

An Arduino-Based Antenna Rotator Controller

A lightweight rotator is updated with a simple digital controller.



Giovanni Carboni, IZ5PQT

Recently, I purchased a lightweight antenna rotator of the type used for TV antennas. It is a “New Old Stock” Hirschmann Hit Ro 250 (see Figure 1). I thought that I could use it to orient a simple VHF Yagi or the small untuned loop (see the lead photo) that I use for MF and HF reception. These rotators usually rely on the constant rotation speed of the driving motor to orient the antenna to a given direction. To periodically calibrate the angular position, these rotators have a mechanical stop at 0 degrees and at 360 degrees — usually corresponding to north when the antenna is installed.

I decided to build a new completely electronic unit based on the Arduino microcontroller. The new controller replaces the old one; it is totally silent, compact, and allows you to preset and read the position on a backlit LCD. The rotator position is stored in the Arduino EEPROM and is automatically recalled when the unit is switched on.

The Rotator

The operating principle is shown with reference to the dashed section of the Figure 2 schematic. The motor has a center-tapped stator winding. Power is applied between the center tap, line 1 on the three-pin terminal strip, and either side of the windings (line 2 or 3), depending on the rotation direction. A non-polarized electrolytic capacitor C3 connected across lines 2 and 3 is required for motor startup. To calibrate the controller, the rotator is initially rotated counter-clockwise (CCW) until the 0° end stop, when the motor stalls. Assume that ac power is applied between terminals 1 and 2 to get CCW rotation. Then when the motor stalls, the current I3 drops from about 0.85 A to 0.65 A. This current jump is detected by a dedicated sensor U2, so that the Arduino can determine the zero position.

The rotator position is stored in the Arduino EEPROM and is automatically recalled when the unit is switched on.

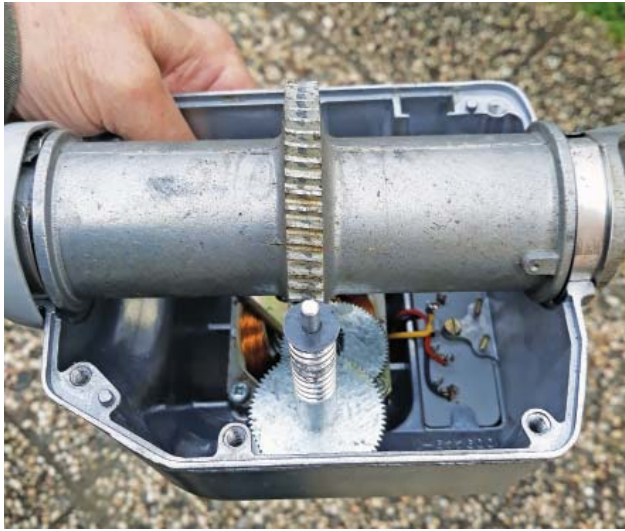


Figure 1 — An inside view of the rotator.

Controls

On the front panel, there is a 16 × 2 backlit LCD, the power switch, and two momentary contact center-off SPDT toggle switches to operate the controller (see Figure 3). When the unit is powered up, it recalls the rotator direction from the internal memory and shows it on the display. Then the left toggle switch is used to preset the position (up or down) in degrees. If you go beyond the desired position, you can correct it by hitting the **UP** or **DOWN** switch again, as many times as you wish, but the motor will not move yet. The starting position and the positive or negative displacement Δ required to move the rotator to the programmed direction are shown on the display in Figure 3. When the second switch is toggled up the controller activates the motor in the proper direction (CW or CCW) for a time T_{rot} proportional to Δ . The final position is shown on the display and written to memory. If the second switch is toggled down the rotator performs a calibration going back to zero. The display in this case shows the current reading continuously, which will show the aforementioned drop when the motor is stalled. The calibration is also performed automatically when the unit is switched on for the first time or if the memorized direction is zero.

Current Measurement

U2, an Allegro ACS712 5 ampere version, is used for current measurements. I use a breakout board. The ac output of the current sensor is sent to the U3 op-amp precision rectifier circuit (https://en.wikipedia.org/wiki/Precision_rectifier). The resulting dc output is sent to the Arduino.

Arduino Nano

I use the Arduino Nano microcontroller board. When planning the position of the various components, leave enough space to allow plugging the USB cable into the Arduino board.

The Circuit

The complete schematic diagram is shown in Figure 2. An 18 V ac center-tapped transformer powers both the motor and the electronics. It replaces the original 16 – 17 V transformer. A full-wave rectifier (D1, D2) followed by a 7805 voltage regulator provides the voltage for all the electronics.

The controller fits inside a metallic cabinet approximately 4.8 (W) by 4 (H) by 5 (D) inches (see Figure 4). I mounted the power supply and the Arduino board on a 2 × 4.8 inch piece of perforated copper epoxy motherboard behind the front panel.

The Arduino board socket is made from two female header strips (15 contacts). A break-away 8 + 8 pin male strip mounted on the board is used to connect the LCD. Connections to the toggle switches, the relays, and the current sensor are also routed via short male pin strips, three contacts for each toggle switch, one or two for the rest. The motherboard also accommodates the LCD contrast trimmer R6. This should be set once and forgotten. The relays, the ACS712 sensor, and the precision rectifier are mounted on a separate Bakelite board and all connections to the motherboard are made with female-female jumper wires.

The 16 × 2 LCD module is mounted with four screws onto the front panel. It has 16 external connections, and I strongly suggest that you first solder a row of 16 male pins into them, and then use female-female short jumper wires (12 needed) to connect it to the motherboard.

For the two relays, RYA and RYB, I used two small single relay modules (seen in the lower left of Figure 4) intended for use with the Arduino. The modules incorporate a driver transistor, a protection diode, and LED. Capacitors C1 and C2 across the contacts are crucial to suppress transients. The U3 based precision rectifier is mounted on a small piece of Veroboard.

These rotators usually rely on the constant rotation speed of the driving motor to orient the antenna to a given direction.

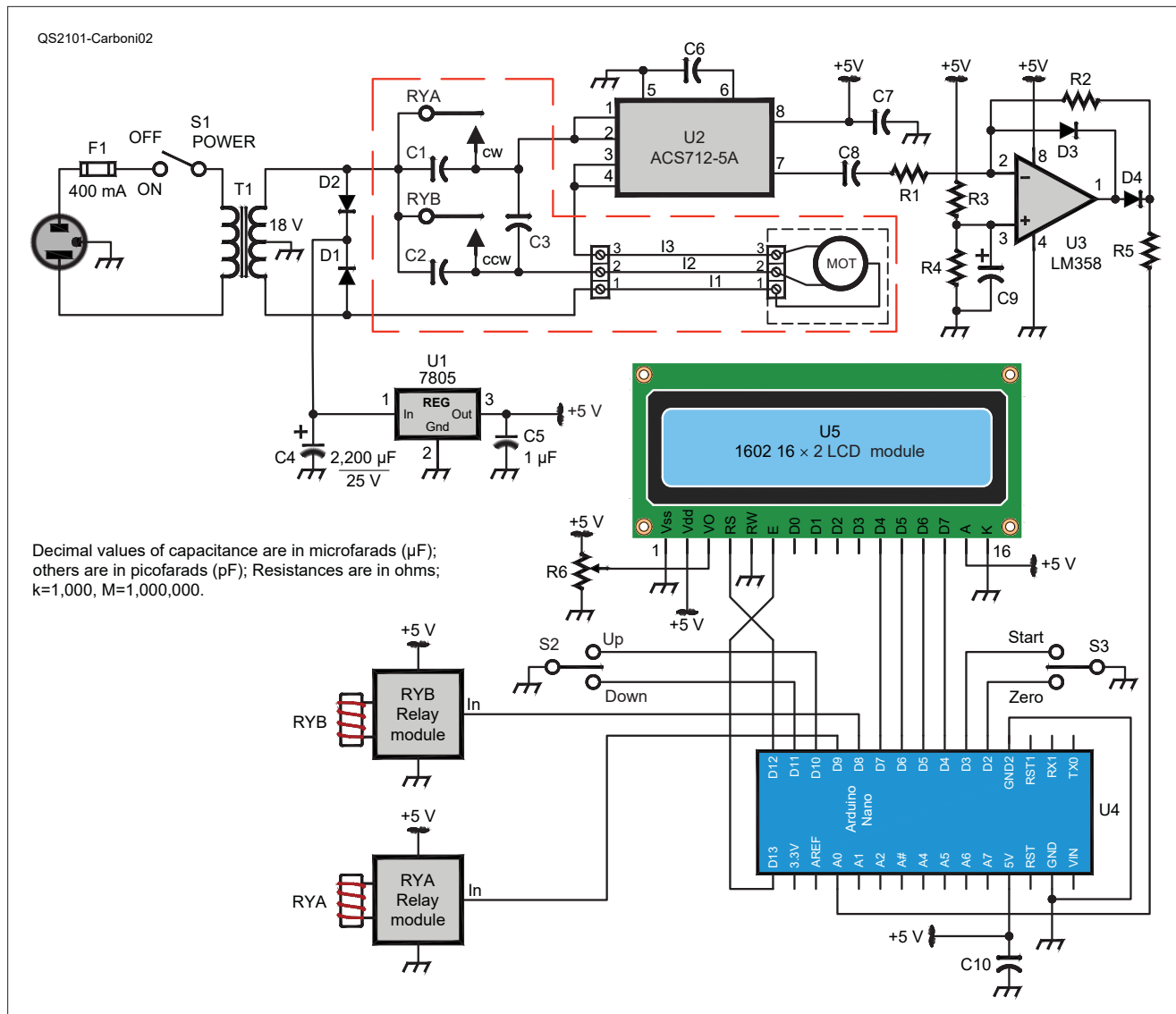


Figure 2 — The schematic of the controller. Non-electrolytic capacitors should be plastic film or ceramic.

C1, C2 — 330 nF
 C3 — 120 µF, 63 V, non-polarized electrolytic (you can use two 220 µF electrolytics wired back-to-back)
 C4 — 2,200 µF, 25 V, electrolytic
 C5 — 1 µF
 C6 — 1 nF
 C7, C10 — 100 nF
 C8 — 2.2 µF
 C9 — 10 µF electrolytic
 D1, D2 — 1N4007 diode
 D3, D4 — 1N4148 diode
 F1 — Fuse 400 mA
 R1, R3, R5 — 10 kΩ, ¼ W
 R2 — 33 kΩ, ¼ W
 R4 — 4.7 kΩ, ¼ W

R6 — 5 kΩ trimmer
 RYA, RYB — 5 V SPDT relay modules for Arduino or equivalent, <https://vetco.net/products/single-relay-module-for-arduino-d25/vupn5949>
 S1 — Power switch
 S2, S3 — Momentary contact center-off SPDT toggle switch
 T1 — Power transformer, 18 V center-tapped 1 A secondary, primary adapted to line voltage
 U1 — 7805 5 V regulator
 U2 — ACS712 05A (5A version) current sensor IC or breakout board; <https://www.banggood.com/buy/low-current-sensor-breakout-ac712.html>
 U3 — LM358 IC
 U4 — Arduino Nano Board
 U5 — LCD 1602 16 × 2 alphanumeric module



Figure 3 — The front-panel layout. The power switch is at the bottom right. The toggle switch on the left programs the direction (up or down). The toggle switch on the right starts the motor (up) or zeroes the rotator (down).

The gain is set by $R2/R1$. The quiescent output voltage after $R5$ should be 1.6 V. Don't substitute a 741 op-amp for the LM358.

A word of caution: don't place the ACS712 current sensor too close to the power supply transformer, because its stray magnetic field will be detected by the Hall probe and will appear as noise on the output.

Setup and Software

Once set up, no computer connection is needed to operate the controller. There are two sketches on the www.arrrl.org/qst-in-depth web page. The first, *myrotor_setup.ino*, will assist you in setting up the controller, and then you can forget it. The second, *myrotor_1.2.ino*, is the actual operating sketch.

Note that if Arduino Nano is not recognized when you plug the USB cable into your PC, you should probably install the relevant drivers.

For setup, load *myrotor_setup.ino* using the Arduino IDE. Once the sketch is loaded, you will see the welcome screen. Then the controller waits for you to send a command. Toggling the left switch up and down will turn the rotator CW/CCW. If the direction is wrong, swap wires 2 and 3 on the terminal board. If everything is okay, toggle the switch down. The rotator will rotate CCW toward zero, continuously reading current, I , and time, T . Take a note of the current, I , reading. When the rotator hits the end stop, the current will drop. Note it, then compute the average of the readings before and after the stop. This determines the parameter *Istall* to be inserted in the operating sketch.

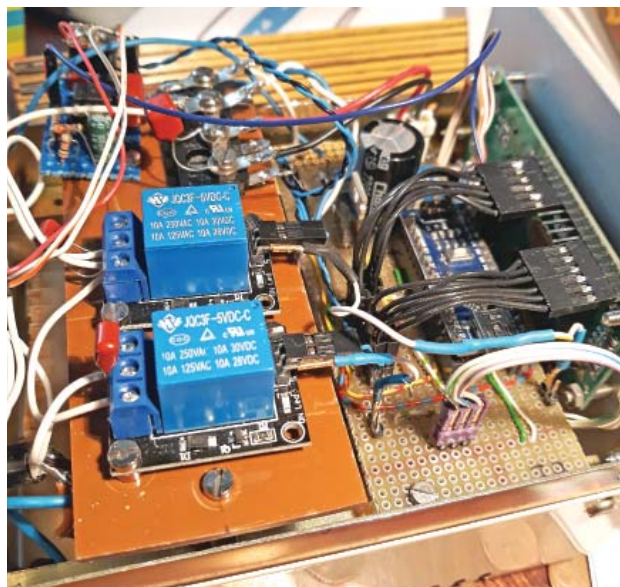


Figure 4 — Internal view shows the relay modules mounted on a Bakelite strip. All interconnections to modules and switches use jumper wires and pin connectors (no soldering). The three-conductor flat cable on the lower right goes to the toggle switch. The power transformer (not visible) is below the Bakelite strip.

Finally, with the rotator at 0 degrees, toggle up the left switch until the rotator reaches the 360-degree stop, and make a note of the time. Multiply it by 1,000. This gives you the second parameter *TT*.

Open *myrotor_1.2.ino* in the Arduino IDE, and edit the two parameters, *Istall* and *TT*, inserting the numbers you obtained into the sketch. Load the sketch. The controller will bring the rotator to zero and will automatically stop. Then it will be ready to operate, and you can disconnect the USB cable. The relationship between the Arduino readings and the motor current is explained on the www.arrrl.org/qst-in-depth web page.

Giovanni Carboni, IZ5PQT, first discovered ham radio by reading the 1958 *Radio Amateur's Handbook*, but study, family, and work kept him from getting licensed until 2009. He graduated with a degree in physics from University of Pisa, and subsequently conducted experimental high-energy physics research at the European CERN Laboratory in Geneva, Switzerland. From 1995 to 2016, he was Professor of Physics in Tor Vergata University in Rome. Now retired, he lives in Pisa, where he enjoys ham radio digital modes, VHF meteor scatter, experimenting with antennas, and a wide range of homebrew projects. You can reach Giovanni at iz5pqt@gmail.com.

For updates to this article, see the QST Feedback page at www.arrrl.org/feedback.

