Yaesu G5500 Control from Raspberry Pi

1. This software can be used to control a Yaesu G5500 az-el rotator connected from its rear 8 pin DIN connector to a Raspberry Pi using the circuitry shown in Figure 1.

2. This software can be used as a drop-in replacement for hamlib's **rotctld**. It will listen on port 4533 for a tcp socket connection from a client application. This is the same port hamlib's rotctld uses and this program attempts to conform to the same socket protocol. The rotctld protocol is documented but contains some inconsistencies and is subject to interpretation. Thus my attempt to emulate rotctld may or may not work with your particular client application. So before building any electronics, use the tests below to check whether your preferred client application can talk to this program. The only client known for sure to work with this program is HamClock.

3. Completely separate from rotctld, this software will also listen simultaneously to port 8008 for a **web connection** from a browser or a command line program such as curl. RESTful commands can be used to interrogate and control the rotator, or a simple web page can be presented for GUI operation directly from within a browser. The possible URL commands consist of **`http://<IP>:8008/`** followed by one of the following, where <IP> is the network address of the computer running this program:

    - `get_pos`
    - `set_pos?alt=X&az=Y`
    - `move?direction=[up,down,left,right]`
    - `park`
    - `stop`
    - `get_info`
    - `dump_caps`
    - `get_cmdpos`
    - `help`
    - `index.html` (or empty)

    Entering index.html from a browser will bring up the web page shown in Figure 2. Current az and el axis positions are marked in red, commanded positions in green. Click anywhere in the graphics to set new values or enter values in the text fields and type Enter or click Set.

4. If you want to use the rotctld interface, you should run the simulator with your chosen client before building the electronics to make sure it will work. It's easy to try the simulator. Just build it and run as follows:

```
make -f Makefile_sa g5500pi
./g5500pi -s 3
```

   Leave the program running and try to connect to the host running g5500pi on port 4533 from your client and control the simulated rotator. It should present with a dual axis rotator with azimuth range 0 - 450 degrees and elevation range of 0 – 180 degrees. You can also try "-s 2" for an az-el rotator but with elevation restricted to 90 degrees; or "-s 1" for a pure az-only rotator. If your client works, you're good to proceed with building the electronics.

   You can also run the simulator this way if you just want to try out the web interface.

5. After construction of your electronics but before connecting it to the Pi the first time, you MUST perform these preliminary steps first:

   1. install the rotator as described in the Yaesu G5500 User Manual, taking care that the CCW limit points true north and the down elevation limit points level.
   2. perform the calibration of the pots labeled FULL SCALE ADJUST as per the G5500 User Manual
   3. set both pots labeled OUT VOL ADJ fully CCW.

6. When the previous preliminaries are finished, go ahead and connect everything together. Begin testing your electronics by running the g5500pi.sh script as follows:

```
./g5500pi.sh
```

This will prompt you for a direction and you can enter **cw**, **ccw**, **up**, **down** or anything else to stop. Under the hood, the script uses the following pin assignments:

Purpose	**BCM**	**Header**
AZ CW	25	22
AZ CCW	8	24
EL UP	7	26
EL DOWN	1	28

7. If these all control the rotator correctly, move on to testing the ADCs that measures axis angles. Build this test program using:

```
make -f Makefile_sa piADS1015
```

and run by entering the ADC1015 I2C bus address and channel number. The default address is 48 and the channels are assigned as follows:

Purpose	**Channel**
AZ	0
EL	1
Power Ok	2

For example, to read the raw ADC for azimuth, the command is:

```
./piADS1015 48 0
```

If piADS1015 reports no /dev/i2c-1, run "sudo raspi-config" and enable Interface -> I2C and reboot. If still nothing, run “sudo apt install i2c-tools” then run “i2cdetect 1” to check for a device responding to address 48.

Once you get piADS1015 working, the reported Az and El ADC values should range somewhere from a few tens to a few thousand. You can run this while the axis is moving and watch the number change. They increase CW for Az and Up for El.

The Power channel should read at least 1000. This is only used by the software to detect whether the G5500 is even powered on.

8. If this all checks out, your hardware is good and you know your client app will work. Run the program as a long-running daemon by typing:

```
./g5500pi &
```

9. You should now be able to connect from your client or from a web browser and control the rotator.

Don't be alarmed if the rotator starts moving all by itself. The first time the program runs it must calibrate the ADC range for each axis. It does this by rotating each axis through the full range of motion and recording the end values in a file named $HOME/.hamlib_g5500_cal.txt. From now on, those saved values will be used and the calibration step will not occur. If you want to redo the calibration for some reason, just remove that file and restart the g5500pi program. The

program will present error number -14 to the client if you try to use it while the calibration procedure is underway. If this confuses your client app, disconnect until the rotator stops moving.

10. This software is structured in such a way that it can also be used to add a bona fide G5500 backend to the real hamlib rotctl and rotctld. To do this, fork their project on github, download the project source and work through their README.developer file to insert g5500_direct.c as a new driver into their infrastructure. You'll also want to work through the same testing steps described above to qualify your new electronics. I have done this and confirmed it all does work. If someone wants to push this back up to hamlib feel free, but that someone won't be me.

    Note that by incorporating the driver into the hamlib infrastructure, there's much less concern whether the result will work with all client apps because hamlib will be performing all the socket protocol functions. But you can still play with the simulator by using a configuration parameter from rotctld as follows:

    ```
    rotctld -m 2401 –set-conf=simulator=1
    ```

    where 2401 is whatever the next rotator model ID happens to be for this driver and the number 1 is one of 1-3 as described above for the stand-alone program simulator.

11. Good luck, have fun and 73, Elwood, WB0OEW

12. Helpful resources:

    - **G5500 manual**: https://www.yaesu.com/downloadFile.cfm?FileID=8814&FileCatID=155&FileName=G%2D5500%5FIM%5FENG%5FE12901004.pdf&FileContentType=application%2Fpdf
    - **ADS1015 data sheet**: https://www.ti.com/lit/ds/symlink/ads1015.pdf
    - **Adafruit ADC ADS1015 breakout**: https://www.adafruit.com/product/1083
    - **hamlib project**: https://github.com/Hamlib/Hamlib

Yaesu G5500 Control from Raspberry Pi

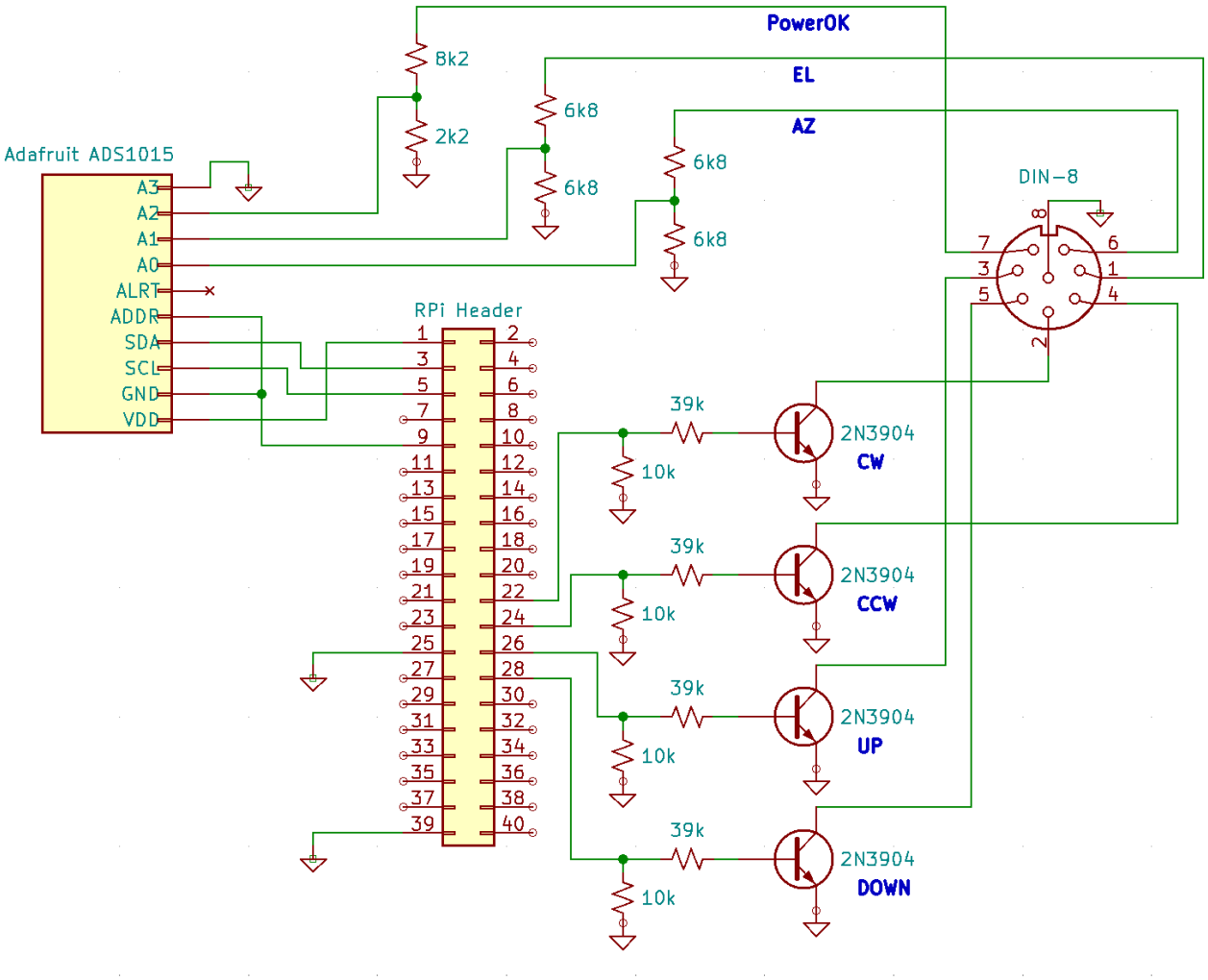

Figure 1: Circuit connecting Raspberry Pi to Yaesu G5500 controller

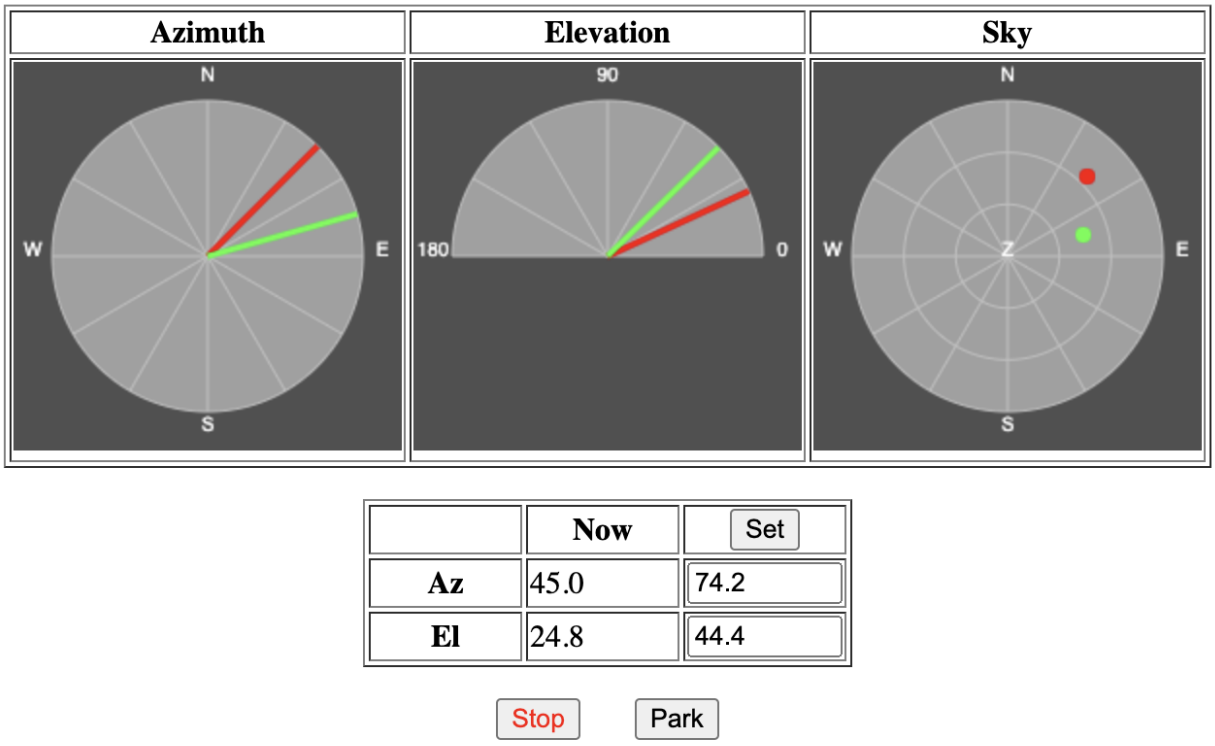

Figure 2: Screen shot of web connection to g5500pi