

Converting a G5500 AZ/EL Rotator to an RA/DEC Tracking System

Richard A. Russel

Abstract: The YEASU G5500 AZ/EL rotor is used for amateur radio satellite tracking. The interface between the computer and the G5500 is expensive. This paper shows the development of an arduino based interface for tracking astronomical sources in Right Ascension (RA) and Declination (DEC). The design includes an arduino, relay shield, and a real-time clock. The software uses the data from the real-time clock to calculate sidereal time. Based on the known location of the telescope, the RA and DEC are calculated. The RA and DEC are then converted to azimuth and elevation coordinates. The interface to the G5500 reads the AZ/EL antenna position by converting the G5500 voltages to position data. The position data is compared to the target AZ/EL . The G5500 controls the rotor movements by closing 4 lines to ground, one each for up, down, left, and right. The interface uses relays, controlled by the arduino, to close these lines. By continuously calculating the updated AZ/EL, the interface successfully tracks any given RA/DEC coordinate that is above the horizon.

Introduction

Amateur radio astronomers can utilize equipment from amateur radio systems. The Yeasu G5500 (1)azimuth/elevation rotator can be modified to point and track astronomical objects. An arduino (2) was interfaced to the G5500 to control the rotator movement. The arduino software calculates sidereal time, reads the antenna position from the G5500, and then calculates the required azimuth and elevation position required for any right ascension and declination (RA/DEC) given. The output of the arduino uses relays to cause the movement of the G5500 in azimuth and elevation, while using the USB serial interface to send the RA/DEC to the computer which displays the position on the Stellarium program.

G5500 Rotator Interface

The G5500 consists of a mast mounted rotator which is connected to a control box which is in the radio shack. (figure 1).

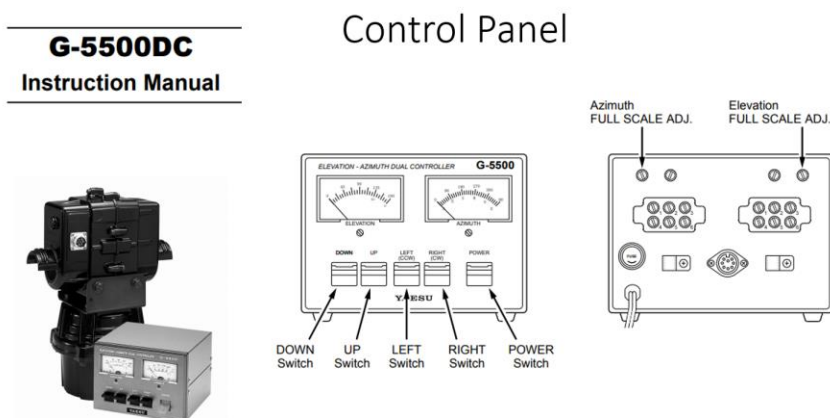


Figure 1: YEASU G5500 (1)

The interface from the control panel to the rotator is per the G5500 manual (1). The interface to the arduino utilizes the 8 – pin DIN connector on the back of the control panel. (figure 2)

GS 5500 External Control

External Control

IF the optional GS-232 Computer Control Interfaces Unit is installed, the RS-232C cable from the computer routes through this grommet, and is affixed in place by the nylon cable clamp.

Pin	Function
1	Provides 2 to 4.5 VDC corresponding to 0 ° to 180 °
2	Connect to Pin 8 to rotate right (clockwise)
3	Connect to Pin 8 to rotate up
4	Connect to Pin 8 to rotate left (counterclockwise)
5	Connect to Pin 8 to rotate down
6	Provides 2 to 4.5 VDC corresponding to 0 ° to 450 °
7	Provides DC 13 V to 8 V at up to 100 mA
8	Common ground

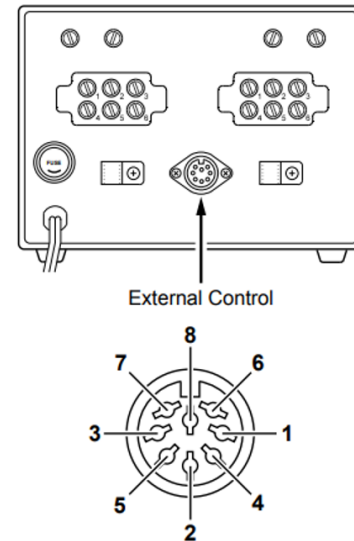


Figure 2: G5500 External Control Interface (1)

The azimuth and elevation positions are interfaced on pin 1 for elevation and pin 6 for azimuth. The voltage is measured and proportioned to the antenna position. The control functions utilize pins 2-5. Each control is activated by connecting the pin to ground (pin 8). The arduino will interface with these pins to read the antenna position and control the antenna position.

Arduino Interface to the Control Box

The arduino interface requires the reading of two voltages between 0 and 5V. The arduino UNO (2) analog input A0 and A1 was used (figure 3). A0 was linked to pin 6 for azimuth and A1 was linked to pin 1 for elevation.

Arduino UNO R3

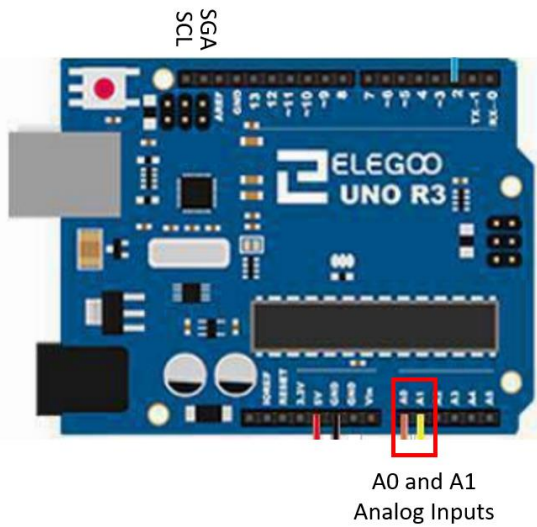


Figure 3: Arduino UNO (2)

The control interfaces utilizes an arduino relay shield. The chosen relay shield (figure 4) has four relays and is controllable by the arduino UNO.

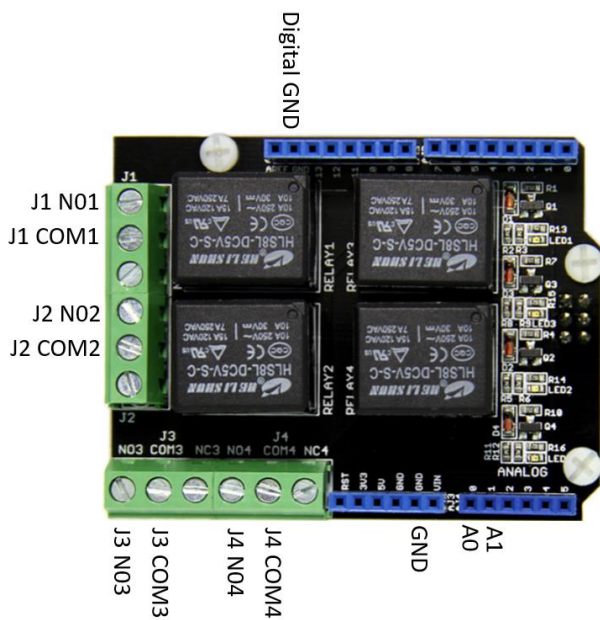


Figure 4: Relay Shield (3)

Each relay has a common connector (ex. J1 COM1) and a normally open connection (ex. J1 N01). The relay shield is designed to be connected directly to the arduino UNO.

The wire connections are shown in Table 1.

Board	Terminal	Terminal	Board
Relay Shield	J1 CM01	J2 CM02	Relay Shield
Relay Shield	J2 CM02	J3 CM03	Relay Shield
Relay Shield	J3 CM03	J4 CM04	Relay Shield
Relay Shield	J4 CM04	Analog GND	Relay Shield

Board	Terminal	Terminal	Board
Relay Shield	A0	PIN 6 Az	G5500 8 Pin DIN
Relay Shield	A1	PIN 1 Alt	G5500 8 Pin DIN
Relay Shield	J1 N01	PIN 5 Alt Down	G5500 8 Pin DIN
Relay Shield	J2 N02	PIN 3 Alt Up	G5500 8 Pin DIN
Relay Shield	J3 N03	PIN 2 Az Right	G5500 8 Pin DIN
Relay Shield	J4 N04	PIN 4 Az Left	G5500 8 Pin DIN
Relay Shield	Analog GND	Pin 8 Ground	G5500 8 Pin DIN

The last physical connection is the addition of a real time clock. This is required to calculate sidereal time. The real time clock arduino module is shown in figure 5.



Figure 5: DS 3231 RTC

This module was wired to the arduino using the following wire connections. (Table 2)

Board	Terminal	Terminal	Board
DS3231 RTC	GND	Digital GND	Relay Shield
DS3231 RTC	VCC	Analog 5V	Relay Shield
DS3231 RTC	SDA	SDA	Arduino UNO
DS3231 RTC	SCL	SCL	Arduino UNO

Software Design

The arduino software is entered using the arduino IDE (2). The requirements for this build are:

- 1) Enter the RA/DEC of the celestial object
- 2) Read the real time clock (RTC) and convert to sidereal time
- 3) Using the latitude/longitude of the antenna calculate the required azimuth and elevation needed to point at the celestial object.
- 4) Read and calibrate the antennas actual azimuth/elevation from the G5500 control box.
- 5) Compare the difference between the required and actual antenna positions
- 6) Operate relays to move the antenna to the required position while continuously reading the actual position.

Real Time Clock Interface and Sidereal Time Calculation

The RTC software was derived from the arduino website <https://www.arduino.cc/en/Reference/RTC> . It provides universal time output based on the computer clock. An optional battery allows for the RTC module to maintain time, even if power is dropped from the arduino.

The software for converting the universal time to the local sidereal time was derived from “Alex's Sidereal Code based on Adrian Jones Equation Clock” code: <http://woodsgood.ca/projects/2015/06/14/equatio-sidereal-solar-clock/> . This code takes into account the latitude and longitude of the antenna and calculates sidereal time.

Calculating Required Azimuth and Elevation for a given RA/DEC

The formulas to convert RA/DEC to azimuth and elevation were derived from: <http://www.stargazing.net/kepler/altaz.html#twig01> . The conversion starts with using the local sidereal time (LST) and the required RA to calculate the hour angle (HA) (equation 1).

$$HA = LST - RA \quad (1)$$

Note: If HA is negative, add 360 to bring in range of 0 to 360.

Calculating elevation:

$$\sin(elevation) = \sin(DEC) * \sin(LAT) + \cos(DEC) * \cos(LAT) * \cos(HA) \quad (2)$$

$$elevation = \text{asin}(\sin(elevation)) \quad (3)$$

Calculating Azimuth:

First calculate an intermediate term (A) then determine azimuth based on the value of the HA (equations (4) and (5)).

$$\cos(A) = \frac{\sin(DEC) - \sin(elev) * \sin(LAT)}{\cos(elevation) * \cos(LAT)} \quad (4)$$

$$A = \text{acos}(\cos(A)) \quad (5)$$

Use the value in equation (5) and use the following rule to calculate azimuth:

If the sin(HA) is negative, then azimuth = A, otherwise azimuth = 360-A

This formula was put into a subroutine and periodically updated to calculate the required azimuth and elevation over time.

Controlling the Antenna Rotator based on Required Azimuth and Elevation

The next step is to read the actual antenna azimuth and elevation from the G5500 controller. The arduino analog input pins were setup to read the voltage from the G5500 pin 1 (elevation) and pin 6 (azimuth). Note that it is necessary to calibrate the positions against the voltage to get the actual values in degrees.

The difference between the required azimuth and elevation against the actual azimuth and elevation was calculated. The resulting values determined if the azimuth needed to be adjusted to the right or left and the elevation needed to be adjusted to up or down.

Based on these values, the arduino activated the proper relay to connect the azimuth or elevation pins to ground on the G5500. This caused the movement of the antenna. A read cycle was conducted in between moves. To prevent hunting, the movement was stopped if the antenna was within 1% of the required value.

Stellarium Interface

Stellarium is a free open source planetarium software. (4) The arduino software emulates a telescope interface. Stellarium connects with the arduino through the serial USB interface. Arduino sends Stellarium the current RA/DEC of the celestial object. Stellarium then displays the pointing location of the antenna. (figure 6)

The value of RA/DEC was formatted based on the Meade LX200 interface specification:

<https://www.meade.com/support/LX200CommandSet.pdf>

The subroutine that performs the interface is:

```
void communication()
{
  int i = 0;
  input[i++] = Serial.read();
  delay(5);
  while ((input[i++] = Serial.read()) != '#') {
    delay(5);
  }
}
```



```

input[i] = '\0';

if (input[1] == ':' && input[2] == 'G' && input[3] == 'R' && input[4] == '#') {
  Serial.print(RA_transmit); //sends RA to Stellarium
}

if (input[1] == ':' && input[2] == 'G' && input[3] == 'D' && input[4] == '#') {
  Serial.print(DEC_transmit); // sends DEC to Stellarium
}
delay(200);
} // End Communications

```

The Stellarium interface sends out a request (GR#) for RA and (GD#) for DEC. On receipt of this input, the arduino software sends a reply using the Serial.print command.

The result is that the RA/DEC is displayed on the Stellarium celestial sphere as shown in figure 6.

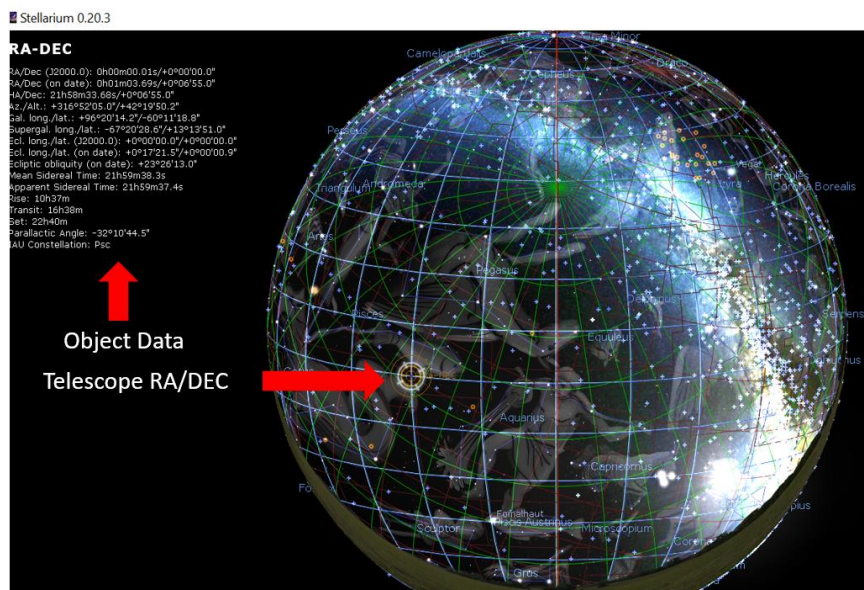


Figure 6: Stellarium Display

Summary

This project provided an excellent exercise in converting an amateur radio rotator to a telescope pointing device. The use of an arduino allowed for the calculation of the sidereal time, using the RTC module, and then the required azimuth and elevation for any RA/DEC. The relay shield provided an excellent interface to the G5500 control functions. Finally, Stellarium provided an excellent display device to show where the antenna is pointing at any time.

This project requires some basic arduino programming knowledge but is within the capability of a new programmer.

References

1. **YAESU G5500**. Retrieved from www.yaesu.com
2. **arduino**. Retrieved from www.arduino.cc
3. **Hailege 5V 4 Channel Relay Shield**. Retrieved from <http://www.hiletgo.com/Product/>
4. **Stellarium**. Retrieved from <https://stellarium.org/>



Richard A. Russel (AC0UB)

Dr. Rich Russel is the vice president for SARA and the current science lead for the Deep Space Exploration Society. He is a retired Northrop Grumman Senior Systems Engineer and served as the Chief Architect for the Satellite Control Network Contract (SCNC). In this capacity he was charged with planning the future architecture of the Air Force Satellite Control Network (AFSCN) and extending the vision to the Integrated Satellite Control Network (ISCN). Dr. Russel has been the lead architect and integrator for the Space-Based Blue Force Tracking project for U.S Space Command, the Center for Y2K Strategic Stability, and CUBEL Peterson. Dr. Russel also has led the SPAWAR Factory team in the deployment of the UHF Follow-On Satellite system. He has a Doctorate in Computer Science, an Engineers Degree in Aeronautics and Astronautics, a Master's in Astronautical Engineering, and a Bachelor's in Electrical Engineering. He is also certified as a Navy Nuclear Engineer and he is a retired Navy nuclear fast attack submariner and Navy Space Systems Engineer.