBoard, and designed a 3D-printed front panel for a case I had here. The result is shown in **Figure 7**. The 3D-printed panel was easily created using TinkerCAD [3]. I used polylactic acid as my printing filament but for a shack that gets hot

or a unit that will sit in the Sun, polyethylene terephthalate Glycol (PETG) is a much better choice for UV and heat resistance. Note that I have used a textured build plate which gives a professional-looking textured finish seen on a lot of commercial enclosures.

The Arduino board used for the project connects to the PC via the USB connection and creates a COM port that supports the Yaesu GS-232 rotator command set so this interface will work with any PC-based software that is compatible. You can even issue commands through the Arduino serial monitor (set to 9600 Baud and carriage return for the line end selection) to test your configuration. There is also a set of commands you can issue to configure the K3NG software parameters and write to EEPROM. The set of serial commands available is documented at [4]. If you are a beginner and struggle with the serial interface, drop me an email and I'll try to include more details in a future column.

The project supports both azimuth-only and azimuth/elevation rotators, and a whole host of display options, including a nice-looking Nexion option which has been developed by VK4GHZ [5]. To configure the code for the G-5500 DC rotator, I made some changes to add elevation control and to modify the default behaviour to be an azimuth rotator that turns round 0° (rather than 180°). I enabled the basic 4-bit LCD option and added a rotation indicator on pin D10. I also added four push buttons to the front panel so that I can control the turning manually from my interface as well as from the main rotator controller. The more

ambitious among us may wish to consider a joystick so you can control elevation and azimuth at the same time. This software is versatile and supports many options. I have yet to investigate including a GPS connection for location information. It can also locate the Sun and Moon and point your antenna(s) in that direction so there's lots to play with in this great experimenter's project.

More GPS disciplining

Back in January 2026, we briefly discussed the GPS disciplining of oscillators and briefly touched on a device that is available on all the Far East sales sites which can be obtained at the time of writing for around £65 including delivery. My version of this device is shown in **Figure 8**. The inside of my unit is shown in **Figure 9**, and appears to use a second-hand oven-controlled crystal oscillator unit and a plug-in GPS receiver. It is also clear that the software in the unit is running on an old favourite, the 'blue pill' microcontroller board, which should contain a STM32F103C8T6 microcontroller unit (MCU). This board is available from multiple sources, but tends to suffer from being a favourite with device counterfeiters. Often you find this board supplied with a CH32F103 MCU, which will fail to program using STM software unless the 'target ID override' option is selected. This MCU board is also well known for generating switching noise at HF so you may need to add some ferrite to the DC feed to the unit.

Once up and running for a while, my unit

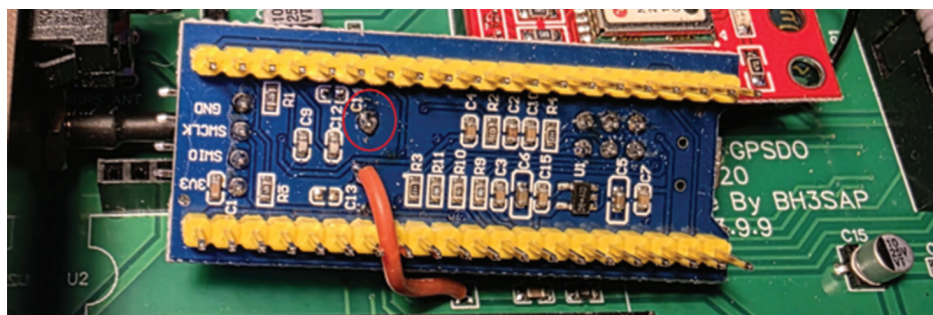


FIGURE 11: Flying lead moved from the circle to the position shown.

```

sketch_jun15a.ino
61 }
62 }
63 Serial.println("-----");
64
65 Serial.print("Address: ");
66 Serial.println(address);
67
68 Serial.print("Local name: ");
69 if (peripheral.hasLocalName()) {
70   Serial.println(peripheral.localName());
71 } else {
72   Serial.println("(none)");
73 }
74
75 Serial.print("RSSI: ");
76 Serial.print(peripheral.rssi());
77 Serial.println(" dBm");
78
79 if (peripheral.hasAdvertisedServiceUuid()) {
80   Serial.print("Service UUIDs: ");
81   for (int i = 0; i < peripheral.advertisedServiceUuidCount(); i++) {
82     Serial.print(peripheral.advertisedServiceUuid(i));
83     Serial.print(" ");
84   }
85   Serial.println();
86 }

```

Output: Serial Monitor X

Message (Enter to send message to 'Arduino NANO 33 IoT' on '/dev/ttyACM2')

New Line 9600 baud

Address: 63:81:f0:3e:7f:24
 Local name: (none)
 RSSI: -69 dBm
 Scan complete. Devices found: 15

FIGURE 12: Nano 33 IoT board BLE scan sketch and output

seemed to close in on the 10MHz target but then have stability issues where it over- or undershot the target frequency and the output was unstable.

I've done two things to my unit which have improved it dramatically. Firstly, I have replaced the firmware with the code that has been developed and is available at [6]. This can be programmed into the blue pill board from the .bin file provided in the github using the STM32 Cube programmer and the STLink programmer. This is a common route to programme these devices. The connections required between the board and the programmer are shown in Figure 10. Make sure the blue pill board isn't powered from a separate source when you do this.

By conducting some research in several online forums, there is a consensus of opinion is that there may also be a manufacturing issue with this device whereby a flying lead

from the PCB that should be connected to pin 5 of the MCU, is actually connected to pin 6. Figure 11 shows the corrected flying lead which has been removed from the 8MHz crystal connection (circled) and placed as shown. With the new firmware and the wiring modification, my unit now seems to stabilise at around 0.1 parts per billion (ppb) and maintain reasonable accuracy.

This accuracy is more than good enough for my laboratory frequency reference so this unit may be a good low-cost bodger's project to try.

The github at [6] contains some nice detail on the software operation, the menu system and the screens that have been developed. I've also used a TinyPFA [7], which is a Nano-VNA with different firmware, to compare the output frequency with my Leo Bodnar unit. They are identical to around 0.1 ppb (that's an error of just 1Hz at 10GHz). I haven't

done any long-term measurements but all in all it is a nice project for the experimenter and lower cost than some of the alternatives. You can, of course, download the source code and see exactly how this unit is operating and modify the code for your own requirements if required – it's open source!

A new kid on the block

Arduino have released a new board called the Nano 33 IoT. It's an interesting MCU board as it contains a low-power Arm® Cortex® M0 32-bit SAMD21 MCU and a NINA-W10 module from uBlox to give Wi-Fi and Bluetooth connectivity in the 2.4GHz band. It's the same size as all the other nano boards at only 45 x 18mm but provides a host of Internet of Things (IoT) functionality. As a simple test for my new board, I issued the following prompt to ChatGPT:

Acting as an Arduino coding expert, create a test sketch for the Arduino Nano 33 IoT board to scan BLE and list found devices to the serial port. Repeat the scan every 2 seconds.

This created a sketch to scan the Bluetooth low-energy (BLE) band at 2.4GHz and list devices it discovered to the serial port. The Arduino window running the sketch is shown in Figure 12. I found that the embedded firmware on my board was outdated but soon fixed this by using Tools/Firmware Updater in the Arduino IDE.

You can get this board to connect to any kind of existing Wi-Fi network, or use it to create your own Arduino access point (AP). There is a set of example sketches at [8] and a host of other information at [9].

There is also a set of examples to show how you can connect the device to the cloud and then control it from your phone by using a platform known as Blynk. This is a low-cost IoT platform that lets you connect hardware (such as Arduino, Raspberry Pi, or ESP32) to the cloud and build custom mobile or web applications to control and monitor devices remotely, without needing advanced back-end or application development skills [10].

What a great time to be a bodger...

References

- [1] blog.radioartisan.com/
- [2] groups.io/g/radioartisan
- [3] tinkercad.com/
- [4] github.com/k3ng/k3ng_rotator_controller/wiki/820-Command-Reference
- [5] vk4ghz.com/
- [6] github.com/dankar/gpsdo-fw
- [7] tinydevices.org/wiki/pmwiki.php?n=TinyPFA. Homepage
- [8] docs.arduino.cc/libraries/wifinina/
- [9] store.arduino.cc/products/arduino-nano-33-iot-with-headers
- [10] blynk.io/

Table 1: The pin connections on the G-5500 DC Yaesu control box.

Pin	Function
1	Provides a voltage corresponding to the elevation, roughly 0V to 4.5V
2	Connect to ground to rotate right (clockwise)
3	Connect to ground to rotate up
4	Connect to ground to rotate left (counter clockwise)
5	Connect to ground to rotate down
6	Provides a voltage corresponding to the azimuth, roughly 0V to 4.5V
7	Provides a poorly regulated DC voltage (drops when the motors are turning)
8	Ground