Sun Tracking Base Design and User's Guide

23 June 2025

John Smigel

Rev. 1.1

Table of Contents

1 Overview

This project's purposes were to:

1) Study the potential energy gain of tracking the sun with a rotating base compared with problems associated with using such a base.
2) Design a sun-tracking base that is easily constructed and inexpensive.

A sun-tracking base has been designed and built. The base structure is made from pressure-treated 2x4's that are easily available at a home store. The base parts are assembled with screws and bolts so it can be easily disassembled, if necessary. The base folds into a compact size for transport and storage.

Part of the motivation behind evaluating mechanically tracking base designs is to quantify the potential benefits of electronically tracking (if practical) and other mechanical tracking approaches, mirrors or lens motion, for example.

The sun is tracked using equations defining its observed position for a given day, time, and ground location. A micro controller (uController) board performs automatic control of the platform orientation angle. The platform rotates in one dimension and is designed to track sun motion as it moves from east to west. The maximum sun overhead height changes with time of year in most locations and is manually adjusted using solar panel racks. Most of the solar panel equipment and accessories used have been from the Eco-Worthy company. They sell relatively high-quality products with a lower price than most other suppliers.

The controller uses an Arduino-compatible uController board and associated sensors. The initial prototype components were from an Elegoo 37 Sensor kit and included a uController board (based on the ATmega 2560 uController chip). The kit also includes sample designs and Arduino-based software for an LCD display and a rotary encoder (and many other sensors not used for this project). Arduino is an open-source prototyping platform with "easy-to-use" hardware and software. Some Sensor kit components have been replaced with alternatives that are designed for use in a small enclosure. A small water-resistant enclosure is used to house the control equipment.

The controller is powered by the 48 V (48 V to 54 V range) battery bank that is charged by the solar panels. A 48 V-to-12 V converter supplies both the controller power required and power for the linear actuator that rotates the base platform. The controller board has a 5 V regulator that converts the 12 V input to a 5 V output used by the other digital components (LCD, Real Time Clock, Rotary Encoder, and Relays).

The design includes a manual switch that can be used to rotate the platform.

Some people (online) feel that the cost, added complexity, and extra space required to provide sun tracking capability is not justified. They argue it is better to just use the money and space to add more panels. I do not necessarily dispute this, but am trying to quantify the tradeoffs. It has been reported that an energy increase of about 40% can be obtained by adding solar tracking to a panel or panels.

2 Sun-Tracking-Base Pictures

Figures 2-1 to 2-3 show the assembled base during indoor testing.

Figure 2-1. *Front View of Base.*

Figure 2-2. *Side View of Base.*

Figure 2-3. *Other Side View of Base with Controller.*

Figures 2-4 to 2-7 show close ups of the completed controller box.

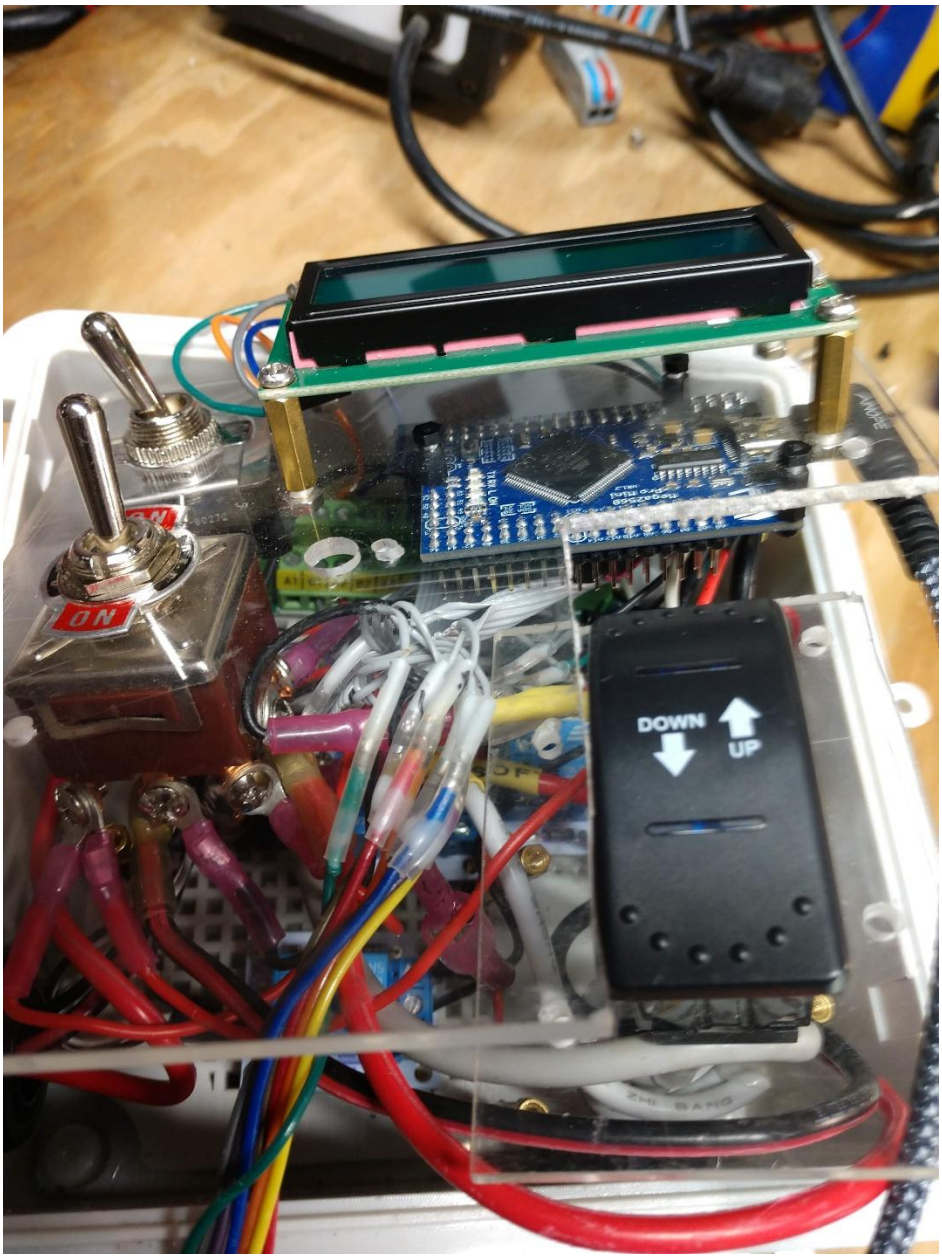


Figure 2-4. *Controller Box Detail 1.*

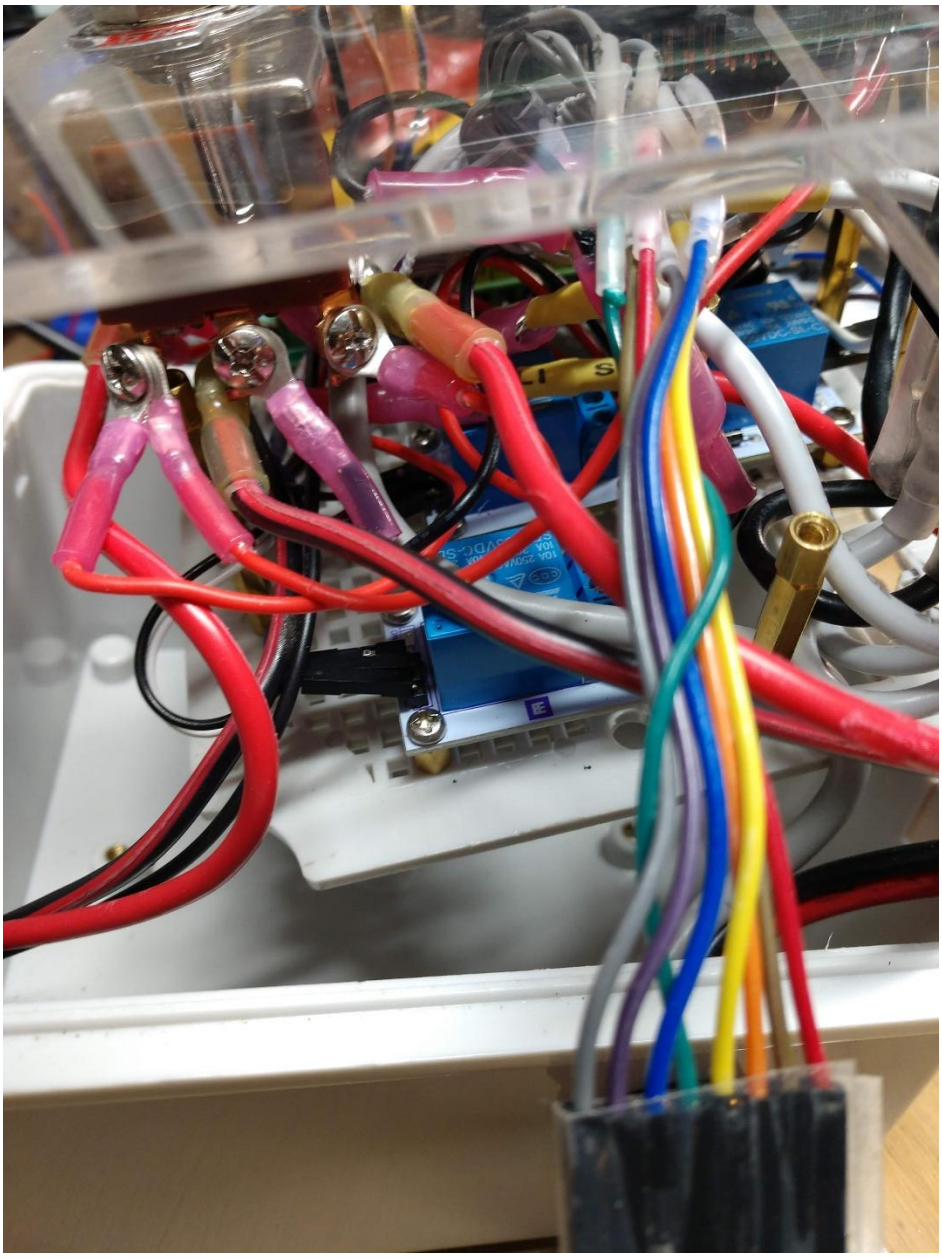


Figure 2-5. *Controller Box Detail 2.*

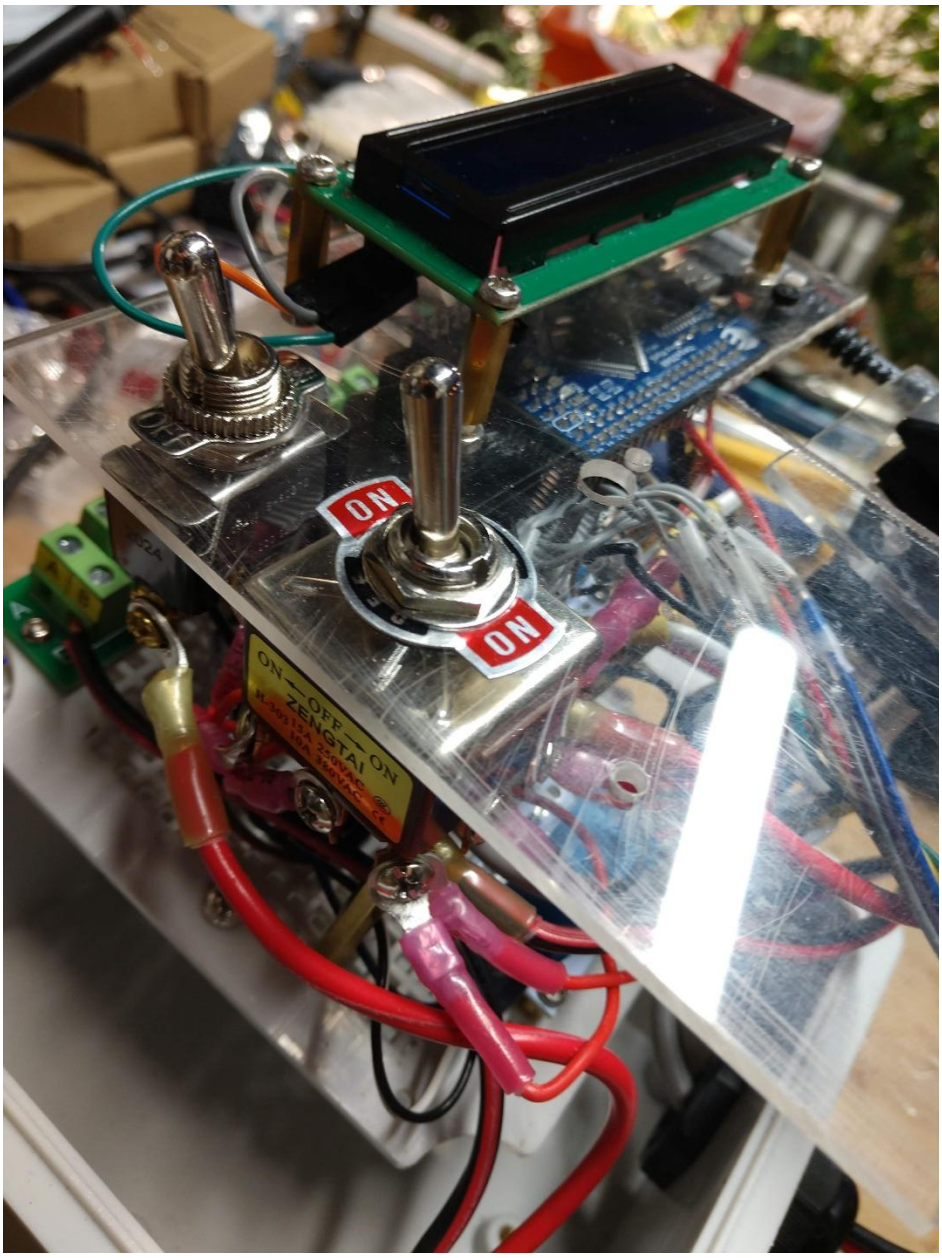


Figure 2-6. *Controller Box Detail 3.*

Figure 2-7. *Controller Box Detail 4*

Figure 2-8 is a video example of the actuator moving the platform over its range of motion. The motion is being controlled by the manual switch.

VID_20250413_155141473.mp4

Figure 2-8. *Video of Platform Range of Motion using the Manual Switch.*

Figures 2-9 and 2-10 show the disassembled lower and upper base pieces ready for transport.

Figure 2-9. *Disassembled and Folded Lower Base.*

Figure 2-10. *Disassembled Upper Base.*

Figures 2-11 to 2-14 show the base and solar panels during outdoor testing, viewed from four different angles.

Figure 2-11. *Base with Panels Mounted Outside, Left Side View. (Note: the panels are set at two slightly different elevations, showing the manual elevation increment of these racks)*

Figure 2-12. *Base with Panels Mounted Outside, Right Side View.*

Figure 2-13. *Base with Panels Mounted Outside, Front View.*

Figure 2-14. *Base with Panels Mounted Outside, Back View.*

3 User Interface and Operation

The interface with the user is performed using an LCD display and a 4x4 flexible keypad. These devices are shown in Figure 3-1. If the date has not been set yet, or has been reset, the date shows the default of 01/01/2000. The LCD display has 2 lines, each with up to 16 characters. There are two toggle switches: 1) top: turns the base on and off and 2) bottom: selects either manual or automatic control. When manual is selected, the base angle is adjusted using the rocker switch at the bottom right of the controller box. To turn the power on, the upper toggle switch is moved left to the on position. The manual/auto switch below the on/off switch has 3 positions: 1) up = automatic (auto), 2) middle = off, 3) down=manual.

Figure 3-1. *LCD Display and Keypad Used for the User Interface.*

When the on/off switch is turned on and the auto/manual is switched up to auto, the controller starts in a default mode. There are 6 modes that are selected by the “C” key (Command) on the keypad. These modes and example outputs are:

0) **Setup**: Leaves the display free to show and enter date, time, start, and end tilt angle inputs (see Figure 3-6 for parameter display examples)
1) **Run**: Main mode that rotates the platform to follow the sun

Figure 3-2. *Run Mode Display.*

2) **Measure**: Shows the calculated tilt angle and rotary encoder count as the platform is manually rotated (disconnect the actuator)

Figure 3-3. *Measure Mode Display.*

3) **East Home Calibration**: Calibrates the encoder when the platform is in the East/Home position (use before Measure mode, no display)
4) **Show Lift**: Shows the calculated maximum panel back lift (height) and the input (manually measured) lift value

Figure 3-4. *Show Lift Display.*

5) **Show Panel Elevation**: Shows the maximum and actual panel elevation angle. The actual panel elevation is calculated from the input measured actual lift and panel length. In this example, the panels should be lowered by 2 degrees to point more directly at the sun at its peak elevation.

Figure 3-5. *Show Panel Elevation Display.*

6) **Enter/Show Parameters**: Entering mode 6 is useful for viewing and entering parameter when set to “auto Run.” Auto run goes into the run mode as soon as setup mode is completed. It can be difficult to enter and show input parameters in run mode because the run mode can be using the LCD. Mode 6 does no functions and leave the LCD free to view and enter parameters. The input parameter displays are shown in Figure 3-6.

A battery in the real-time clock (RTC) board preserves the date and time. If the date is not correct, it can be changed by entering a new date in the format MMDDYY and then pressing the # key.

The * key cycles between the input parameters that can be changed. These parameters are: date, time, min angle, and max angle. The format for the time input is HHMM. The seconds are set to zero. The angle entry format is +-XX.X. Figure 3-6 shows examples of the four parameter entry displays.

Figure 3-6. *Parameter Entry Displays.*

You can view the latitude and longitude settings by pressing the “A” key (example in Figure 3-7). To change latitude or longitude, you must modify the code. The “B” key shows the times of sunrise and sunset (example in Figure 3-8). The “D” key shows the current tilt angle and the maximum sun elevation for the current day and time (example in Figure 3-9).

Figure 3-7. *Latitude and Longitude Display Example.*

Figure 3-8. *Sunrise and Sunset Display Example.*

Figure 3-9. *Current Tilt and Maximum Sun Elevation Display Example.*

The platform should be located on approximately level ground and oriented so the rotation axis is aligned with a line connecting true north to true south. If the ground is not level, the panel tilt and elevation need to be adjusted accordingly.

To command a particular mode, press the C key followed by a mode number.

The main mode is the run mode, mode number 1. After entering a “1” for run mode, the platform rotates to the home (east) position. The home position is all the way down on the left side when facing the controller. Motion stops at the home position. This enables rotary encoder calibration because the home tilt is known, -45 degrees. If the sun’s azimuth angle is higher than the home azimuth, then the platform rotates to point toward the estimated sun position.

If the auto-run flag is set in the code, the base enters the run mode automatically after completing setup. If setup is set as the starting mode after reset/start, the base will return to run mode, even if the controller resets for some reason.

The platform returns to the home position and enters a “sleep” state after a time set in the controller code. This sleep time is relative to sunset. The platform remains home (east) until the sun azimuth enters the platform pointing range in the morning. The home side of the platform needs to point towards east where the sun rises.

The platform may move back and forth several times when searching for a new tilt angle (before it finds the correct angle).

When in run mode, the platform tilt is set to the calculated sun azimuth. The azimuth output on the display has the zero-azimuth position as directly south. Note that the standard is for north to be zero, but since the sun is typically in the south in the northern hemisphere, I use south as the zero point for convenience. This is done by adding 180 degrees to the standard azimuth value.

Using the sun’s azimuth (when defined with true south as zero) approximates the correct tilt angle for the panel when the sun is at its strongest. This is the correct tilt angle when the sun’s path passes directly overhead (zero minimum declination) or is at its minimum declination (at south azimuth==0). At other times, the sun is not strong enough to make the exact pointing direction important. This assumes that the manual adjustment for the panel elevation/lift is set to an angle near the minimum solar declination (maximum sun elevation) for the current season and location. These values are shown by the controller using modes 4 and 5.

The LCD and backlight shut off automatically after sunset. The microcontroller board will still have a green LED lit and the RTC board will have a red LED lit indicating power to it is on. The LCDs will turn back on when the controller exits the sleep mode (when it needs to rotate). If you want to turn the display back on before this, you can press the “C” and switch to mode 6. You will need to modify the code if you need the display to stay on at night while in run mode.

The controller can detect several error conditions. The display will show an error code and the controller will stop operating. The error codes are:

1. Could not rotate platform for calibration
2. Failed to get to commanded tilt after maximum number of attempts
3. Could not rotate to far west position
4. Could not rotate platform during operation

3.1 Manual Solar Panel Elevation Adjustment

The current design requires manual adjustment for pointing the panels in the north-south dimension as the sun moves higher in the summer or lower in the winter. The approximate best panel elevation is equal to the sun's minimum declination. The declination angle is 90 minus the sun's elevation angle in degrees. For example, if the sun moves directly overhead (90 degrees elevation) so that the minimum declination is zero, the panels are kept flat and pointing upward (panels are horizontal). As the sun moves lower in the south, the back of the panels is lifted to point directly at the sun when it is at its maximum elevation (around noon). The controller calculates the panel back lift distance that gives the desired elevation. The desired back panel lift distance is output by the controller in mode 4. The racks have marks on them indicating the elevation angle, but these are difficult to read. It is easier to measure the height of the panel backs.

Figure 3.1-1 shows the calculation of the panel elevation angle from the back lift value and length of the panel.

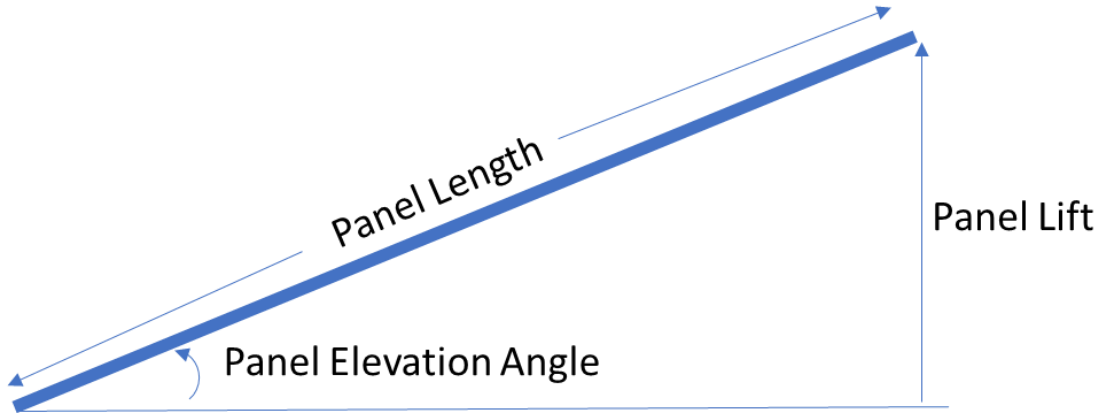


Figure 3.1-1. *Relationship Between Panel Elevation Angle and Lift.*

Using Figure 3.1-1 above, the Panel Lift and Panel Elevation are related according to

Panel Lift = Panel Length * sin(Panel Elevation Angle).

or

Panel Elevation Angle = $\sin^{-1}$(Panel Lift/Panel Length).

4 Rotating Base Design

My base design is shown in Figure 4-1. This design is slightly updated from the base shown in the pictures. The update makes the base slightly shorter in the longer dimension. This makes the base slightly lighter, smaller, and uses less wood. The base length only needs to be long enough to mount the solar array racks.

The base is like a see-saw. The lower part is two A-frames connected by horizontal cross pieces. The A-frames use hinges and wire tension support to allow folding for compact storage and easier transportation.

The specific base design sizes are for older Eco-Worthy 100W Panels (4) of size 39 7/8" by 18 1/8". This can be modified for newer panels that are shorter and wider, but about the same area.

The design also uses Eco-Worthy solar panel racks to mount the solar panels to the top part of the base. Two sets of 2 racks are needed to mount four 100-watt panels. Note that at the time I'm writing this, the current Eco-Worthy racks fit two of the older 100-watt panels, but do not quite fit two newer panels. I have had to extend the racks slightly to fit newer panels. The peak output power of 4 panels in series is about 400W. Actual power depends on sun strength/season and weather.

The base tilt angle follows the sun using a linear actuator and a custom digital controller. Details of the controller design are given in section 6.

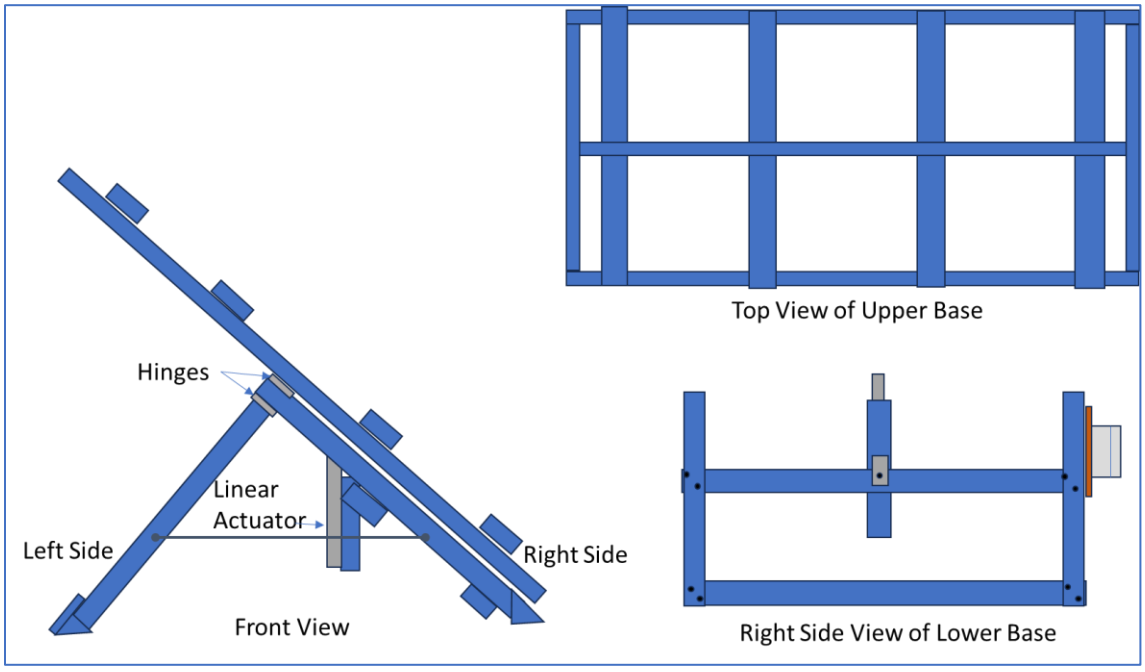


Figure 4-1. *Rotating Base Design Summary.*

4.1 Actuator Position and Mounting

I initially just connected the actuator to the most convenient locations on the bottom and top parts of the base. This worked, but not well. The ability to move the platform and the speed varied too much with tilt angle. The best mounting position keeps the actuator as perpendicular as possible to the tilting platform top part as it is rotated. The best mounting position can be determined with a little geometry from Figure 4.1-1.

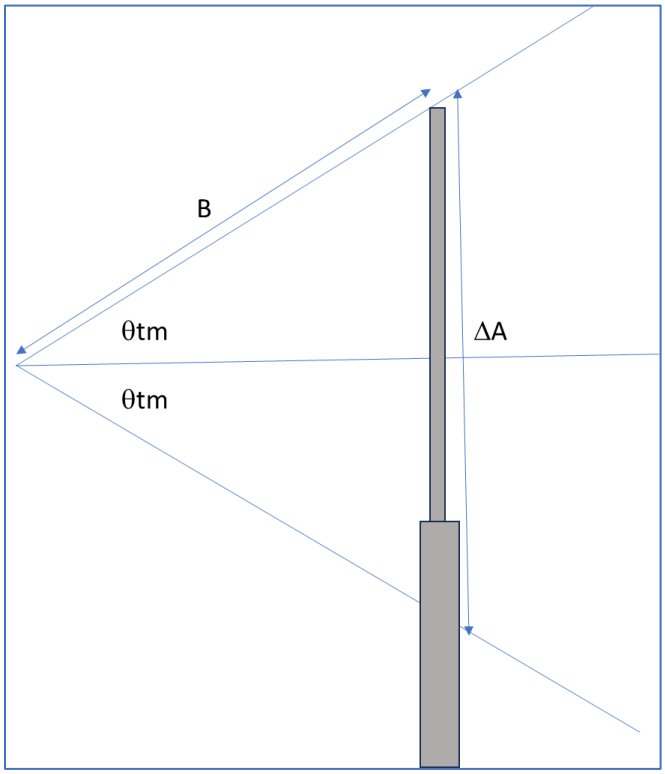


Figure 4.1-1. *Calculation of the Best Actuator Mounting Location.*

Figure 4.1-1 shows the actuator and lines representing a side view of the limiting motion of the moving part of the base. The rotation is about the point on the left where the lines connect. We want to calculate the distance, B, where the actuator needs to be mounted for it to span the desired tilt angle between fully extended and fully contracted. The distance, ΔA, is the maximum actuator length change. B is the distance from the point of rotation that the actuator top should be mounted. From the geometry,

$$B\sin(\theta tm) = \Delta A/2 \qquad (4\text{-}1)$$

or

$$B = \Delta A/(2\sin(\theta tm)).$$

For this design,

$\Delta A = 12”$

and

$\theta tm = 45$ degrees,

resulting in

$B = 8.48”$

Therefore, the actuator top should be mounted about 8.5” from the center of rotation. For the controller code listed in Appendix B, the actuator needs to be mounted on the east side of the platform. To find the point where the bottom of the actuator should be mounted, extend the top mounting position down by the full actuator length when it is rotated with the actuator fully up. The full length for the selected actuator is 28.4” (from Figure 16.4-1). An actuator mounting board is cut and mounted on the bottom cross support that enables the bottom of the actuator to be connected to this location.

Detailed photos of the actuator mounting are shown in Figures 4.1-2 to 4.1-4. The actuator is mounted using two pieces of 2”x4” wood, two corner joints, and one 3” lag screw with 1” washer. One of the corner joints is bent to match the angle that the bottom actuator mounting board makes with the horizontal support board. A notch is cut in the horizontal support board so the mounting board is in the desired position, as calculated above. The mounting board does not need to be perfectly vertical; however, it needs to be close to vertical. Otherwise, the torque on the mount board may just bend/rotate the board instead of rotating the platform.

Figure 4.1-2. *Mounting Boards Installed Before Mounting the Actuator.*

Figure 4.1-3. *Mounting Boards with the Actuator Mounted.*

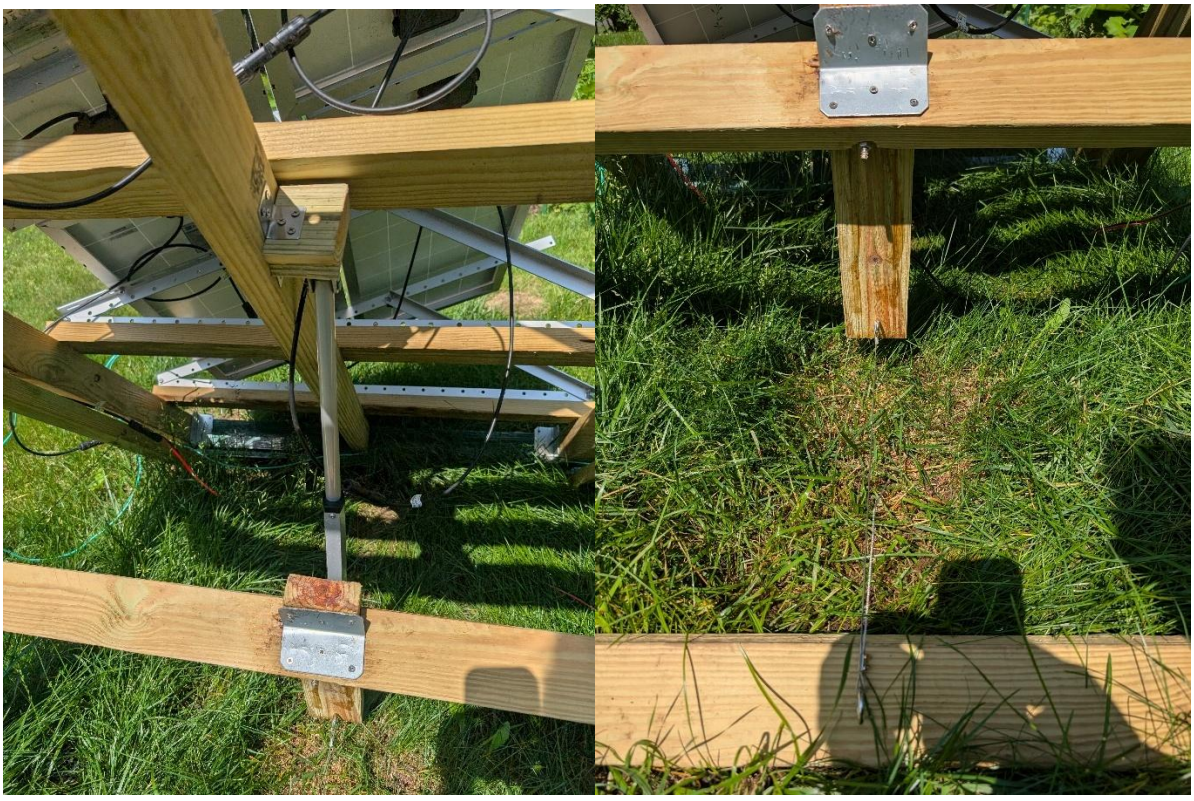


Figure 4.1-4. *Additional Support Corner Joint is Bent to the Correct Angle. A wire connected with 2 eyebolts prevents the bottom of the actuator mount from being pulled up.*

4.2 Base Wood Components

The base wood pieces are listed in Table 4-1. The base pieces are held together with screws, bolts, and screwed-in corner joints.

Table 4-1. Wood Base Pieces

Description	Length(in/ft)	Number	# of 2"x4"x8'
A Frame Left	37.5/3.13	2	1
A Frame Right	40/3.34	2	1
Lower Base Horizontal Supports	47/3.92	3	2
Long Top Edges	71/5.92	2	2
Top Center	68/5.67	1	1
Short Top Edges	42/3.5	2	1
Top Cross Supports/Mounts	45/3.92	4	2
Linear Actuator Mounts (upper; lower)	2.5/0.21; 17/1.4;	2	Scrap from above

4.3 Base Wood and Actuator Connections

Specific components for connecting the base wood pieces, including the actuator, are listed in paragraph 6.2, Parts List, Table 6.2-1.

5 Results and Discussion

I have tested the base and it does a good job keeping the platform tilt close to the commanded tilt value. The measured tilt angle is not as accurate as I expect and is discussed in the following section about remaining problems. I have been able to remove the bottom of the base from the top and transport the pieces from my basement outside. So, the portability goal has been achieved.

The base components cost approximately $300.00. The most expensive components are the linear actuator, pressure-treated lumber, and stainless-steel bolts. This cost is slightly more than Eco-Worthy's single-axis solar trackers. Eco-Worthy's metal solar trackers that cost between $300 and $590. The more expensive tracker tracks in 2 dimensions.

The low-cost goal has not yet been achieved. Since a rotating base costs as much or more than the solar panels, it would be better to buy more panels instead of a rotating base if the additional space is available for more panels. If space is limited and you need to maximize power output, a rotating base may make sense.

The base has been tested with four 100 W Eco-Worthy panels connected in series. On May 17, 2025, the platform started rotating (>-45 deg azimuth) at about 11:30 am and stopped rotating (~40 deg) at about 2pm. Therefore, at this time of year, the period when the platform is actively following the sun is about 2.5 hours long. This is when the sun is strongest. At 2pm, when rotation stops at the west rotation limit, the panels produce about 28 6W (5.3 amps @ 54 V). This is when the sun is partly out. Since the batteries were full, this may not be the maximum power that could have been produced. The charge controller enters "keep" mode and limits power to the batteries to only what is being consumed. I was running my aquarium off this solar power at the time. The aquarium uses between 285 W and 385 W, depending on if the heater is on.

I adjusted the angle calibration factor to 0.33 degrees/count to get the platform to point at approximately the correct angle. I plan to test this more over the next few days and compare power generated with and without rotation. Below is a log of the results so far.

May 19, 2025 – Testing power generated with rotating array on sunny day. Starting power stored at 10:15 is 624.8 kWh. Sun is just starting to hit panels. Running aquarium from batteries to keep power supply needed. Keeping manual elevation/lift slightly high at 17" (recommended 15.7") next lower increment is 14". Note: did not start tracking the sun until about 12 noon (-25 deg solar azimuth). I had added an error check to detect if the platform is stuck or reached the west limit. This limit was being triggered when the motion was first trying to start. Because of slack in the actuator mounting, it takes up to 10 attempts at moving the platform from the east home position until motion is detected. Setting the number of attempts with no motion from 3 to 10 fixed the problem. Peak of 5.5 A, some intermittent clouds.

1pm - Total Power: 625.5 kWh, battery current: 5.2 A

6:30pm – Total Power: 626.6kWh, battery current: 0.3 A, load power: 48 W, Battery Voltage: 53.0 V. Power stored: 1.8 kWh.

May 20, 2025 – Updated controller code to turn off display at night. Starting total power: 626.6 kWh. Not a good day to measure solar power. Batteries got full in morning then it got cloudy. Ending total power at 6:10pm: 627.4 kWh, battery current: 0.3 A, battery voltage: 52.7 V. Power stored: 0.8 kWh.

May 21, 2025 – Platform is still running & rotating. Cloudy today, so not good for measuring solar generation capability. Total power at 6:15pm: 627.7 kWh, battery current: 0.3 A, battery voltage: 52.9 V. Power stored: 0.3 kWh.

May 22, 2025 – Raining hard today. Good test of if it is rain-proof. Total power at 6:15: 627.9 kWh, battery current: 0.2 A, battery voltage: 53.0 V. Power stored: 0.2 kWh. (Not powering aquarium today or yesterday).

May 23, 2025 – Mostly cloudy and a little rain. Passed the rain test yesterday, about 2" of rain. Still in rotating mode. Total power at 8:20pm: 628.7 kWh, battery voltage: 53.0, Power stored: 0.8 kWh. **Note**: I have not noticed any of the "Remaining Problems" listed below in the last 5 days of continuous operation. Resetting may be occurring because I have it set to automatically restart running on reset.

Remaining Problems

1) The rotary encoder tilt angle estimate remains somewhat erratic. These encoders say they have a standard output of 600 pulses per revolution. In addition to the pulse output, the encoder outputs a signal that indicates rotation direction. The pulse output is connected to a uController interrupt line. When an interrupt occurs, the encoder interrupt service routine (ISR) counts either up or down. The number of counts is converted to a change in rotation using a calibration factor related to the number of pulses per revolution. I initially assumed that there would be one interrupt per pulse. This would give a calibration factor of 360 degrees rotation/600 pulses = 0.6. However, I appear to get up to 1200 interrupts per revolution or up to twice as many interrupts as I initially expected. The usual number is 1091 interrupts per 360 deg. Also, the rate of interrupts is not always the same. It may depend on speed of rotation. Some interrupts may be being missed. I changed the code to start and stop rotating while moving to the commanded angle to slow the average rotation speed. Operation has been consistent lately and fairly accurate with a 0.33 degrees/interrupt calibration factor (A factor of 0.3 would be two interrupts per pulse).
2) Another problem is that the microcontroller resets itself intermittently. This tends to happen after the controller has been running for a long time (18-24 hrs.). I typically can leave it running for an entire day without it resetting. A workaround for this problem is to set the autorun flag to true. This option puts the base in run mode after completing setup when restarted/reset. The disadvantage to autorun is that the platform can start moving on its own if the controller resets itself.
3) The microcontrollers used in the design have a common ground for the 12 V input and the 5 V output. This means it is not possible to completely isolate the digital 5 V power from the analog 12 V power. Noise from 12 V components can cause the microcontroller and/or display to fail. I have minimized this problem by using a large capacitor on the 5 V power and by having a ground connection to a solid ground point. Occasionally, a failure still occurs that I attribute to 12 V power noise.

4) The real-time clock board sometimes resets itself for an unknown reason. I notice this tends to happen after I use the manual tilt control, but I have not determined if this is causing the problem. When it resets, you need to reenter the date and time.
5) The platform may need to be manually moved away from the fully-west position before entering the run mode. This is because it first tries to rotate to verify motion is possible. If it is already at the west limit, it may report an error that motion control is not working. The controller code can detect most, but not all error and problem situations. If an error is detected, an error code is output and the controller stops. I have the default west tilt limit set at about 40 degrees, so the array should not end up at the west angle limit when in the auto/run mode.

Potential Design Enhancements

1) The 12 V power is currently switched between either the automatic digital components or the manual rocker switch. This means the digital components, rotary encoder, and display can't be used to show the tilt value when set using the rocker switch. It should be possible to use the microcontroller, rotary encoder, and display to show the manual tilt by keeping the 12 V power to these components connected when the manual switch position is used. I have not tried this yet.

2) Another linear actuator could be installed to control the height of the back of the solar panels (I have been calling this lift). This would make it easier to point towards the sun in this dimension. Currently four points need to be manually disconnected on the panel racks to change the North-South orientation. An additional digital switch would be needed to select which actuator is being controlled.

3) If the panel motion can be controlled in both dimensions, it would be desired to have a wind sensor and automatically place the arrays in a horizontal position if it gets too windy. Also, tie-downs can be added to the base to prevent being blown over in very windy conditions.

4) It might be possible to mount the racks directly to the lower frame joists and not have the four cross-mounted boards. This would reduce complexity, weight, and cost. The platform would not be as strong and there would be limited locations to secure the racks. Rack mounting holes would need to line up with the lower frame joists.

6 Controller Design

The initial controller design was based on components included in "ELEGOO Mega R3 Project Most Complete Ultimate Starter Kit with Tutorial Compatible with Arduino IDE", https://www.amazon.com/dp/B01EWNUUUA?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_6 that includes an Arduino Mega 2560-compatible microcontroller development board. It costs about $66. The kit also includes a keypad, real time clock (RTC) board, LCD display, and a rotary encoder. You don't need this kit and can buy the needed components separately, if desired. In some cases, slightly different components have been used in the final design to fit in the controller box and provide all the required functionality.

The controller sets the platform tilt angle to point at the sun as it moves from east to west. A controller diagram is shown in Figure 6-1.

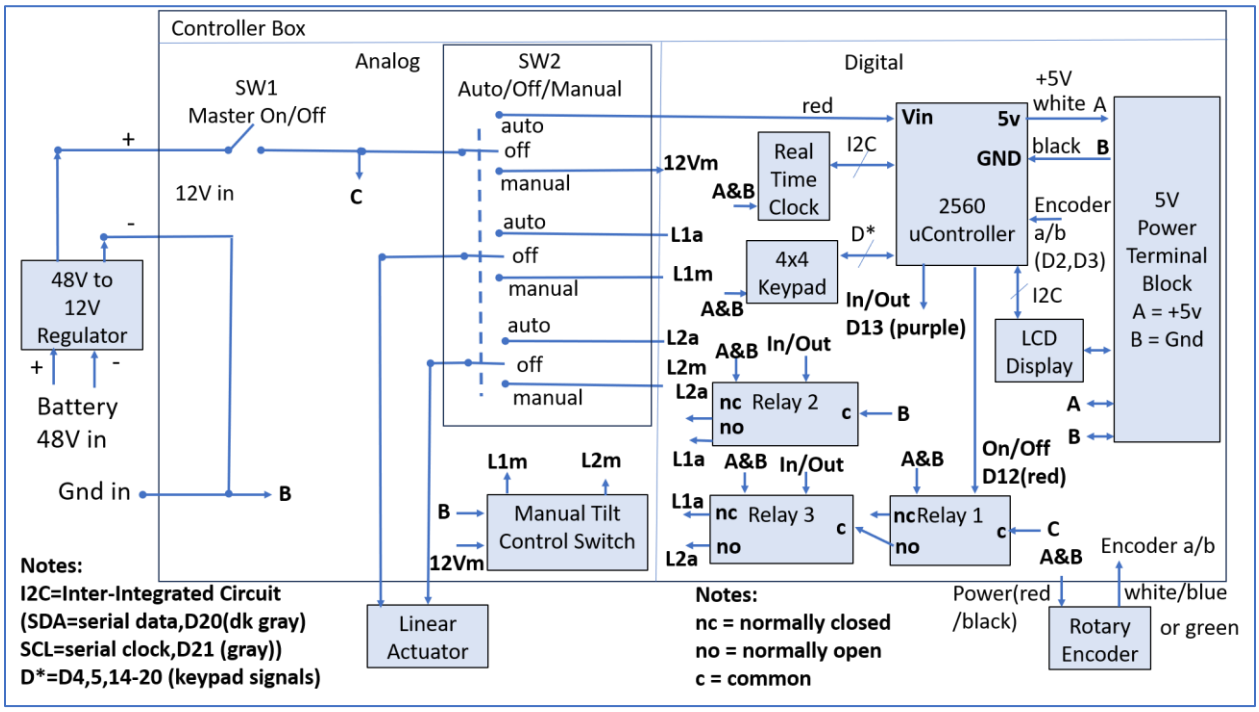


Figure 6-1. *Controller Block Diagram.*

The 2560 microcontroller (uController) is programmed to calculate the sun position as a function of the location (latitude and longitude), date, and time of day. These calculations are summarized in Appendix A. The Arduino-compatible controller board is programmed in the C language and the code is listed in Appendix B. Arduino boards have two top-level functions: 1) setup and 2) processing loop (loop). The code monitors the keypad inputs and performs the appropriate actions. The platform tilt is controlled by setting the linear actuator direction (out

or in) and by turning it on or off in the code. The 5 V relays control the actuator on/off and the actuator direction.

A rotary encoder receives 5 V power from the controller and sends back 2 phase channels (channels a and b) that can be monitored to determine the platform rotation.

All digital components need 5 V power that is produced by the 2560 microcontroller and distributed using a power distribution block.

The controller and power converter use about 47-48 W.

6.1 Component Descriptions

The following paragraphs describe each component used in the design. In many cases alternate components can be used.

6.1.1 Microcontroller Board

A compact and low-power board that is based on the Microchip (former Atmel) Atmega 2560 16au (16 MHz) uController chip is used as the programmable part of the design. The Elegoo development board is called Mega 2560 R3. There are several manufacturers of Arduino mega 2560-compatible boards. Two such boards have been used and tested for this application:

1) ePalZoneXP MEGA 2560 R3 Pro Mini (with male pin headers), https://www.amazon.com/gp/product/B0CHN2R9VL/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1
2) Songhe MEGA 2560 PRO Embed (with male pin headers), https://www.amazon.com/dp/B07TGF9VMQ?ref=ppx_yo2ov_dt_b_fed_asin_title

There are other board options and the options available appear to change regularly. I originally could find a version of the ePalZoneXP board with the male pin headers already soldered. Now they only offer the board without any headers soldered on and you must solder the headers on yourself. I like the construction of the ePalZoneXP product better than the others I have tried and it has a USB C port instead of the micro-USB port on most other similar boards. The Elegoo development board could be used, but it has female headers and is larger and would require a larger enclosure. Note that I currently use and have done most testing with a Songhe board after accidentally breaking my ePalZoneXP board by incorrectly connecting the power.

For additional information on the MEGA 2560 board or the ATmega 2560 chip, download the pdf data sheets from the web.

Figure 6.1.1-1 is a picture of the ePalZoneXP board and male pin headers. This design only uses D0-D32, Vin, GND, and 5 V.

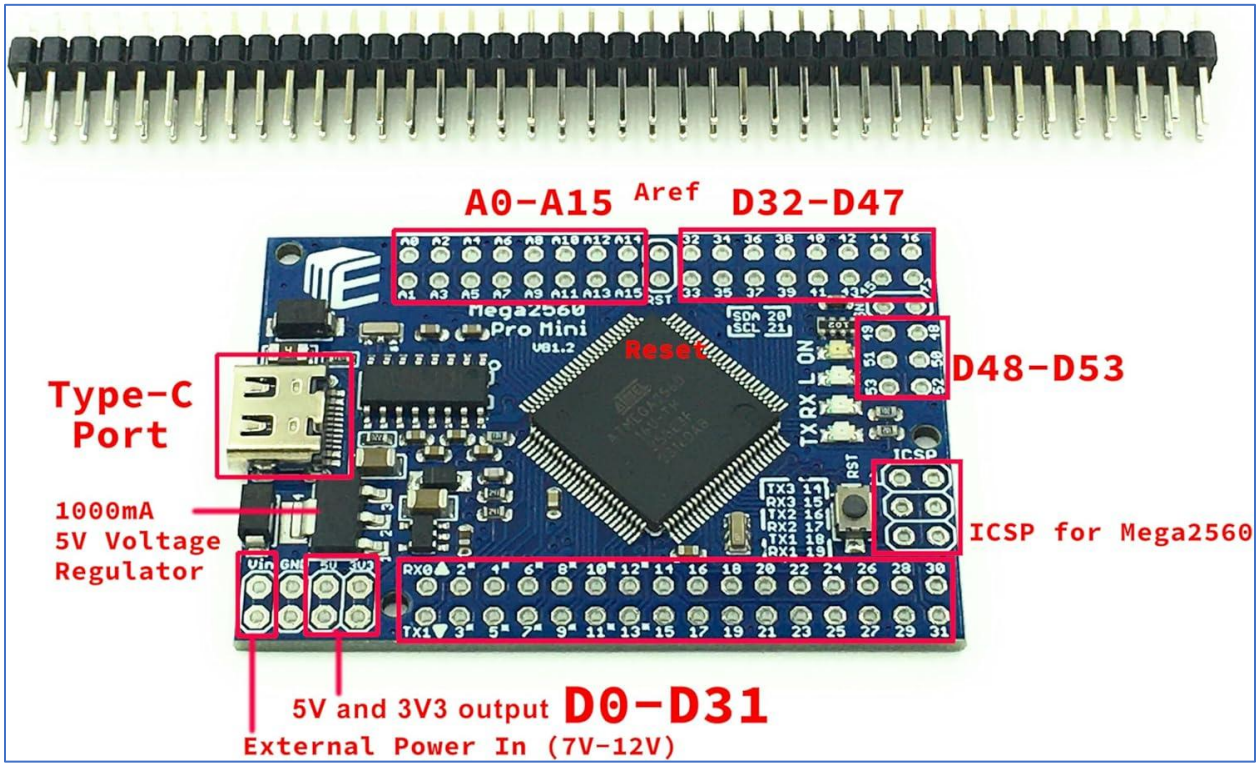


Figure 6.1.1-1. *MEGA 2560 Microcontroller Board.*

The connections to the MEGA 2560 board are made using a ribbon cable described next.

6.1.2 Ribbon Cable

There are 32 digital data bus outputs in 2 rows of 16 on the mega 2560 board. Standard 2-row ribbon cables and female connectors appear to only come with 34 sockets. The Songhe mega 2650 board can take up to 42 sockets in 2 rows. These are the 32 data lines, Reset and AREF, and the eight power input/output pins (two for each of 4).

I want to use larger wires for power than are used with the ribbon cables, so I just shift the 34-socket connector so that pin 1 is aligned properly and the last 2 pins are not used. I plug individual sockets into the power pins. If using the Songhe board, cut the Reset and AREF pins short (these pins are not used in this design) so the connector can fit in the same way.

Figure 6.1.2-1 is a picture of the ribbon cable, Fielect 2pcs IDC 34 Pin Connector Gray Wire Flat Ribbon Cable Connector Length 30 cm 2.54 mm Pitch,

https://www.amazon.com/dp/B0817MJL55?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1&th=1.

Figure 6.1.2-1. *Ribbon Cables.*

I cut one 34-socket connector off one of these ribbon cables and spliced in other connectors, as required to connect to other components. Not all 34 cables/sockets are used.

I tried making custom sockets with multiple cables connected, but failed completely.

6.1.3 Terminal Block

The terminal block is used to distribute 5 V power to the digital components. This makes it easier to remove and replace components. The block I used is: OONO 16 Amp 2x12 Position Terminal Block Distribution Module for AC DC, https://www.amazon.com/dp/B08TBXQ7H6?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_6

Here is a picture of the block:

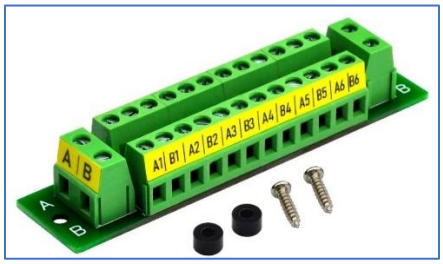


Figure 6.1.3-1. *Terminal Block for 5 V Power Distribution.*

6.1.4 LCD Display

Most character LCD devices use a raw input protocol called HD44780. This protocol uses up to 16 inputs to control the display. This relatively large number of inputs is needed because the protocol sends up to 8 data lines in parallel. Recently, most of the LCDs produced come connected with a serial board that connects to the display with only 4 inputs (2 power and 2 in/out). The serial protocol used is the standard I^2C, inter-integrated Circuit protocol. This protocol uses 2 bidirectional lines called SDA, serial data, and SCL, serial clock. When I had to replace a broken LCD, I switched to the serial protocol option. There are many options from different manufacturers. One of the ones I have used is: Teyleten Robot LCD1602 LCD Display Screen Module 16X2 Character Serial Blue Backlight LCD Module PCF8574T PCF8574 IIC I2C for Raspberry Pi Arduino STM32 DIY, https://www.amazon.com/dp/B07T8ZG5D1?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_3

LCD pictures are shown in Figure 6.1.4-1.

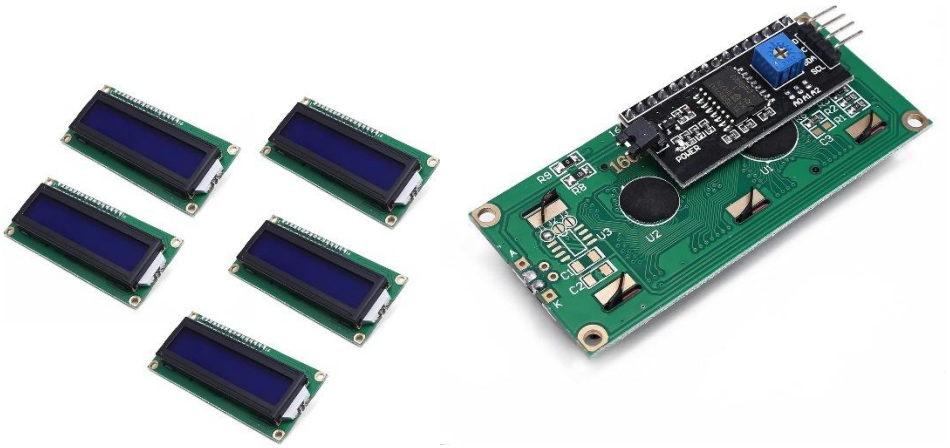


Figure 6.1.4-1. *LCD Modules.*

Software drivers for these LDC's are available for the Arduino boards.

Most of the power will be consumed by the backlight which can be controlled like any other LED.

The LCD controller itself uses very little power but if you want to blank the display, without disturbing its contents, you can use the 'display() and noDisplay() methods.

These are the backlight API calls that many libraries implement:

backlight(); *// turn on backlight*

noBacklight(); *// turn off backlight*

setBacklight(dimvalue); *// set backlight intensity*

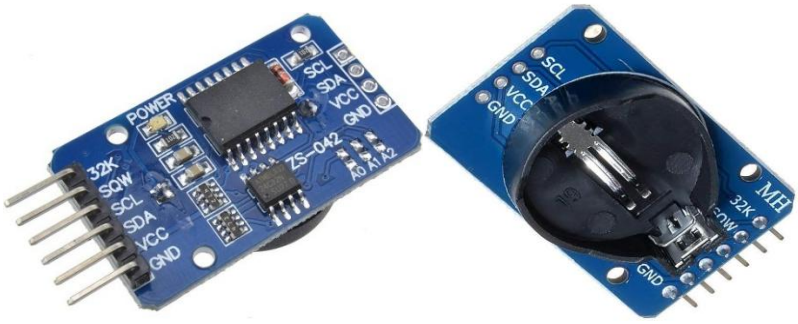
on(); *// turn on pixels and backlight*

off(); turn off pixels and backlight

6.1.5 Real Time Clock (RTC) Board

Many alternative RTC boards are available from different vendors. This design uses a board that is compatible with the DS3231 integrated circuit (IC) chip. The DS3231 is a low-cost IC providing a highly accurate, real-time clock for use with Arduino, Raspberry Pi, BBC micro-bit and other popular small computing devices. The IC is typically mounted on a circuit board or module, along with other hardware, such as header pins, supportive electrical components, and even EEPROM memory chips, for convenient attachment to a breadboard or an Arduino. The board used in this design has a battery backup to retain the stored date and time. The inexpensive boards that are available on Amazon all appear to use a poor design that tries to recharge the battery. Since lithium cell batteries last a long time, recharging is unimportant. What is important is that any batteries used are charged properly so they do not overheat and/or explode. The charging circuit on these boards is inadequate to properly charge a rechargeable lithium battery. It is recommended to modify the board by removing the battery charging and using a non-rechargeable battery. It is important to disable charging if a non-rechargeable battery is used to prevent it from exploding. Disabling charging only takes a few seconds and is done by removing a resistor from the board.

A board tested for this design is: DS3231 Real Time Clock Module High Precision AT24C32 IIC RTC Sensor for Arduino Nano Mega2560 Leonardo Raspberry Pi, https://www.amazon.com/dp/B07V68443F?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1



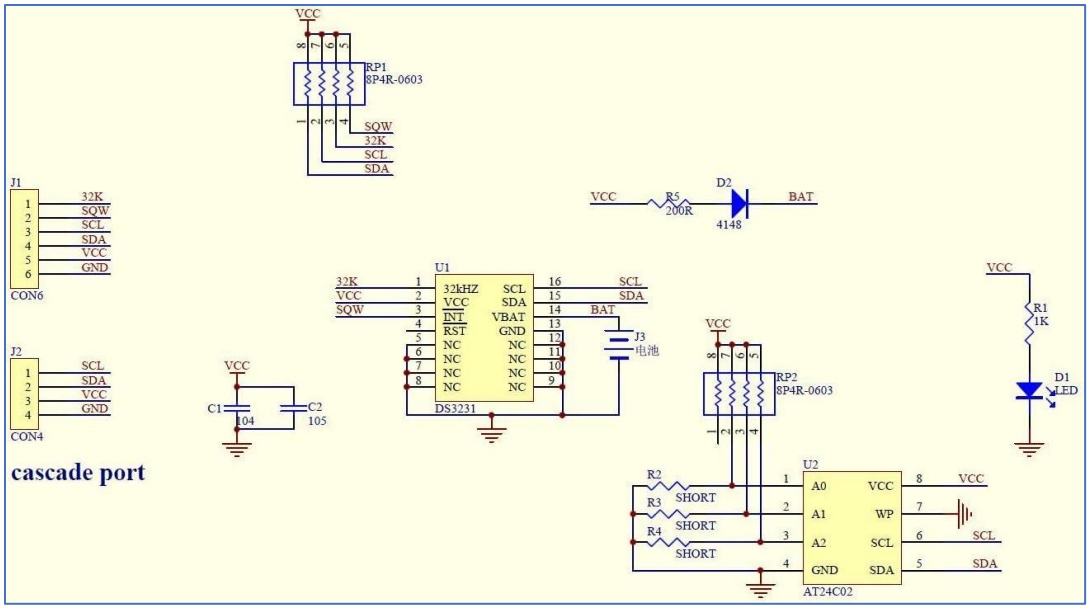


Figure 6.1.5-1. *RTC Pictures and Schematic.*

6.1.6 Keypad

The design uses a common flexible 16-key keypad. A picture and schematic are shown in Figure 6.1.6-1. The key pressed is determined by monitoring the row and column outputs. A specific item is: 4 x 4 Matrix Array 16 Key Membrane Switch Keypad Keyboard for Arduino/AVR/PIC, https://www.amazon.com/dp/B015M1CEIC?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1

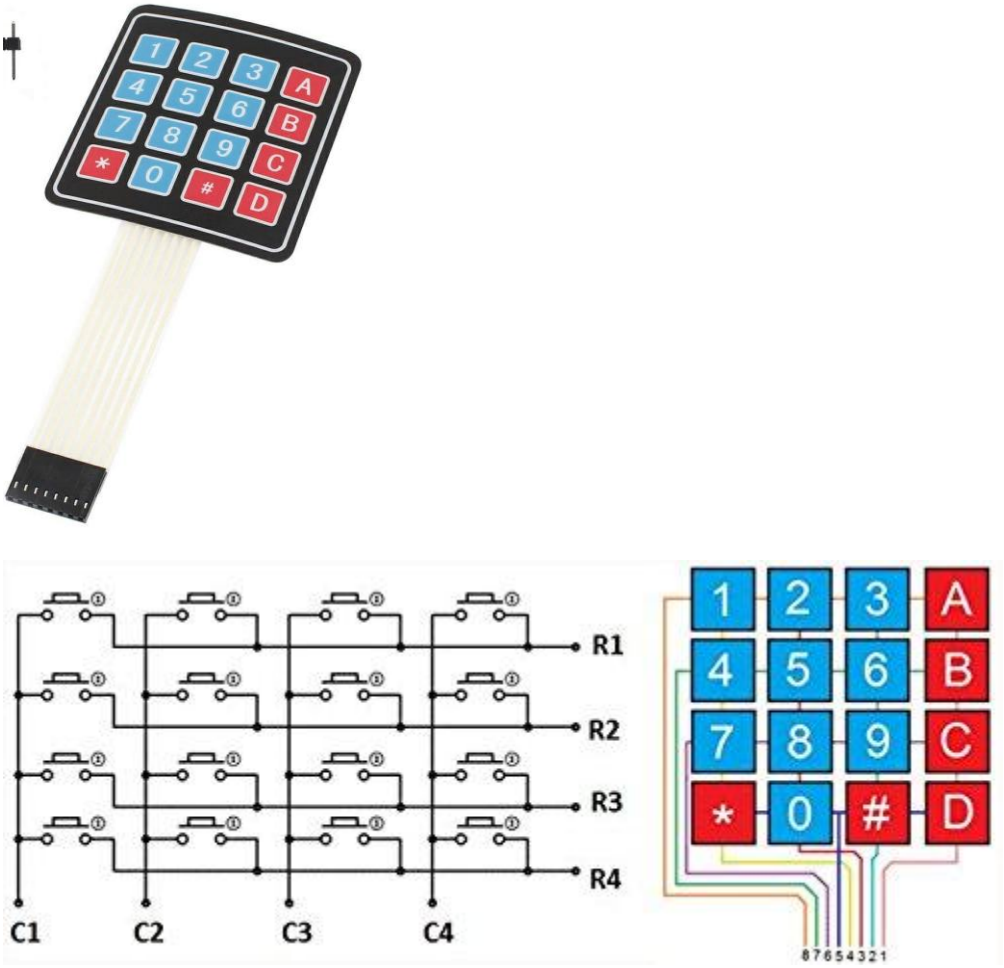


Figure 6.1.6-1. *Keypad Picture and Schematic.*

Keypad drivers are available for Arduino boards.

6.1.7 Relays

Initially I used a single relatively-large relay box designed specifically to control an actuator, OONO Forward and Reverse Relay Module for Motor/Linear Actuator, Reversing Relay Module (DC 12 V), https://www.amazon.com/dp/B0879GGVPZ?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1&th=1

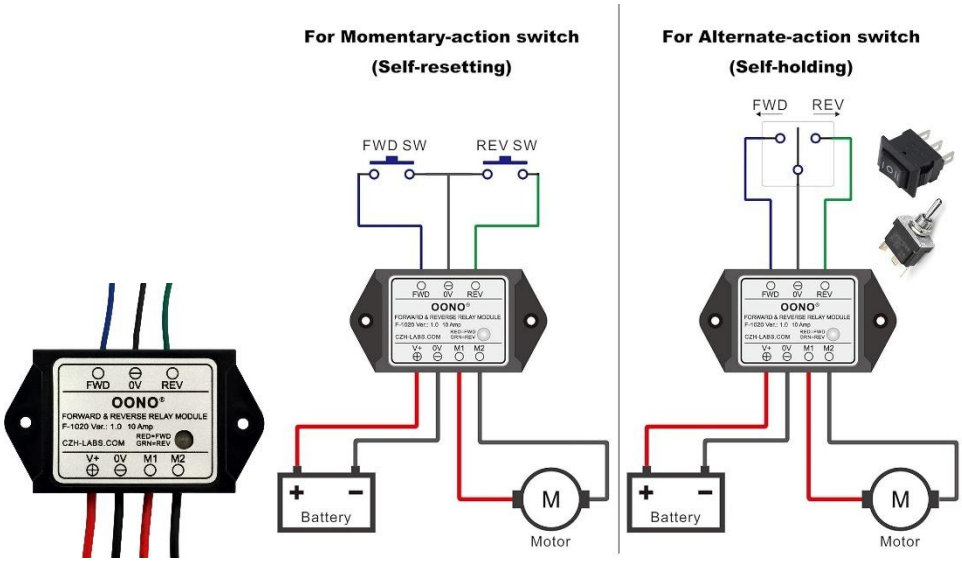


Figure 6.1.7-1. *OONO Relay Module (not used in final design).*

I was having problems with the 12 and 5-volt source stability when the relays and actuator turn on and off. The power glitches that result often caused the microcontroller board to crash or reset. I have tried many things to fix this problem. One thing I tried is replacing the OONO relay with 3 smaller individual relays to implement the same functionality. The 3 smaller relays did not completely fix the problem, but I am keeping that design because it uses less space and seems to not cause as many power glitch problems. I may be unfairly blaming the OONO relay. I later figured out that the main problem was that I was not grounding the power ground.

The relays I use are: Velleman 5V Relay Module Arduino Compatible, https://www.amazon.com/dp/B0CT5PBNFK?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1

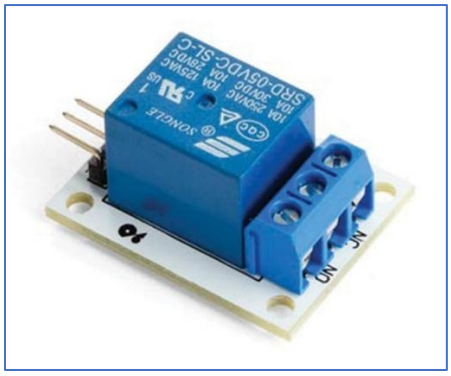


Figure 6.1.7-2. *Arduino-compatible 5V Relay.*

One relay is used to disconnect the 12 V power from the actuator (turn it off). The other 2 relays swap the polarity of the actuator output to control its direction (in or out). These relays have a red LED that turns on when the relay is activated (when the normally open (NO) side is

closed). Because the outputs are tied together, both relays need to be turned on or off before 12 V power is applied to avoid shorting the 12 V supply.

The biggest problem was that I did not have the ground input grounded to a solid and stable ground point. Adding a ground connection and additional capacitors on the 5V power fixed this problem.

6.1.8 Manual Tilt Control Switch

A tilt control switch is included in the design to allow manual control of the platform tilt angle. This switch performs a similar function to the Velleman relays and OONO relay module, only controlled by a switch instead of a digital input. The switch used is: Nilight Down Up Polarity Reverse Switch DPDT 20A 7PIN Momentary Rocker Switch ON Off ON Switch 12 V 24 V Toggle Switch Jumper Wires for Control Motor for Hoist, Crane, Linear Actuator, https://www.amazon.com/dp/B0BJ2C8TGC?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_7&th=1

A switch picture and connection instructions are shown in Figure 6.1.8-1.

Figure 6.1.8-1. *Manual Tilt Control Switch.*

6.1.9 On/Off Switch

The controller power is turned on and off using a simple single-pole-single throw (SPST) switch: Gardner Bender GSW-11 Heavy-Duty Electrical Toggle Switch, SPST, ON-OFF, ¾ HP 125-250 V AC, Screw Terminal, https://www.amazon.com/dp/B00004WLK5?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_4&th=1

Figure 6.1.9-1. *SPST Power Switch.*

6.1.10 Auto/Manual Switch

A triple pole, double throw switch (3PDT) is used to switch between auto and manual modes. Three poles are used to switch three connections between the manual switch and the auto circuit: 1) 12 V power, 2) Actuator Control A, 3) Actuator Control B. The actuator control must be switched to avoid connecting the different outputs together. This prevents shorting the supply and applying voltage to the inputs of the device not being used. The specific switch used is: mxuteuk Heavy Duty Rocker Toggle Switch 3 Position 3PDT 9 Terminal ON/Off/ON 15 A 250 V 10 A 380 V Toggle Switches with Metal Knob Cover Cap Waterproof TEN-303, https://www.amazon.com/dp/B084YQYJHC?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_5

6.1.11 Controller Housing

The controller housing is just big enough to jam in all the components. I chose a clear cover so you can see the display with the cover closed. The box should be water resistant. I use separate bulkheads (cable glands) to route the power cables in separately from the cables to/from the actuator and the encoder. Since multiple cables are going through the bulkheads, some special sealing of the bulkheads may be required to make them watertight.

I'm not giving details about how the components are mounted in the box, but you can see where most of the components are in the controller figures.

The specific box used is: Junction Box, Hinged Clear Cover IP67 Waterproof ABS Project Box with 2 NPT 1/2" Cable Gland, Electrical Box Enclosure with Mounting Plate & Wall Bracket 5.9 x 5.9 x 3.5inch(150x150x90mm), https://www.amazon.com/dp/B09F9383LH?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_4&th=1

Figure 6.1.11-1. *Controller Enclosure.*

6.1.12 Rotary Encoder

The rotary encoder selected for the design is: TWTADE/ 600P/R Incremental Rotary Encoder DC 5-24 V Wide Voltage Power Supply 6 mm Shaft AB two phases, https://www.amazon.com/dp/B07PLVGR1R?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1

Figure 6.1.12-1. *Rotary Encoder.*

There are several alternate similar encoders available. Driver code for this type of encoder is available for Arduino boards.

6.1.13 Rotary Encoder Housing

The rotary encoder needs to be mounted to the stationary part of the base with the encoder shaft connected by a wire to the rotating part of the base. The encoder needs to be mounted near the center of rotation. A box to house and mount the encoder is: Zulkit Junction Box ABS Plastic Dustproof Waterproof IP65 Universal Electrical Boxes Project Enclosure with Fixed Ear Black 3.27 x 2.28 x 1.30 Inch (83 x 58 x 33 mm)(Pack of 2),

https://www.amazon.com/dp/B0BRW7CV1P?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_20&th=1

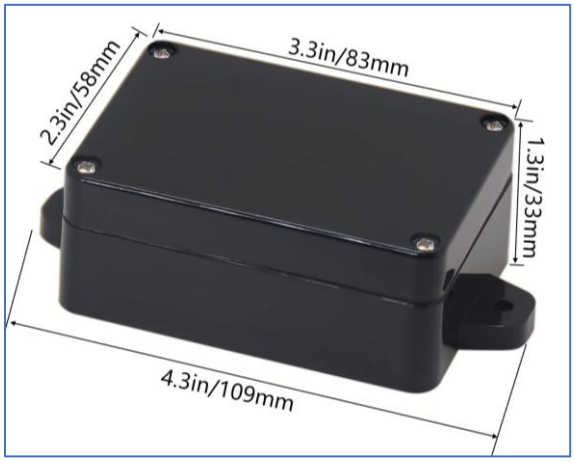


Figure 6.1.13-1. *Rotary Encoder Housing.*

A hole for the encoder shaft and holes for the mounting screws must be carefully drilled in the base of the box. A hole for the cables to pass through must also be drilled in the box.

6.1.14 Linear Actuator

I selected an Eco-Worthy linear actuator to do the platform rotation. I made a lucky guess about the size and power needed. The linear actuator needs to be mounted approximately perpendicular to the platform when it is not rotated. Geometry can be used to estimate the best locations to mount the ends of the actuator. Actuator parameters are shown in Figure 6.1.14-1.

The actuator used is:

ECO-WORTHY Heavy Duty 330lbs/1500N Solar Tracker Linear Actuator Multi-Functions with Mounting Brackets (12 V, 12") IP54 Waterproof 300 mm Stroke Linear Motion Actuator, https://www.amazon.com/dp/B00NM8H5SW?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_3&th=1

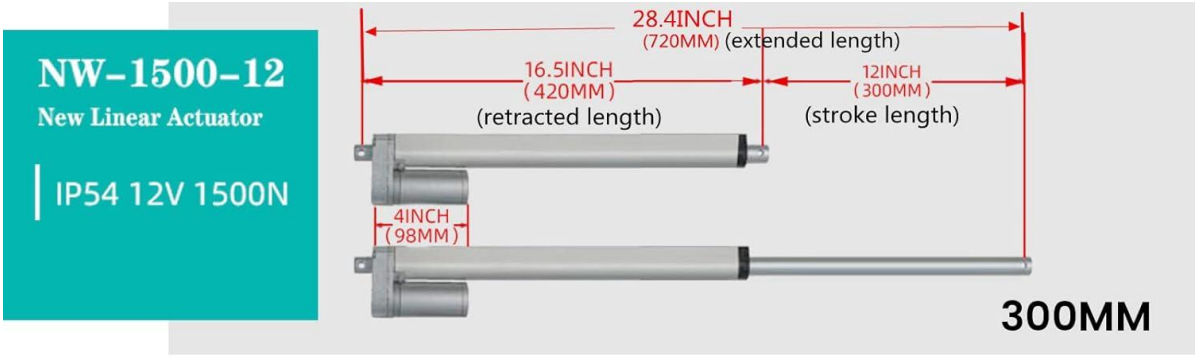


Figure 6.1.14-1. *Linear Actuator Parameters.*

6.1.15 48 V to 12 V DC-to-DC Converter

A voltage regulator is required to convert 48V battery bank power to the 12 V power needed by

the MEGA 2560. The nominal input voltage for the MEGA 2560 board is 9 V, but it is rated to a maximum input of 12 V. You should note that some of the 2560 board options have been reported as not having input regulators that last. This is probably true if used at the 12 V limit for an extended period.

Note that 48V is typically the minimum voltage for “48-volt” lithium battery systems. The voltage ranges between 48 V and 54 V. The 48 V reference value is used because it is a multiple of the standard 12 V car battery.

The regulator used in this design is: Nilight 48 V 36 V-to-12 V Voltage Converter 240 W 20 A Voltage Regulator Step Down to 12 VDC Waterproof DC to DC Converter Reducer Power Supply Transformer Module for Golf Cart Scooters, https://www.amazon.com/dp/B0CYQ8898J?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1&th=1

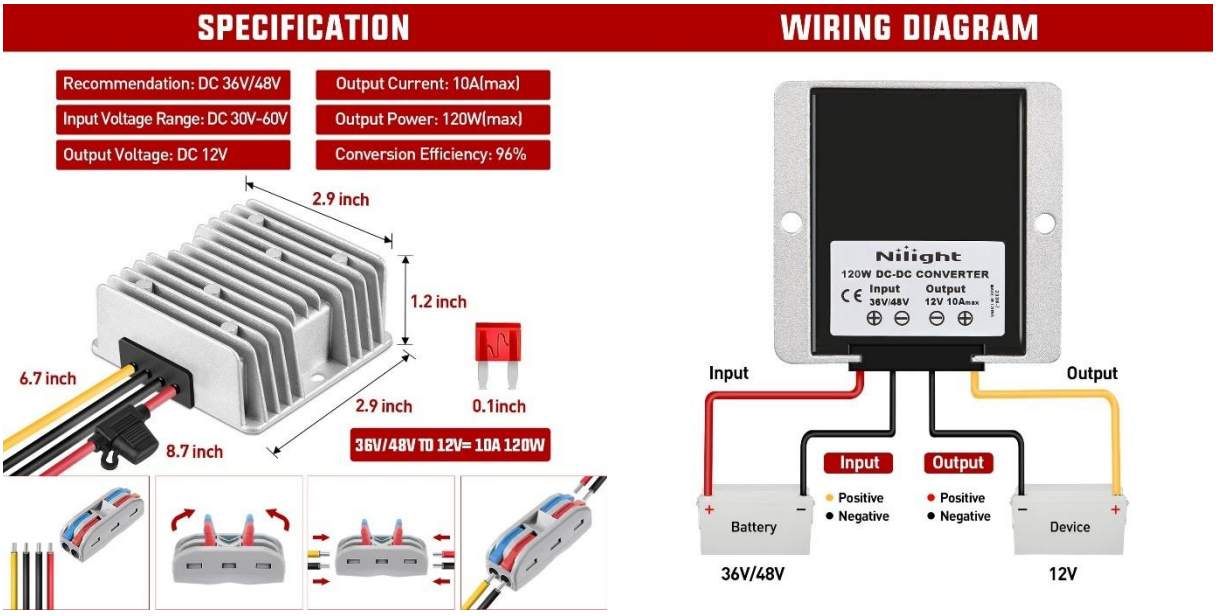


Figure 6.1.15-1. *DC-to-DC Converter.*

6.1.16 Miscellaneous Useful Tools

Some useful tools are:

1. portable drill and bits
2. screw drivers
3. tape measure and square
4. heat gun and/or soldering iron
5. wire cutters and crimpers
6. wrench set
7. multimeter
8. wood chisel and hammer
9. table, circular, radial arm, or miter box saw
10. laptop computer and USB cable.

6.2 Parts List Summary

Table 6.2-1 lists most parts and their approximate cost (August 2024). Links to specific products are included in the design component descriptions or after Table 6.2-1.

Table 6.2-1. *Parts List.*

Description	Unit Cost ($)	#	Total Cost ($)	Comments
Pressure Treated 2”x4”x8’ Lumber	4.58	10	45.80	Can probably find for <$$$
5/16" x 3" Hex Head Lag Bolts, 304 Stainless Steel 18-8	0.83	18	14.94	
5/16-18 x 3-1/2"Stainless Steel Hex Bolts & Hex Nuts & Large Diameter Flat Washers & Lock Washers, 304 Stainless Steel 18-8	1.38	8	11.04	
3/8" x 1" OD Flat Washer, 1" Outside Diameter, 0.080" Thickness, 18-8 (304) Stainless Steel	0.18	18	3.24	
Corner Brace, 4”x2” Galvanized L Angle Bracket	3.87	5	19.35	Heavy Duty
Stainless Steel L Bracket, Heavy Duty	0.47	5	2.35	1.5”x1.2”x1.2”
T-Strap Barn Door Hinges	3.00	2	6.00	
Stainless Steel 3-inch Door Hinges	2.50	2	5.00	
#8 x 1-1/2" Flat Head Sheet Metal Screws, 304 Stainless Steel (18-8), Phillips Drive, Self Tapping	0.11	15	1.65	Any screws included with hinges are not adequate
Braided Stainless Steel Wire*		3		Used scrap hangers from LED shop lights
Songhe MEGA 2560 PRO Embedded uController	17.99	1	17.99	with male pin headers
Solder Seal Wire Connectors	0.03	25	0.75	Heat Shrink Connectors
Ribbon Cable with 34-socket connectors	4.30	1	4.30	
Terminal Block	10.99	1	10.99	
LCD Display, 16 characters by 2 lines	2.78	1	2.78	
Real Time Clock Board	2.80	1	2.80	
Keypad, 4x4 membrane switch	5.00	1	5.00	
ELEGOO 120pcs Multicolored Dupont Wire 40pin Breadboard Ribbon Cables	6.98	1	6.98	Male to Female, Male to Male, & Female to Female
5 V Relay Module	5.88	3	17.64	
Manual Tilt Control Switch	8.27	1	8.27	
On/Off Switch	4.54	1	4.54	
Auto/Manual Switch	8.99	1	8.99	
Controller Housing (box)	15.79	1	15.79	
Rotary Encoder	16.68	1	16.68	
Rotary Encoder Housing (box)	8.99	1	8.99	
Linear Actuator	42.99	1	42.99	
DC-to-DC Converter	14.24	1	14.24	

Capacitors (electrolytic from old stock)				Connected to 5 V power
Component Mounting Standoffs and Screws*				
Wire and Connectors*				
Acrylic for mounting components*				
TOTAL:			**299.09**	Does not include tools and misc. parts from scrap

***Amazon Links:**

Lag bolts: https://www.amazon.com/dp/B09F67TV9R?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1&th=1

Stainless Steel Hex Bolt Set:
https://www.amazon.com/dp/B097T5D76M?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_10

Small Standoffs for Controller (2mm): 350Pcs M2 Male Female Nylon Hex Spacer Standoff Screw Nut Assorted Assortment Kit Threaded Pillar for PCB Circuit Board Motherboard Standoffs Black: https://www.amazon.com/dp/B0D7V6YQP7?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1&th=1, $13.00

Medium Standoffs for Other Components: 320Pcs M3 Motherboard Standoffs&Screws&Nuts Kit, Hex Male-Female Brass Spacer Standoffs, https://www.amazon.com/dp/B06Y5TJXY1?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_3&th=1, $15.00

Bullet Wire Connectors: 110pcs Heat Shrink Bullet Wire Connectors Kit, Waterproof Bullet Connectors, Bullet Terminals Wire Connectors Female & Male Wire Crimp Connectors, https://www.amazon.com/dp/B0CS6NH43Q?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_22&th=1, $8.49

Ring Connectors #10, 16-14 AWG: Haisstronica 100pcs #10 Blue Marine Grade Heat Shrink Ring Connectors, Tinned Red Copper 0.7 mm Ring Terminals Connectors, Brazed-Seam Insulated Electrical Crimp Wire Terminals, https://www.amazon.com/dp/B0B2J13QM6?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_2&th=1

Ring Connectors #10, 22-16AWG : Sanuke 105Pcs Heat Shrink Ring Terminals 0.8 mm Tinned Pure-Copper Marine Grade Waterproof Heat Shrink Wire Connectors Red, https://www.amazon.com/dp/B0CXXDFXNH?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_2&th=1

Solder Seal Connectors: 430PCS Solder Seal Wire Connectors, Durable Heat Shrink Butt Wire Connectors, https://www.amazon.com/dp/B0D6YHZD4V?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1&th=1

M3 Mounting Screw: 420pcs M3 Screw Kit, Premium M3 Button Head Socket Cap Screw Assortment with Nuts and Washers, M3 x 6mm /8mm /10mm /12mm /16mm /20mm /25mm (Silver), https://www.amazon.com/dp/B0CSX4L42C?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_7&th=1, $8.99

Solar Panel Connectors: Solar Connectors Male/Female IP67 Waterproof Solar Panel Cable Connectors (6 Pairs), https://www.amazon.com/YXGOOD-Connectors-Female-Waterproof-connectors/dp/B08P72QZFN/ref=sr_1_18?crid=164AOCT23T6KC&dib=eyJ2IjoiMSJ9.uLSWhQXgGqhG13FeUR3fkJnGrg_hTsnM9RKKIPOaQnWVAm7bWfTC04n70MxcOfpWxBB0RRsGZM5VlPELpVw3NsIKrK-ElAS6k2-eJXf4QJ_Ig43vjoTojXPjXS3BOoHuNXQT8I_IHm7kWckE0Yi_NGkT7VrP7lwEG72OHEjN1uN1vGx1Sp1apajQaky46jrr3NebEGE-1gkwRiAEJY_zhJ4s4zkUGnkAjWaSlexoH2Vy5GENU8JahQyuMM2jGGSEbNJboFZk-TuzBj9Jtup4z31gqoIDY-s4ulkWvzR10lKA3D53WgQVwtUdXSekSBYSLKaxhVwQXbxHinD_CBIwfIqmw7wobvw4ZhlWx93lCEo.j3VHYxtY8GwDAQ2GSlSl5iSooV50-rTbhxh7s7BCZ8w&dib_tag=se&keywords=Solar%2BPanel%2BConnectors&qid=1741727993&s=lawn-garden&sprefix=solar%2Bpanel%2Bconnectors%2Clawngarden%2C141&sr=1-18&th=1, $6.99

Solar Panel Mounting Racks: ECO-Worthy 2 Sets 45inch Adjustable Solar Panel Tilt Mount Brackets with Foldable Tilt Legs, https://www.amazon.com/dp/B0CP3QVV6B?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_3

Acrylic: I used small leftover pieces of ¼" thick acrylic sheets from another project. https://www.amazon.com/dp/B0CKXNW5QH/ref=dp_iou_view_item?ie=UTF8&th=1

Table 6.2-2. *Ribbon Connector Signal Allocation.*

Connector Pin	Processor Signal Designation	Allocated Signal Description	Wire Color/Destination Pin
1	Tx0/D0	Not Connected	
2	Tx0/D1	Not Connected	
3	D2	Encoder Phase a	White
4	D3	Encoder Phase b	Green or Blue
5	D4	Keypad	Red/2
6	D5	Keypad	Brown/8 (left pin)
7-12	D6-D11	Not used for serial LCD	
13	D12	On/Off Actuator Relay	Red
14	D13	In/Out Actuator Relay	Purple
15	D14	Keypad	Orange/6
16	D15	Keypad	Yellow/5
17	D16	Keypad	Green/4
18	D17	Keypad	Blue/3
18	D18	Keypad	Purple/2
20	D19	Keypad	Gray/1 (right pin)
21	SDA/D20	I2C Bus SDA (RTC & LCD)	Dark Gray
22	SCL/D21	I2C Bus SCL (RTC & LCD)	Gray

References

1. NOAA, General Solar Position Calculations, NOAA Global Monitoring Division.
2. Jean Meeus, *Astronomical Algorithms*, 2nd Edition, https://www.amazon.com/Astronomical-Algorithms-Jean-Meeus/dp/0943396611/ref=sr_1_1?crid=3M55B846TEO23&dib=eyJ2IjoiMSJ9.Vsa41KBjq5SksQBGt8JPtDyoOp-Tsl2DL0MLOgGcc4uD19imQQ-LOmSY5gXoPmC0.649SpXNu81kE8LtcGidz0XqEdrhZhe2CpRHd3qyrm68&dib_tag=se&keywords=jean+meesus+astronomical+algorithms&qid=1747858644&sprefix=jean+meesus+astronomical+algorithms%2Caps%2C177&sr=8-1.
3. Jean Meeus, *Astronomical Table of the Sun, Moon and Planets*, 3rd Edition Hardcover – January 1, 2016, https://www.amazon.com/Astronomical-Table-Sun-Moon-Planets-dp-1942675038/dp/1942675038/ref=dp_ob_title_bk.
4. Wikipedia, Position of the Sun, https://en.wikipedia.org/wiki/Position_of_the_Sun.

Appendix A. Solar Position Calculation

The solar position at any time and any observed location can be estimated by calculations described in this appendix.

Sun position algorithms are often derived from Jean Meeus's Algorithms (References 2 and 3). The specific algorithms implemented in the controller code use NOAA's equations. NOAA says the equations are derived from Jean Meeus's algorithms.

NOAA has online calculators, an Excel spreadsheet, and a document giving the algorithm used.

Here are the equations from the NOAA document (reference 1 pdf on the Internet):

First, the fractional year (γ) is calculated, in radians is

$$\gamma = 2\pi/365 * (\text{day\_of\_year} - 1 + hour - 1224).$$

(For leap years, use 366 instead of 365 in the denominator.)

From γ, we can estimate the equation of time (in minutes) and the solar declination angle (in radians),

$$eqtime = 229.18*(0.000075 + 0.001868\cos(\gamma) - 0.032077\sin(\gamma) - 0.014615\cos(2\gamma) - 0.040849\sin(2\gamma))$$

$$decl = 0.006918 - 0.399912\cos(\gamma) + 0.070257\sin(\gamma) - 0.006758\cos(2\gamma) + 0.000907\sin(2\gamma) - 0.002697\cos(3\gamma) + 0.00148\sin(3\gamma).$$

Next, the true solar time is calculated in the following two equations. First the time offset is found, in minutes, and then the true solar time, in minutes. The time offset is

$$time\_offset = eqtime + 4*longitude - 60*timezone,$$

where *eqtime* is in minutes, *longitude* is in degrees (positive to the east of the Prime Meridian), and *timezone* is in hours from UTC (U.S. Mountain Standard Time = –7 hours). Then

$$tst = hr*60 + mn + sc/60 + time\_offset$$

where *hr* is the hour (0 - 23), *mn* is the minute (0 - 59), and *sc* is the second (0 - 59).

The solar hour angle, in degrees, is:

$$ha = (tst / 4) - 180$$

The solar zenith angle (ϕ) can then be found from the hour angle (*ha*), latitude (*lat*) and solar declination (*decl*) using the following equation:

$\cos(\phi) = \sin(lat)\sin(decl) + \cos(lat)\cos(decl)\cos(ha)$

Then the solar azimuth (θ, degrees clockwise from north) is found from:

$\cos(180-\theta) = -\sin(lat)\cos(\phi) - \sin(decl)\cos(lat)\sin(\phi)$

For the special case of sunrise or sunset, the zenith is set to 90.833 (including the approximate correction for atmospheric refraction at sunrise and sunset, and the size of the solar disk).

Sunrise and sunset equations can be found in the controller code.

The size of the solar disk is about 0.5 degrees. The NOAA solar position accuracy is expected to be about 0.01 degrees.

The solar position algorithms are derived starting in elliptical coordinates. The position of a fictional object in a related circular orbit is related to the position of an object in an elliptical orbit. First equations defining the position in elliptical coordinates are found. Then coordinate transformations are used to convert to the position that would be observed from any location on the earth's surface and at any time. The sun's angular position from a location on earth is defined by its azimuth angle and its elevation (or depression) angle. These two angles, along with the range to the sun, define its position. Only the angles are needed to determine the desired solar panel platform pointing direction.

Note that the "equation of time" is more accurately called a position correction factor time adjustment. This time adjustment accounts for some of the time-dependent variation in the solar position equations.

More details can be found in the references.

Appendix B. Controller Code

```
/* SolarTrack Rev 1 - automatically controls solar base tilt angle
to track the sun for motion from sunrise to sunset.
John Smigel 10/13/2024 Updated 4/17/2025 */

/* example print to serial port:
  dtostrf(pointingErrorDeg,5,1,Txt);
  sprintf(Text,"pointingError: %s deg",Txt);
  Serial.println(Text); */

#include <math.h>
#include <Wire.h> // for RTC and I2C LCD

#include <LiquidCrystal.h> // include the library code

// for I2C display
#include <LCD.h>
#include <LiquidCrystal_I2C.h>

#define I2C_ADDR 0x27 //Define I2C Address where the PCF8574A is
#define BACKLIGHT_PIN 3
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7

#include <Keypad.h>

#include <DS3231.h>
DS3231 clock;
RTCDateTime dt;

void printParam(char* paramName, char* paramTxt);
void SolarAngles();
float Sunrise();
float Sunset();
int DayCalc(int monthin, int dayin, int yearin);
void TimeChangeDates();
void CalibrateEncoder();
void SetBaseAngleDeg(float setAngle);
```

```
bool useI2C = false;
bool autoRun = true;
bool displayOn = true;

//LiquidCrystal lcd(6,7, 8,9,10,11); // initialize the lcd library with the numbers of the interface pins (Dx)
//Above order is: RS,E,  D4,D5,D6,D7

//Initialize the LCD
LiquidCrystal_I2C lcd(I2C_ADDR, En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);

const byte ROWS = 4; //keypad dimensions: four rows
const byte COLS = 4; //four columns

byte rowPins[ROWS] = {5, 4, 14, 15}; // define the row pinouts (Dx) for the keypad pins
                                     (8,7,6,5)
byte colPins[COLS] = {16, 17, 18, 19}; // define the column pinouts (Dx) for the keypad pins (4,3,2,1)
char hexaKeys[ROWS][COLS] = { // define the symbols on the buttons of the keypads
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// Define the rotary encoder pin connections and other initialization
int CLK = 2; //CLK->D2 (interrupt 0 pin)
int DT = 3; //DT->D3
const int interrupt0 = 0; // Interrupt 0 is pin 2
static int myCount = 0; // Define the count
static float floatCount;
static int offset = 0; //was 200
static int lastCLK = 0; // CLK initial value

int sysMode = -1; // 0=setup, 1=run, 2=Measure (shows tilt&count), 3=East Home calibration, 4=show max/act lift, show max/act panel elevation
int defaultMode = 0; // mode to enter on boot or reboot; -1=no mode yet,0=setup,1=run
int datePrinted = 0;
int timePrinted = 1;
int startAnglePrinted = 1;
int endAnglePrinted = 1;
char date[20];
char time[20];
char startAngleTxt[20]; // display is 16 characters so max string size is 17
char endAngleTxt[20];
char debugTxt[25];
```

```
char inputString[25];
char myInput[20];
char myMonth[5];
char myDay[4];
char myYear[6];
char myHour[4];
char myMinute[4];
const char * months[] =
{"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
int monthNum;
int inputStringPtr = 0;
unsigned long delayValuems;
unsigned long startDelayValms = 256; // was 1000
int maxIterations = 4;
int maxNoMove = 14;
int numIterations;
int actuatorDirection;
int lastActDirection = -1;

int startAngle = -42.5;
int endAngle = 40.5;
float latitude = 41.322489; // 54 Corey Ln
float longitude = -72.225331;
float panelLength = 42.; //inches
float panelLift = 17.0; // inches
float maxLift; // calculated back panel lift for max elevation
float panelElevation;
float maxElevation;
float timeHours;
char latitudeTxt[18];
char longitudeTxt[18];
char latLongTxt[35];
char sunsetTxt[18];
char sunriseTxt[18];
char risesetTxt[35];
char Txt[180];
char Txt1[100];
char Txt2[100];
char Text[180];
int digitInt;

// for Solar Angles Calculations
int year;
int hour;
int minute;
int second;
```

```
int month;
int day;
float timezone; // Eastern Daylight savings time is -4, Eastern Standard Time (winter) is -5
bool standardTime;  // true if standard time, false if daylight savings time
// Clocks fall back to EST the first Sunday in Nov.; spring forward to Daylight Savings Time 2nd Sunday in March

const float dtor = M_PI/180.;
float latitudeRad;
const int monthDays[] = {31,28,31,30,31,30,31,31,30,31,30,31};
int dayOfYear = 0;
float eqOfTimeMin;
float decl;
float declDeg;
float fracYearRad;
float leapYear;
float solarZenithDeg;
float solarAzimuthDeg;
float sunriseHours;
float sunsetHours;
float solarNoonMin;
float solarNoonZenithDeg;
float sunsetHour;
float sunriseHour;
float sleepReSunsetHour = -1.; // time to sleep relative to sunset (hrs)
float lastEl;
float lastAz;
float lastTilt;
int lastMode = -1;
int springDay; // day of year to spring forward
int fallDay; // day of year to fall backward
bool initialized = false;
bool platformError;
bool sleeping;
bool azimuthOutOfBounds;
bool setupPrinted = false;
bool foundWest = false;
float azimuthLimited;
int systemError = 0;
int revision = 0; // code revision number

const int key[] = {1,4,4,0,2,5,0,3,6,1,4,6};
int myDayOfWeek;

int paramNum = 0; //0=date, 1=time, 2=start angle(deg), 3=end angle(deg)
```

```
bool encoderCalibrated = false;
bool enteringMode = false;
bool paramModePrinted = false;
bool enableDisplaySleep = true;
float angleGateDeg = 1.75; // gate tolerance for base pointing toward sun
float encoderAngleDeg;
float encoderErrorDeg;
float encoderScaleDPC = 0.33; // 0.35; //0.43; //0.4533;  //conversion from counts to deg(deg/count); encoder says 600 pulses per rev = 0.6 deg/pulse, seems to produce 794 to 1200 counts/360 deg
//Make above smaller to make tilt go farther
float pointingErrorDeg = 4.0;
float encoderHomeAngleDeg = -42.5;

int LINOnOff = 12; // actuator on/off select (0=off,1=on)
int LINInOut = 13; // actuator in/out (0=in,1=out)

// initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup() {
  //Common for both LCD types:
  lcd.begin(16, 2);  // set up the LCD's number of columns and rows
  lcd.print("Setup");  // Print current mode to the LCD
  // for I2C LCD below
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(HIGH);

  Serial.begin(9600);
  clock.begin();  // For RTC Initialize DS3231 (need device installed or loop will hang)
  //pinMode(CLK, INPUT); // for ELeggo encoder
  //pinMode(DT, INPUT);
  pinMode(CLK, INPUT_PULLUP);
  pinMode(DT, INPUT_PULLUP);
  attachInterrupt(interrupt0, ClockChanged, CHANGE);// Set the interrupt 0 handler(input D2), trigger level change

  // Linear actuator control
  // on OONO FWD=Blue wire=red LED = In, REV=Green wire=Green LED = out
  pinMode(LINOnOff,OUTPUT);
  pinMode(LINInOut,OUTPUT);
  digitalWrite(LINOnOff,LOW); // Off (Start with relays off)
  digitalWrite(LINInOut,LOW); // In/Down
}
```

```
void loop() {
 // float startAngleLimit;
 // float endAngleLimit;
  float el;
  float zEl;
  float sEl; // sun elevation at current time
  dt = clock.getDateTime(); // RTC handling
  if (!initialized) {
    SolarAngles(); // Zenith and Azimuth at current time
    TimeChangeDates();
    if (dayOfYear>=springDay&&dayOfYear<fallDay) {
      standardTime = false;
      timezone = -4.;
    } else {
      standardTime = true;
      timezone = -5.; // Eastern Daylight savings time is -4, Eastern Standard Time (winter) is
-5
    }
    sunsetHour = Sunset();
    sunriseHour = Sunrise();
    //startAngleLimit = startAngle;
    //endAngleLimit = endAngle;
    initialized = true;
    platformError = false;
    sysMode = defaultMode;
  }
```

```
  char customKey = customKeypad.getKey();
  if (customKey) {
    digitInt = customKey - '0';
    if (digitInt>=0&&digitInt<=9) {
      if (enteringMode) {
         sysMode = digitInt;
         enteringMode = false;
      } else {
        inputString[inputStringPtr] = customKey;
        inputStringPtr += 1;
        inputString[inputStringPtr] = 0;
        lcd.setCursor(0, 1);
        lcd.print("                "); // clear line
        sprintf(myInput,"%s",inputString);
        lcd.setCursor(0, 1);
        lcd.print(myInput);
        if (inputStringPtr>16) inputStringPtr = 0;
      }
```

```
  } else if (customKey=='*') {
    if (!displayOn) {
      displayOn = true;
      lcd.display();
      lcd.backlight();
    }
    inputStringPtr = 0;
    paramNum += 1;
    if (paramNum>3) paramNum = 0;
    switch (paramNum) {
      case 0: // date
        datePrinted = 0;
        break;
      case 1: // time
        timePrinted = 0;
        break;
      case 2: // start angle
        startAnglePrinted = 0;
        break;
      case 3: // end angle
        endAnglePrinted = 0;
       break;
    }
  } else if (customKey=='#') {
      switch (paramNum) {
      case 0: // date
        // Date format for setting: MMM DD YYYY (ex: Oct 27 2024), Time format: HH:MM:SS
        sprintf(time,"%02d:%02d:%02d",dt.hour,dt.minute,dt.second); // use current time
        Serial.println(time);
        // input format needs to be MMDDYY // convert input month to number/characters
        myMonth[0] = inputString[0];
        myMonth[1] = inputString[1];
        myMonth[2] = 0; // null terminator
        myDay[0] = inputString[2];
        myDay[1] = inputString[3];
        myDay[2] = 0;
        myYear[0] = '2';
        myYear[1] = '0';
        myYear[2] = inputString[4];
        myYear[3] = inputString[5];
        myYear[4] = 0;
        monthNum = atoi(myMonth);
        sprintf(date,"%s %s %s",months[monthNum-1],myDay,myYear); // use input date
        Serial.println(date);
        inputStringPtr = 0;
        lcd.setCursor(0, 1);
```

```
        lcd.print("                ");
        clock.setDateTime(date,time);
        dt = clock.getDateTime(); // RTC handling
        datePrinted = 0;
        break;
      case 1: // time
        sprintf(date,"%s %02d %4d",months[dt.month-1],dt.day,dt.year); // use current date
        Serial.println(date);
        myHour[0] = inputString[0]; // time input format is HHMM
        myHour[1] = inputString[1];
        myHour[2] = 0; // null terminator
        myMinute[0] = inputString[2];
        myMinute[1] = inputString[3];
        myMinute[2] = 0;
        sprintf(time,"%s:%s:00",myHour,myMinute);
        Serial.println(time);
        inputStringPtr = 0;
        lcd.setCursor(0, 1);
        lcd.print("                "); // clear line
        clock.setDateTime(date,time);
        dt = clock.getDateTime(); // RTC handling
        timePrinted = 0;
        break;
      case 2: // start angle
        startAngle = atoi(inputString);
        sprintf(startAngleTxt,"%d deg",startAngle);
        Serial.println(startAngleTxt);
        startAnglePrinted = 0;
        break;
      case 3: // end angle
        endAngle = -atoi(inputString);
        sprintf(endAngleTxt,"%d deg",endAngle);
        Serial.println(endAngleTxt);
        endAnglePrinted = 0;
        break;
    }
  } else if (customKey=='A') { // print latitude & longitude
    dtostrf(latitude,8,4,latitudeTxt);
    dtostrf(longitude,8,4,longitudeTxt);
    sprintf(latLongTxt,"%s/%s",latitudeTxt,longitudeTxt);
    printParam("lat/Lon:",latLongTxt); //latitudeTxt);
  } else if (customKey=='B') { // print sunrise and sunset
    dtostrf(sunsetHour,6,1,sunsetTxt);
    dtostrf(sunriseHour,6,1,sunriseTxt);
    sprintf(risesetTxt,"%s/%s",sunriseTxt,sunsetTxt);
    printParam("Rise/Set:",risesetTxt);
```

```
  } else if (customKey=='C') { // Enter command to mode
    enteringMode = true;
    sysMode = -1; // no mode so not printing
    if (!displayOn) {
      displayOn = true;
      lcd.display();
      lcd.backlight();
    }
    printParam("Enter Command","1:Run");
  } else if (customKey=='D') { // print elevation at solar noon & current tilt
    SolarAngles(); // Zenith and Azimuth at current time
    zEl = 90.-solarNoonZenithDeg;
    dtostrf(zEl,5,1,Txt);
    encoderAngleDeg = (myCount-offset)*encoderScaleDPC;
    dtostrf(encoderAngleDeg,5,1,Txt1);
    sprintf(Text,"%s/%s",Txt1,Txt);
    printParam("Tilt/Noon El",Text);
  } // key type checks
 } // key pressed

 if (sysMode==0) { // setup mode
   delay(500);
   if (!setupPrinted) {
     sprintf(Text,"Rev %d",revision);
     printParam("Setup ",Text);
     setupPrinted = true;
   }
   if (autoRun) {
     delay(5000);
     sysMode = 1;
   }
 }

 if (sysMode==1) {
   delay(1000); // this may make keyboard not respond
   if (!encoderCalibrated&&systemError==0) {
     CalibrateEncoder();
   }
   SolarAngles(); // Zenith and Azimuth at current time
   encoderAngleDeg = (myCount-offset)*encoderScaleDPC; // get encoder angle and sun
azimuth angle (DPC=DegreesPerCounts)
   //startAngleLimit = startAngle;
   //endAngleLimit = endAngle;
   azimuthLimited = solarAzimuthDeg;
   azimuthOutOfBounds = false;
   if (azimuthLimited>endAngle) {
```

```
    azimuthLimited = endAngle;
    azimuthOutOfBounds = true;
   }
   if (azimuthLimited<startAngle) {
    azimuthLimited = startAngle;
    azimuthOutOfBounds = true;
    sleeping = true;
   }
   if (sleeping&&timeHours>sunsetHours&&displayOn&&enableDisplaySleep) {
    displayOn = false;
    lcd.noDisplay();
    lcd.noBacklight();
   }
   if
((timeHours>(sunsetHour+sleepReSunsetHour)||solarAzimuthDeg<startAngle)&&!sleepin
g) {
    enableDisplaySleep = true;
    sleeping = true;
    if (systemError==0) {
     CalibrateEncoder(); // returh to home, calibrate & stop
    }
   }
   if (solarAzimuthDeg>startAngle&&sleeping&&solarAzimuthDeg<endAngle) { // Wake
    sleeping = false;
    displayOn = true;
    lcd.display();
    lcd.backlight();
   }
   pointingErrorDeg = encoderAngleDeg - solarAzimuthDeg;
   el = 90. - solarZenithDeg;
   if
(lastAz!=solarAzimuthDeg||lastEl!=el||lastTilt!=encoderAngleDeg||lastMode!=sysMode) {
    if (systemError==0) {
     dtostrf(el,5,1,Txt);
     dtostrf(solarAzimuthDeg,4,0,Txt1);
     dtostrf(encoderAngleDeg,5,1,Txt2);
     sprintf(Text,"%s/%s/%s",Txt,Txt1,Txt2);
     if (!sleeping) {
      printParam("Run-El/Az/Tilt",Text);
     } else {
      printParam("Slp-El/Az/Tilt",Text);
     }
     lastAz = solarAzimuthDeg;
     lastEl = el;
     lastTilt = encoderAngleDeg;
     lastMode = sysMode;
```

```
    }
   }
   if (abs(pointingErrorDeg)>angleGateDeg&&!sleeping&&systemError==0) {
    SetBaseAngleDeg(azimuthLimited);
   }
   if (systemError!=0) {
    sprintf(Text,"%d",systemError);
    digitalWrite(LINOnOff,LOW); // make sure relays are off
    digitalWrite(LINInOut,LOW);
    if (!sleeping) {
     printParam("Run-ERROR: ",Text);
    } else {
     printParam("Slp-ERROR: ",Text);
    }
   }
 } // End run mode 1

 if (sysMode==2) { // Measure mode
    SolarAngles();
    sEl = 90. - solarZenithDeg;
    encoderAngleDeg = (myCount-offset)*encoderScaleDPC; // offset is 200
//     dtostrf(,5,1,Text);
    dtostrf(encoderAngleDeg,5,1,Txt1);
    sprintf(Txt,"%s/%d",Txt1,myCount);
    printParam("Show Tilt/Count:",Txt);
 } // End of mode 2 (SHow tilt/count) code

 if (sysMode==3) { // East Home Calibration
   floatCount = encoderHomeAngleDeg/encoderScaleDPC + offset;
   myCount = round(floatCount); //encoderAngleDeg = count*encoderScaleDPC;
   encoderCalibrated = true;
   encoderAngleDeg = (myCount-offset)*encoderScaleDPC;
   dtostrf(encoderAngleDeg,5,1,Txt1);
   sprintf(Txt,"%s/%d",Txt1,myCount);
   printParam("Home Tilt/Count:",Txt);
 }

 if (sysMode==4) { // Panel lift max/act
   maxLift = panelLength*sin(solarNoonZenithDeg*dtor);
   dtostrf(maxLift,5,1,Txt1);
   dtostrf(panelLift,5,1,Txt2);
   sprintf(Txt,"%s/%s",Txt1,Txt2);
   printParam("Lift Max/Act.",Txt);
 }

 if (sysMode==5) { // Panel elevation max/act
```

```
  SolarAngles();
  panelElevation = asin(panelLift/panelLength)/dtor;
  dtostrf(panelElevation,5,1,Txt1);
  maxElevation = solarNoonZenithDeg;
  dtostrf(maxElevation,5,1,Txt2);
  sprintf(Txt,"%s/%s",Txt2,Txt1);
  printParam("Pnl El Max/Act.",Txt);
 }

 if (sysMode==6) { // Enter parameters
   if (!paramModePrinted) {
    sprintf(Text,"Rev %d",revision);
    printParam("Enter Parameters ",Text);
    paramModePrinted = true;
    enableDisplaySleep = false;
    sleeping = false;
    displayOn = true;
    lcd.display();
    lcd.backlight();
   }
 }

 if (!datePrinted==1) {
  sprintf(date,"%02d/%02d/%d",dt.month,dt.day,dt.year);
  printParam("Date",date);
  datePrinted = 1;
 }
 if (!timePrinted==1) {
  sprintf(time,"%02d:%02d:%02d",dt.hour,dt.minute,dt.second);
  printParam("Time",time);;
  timePrinted = 1;
 }
 if (!startAnglePrinted==1) {
  sprintf(startAngleTxt,"%d deg",startAngle);
  printParam("Start Angle",startAngleTxt);
  startAnglePrinted = 1;
 }
 if (!endAnglePrinted==1) {
  sprintf(endAngleTxt,"%d deg",endAngle);
  printParam("End Angle",endAngleTxt);
  endAnglePrinted = 1;
 }
} // loop

void printParam(char* paramName,char* paramTxt) {
 lcd.setCursor(0, 0);
```

```
 lcd.print("                ");
 lcd.setCursor(0, 0);
 lcd.print(paramName);
 lcd.setCursor(0, 1);
 lcd.print("                ");
 lcd.setCursor(0, 1);
 lcd.print(paramTxt);
}

void ClockChanged() { // Interrupt handler for angle encoder
 int dtValue;
 int clkValue = digitalRead(CLK); // Read the encoder CLK or A pin level
 if (lastCLK!=clkValue) {
  dtValue = digitalRead(DT); // Read the DT or B pin level
  lastCLK = clkValue; // counterclockwise rotation is negative
  myCount += (clkValue!=dtValue ? -1 : 1);// CLK and DT inconsistent: +1, otherwise -1
//   Serial.print("\ncount:");
//   Serial.println(myCount);
 }
}

// Sun Zenith and Azimuth angles
void SolarAngles() {
 float timeOffsetMin;
 float trueSolarTimeMin;
 float solarHourAngleDeg;
 float cosSolarZenithAngle;
 float cosSolarAzimuthre180;
 float solarNoonHours;
 float solarZenithRad;
 float denom;
 int leapYearInt;
 int mcount;
 year = dt.year;
 if (year%4==0) {
  leapYear = 1.0;
 } else {
  leapYear = 0.0;
 }
 leapYearInt = leapYear + 0.5;
 month = dt.month;
 day = dt.day;
 hour = dt.hour;
 minute = dt.minute;
 dayOfYear = 0;
 for(mcount=0;mcount<month-1;mcount++) { // accumulate days up to previous month
```

```
    dayOfYear += monthDays[mcount];
    if (mcount==1&&leapYear==1) {
      dayOfYear += 1; // account for 29 days in Feb leap year
    }
  }
  dayOfYear += day;
  fracYearRad = (2.*M_PI/(365.+leapYear))*(dayOfYear-1.+(hour-12.)/24.);
  eqOfTimeMin = 229.18*(0.000075+0.001868*cos(fracYearRad)-
0.032077*sin(fracYearRad)-0.014615*cos(2*fracYearRad)-0.040849*sin(2*fracYearRad));
  decl = 0.006918-0.399912*cos(fracYearRad)+0.070257*sin(fracYearRad)-
0.006758*cos(2*fracYearRad)+0.000907*sin(2*fracYearRad)-
0.002697*cos(3*fracYearRad)+0.00148*sin(3*fracYearRad);
  timeOffsetMin = eqOfTimeMin + 4.*longitude - 60.*timezone;
  trueSolarTimeMin = hour*60. + minute + second/60. + timeOffsetMin;
  solarHourAngleDeg = (trueSolarTimeMin/4.) - 180.;
  latitudeRad = latitude*dtor;
  cosSolarZenithAngle = sin(latitudeRad)*sin(decl) +
cos(latitudeRad)*cos(decl)*cos(solarHourAngleDeg*dtor); // direction of sun's rays from
vertical 0=overhead
  solarZenithRad = acos(cosSolarZenithAngle);
  solarZenithDeg = solarZenithRad/dtor;  // acos(cosSolarZenithAngle)/dtor;
  denom = cos(latitudeRad)*sin(solarZenithRad);
  if (denom>0.) {
    cosSolarAzimuthre180 = -(sin(latitudeRad)*cosSolarZenithAngle-
sin(decl))/(cos(latitudeRad)*sin(solarZenithRad));
  } else {
    cosSolarAzimuthre180 = 1.0;
  }
  solarNoonZenithDeg = acos(sin(latitudeRad)*sin(decl) +
cos(latitudeRad)*cos(decl))/dtor;
  timeHours = hour + minute/60. + second/3600.;
  solarNoonHours = (720.-4.*longitude-eqOfTimeMin)/60.+timezone;
  if (timeHours>solarNoonHours) {
    solarAzimuthDeg = -acos(cosSolarAzimuthre180)/dtor+180.;
  } else {
    solarAzimuthDeg = acos(cosSolarAzimuthre180)/dtor-180.;
  }
}

float Sunrise() {
  float horizonHourAngleRad;
  float sunriseHours;
  SolarAngles();
  horizonHourAngleRad = acos((cos(90.833*dtor)/(cos(latitudeRad)*cos(decl))-
tan(latitudeRad)*tan(decl)));
```

```
  sunriseHours = (720.-4.*(longitude+horizonHourAngleRad/dtor)-
eqOfTimeMin)/60.+timezone;
  return sunriseHours;
}

float Sunset() {
  float horizonHourAngleRad;
  float sunsetHours;
  SolarAngles();
  horizonHourAngleRad = acos((cos(90.833*dtor)/(cos(latitudeRad)*cos(decl))-
tan(latitudeRad)*tan(decl)));
  sunsetHours = (720.-4.*(longitude-horizonHourAngleRad/dtor)-
eqOfTimeMin)/60.+timezone;
  return sunsetHours;
}

int DayCalc(int monthin, int dayin, int yearin) {
  float zenith;
  int dayVariable;
  int intLeapYear;
  int myKey;
  int qLast2;
  myKey = key[monthin-1];
  intLeapYear = leapYear + 0.5;
  if (monthin==1) {
    myKey = intLeapYear-1;
  } else if (monthin==2) {
    myKey = 4-intLeapYear;
  }
  qLast2 = 1.25*(yearin-2000);
  dayVariable = (qLast2+dayin+myKey-1)%7;
  return dayVariable;
}

void TimeChangeDates() {
  bool springDateFound = false;
  bool fallDateFound = false;
  int yearDay;
  int currentMonth;
  int currentDay;
  int dayOfWeek;
  int monthEnd;
  int leapYearInt;
  leapYearInt = leapYear + 0.5;
  yearDay = 2;
  currentMonth = 1;
```

```
 currentDay = 2;
 SolarAngles();
 monthEnd = monthDays[0]; // Start with Jan
 while (yearDay<366&&!fallDateFound) {
  dayOfWeek = DayCalc(currentMonth,currentDay,year);
// Clocks fall back to EST the first Sunday in Nov.; spring forward to Daylight Savings Time 2nd Sunday in March
  if (currentMonth==3&&dayOfWeek==1&&!springDateFound) {
   springDay = yearDay;
   springDateFound = true;
  }
  if (currentMonth==11&&dayOfWeek==1) {
   fallDay = yearDay;
   fallDateFound = true;
  }
  yearDay += 1;
  currentDay += 1;
  if (currentDay>monthEnd) {
   monthEnd = monthDays[currentMonth];
   currentMonth += 1;
   if (leapYearInt==1&&currentMonth==2) {
    monthEnd = 29;
   }
   currentDay = 1;
  }
 }
}
```

```
void CalibrateEncoder() {
 int currentCount;
 int newCount;
 bool moving;
 bool tryingIn = false;
 moving = true;
 currentCount = 10000;
 myCount = currentCount;
 if (azimuthLimited>=endAngle) {
  digitalWrite(LINInOut,LOW); // actuator in(make sure can move some)
  tryingIn = true;
 } else {
  digitalWrite(LINInOut,HIGH); // actuator out(make sure can move some)
 }
 digitalWrite(LINOnOff,HIGH); // turn on actuator
 delay(3000);
 dtostrf(floatCount,10,6,Txt);  //seems to only work with prints below in - needed to get interrupt from encoder & change myCount
```

```
 sprintf(debugTxt,"\nmyCount: %d currentCount: %d",myCount,currentCount);
 Serial.print(debugTxt);
 digitalWrite(LINOnOff,LOW); // turn off actuator
 if (myCount==currentCount) {
  if (tryingIn) { //try other direction
   digitalWrite(LINInOut,HIGH); // actuator out(make sure can move some)
  } else {
   digitalWrite(LINInOut,LOW); // actuator in(make sure can move some)
  }
  digitalWrite(LINOnOff,HIGH); // turn on actuator
  delay(3000);
  dtostrf(floatCount,10,6,Txt);  //seems to only work with prints below in - needed to get
interrupt from encoder & change myCount
  sprintf(debugTxt,"\nmyCount: %d currentCount: %d",myCount,currentCount);
  Serial.print(debugTxt);
  digitalWrite(LINOnOff,LOW); // turn off actuator
  if (myCount==currentCount) {
   platformError = true; // Could not move platform
   systemError = 1;
   moving = false;
  } else {
   digitalWrite(LINInOut,LOW); // actuator in (move back towards home)
   digitalWrite(LINOnOff,HIGH); // turn on actuator
  }
 } else {
  digitalWrite(LINInOut,LOW); // actuator in (move back towards home)
  digitalWrite(LINOnOff,HIGH); // turn on actuator
 }
 while (moving) {
   delay(6000);
   dtostrf(floatCount,10,6,Txt);  //seems to only work with prints below in - need more
delay? Says not moving before stops moving
   sprintf(debugTxt,"\nmyCount: %d currentCount: %d",myCount,currentCount);
   Serial.print(debugTxt);
   if (myCount==currentCount) {
    moving = false;
   } else {
    currentCount = myCount;
   }
 }
 Serial.print("\nnot moving...\n");
 digitalWrite(LINOnOff,LOW); // turn off actuator
 floatCount = encoderHomeAngleDeg/encoderScaleDPC + offset;
 myCount = round(floatCount); //encoderAngleDeg = count*encoderScaleDPC;
 encoderCalibrated = true;
 encoderAngleDeg = (myCount-offset)*encoderScaleDPC;
```

```
}

void SetBaseAngleDeg(float setAngle) {
  float pointingErrorDegMag;
  int lastCount;
  int numNoMove;
  bool foundAngle;
  bool initialized;
  foundAngle = false;
  initialized = false;
  numIterations  = 0;
  numNoMove = 0;
  delayValuems = startDelayValms; // currently 200
  lastCount = myCount + 1; // make start different
  while (!foundAngle) {
//    sprintf(debugTxt,"\ncount: %d",myCount);
//    Serial.print(debugTxt);
    encoderAngleDeg = (myCount-offset)*encoderScaleDPC; // get encoder angle and sun
azimuth angle (DPC=DegreesPerCounts)
    if (lastCount==myCount) {
      numNoMove += 1;
    } else {
      lastCount = myCount;
    }
    if (numNoMove>maxNoMove) { // not moving found west limit for first time or error
      foundAngle = true; // stop if not moving
      if (foundWest) {
        systemError = 3;
      } else {
        if (actuatorDirection==1) {
          foundWest = true;
          endAngle = encoderAngleDeg;
        } else {
          systemError = 4;
        }
      }
    }
    pointingErrorDeg = setAngle - encoderAngleDeg;
    pointingErrorDegMag = abs(pointingErrorDeg);
    //dtostrf(encoderAngleDeg,10,6,Txt);
    //sprintf(debugTxt,"\nencoderAngleDeg: %s",Txt);
    //Serial.print(debugTxt);
//    dtostrf(pointingErrorDeg,10,6,Txt);
//    sprintf(debugTxt,"\npointingErrorDeg: %s",Txt);
//    Serial.print(debugTxt);
    if (pointingErrorDegMag>angleGateDeg&&numIterations<maxIterations) {
```

```
    digitalWrite(LINOnOff,LOW);  // Turn off actuator (while changing direction)
    if (pointingErrorDeg>0) { // requested angle is higher
      digitalWrite(LINInOut,HIGH); // actuator out
      actuatorDirection = 1;
    } else {
      digitalWrite(LINInOut,LOW); // actuator in
      actuatorDirection = 0;
    }
    if (!initialized) {
      lastActDirection = actuatorDirection;
      initialized = true;
    }
    if (actuatorDirection!=lastActDirection) { // check for direction change
      lastActDirection = actuatorDirection;
      numIterations += 1;
      numNoMove = 0;
      if (delayValuems>1) delayValuems>>1;
    }
    digitalWrite(LINOnOff,HIGH);  // Turn on actuator
    delay(delayValuems); // delay to allow some motion
    digitalWrite(LINOnOff,LOW);  // Turn off actuator
    delay(delayValuems); // delay to pause motion
  } else {
    foundAngle = true;
    digitalWrite(LINOnOff,LOW);
    digitalWrite(LINInOut,LOW);
  }
} //foundAngle while loop end
digitalWrite(LINOnOff,LOW); // make sure relays are off
digitalWrite(LINInOut,LOW);
if (numIterations>=maxIterations) {
  systemError = 2;
}
}
```

Appendix C.

Below is a summary of my technical training and awards:

Education/GE & Lockheed Martin Training Courses

- BSEE degree, Summa Cum Laude, 3.84 GPA, 1st in Class, University of Hartford (1980)
- MSEE degree with 4.0 GPA - Syracuse University, Syracuse, New York (1983)
- GE A, B, & C Advanced Courses in Engineering
- Edison Engineering Program
- Introduction to Sonar
- Sonar Space/Time Signal Processing
- Advanced Matrix Theory and Adaptive Processing (1982)
- Discrimination and Classification
- Digital Signal Processing (MIT)
- A Second Course in Signal Processing, 1988 (Cornell University)
- Fundamentals of Turbulence (1991, Syracuse University)
- Underwater Acoustic Modeling and Sonar Performance Prediction (1992)
- Submarine Operations
- C++ Programming
- Signal Classification and Recognition (1994)
- AN/SQQ-89 Sonar System (1999)
- Advanced Tracking Algorithms (2014, Lockheed Martin, Moorestown, NJ)

General Awards & Memberships

- CRC Chemistry Award for Highest GPA in College Chemistry
- President of University of Hartford Microprocessor Club
- Member, IEEE
- National Broadband Processing Working Group, Leader
- National Defense Industrial Association (NDIA) Technology Working Group
- Member, Syracuse NY Advanced Technology Working Group

Lockheed Martin/Martin Marietta/General Electric, Syracuse, NY

- US Patent 8,816,632 B2 Award, 2011 (Jointly with D. Winfield and F. Rotundo, Radio Frequency Power Transmission System)
- Trade Secret Awards, 2009
- Named Lockheed Martin Fellow, 2006
- Special Recognition Awards, 2004, 2010

- USS SRA Team Award, DD(X) Integration, 2005
- Department Award - Tech Ops, 2001
- Program Manager's Awards, 1997 & 1998
- ASTECS Proposal Extra Effort Award, 1995
- Recognition Awards for Valued Employees (RAVE), 1991 &1994
- Section Manager's Award, 1992
- Stock Option Award for Valued Technical Contributors, 1992
- General Manager's Award, 1991
- General Manager's Technical Excellence Award, 1988
- Named Engineer of the Year, 1988
- Graduated First in GE A, B & C Advanced Courses in Engineering (ACE)
- GE Edison Engineering Program Graduate

(Note: most awards and memberships in junior high, high school, and college are not listed)

6.3 Technical Autobiography – John Smigel

I first started having a passion for learning math, science, and English in about 6th grade (age 11). In 6th grade I was selected along with about 3 other students to form a special language skills class that met outside of normal classes. We trained outside normal class time on reading, information retention, and literacy.

However, I had a learning setback in 7th grade. I still remember the test we were given to determine which of the 3 "smartness" class levels to which we would be assigned. The levels were J, F, and K (for John F. Kennedy). J was for smartest, F was average, and K was below average. I do not do well with timed tests; I will not stop trying to answer a question until I'm sure it's correct. Therefore, I didn't finish all the test. I knew my parents were extremely disappointed that I was not assigned to the smartest level, but only to the middle, F, level. However, I did achieve many scholastic awards in junior high school and was near the top of the class.

I became interested in quantum and atomic physics, reading library books on these subjects. But I had a 7th-grade science project disaster. I created a simulation of an atomic chain reaction. This was a box containing mouse traps (set) and ping-pong balls, sealed with plastic wrap on top. Adding a single ball through a hole started the chain reaction. Unfortunately, older students stole my project from the classroom before it was evaluated and had much fun with it. I was scolded, disciplined by the nuns, and failed the project because ping pong balls and mouse traps were all over the school!

I also started excelling in math in 7th grade, Mrs. Dufault's class, fortunately not a nun & I had a bit of a crush. Whenever she asked any question, I would raise my hand, anxious to answer. She eventually told me, "Don't raise your hand any more John, I know you know the answer." I'm pretty sure I would have been considered a "savant" in math and "on the spectrum," equally

lacking in other skills, including social, geography, and art (the only D's I ever received were in geography and art from sister Nancy).

Unfortunately, the "smartness" group you were assigned in junior high school also determined your placement in high school. I was not initially assigned to the advanced, college-bound tier in high school, also to my parent's dismay. Rather quicky, my teachers recognized that was a mistake. I was only assigned to the basic geometry class with Mr. Argassi. I was so much above the other students that it became a problem for him. I always immediately knew all the answers and scored perfect on every test. The other students became aware of this and constantly tried to copy my test answers. Eventually, Mr. Argassi separated me from the class during tests and placed me in a different adjacent room. I graded other tests for him during the tests, rather than having to take any tests. He and my other teachers soon had me transferred to the advanced college-bound classes/curriculum.

I LOVED math, physics, chemistry, and biology in high school. I also developed a passion for computers and programming. At that time computers and programming were not used or taught in school, but the school was starting to plan to introduce them. There were not any "personal" computers yet, just some portable computers (basically large fancy calculators) and teletypes connected to remote main-frame computers using telephone dial-up modems. Apparently, teachers recognized my extreme nerdiness because I was singled out to use the new school computers. I did this after school. There was a stand-alone computer (probably a Wang, like my physics teacher, Mr. Leiper owned) in the back left of Mrs. Parton's room (Advanced Algebra Teacher). I was taught and trained how to use it. There was also a teletype terminal at the front right of our physics classroom. Mr. Leiper taught me how to use it. I was the only student taught to use the computers at that time (1974-1976).

I taught myself the BASIC computer language and became obsessed with a Star Trek program available on the teletype (being a Trekkie of course). The Star Trek game was originally developed in 1971: See history here: https://en.wikipedia.org/wiki/Star_Trek_(1971_video_game). You could enter commands like "Fire Phasors" and the program would tell you what happened. There was a random aspect to the game and what happened.

My obsession with computers and programming continued into college at the University of Hartford. I was in the Electrical Engineering program and quickly found the computer lab that contained a Data General Eclipse minicomputer, 2 teletypes, a line printer, a card reader, and a bunch of card punch stations. I immediately started learning everything I could about the equipment. I lived at the computer lab. I was there when it opened and they had to kick me out when it closed. I took the dozens of extra technical manuals they had home and had a stack of manuals about 4 ft high, and read them cover-to-cover. Fairly quickly, the computer lab

decided to hire me as a computer operator, my first real job. I helped start, shut down, and maintain the equipment. I also helped other students with programming and operating the equipment.

I wrote a Star Trek game version (derived from a high school version) that ran on the teletypes. At first there were no video terminals and all programs were entered with hole-punched cards. The center obtained one new computer interface device called a video terminal that had a video/CRT text display and a keyboard. At first only the operators were allowed to use the one new video terminal. I modified my Star Trek program to run on the video terminal and created video effects by writing and erasing characters all over the screen. The phasors were especially popular. I believe this was one of the first modern-style computer video games ever created. The head operator distributed my game (without asking me) to all other universities in the USA that had Eclipse minicomputers via the Eclipse user's group. I was told that my game (called Star T) became very popular at larger schools with multiple video terminals. Many years later, I went to a PC game museum exhibit in Syracuse, NY that featured what was credited as the first and most successful PC video game. It was a PC video Star Trek game version that was very similar to my Star T game. My latest game creation is JLMahJongg, an American Mah Jongg game.

After graduating 1st in my university class, I got a job with General Electric in Syracuse in their 3-year Advanced Course in Engineering (ACE). The ACE was considered the most grueling and difficult engineering training program in the country. This included getting a master's degree in electrical engineering at Syracuse University (SU). I continued to over-achieve, getting the highest grade on every weekly project and course. My weekly reports were saved and stored as the "model correct answer" for most projects. Other students hated me, especially in classes at SU where grades were on a curve and the instructor showed the grades. In SU matrix algebra class, I single-handedly caused students to drop out (fortunately not ones in the ACE).

My first 6-month rotating job assignment was a blast. I worked in test equipment engineering for Al Landry. I became his personal pet because I was good at BASIC computer programming, a new tool for him. I developed several computer programs for him to help manage the department, including an inventory manager and a construction lab scheduler/tracker. I designed and built (or had built by our model shop) many custom pieces of test equipment that were used to test the sonar and radar products we manufactured. I became the person interfacing with the model shop and tracking status using my BASIC programs.

My next assignment was developing a detailed combined radar and missile simulation to determine the most efficient techniques for radars to use against anti-radiation missiles (ARMs). After that I developed software specifications for sonar automated detection and tracking algorithms. I also worked on developing Enhanced Modular Signal Processor (EMSP)

technology. I implemented and sized key algorithms in the new technology. My next assignment was in Solid State radar (SSR) systems engineering on the AN/TPS-59 final integration and test team. I wrote the final technical document for the radar and helped perform radar calibration and first article flight tests, required for customer acceptance. Because of my job, I had special access to radar test data that was output using a special connection to a time-share computer. Radars at that time did not have the processing power to do real-time automated detection and tracking. All detection and tracking were done manually by the operators on the plan position indicator (PPI) display. I obtained raw radar output data and developed an automated detect and track program for the radar output, based on my sonar experience. This was new technology for radar and it helped find some unknown problems with our new AN/TPS-59 radar.

For most of the rest of my career, I worked in Advanced Engineering for Don Winfield, developing and evaluating new technologies. I developed several new breakthrough technologies based on adaptive beamforming. This included a new algorithm for passive submarine detection and a technique for evaluating towed array self-noise. I was still very young and unknown. My adaptive beamforming mentor, Dr. Ayhan Vural, thought my algorithm was so good that he presented it to the Navy at the Naval Underwater Systems Center in Connecticut (at that time – later moved to Rhode Island). I was shocked at the number of people who attended the presentation of my new algorithm (in an auditorium). We were discouraged from publicly disclosing any of our new technology. I submitted another patent disclosure to our patent attorney; but unfortunately, it was too late when it was discovered that he was negligent and not processing any disclosures into patents. He was immediately fired. That's why I don't have more patents. Eventually, another Navy lab engineer who attended the above presentation developed a similar non-adaptive algorithm that was implemented in all the sonar systems. It was considered one of the best new algorithms. It's probably still used. My version was even better.

I was assigned to a team working for the defense department to determine if a particular classified technology could be developed (and possibly used against us). We were responsible for the main design and construction. Another part of the team was from a government-contracted company (ARETE in beautiful La Jolla, CA) employing the country's smartest people in signal processing who were tasked with developing an optimal algorithm. I also developed and implemented our best algorithm and detailed test simulations. I remember checking the math of ARETE's algorithm at home in my spare time. It required one equation that was several pages long. My algorithm performed as well as ARETE's "optimal" one (again even better), with much less computational requirements. I won. The head of ARETE later flew across the country to meet me (at GE) and try to convince me to come work for them. I stayed with GE and was

responsible for signal processing implementation. I did the field system installation and testing on a submarine out of San Diego; and later aboard surface ships in the Atlantic and Adriatic.

I developed a reputation as an expert in advanced adaptive signal processing algorithms and able to perform seemingly impossible tasks. We teamed again with the Navy and GE corporate research and development to demonstrate a new massively parallel computer technology (iWARP). I was responsible for defining the adaptive algorithm for the iWARP demonstration. It was successfully tested on a submarine. A classified journal reviewed my algorithm as "one of the most significant advances ever achieved." Most of the technology I have developed is classified, so I can't describe it here, except in vague, unclassified terms.

Another project for which I received an award is a proposal effort for a new Navy shipboard electronic surveillance (ESM) system (called ASTECS in 1995). This was a new business for us and we were trying to break in to it. An external team of ESM algorithm experts were called in to define the algorithms and write the technical proposal. I was called in by the red team proposal reviewers to fix the technical proposal that was considered horrible. I worked with the experts to learn about ESM algorithms and rewrote the proposal. We won the proposal; and that started a major new multi-billion-dollar business in Syracuse that is still going.

When GE's defense business was sold to Martin Marietta, their San Diego office that developed new technology had just started a major joint program between the Navy and industry to develop a lightweight, broadband, variable-depth sonar (LBVDS). I had already developed a prototype environmentally-adaptive sonar processor called the Environmental Evaluation Processor (EEP) on my own time and with GE IR&D funds. My processor was selected as the core of the LBVDS processing and I became the processing and analysis lead for the industry-side of this multimillion-dollar project. It was like I was a rock star for a few years because this was the only large technology project funded by the office of naval research (ONR). My prototype processor was successfully built and demonstrated; leading to a full-scale production system that was also built and demonstrated. Unfortunately, the LBVDS technology was targeted for the new DD-21 destroyer that was won by our competitor, Raytheon.

My Martin Marietta employer was merged with Lockheed and became Lockheed Martin. Much of the sonar/radar technology and development was moved to other locations. I decide to switch from primarily sonar to primarily radar. This was difficult since I was not a radar expert at my job level. The most difficult and dreaded job is getting assigned as a large proposal author. I ended up assigned to 2 large radar proposals at the same time, a ballistic projectile classify radar and a large surveillance radar. I was responsible for key parts of both proposals: the classification algorithm and the anti-jamming (Electronic Counter-Countermeasures (ECCM) or Electronic Support Measures (ESM)) subsystem. The current classification algorithm from a partner company was not meeting requirements; so I developed a new algorithm that was

considered amazing. It is still used as a key component in this critical Army radar system. The radar ECCM area was relatively new to me. I had to rapidly become an expert to write my assigned proposal sections. We won the projectile-classifying radar proposal, but not the ESM one. However, the proposal sections were rated with colors from red (bad), yellow, green, blue, to purple (outstanding). On the proposal we lost, my ESM sections were the only ones rated purple.

I was scheduled to go away on vacation during this two-proposal period, but told my wife I could not go (In hindsight, a mistake). She went without me and it almost ended our marriage. On the positive side, my hard work and success on these proposals helped get me awarded a Lockheed Martin Fellowship. Being a Lockheed Martin Fellow is the highest technical honor awarded.

I continued working on new radar and sonar technology, including improving the new Space Fence radar algorithms and evaluating my classification algorithm performance for the ballistic projectile classifying radar mentioned above. I was also asked to develop improved technology for the US ballistic missile defense systems. I was lead systems engineer for part of a large (multi-billion, decades-long) multi-national project (with Italy and Germany) for a new ballistic missile defense system (approved unclassified description here: Media Release (long), MEADS FT2 (short), and Ad). Part of my responsibility was the environmentally adaptive algorithms, including ESM and algorithms in the surveillance radar (SR) to counter nuclear ballistic missile threats. The SR was my baby; I eventually was appointed lead systems engineer. MEADS was designed for the Army as a major improvement and replacement for the well-known and old Raytheon Patriot missile defense system. MEADS is not used (to my knowledge) for mostly-political reasons.