

Open CPV Solar Tracker Control v0.1

Dec 31st 2022

© 2021, Ruediger F. Loeckenhoff, MIT License

Summary: Highly precise and universal sensor-based solar tracker control with minimal hardware and full astronomic tracking for Concentrating Photovoltaic Systems.

Keywords: Solar, energy harvesting, robotics, sustainability, home automation, monitoring, weather

Open CPV Solar Tracker Control v0.1	1
1 About this project	2
2 A Brief Introduction to CPV and Solar Tracking	2
1.1 Concentrating photovoltaics CPV	2
1.2 Mechanics	3
1.3 Astronomical vs Sensor Tracking.....	6
3 The Board	7
3.1 Selector Pads	7
3.2 Connections	9
4 Code upload	10
4.1 Burning the bootloader	10
4.2 Uploading the code	10
5 First Startup	10
6 Encapsulation and light sensor.....	11
7 Calibration.....	12
1.4 Calibration_step 1: Set your parameters.....	13
1.5 Calibration_step 2: Find sun	14
1.6 Calibration_step 3: Correct elevation up limit	14
1.7 Calibration_step 4: Correct west	15
1.8 Calibration_step 6: Correct elevation down limit.....	15
1.9 Calibration_step 8: Start automatic calibration	15
1.10 Calibration explained.....	16
8 Operation	18
9 More to be done.....	19
10 Appendix.....	19
1.11 EEPROM Parameter List and Commands List	19
1.12 MIT License.....	26
1.13 Previous Work and Acknowledgements.....	26
1.14 References	26
1.15 Revision Index.....	27

1 About this project

I have a long professional experience in the field of High-Concentration-Photo-Voltaic (H-CPV) cells and systems. These systems need very precise dual axis sun tracking and I am not aware of any applicable open source solution. So, I decided to learn Arduino and actually invested months of my private time in this project. It was also a lot of fun. I really hope that it will be picked up by the CPV community and implemented in full size solar trackers.

The basic idea is, that the hardware can be very simple, if you use a compass to follow the calculated sun path. Whenever there is sun, a light sensor takes over to allow a very precise tracking better than 0.1° .

The other CPV trackers I have seen so far used limit switches and encoders which I believe is a bad idea. Such external sensors require cables which are expensive and which can break when the tracker is moving. A defective limit switch can destroy the tracker by making it move beyond limits. Furthermore, limit switches are hardly compatible with the operation in the tropics, because here the tracker needs to be able to turn both ways from the morning position (east) which requires an azimuth range of more than 360° .

In an earlier stage of this project I used an assembly of breakout boards including an MPU9250 6 axis compass and a DS3231 real time clock. This works very well.

(See <https://hackaday.io/project/185130>).

Yet this assembly couldn't really be considered a professional solution yet so I am very grateful that in 2021 Ermanno Antonelli joined the project to develop a completely integrated board. Our new control board integrates all components and it is tilting and turning together with the solar generator. All Eagle and CAD-CAM files are published together with this description, so you can order fully assembled boards from providers such as PCBWay. You just need to add a power supply and external cabling and of course you need to make it waterproof.

The sophisticated serial interface allows adapting the control to the tracker mechanics and motors. E.g. I developed the control on a 3D-printed table-top tracker, I tested it on a real 11 m^2 steel tracker and the motor controller should have enough power for a 100 m^2 tracker. The code is well documented and I hope for comments and improvements.

Apart from that, it is also a great and well documented robotics project and of course it also works for conventional PV – yet my heart beats for CPV.

Please see the videos at

<https://youtu.be/5Q2Ws1LjQEw> and <https://youtu.be/nsTUIzNMmE>

2 A Brief Introduction to CPV and Solar Tracking

1.1 Concentrating photovoltaics CPV

In CPV-Systems lenses or mirrors focus the sunlight on tiny solar cells with ultra-high efficiency. Typically, the concentration ratio is in the range of 300-2000 times the energy density of direct sunlight. Only direct sunlight can be focused and the focus moves away from the solar cell when the system is not perfectly aligned to the direction of the direct sunlight. The

aligned (left) and the misaligned (right) cases are sketched in Fig. 1. Typically, a CPV solar tracker should have a precision of 0.1° or better.

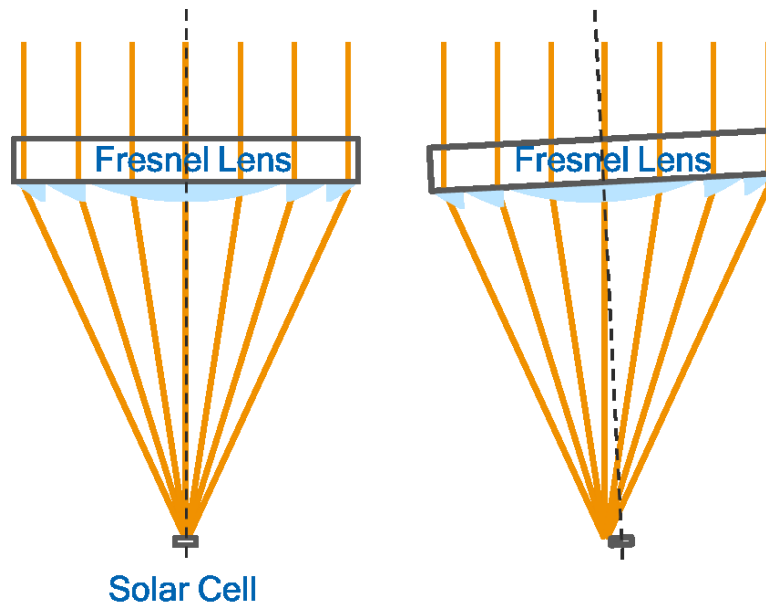


Fig. 1: Aligned and misaligned CPV module

1.2 Mechanics

This project is not primarily about building solar tracker mechanics. But I should explain what kind of solar tracker is suitable for this control and also share some of my knowledge. The present version of the software assumes an “azimuth drive” with a vertical axis which is fixed to a foundation (in case of a real solar tracker) or e.g. a wooden base plate (in case of a table top demonstrator). In other words: the azimuth drive adjusts the compass direction. On top of this azimuth drive there is a second “elevation drive” with a horizontal axis. It adjusts the pointing direction of the tracker above the horizon. We will call this angle above the horizon “elevation”. The direction of the horizontal elevation rotation axis is changing, as the vertical axis is turning.

E.g. 0° azimuth is north. When the sun is coming up in the east, the elevation is 0° and the azimuth is 90° . When the sun is close to Zenith the elevation is $\sim 90^\circ$ and the azimuth is changing rapidly.

The simplest and cheapest configuration for a demonstrator that I can think of consists of 2 12V 2 RPM motors as shown in Fig. 2, left. Depending on your toolbox you can use 3D-printed parts or wood and glue to build a solar tracker around these two motors. In case of the azimuth drive it is a good idea to add some friction, e.g. with felt pads, to suppress the backlash and prevent oscillations. For the elevation drive you might use some counterweights.

In the description of the EEPROM parameters below you will see that you can set the speed of the up and down movements for the elevation separately. This allows you to have equal up and down speeds and step sizes even if the elevation is not balanced and under load.

You can see the cross-shaped shading beam in front which is casting a shadow on 4 LEDs which act as light detectors. This kind of tracking sensor is very easy to make and it will work. During the development of this control I have been working with a very compact imaging tracking sensor which I have developed for my employer and which allows a tracking preci-

sion of 0.01° or better for clear weather conditions[1]. Please contact me, if you should be interested.



Fig. 2: Simple table top tracker and my table top tracker made from 3D-printed parts and Fischertechnik.

I have built the tracker on the left-hand side of Fig. 2 and it works. However, I wanted test hardware which is similar to real solar trackers that usually use a slew drive for azimuth and a linear drive for elevation. So, I also built the tracker on the right hand side. It uses Fischertechnik wheels for azimuth and a small linear drive for elevation. The shading beam is made from cardboard because weight is an issue for the little linear drive. In this case the tracker is powered with a USB-12V boost converter cable which proved to be very handy.

I claim that this control is so universal, that it could track a toy tipper truck to the sun which is running in circles and moving its platform as in Fig. 3. It would be great if someone could prove me right.

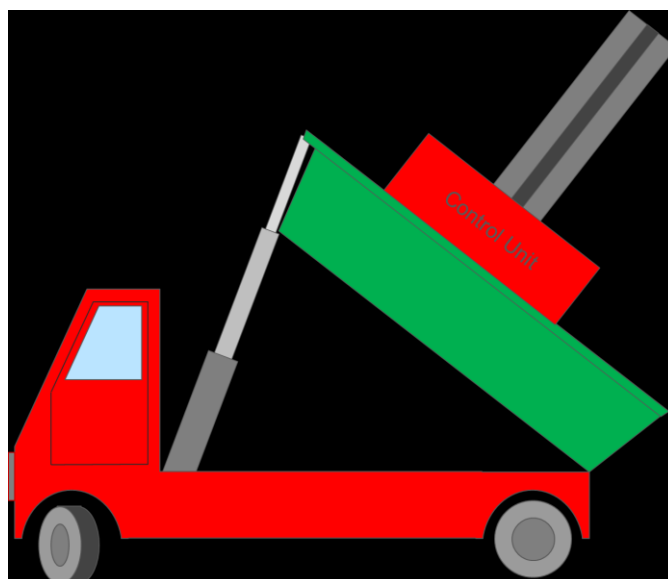


Fig. 3: Toy tipper truck tracker. Please, would someone build it for me?

There are a lot of servo-based Arduino solar tracker projects. These are great for educational purposes but impractical for big industrial trackers. Servos require potentiometers and it is hard to find a suitable place for them within the tracker mechanics. They are also not all to reliable when exposed to moisture.

As far as real applications are concerned, I strongly believe in a 3 wheeled solar tracker with a winch for the elevation movement. Used car wheels are a resource which is readily available anywhere in the world. They can carry tons and they can run on uneven ground such as gravel. Even a badly worn car wheel is still good enough for a solar tracker. It will not run more than 10 meters a day or 3.6 kilometres per year. If it should finally burst you just need to bring your jack and another old tire for repair. Anyone can do that anywhere in the world.

The winch is working against gravity. This means that it is preloaded and thus works with zero backlash.

Fig. 4 is a picture of a patent application that I have filed and which is now discontinued and free for anyone to use. If you want to build this tracker, please have a look at Fig. 5 and get in contact with me such that I can share some more thoughts.

DE 10 2019 004 468 A1 2020.12.31

Anhängende Zeichnungen

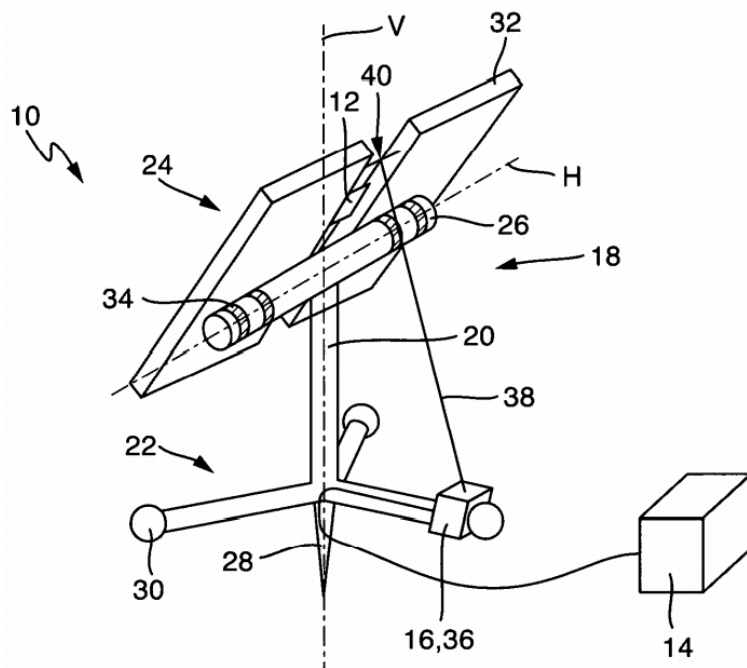


Fig. 4: Wheel and winch tracker according to a discontinued patent application.

Wind load is always an issue for solar trackers. They need to go to stow when the wind is faster than e.g. 20 m/s. The winch may be designed in such a way, that the wire is wound around the wheel and pre-tensioned by a weight. It thus can slip in one direction but not in the other. Even without electrical power the wind will move the generator and the wire will always keep tensioned just like when a sailor tightens a rope around a bollard to pull a huge ship closer to the quay wall with each wave. Please see the sketch below.

Actually, this tracker concept made me understand that a tracker control should not depend on precise mechanics. It should orient itself by magnetism, gravity and of course the rays of the sun. Thus, the dual axis tracker mechanics can be whatever, as long as they are able to move in small steps and as long as the precision is not compromised by backlash.

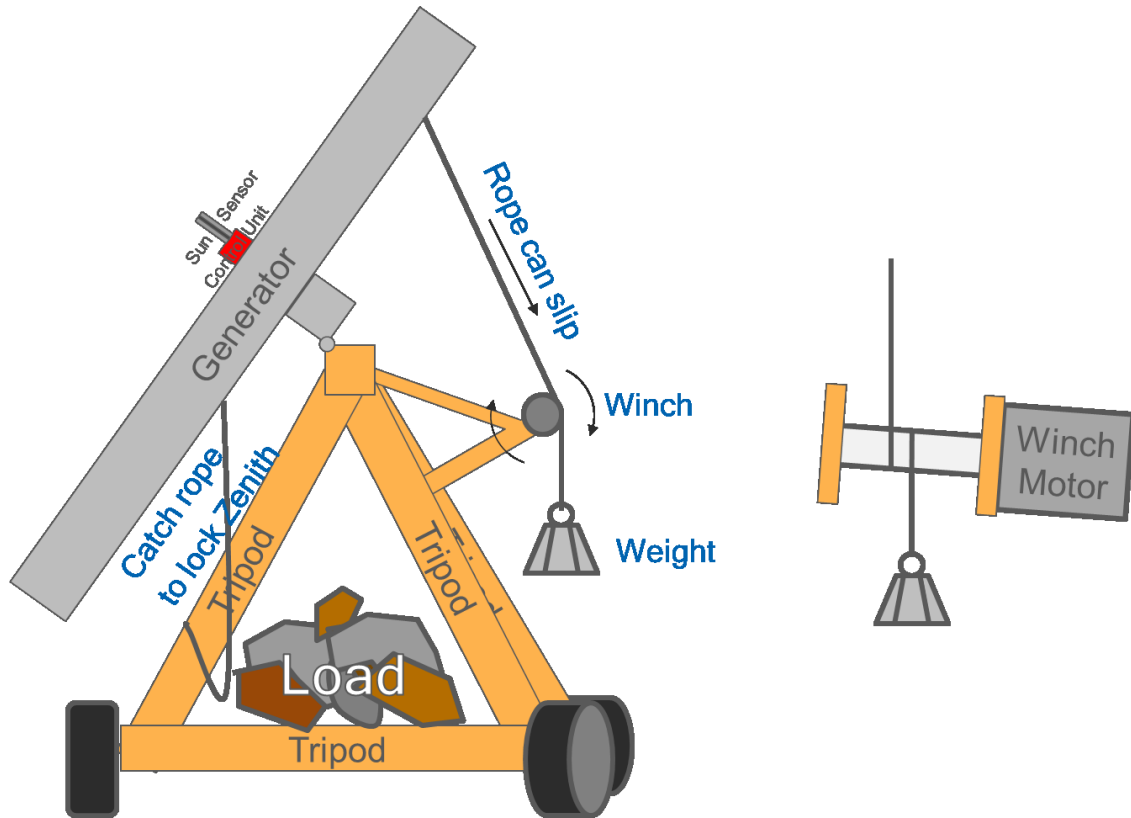


Fig. 5: One of my preferred arrangements

1.3 Astronomical vs Sensor Tracking

There are two basic approaches to solar tracking. You can calculate the sun position from time, date, longitude and latitude and move the tracker to the calculated position. This is called **astronomic tracking**. For a precise tracking the mathematical formulas can be quite complex. Here we can use a simplified version [3]. On the other hand, there is **sensor-based tracking** where light detectors sense the direction of the sunlight. In this project we follow a **hybrid tracking** approach where the tracker moves to the calculated direction whenever there is no sufficient light for sensor-based tracking. As soon as the amount of light on the detector surpasses a certain level, the control switches to sensor-based tracking mode.

Astronomic Tracking with encoders: The most common way to assume the calculated orientation is to drive each axis against a limit switch and count encoder steps from there [4]. This can take a lot of time since trackers tend to be slow and you need for the encoder calibration each time you power on or reset the tracker. You will also need to set the encoder-steps per degree for an exact calculation. If a limit switch fails, the tracker can self-destruct if you have not added an overpower protection or an automatic shutoff for out of range tracker orientations.

Astronomic Tracking with servos: This is a very common and simple approach for Arduino projects. However, you will not find a servo big enough to move a 50 m² tracker.

Compass based astronomic tracking: Actually, this is the main idea behind this project where a 9 axis compass measures tilt and compass direction. The drives are operated by simple 12 or 24V DC motors.

The tilt is calculated from the gravitation in x and z. This tilt sensing is quite precise and the gravitational field is very reliable.

Getting the compass direction from the magnetic field is much trickier. The magnetic vector usually is not parallel to the ground, it is weak and the magnetic sensors have quite a large zero drift. Furthermore, solar trackers are commonly made out of magnetic metal which distorts the earth magnetic field and the DC motors add more errors. Therefore, I had to find and program a calibration routine which corrects for all of those errors. You can find it described below in the section which concerns the first start-up.

3 The Board

Please have a look at the rendering of the board in Figure 6. In particular, the labels on the backside are important that show the use of the pins.

We published a complete set of Eagle and CAD/CAM files together with a bill of materials (BOM) for the tracker control board in Version V2.0_f at

https://www.pcbway.com/project/shareproject/Open_Concentrator_Photovoltaics_CPV_Solar_Tracker_Controller_Board_18449b11.html.

PCBWay has successfully assembled functional boards based on this data-set. The repositories also include Eagle files for a programming and test board which acts as a shield for the Arduino UNO. The repository includes all information that you need for an order at PCBWay.

3.1 Selector Pads

There are several selector pads that let you adapt the board to certain conditions.

SENS | LED - Choose the type of light sensor

You will need to choose which kind of light sensor you want to use. Usually you will want to use the LEDs which act as photo diodes. Then you have to make a solder bridge between the middle and left pad beneath SENS | LED. If you connect middle to right, the board is set up for the other option that uses an imaging light sensor with a better resolution. Contact me at Ruediger.Loeckenhoff@gmx.net, if you are interested in the imaging sensor.

VDD/SER – Power through serial adapter, use with care

Usually the VDD from the serial adapter (FTDI) is not connected to the VDD of the controller board. If you close this gap with a solder bridge, you will establish that connection. Please mind that this can lead to problems and even destroy the tracker control board if the voltages are not at the same levels.

P | PB which version is your ATMEGA328?

P is the old and PB is the new version of ATMEGA328. There are some differences concerning the pins, so please make solder bridges on the right hand side for PB and on the left hand

side for P. Usually it will be the newer PB chip, which will give you an additional serial port at PE0 and PE1. These pins can also be used as general digital pins. If you don't need those pins, you can leave these selector pads as they are. If you are using an ATMEGA328p you must make solder bridges between the middle and left pads.

ICSP | RUN

For ICSP programming (bootloader) you need to make a bridge on the left hand side to supply the ATMEGA328 with 5V from the programmer. After this, please open this connection and make another bridge on the right hand side underneath RUN to supply the ATMEGA and other chips with 3.3V from the onboard regulator. If you are using the pogo board for programming and testing, you don't need to make these bridges yet since the selection will be handled from the pogo board.

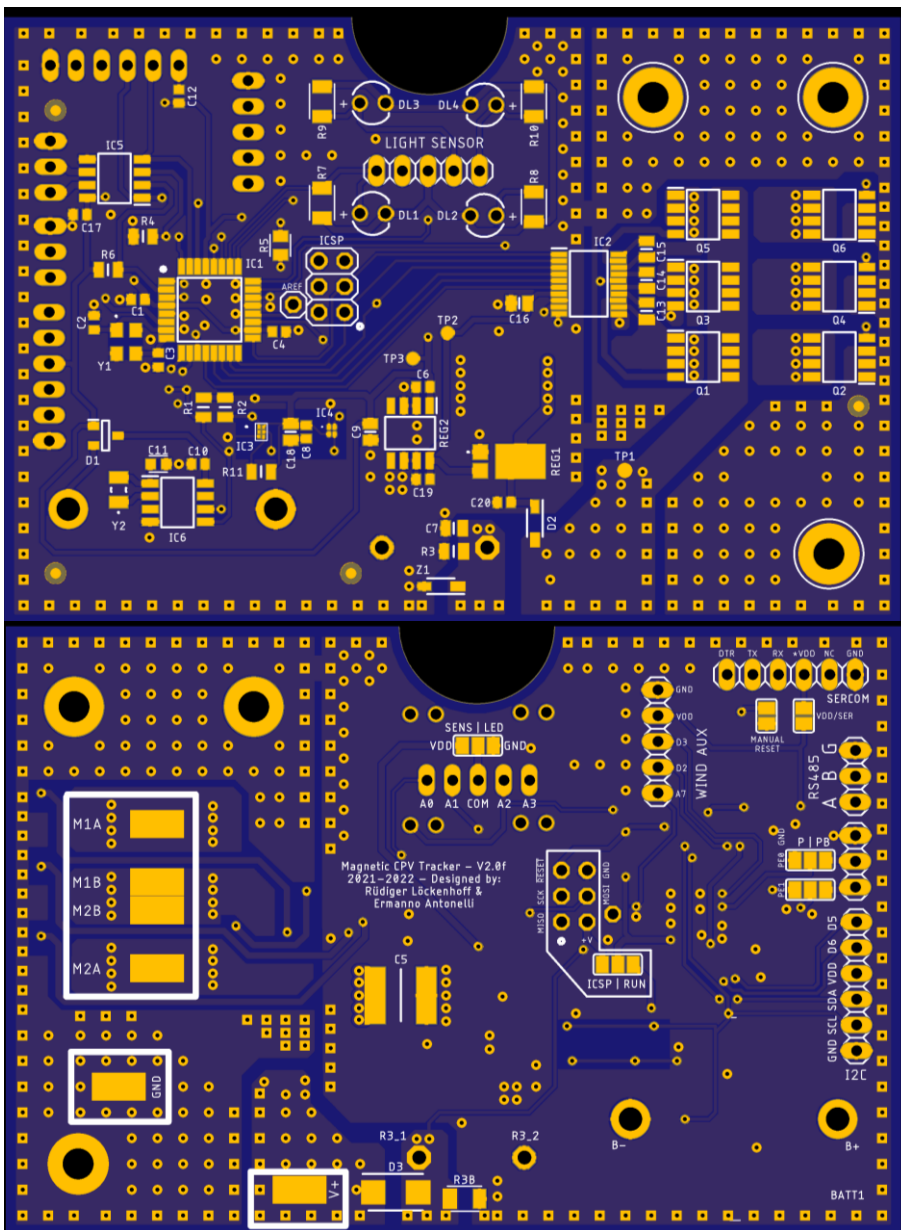


Figure 6: Front and back side of the board with labeled pins.

3.2 Connections

M1A M1B: Azimuth Motor

Solder the wires of the Azimuth motor here. Please make sure to figure out the polarity first. You can do this in calibration step 3, where you have manual control over the motors.

M2A M2B: Elevation Motor

Same for the Elevation motor.

V+ GND: Power

Connect +12V to +36V To V+ and GND to GND. Voltage reversal may destroy the board. There may be a certain degree of voltage reverse tolerance because of the TVS diode. Don't count on it.

SERCOM: Serial Communication

Connect a serial adapter (FTDI) here for serial communication with the computer. The pins are in the usual order. Make sure to set the serial adapter to 3.3V.

WIND AUX (D3 and GND)

“Real” solar trackers are required to go to stow when the wind is getting faster than e.g. 20 m/s. I am assuming that you will use a wind wheel which closes a reed relais several times per rotation. In this case, connect the wires from the relais between D3 and GND. The wind speed for go to stow is stored as the ASCII sign **v** in the EEPROM. Please refer to the EEPROM Parameters List.

There are several more connections that you can use. For instance you might have a wind sensor that genereates a voltage rather than pulses. You can connect this to A7, but you would have to adjust the Arduino code.

RS485 A B G: RS485 Field Communication Port

The board is ready for half duplex RS485 communication. However, it is not implemented in the code yet. First tests (answer ping with pong) have been successful. I.e. the hardware is all there but you will have to extend the Arduino code to communicate via RS485.

SDA SCL: I2C Lines

These are connected to A4 A5 which are the native I2C pins of the ATMEGA328.

D2 D5 D6 ARef: Additional Ports for General Use

These are unused pins of the ATMEGA328 with their usual Arduino labels. We made it a rule to make all connections available that may be useful for alternate applications of this board.

4 Code upload

The whole process using the pogo board is described in a video:

<https://youtu.be/nsTUlizNMmE>

I am using Arduino 1.8.7. Other versions probably work too, but I didn't test them.

4.1 Burning the bootloader

Upload ArduinoISP from the Arduino examples onto the Arduino UNO of the pogo board. Install minicore. Set the parameters displayed in Fig 7. Connect the tracker board to the pogo board. Alternatively, if you don't have the pogo board, select ICSP with a solder bridge and use an ICSP programmer connected to the ICSP pin header.

Select the correct COM port. "Burn bootloader". If it doesn't work, check the connections.

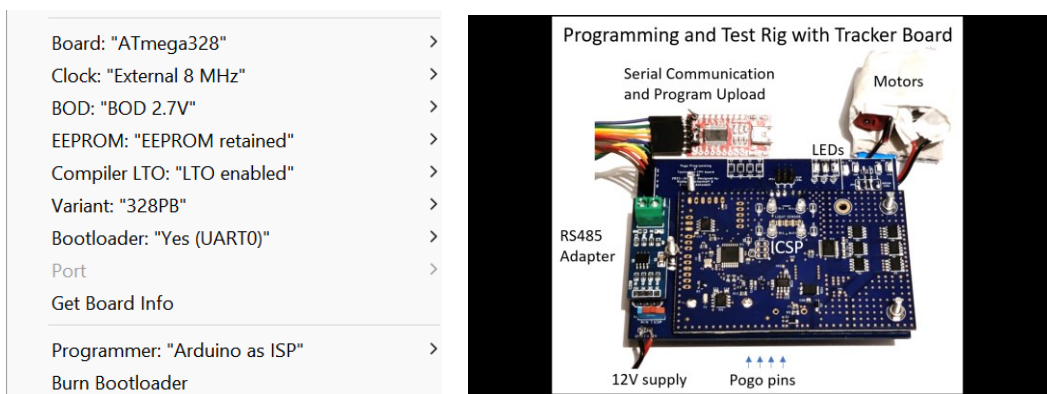


Fig. 7 correct settings for the ATMEGA328pb chip in Minicore and picture of a tracker controller board on the pogo board.

4.2 Uploading the code

You can either connect the FTDI adapter directly to the SERCOM port or connect it to the "FT232 USB Serial Adapter" port of the pogo board. Choose the correct com port. Use the same settings as in Fig. 7. Open the latest version of the tracker firmware. Upload the code

5 First Startup

Connect the FTDI adapter set to 3.3V to the SERCOM port of the tracker control board.

After Reset, you should see some printouts on the monitor, in particular "calibration step 1", which shows you, that the control has detected a first start and wants to calibrate. Ignore this for the time being and

- Type "~" to display parameters in EEPROM, which have been restored to factory pre-sets automatically during the first start.

```

COM3
presets restored
presets restored
calibration_step 1 780
set EEPROM values, see handbook, / to finish
calibration_step 2 983
!
*!0 ;111 <0 =111 >0 ?30 @3600 A111 B500 !
*!1 C8 D8 E8 F8 G8 H8 I10 J10 !
*!2 K10 L10 M10 N10 O1500 P111 Q111 R=100 !
*!3 S100 T0 U3400 V=1600 W0 X0 Y0 Z0 !
*!4 [900 \1 ]150 ^111 _111 `111 a111 b850 !
*!5 c900 d450 e900 f850 g50 h111 i400 j50 !
*!6 k50 l100 m100 n500 o500 p3 q3 r20 !
*!7 s20 t3 u16000 v300 w10 x10 y1000 z0 !
*!8 :21 :5 :31 :14 :5 :2 !
Autoscroll Show timestamp Newline 19200 baud Clear output

```

Fig. 8: Plot of factory presets

- If you enter “~” several times, you should see the real time clock in line *!8 counting up. You can set the correct time now, as explained in the section “time and date”.
- Type “%” to toggle the display of the magnetic readings and elevation. If you can see values similar to the ones below you know that all components work correctly.

```

COM6
C:\Users\ruedi\Documents\Arduino\221227_One_board_Tracker\02_setup.ino
1
-100.00
calibration_step 1 200
set EEPROM values, see handbook, / to finish
calibration_step 2 337
AzC=91.34 AzC2=90.50 ELC=1024.89 mx=-616.20 my=-80.90 mz=479.50 mxh=496.93 a_xyz=-23.00,-24.05,670.45
AzC=87.07 AzC2=90.16 ELC=1020.32 mx=-614.00 my=-77.70 mz=473.50 mxh=500.81 a_xyz=-18.80,-23.70,671.30
AzC=88.89 AzC2=90.03 ELC=1019.81 mx=-616.90 my=-79.90 mz=472.00 mxh=504.48 a_xyz=-18.30,-24.40,670.25
AzC=91.02 AzC2=90.13 ELC=1019.83 mx=-614.60 my=-81.30 mz=476.80 mxh=501.21 a_xyz=-18.30,-23.95,669.50
AzC=91.01 AzC2=90.22 ELC=1023.26 mx=-613.30 my=-80.40 mz=480.20 mxh=495.65 a_xyz=-21.65,-23.40,675.20
AzC=90.61 AzC2=90.26 ELC=1021.01 mx=-615.80 my=-80.90 mz=477.60 mxh=501.00 a_xyz=-19.50,-22.95,673.45
AzC=89.83 AzC2=90.21 ELC=1019.94 mx=-614.40 my=-80.20 mz=476.30 mxh=501.01 a_xyz=-18.50,-23.15,673.10
AzC=91.75 AzC2=90.37 ELC=1022.62 mx=-614.50 my=-81.40 mz=478.80 mxh=497.79 a_xyz=-21.10,-23.50,676.70
AzC=89.79 AzC2=90.31 ELC=1021.09 mx=-613.20 my=-79.70 mz=479.10 mxh=498.05 a_xyz=-19.60,-22.15,674.20
AzC=88.95 AzC2=90.17 ELC=1019.80 mx=-611.20 my=-78.80 mz=480.70 mxh=497.11 a_xyz=-18.20,-22.05,667.10
AzC=90.81 AzC2=90.24 ELC=1019.55 mx=-612.70 my=-80.90 mz=475.60 mxh=499.89 a_xyz=-18.20,-21.30,675.30
AzC=95.83 AzC2=90.80 ELC=1022.93 mx=-611.00 my=-84.60 mz=473.50 mxh=495.18 a_xyz=-21.30,-20.90,674.00
AzC=91.26 AzC2=90.84 ELC=1020.61 mx=-612.10 my=-81.00 mz=476.50 mxh=498.02 a_xyz=-19.20,-21.95,675.95

```

Fig. 9: Plot of magnetic readings

- AzC is the azimuth compass reading in 0.1° and will have an arbitrary value before calibration. AsC2 uses a floating average to reduce noise. ELC is the elevation measured with the tilt sensor. mx, my, mz are the magnetometer readings and mxh is the horizontal component of the magnetic field in x-direction.
- mx, my, mz are the magnetic components.
- a_xyz are the accelerometer readings. This particular MXC4005XC sensor has a large z-offset. The last value a_z should be 1000 mg rather than 660 mg when the board is lying flat. You can adjust this parameter by adding 340 mg typing 340: into the serial monitor. The serial communication is described below in more detail.

6 Encapsulation and light sensor

The control board is designed to be glued onto an aluminium plate with its front side with silicone. This will ensure a good thermal contact and a hermetic housing. The LEDs should stick up through holes in that plate to face the sun. Install a shading beam, such that each LED is covered halfway when the shading beam is adjusted to the sun. The 15mmx15mmx200mm shading beam should be mounted on pins to let dirt and rain pass underneath. It can be

made on a lath or it can be 3D-printed. In any case, it should be fastened with an M3 countersunk screw.

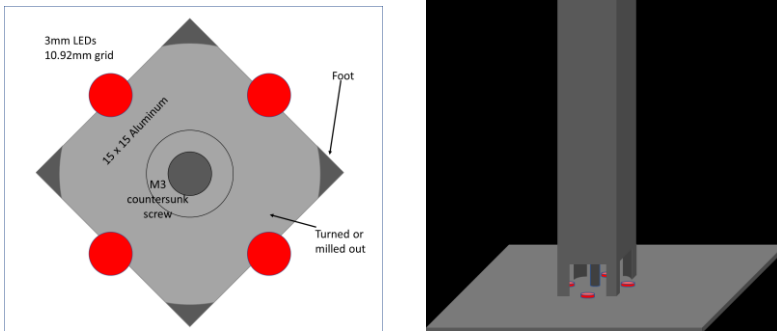


Fig. 10: Shading beam mounted over the LEDs

Mount the Plate as shown in Fig. 11. The tilt of the board can be adjusted with the nuts on the three bolts.

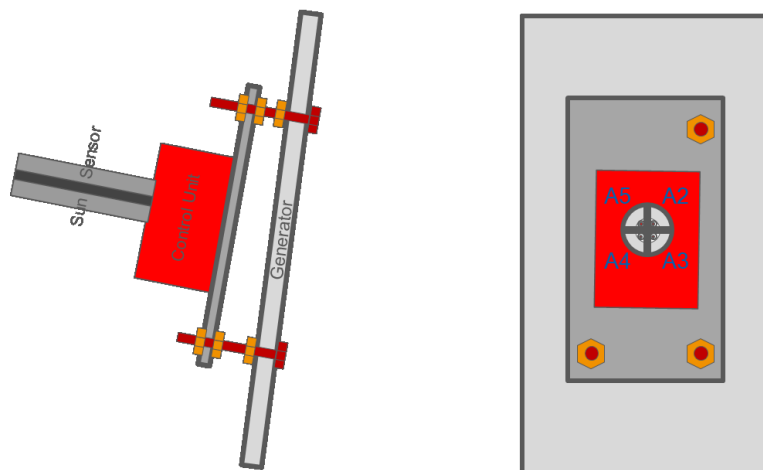


Fig. 11: To be revised. Old Fig. for the control box with breakout boards. It will work just the same way for the integrated board.

This section will require a revision with photos. I didn't have time to finish that step yet.

7 Calibration

For the calibration the control unit should be mounted on a functional tracker which can freely rotate by more than 360° in azimuth.

I strongly recommend that you should first test your control unit on a small and safe benchtop tracker before taking it to a full-size tracker outside. This way you can also get yourself accustomed to all functions. Full size solar trackers can be quite dangerous. They can self-destruct when they move beyond limits, parts may detach and fall down. AC cables may get damaged and give you an electric shock. Most importantly, these trackers are usually strong enough to crush your hand or head. It doesn't necessarily help, that they are so slow. You may get yourself in a situation, where you cannot pull your hand or head out or stop the

tracker and it will just keep on going and going and going. Beware, I refuse all responsibility as stated in the MIT license below. Always work in company when you are dealing with heavy machinery.



Make sure that the cables can follow the tracker movement. To be on the safe side, they should allow the tracker to rotate several times before they rip.

Establish a serial connection.

For the outdoor testing of a big tracker it is best to have the USB-Serial converter close to the control unit. I have been using a 20m active USB cable to bridge the distance to a comfortable spot from where I could control the tracker.

As explained above, the control will automatically go into calibration mode at the first startup. In particular it will prompt you:

“Set EEPROM values, see handbook, type / to finish.”

1.4 Calibration_step 1: Set your parameters

- Reset by opening the serial interface.
- The Arduino serial monitor is prompting:
calibration_step 1
set EEPROM values, see handbook, / to finish
calibration_step 2 406
- You can reset the tracker to factory defaults by entering “#” and confirming with “y”.
- You can display a list of the user defined parameters in EEPROM by sending “~” over the serial interface (see Fig. 8)
- You can copy and save this list in a text file.
- You can restore a full line by copying it from the text file to the entry line of the Arduino serial monitor and pressing send or the return key.
- You can edit a line before sending it.
- Line *!8 shows the present date and UTC-time in the format yy mm dd hh mm ss. Copy this line to the serial monitor, correct it as necessary and confirm with send or return.
In particular you need to make the following entries now:
\\ : Number of double steps in elevation. E.g. 1 will yield 2 steps between three positions.
[: Upper limit of the elevation range. This needs to be lower than the mechanical limit.

J : lower limit of the elevation range. This needs to be higher than the mechanical limit.

C, D, F, G: duty cycles for the elevation drive. Reduce them to a max. of 8, if the elevation motor.

E, H : Same for azimuth

- For a complete list with explanations refer to the Appendix.
- When you are done, enter “/” to go to the next step.

1.5 Calibration_step 2: Find sun

The serial monitor will prompt:

```
Find sun: U D L R c_ontinue s_top f_inish  
calibration_step 3
```

I assume that you run the control on top of a full size CPV tracker with CPV-modules installed. The control unit is sitting on the solar generator between the CPV-modules. In the precise sun sensor mode, the control unit aligns itself to the sun. So you first need to make sure that the CPV modules have the correct orientation to the control unit with its sun sensor.

Adjusting all CPV modules to the sensor would be very tedious. It is much better, to adjust the control unit on the generator. Since the compass sits inside this very same control box, we first need to calibrate the sun-sensor-based tracking. That means, that you need direct sunlight for this step!

Your job is to orient the tracker orientation close enough to the sun to let the sun sensor tracking take over. “U” will move the tracker up by one step, “D” will move down, “L” will move left and “R” will move right (clockwise). “c” → “R” will continue in the clockwise direction indefinitely, until you type “s” for stop.

At some point, the sun sensor should take over and the tracker will orient itself to the sun. Have a look inside the CPV modules (use dark glasses) to check the alignment. Now adjust the alignment of the control unit to the generator by moving the nuts on the rods up or down. The generator will follow in the opposite direction. You can stop, when the generator is reasonably well aligned. Fine adjustments can be done later. Just be aware, that these later adjustments will also affect the astronomical, compass based tracking.

If the control unit doesn't switch to sun sensor mode, you may need to adjust the parameters “i” to “o”, in particular the total sensor signal for direct sun “i”.

If you are just testing your unit on a benchtop tracker, you don't need to adjust the mechanics but you may want to use this chance to test the tracker movements and decide whether every motor is running in the right direction at the right speed.

When you are done, enter “f” to finish.

1.6 Calibration_step 3: Correct elevation up limit

Now the control will prompt:

```
corr 70° u U d D, f finish  
calibration_step 4
```

where 70° is replaced by the elevation up limit that you have entered as parameter “**U**”. Use **U** or **D** for big up or down steps and **u** or **d** for small steps. These commands change the “elevation up correction” parameter “**R**” for astronomical tracking in the EEPROM and the control unit will react accordingly. If you are not happy with the precision or if the elevation steps up and down without ever coming to a rest, you may need to reset the tracker and try other values for “calibration elevation error” “**t**” in step 1. You can skip step 2 by typing “**f**” to return to step 3.

Use a water level and maybe a wedge (e.g. 20° in the example) to measure the elevation angle.

1.7 Calibration_step 4: Correct west

The control unit prompts:

```
corr West r R | L, c ontinue, s top, f  
calibration_step 6
```

Move the tracker to west by typing “**c**” “**L**” for a continuous counter clockwise movement. Stop with “**s**”, do stepwise corrections with “**R**” and “**L**” for clockwise or counterclockwise moves. Other serial inputs may cause hang-ups → reset. “~” is allowed. A rough orientation to west by a few degrees will be enough. In this step you are just defining the turnaround point for the automatic calibration. The calibration precision is not affected. Finish with “**f**”.

1.8 Calibration_step 6: Correct elevation down limit

You will see a prompt where 30° is replaced by your lower elevation limit:

```
correct 30° u U d D f  
calibration_step
```

30° is replaced by the elevation down limit “**J**” that you have defined in step1. Proceed like in step 3.

1.9 Calibration_step 8: Start automatic calibration

Now you are done and the automatic calibration starts. The output of the serial interface allows you to analyze the calibration especially in case it doesn’t work as expected. It starts with:

```
calibration_step 8 56367  
starting automatic calibration  
calibration_step 9 71320  
29  
28...
```


The mz neutral correction temp_mz is measured and it is written to EEPROM at the end of the calibration routine. Now, the tracker moves up in steps to measure the magnetic readings that define the turnaround points during the 400° azimuth swings.

When the tracker has reached the uppermost elevation position it will start with a short counterclockwise move and then performs a 400° clockwise swing followed by a move down, a 400° counterclockwise swing and so on until it reaches the lowermost position. During this procedure it shows the compass readings for analyses. The last prompt is something like:

```
mxh<469.16
my<1103.22
mxh>-219.61
my>424.91
mxh_n=124
my_n=764
mxh_r=344
my_r=339
ms/deg=93
complete --> restart
```

The first 4 lines show the maximum and minimum values of the horizontal x and y magnetic readings. These values are converted to mxh_n and myh_n, which are the neutral values and the ranges mxh_r and myh_range.

The tracker uses a floating average to improve the precision of the azimuth measurement, even when it is moving. For this correction we need the speed of the azimuth movement given in milliseconds per degree. In this case, a 360° turn will take 37.8 seconds. This is the pretty fast tabletop tracker that I have shown you in the “mechanics” section.

1.10 Calibration explained

I want you to understand what happens during the automatic calibration. Below you can see a graph of the magnetic y “my” reading and the mx_horizontal reading, which is calculated as the horizontal component of the mx-mz vector. I have set the number of elevation steps to its minimal value, which is “\=1”. Since “\” is the number of double steps, we have two steps in elevation or 3 positions. The maximum elevation “[“ is set to $700 \cdot 0.1^\circ = 70^\circ$ and the minimum elevation is set to $300 \cdot 0.1^\circ = 30^\circ$.

The graph Fig. 12 shows the magnetic readings for the >360° azimuth swings at different elevations in different colors.

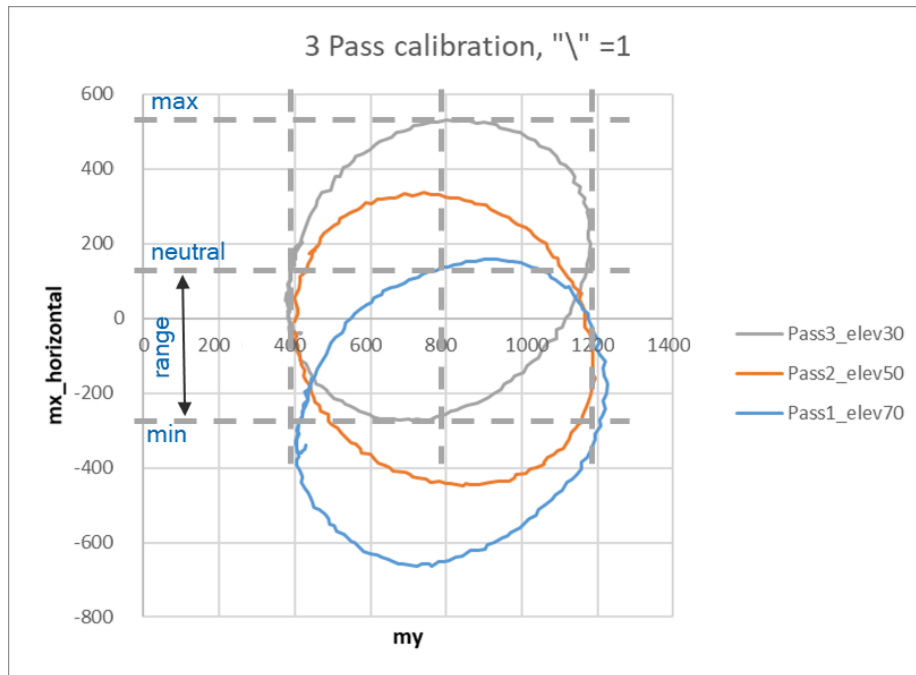


Fig. 12: Magnetic readings during a 3-pass calibration

Of course, the Arduino Uno doesn't have enough memory to record such graphs. But it can record the maximum and minimum magnetic readings for my and mx_horizontal. I have shown this with dashed lines. The neutral position is the average of max and min and the range is half of the difference. During normal operation, we are subtracting the average and we are scaling by $1/\text{range}$. This approximates the readings of my and mx_horizontal to a circle of radius 1 around the origin for each of the discrete elevations, such that we can calculate the azimuth. For elevations in between, I introduced a linear interpolation of the neutral positions and the range.

You will notice, that the neutral my position is pretty far off zero, while mx_horizontal is approximately zero. This is because mz has already been set to zero in the 30° west position in step 6 and mx in the 70° west position in step 8 of the calibration. Big offsets of mx and mz would make the azimuth calculation very sensitive to elevation so I needed to eliminate them before the azimuth calibration runs.

These readings were taken with my benchtop tracker. You can see that the passes constitute ovals rather than perfect circles. I guess this is because of steel parts in my lunch table which create an inhomogeneous magnetic field. For a big installation you won't have that, as long as you don't decide to put the tracker on steel rails as a foundation. Yet, it is a good idea to choose a place for the control box which is not too close to the motors and which has a symmetrical arrangement of steel parts around it. During my tests on an 11 m^2 steel tracker I have put it centered in the upper half of the generator as in Fig. 13.

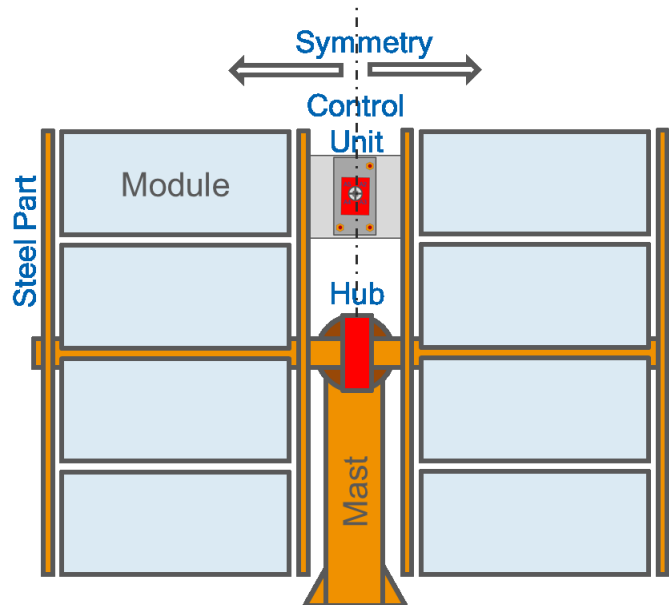


Fig. 13: Symmetric mounting position on a steel tracker far away from the motors

8 Operation

There are only a few commands to operate the tracker. You can find them in the appendix. Most manipulations are rather done by changes of the user defined EEPROM parameters. Probably you will first want to test the astronomical tracking. So first make sure that the time is right. Enter “~” and check line *!8 of the serial output against UTC (not local time). Also set or check longitude “W” and latitude “X”. Meanwhile the tracker will already move to the calculated azimuth position. Cover the sensor to make sure that it is not going to sensor mode. While the sun is shining you will see whether the direction is realistic. If not, you can play with parameter “T” to introduce a correction of the compass direction.

Now you can move to different positions by changing the cleaning position “c”, “d” and going to this position with “\$1”. You can move up and down for one azimuth position. An error of a few degrees is normal. The tracker may readjust slightly as it is moving up or down.

If you are not happy with the result, you may attempt to recalibrate with more positions. Usually that should not be necessary since the tracking sensor covers about +/- 20°. The astronomical tracking just needs to be precise enough to keep the tracker within this error margin.

It's fun to type “1:<” to get into turbo mode and see how the tracker would move during a day, but much faster. You will have to set the clock again afterwards.

If you are happy with all that, remove the cover from the sensor. It should find the direct sun. You may adjust the parameters “i”..”o” for the sensor sensitivity and “F”..”N” for the motor step parameters. High sensitivity at high hysteresis together with small steps at low duty cycles is preferred for precise and efficient tracking.

If the tracker is running in the wrong direction in sensor mode, you may have made a mistake in the cabling or you may have rotated the sensor by accident. This can either be corrected in the code or better with a soldering iron.

Monitor your tracker for a couple of days to see whether the astronomical errors are building up. Based on this experience, you can decide about the number of days “>” that may pass before the automatic recalibration.

9 More to be done

1. Extensive tests on sun. Several months.
2. Establish an RS485 protocol for the communication of the control board with the field control using the onboard RS485 communication chip.
3. Replace external wind sensor by internal detection of wind-oscillations.
4. All in one circuit board.

Not all applications will require all of the features above. Tracker fields will receive the go to stow signal through the RS485 bus, thus individual wind sensing is not necessary. We have 20% of 32 kB left. Plenty of room.

10 Appendix

1.11 EEPROM Parameter List and Commands List

This paragraph explains all user definable parameters and how to enter them in the serial monitor of the Arduino IDE. All parameters are signed integers. Degrees are expressed in 0.1° steps. E.g. an elevation of 900 means that the tracker is facing the Zenith at 90°. All parameters have limits and the software will automatically correct entries which are out of limit to the closest allowed value. There are also presets which can be restored with “#” → “y”.

I recommend printing out the table in Fig. 14 below as a separate page, preferably on cardboard and maybe laminated. Computer screens are not always easy to read when you are working outside. The color code defines blocks with similar parameters, e.g. parameters that affect the motor movements in light red.

There are three ways to enter parameters into the EEPROM.

- 1) You can either type “~” which will display the whole parameter block. Copy a complete line to the serial input line and change the parameters as required. Hit return. This will change 8 parameters at the same time. However, there are predefined limits. If your new value is out of limits, the control will automatically save the closest allowed value (lower or upper limit).
- 2) You can type in a relative change of a single parameter. E.g. “10T” will increase the azimuth correction T by 10 which is 1°.
- 3) Or you can enter 15° as an absolute value by typing “150:T” where “.” marks an absolute value. The reversed order “T150” will not result in any change because it is a relative change of 0 and “150” is discarded. Entering “T”, with or without a change, will first display the start position of the byte in the EEPROM (50) and the new value of T (e.g. 15). “350:T” will display a value of 300, since 300 is the hard coded upper limit of T. Below I will explain the parameters in detail.

Fig. 14: EEPROM parameters and commands list (below)

EEPROM Parameters List

*!0	; -32k..32k 1234 after calibration	< 0..1 Turbo Mode	=111	> 0..1000 Days since calibration	? 0..1000 Days for calibration	@ 0..32767 Count for RTC correct	A111	B 200..20k Brake cycles	!
*!1	C 1..16 Up duty cycle	D 1..16 Down duty cycle	E 1..16 Left/right duty cycle	F 1..16 step up duty cycle	G 1..16 Step down duty cycle	H 1..16 Left/right duty cycle	I 1..255 Ramp cycles up	J 1..255 Ramp cycles dwn	!
*!2	K 1..255 Ramp cycl. left/right	L 1..5k Step up plateau	M 1..5k Step down plateau	N 1..5k Left/righ plateau	O 1k..5k Cont. move plateau	P111	Q111	R -100..100 Elevation up correct.	!
*!3	S -100,,100 Elevation down corr.	T -300..300 Azimuth correction	U 3k..3.4k Azim. right limit	V -1.6..1.2k Azim left limit	W 0..3599 longitude	X -800..800 latitude	Y -32k..32k mx neutral position	Z -32k..32k My neutral position	!
*!4	[700..900 Elevation up limit	\ 1..12 Elev double steps] 0..300 Elevation down limit	^ Reserved for ID	board only _ -1k..1k acceler. z-offset	`111	a111	b 0..900 Elevation Stow	!
*!5	c-900..2.7k Azimuth clean	d 0..900 Elevation clean	e-900..2.7k Azimuth Home	f 0..900 Elevation Home	g-50..200 Elev.trigger sleep	h 111	i 10..20000 sensor direct sun	j 10..500 sensor step azimuth	!
*!6	k 10..500 sensor step elevation	l 10..1000 sensor run azimuth	m 10..1000 sensor run elevation	n 0..800 sensor hysteresis	o 0..800 sens direct. Hysteresis	p 1..200 astro error step azim.	q 1..200 astro error step elev.	r 1..200 astro error run azim.	!
*!7	s 1..200 astro error run elev	t 1..20 calibration elev. err.	u 0..32767 Elev. over- current	v-1..1023 windspeed stop	w1..100 wind trig- ger time	x1..100 wind re- turn time	y azi rotation speed	z azim left of east	!
*!8	:21 year	:5 month	:1 day	:17 hour	:45 minute	:33 second	!		

Commands List

#	Restore factory defaults for all EEPROM parameters
%	Show compass readings in the serial monitor
\$0	return to normal tracking (careful , this also works from calibration routines and may have undesired results)
\$1	Disables sensor tracking and sets cleaning position "c" "d" as target. The tracker will move to the cleaning position. You can also use this for manual moves. Type \$1 again, for changes of "c" and "d" to take effect.
\$8	Initiate a recalibration
\$9	Initiate a new calibration
\$	This will jump into calibration step 3, where you can run the tracker manually. This comes handy in some cases but it is also dangerous because all limits are disabled. Return with \$0. Be in company if you should attempt this.

10.1.1 Time and Date *!8

Typing "~" into the Serial Monitor will display e.g.:

```
*!8 :20 :7 :2 :5 :39 :59 !
```

as the last line. This is 5:39:59 am on July 2nd 2020.

Modify this line to the present UTC time and date and enter it into the Serial Monitor to set the new time and date. You can find the UTC world time on the internet. Your local time will yield wrong results (if you are not living in Greenwich). You can also choose other times and longitudes/latitudes to see how the tracker will behave.

10.1.2 Marker for primary calibration ;

This value is used to detect a missing primary calibration and it will change to 1234 after calibration. You cannot change this value manually. After the primary calibration, the upper range of the EEPROM will be used for a safety copy, even in the event of a power failure during recalibration.

10.1.3 Days since calibration > 0..1000

This value will keep increasing by 1 at UTC midnight. When it reaches the value “?” the tracker will recalibrate at night without the need for user inputs. Usually, you don't need to change this parameter, but you can.

10.1.4 Days for calibration ? 1..1000

After how many days should the tracker do an automatic recalibration? You may start with a rather high value and monitor how the astronomical tracking slowly drifts away. When you believe that it is becoming unacceptable, set the present value of “>” as the new value for “?” and the tracker will recalibrate after sunset.

10.1.5 Counter overflow value for RTC time correction @ 0..32767 seconds

We want to have the tracker running indefinitely without the need for user interference and the RTC would slowly drift away. Therefore, we detect situations, where the astronomical tracking is ahead of the sun sensor-based tracking and reduce the time by 2s, when an overflow occurs. The overflow value is given in seconds on sun. This also works the other way around. If the astronomical tracking is falling behind, 2s are added at an overflow in negative direction. The default value of 3600 seconds means, that a correction occurs when the astronomical tracking has been in front or behind for one hour.

10.1.6 Turbo Mode < 0..1

You can activate the turbo mode to speed up the progression of the time by a factor of about 50. Thus, you can watch how the tracker will move during one day.

(Internally, each time the seconds in the RTC show 00, they will be set to 59 again. Thus, each minute lasts only a little more than a second.)

10.1.7 Number of break Cycles B 200..20000

How long should the motor be in the short circuit break condition before it is ready for another movement? In order to get the number of brake cycles, multiply this time by 4000, where 4000 Hz is the Arduino UNO PWM time @ 16 MHz. 8 MHz Arduinos will only have 2000 Hz PWM. Up to 20000 corresponding to 5s are possible.

Type e.g. 500:@ into the Serial Monitor.

10.1.8 Plateau Duty Cycles C, D, E: 1..16

The motor control is using duty cycles from 0 (no operation) to 16 (full speed). If you are running 12V motors on 24V, keep the duty cycle below 13. Also check with the supplier if the motor will tolerate 50% duty cycle at 24V and 4000 Hz.

The parameters

C: up

D: down

E: left/right

specify the plateau duty cycles under full speed operation. The duty cycles C (up) and D (down) can be set separately in case the elevation axis is under load during operation.

10.1.9 Step duty cycles F, G, H: 1..16

The motors can run in small steps for tracking. These steps have separate plateau duty cycles that should be as low as possible to reduce the wear of the motors.

F: step up

G: step down

H: left / right

10.1.10 Ramp repeat I, J, K: 1..255

The PWM is ramped from 0 to plateau PWM duty cycle (max. 16). The parameters I, J, K define how often a certain PWM duty cycle should be repeated. The maximum value is 255 corresponding to 64 ms on each step. Consequently, a full ramp from 0 to 16 takes one second.

I: up

J: down

K: left/right

10.1.11 Plateau repeat step L, M, N: 1..5000

Each step has a plateau time given by the number of PWM cycles. At 4 kHz a maximum plateau time of 5000 corresponding to up to 1250 ms is possible.

L: up

M: down

N: left right

10.1.12 Full speed plateau time O: 1..5000

Each motor movement is a step – even continuous movements. The only difference is, that during continuous movements the plateau counter is reset to 0 in every pass of the main loop. Consequently, the movement continues.

O needs to be chosen such that the plateau time is longer than the time required to pass through the main loop. 1400 (350 ms) has proven to be a good value. If you should experience interruptions of the motor movement, you can set it a little higher.

10.1.13 Elevation up and down correction R, S, -100..100

These values are used during the primary calibration to correct for errors of the tilt measurement. They are locked during normal operation

10.1.14 Azimuth correction T: -200..200

Use this to correct the astronomical azimuth tracking by up to +/- 20°.

10.1.15 Azimuth limits U 3000..3400, V -1600..1200

You may choose the limits for the azimuth. The recommended settings of -1600..3400 azimuth range will mean +/- 250° rotation from the East position. East is chosen to be the neutral position because the sun always comes up in the east and may move both ways round depending on where you are.

10.1.16 Longitude and Latitude W -1800..1800, X -800..800

Look up the coordinates for your site on the web. E.g. 51° 31' N , 0° 7' W

Now you need to calculate the Longitude and Latitude in 0.1° steps:

Longitude = $10 * (0 + 7/60) = 1$ → type 1:W in Serial Monitor

Latitude = $10 * (51 + 31/60) = 515$ → type 515:X in Serial Monitor

10.1.17 Neutral positions for magnetic x and z: Y, Z -10000 10000

The tracker uses the horizontal components of the magnetic field to calculate the azimuth. An automatic calibration routine is used to find the neutral position of the magnetic y reading my. For the other direction we need to determine the horizontal x component mx_horizontal from mx and mz. The automatic calibration routine also measures the neutral position of mx_horizontal but high errors of mx and mz may result in a higher sensitivity of the compass reading to the elevation position. Therefore, the neutral position of mx and mz is estimated during the calibration routine and stored to the positions Y and Z in increments of 10. Y and Z are not user definable.

10.1.18 Elevation up and down limits [800..900,] 0..300

Not all trackers can perform movements in the full elevation range of 0 to 90° ($0..900 * 0.1^\circ$). Consequently, you can limit the movement range. These values can only be changed at the beginning of a new calibration. They will override each target position which is out of range. Sensor tracking is disabled, when the astronomical position is out of limits.

[upper limit,
] lower limit of the elevation range

10.1.19 Half number of elevation steps \, 1..12

The backlash “\” lets you configure the number of elevation steps at the beginning of a new calibration. It's defined as the half number of steps to make sure that only even numbers of steps and odd numbers of elevation positions are possible. Keep this number as low as possible. Many calibration positions will mean that your azimuth drive is wearing down in each recalibration. In many cases a value of 1, corresponding to three elevation positions (uppermost, lowermost and in between) will be enough to achieve a sufficiently precise astronomical azimuth tracking. Example: To set 4 elevation steps and 5 elevation positions type

5:\

10.1.20 Reserved: Tracker ID ^

RS485 field communication requires a unique tracker ID. It should be in one block with the other parameters that can only be changed during the primary calibration.

10.1.21 Elevation target position for stow b 0.900

When the wind gets too strong, the tracker should lay itself flat at a close to Zenith position to avoid high wind loads. The wind may come from any direction, so an azimuth stow position

makes no sense and running in azimuth during storm may damage the motor. You can decide whether the tracker should be completely flat during stow or whether it should be slightly tilted to let rain run off. The stow position is given as a relative value compared to the target position. E.g. Elevation up limit is 900 so type.

-50:b

to set the elevation stow position to 850 = 85°

10.1.22 Azimuth position for cleaning c -900..2700

You can get into cleaning mode by typing \$1 from normal tracking and the tracker will go into a predefined position in a full 360° range from -900..2700.

10.1.23 Elevation position for cleaning d 0..900

E.g.

0:d

10.1.24 Azimuth home position e -900..2700

The tracker goes to a defined home position at night. Type the following, to set the target position to 90° azimuth:

900:e

10.1.25 Elevation home position f 0..900

Elevation position during the night. Slightly below 900 (90°) is recommended to let rain run off.

850:f

10.1.26 Elevation position to trigger sleep g -50..200

Night is defined by a calculated elevation position which is below g. The higher you set g, the earlier night will start and the later it will end. It is a good idea to set this value slightly below the elevation down limit such that the tracker will wake up a little early and have time to move to the sun. This position is defined in 0.1° steps. Values lower than 0 are possible since this is only a value used in the calculations. E.g. type

100:g to set the wake up position to 10° elevation

10.1.27 Sensor level to detect direct sun i 10..20000

When the sensor is receiving a certain level of light, the tracker is switching into sensor-based sun tracking. The higher the value, the later the tracker will go to sensor-based tracking. I allowed for a max. value of 20k such that you can set the tracker to never enter sun sensor mode.

10.1.28 Sensor error for azimuth step j 10..500 and elevation step k 10..500

The software calculates the relative difference of the upper and lower sensor elements. When the value is beyond a certain limit, a tracking step is initiated. The lower the value j, the earlier the step will start.

10.1.29 Sensor run error for azimuth l 10..1000 and elevation step m 10..1000

Make sure that these values are higher than the step errors.

10.1.30 Sensor tracking hysteresis n 0..800%

We want to prevent that the tracker is moving from astronomical tracking to sensor tracking and back all the time. This is why we enter a hysteresis. The sun sensor reading is increased by n percent when the tracker already is in sun sensor mode.

10.1.31 Sensor tracking direction hysteresis o 0..2000%

The same principle applies to direction reversals in tracking step mode. A high value makes steps in the same direction easier while there is no effect of this value on direction reversals.

10.1.32 Compass errors for azimuth p and elevation q steps 1..200

A value of 100 means that a tracking step will be initiated when the compass measures an azimuth (elevation) deviation from the calculated target of more than 10°. You may consider setting these values higher than the values for full speed movements. In this case, there won't be any tracking steps in compass based tracking which may reduce the wear of the motors.

10.1.33 Compass errors for azimuth r and elevation s corrections at full speed 1..200

Deviations in degree that will trigger full speed corrections.

10.1.34 Compass errors for elevation steps during calibration t 1..200

It is important that the elevation is set correctly during calibration. This is why we are setting a lower trigger level for the elevation elevation steps during calibration. A high value will compromise the magnetic compass precision, a low value may mean, that the elevation keeps clicking forever in the calibration routine and the routine will not continue.

10.1.35 Elevation overcurrent u 0..32767 mA

This defines the current in mA at which the tracker will come to a complete halt and display "overcurrent". Even so the VNH5019 only allows 16A continuous, I set the upper limit to the highest signed integer since the currents may fluctuate. The default is 16000. Reduce it as required.

10.1.36 Stow windspeed in clicks per 10s v -1..1024

The present version of the software assumes a sensor with a reed relay which will close at certain angles while the wheel is rotating. Thus it is generating "clicks". The more clicks there are in 10s, the faster the wind.

-1:v will always enable storm (for testing)

1024:v will disable storm

300:v preset for 20 meters per second and N25FR sensor*/

10.1.37 Wind trigger time in seconds w 1..1000

A wind gust above the stow wind speed will be detected but it only triggers "go to stow" if the windspeed remains above the limit for longer than the wind trigger time w which may be set in the range of 1 to 1000 seconds.

10.1.38 Wind return time in seconds x 1..1000

In the same way, the tracker only returns from stow if the wind speed is below the limit for longer than the wind return time x.

10.1.39 Milliseconds per degree azimuth y

Actually, this value is not user defined, but you can see it in the parameter block for reference. Magnetic readings are so noisy that they need to be averaged, even when the tracker is moving. To do that properly, we need the rotation speed.

10.1.40 Azimuth left of east z

Again, this is not a user defined parameter. The control sets it automatically when it is moving over the east position to remember whether it has been right or left of east. A hysteresis suppresses rapid changes, that might wear down the EEPROM.

1.12 MIT License

Copyright 2021 Ruediger F. Loeckenhoff (alias SolHunter)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Comment: This whole document is to be regarded as part of the documentation files and consequently permission is also granted to all information which is not protected by patent and the exclusion of any kind of liability according to the license above also refers to the recommendations for possible arrangements of mechanical and electronic parts.<

1.13 Previous Work and Acknowledgements

The basic concept of compass tracking has been presented at the CPV-17 conference on behalf of my employer AZUR SPACE Solar Power GmbH [0]. Yet, I am publishing the present work on a strictly private basis. All necessary rights for this open source publication have been transferred to me. AZUR SPACE doesn't hold any rights that might imply any form of liability.

I greatly thank my employer for his support. May this work facilitate the success of CPV.

1.14 References

0. R. Loeckenhoff, CPV-17, Freiburg, "A minimalistic, universal and highly precise sun sensor and compass based CPV-tracker control".
1. R. Loeckenhoff, "Sonnenstandssensor", DE102018001181B3
2. R. Loeckenhoff, "Sonnennachführungsvorrichtung", DE102019004468A1

3. Saheli Ray, "Calculation of Sun Position and Tracking the Path of Sun for a Particular Geographical Location" in International Journal of Emerging Technology and Advanced Engineering, Vol. 2, Issue 9, Sept 2012.
4. Muhammad E. H. Chowdhury, Amith Khandakar, Belayat Hossain, Rayaan Abouhasera, "A Low-Cost Closed-Loop Solar Tracking System Based on the Sun Position Algorithm", Journal of Sensors, vol. 2019, Article ID 3681031, 2019; <https://doi.org/10.1155/2019/3681031>
5. www.arduino.cc
6. "Pololu Dual VNH5019 Motor Driver Shield for Arduino (ash02a)", <https://www.pololu.com/product/2502>.
7. maxim integrated, "DS3231 Extremely Accurate I2C-Integrated RTC/TCXO/Crystal", <https://datasheets.maximintegrated.com/en/ds/DS3231.pdf>, 2015.
8. invensense, "MPU-9250 Product Specification Revision 1.1"

1.15 Revision Index

01.06.2021	R. Loeckenhoff	Initial open source publication
06.06.2021	R. Loeckenhoff	Detailed instructions for housing and one correction in the wiring (Wind sensor on D3)
13.06.21	R. Loeckenhoff	Improvements to sensor tracking with floating average and direction hysteresis o extended to 2000%.
22.08.2022	R.Loeckenhoff	On Track on average feature.
31.12.2022	R. Loeckenhoff	Revised for integrated tracker board.