Home brew Yeasu Rotor Controller

Background:

I have operated for many years as a microwave rover with one or more dishes mounted on the rear of my vehicle.

In 2008 I built a rotor controller based on the MSP430 micro-controller as part of a course taught by Tommy Sullivan W1AUV. I have used the MSP430 controller successfully for several years but my MSP430 design used a device which only has a single serial port and I wanted two ports, one to interface with Roverlog, and another to interface to a display module.

This paper describes an Arduino based controller which replaces the MSP430 controller.

Arduino is an "open hardware" project which supports several micro-controller boards, add on boards for I/O and has an extensive library of I/O functions.

You program the arduino in the C language using a simple Integrated Development Environment. Which includes a serial monitor which is used for debugging. The ardino environment is programmed in Java and can be run on a Windows machine, a Linux box, or an Apple computer.

You can learn more about the arduino project by going to:

*www.**arduino**.cc*

In order to "remember " what I want to do, I first write down my "requirements / goals" which are as follows:

Home brew Yeasu Rotor Controller

Requirements:

1. The micro-controller shall interface with the Roverlog program running in a personal computer via a USB virtual com port.
2. The micro-controller shall display the measured rotor position in degrees on a dedicated display device that is readable in daylight and nightlight.
3. The micro-controller shall receive angle position commands from the PC as an ASCII string in the format: Axxx [CR]

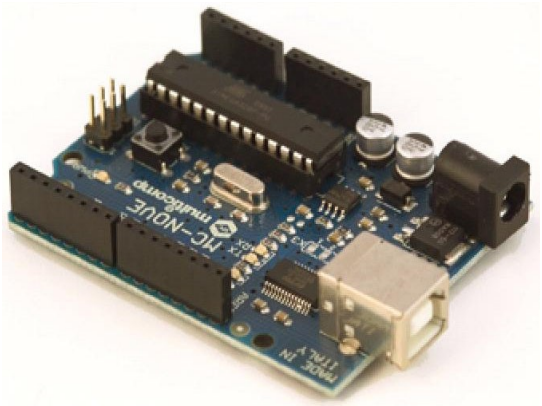
   where xxx is an angle between 0 degrees and to 400 degrees, and CR is the ASCII symbol for carriage return (0x0d). Leading zeros are suppressed.

   Example: 90 degrees is commanded as ASCII characters: 41 39 30 0d
4. The micro-controller shall respond to Roverlog "bump" commands by modifying the commanded position by 3 degrees. A CW bump command is the ASCII character "+", a CCW bump command is the ASCII character "-".
5. The micro-controller shall interface with a Yeasu G-800SA rotor to obtain position data.
6. The micro-controller shall report measured position to the PC in a format compatable[1] with Roverlog. Position reports will be formatted as A=xxx.x[CR] degrees.
7. "Manual control" shall be supported for trimming and alignment to benchmarks.
8. The controller design shall support a system positioning accuracy requirement of $\pm$ 2 degrees.
9. Power shall be derived from 14 VDC vehicle power.
10. The controller shall provide a control to automatically "park" the rotor at 180 degrees.

Note: 1. Roverlog will be setup for use with a **RC2800DC- rotor**.

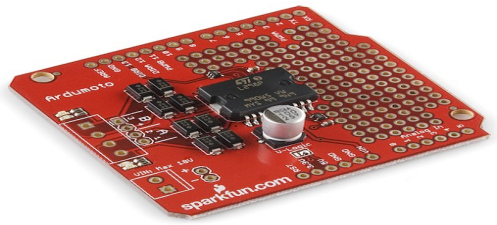
The major components for my project are:

1. Arduino™ Duemilanove Compatible Development Board

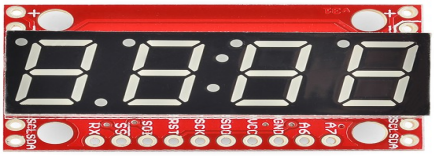


Home brew Yeasu Rotor Controller

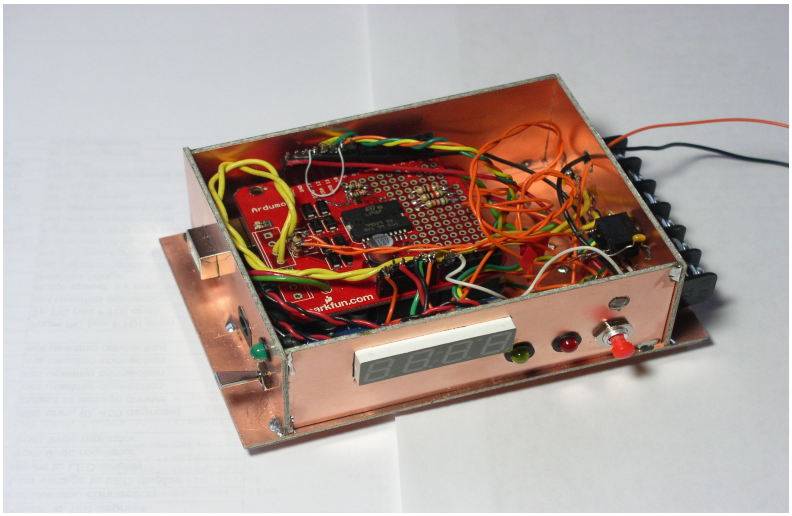
2.Ardumoto - Motor Driver Shield



3 Yellow 7 Segment display



I installed my project in a home brew PCB box



KJ1K/Rover 2/08/2013

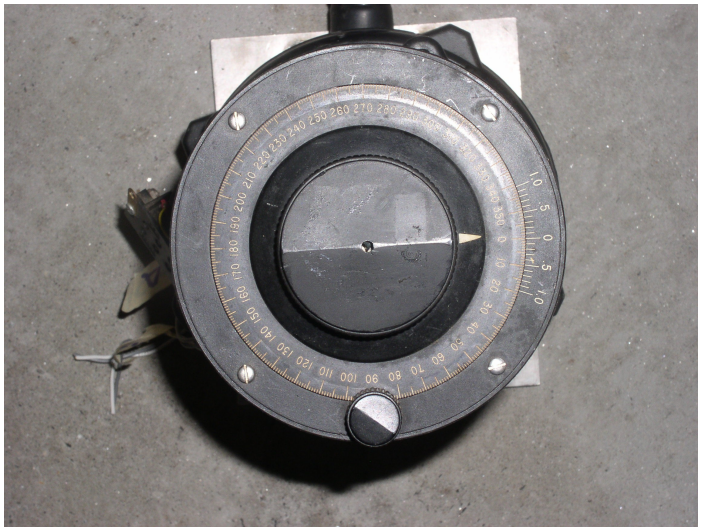
Home brew Yeasu Rotor Controller

Discussion/Performance:

I had to make quite a few cuts and jumpers on the Ardumoto Motor Driver Shield in order to make it work. If I had to do it again I would not use that PCB.

I had a difficult time obtaining the desired angle accuracy. Initially I used a linear mapping of the 10 bit analog to digital converter output to mechanical angle. The angle errors were as large as 10 degrees and did not seem to be well behaved (I tried curve fitting the data).

Tommy Sullivan suggested that I use a look up table instead and that is what I eventually did. In order to get the data required to build my lookup table I used a precision angle readout in conjunction with a laser level that I held against a filing cabinet. The angle readout is shown below:



Additionally I had a problem putting the table into my code. Not being an experienced programmer I mistakenly thought that if I declared an array as a "const int" that the array would be stored in flash, but that is not true for the arduino implementation. Instead I had to use the "PROGMEM" statement when I declared my array and then use peculiar statement syntax when retrieving the data. The process is well described at:

http://arduino.cc/en/Reference/PROGMEM

I believe the rotor accuracy after calibration is about ± 2 degrees.

Program Code:

The program code is available for anyone that wants to use/improve it.

KJ1K/Rover 2/08/2013