Flat Maestro LED light panel

Chris Woodhouse FRAS 2023. V2

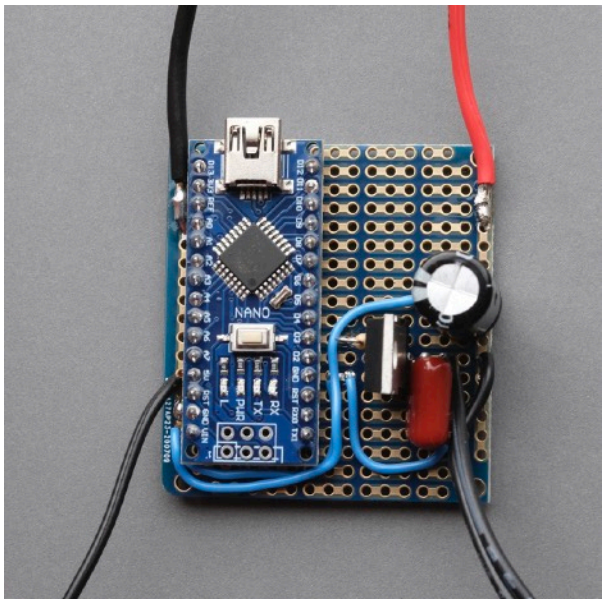
This project is an ideal entry point to making your own hardware and either using the supplied software, or modify for your own requirements. I used the free Arduino and Visual Studio versions for non-commercial purposes but bought the Visual Micro extension, for convenience.

The hardware consists of:

A2 light panel: There are many panels available, many with touch button controls on the panel itself. These are not ideal, and I chose a model that turns on with a simple 12 V power source. This model has an external in-line controller, which is optionally inserted in-between its 12 V power supply and the panel. (**Choose a panel to run off 12 volts DC, not mains).** I discarded the controller and the plug-in-the-wall power supply and clipped off the leads off the controller to re-purpose. The A2 panel covers a pair of scopes side-by-side. A3 may suffice for a smaller optical setup. For convenience, I inserted the panel into a spare picture frame and hung on the observatory wall. For this I trimmed down an old aluminum Nielsen frame, which I happened to have spare.

This A2 panel consumes 900mA at 12Volts. That is quite bright, and I found I needed to reduce the brightness by 4x to make it more usable. In my case, I required an in-line 10 ohm 2W resistor to reduce the panel current.

Arduino: I used a NANO, but it is possible to use other models. The thing to watch out for is the ability to set alternative PWM frequencies: The standard NANO PWM output is at ±500Hz. This is not fast enough to prevent severe banding with short (<1 second) exposures on CMOS cameras. There are several free Arduino libraries that support higher PWM frequencies. I chose AVR.PWM which has a minimum frequency of 65KHz. The library frequencies and the pins they control vary by Arduino model and, if you choose a different Arduino model, you may need to change the connections and pin assignments in the Arduino code.

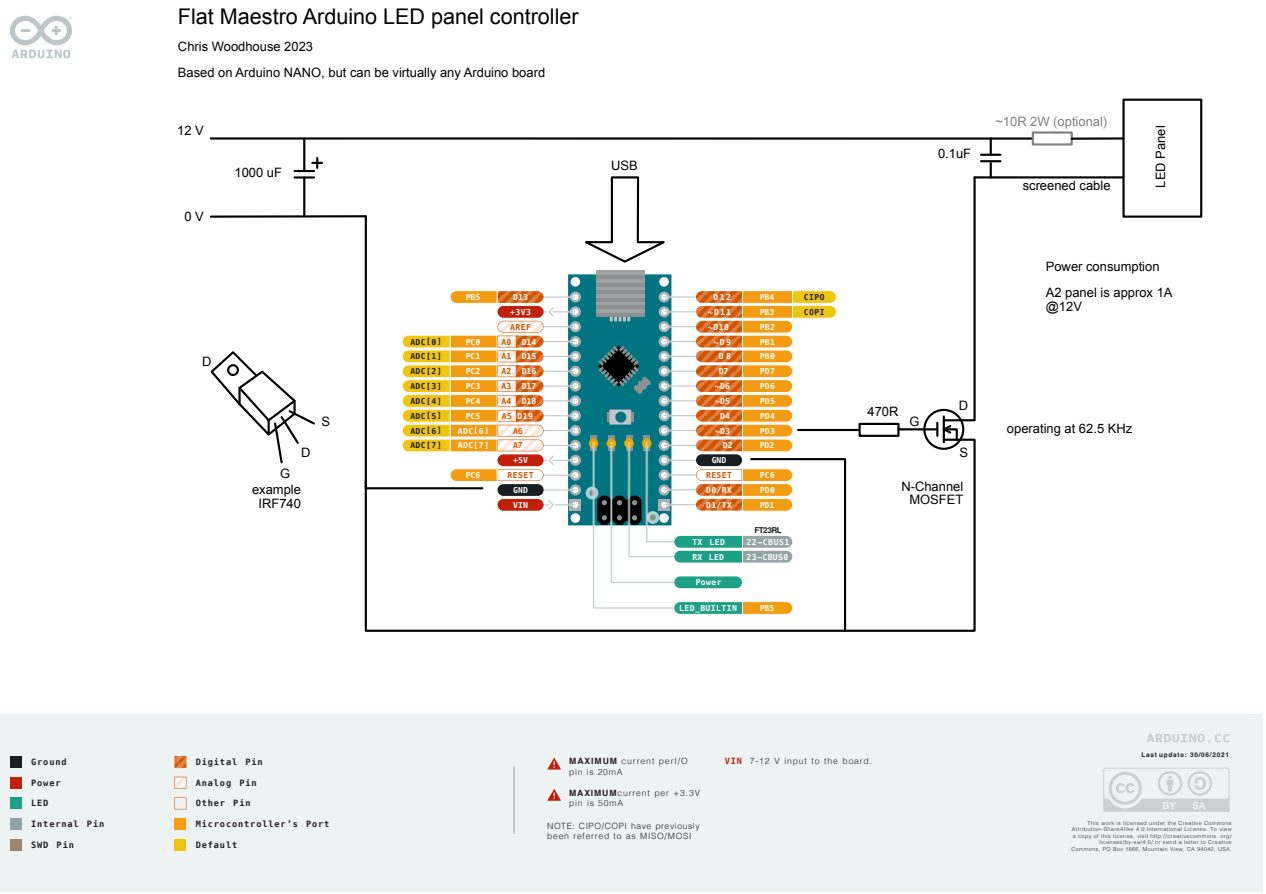


The NANO is powered through its USB port. The power to the panel comes from the 12 volt supply and switched through a power transistor to ground. I soldered my NANO to a small breadboard, but, with so few connections, it is possible, with care, to do direct soldering. In the final assembly, I insulated

exposed terminals to avoid shorts as well the heatsink of the transistor as it is connected to the drain and is a potential short risk.

Power Transistor: I used an N-channel MOSFET. I chose an IRF740. I did not require a heatsink. Other models will suffice. It is switching 1 A maximum (0.28 with the 10R resistor) on this A2 panel, and is over-specified. The gate of the transistor is connected to the NANO with a 470R resistor. The gate capacitance is quite high (~1nF) and this forms an RC network to reduce high frequency components.

Capacitors: I put an electrolytic on the 12 V line and a 100nF on the output to the LED panel, for good practice.



Software: The software comprises two programs; the Arduino code and the ASCOM driver. Both are published on GitHub and are kept up to date. I used Visual Studio for coding with the Visual Micro plugin for the Arduino. One could also use the free Arduino IDE too for its code. In both cases, the Arduino code is uploaded through its USB, in boot loader mode.

Feel free to copy the code and try out your own variations. The ASCOM installer script and the installer .exe file are in GitHub too, or you can load them into Visual Studio, or similar, and change things.

https://github.com/chris125577/FlatMaestroASCOM
https://github.com/chris125577/FlatMaestroArduino

If you want to do your own editing, you will need to load the ASCOM extensions to Visual Studio and the ASCOM developer components and INNO setup to give access and use the installer script generator. The ASCOM driver passes ASCOM ConformU with flying colors. More details on the process are on the ascom-standards.org website.

The ASCOM device covers flat panels and covers. Here, we are just using the few light panel commands. It works by sending a simple numeric string to the Arduino to set the brightness level (0-100). 0 turns it off entirely. The Arduino unit replies with its brightness setting as a string. When the Arduino unit is powered down, it forgets the light level. If you want it to remember at power up, a single line in the Arduino code requires un-commenting.

Enclosure: In my observatory, the tiny Arduino is secured to inside my master interface box with Velcro. One could also do a soap-on-a-rope construction and have a small plastic inline box with the LED cable coming out one side and the USB cable on the other.

In Use:
I experimented with worse-case flat exposures with Luminance and SII filters. I struggled with the initial hardware as my Lum filter exposures were less than 0.1 seconds, even at the lowest brightness setting. Inserting the in-line 10 ohm 2 W resistor reduced the output by 4x, and it then achieved sensible exposure times with a lowish panel brightness, with a peak current of 280 mA. The PWM output waveform never achieves the full 0–100%, but is more like 4–96%. The software scales a 1–100% command to 4–96 %. For a zero input, it disables the output entirely, to turn the panel off. I recommend using brightness values in the range 5-90%. The trick is to use the flat panel wizard and restrict oneself to a handful of exposure durations, which permits a small library of master darks for flat calibration. In my case, I selected 0.2, 0.3, and 2 seconds to cover both cameras, at multiple gain settings, and with LRGBHSO filters.

I have my panel mounted opposite my telescopes when in the park position. This means I can do flat frame exposures with the roof closed. In NINA, for example, I can now either set the panel to an average brightness and vary the exposure length to obtain the right flat level, or do dynamic brightness, with a fixed exposure time. The latter is useful if you are additionally having to do flat darks, as it permits a few select exposure times to have a library of ready master flat darks for calibration and speeds up the FlatWizard execution (and means you do not have to continually cover/uncover the scopes).

The latest TargetScheduler plugin supports automatic flats, using one of two modes; immediate and completion and for multiple imaging (server/client) arrangements. There are two new instructions; Target Scheduler Immediate Flats, in which flat exposures are taken at a certain cadence, immediately after light exposures, to ensure timely and aligned exposures. The second, Target Scheduler Flats, only takes flats upon target completion. This is more suitable for sealed optical systems. I put the Target Scheduler Flats instruction in my shutdown sequence, just after parking the mount and closing the roof. The TS plugin currently does not support sky flats.

Clear skies