Overview

This makes two timed flashes that enable you to determine an occulted star's disappearance and reappearance times.

The flasher works either standalone or by computer control so it has no pushbuttons, status light, piezo beeper, nor local display.

How the flasher works

The flasher reads these input data from a micro SD card:
- predicted date and time of the occultation
- combined target star + asteroid magnitude – this is used to set the flash brightness
- desired flash duration
- interval you want between flashes.

The GPS receiver gets date, time and geographic location, once per second, from two NMEA (National Maritime Electronics Association format) sentences.

```
$GNGGA,123502.000,4042.2500,N,08823.4690,W,1,11,0.76,234.3,M,-33.9,M,,*4F
$GNRMC,123502.000,A, 4042.2500,N,08823.4690,W,0.10,0.00,120524,,,A*6F
```

and from the receiver's pulse-per-second (PPS) signal that precisely marks the start of every UTC second.

An Arduino Mega compares the current time and the prediction time and decides when to make two timed flashes, one on either side of the prediction – John Moore coined the term "goalpost" flashes for these. It also makes optional "zero-second" flashes at the start of every minute. Flashes are triggered by the PPS and always begin and end at the start of a second.

Flash times are logged to the SD card and, if a computer is attached, communicated via USB.

After you record the occultation (and flashes) you will use Pymovie and PyOTE to extract the star's disappearance and reappearance times.

Hardware

The flasher has four basic parts:

An Arduino Mega 2560 Rev 3 microprocessor board. Arduinos have an open architecture and Mega clones are commonly available and perfectly adequate. The official Mega and most clones use an old USB type B socket but this Keyestudio clone has a modern USB-C connector and may be preferable.

The Adafruit Ultimate GPS Logger Shield is available either from Adafruit or DigiKey. It has a GPS receiver with an integral antenna, a micro-SD card drive, and lithium battery (CR1220) backup. The shield comes with its header pin strips unsoldered, so you'll have to solder those on yourself.

A potentiometer (here is a selection) is mounted on the GPS shield's breadboard area or via wires to the enclosure. It is wired in series with a fixed resistor and the flash LED. The fixed resistor prevents high-current damage in case you inadvertently lower the potentiometer resistance to zero.

This LED comes with a built-in resistor so you won't have to mount a separate resistor on the breadboard.

Here is a header kit that allows you to interconnect Arduinos and peripheral components. A 6-pin female header is needed to connect the newest GPS Logger shield to the Mega.

You may want to provide a screw terminal block or socket and plug for the LED wire.

You'll need a micro-SD card. Data transfer rates are modest so you don't need a pricey HD video-capable card.

You might simply put the flasher in a ziplock bag at the base of your telescope, but if you want a proper enclosure use plastic or wood so the GPS signal won't be attenuated.

Wiring

The flasher is usually powered via the Mega's USB port but when it is used standalone without a computer it may be powered via the 2.1 X 5.5 socket @7.5-12 VDC, center positive. The Arduino operates at 5 volts via an onboard voltage regulator.

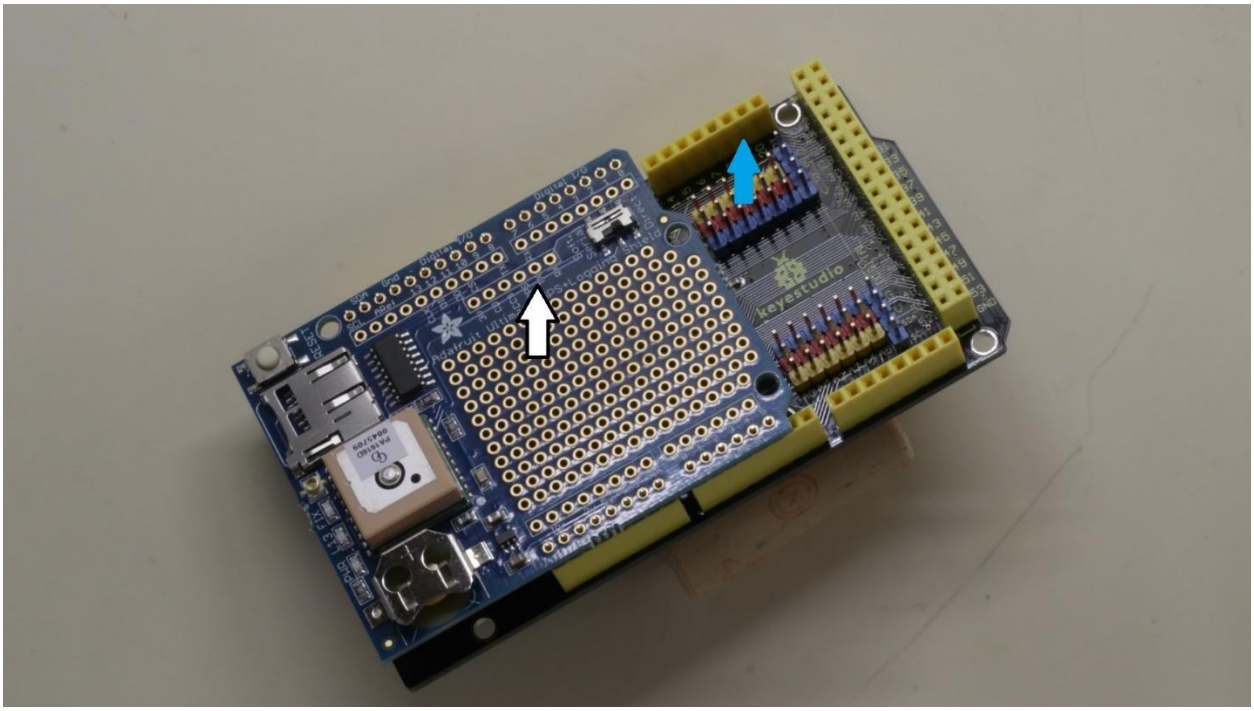
The GPS shield gets its power via the Mega headers (yellow in the pictures below) but the GPS data signals require these separate wires:

Mega digital pin 18 (Serial 1 TX) to GPS RX
Mega digital pin 19 (Serial 1 RX) to GPS TX
Mega digital pin 20 (hardware interrupt 1) to GPS PPS

Note: most of the photos below show an older version of the Adafruit Ultimate GPS Logging shield, but the last three pictures show the latest shield.

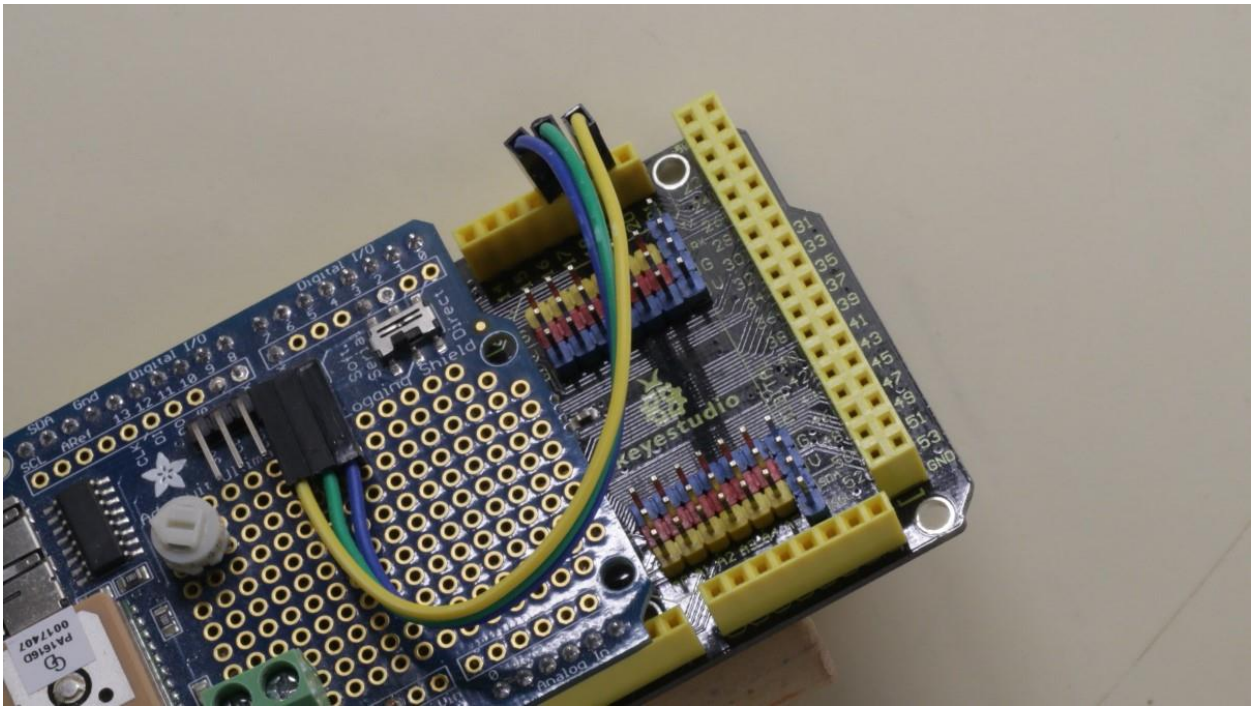
Here is a picture of the GPS shield placed on the Mega. The actual receiver is the square brown device between the SD drive and the battery holder. The silver area—it looks white in the picture—on top of the receiver is a tiny patch antenna. When you look closely at the edges of the GPS shield you'll see the tips of the header pin strips that have been inserted into the Mega headers, ready to be soldered.

The small slide switch on the GPS shield must be moved to the "Soft Serial" position.

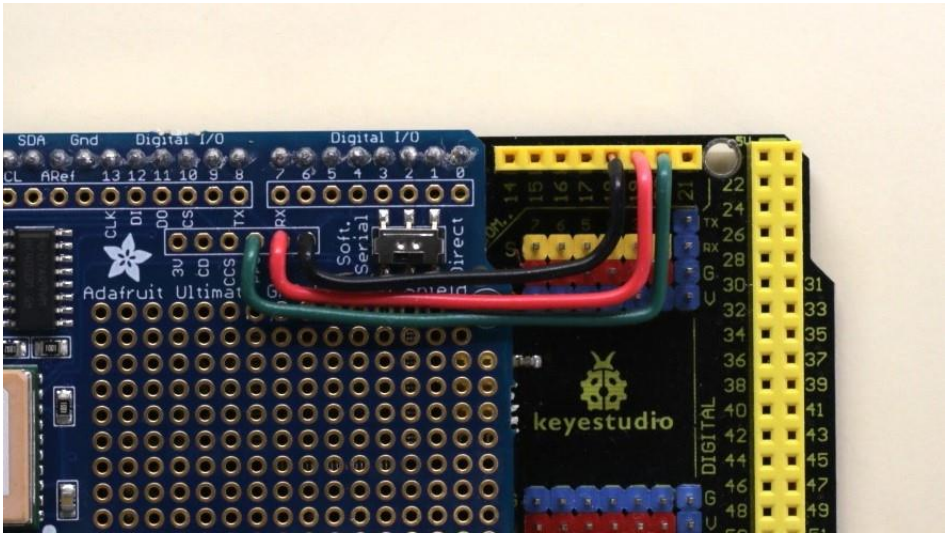


The white arrow points to the solder holes for the GPS signal lines PPS, RX and TX, but at the blue arrow Mega pins 18, 19 and 20 are part of the header and are not easily soldered. Below are photos that show four alternative ways to connect these.

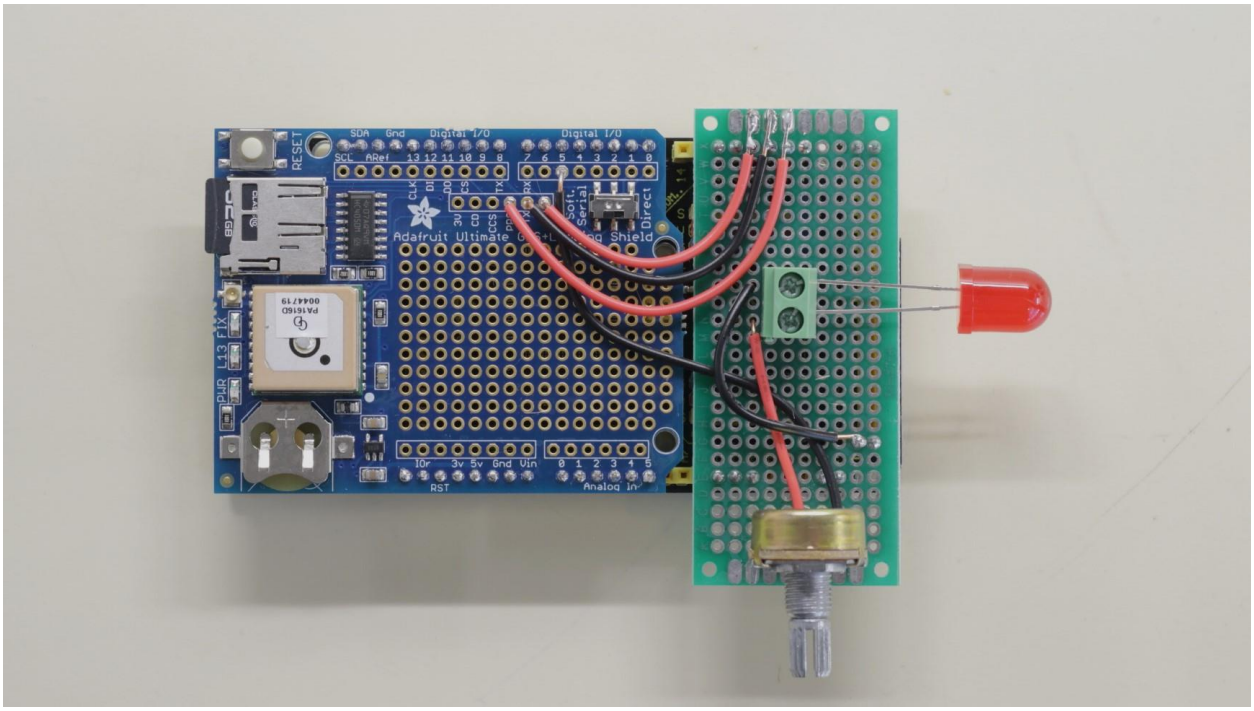
In this one header pins are soldered to the top of the GPS shield and standard breadboard flying leads are used to connect the shield and the Mega.



A second way is to solder wires to the GPS shield and simply plug them into the Mega header. Use only solid wire and be careful with the wire gauge—if the wire is too thin it won't make a reliable connection in the header socket, and if it's too thick it may damage the header. The wires in the picture are 22 gauge (AWG) and have copper conductors about 0.65mm in diameter.

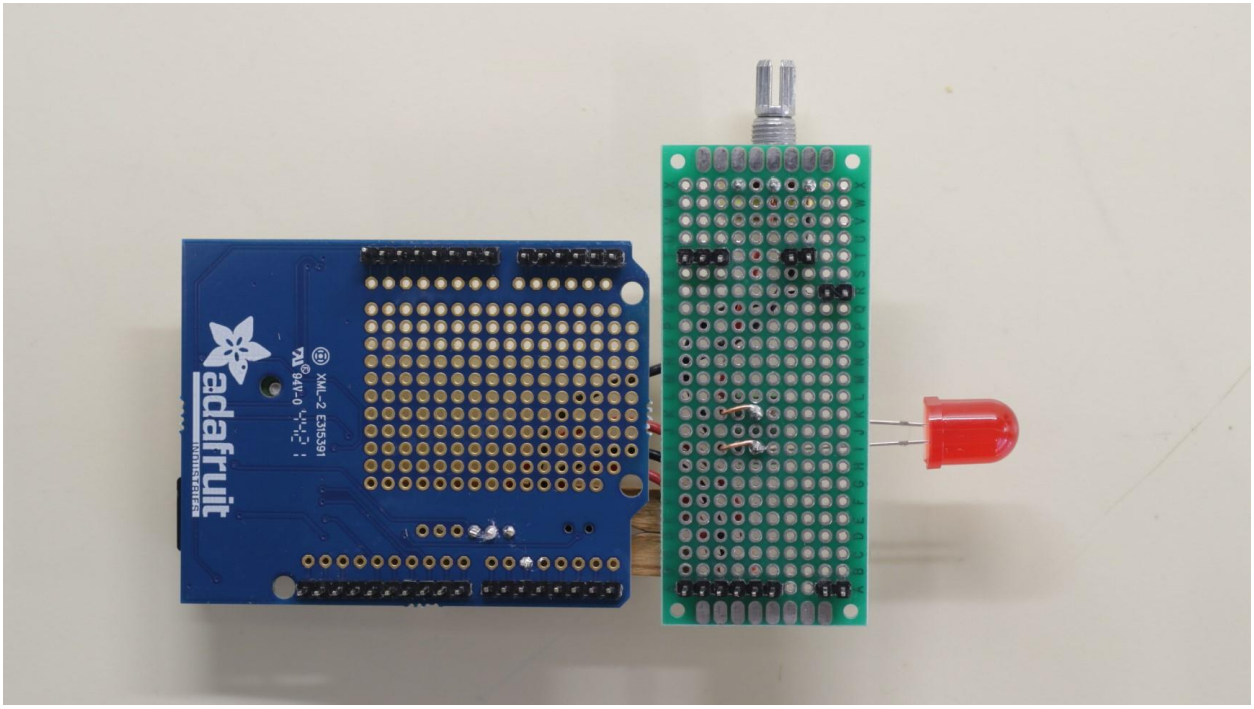


A third way is to mount a small perf board over the uncovered part of the Mega, using header pins just like you did for the GPS shield, and solder wires between the GPS shield and the tops of the header pins. The perf board can be used to hold the potentiometer and a connector block for the LED wire – here it's holding a test LED.

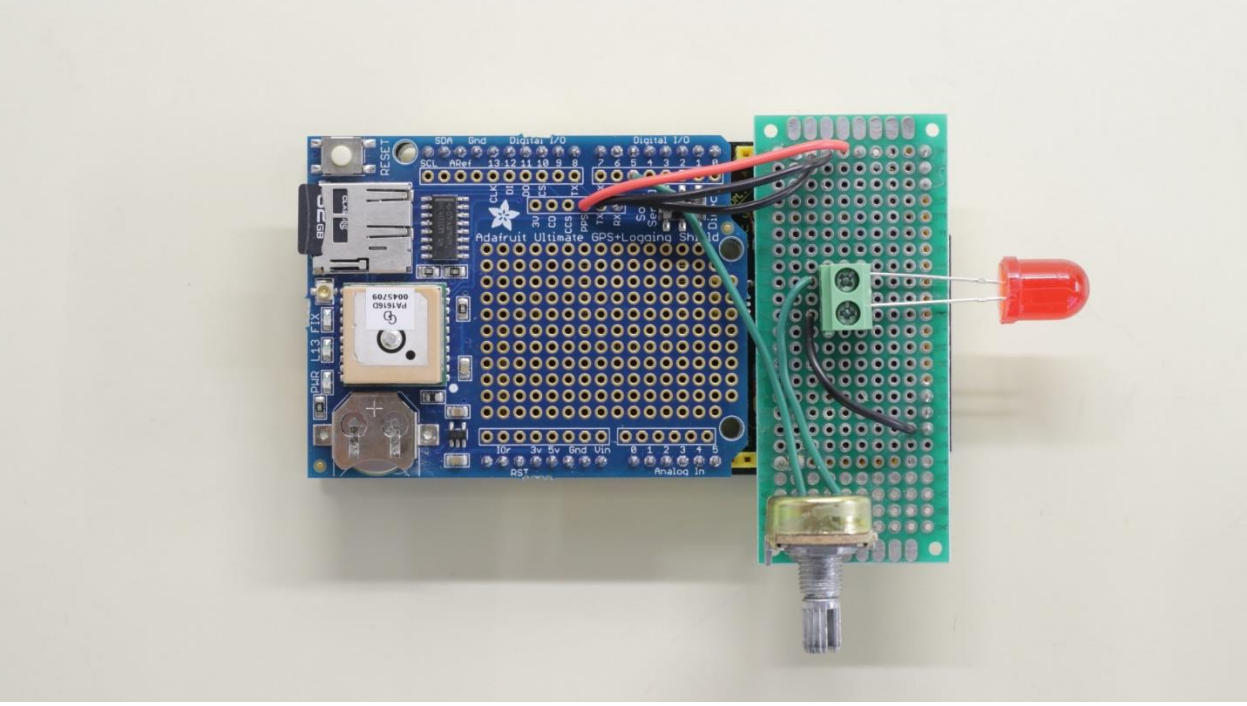


Header pins are close together so on the perf board I positioned and soldered the wires alongside them, instead of wrapping around each pin, and also soldered to the solder pads at the edge of the board to strengthen the connections.

Here is the back of the above perf board, showing the placement of header pins at the corners to position the board on the exposed part of the Mega.



Another way is a hybrid approach, where standard header pins are used to anchor the perf board, but wires (22 gauge) are soldered to locations for Mega pins 18,19 and 20 (for GPS) and for the LED negative connection (black wire to GND at the right). The wires themselves become header pins:

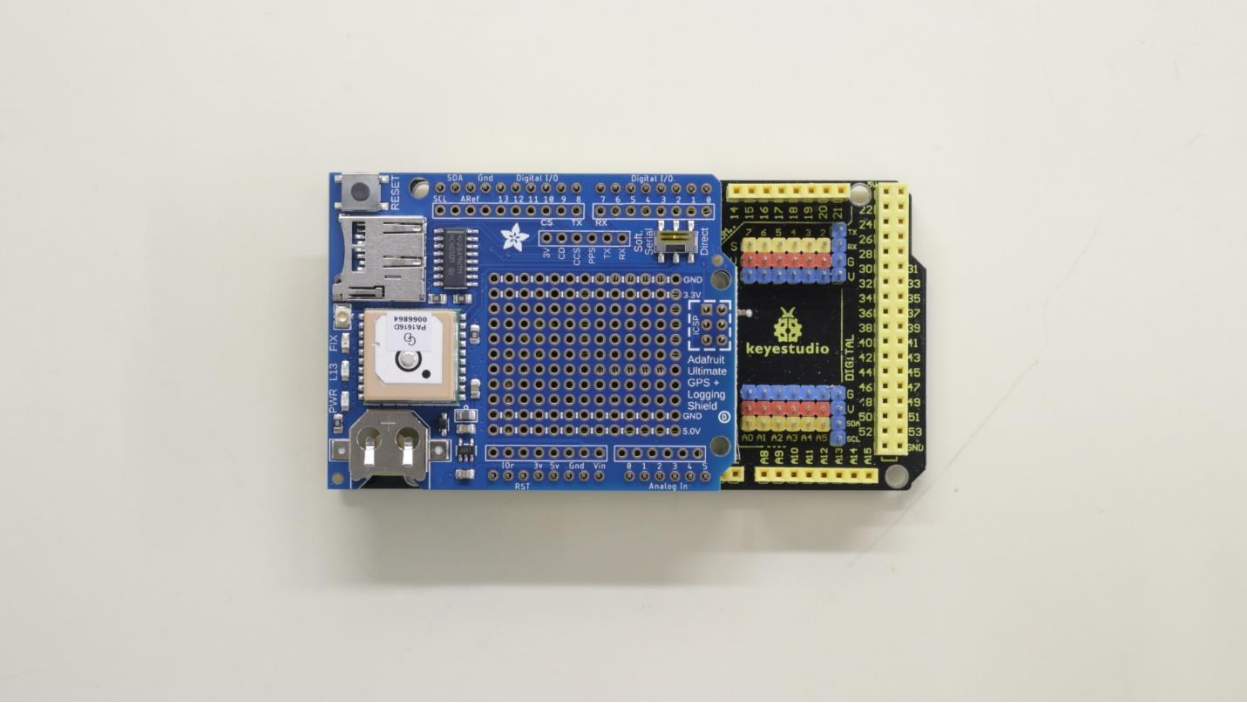


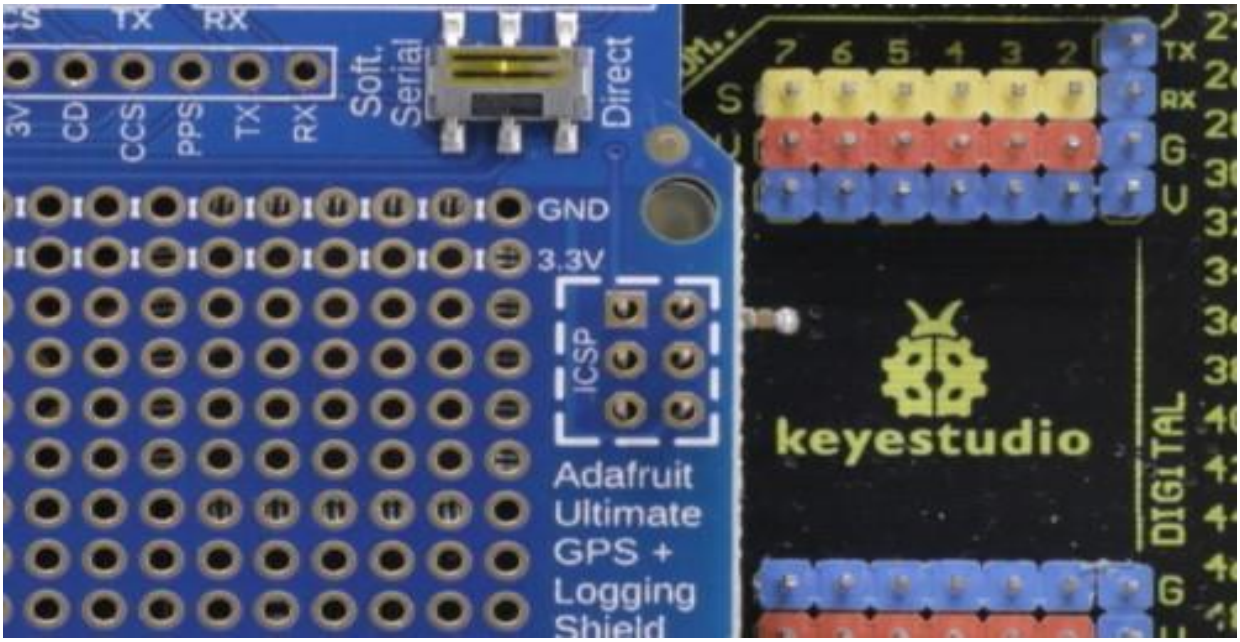
The LED is wired like this:

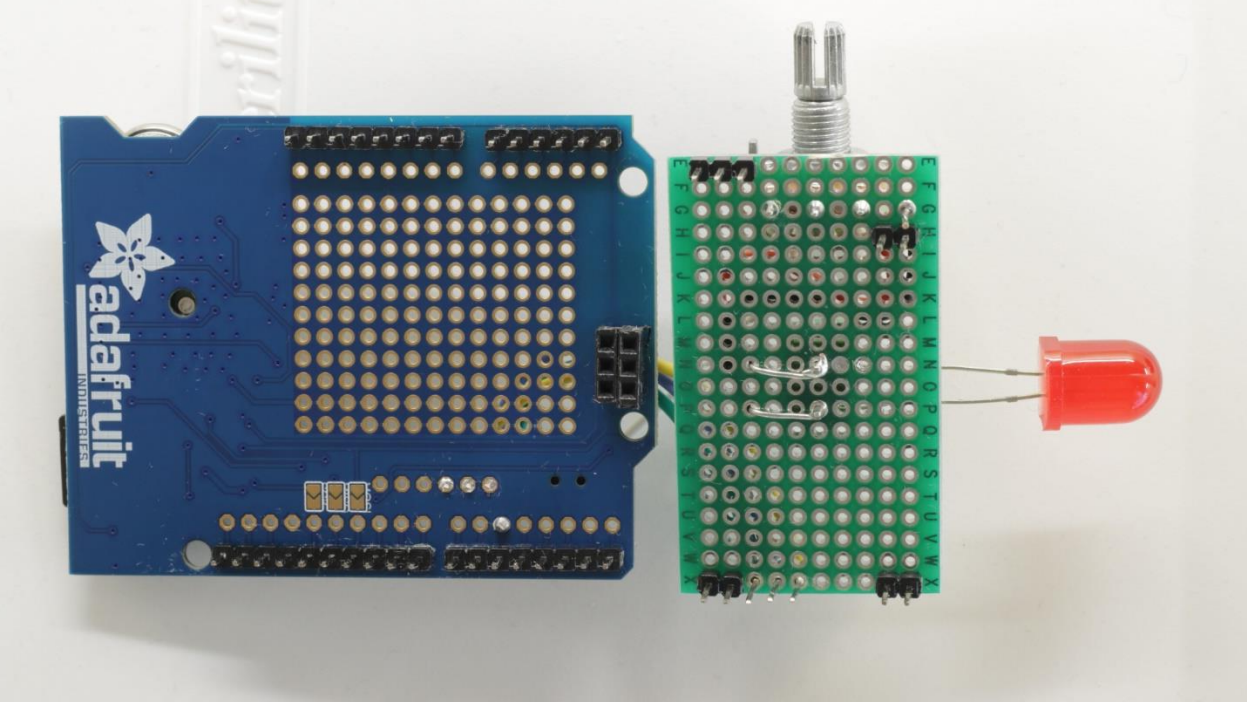
Mega digital pin 5 via potentiometer and fixed resistor to LED anode.
Mega ground (GND; there are several on the Mega & GPS shield) to the LED cathode.

On a naked LED the anode (positive) lead is longer than the cathode (negative) lead. If the LED is pre-wired the anode wire is normally red and the cathode wire black.

The above photos show an older Adafruit Ultimate GPS Logging shield. The newest version of the Adafruit shield, shown below, connects its SD drive to the Mega via a header that mates with six ICSP pins on the Mega. This requires a 6-pin female header soldered underneath the GPS shield.


keyestudio
Adafruit
Ultimate
GPS +
Logging
Shield


TX
RX
Soft.
Serial
Direct
3V
CD
CCS
PPS
TX
RX
GND
3.3V
ICSP
Adafruit
Ultimate
GPS +
Logging
Shield
keyestudio
DIGITAL



The older and newer GPS shields use different SD libraries, explained in more detail below. The wiring for RX, TX, PPS and LED is the same for both types of shield.

Flasher placement

Flashes raise the background illumination of the star field that is being recorded, so locate the LED where its light will be seen by the camera. The camera doesn't image the LED.

The Mega and GPS shield should be placed where the power (USB usually) cable can reach and where the GPS antenna can see the sky. The GPS shield has a uFL connector for an external antenna, if needed.

The Mega and the GPS shield both have their own LEDs that flash when processing data; some black electrical tape can hide these.

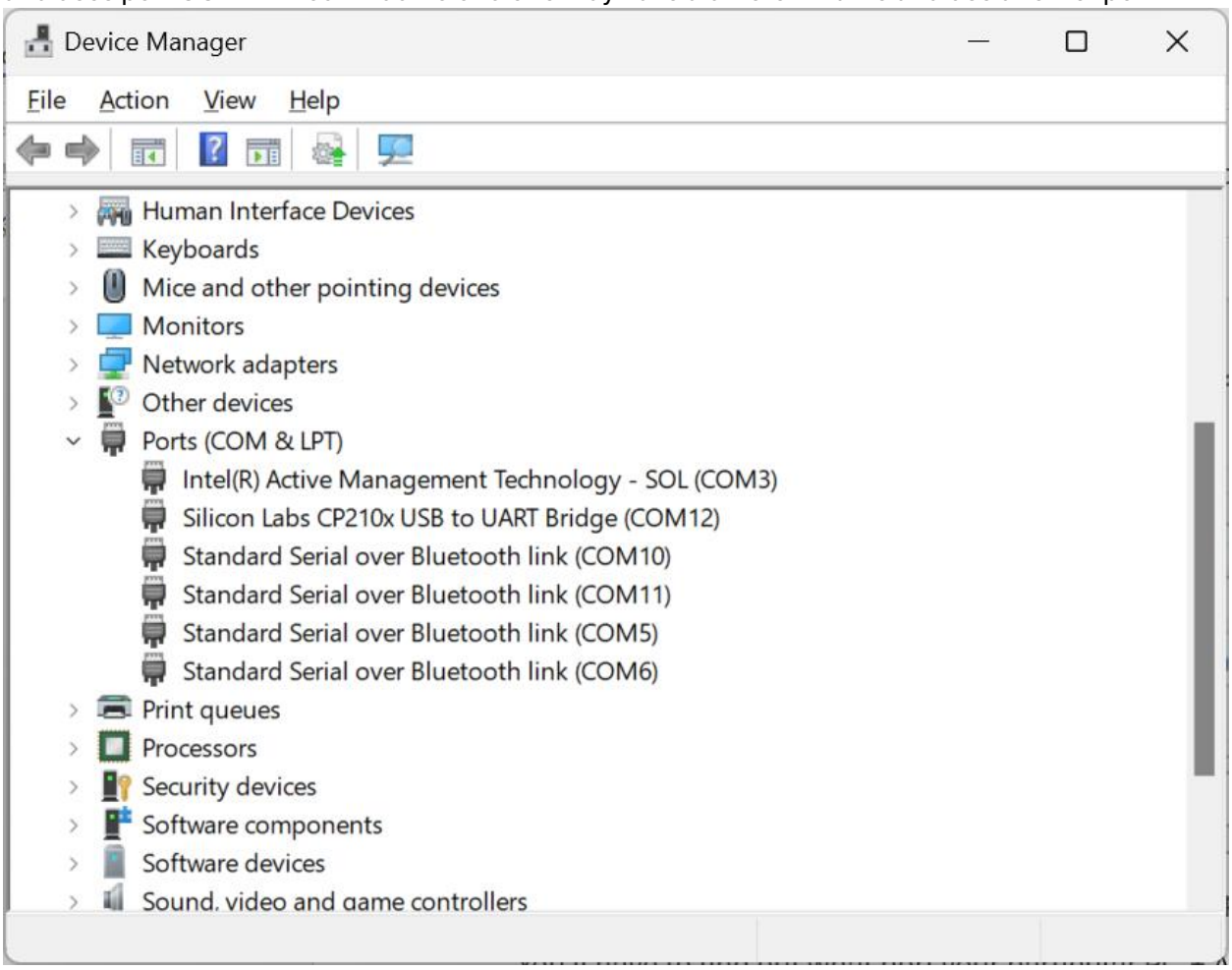
USB and COM port

To program the Arduino and to operate the flasher in real time you'll use its USB port.

First connect the Mega via USB and find out what COM port your computer + Arduino combination uses:

Windows Start menu > Device Manager > Ports (COM & LPT)

My Keyestudio Mega clone is identified as “Silicon Labs CP210x USB to UART bridge”,
and uses port COM12. Your Arduino or clone may have a different name and use another port.

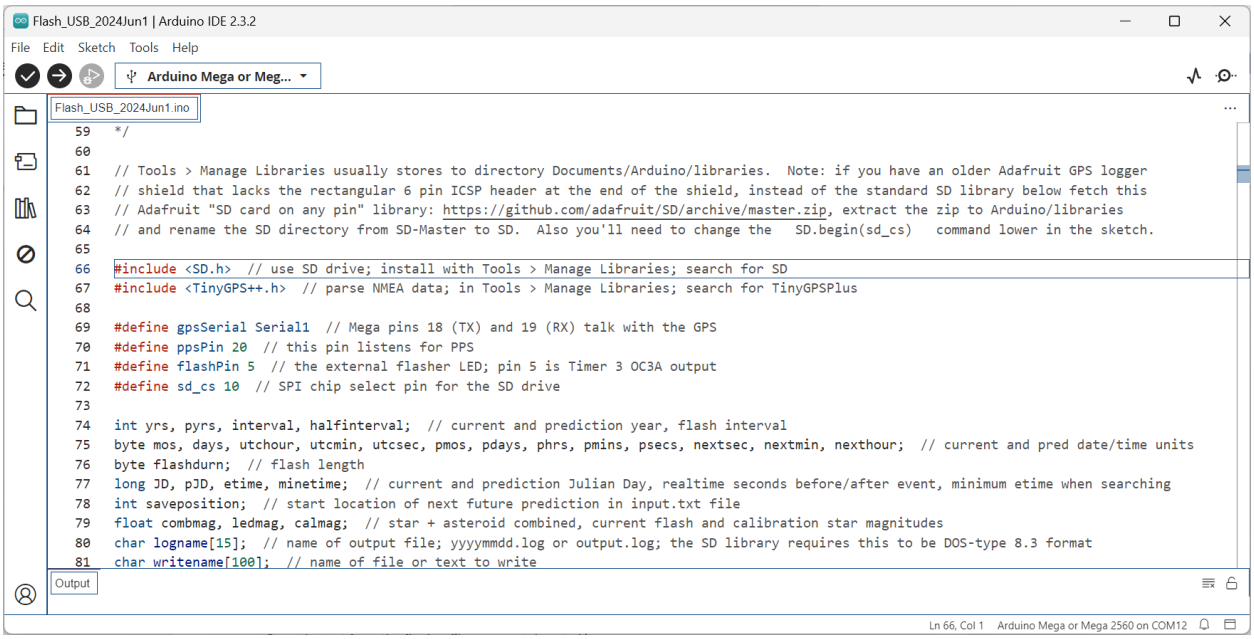


Once you know the port it will usually remain the same between connected sessions.

The Arduino IDE

To program the Mega, you will use the Arduino IDE (integrated development environment) to edit and up/download the compiled code. The IDE is free but you can make an optional donation to further Arduino development.

The main IDE screen, here showing part of the flasher sketch – Arduino programs are usually called “sketches” – looks something like this:



Copy the text from the flasher file you want, located in

https://groups.io/g/IOTAoccultations/files/Aart's Arduino Timers

If there is any Arduino code in your newly-downloaded IDE (It starts with a blank startup sketch), first Edit > Select All and delete it, and then paste your copied text into the IDE window.

Then save the sketch to a new sketch directory, using File > Save As. Arduino software, the IDE and related files, are usually stored in your Documents / Arduino directory. Sketches may be stored there by default, but the contents of the Arduino directory are deleted when you upgrade to a new IDE version, so you are strongly advised not to store your sketches there. Create and name a working directory something like "Arduino Sketches" inside your Documents directory, and store your sketches there.

Editing the code is quite easy but won't be discussed here. The Arduino family uses the C++ programming language, also relatively easy to learn but beyond the scope of this document. There is a very large Arduino community and a corresponding wealth of information on the internet – just start a search with "arduino" to get an answer to just about everything.

In the bit of code shown in the image above notice the comment advising you to install the SD and TinyGPS++ libraries. Libraries are utilities that perform specialized functions – these provide access to the SD drive and to the GPS receiver. Libraries are installed from the IDE using Tools > Manage Libraries and are usually placed in your Documents / Arduino / Libraries directory.

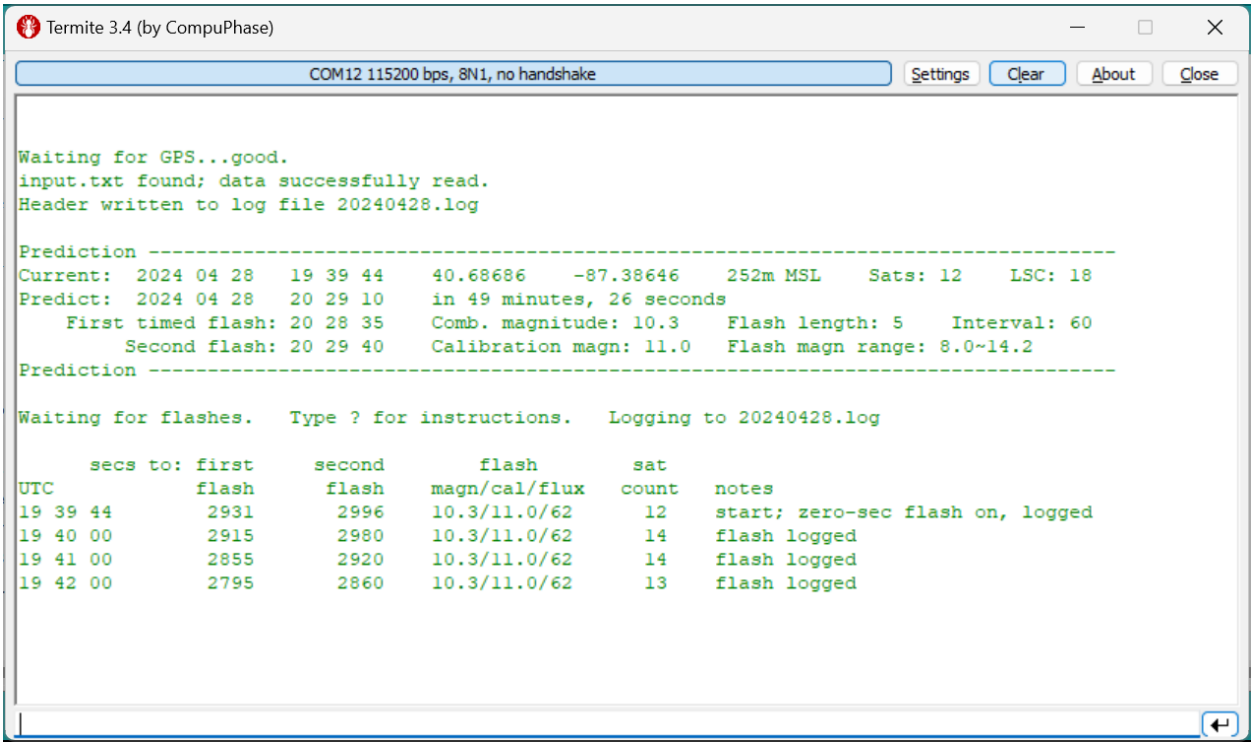
With the latest Adafruit GPS shield – the one with the ICSP header – both the SD library and the TinyGPS++ library are installed in the standard way, but if you have an older GPS shield you'll need a special Adafruit "SD card on any pin" library. Instructions are in the sketch comments.

After you install the two libraries the C++ code in a sketch should be complete and you can compile and upload it to the Arduino Mega. Near the top left of the IDE see the checkmark icon; clicking it

will check the code for errors and write error messages. The right-arrow icon next to the checkmark will compile the code and, if it finds no errors, will then upload it to the Mega. After this is done and the flasher's components are connected, it will become operational.

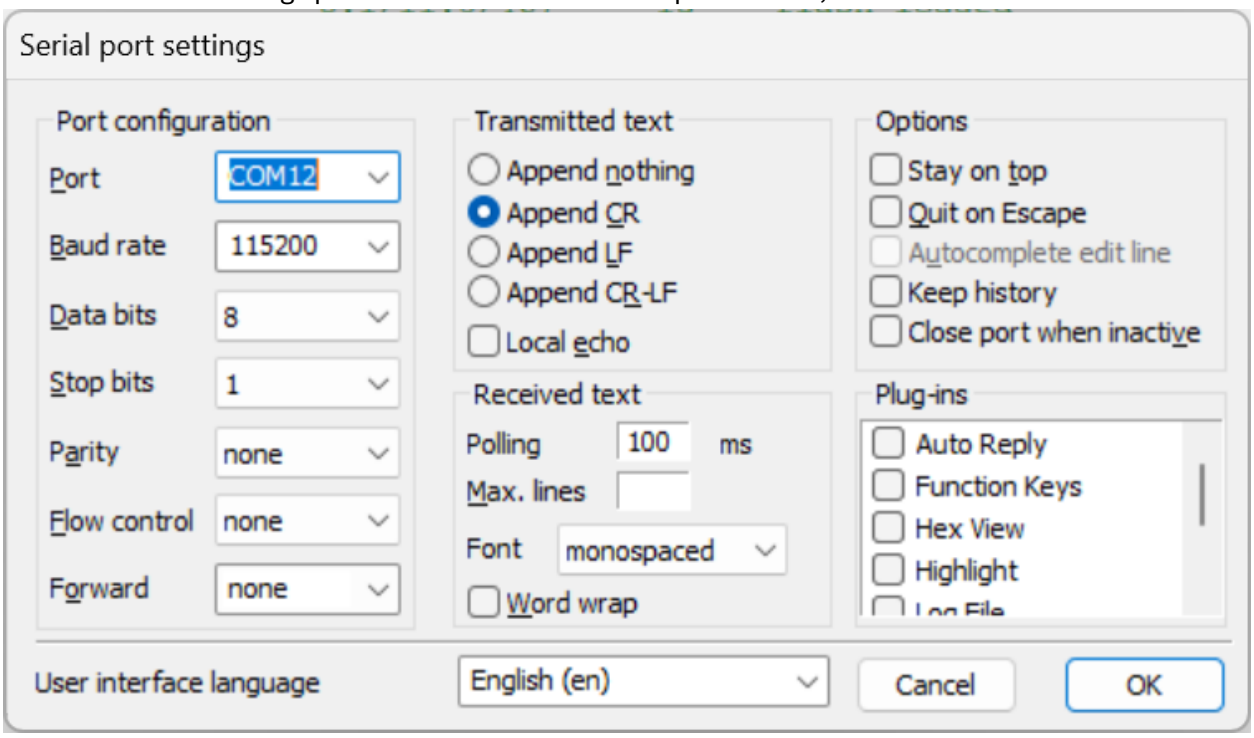
Connect to the flasher

To operate and manage the flasher via computer you will use the USB connection and a terminal window:



This is Termite, a terminal program that is ideal for our purpose. Termite is clean, easy to use, robust, and it is free. Download and install as directed; you'll see an icon on your desktop.

Use the Termite Settings panel to set communication parameters; most are the defaults.



Select the port you found in Device Manager, 115200 baud, append either carriage return (CR) or line feed (LF) – but not both – to transmitted text, and select monospaced font.

Prediction input

Prediction information is read from a file named input.txt on the SD card. If your laptop has an SD drive, the input.txt data may be written directly onto an SD card which is then inserted into the flasher's card reader, but usually it's better to leave the card permanently in the flasher and use Termite to edit and read input.txt. Keyboard commands and data entry will be discussed below but first let's see what the data look like.

Each input.txt prediction is a single line of nine data values: eight integers and one decimal-pointed number:

```
2024 5 1    3 45 12    10.7    5    60
yyyy mm dd    hh mm ss    n.n    s    ss
```

These are prediction date, prediction time, combined magnitude, flash duration, flash interval

Prediction time – UTC of course – is the expected time of mid-occultation, at the observing location, as provided by OccultWatcher or Occult.

Combined star + asteroid magnitude, the only number that may have a decimal place, will be used to auto-adjust the LED brightness.

Flash Duration should be longer than at least a few frames. 5 seconds makes it easy to find the flashed frames when searching through a long recording.

Flash Interval is the total interval between flashes—i.e. the occultation will happen near the middle of this interval. Flashes 30-60 seconds apart are good for an asteroidal occultation that may last a few seconds.

Put spaces between the numbers.

Multiple predictions may be entered, one per entry line. Predictions will be processed in chronological order but it is not necessary to enter them in order.

```
2024 5 1    3 45 12    10.7    5    60
2024 5 1    4 12 17    12.7    5    60
2024 5 1    7 37 30    11.8    5    60
```

If any predictions have the same date and time, only the last one entered will be used; effectively it overwrites any previous entries that have the same time.

Predictions are saved to the input.txt file as they are added, but a prediction becomes active only when the flasher starts, so restart the flasher after you add the last entry.

Be sure to input exactly 9 numbers for each prediction. Values are checked – e.g. month must be between 1 and 12 – and if any are invalid the prediction will be ignored, but if an entry line has fewer or more than nine numbers then it and all the subsequent entry lines will be misread and ignored.

Flasher routine operation

In routine operation, after the LED is calibrated, you will operate the flasher like this:

- add one or more predictions to input.txt:
  yyyy mm dd hh mm ss combined magnitude duration interval

- restart the flasher

The flasher will read input.txt, find the next prediction, set the flash brightness to the combined magnitude, and start the countdown to the two goalpost flashes.

During the countdown, zero-second flashes will be made at the start of every minute. These tell you that the receiver is getting GPS data and that the flasher is operating properly.

After each prediction's second goalpost flash is made, the flasher will restart and look for the next prediction.

After all the predictions have finished the flasher will stop the zero-second flashes so they won't interfere with other imaging. You can just ignore the flasher until it is needed again.

The above operational steps are all there is for standalone use, but when the flasher is connected to a computer via USB it can be monitored and managed in real time; details are described below.

Operational details

Power the flasher by plugging into the USB port when you start your other telescope equipment.

Start Termite when you are ready to use it. Connecting or disconnecting Termite doesn't interfere with the flasher so you can exit and restart Termite whenever you want, or simply minimize or unminimize its window as needed.

```
Waiting for GPS...good.
input.txt found; data successfully read.
Header written to log file 20240428.log
```

When the flasher is first plugged in or restarted, the screen will pause at "Waiting for GPS..." until the incoming satellite data are confirmed to be valid, and then will append "good". If the receiver's downloaded almanac of satellite positions is up-to-date, "good" may take just a few seconds but if the almanac has become outdated or if the geographical location has moved it may take a minute or two to download renewed almanac data for the first few satellites.

Note the information about the input file; note also that the prediction date becomes the log file name.

```
Prediction --------------------------------------------------------------------------------
Current:  2024 04 28   20 01 39    40.68693    -88.39642    253m MSL    Sats: 14    LSC: 18
Predict:  2024 04 28   20 29 10    in 27 minutes, 31 seconds
    First timed flash: 20 28 35    Comb. magnitude: 10.3    Flash length: 5    Interval: 60
          Second flash: 20 29 40    Calibration magn: 11.0   Flash magn range: 8.0~14.2
Prediction --------------------------------------------------------------------------------
```

After the GPS data are "good" and If the flasher finds a valid prediction in the input.txt file, its details will be shown in a panel like the above. Most of the information is self-explanatory: the prediction data you entered via input.txt, the current GPS time and location, the time remaining until the prediction, and the actual clock times of the two goalpost flashes before and after the expected occultation.

A couple of the items in the prediction panel should be noted:

LSC, leap seconds correction, or offset, is what the GPS receiver uses to calculate the difference between GPS time (generated by atomic clocks in the satellites; does not use leap seconds) and UTC (does use leap seconds). The flasher shows the LSC here and also logs the LSC at every flash so you'll always be aware of the value being used.

The calibration magnitude at the bottom is used to center the adjustable flash magnitude range. Details of this will be discussed later.

Countdown display

```
      secs to: first     second        flash        sat
UTC            flash      flash    magn/cal/flux   count   notes

20 01 39        1616       1681    10.3/11.0/62      14    start; zero-sec flash on, logged
20 02 00        1595       1660    10.3/11.0/62      14    flash logged
20 03 00        1535       1600    10.3/11.0/62      15    flash logged
20 04 00        1475       1540    10.3/11.0/62      13    flash logged
```

After a prediction is read from input.txt and becomes the current, active prediction, a countdown display like the above will show the seconds remaining to its two goalpost flashes, and related information.

Flash brightness is expressed as a magnitude, so you can easily compare it to the adjacent calibration magnitude. The flash LED brightness is also shown as a raw flux which has a linear range between 1 and 511. More about these later.

The satellite count is typically between 10-18.

The Notes area shows information about the flashes and what has been written in the log.

Notice the three zero-second flashes in the above example. When there is an active prediction ZSFs default "on", and so does ZSF logging, but both may be turned off.

A couple of log file examples will be shown below.

The countdown display returns after keyboard commands have been executed and a new countdown will start after a prediction is read or the flasher is restarted. The countdowns and ZSFs stop after the last prediction has been flashed.

Keyboard Commands

```
Input instructions ------------------------------------------------------------------------
 - press Enter for countdown times     ? for these instructions
 - see [p]rediction information       [nn.n] to set flash magnitude
 - make a [f]lash     toggle: [b]link    [a]lways on    [z]ero-sec on+log / on-log / off
 - [write a comment to the log file]   100 characters maximum per line
 - show [i]nput.txt file    [e]mpty input.txt file    show log [d]irectory
 - show current [l]og file or older [yyyymmdd.log] log file    [remove yyyymmdd.log]
 - add prediction  [yyyy mm dd    hh mm ss    nn.n    s    s]  to input.txt
     UTC date & time, combined magnitude, flash duration, flash interval
 - [r]estart flasher    set [c]alibration star magnitude (currently 11.0)
 - AFTER PREDICTIONS ARE ADDED, [r]ESTART TO MAKE USE OF THE NEW DATA
Input instructions ------------------------------------------------------------------------
```

Entering **?** from the keyboard will list the commands you may use while the flasher is waiting for its flash times. Any of these may be done at any time, but it's best to minimize activity and USB traffic while capturing images or when flash times are nearby.

Tapping the **Enter** key at any time will show you the current UTC, countdown seconds and other information.

```
12 24 06         635        700    11.7/12.0/42      15    not logged
```

p will show the current, active prediction.

```
Prediction ------------------------------------------------------------------------------
Current:  2024 05 13   12 25 07    40.68683    -88.39641    244m MSL    Sats: 16    LSC: 18
Predict:  2024 05 13   12 35 16    in 10 minutes, 9 seconds
    First timed flash: 12 34 41    Comb. magnitude: 11.7    Flash length: 5    Interval: 60
          Second flash: 12 35 46    Calibration magn: 12.0   Flash magn range: 9.0~15.2
Prediction ------------------------------------------------------------------------------
```

Various flash behaviors can be made:

f makes a **flash** at the next second and logs the time

```
12 24 58         583        648    11.7/12.0/42      16    flash logged
```

a =**always-on** causes the LED to turn on and remain on.

```
12 25 46         535        600    11.7/12.0/42      15    LED on
```

b = blink causes the LED to flash repeatedly at the rate of one second on, one second off. The on-off transitions are PPS-triggered and precise.

```
12 25 50         531        596    11.7/12.0/42      15    blink
```

Typing the same letter again or tapping the Enter key will turn the LED off. Note the message to remind you when the LED is on or blinking. Always-on and blink will turn off automatically one minute before the first goalpost flash.

c lets you set a new **calibration** magnitude. This is explained below.

You can write a comment to the log file
Anything that doesn't look like one of the other keyboard commands will be written to the log file as a comment and will be echoed back to the terminal window:

```
log: clouds 2/10; temperature 15C
```

l displays the current **log** file

```
20240513.log ----------------------------------------------------------------------------

prediction: 2024 05 13   12 35 16    comb magn: 11.7   flash length: 5    interval: 60
date         UTC   secs to pred    latitude  longitude  alt(MSL)  flash/cal   sats  LSC
2024 05 13   12 24 00        676    40.68682  -88.39642   243m     11.7/12.0    16    18
2024 05 13   12 24 58        618    40.68683  -88.39640   245m     11.7/12.0    16    18

prediction: 2024 05 13   12 35 16    comb magn: 11.7   flash length: 5    interval: 60
date         UTC   secs to pred    latitude  longitude  alt(MSL)  flash/cal   sats  LSC
2024 05 13   12 26 00        556    40.68683  -88.39639   245m     11.7/12.0    16    18
clouds 2/10; temperature 15C
20240513.log ----------------------------------------------------------------------------
```

In this example notice that the header

```
prediction: 2024 05 13   12 35 16    comb magn: 11.7   flash length: 5    interval: 60
date         UTC   secs to pred    latitude  longitude  alt(MSL)  flash/cal   sats  LSC
```

appears twice; this is because the flasher was restarted.

yyyymmdd.log displays the log file contents for the entered date.

```
20240511.log ------------------------------------------------------------------------------

prediction: 2024 05 11   12 30 55    comb magn: 10.6   flash length: 5    interval: 60
date         UTC   secs to pred    latitude  longitude  alt(MSL)  flash/cal   sats  LSC
2024 05 11   12 20 00       655    40.68686  -88.39640   233m     10.6/12.0    17    18
2024 05 11   12 21 00       595    40.68686  -88.39638   234m     10.6/12.0    17    18
2024 05 11   12 22 00       535    40.68685  -88.39639   234m     10.6/12.0    16    18
2024 05 11   12 23 00       475    40.68685  -88.39641   235m     10.6/12.0    17    18
2024 05 11   12 24 00       415    40.68689  -88.39641   235m     10.6/12.0    17    18
2024 05 11   12 25 00       355    40.68689  -88.39640   234m     10.6/12.0    14    18
2024 05 11   12 26 00       295    40.68689  -88.39638   234m     10.6/12.0    15    18
2024 05 11   12 27 00       235    40.68691  -88.39643   233m     10.6/12.0    15    18
2024 05 11   12 28 00       175    40.68690  -88.39644   234m     10.6/12.0    15    18
2024 05 11   12 29 00       115    40.68689  -88.39641   234m     10.6/12.0    17    18
2024 05 11   12 30 20        35    40.68689  -88.39639   234m     10.6/12.0    15    18
2024 05 11   12 31 25       -30    40.68688  -88.39639   233m     10.6/12.0    16    18
20240511.log ------------------------------------------------------------------------------
```

If you do multiple occultations in one day they will be logged to the same file but will be clearly distinguishable by their headers and flash times.

d list the **directory** of log files. Note that these can be in a haphazard order.

```
Log files(unsorted) -----------------------------------------------------------------------
20240429.LOG  20240513.LOG  20240501.LOG  20240511.LOG  20280229.LOG
Log files(unsorted) -----------------------------------------------------------------------
```

As mentioned previously, entering a nine-value string like this writes a prediction to input.txt
2024 5 13 12 35 16 11.6 5 60
Predictions may be up to several years in the future.

i displays the content of the **input.txt** file.

```
input.txt ---------------------------------------------------------------------------------
2024 5 13  12 35 16  11.6  5  60
input.txt ---------------------------------------------------------------------------------
```

e lets you **empty** the input.txt file (**y** to confirm).

```
Confirm you want to empty input.txt?  y/n
12 26 25         496        561    11.7/12.0/42      16    input.txt emptied
```

remove yyyymmdd.log lets you delete the log file for that date (**y** to confirm).

```
Confirm you want to remove 20240428.log from the SD card?  y/n
```

```
12 26 52          469         534     11.7/12.0/42       16     20240428.log removed
```

remove can also delete other files from the SD card. There are two special files:
- output.log is a default file that will log manual flash and ZSF information when there is no specific prediction.
- saved.txt contains some flash settings in between startups.
Either of these may be removed if necessary; they will be recreated when needed.

Entering a magnitude-sized number like **11.2** immediately sets the LED brightness to that value. Remember it's a magnitude so every ~0.75 doubles/halves.

```
12 28 42          359         424     11.2/12.0/67       16     LED magn change
```

In the above example, 12.0 is the calibration magnitude and 67 is the linear LED flux.

When the LED is always-on or blinking you may adjust the brightness in real time as you inspect the capture software preview images or look at its brightness histogram.

```
12 48 22                              11.0/11.0/32       12     LED on
12 48 33                              11.5/11.0/20       13     LED on
12 48 41                              11.8/11.0/15       13     LED on
12 50 44                              11.8/11.0/15       14     flash logged
```

During blink and always-on the LED level changes are displayed as above but will not be logged. If you want to log an LED level, just make a **[f]**lash; this will also turn off the LED.

If you enter a magnitude that is close to the extremes of the brightness range, the warning "range caution" will appear in the notes.

z switches **zero**-second flashes behavior between three states:
- zero-second flashes on and logged – this is the default when there is an active prediction.
- zero-second flashes on but not logged – select this if you don't want to clutter the log file.
- zero-second flashes off – this is the default when there is no active prediction.

ZSFs are turned off 1 minute before the first goalpost flash.

```
12 46 53                              11.0/11.0/32       12     zero-sec flash on, logged
12 46 57                              11.0/11.0/32       11     zero-sec flash on, not logged
12 46 59                              11.0/11.0/32       11     zero-sec flash off
```

r restarts the flasher and tells it to look for the next prediction. A restart takes about one second.

Calibrating the LED level

Calibration centers the LED control range on a given stellar magnitude.

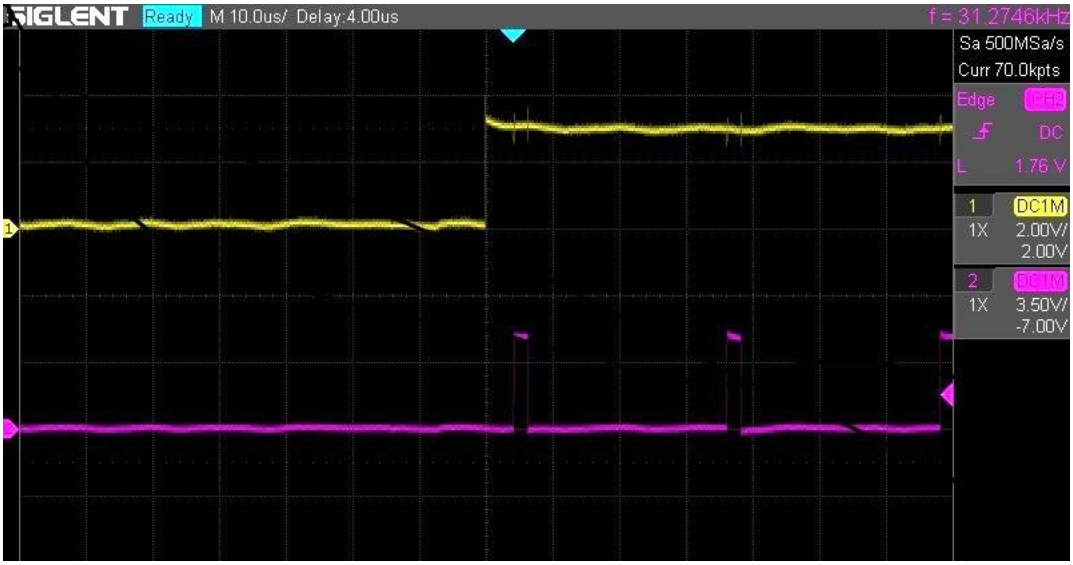
When we record an occultation, we adjust the camera exposure – frame rate and gain – so that the target star can be measured accurately, and we do this by inspecting the capture program's preview and histogram displays to see that the star's image is not too dark but also not saturated.

The flash must also be properly exposed but because the camera settings are already fixed for the target star, we instead adjust the flash LED's own brightness to make its exposure correct. In this flasher the brightness is set in two steps. In the first calibration step the potentiometer sets the LED current and its intrinsic brightness. This adjustment is made while imaging a star of known magnitude, adjusting the camera exposure, and then adjusting the potentiometer so that the background LED light is also properly exposed.

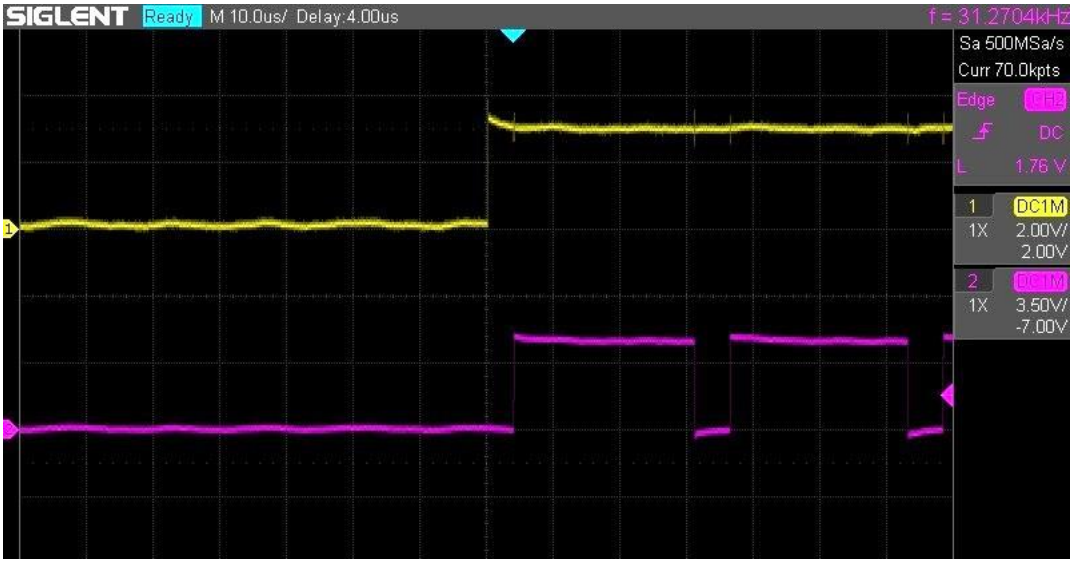
After calibration, a pulse width modulation (PWM) control will further adjust the LED brightness, either automatically for an input prediction star + asteroid combined magnitude, or via a manually entered value.

PWM operates by pulsing the LED at a high frequency (31kHz), always at its calibration current and intrinsic brightness but adjusting the duty cycle to put more or less power into each pulse.

Below is an oscilloscope trace of the flasher output (purple) at a low duty cycle:



and here with a high duty cycle:



The upper yellow trace is the start of the PPS signal from the GPS receiver. At the moderate frame rates used for occultation recordings the 31kHz pulses appear continuous to the camera, so the LED flashes vary only in brightness as the duty cycle is changed.

The overall PWM range is 1-511, i.e. the LED can be "on" from 1/511 to 511/511 of the time, a range of about 6.75 magnitudes. The lower end is limited to 1 instead of 0 so that the pulses (and the flash made from the pulses) won't disappear for dim stars; the upper limit is the highest value of the counter that is used to set the PWM duty cycle.

Instead of the raw PWM range 1-511, you will adjust the LED brightness by inputting its magnitude. It is useful to think of the LED level as a magnitude, similar to that of a star. They won't be exactly equivalent because the star image is a point while the LED illuminates the entire image background. But when star and LED use the same brightness scale (a smaller number is brighter; unit magnitudes differ by a factor of 2.512) it's less confusing, and it is very useful to know that when a star of magnitude 12.7 is well exposed, an LED background flash of "magnitude" 12.7 will also be well exposed, and when you change your target (and exposure) to an 11.2 magnitude star, setting the LED magnitude to 11.2 will also show the flash properly exposed.

At the calibration magnitude the PWM duty cycle is set to 32/511, which is roughly the middle of the magnitude (not linear) range. The effective magnitude range is displayed in the prediction panel.

Potentiometer adjustment

To calibrate the LED:

- Point the telescope at a star whose magnitude is known and similar to that of the occultations you normally observe.

- Adjust your camera frame rate and gain so this star is well exposed, not too dim and not overly bright.

- Make the LED **always-on** or **blink.**

- Tell the flasher you want to **calibrate**, and set the calibration magnitude to be the same value as the star's magnitude. This step sets the LED magnitude to the same value and also sets the PWM duty cycle to 32, the middle of its magnitude range.

- While you inspect the capture software preview screen and histogram, adjust the potentiometer so that the LED also becomes well exposed, not too dim and not overly bright. Adjust only the potentiometer. Don't use the camera controls for this. Also don't use a keyboard input to set the flash magnitude. Setting the potentiometer completes calibration.

The LED level is now the calibration magnitude, in the center of the overall ~6.75 magnitude PWM range. The PWM range is the amount that can be auto-set (i.e. the star + asteroid combined magnitude) or that you can manually input via the keyboard, a little over three magnitudes on either side of the calibration level.

The calibration magnitude is saved to a file named saved.txt on the SD card, and for subsequent occultations—as long as you don't readjust the potentiometer—the predicted star + asteroid combined magnitude from input.txt will become the LED magnitude, which should produce a well-exposed flash.

After calibration the LED brightness will be like any other magnitude, e.g. if you calibrate at magnitude 12.2 and then use the keyboard to set the flasher to magnitude 11.2, the LED will become one magnitude or ~2.5 times brighter.

You can test the calibration by manually setting the LED level to the magnitudes of a few other stars to see how the PWM control will work. Point to each star, set the LED magnitude to the star's magnitude (just type in the value), adjust the camera so that the star is well exposed, tell the flasher to blink, and then see if the blinking flashes are also well exposed. There will be some variation – the camera will have different sensitivity to different colors, and atmospheric extinction may affect the imaged brightness – but whenever a star is imaged and recorded okay the LED exposure should also look reasonably okay.

You can perform the calibration step as often as you like. If you tend to do occultations within a limited magnitude range perhaps you'll need to calibrate only rarely. Or if you have variable skies you may want to calibrate every evening before doing a series of automated recordings.

In actual use you may wish to readjust the LED level even after it has been auto-set to the combined star + asteroid magnitude. This is easy – just type in another a higher or lower magnitude – and it's normal because of color and atmospheric variation between stars. PyOTE is quite tolerant so you don't have to be extremely careful about this. But if you find that your readjustments are often biased in one direction, either more or less bright, you can tweak the potentiometer a small amount to remove the bias.

Note that if you set the LED level to a bright magnitude so that the flux reaches its maximum 511 value, the LED will be constantly on during the flash and will no longer be PWM-pulsed; only the potentiometer adjusts the brightness level.

Logging

The timed goalpost flashes are always logged. Zero-second flashes are logged by default but their logging may be turned off/on as needed, using the command input **z**.

Log file contents can be displayed in Termite at any time (command input **l** or **yyyymmdd.log**) and can be easily copied & pasted, so you won't need to remove the SD card to see a log file.

When there is no SD card

If the SD card is not readable, the prediction panel displayed after a restart will say "input.txt NOT FOUND" and "NO LOG FILE", and the countdown lines will show magnitudes like: "0.0/0.0/32"

Flashes can be made manually. Here is a quick emergency procedure:

[r]estart the flasher to see if the SD card can be read. You can try the physical reset button on the flasher too.

If that doesn't work, do this:

- set a **[c]**alibration magnitude – if you remember what the last calibration magnitude was, then use that; otherwise just use the occultation target star magnitude.
- use your camera software to adjust the frame rate and gain for the target star.
- turn on **[b]**link.
- set the flash level by directly entering the target star's magnitude.
- see how the blinks look in the capture software, and adjust by entering higher or lower magnitudes via the keyboard. If the flasher magnitude range reaches a limit, adjust the potentiometer.
- turn **[b]**link off.
- make manual **[f]**lashes when you like. The default flash duration is 5 seconds, so be careful not to make a flash too close to your expected occultation time.
- write down, or copy & paste, the times that are displayed.

Zero-second flashes can also be turned on and off, but be aware that they will not turn off automatically so be careful when you get close to the occultation event times.

Analysis

(thanks to Bob Jones for help with the wording in this section, and thanks to Michael Camilleri for permission to link his papers on flash timing and analysis)

Flashed occultations are best analyzed using PyMovie and PyOTE because they have special tools to extract flash times and to measure the intervals between flashes, frame edges and the occultation disappearance and reappearance. PyMovie and PyOTE are feature-rich and won't be discussed in detail here, but the following should help you with the flash-timer parts of the data.

PyMovie

In PyMovie you place measurement apertures by clicking at a location in the image and then selecting the aperture type. For timing flashes specifically:

- at an unoccupied background area (read cautions below), right-click and select "Add static aperture (no snap)" and name it something like "flash"

- right-click on the new aperture and select "Turn white (special 'flash tag' aperture)"

This will add a light curve for the flashes to the .csv file, from which PyOTE can extract times.

Here are the cautions:

If the imaging camera has a global shutter the flash aperture may be placed anywhere, but if the camera has a rolling shutter, different lines on the image happen at different times, so put the flash aperture on the same horizontal line as the target star, or as near as you can. See further cautions in the analysis section below.

PyMovie apertures automatically track stars to accommodate drift, wind and periodic error but the white flash aperture necessarily remains stationary – there is no star to track – so take care to avoid an area where drifting stars might wander into the white aperture.

PyOTE

In PyOTE you will identify where in the PyMovie light curve the flashes start and enter their times. From these PyOTE will extrapolate the times of frame edges, and ultimately the occultation D & R times. Flash times are measured first, then the light curve table populated with frame times, and then you can measure the target star D & R times. Here are details for the flashes:

- before you start PyOTE, have the following ready:

    - from your flasher log, the times of your two goalpost flashes, in hh, mm, ss.

    - from your camera software, the frame rate of your recording, in seconds per frame, e.g. 0.100, not 10 frames/sec.

- open the PyMovie .csv file with "Read light curve".

- click on the "Lightcurves" tab and for the [name you used for the PyMovie flash aperture] curve check the "target" box. This will display and let you measure the flash aperture light curve.

- go to the "Manual Timestamps" tab.

- on the light curve panel select data points on both sides of the rise of the first goalpost flash, and then click “Calc flash edge”. This measure where in the frame the first flash began.

- do the same to locate the rise of the second goalpost flash: select points on either side of the rising edge, and then click “Calc flash edge”.

- click on “Manual timestamp entry”. A table shows the already filled-in frame numbers for the rising edges of the two goalpost flashes, as just calculated. Notice that the frame numbers have a decimal fraction that shows where in the frame each flash started.

- from the flasher log data, into the table enter the hour, minute and seconds values for the flashes you just measured. You don’t have to enter zeroes after the seconds decimal point.

- specify the frame rate. If you are using analog video, select NTSC or PAL. For a digital camera, check the “Custom frame time” box and enter the seconds per frame value of the recording.

- click OK. If the error in the information window shows 0.0%, then the frame count between flashes is okay and all the frames will become timestamped with GPS times.

- click on the “SqWave model” tab.

- click on “Save current light curve to .csv” and give your new copy a name. Include something like “..._GPS_flash” in the name to identify how the frame times were estimated.

Now proceed to analyze the occultation light curve D & R times. When you submit your report, include the new .csv file that has the GPS times and tell the reviewer that you used a GPS-based flash timer.

For rolling shutter cameras time analysis becomes tricky if star apertures drift away from the stationary flash aperture line but it’s still possible to extract correct the flash and occultation times. Michael Camilleri has studied flash timing extensively and has kindly given permission to link this informative Journal of Occultation Astronomy article and a detailed paper that explain how to extract accurate times from rolling shutter cameras.

Troubleshooting

No output; GPS red light flashes continuously
An ongoing flash means there is no GPS lock (normal lock is one red flash every 15 seconds).
- Is a CR1220 backup battery installed? (+ side is up). Without an adequate battery the flasher may take a long time to download a satellite almanac and be able to read data.

- Signal strength is weak or intermittent. Although the receiver's own antenna is usually better than an external antenna, if the flasher is in a metal building or near electrically noisy equipment an external antenna may be necessary.

No output; GPS flashes normally every 15 seconds; prediction panel shows no prediction
- there is no SD card, or no **i**nput file, or no prediction in the input file.
- the prediction is not in the future, or some other input data value is invalid.
- check wiring between GPS shield and Mega pins 18, 19 and 20.

Output to screen is okay but no flashes
- check wiring between output pin 5, potentiometer and LED.
- is potentiometer set to very low brightness?
- was keyboard entry used to set a very dim magnitude?

There are no zero-second flashes
- there is no active, future prediction
- ZSFs are turned off

Output is garbled.
- Make sure the flasher's 115200 bps baud rate is matched by Termite.
- Make sure the GPS shield's switch is set to "Soft Serial".

2024 September 27