Using Microcontrollers in Amateur Radio, an AZ EL Controller Application

A Presentation For The Southwest Ohio Digital Symposium

Presented by Bill Erwin – N9CX January 10, 2009

Things I Hope To Leave You With

- Share my experience with the rotor controller project
  - Explain why I made the choices I did
  - How it works
  - Status of the project

- But more than that:
  - Why a microcontroller was a good choice for this project
  - What software development environments are all about
  - Tempt you to consider experimenting with microcontrollers

Feel free to ask questions at any time !

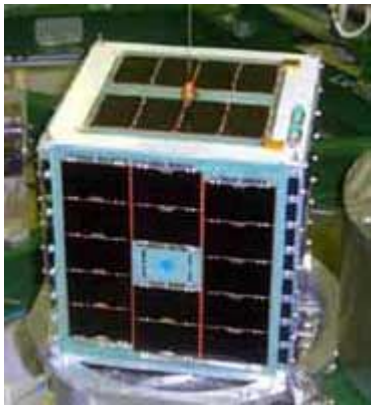
Motivation For This Project

- Developed an interest in LEO (Low Earth Orbit) satellites
  - Led to an interest in a better antenna system
  - Wanted to track LEOs with small beam antennas
- Commercial rotors & controllers are available
  - Didn't want to commit that much money at this stage of interest
- Decided to use inexpensive rotors & build my own controller

Low Earth Orbit Satellites

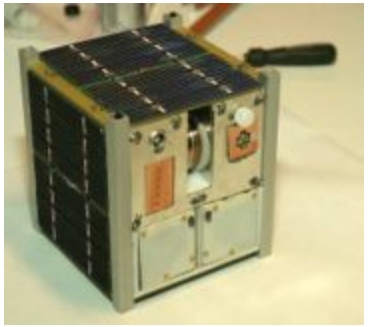
- Basically LEOs are orbital repeaters
- AMSAT has a lot of information on the WEB
- LEOs offer some special challenges
  - They move fast.
  - Short contacts
  - Low RF power

LEO Satellites Vary In Both Size & Complexity



AO-51 (Echo)
~800 km orbit
voice repeater
PakSat BBS
PSK31 Digital

SuitSat-1 (AO-54)
Russian space suit
Launched from ARISS ~355 km
telemetry only
temp & battery



N-Cube2
10x10x10 CM
I LITER VOLUME
(University Projects)

~690 km orbit

Satellite QSOs Are Interesting!

- There are a lot of “things” involved in working the LEO satellites!
  - Computer screen
  - Keyboard
  - Mouse
  - Downlink frequency
  - Uplink frequency
  - Doppler effects
  - Code paddles or a microphone
  - Azimuth of the satellite
  - Elevation of the satellite

So Many things – So Little Time!

- The window for a QSO is often less than 8 minutes.
- If you can automate a few “things”, your QSOs may have more “talk” time.
- This project is about automating the rotors for directional azimuth & elevation antennas.

My Approach To The Project

- Research the WEB for similar projects
- Evaluate what I might do that is different
- Understand how rotors work
- Keep it (relatively) cheap
- Breadboard parts of the design to verify critical assumptions

- Rotor controller needs an LCD display & flashing LEDs!

Why Use A Microcontroller Anyway?

Choices To Make

- Features
- Rotors
- Software Development tools & Environment
- Microcontroller

Desirable Features

- Work with the Nova tracking software
- Have 2 main modes: “manual” & “autotrack”
- Self-calibrate to any Pulser type rotor
- Remember antenna position during powerdown.
- Reliable beam positioning – within 5 degrees.
- Easy to update the controller software.
- Minimize cost

The Rotor – You must understand the thing you are trying to control!

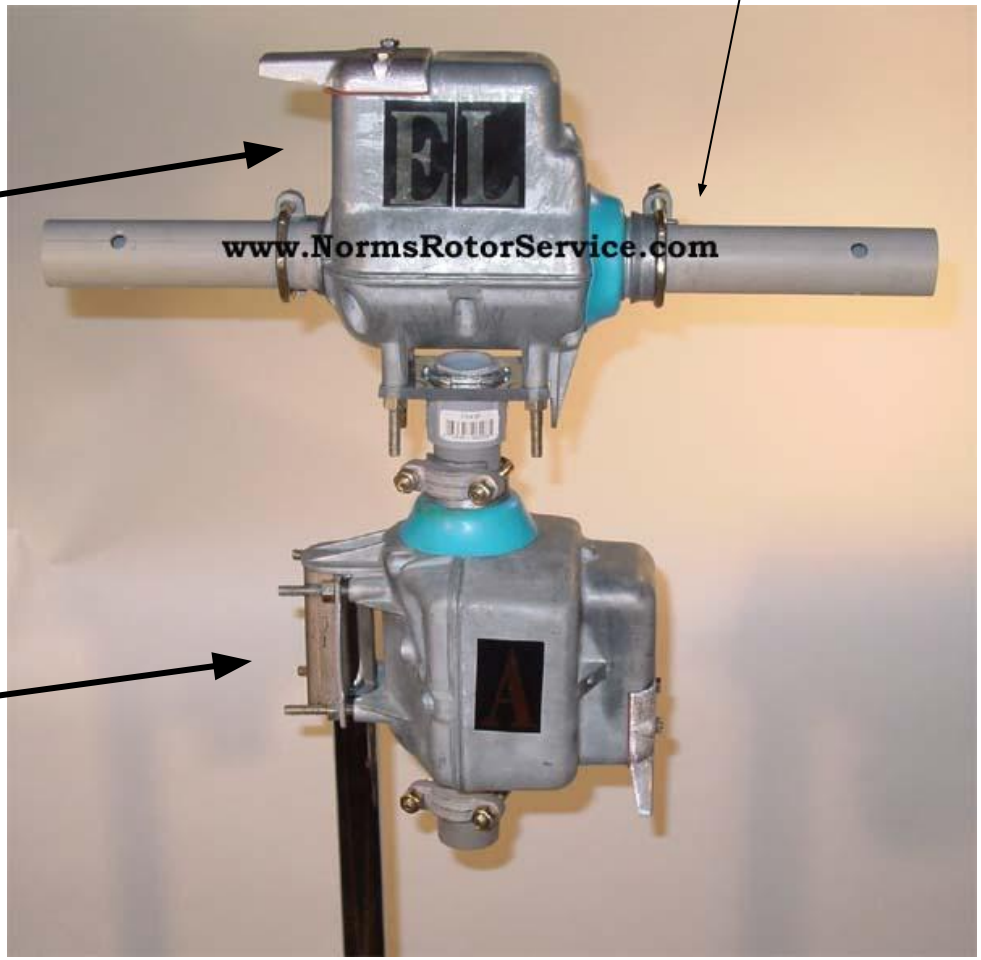
The Alliance U100

Yes – You Can Stack Them

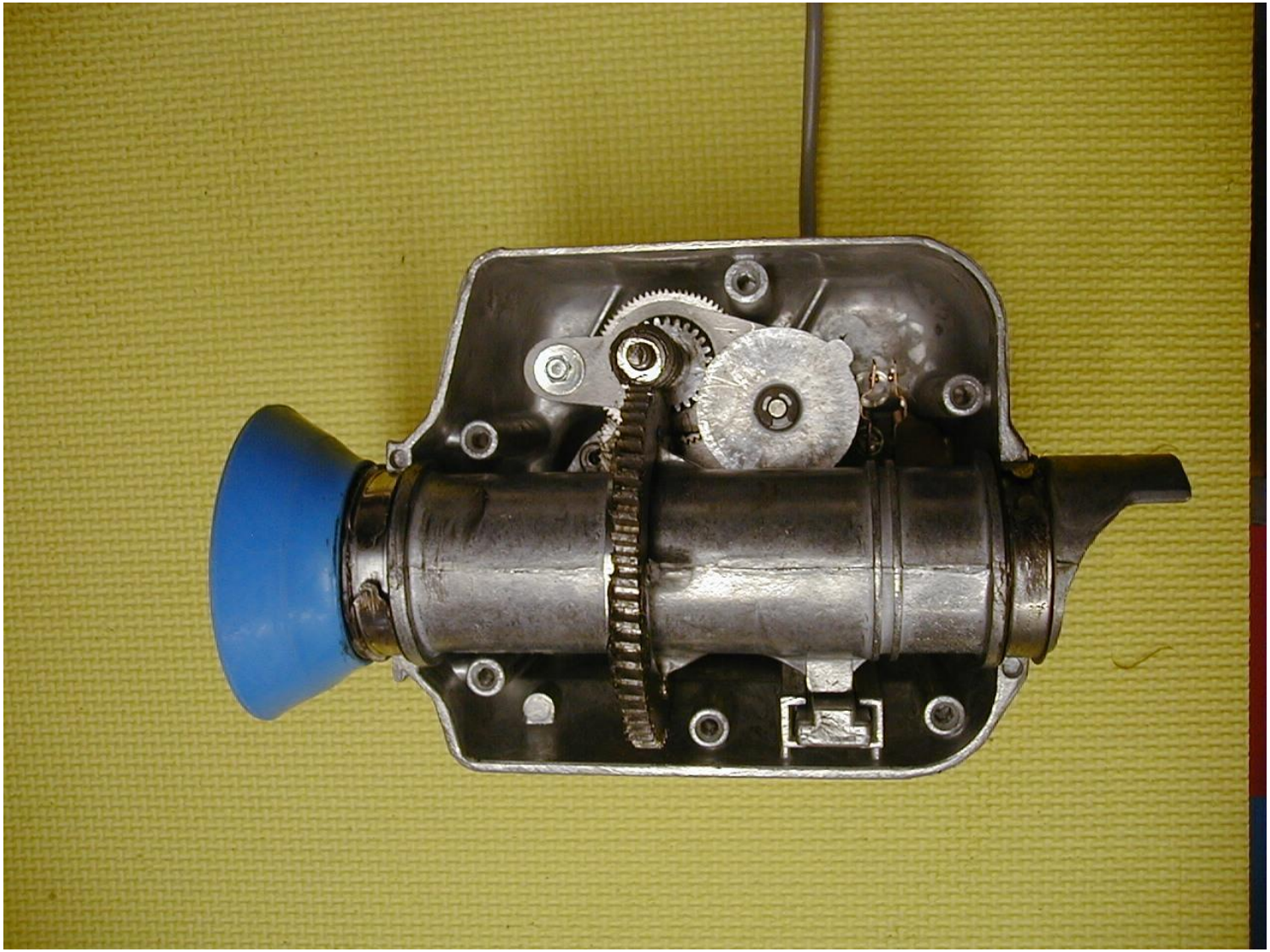
The ability to put a pipe through the rotor body is fairly unique.

Elevation

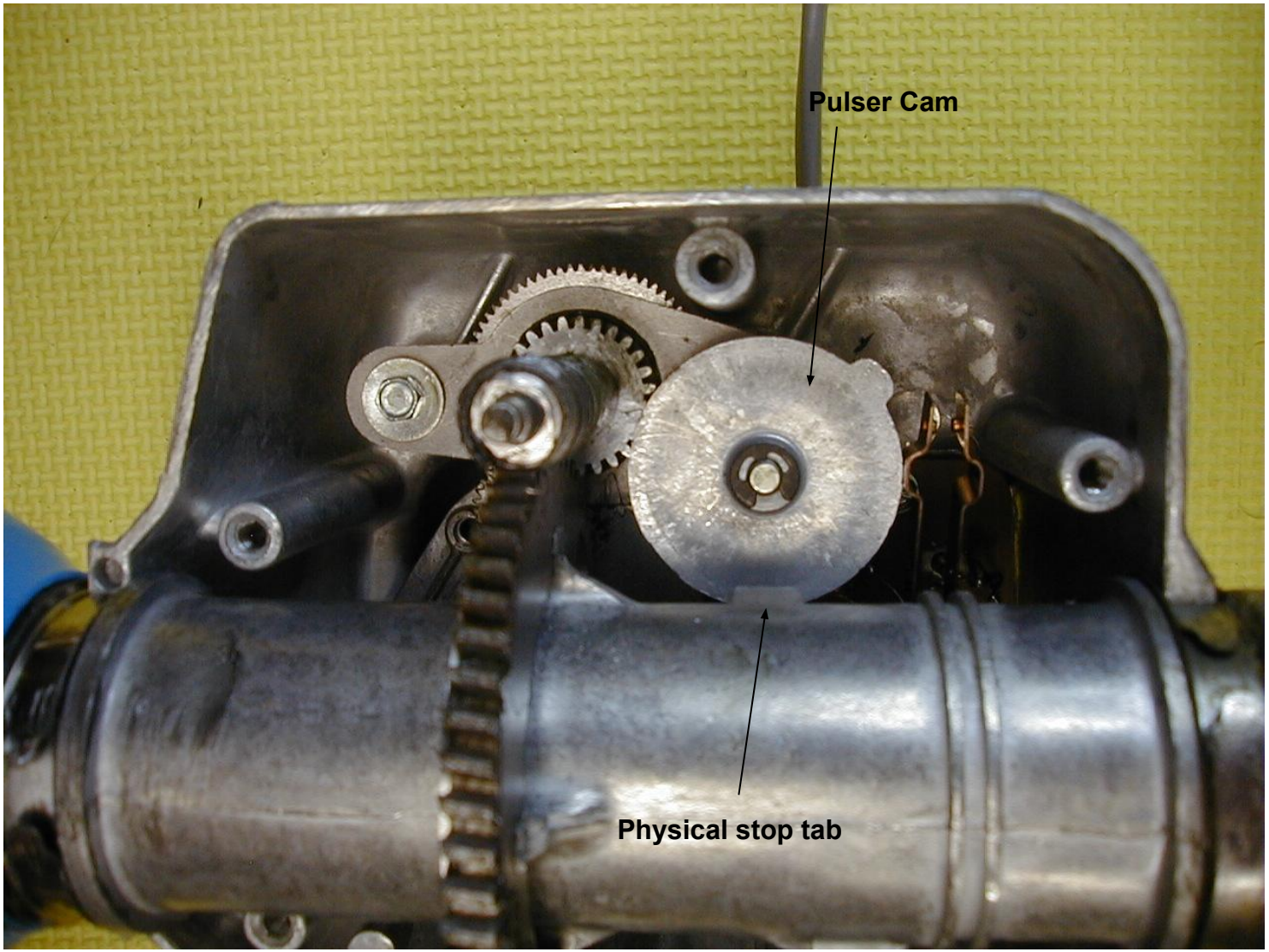
Azimuth



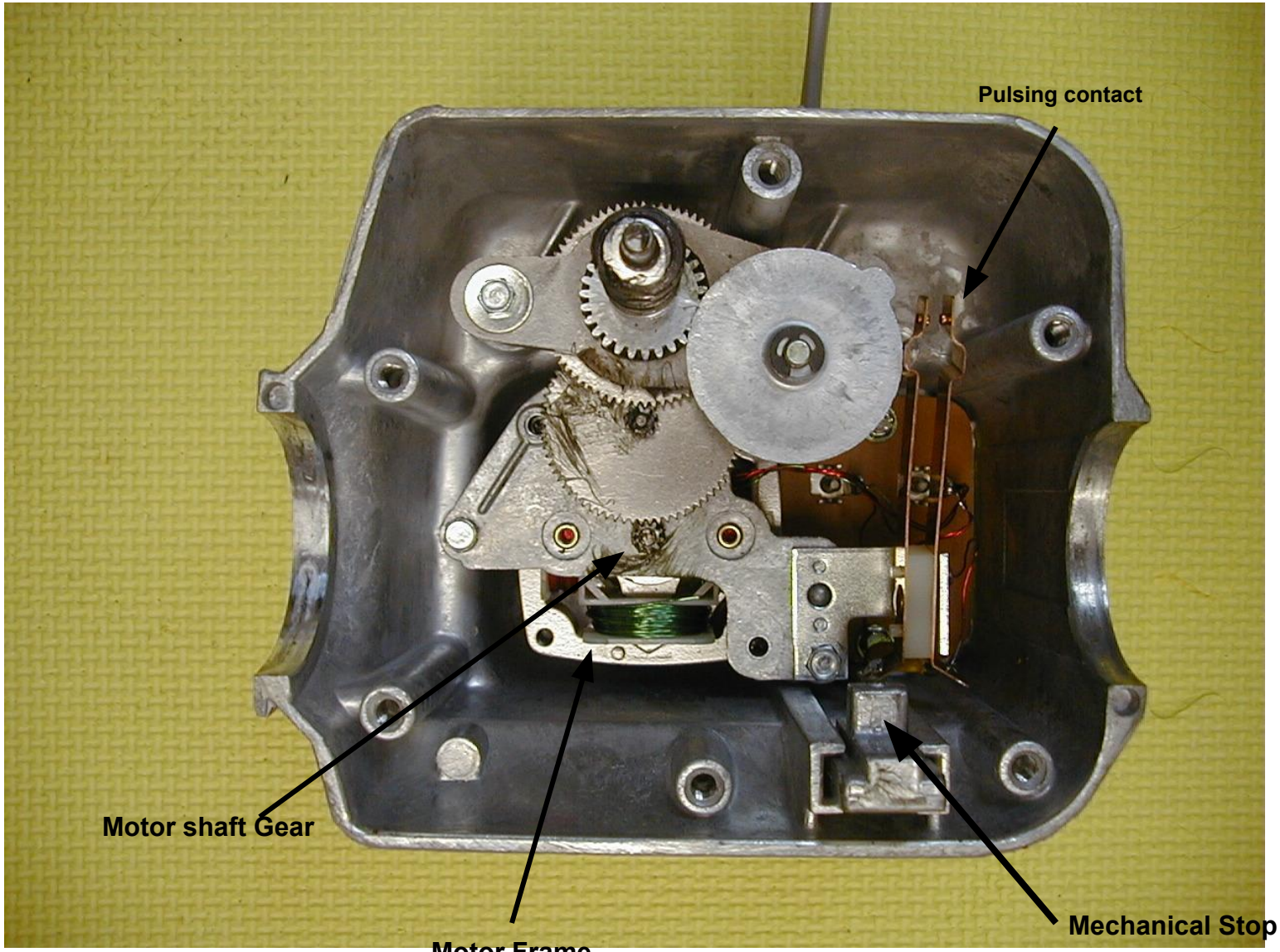
Anatomy Of A U100 Rotor #2



Anatomy Of A U100 Rotor #3

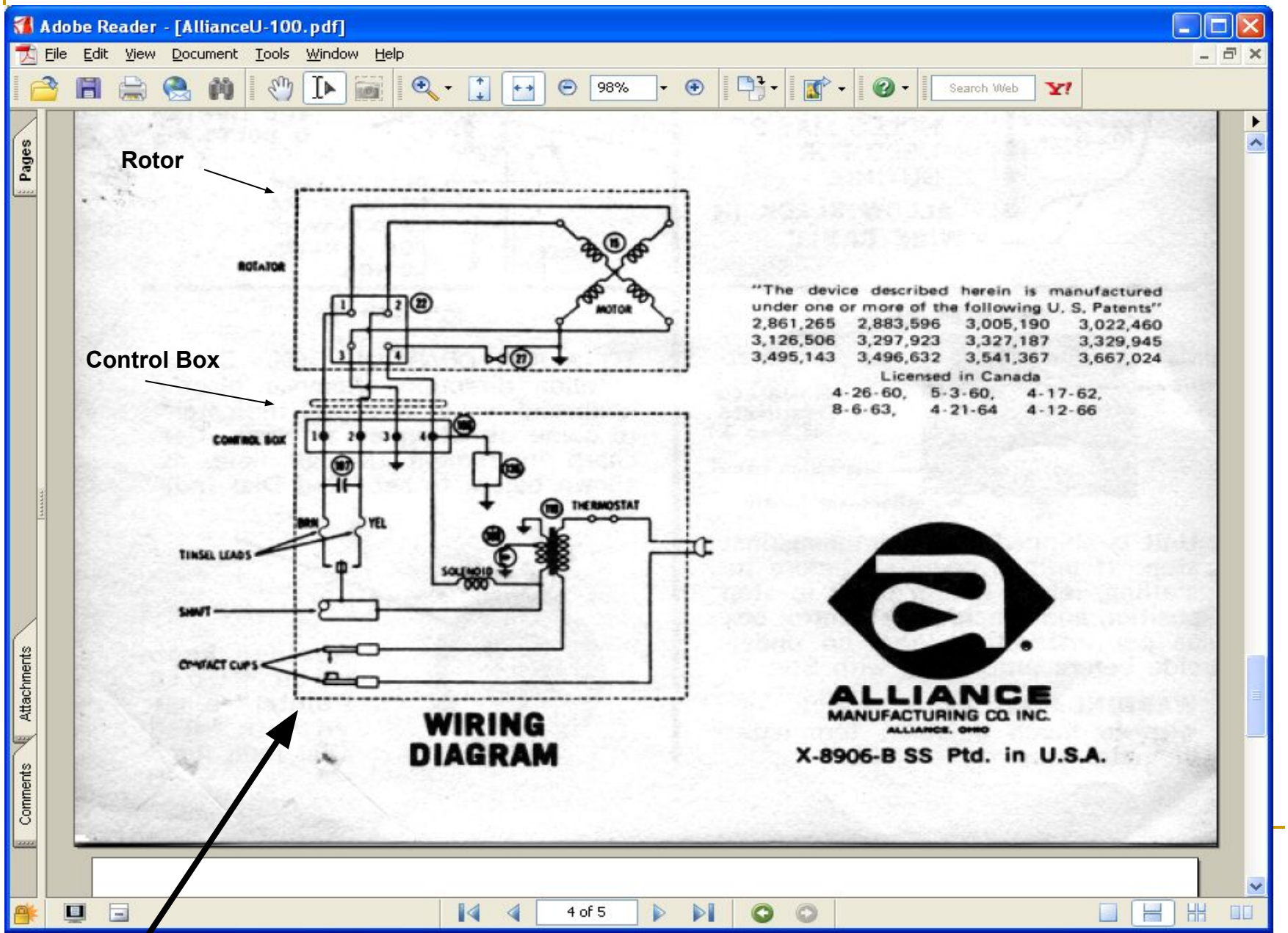


Anatomy Of A U100 Rotor #4



Commercial Controller for the U100 Rotor

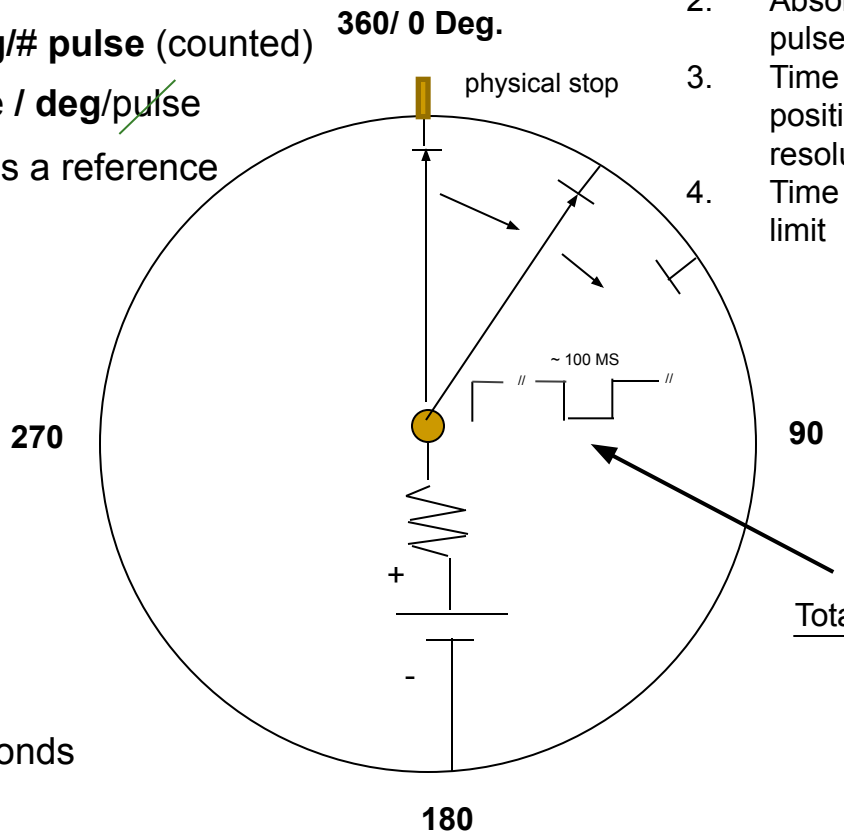
The Original U100 Rotor Schematic Diagram



Model of the U100 Rotor

Calibration Mode calculates:

1. deg/pulse = 360 **Deg/# pulse** (counted)
2. tics/deg = **tics**/pulse **/ deg**/pulse
3. Use physical stops as a reference



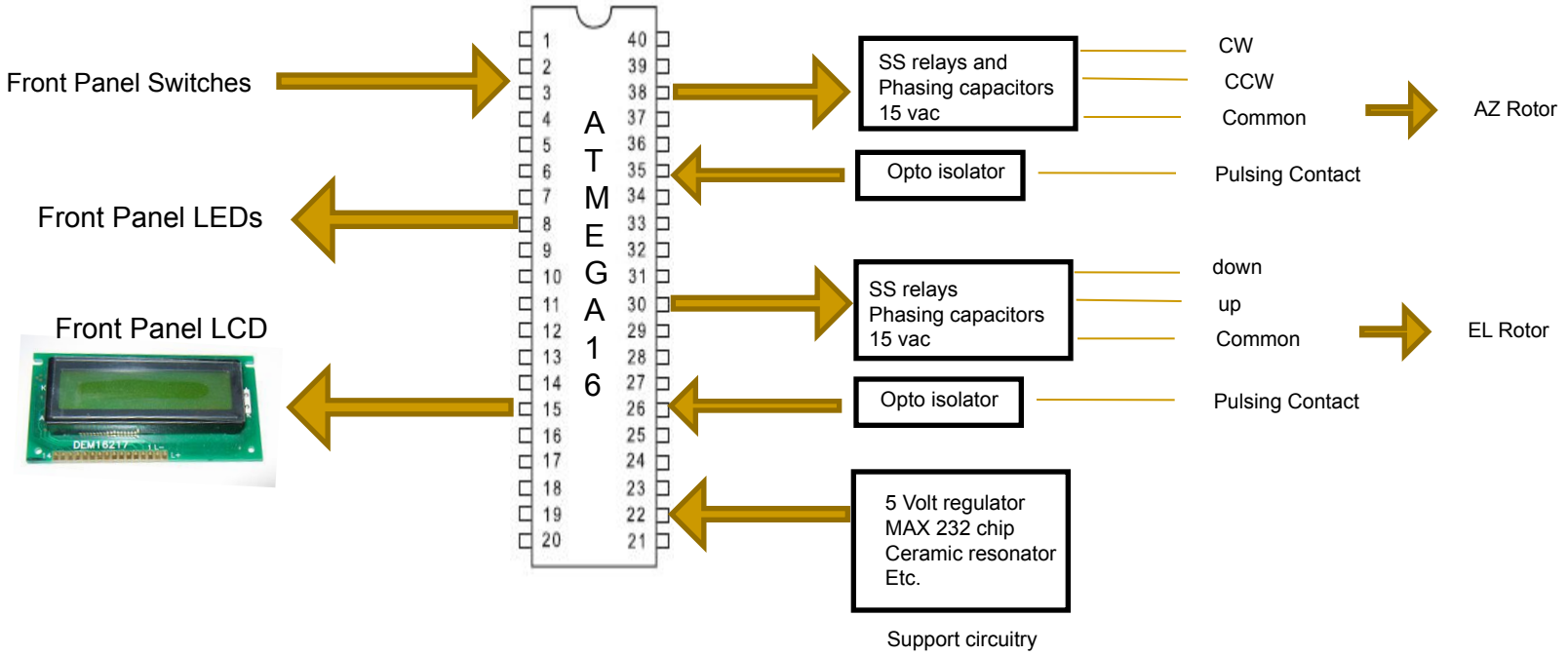
Total feedback from the rotor

Strategy

1. Do an initial calibration to detect rotor's pulse characteristics.
2. Absolute direction is known at each pulse & at rotor physical stops.
3. Time between pulses to estimate position of rotor to a finer degree of resolution.
4. Time between pulses to detect rotor limit or problems.

Note – A "tic" is 5 milliseconds

Block Diagram Of the Rotor Controller

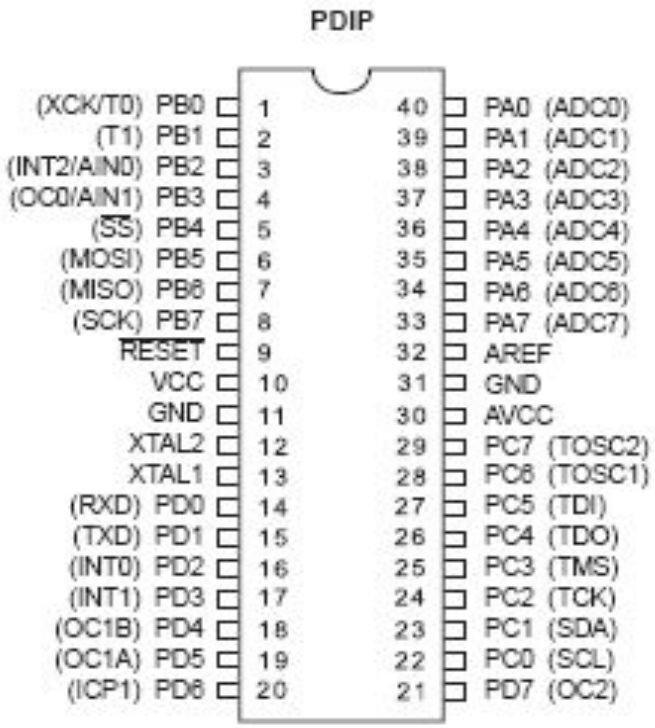


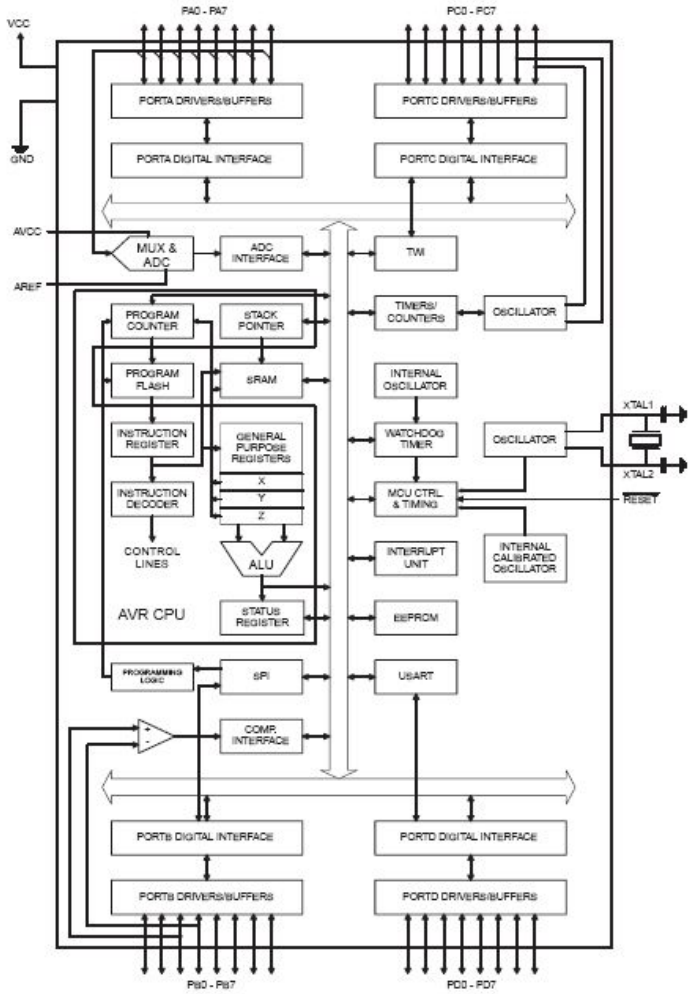
Note - This diagram does not indicate pin assignments

A FEW OF THE ATMEGA 16 FEATURES

- **THE DATA SHEET IS 358 PAGES !**
- **– 32 x 8 General Purpose Working Registers**
- **– Up to 16 MIPS Throughput at 16 MHz**
- **– 16K Bytes of In-System Self-programmable Flash program memory**
- **– 512 Bytes EEPROM**
- **– 1K Byte Internal SRAM**
- **– Two 8-bit Timer/Counters with Prescalers**
- **– One 16-bit Timer/Counter with Prescaler**
- **– Real Time Counter with Separate Oscillator**
- **– Four PWM Channels**
- **– 8-channel, 10-bit ADC**
- **– Byte-oriented Two-wire Serial Interface**
- **– Programmable Serial USART**
- **– Master/Slave SPI Interface**
- **– 32 Programmable I/O Lines**
- **YOU CAN NOT USE ALL AT SAME TIME – SHARE I/O PINS**

Microcontroller – Atmel Atmega16





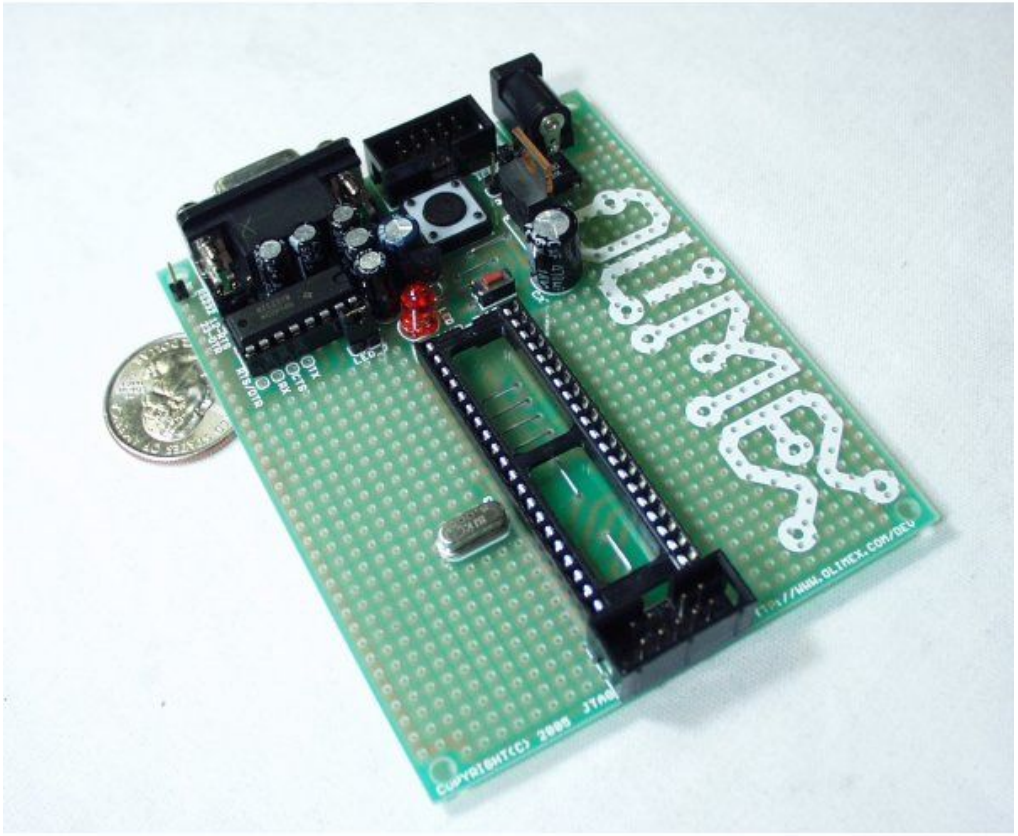
YOU GET A LOT OF FUNCTIONALITY IN A SINGLE PACKAGE

Microcontroller – Save Time By Buying a Proto board

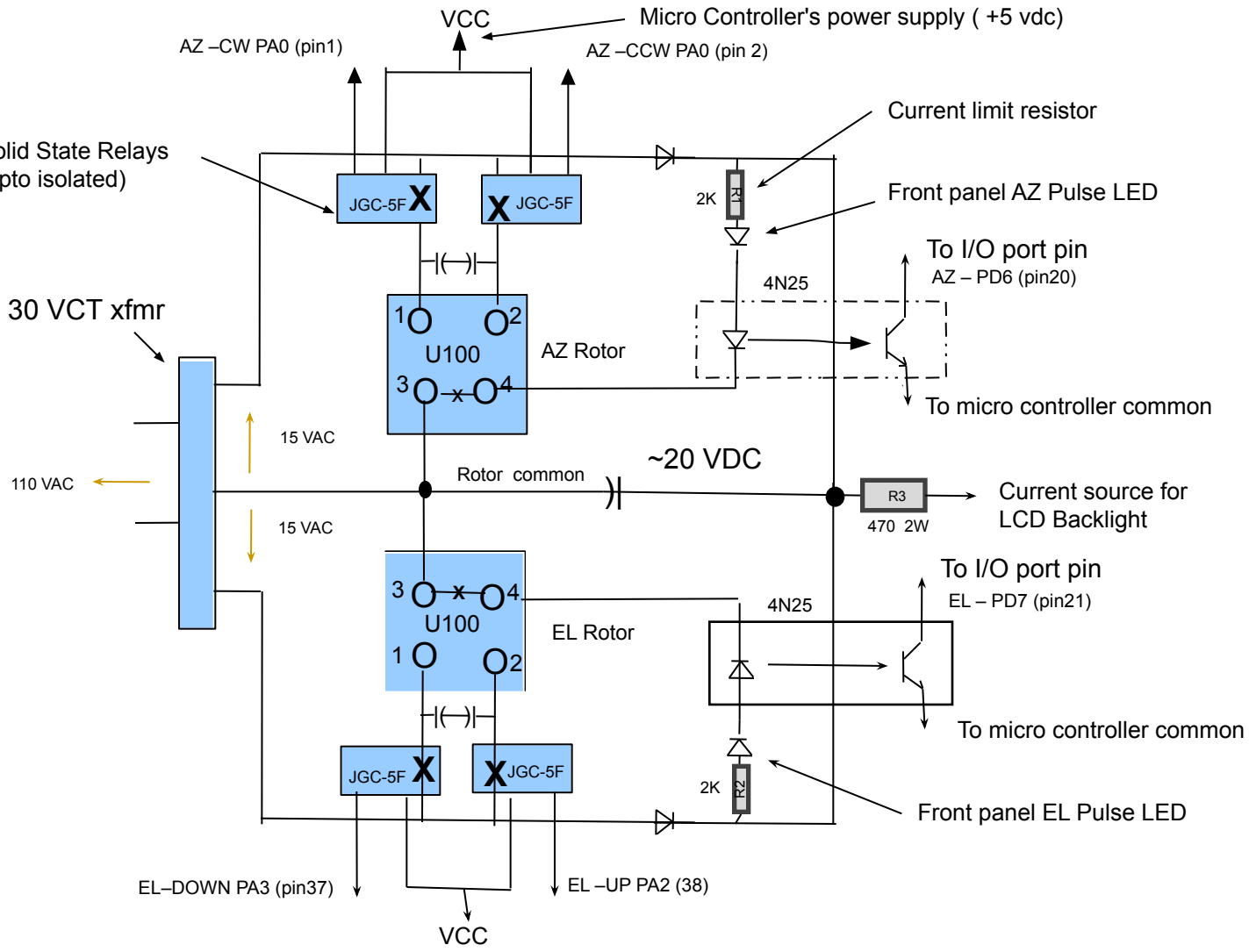
I use this development board for almost all of my projects.

Saves a lot of soldering and cost about $17.00

I get it from “Spark Fun”.



Partial Schematic of the Rotor Controller System – Rotor interfaces

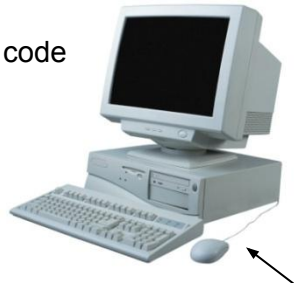


Front Panel Switches – Interface to the Microcontroller

- Micro Controller Port assignments (Active low)
  - Port A Pin 0 - Azimuth ClockWise (CW)
  - Port A Pin 1 - Azimuth Counter ClockWise (CCW)
  - Port A Pin 2 - Elevation Up
  - Port A Pin 3 - Elevation Down
  - Port A Pin 4 - Calibrate momentary pushbutton
  - Port A Pin 5 - Auto Track momentary pushbutton
  - Port A Pin 6 - Azimuth Pulse input
  - Port A Pin 7 - Elevation Pulse input
- LCD Port assignments (4 bit data interface)
  - Details are in a header file

Development Environment

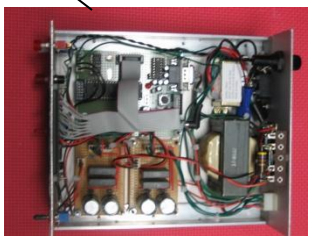
Fedora Core 7 GNU/LINUX
With AVR-GCC tool chain



Write/debug source code

Program flash memory
using “avrdude” utility

Debug data Serial port

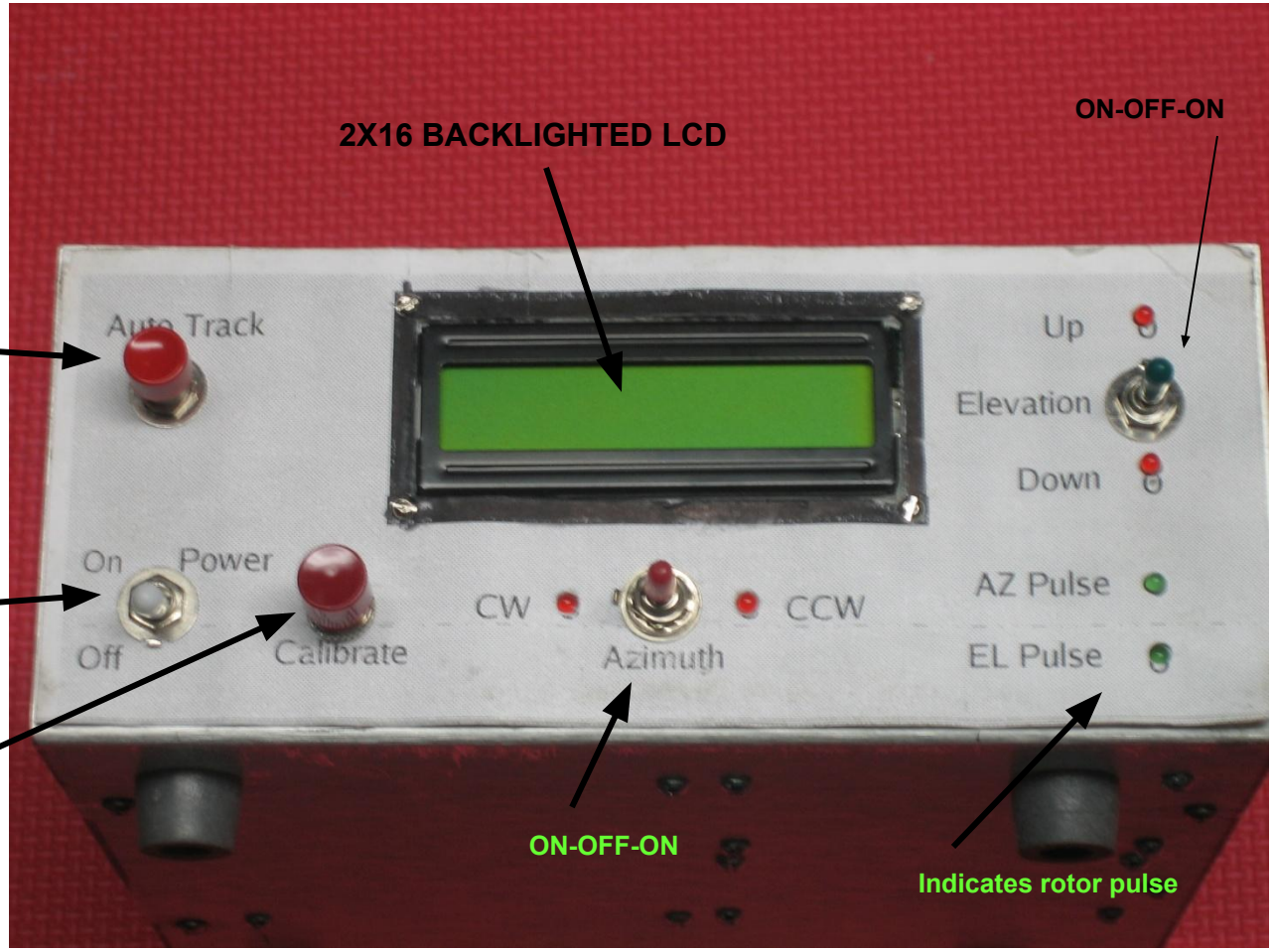


Rotor control cable

Novacomm1 protocol

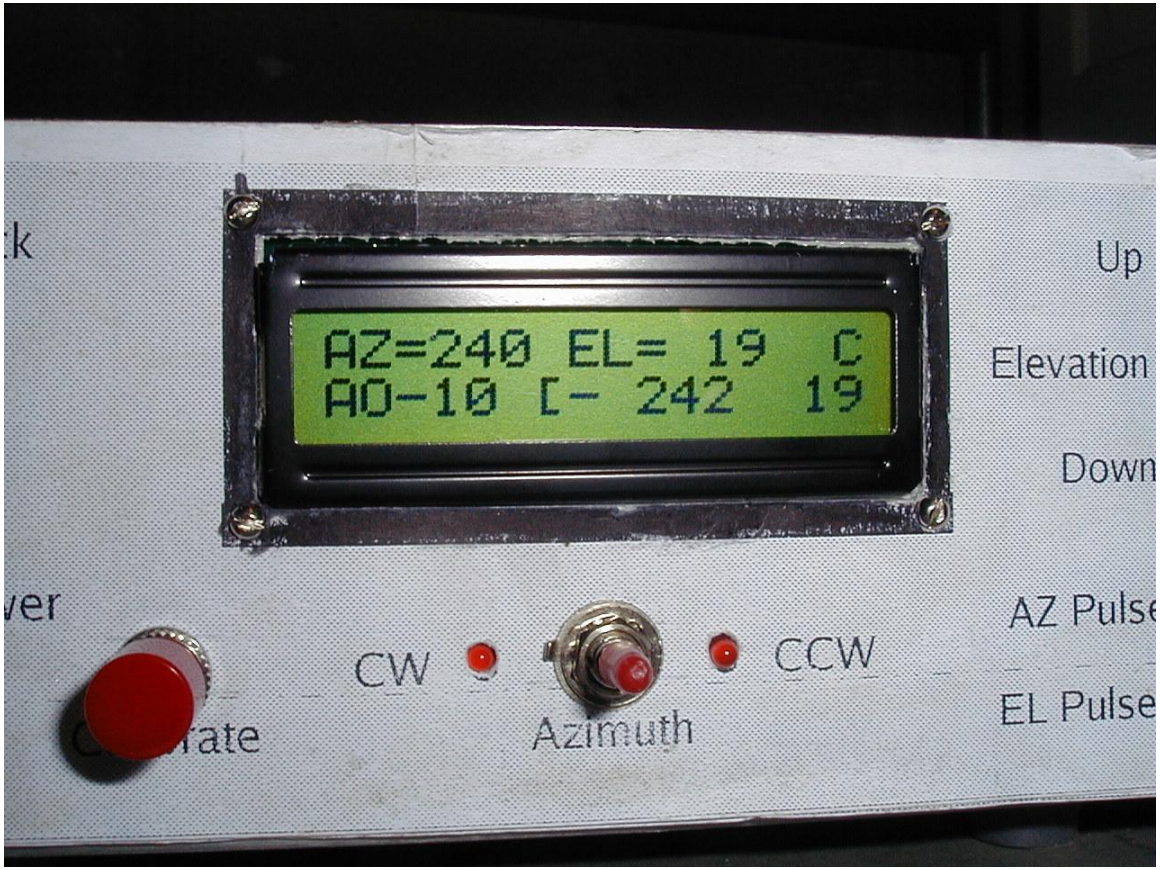
Windows – running NOVA

Features On My Rotor Control Box

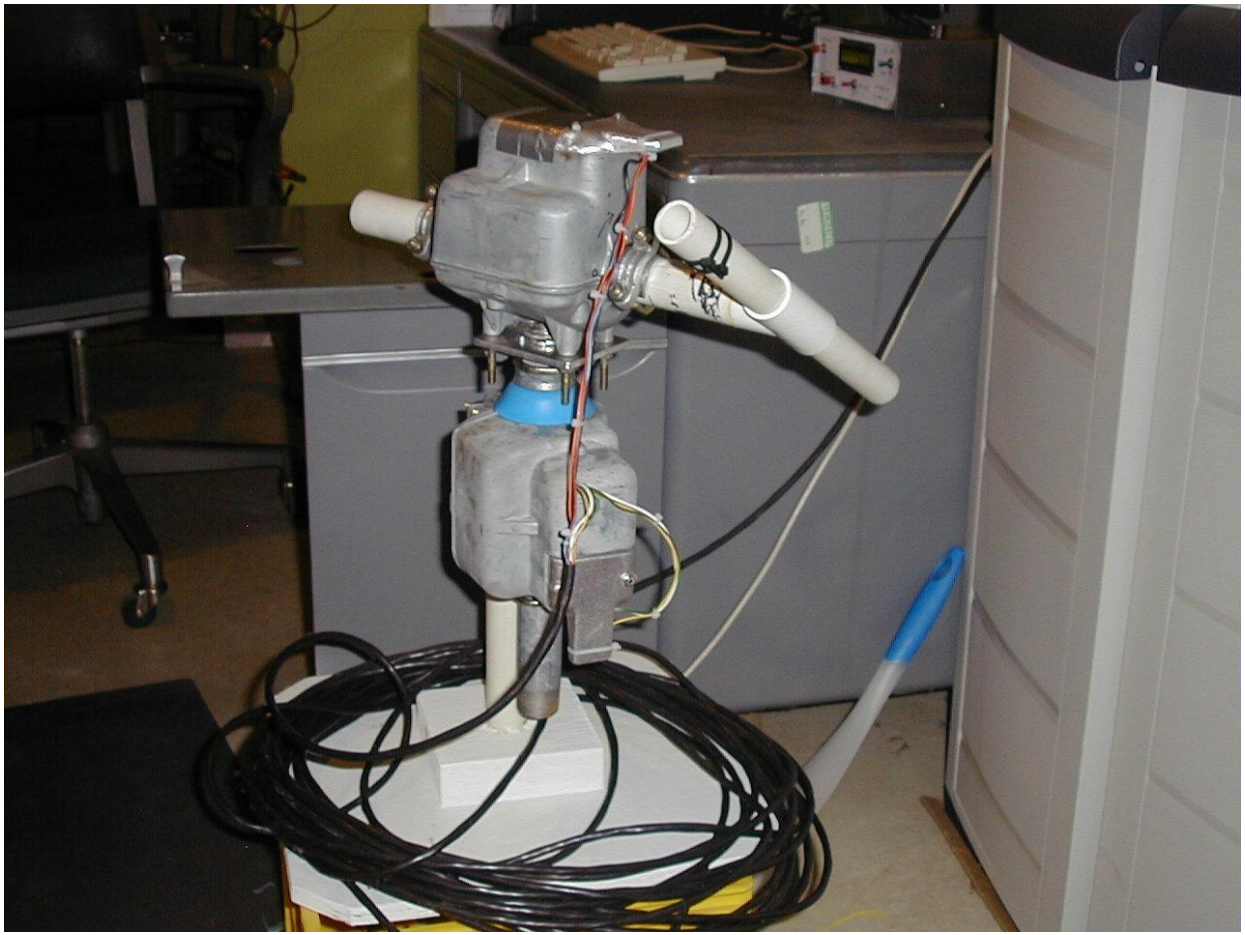


Manual Mode - AZ & EL Reading

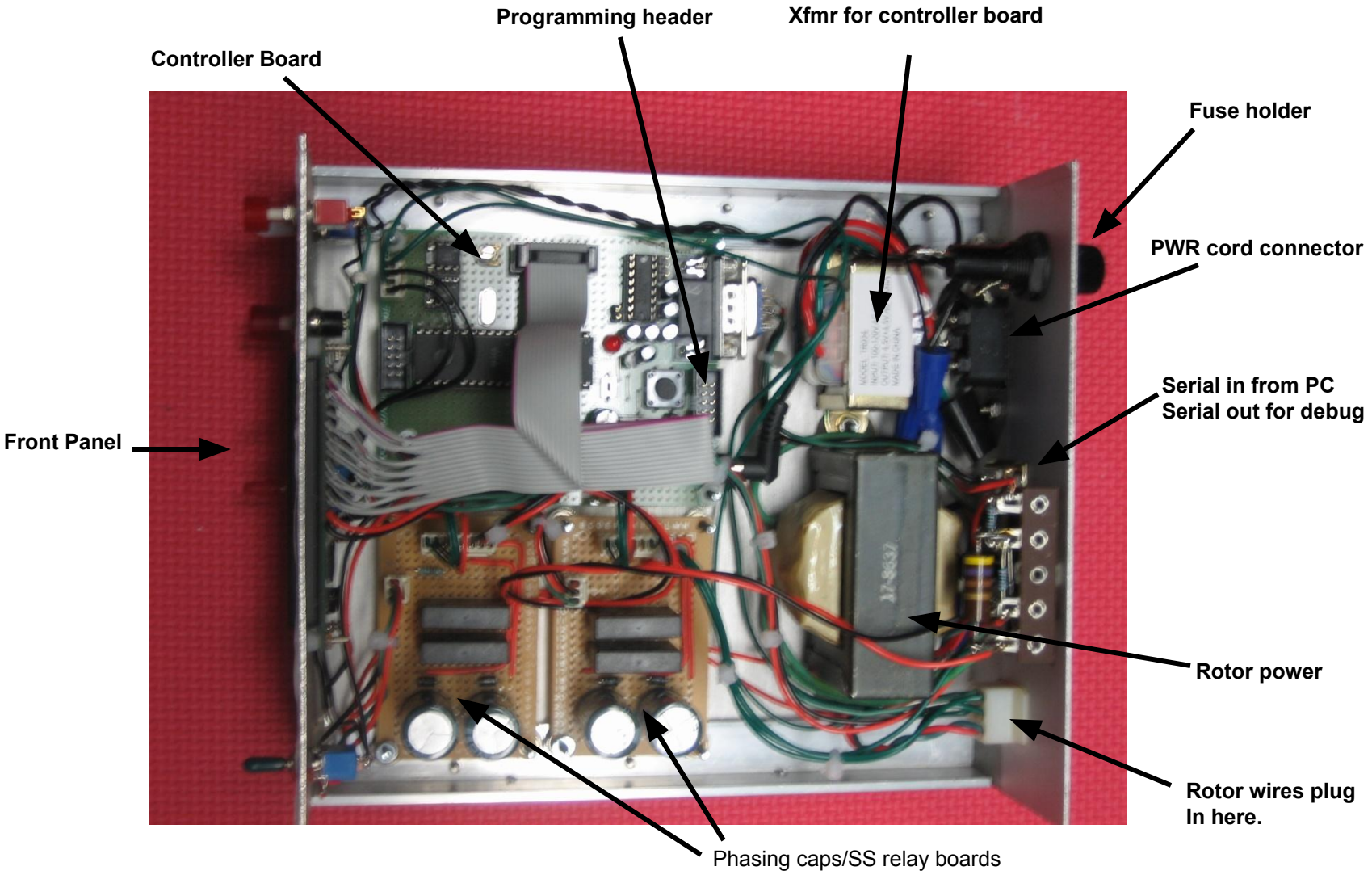
Auto track mode – tracking AO-10 satellite



The Rotor Teststand



A Look Under The Hood



A Few Software Statistics

- ATMEGA 16 Controller
  - 16KBytes Flash (Program) memory
  - 512 Bytes of EEPROM
  - 1 K SRAM
- Software Sizes
  - Program 13394 Bytes
  - Data 262 Bytes – Initialized read only data
  - BSS 399 Bytes – initialized read/write data
  - Total 13995 Bytes
- 30 source files
  - All source is written in “C”
- AVR-GCC Tool Chain programs

High Level Software Design

- BACKGROUND processing every 5 milliseconds
  - Watch every switch in the system
    - Monitor & debounce every switch in the controller
    - Advertise debounced state to the FOREGROUND processing
  - Maintain software timers
    - Decrement every interrupt (5 ms)
- FOREGROUND processing
  - Manage a simple “state machine” based on operating modes:
    - Calibrate
    - Initialize
    - Manual
    - Auto
  - Manages the front panel LCD display & LEDs
  - Fault detection/recovery strategy

Field Day 2008 Satellite Antenna Setup

Performance Of The Controller

- Used successfully in last two Field Days
- Sensitive to drag on the beams – coax
  - False detection of physical stop or obstruction
    - Dressing the coax better resolved this
    - Have changed “late pulse” detection parameters
- Be sure the beams are oriented properly before raising the mast <Hi Hi>
- I consider it a success but it has not seen extensive use

Things Left Undone

- Need to get a better schematic in electronic form
  - Scattered around in a notebook now
- Finish the front panel
  - Print another front panel template and put plastic over it
- Need to paint the box
- Understand other Pulser rotors better (AR-22)
  - Mainly for azimuth rotor use
  - Motor power requirements may not be compatible
- Adapt to “Potentiometer” type rotors - Perhaps
  - Made some accommodations, but didn’t finish this
- A few things in the software to clean-up

Closing Thoughts About Antenna Rotors

- Pulsers have many issues to consider
  - Resolution - must interpolate
  - Calibration process
  - Must have persistent memory (power-down) for AZ & EL position
  - Can find them reasonably priced at hamfests

- Potentiometer type rotors seem less complicated
  - Always know where the rotor is
  - No persistent memory required for power-down
  - No interpolation required
  - No directional history needed
  - Less opportunity to get out of sync.
  - Nova tracking software may do most of the work for you
  - But – these rotors may be expensive!

FROM A SOFTWARE PERSPECTIVE

- This was an interesting microcontroller project!
- Microcontrollers can be used for a lot of amateur radio projects!
- If you are patient and persistent you will be successful!