Elwood Downey, WBØOEW

8274 N Sunset Ranch Loop, Tucson, AZ, 85743-8201: **ecdowney@ClearSkyInstitute.com**

Autonomous Satellite Tracker

This two-axis tracker needs no prior alignment with respect to the Earth's surface, and tracks satellites with high accuracy entirely by reference to built-in spatial sensors.

This paper describes a completely autonomous Earth satellite tracking mount. In conjunction with a 9-DOF (degrees of freedom) sensor attached to the antenna boom and a GPS receiver, the 2-axis gimbal will track any Earth satellite within 2 degrees in real time without any orientation setup calibration of any kind. The Tracker system includes a built-in web server and Wi-Fi access point, which allows all monitoring, command and TLE upload from any web browser including smart phone. All components are off-the-shelf so no custom electronics, machining or other skills are required except that needed to attach an antenna boom to a flat plate. All electronics can be powered from a single dc supply from 7 to 18 V such as a LiPo battery pack or solar charging system. As of the time of writing, total cost of the electronics and gimbal is approximately $350, not including antenna.

Introduction

Observers have been tracking Earth satellites with gimbal mounts since the beginning of the space age.[1] One of the challenges has always been to align these mounts so the theoretical calculations of satellite azimuth and elevation could be transformed to the mount coordinate system. After reading about the availability of low cost MEMS devices that directly measure spatial orientation, I wanted to build a mount that avoided the tedious calibration step by measuring directly the pointing direction of the payload.[2]

Going one step further, I also wanted to eliminate the need to have any prior knowledge of — or make assumptions about — the gimbal geometry, axis orthogonality, motor assignment and axis rotation angles. Doing so would simplify the mechanical requirements of the gimbal and wiring, and allow the use of simple off-the-shelf hobby servo motors and robot hardware for use with light weight antennas.

The final goal was the ability to control and monitor the entire system from my smart phone without needing to first install an app. The most flexible way to accomplish this is by providing a web server, so I needed enough memory in the controller to accomplish this.

Achieving the Goals

The first goal is achieved by measuring the spatial orientation of the antenna directly using a combination of 3D magnetic, accelerometer and gyroscope sensors. When packaged together, these devices are referred to as having 9 Degrees-of-Freedom, or simply 9-DOF sensors. The magnetic sensor provides the direction of the local magnetic field, including tilt. This is combined with knowledge of the local vertical gravity vector from the accelerometer to produce local elevation and azimuth with respect to magnetic north. To get a bearing from true north, the latitude and longitude from the GPS is combined with the *World Magnetic Model* to compute the local magnetic declination correction factor.[3] Information from the gyroscope provides additional stability and repeatability information. The final correction is to apply a simple model for atmospheric refraction to elevation based on nominal assumptions for air temperature and pressure. Although these too could be measured quite easily, the maximum effect at the horizon is about one-half degree, which I decided was not worth refining further. Taken together, these measurements provide an absolute measure of antenna direction in the local horizon coordinate system, which is exactly what is produced by the orbit propagator.

Now that we have the measured antenna direction and a computed predicted direction from the propagator in the same coordinate system, the second goal is to drive the gimbal motors in such as way as to reduce any difference between the two. Normally this is done in closed-form by using a transformation matrix determined ahead of time that relates the gimbal axis coordinates to the local horizon coordinates. In order to eliminate the need for determining this matrix, my second goal is achieved by moving the motors by a small amount and just measuring whether the error increases or decreases. This is known as a gradient descent search.[4]

My first attempt at an error metric was to use the great circle distance between the measured and computed positions. However, this leads to a condition known as gimbal lock if the gimbal ends up pointing near the zenith, either intentionally because the satellite pass was high or zenith was reached unintentionally during the search procedure.[5] This is avoided if the errors in azimuth and elevation are measured separately.

The final tracking algorithm can be summarized as follows:

Step 1 – choose one axis motor at random

Step 2 – measure error in azimuth and elevation separately

Step 3 – move the current motor a small amount and stop

[1]Notes appear on page 00

Figure 1 — Tracker gimbal is attached to a tripod and supports an Elk 2m/70cm LPDA antenna.

Figure 2 — Inside view of the Tracker electronics box.

Step 4 – measure the two errors again
Step 5 – if either error increased, reverse the last move and start using the other motor
Step 6 – go to Step 3, repeating forever.

Note that this does not require any knowledge of the gimbal orientation or even what motor operates what axis. The effect is the antenna will make a few small random moves to get started, then one motor will march along steadily until it causes one or the other error measures to increase. Then the other motor will do the same and the process repeats until the antenna is pointing at the satellite. This process repeats forever. So, as the satellite moves, the errors creep up and the algorithm keeps working to reduce them. The smoothness of the motion depends on the time between moves and the angle commanded for each move. These are not critical during a large slew but some care is needed in order to maintain smooth tracking performance. A rigorous approach is not required. It is easy to set reasonable values using trial and error. The algorithm could be made more efficient by introducing control-loop equations for proportional gain, so large errors are reduced more quickly, and integral gain to maintain closer tracking tolerances, but in practice these refinements are not really necessary.

The Web Server

The web server turned out to be straight forward. I already know Javascript, HTML and the HTTP headers that are used between browser and server so I wrote my own server state machine from scratch on top of the basic Arduino Ethernet library. The main page is sent, in effect, as the default *index.html* for the server URL address. All state variables are updated and reported using a consistent NAME=VALUE syntax, where the NAME usually matches the HTML name of the corresponding DOM display element. Setting a new value is performed with a POST command and retrieving values is done by asking for *getvalues.html*. An XMLHttpRequest polls for values to keep the web page updated. More details about using the web interface are provided later.

Implementation Decisions

Figure 1 shows the Tracker gimbal attached to a tripod and supporting an Elk 2 m/70 cm LPDA antenna. Figure 2 shows the inside view of the Tracker electronics box. Figure 3 is a block diagram showing how each electronic subsystem interconnects. Table 1 shows the major bill of materials.

Next, I elaborate the role of each component and share my experiences that

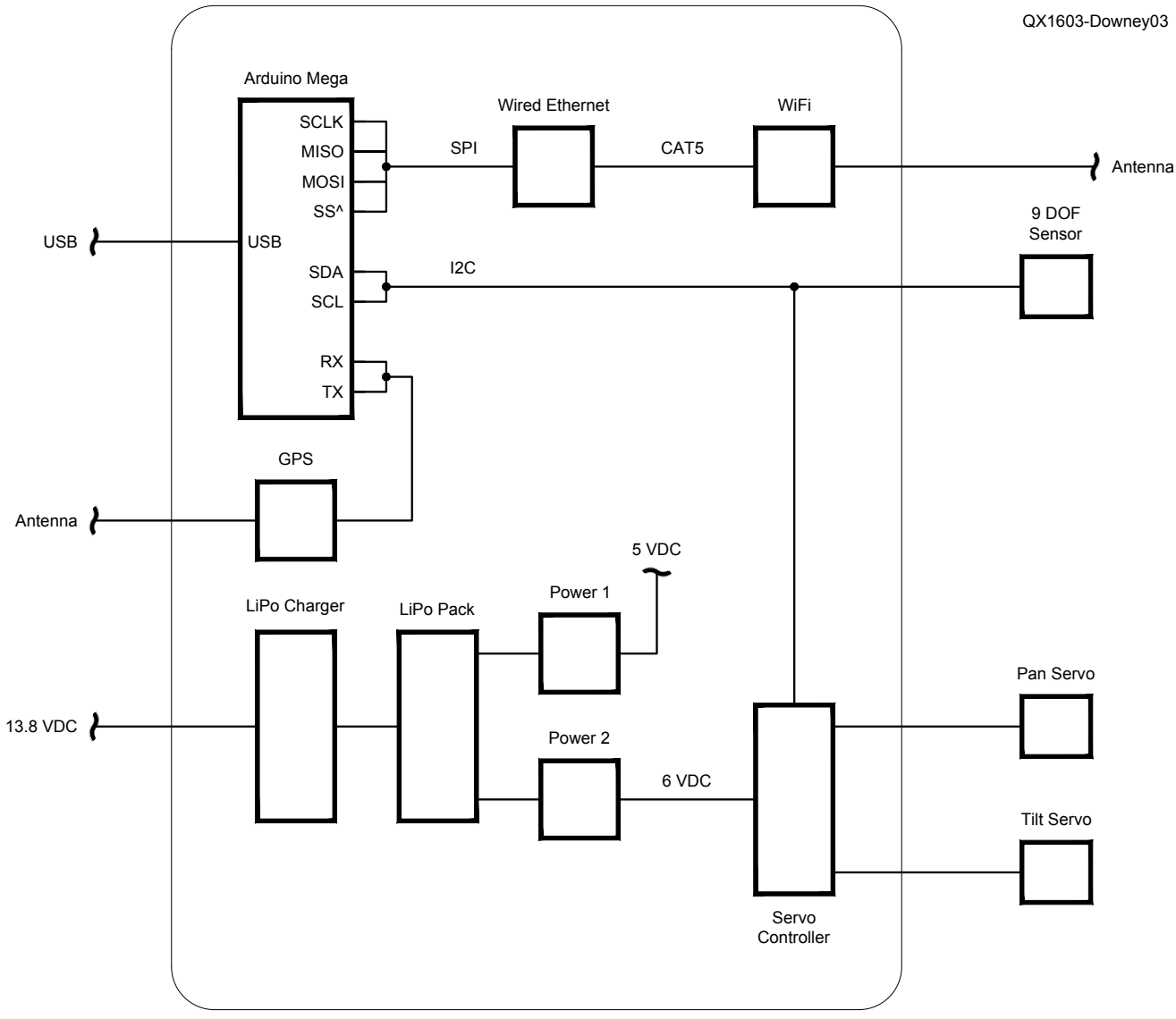


Figure 3 — Block diagram showing how each electronic subsystem interconnects. A partial bill of materials is in Table 1.

Table 1
Bill of Materials

Item	*Source*	*Approximate price*
Ardunio Mega 2560	Amazon: SunFounder Mega 2560 R3, stock number B00D9NA4CY	$18
Wired Ethernet shield	Amazon: SunFounder Ethernet Shield W5100 for Arduino, stock number B00HG82V1A	$16
Wi-Fi router	Amazon: TP-LINK TL-WR702N Wireless N150 Travel Router, stock number B007PTCFFW	$20
GPS module	Adafruit: Part ID 746	$40
Servo controller	Adafruit: Part ID 815	$15
Bosch 9 DOF sensor	Adafruit: Part ID 2472	$35
Pan platform	ServoCity: model SPG785A-CM, 5:1 ratio	$100
Tilt stage	ServoCity: model SPT400, 5:1 ratio	$95
	Total	*$339*

lead to each choice. The main processor is the Arduino Mega 2560. I began with the model Uno but eventually I could no longer squeeze everything into its 32 KB flash memory and 4 KB RAM storage. The Mega has 8 times as much flash memory and twice as much RAM, which is plenty. The Tracker uses about half of the Flash on the Arduino Mega for code and constant strings, and about half of the RAM for mutable variables, leaving 4 KB of RAM for stack.

To control the two hobby servo motors I initially used the Adafruit software servo library. However, the servos did not move smoothly. The cause turned out to be interference to the pulse timing by other libraries that lock out interrupts, even briefly. Servo position is directly related to pulse duration, which is sensitive to changes on the order of a few microseconds, so it doesn't take much timing change to cause unwanted motion. The solution My solution was the 12 channel servo controller from Adafruit. This offloads all the timing from the Arduino and requires only a two-wire connection using the I2C bus to issue the desired pulse length for each channel. It also has the added benefit that the servos actively hold position until a new command is issued.

I chose the Bosch BNO055 9-DOF sensor. It is available on a convenient breakout board from Adafruit and is compatible with their *Sensors* library. The advantage of this sensor package is it includes an onboard processor that performs all the consolidation of the three sensors automatically and outputs directly its absolute spatial orientation as Euler angles interpreted here as azimuth, elevation and roll.[6] As anyone who has tried to manually fuse together these types of sensors knows, this saves quite a lot of tedious mathematics. This sensor also connects to the Arduino using the same I2C bus as the servo controller but does not interfere because they each have a separate bus address.

I chose the GPS module from Adafruit. It has a built in antenna, which works pretty well, but I also allowed for the connection of an external antenna if necessary. This module communicates with the Arduino using a UART, or serial connection. This revealed an additional advantage to using the Arduino Mega: it has four hardware serial ports available, allowing one to be dedicated to the GPS. The Uno only has one that already serves duty with the USB boot loader. A software serial library could be used with the Uno to use other pins but at the expense of higher overhead and a more limited bandwidth.

I wanted Wi-Fi ability so I could control the system from my smart phone. I tried several Wi-Fi modules and shields but found none to be reliable. Even the best one from Adafruit would work for a random time, anywhere from seconds to hours, and then just mysteriously stop. In stark contrast, all models I tried of wired Ethernet proved to be 100% reliable, even including the oldest modules that use the WizNet W5100, so I ended up using a generic version made

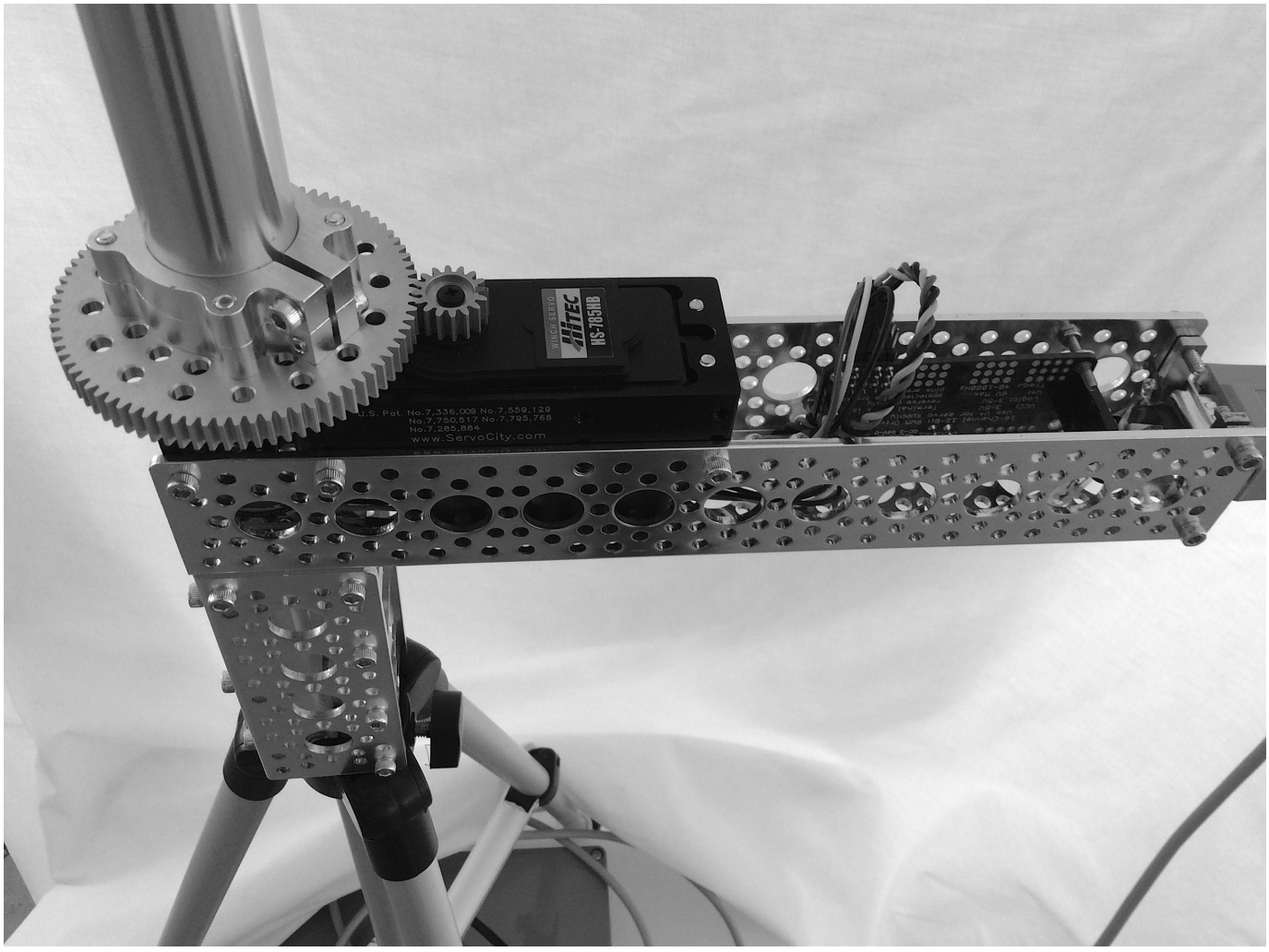


Figure 4 — The azimuth gimbal mounted on a tripod.

by Sunfounder. In order to accomplish my goal for Wi-Fi, I just connected the wired Ethernet directly to a $20 Wi-Fi adaptor made by TP-Link. This combination works beautifully. I have not experienced a single wireless communication glitch. The adaptor I bought can be configured either as its own Access Point to broadcast a separate Wi-Fi network just for the Tracker, or it can transparently bridge the Arduino to an existing Wi-Fi network. The unit comes with a simple Windows utility to perform the required one-time setup. From then on it comes up on its own every time.

I wanted everything to operate from one self-contained power source. I ended up using one LiPo battery and two separate power conditioning modules. One supplies 5 V to the Arduino and its peripherals, and the other is dedicated to powering the servo motors. This approach provides clean power to the electronics and isolates the wide load swings and voltage spikes that occur from the motors. I currently use a 7.4 V 2000 mAh pack, which operates the Tracker for several days of moderate use before needing a recharge. If desired, a solar pack could also easily be used.

The gimbal is one channel-mount pan platform (Figure 4) and one tilt platform (Figure 5) obtained from **ServoCity.com**. Together these provide about 400 degrees of azimuth motion and 135 degrees of elevation motion. Under the pan platform I installed a short section of channel with ¼-20 threaded screw plates for easy attachment to a common camera tripod. The pan platform has a hollow shaft that simplifies cabling to the 9-DOF sensor and tilt motor, and reduces tangles during rotations. I discovered that the servos would make spontaneous and sporadic moves while I am transmitting on 2 m FM with the Elk LPDA antenna. I eliminated this interference by using shielded STP CAT5 cable, taking care that only the end of the shield nearest the Arduino was connected to ground.

Assuming the Bosch spatial sensor is accurately aligned with the antenna, the largest contribution to pointing error is the sensor itself, which claims a maximum magnetic heading error of ±2.5 degrees. The next largest source of error is the orbit propagator software. The code, available in the *QEX files* web page, used here is based on a very clean rendering by Mark VandeWettering, K6HX, of the James Miller, G3RUH, *PLAN-13* code.[7,8,9] After the updates to the solar elements posted in 2014 the code produces topo-centric values within 0.2 degrees compared to a more rigorous *SGP4* code within a few days of the TLE epoch.[10]

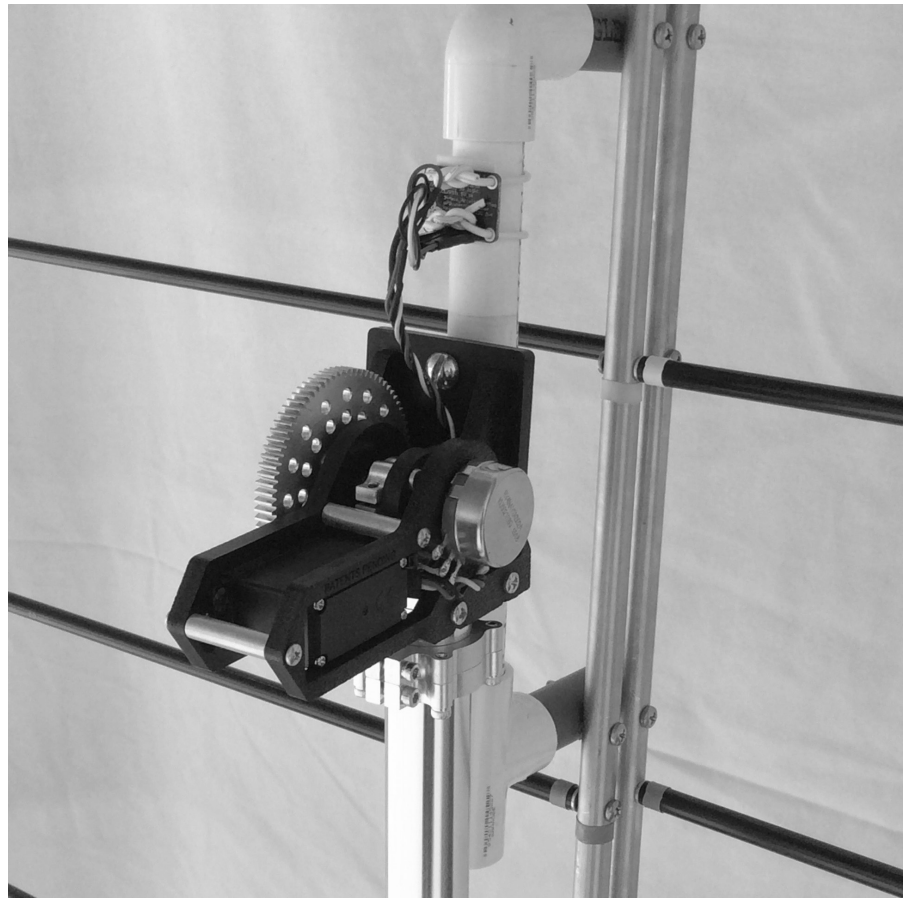


Figure 5 — The elevation gimbal shown with antenna attached.

Installation

In a nut shell, assemble the electronics and the gimbal. Attach them to your support then attach your antenna and the Bosch sensor. Attach your antenna to the tilt platform so it points straight up when the tilt platform is run all the way over on its side such that the plane of the tilt platform is also vertical. I attached my Elk LPDA antenna using two U-bolts after drilling four holes in the tilt plate. Position your antenna of choice on the tilt plate so the antenna is roughly balanced to help reduce the load on the tilt servo. There's plenty of torque so it should be fine to add a rear counter-weight to the boom if it allows you to balance the antenna better. Be aware that when the target is near zenith, the antenna will extend below the level of the gimbal. If you are using a tripod, add a vertical extension, otherwise when the antenna is pointed near zenith it will hit the tripod legs.

Attach the Bosch sensor breakout board such that:

– (1) the short dimension is parallel to the antenna boom,

– (2) the populated side of the board faces upwards and

– (3) the side with the control signals (SDA, SCL etc.) points in the rear direction of the antenna pattern.

Take some care to make this accurate and secure because the overall pointing accuracy is entirely dependent on how parallel the sensor is to the antenna bore site. The position along the boom does not matter, but since one of the sensors is measuring magnetic fields, mount it as far as possible from anything containing iron such as screws or U-bolts. It is not effected nearly as much by aluminum, but I would still stay at least an inch away from aluminum as well.

Power up the Tracker controller. Either connect with Wi-Fi or attach a CAT5 cable to the wired Ethernet controller. The default IP address is 192.168.0.122. If your computer is on the same network you can surf to that address and immediately see the main web page. If you want to change the IP of the Tracker, you have two choices. One choice is to edit the source code file *Webpage.cpp* (on the *QEXfiles* web page) and load a new image into the Ardunio. The other choice is to temporarily change your computer network to 192.168.0.0 so you can surf as above, then use the Tracker web page itself to

set a different IP address, reboot the Arduino then change your computer network back to your desired setting.

Once your web page is accessible, use the Gimbal section at the bottom to experiment with the motion range of each axis. There is no predefined assignment of which servo axis is azimuth or elevation. When setting the minimum and maximum for the elevation servo, make sure to consider the full range of azimuth. The minimum and maximum values are stored in EEPROM so they will retain their values through a power cycle.

Web Page Description

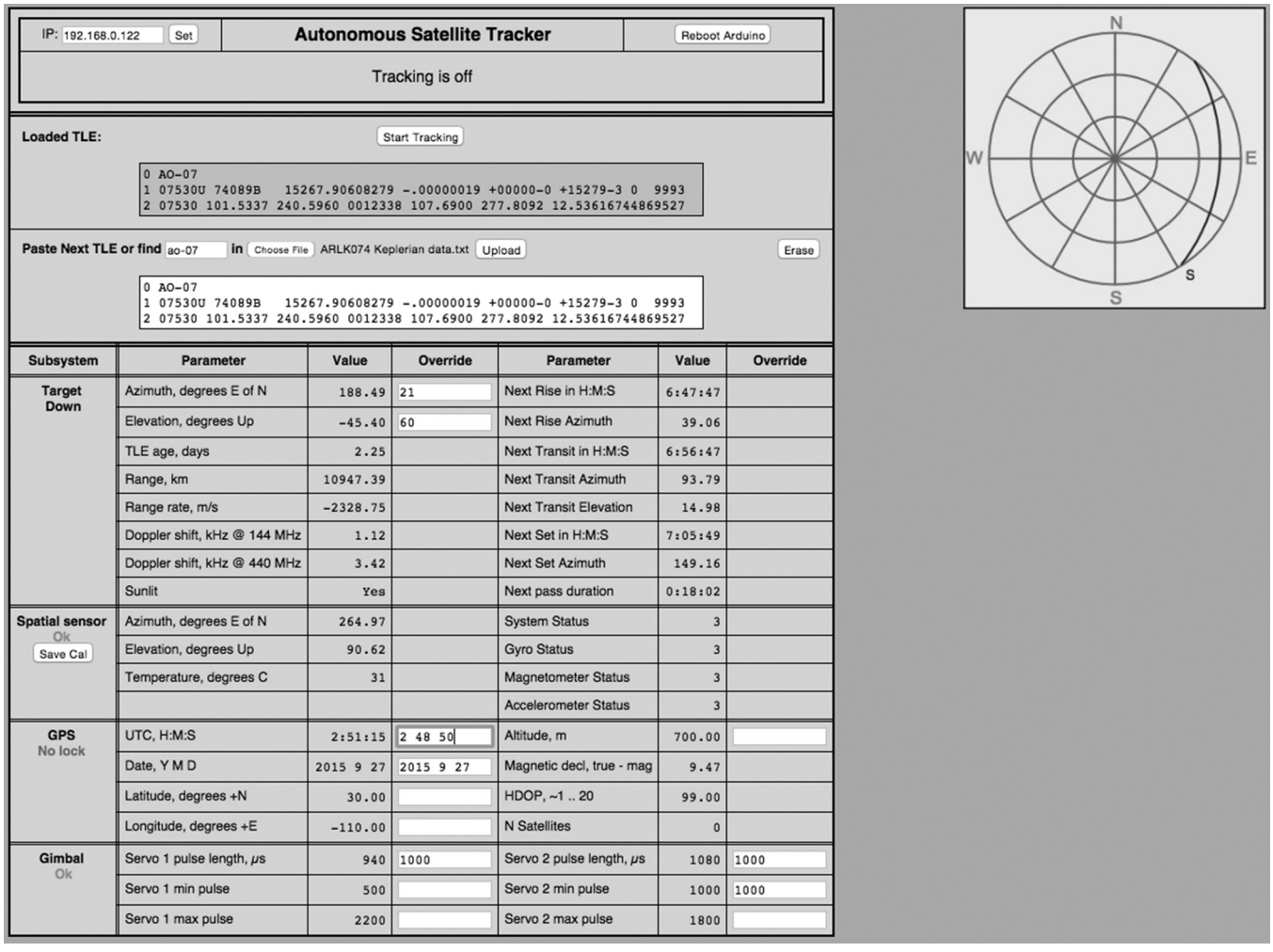
Turn on the Tracker controller and surf to its network address with your browser. You should see the web page shown in Figure 6. The page has two parts. The top part allows setting and inspecting the Two-Line Elements (TLEs) used to define the motion of the satellite of interest. The bottom part is a table showing detailed information for each of the Tracker subsystems of Target, GPS, Sensor and Gimbal. Look through the table carefully because it provides a lot of information and control capability. Most fields are self explanatory.

You will note that some of the data fields can be overwritten. This effectively turns off the automatic setting and allows you to enter your own values. Suggestions for how these can be used will be mentioned below.

Web Page in Detail

Across the very top is the title. Hovering over this title for a moment will display the software version. To the left is the network IP address of the Tracker. If this value is edited and Set, a new value will be stored in EEPROM and will be used the next time the Tracker is powered up or rebooted. To the right is a button to Reboot Arduino, mainly for this purpose. Below the title is the message line. Look here for confirmations, additional information and general messages as you use the page.

Aside from this bit of housekeeping, the top portion of the page mainly allows you to enter and Upload the TLE (two-line elements) for the satellite you wish to track. There are two text areas for showing TLEs. The topmost text area, with the darker background, is read-only and displays the TLE currently loaded into the Tracker, if any. The text area just beneath, with the white background, is writable. Here you can either copy/paste a TLE directly or you can type in the name of a satellite in the field provided and select a file that contains its TLE. The Tracker will scan through the entire file for a name match. The name is not case sensitive. The Tracker expects the file format to have the name on the line just before the TLE in typical fashion. If the satellite is found with a valid TLE, it will appear in the white text area. At this point the TLE is still just in your browser. To actually send it to the Tracker, click Upload. After successfully uploading a valid TLE, it will appear in the darker text area. This is the TLE that the Tracker will follow. You can change, or erase the writeable text area all you want and it won't matter unless you Upload it again.



Subsystem	Parameter	Value	Override	Parameter	Value	Override
Target Down	Azimuth, degrees E of N	188.49	21	Next Rise in H:M:S	6:47:47	
	Elevation, degrees Up	-45.40	60	Next Rise Azimuth	39.06	
	TLE age, days	2.25		Next Transit in H:M:S	6:56:47	
	Range, km	10947.39		Next Transit Azimuth	93.79	
	Range rate, m/s	-2328.75		Next Transit Elevation	14.98	
	Doppler shift, kHz @ 144 MHz	1.12		Next Set in H:M:S	7:05:49	
	Doppler shift, kHz @ 440 MHz	3.42		Next Set Azimuth	149.16	
	Sunlit	Yes		Next pass duration	0:18:02	
Spatial sensor Ok Save Cal	Azimuth, degrees E of N	264.97		System Status	3	
	Elevation, degrees Up	90.62		Gyro Status	3	
	Temperature, degrees C	31		Magnetometer Status	3	
				Accelerometer Status	3	
GPS No lock	UTC, H:M:S	2:51:15	2 48 50	Altitude, m	700.00	
	Date, Y M D	2015 9 27	2015 9 27	Magnetic decl, true - mag	9.47	
	Latitude, degrees +N	30.00		HDOP, ~1 .. 20	99.00	
	Longitude, degrees +E	-110.00		N Satellites	0	
Gimbal Ok	Servo 1 pulse length, µs	940	1000	Servo 2 pulse length, µs	1080	1000
	Servo 1 min pulse	500		Servo 2 min pulse	1000	1000
	Servo 1 max pulse	2200		Servo 2 max pulse	1800	

Figure 6 — The Autonomous Satellite Tracker web interface. Not the satellite track projection in the inset on the upper right.

Monitor and Control

Below the TLE section is the main table for monitor and control. The first table section is for the Target to be tracked. In the left column you will see observing details of the uploaded satellite elements at the time and location shown in the GPS section farther down. In the right column you will see information about the next pass. Note that the Tracker never computes information in the past, so if a pass is already underway (the satellite is currently above the horizon) then the Next Rise information will be for the subsequent complete pass, since that event has already occurred for a pass that is underway. You can override the computed azimuth and elevation. If tracking is enabled, this allows you to point your antenna at any desired fixed sky location.

Beside the table (or below if your screen is narrow) is an all-sky graph that shows the pass as it will look overhead. Again, if you override the time, and jump into the middle of a pass in progress, only the path from that moment onward will be drawn.

The Spatial Sensor

Below the Target Down section of the table is the section for the Spatial sensor. This displays the azimuth and elevation that it is measuring and reporting to the Tracker. It also displays the current temperature and the status of the system processor and each of the individual sensors. These individual status values can range from 0 through 3, where 3 is the best. The Tracker will not use the data unless all system status values report at least 1. Procedures for calibrating each sensor are provided in the Bosch manual.[11] The sensor package will need to be moved around to different orientations to get all sensors at their best values. Use the pulse length override fields in the Gimbal section (see below) to perform these motions. Once all sensors report state 3, their associated internal calibration data can be stored to EEPROM by clicking the Save Cal button. Once saved, these values will be restored each time the Tracker is powered up, and all sensors will usually immediately come up in state value 3. This button is available only when all sensors report status 3. The magnetic sensor is very sensitive to local magnetic fields and iron objects, so if you relocate the Tracker, I recommend that you perform the calibration again and store a new set of values.

GPS

The GPS section shows the reported time and location, and also displays some quality metrics. HDOP is the Horizontal Dilution of Precision.[12] This is an indication of the accuracy of the latitude and longitude, the position values most important to the Tracker. HDOP values range from less than 1, which indicates ideal conditions, up to 20 or more, indicating that location can be incorrect by 300 m or more. The number of satellites used in the fix is reported, where four or more is desirable. If you don't have a GPS connected, or it does not have lock, or you just want to experiment, you can override the time, date, latitude, longitude and altitude to see the effect on the passes. You don't need a GPS at all if you enter these data carefully.

Gimbal

At the bottom of the table is the Gimbal section. These data are in units of raw pulse duration. If you are aware of how hobby servo motors function, you will recall they are commanded to a given rotation angle determined by the length of a pulse issued on their control line. Pulse durations vary by manufacturer and even among devices of the same model. Roughly speaking, pulse lengths range from about 500 µs for one position extreme up to around 2400 µs for the other extreme. Normally pulse durations are set by the tracking algorithm, and bounded by the indicated minimum and maximum limit values. You can directly set specific pulse durations for each motor if you wish. Doing so will automatically disable tracking if it is enabled. This is fun, but also important to determine the safe as-built motion limits of each axis. The limits are stored in EEPROM and used by the Tracker to avoid exceeding the limits of each servo motor.

The Gimbal section also allows you to tune each axis for best tracking performance. Recall from the tracking algorithm that a motor is moved a small amount, stops to allow a stable sensor reading, then moves again repeatedly to track the target. Fields are provided for you to set the stop period and the step size for each move. The stop period should be set to just long enough for the entire gimbal and antenna to stop shaking after a move. The step size should be set to the smallest value that results in a reliable change in reported sensor position.

Operation

After everything is set up and you are comfortable with the safe operation of the Tracker motions, you are ready to track a satellite. Set the system up in a location with a good view of the sky. Turn it on, load the TLE into the white text area and click Upload. Click Start Tracking to begin tracking the satellite. That's all there is to it. Enjoy.

All photos courtesy of the author.

Elwood Downey, WBØOEW, has held the same call sign since he was first licensed in 1974. He is an ARRL Member. Elwood enjoys software, digital modes, antennas, and experimenting. He graduated with a BSEE cum laude in 1977 from Purdue University. Since then he has focused his career on telescope control systems and related astronomical instrumentation, which he finds very fulfilling. His career has taken him to many of the great observatories around the world.

Notes

[1]**siarchives.si.edu/collections/siris_sic_8335**.
[2]**https://en.wikipedia.org/wiki/Microelectromechanical_systems**.
[3]**https://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml**.
[4]**https://en.wikipedia.org/wiki/Gradient_descent**.
[5]**https://en.wikipedia.org/wiki/Gimbal_lock**.
[6]**https://en.wikipedia.org/wiki/Euler_angles**.
[7]**https://github.com/brainwagon/angst**.
[8]**www.amsat.org/amsat/articles/g3ruh/111.html**.
[9]**www.arrl.org/QEXfiles**.
[10]**www.xephem.com**.
[11]**https://www.bosch-sensortec.com/en/homepage/products_3/9_axis_sensors_5/ecompass_2/bno055_3/bno055_4**.
[12]**https://en.wikipedia.org/wiki/Dilution_of_precision_(GPS)Figure Captions**.