Cyclops is a high-power LED driver that enables precise control of light power for optogenetic stimulation. The circuit was developed by Jon Newman while in Steve Potter's lab at Georgia Tech in order to complete his thesis work, which required the delivery of ultra-precise, continuously time-varying light waveforms for optogenetic stimulation [1,2]. This was, and still is, not possible with commercial hardware for optogenetic stimulation. Since its first use, the circuit has been improved in terms of speed, precision, programmability, and ease of use. This document provides construction, usage, and performance documentation for the Cyclops LED driver. This document evolves with the repository. To view old revisions, checkout tags or old commits using their SHA.

Note Github does not render alt text specified in Markdown figures as captions. Therefore, if you are viewing this document on Github, you will need to hover over figures to see their captions.

Maintainer

- jonnew

Table of Contents

Note From the Maintainer

It has been a long road to design and test the Cyclops, coordinate the acquisition of materials, coordinate the manufacturing processes, and to distribute the device to the community. This process has been a lot of work but also a very rewarding learning experience. I am very happy that this device may enable your scientific endeavors and I'm sincerely grateful for your interest in the project. In general, I hope this project will eventually be one small module in a of growing set of **high-quality**, **open-source**, and **afforable** tools that facilitate your research and enable an **open**, **community-oriented** approach to science.

Profits from the sale of Cyclops kits go to funding the Open Ephys non-profit organization. Since I receive no monetary compensation from the sale of these devices, it would mean a great deal to me if you would consider referencing the following paper (for which the Cyclops was developed) in published work that makes use of the device.

> J.P. Newman, M.-f. Fong, D.C. Millard, C.J. Whitmire, G.B. Stanley, S.M. Potter. S.M. Potter. Optogenetic feedback control of neural activity. *eLife* (4:e07192) 2015. doi: 10.7554/eLife.07192 [link]

For instance, in your methods section:

> Optical stimuli were delivered using the open-source Cyclops LED driver (Newman et al., 2015; www.github.com/jonnew/Cyclops).

Pull requests and issue submissions are **welcome** on the github repo and open ephys forum. If you have criticisms, fixes, suggestions for improvement in the docs etc, please let us know so we can implement them ASAP.

Happy stimulating.

Jon Newman MWL@MIT 2017-03

Features

Circuit Features

- Ultra-precise
- High power
- Up to 1.5A per LED
- Wide bandwidth
  - ~2.5 MHz -3 dB bandwidth
  - Maximum 100 ns 1.0A rise and fall times
- Current and optical feedback modes
- Built-in waveform generation
- Over-current protection
- Modular
  - Arduino compatible
  - Accepts external analog, gate, or trigger inputs

Multiple stimulus generation options

- External stimulus sequencer
- External digital trigger
- TTL logic level
- External analog waveform generator
- 0-5V analog signals
- Internal 12-bit DAC
- Synchronized across up to 4 drivers
- Arduino library
- Programmable triggering logic
- Respond to USB input

Performance Specifications

The following oscilloscope traces give indicates of the circuit's precision and speed (Rev. 3.5c) . Note that time series traces are **not** averaged - these traces display per-pulse temporal characteristics. Optical characteristics and optical feedback signal for the Cyclops driver were provided by a Thorlabs PDA36 amplified photodiode set to 0 dB of transimpedance gain. Measurements were performed a single Osram golden dragon LED.

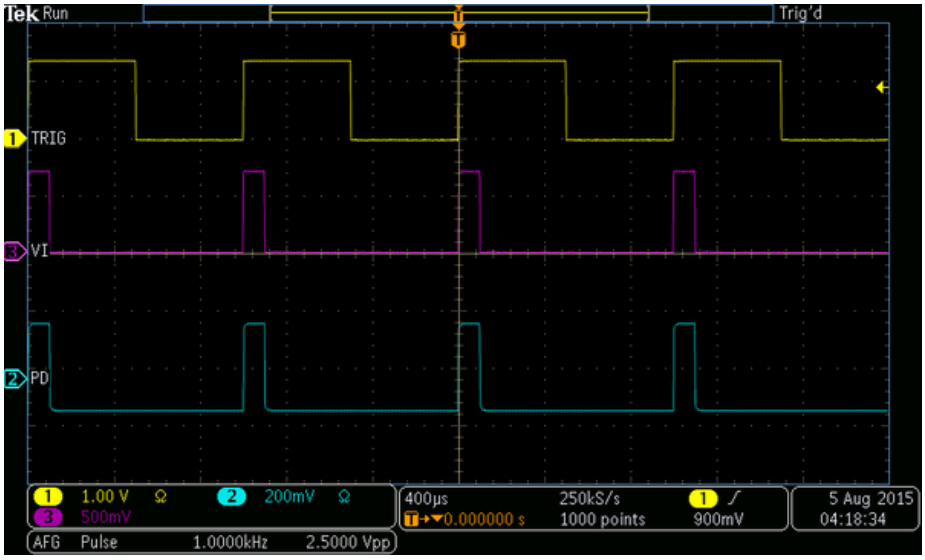


Figure 1: Trigger (yellow), current (pink), and light power (blue) traces during pulsed operation in current feedback mode. Input waveform is a 1 kHz 0 to 750 mV, 10% duty cycle square wave.

The following traces are the same as the previous ones except that the amplified photodiode was used to provide optical feedback. The slowdown compared to current feedback is due to a speed of the photodiode. A faster amplified photodiode would provide crisper rise and fall times

The current-feedback mode -3dB bandwidth was determined by applying a flat noise signal over 50 MHz with mean = 1.0V and Vpp = 500 mV into the `EXT` port with maximal current gain. It occurs at around 2.5 MHz.

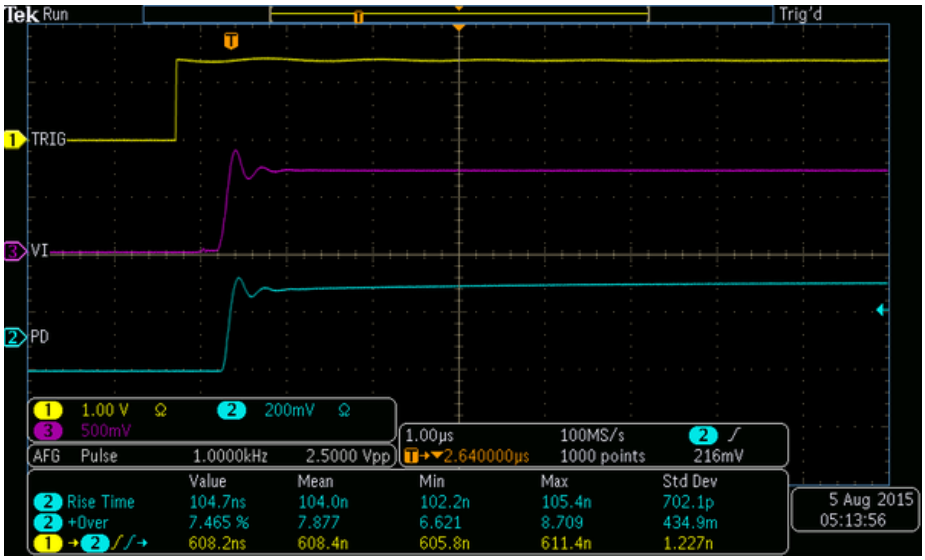


Figure 2: Zoomed traces showing waveform 10-90% rise times. Optical rise time statistics are shown at the bottom of the image.

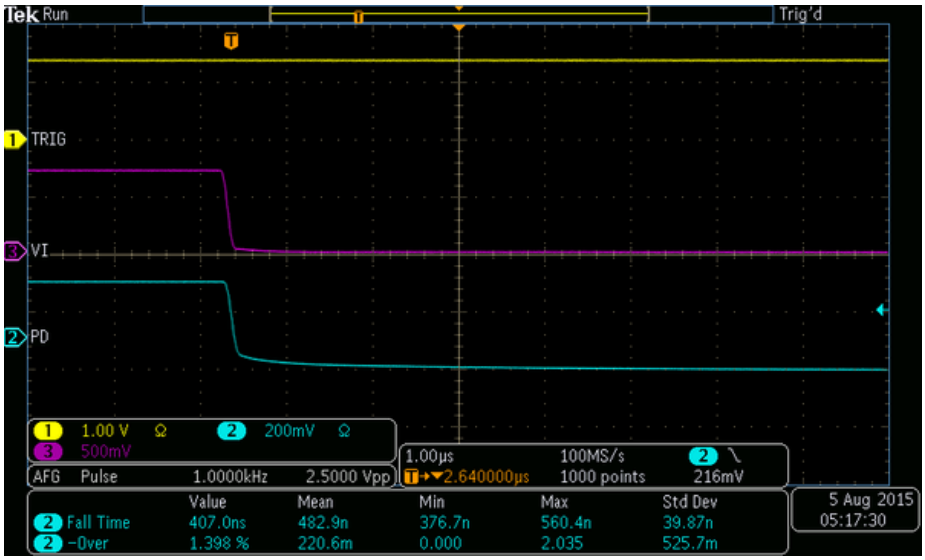


Figure 3: Zoomed traces showing waveform 10-90% fall times. Optical fall time statistics are shown at the bottom of the image.

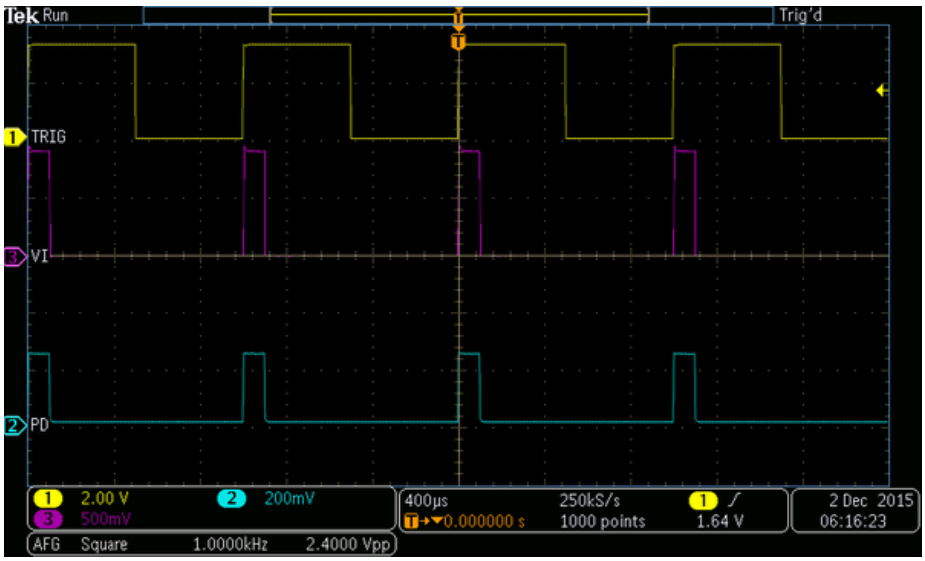


Figure 4: Trigger (yellow), current (pink), and light power (blue) traces during pulsed operation in optical feedback mode. Input waveform is a 1 kHz 0 to 750 mV, 10% duty cycle square wave.

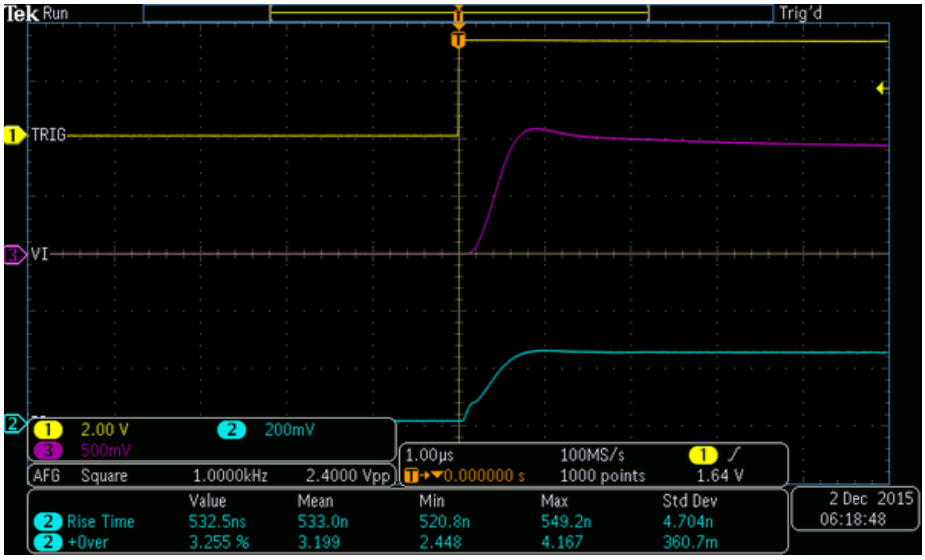


Figure 5: Zoomed traces showing waveform 10-90% rise times in optical feedback mode. Optical rise time statistics are shown at the bottom of the image.

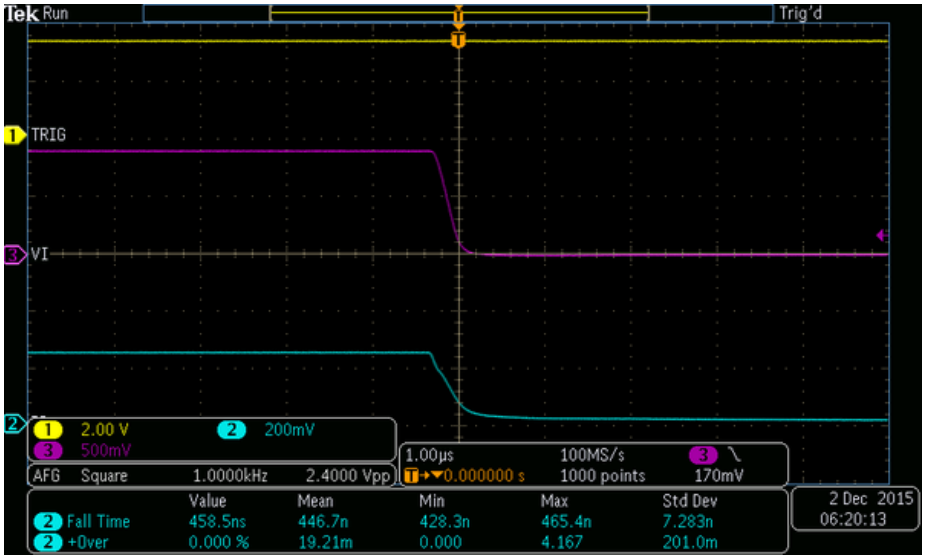


Figure 6: Zoomed traces showing waveform 10-90% fall times in optical feedback mode. Optical fall time statistics are shown at the bottom of the image.

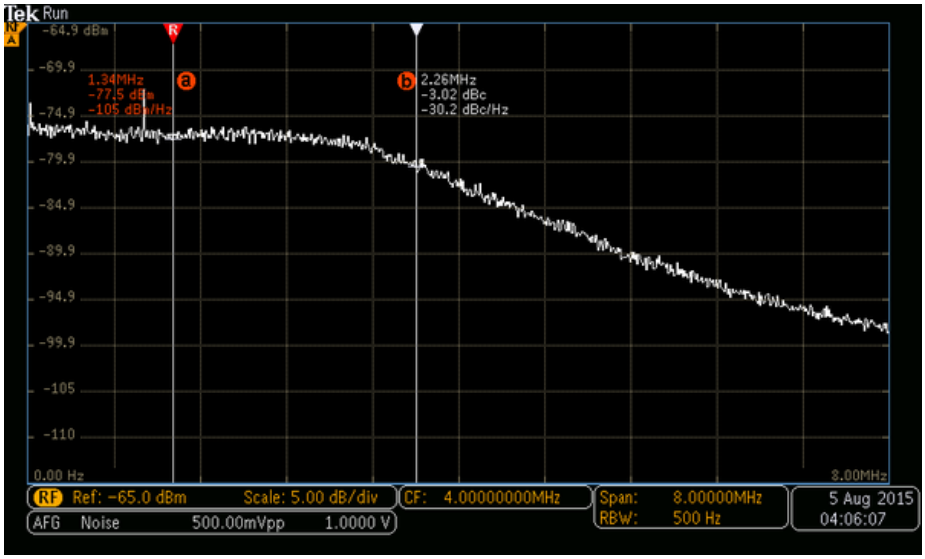


Figure 7: Optical bandwidth in current feedback mode. -3dBm occurs at ~2.4 MHz. Input signal was was noise, flat over 50 MHz, mean = 1.0V, Vpp = 500 mV.

Usage

Figure 8: Cyclops physical interface (Rev. 3.6).

The cyclops is a device that is capable of transforming voltage signals (e.g. sine waves, square pulses, etc.) into optical signals from high-power LEDs. Voltage signals to drive the device can be generated internally using an on-board digital to analog converter or can be delivered from an external source, such as a function generator or stimulus sequencer. The cyclops provides numerous measurements of circuit operation that can be recorded during an experiment such as LED current and stimulus reference voltages. The device can be controlled over a USB interface using its onboard, Arduino-compatible Teensy 3.2 microcontroller board in combination with the Cyclops Arduino library. The device also can

be configured to drive commercially available LED modules from Thorlabs and Doric using its expansion ports.

Below we provide an explanation of the operational modes of the device and the different ways it can be used to generate optical stimuli. Refer to the above diagram to locate the physical switches, dials, and connectors corresponding to verbal or iconic descriptions device settings.

Feedback modes

Current Feedback Mode

To use current feedback mode, push the `F.B. MODE` slide switch to the `CURR` position (AUX CURR). Using the circuit in current feedback mode ensures that the forward current across the LED is precisely regulated according the voltage at the `VREF` pin. This configuration is a standard method for driving LEDs because the relationship between current and LED irradiance is smooth and monotonic. This means that more current across the LED will generate more light power (while staying within the LED's maximum ratings, of course). However, the relationship between current and irradiance is not linear. For most LEDs, it looks like a logarithmic function. Additionally, the efficiency of the LED is inversely related to its temperature. So, as the LED operates and heats up, the amount of light it produces drops even when the current is held constant. The severity of an LED's temperature dependence and current/irradiance nonlinearity depend on the type of LED (roughly, the color and who made it). These properties should be clearly documented in the LED's data sheet. With a quality LED and proper thermal management, the effects of temperature and static current/irradiance nonlinearity are fairly minimal and can be ignored in most situations.

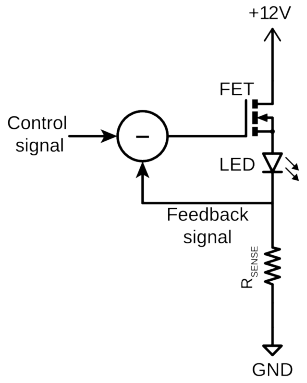
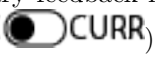


Figure 9: Current feedback configuration.

Auxiliary Feedback Mode

To use auxiliary feedback mode, push the `F.B. MODE` slide switch to the `AUX` position (

). When extremely stable, linear control of light power is required, the auxiliary feedback input can be used to used to compensate for the temperature dependence and static nonlinearity of the current/irradiance relationship of the LED. For example, when the auxiliary voltage is supplied by an amplified photodiode that is somewhere indecent to radiation from the LED, or is sampled from the fiber transporting LED light, the gate voltage is adjusted such that the measured light power matches a DAC-supplied reference voltage. This is the case in the circuit diagram. This configuration is referred to as optical feedback mode. The PDA36A adjustable amplified photodiode from Thorlabs is a good option for supplying optical feedback. However, you can make your own amplified photodiode for a fraction of the price, and a design is included within the cyclops repository. Optical feedback completely linearizes the relationship between a supplied reference voltage and the light power produced by the LED by compensating for the current/irradiance nonlinearities and temperature dependence.

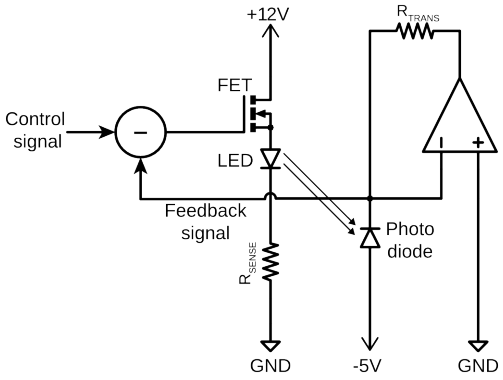


Figure 10: Optical feedback configuration.

Bandwidth Selection

The Cyclops can be operated in two bandwidth modes: `FULL` and `LIM`. This provides user-selectable control over the speed at which the LED can be turned on/off.

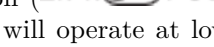
Full Bandwidth Mode

To operate the device in full bandwidth mode, move the `B.W. SELECT` switch to the `FULL` position (

). When the device is operated at full

bandwidth, the user can take advantage very short turn on/off times (1.0 A in about 100 ns rise/fall times). Additionally, continuously varying optical signals are accurately represented up to about 2 MHz. That said, Cyclops is a fairly high-power circuit. The currents and voltage used to drive high power LEDs are many orders of magnitude (like 6 or more...) greater than those recorded during electrophysiology experiments. Also, the Cyclops is a fast circuit. Fast circuits hate long cables because they introduce appreciable delays and parasitics that can adversely affect operating characteristics. Very long cables will introduce ringing into light waveforms with fast edges! Ideally, the LED should be right next to the device. I typically mount my fiber coupled LEDs directly into the banana sockets on the back of the device using copper-clad printed circuit board so that my 'cables' are about 2 cm in length. In any case, the following is allways good advice: **keep the cabling to the LED as short as possible and 'fat' enough to handle high currents (AWG 18 or thicker)**.

Limited Bandwidth Mode

To operate the device in full bandwidth mode, move the `B.W. SELECT` switch to the `LIM.` position (

). In this mode, the feedback circuit within the cyclops will operate at lower speeds. This is useful in situations where the use wishes to place the Cyclops device far away from the LED being driven (e.g. when it is mounted on the head of an animal rather than light being transmitted via optic fiber). In this case, the bandwidth of the device can be lowered to avoid ringing and instability when a long, potentially thin cable is used. I have driven LEDs at 1.5A over AWG 30 cable that is several meters long without issue in bandwidth limit mode. This would cause the device to oscillate wildy in full bandwidth operation. When the device is operated in bandwidth limit mode, the rise and fall times of LED pulse will increase to about 1.5 microseconds, and the -3 dB bandwidth will degrade to ~200 kHz or so. This is plenty fast for almost all neuroscience applications.

Stimulus Generation Options

There are three ways to generate light signals using the driver. The behavior of each of these options is dependent on the feedback mode being used. The behavior of each input option is described in relation to the feedback mode of the driver.

1. 

The test button is always available and will override all other input modes. Using the `TEST` button the behavior of the circuit is:

- 

Source the current specified by the `MAX CURR.` dial.

- AUX CURR Generate the optical power specified by the h * mW level that is specified by the `MAX POWER` dial. The intensity of the LED will be dependent on the auxiliary feedback signal used which defines the 'h' parameter.

2. EXT DAC External input mode is engaged when the `SOURCE` switch is moved to the `EXT` position and user supplied voltage waveforms are present at the `EXT` BNC input. If the user attempts to supply more than 5V to the `EXT` input, the circuit will clamp the input signal to 5V. Using `EXT` mode, the behavior of the circuit is:

- AUX CURR Source the current specified by (EXT Voltage / 5V) * MAX CURR.
- AUX CURR Generate the optical power specified by (EXT Voltage/5V) * h * mW. The intensity of the LED will be dependent on the auxiliary feedback signal used which defines the 'h' parameter.

3. EXT DAC The internal digital to analog converter (DAC) is engaged when the `SOURCE` switch is moved to the `DAC` position and can be used to generate pre-programmed waveforms and waveform sequences triggered by a digital pulse to the `TRIG` input. This feature relies on optional Arduino installation and programming the device using its API. Using the `DAC` mode, the behavior of the circuit is:

- AUX CURR Source the current specified by (DAC Voltage / 5V) * MAX CURR.
- AUX CURR Generate the optical power specified by (DAC Voltage/5V) * h * mW. The intensity of the LED will be dependent on the auxiliary feedback signal used which defines the 'h' parameter.

Using the onboard microcontroller

Cyclops devices include an onboard, Arudino-compatible microcontroller board (Teensy 3.2) that can:

- Generate custom waveforms
- Respond to trigger input
- Provide background over-current protection

The Teensy can be programmed and uploaded to the device using the Arduino IDE in combination with Cyclops Arduino library. *Note that you will need to add the Teensyduino add-on to to the Arduino IDE to program the Teensy.* The Cyclops library contains several examples to help you get started. The relevant public programming interface is shown here:

```
// Each 'channel' defines a board address. A single microcontroller supports up
// to 4 stacked Cyclops boards.
typedef enum
{
    CH0 = 0,
    CH1,
    CH2,
    CH3,
} Channel;

// One Cyclops object should be created for each LED channel. For single LED
// devices (most cases) only CH0 is used.
class Cyclops {

public:
    // Construct a Cyclops object
    //   chan:             Cyclops channel to control
    //   current_limit_mA: Optional current limit for this Cyclops channel
    Cyclops(Channel chan, float current_limit_mA = 1500);

    // Program the DAC output register
    //    voltage: 12-bit integer (0-4095) specify the voltage to program the
    //             DAC with. Voltage is scaled into a 0-5V range.
    //    returns: 0 on success, 1 otherwise.
    int dac_prog_voltage(const uint16_t voltage) const;

    // Load the DAC output register to affect a voltage change on the output
    // pin. This function affects all running cyclops devices that have had
    // their DAC's programmed using the dac_prog_voltage() function.
    static void dac_load(void);
```

```
    // Convenience method for programming and loading a DAC voltage. Equivalent
    // to calling dac_prog_voltage() followed by dac_load()
    //    voltage: 12-bit integer (0-4095) specify the voltage to program the
    //             DAC with. Voltage is scaled into a 0-5V range.
    //    returns: 0 on success, 1 otherwise.
    int dac_load_voltage(const uint16_t voltage) const;

    // Use the DAC to generate a period waveform
    //    voltage:          Pointer to array of 12-bit integers (0-4095)
    //                      specifying the voltage sequence to program the DAC
    //                      with. Output voltage is scaled into a 0-5V range.
    //    length:           Length of voltage sequence.
    //    sample_period_us: Sample period of voltage sequence in microseconds.
    void dac_generate_waveform(const uint16_t voltage[],
                               const uint16_t length,
                               const uint32_t sample_period_us) const;

    // Get an LED current measurement in milliamps
    //    returns: LED current measurement in milliamps.
    inline float measure_current(void) const;

    // Attach/detach user provided interrupt function to TRIG pin
    // mode indicates the pin trasition state at which the function is called,
    // RISING, FALLING or CHANGED
    //    user_func: User define function specifying action to take when the
    //               TRIG pin transtions according the the mode arguement. e.g.
    //               can be a function that generates a wavefomr using
    //               dac_load_voltage() or similar.
    //    mode:      Indicates the TRIG trasition state at which the user_func
    //               is called. Can be either RISING, FALLING, CHANGED, HIGH,
    //               or LOW. See AttachInterput() Arduino function for more
    //               detail.
    void set_trigger(void (*user_func)(void), const int mode);

    // Call this static method after creating all cyclops channels to start the
    // internal overcurent protection engine and waveform generation machinery.
    // This funciton is used to start all channels.
    static void begin(void);

    // *this's channel ID
    const Channel channel;
};
```

A simple example Arduino sketch, which uses the Cyclops to produce a sinusodal chirp, is shown here:

```
#include <Cyclops.h>

#define PI_CONST 3.14159265358979323846

// Parameters of the chirp waveform
#define CHIRP_TIME_MS 5000 // Length of chirp waveform in msec
#define FREQ_START 0.5f    // Start frequency in Hz
#define FREQ_END 30.0f     // End frequency in Hz

// Create a single cyclops object. CH0 corresponds to a physical board with
// jumper pads soldered so that OCO, CSO, TRIGO, and AO are used. Set the
// current limit to 1A on this channel.
Cyclops cyclops0(CH0, 1000);

// Chirp frequency ramp parameter
float beta = 1.0;

void setup()
{
    // Chirp parameter
    beta = (FREQ_END - FREQ_START) / (((float)CHIRP_TIME_MS) / 1000.0);

    // Start the device and zero out its DAC
    Cyclops::begin();
    cyclops0.dac_load_voltage(0);
}

void loop()
{
    // Calculate current chirp amplitude
    float now = ((float)(millis() % CHIRP_TIME_MS)) / 1000.0;
    float freq
        = 2.0 * PI_CONST * (FREQ_START * now + (beta / 2.0) * pow(now, 2));
    unsigned int voltage = (unsigned int)(4095.0 * (sin(freq) / 2.0 + 0.5));

    // Program the DAC and load the voltage
    cyclops0.dac_load_voltage(voltage);
}
```

Construction

If you have questions during device assembly, please direct them to the open-ephys forum so that others may benefit. Pull requests improving this documentation are welcome.

This guide provides instructions for assembling a Cyclops device For more details on performance specifications and usage, please have a look at the complete manual available in PDF or markdown form on the Github repository.

> https://github.com/jonnew/Cyclops/blob/master/MANUAL.md
> https://github.com/jonnew/Cyclops/blob/master/MANUAL.pdf

If you have any questions, do not hesitate to post them to the open-ephys mailing list:

> https://groups.google.com/forum/#!forum/open-ephys

Also, **pull requests and bug reports are welcome**. I would love your help in improving this device and its documentation!

> https://github.com/jonnew/cyclops/issues

Components

Fully assembled cyclops PCBs can be purchased from Circuit Hub

```
[Cyclops on Circuit
Hub](https://circuithub.com/projects/jonnew_/cyclops)
```

This also includes an up-to-date parts list for each PCB with optimized prices. *Note that these parts are for a single PCB without the enclosure, power supply, etc.*

Unpopulated Cyclops PCBs can be fabriated by uploading the gerber files to the PCB fabrication service of your choice. I have had success with:

- OSH Park - made in America, excellent quality. Minimum of 3 boards per order
- Seeed Studio - made in China, very good quality. Minimum of 5 boards per order.

The layer of each gerber file is identified by its file extension:

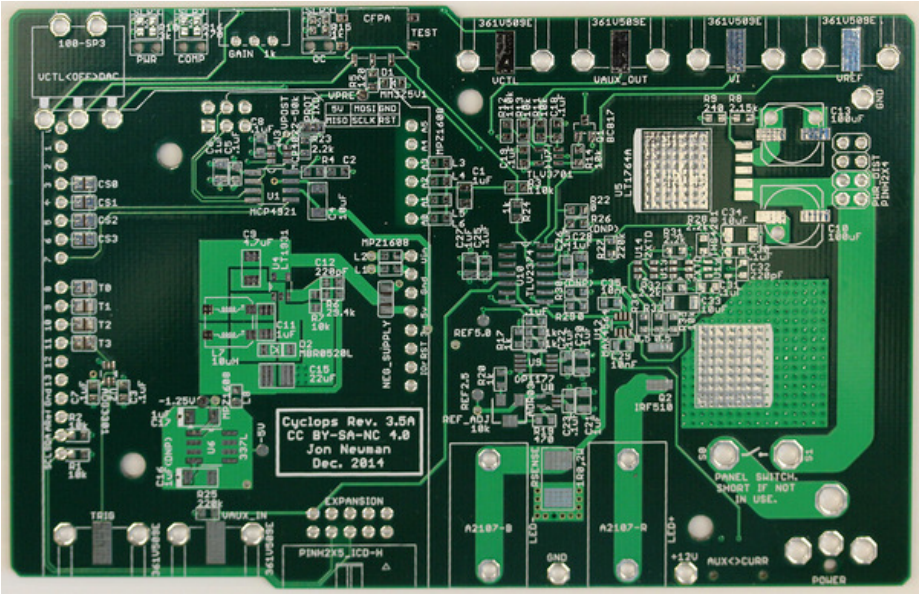


Figure 11: A bare Cyclops PCB, top side, fabricated by Seeed Studio.

```
*.GKO = board outline
*.GTS = top solder mask
*.GBS = bottom solder mask
*.GTO = top silk screen
*.GBO = bottom silk screen
*.GTL = top copper
*.G2L = inner layer 2 copper
*.G3L = inner layer 3 copper
*.GBL = bottom copper
*.XLN = drill hits and sizes
```

PCB stencils, which are useful for applying solder paste to the boards, can be purchased from a service like Seeed Studio using the gerber files located in ./cyclops/device/stencil/. If you plan to hand solder the board, or don't mind dispensing solder paste yourself, then you do not need to purchase these stencils.

A complete device requires several optional components, which are not in the pre-populated Digikey cart. These include:

- An **extruded aluminum enclosure**, which houses the completed board. The enclosure is recommended because the large voltages and current transients used to drive high power LEDs can cause capacitive and inductive interference with nearby recording equipment. Acrylic front and rear panels can be purchased from Ponoko using the links supplied in the BOM. The

instructions below show how these plastic pieces are modified to provide proper electrical shielding.
- An **M8-4 connector** (Optional). This is a rather expensive connector that allows cyclops to drive Thorlabs LED modules or Doric LED modules.

Board Assembly Manual

Board Assembly Materials

To assemble a Cyclops board, you will need the following materials

- A soldering iron and, if possible, a hot-air reflow device.
  - At minimum, a soldering iron regulated to ~370 deg. c) will do the job.
  - In addition to the iron, a hot-air rework tool or reflow oven are recommended and the assembly instructions below assume you are using one of these two options. A low cost, high-quality hot-air rework station can be purchased from SparkFun here.

Figure 12: A soldering iron can be used to assemble the PCB, but a hot air rework station makes things much easier. These can be purchased from Sparkfun.

- Copper braid ('solder wick') for solder removal (e.g this)
- Liquid flux (`no-clean` variants are easiest since they don't have to be thoroughly removed after use)

Figure 13: Wire solder and an soldering iron can be used to construct the PCB, but solder paste combined with a hot air rework station or a reflow oven makes things much easier. We use Chipquik 291ax10.

- Solder paste (e.g. this)
- Stereoscope or loupe (optional but nice for tracking down shorts.)
- Isopropyl alcohol for cleaning flux off the board (e.g. this; optional)
- An anti-static mat (e.g. this; optional but recommended to protect your work. . .)

Board Assembly Instructions

PCB component population and soldering is fairly straightforward and requires standard surface mount construction techniques.

- A tutorial on hot-air soldering can be found here.
- A great tutorial filled with general tips and tricks for surface mount soldering can be found here.

The following steps provide a visual guide to construct your own board. The goal is to create a fully populated PCB like this one:

Following board construction, you should run through the electrical tests outlined in the next section before applying power.

1. Place the bare PCB on a flat surface, preferably one that is static dissipative or anti-static. Alternatively, the board can be mounted in a PCB vice.
2. The silkscreen layer on the PCB (white text) has almost all the information you will need to properly populated the PCB. However, its a good ideal to to open the cyclops design in EAGLE. This will allow you to get detailed information on components before placing them on the board.

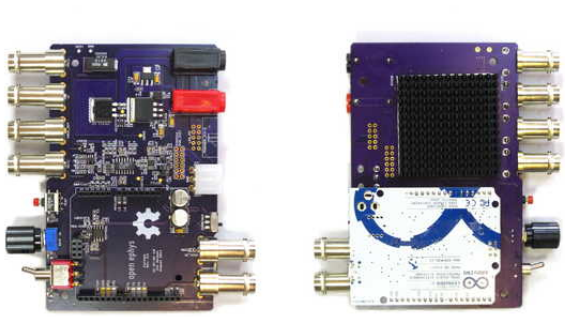


Figure 14: Finished device (revision 3.3).

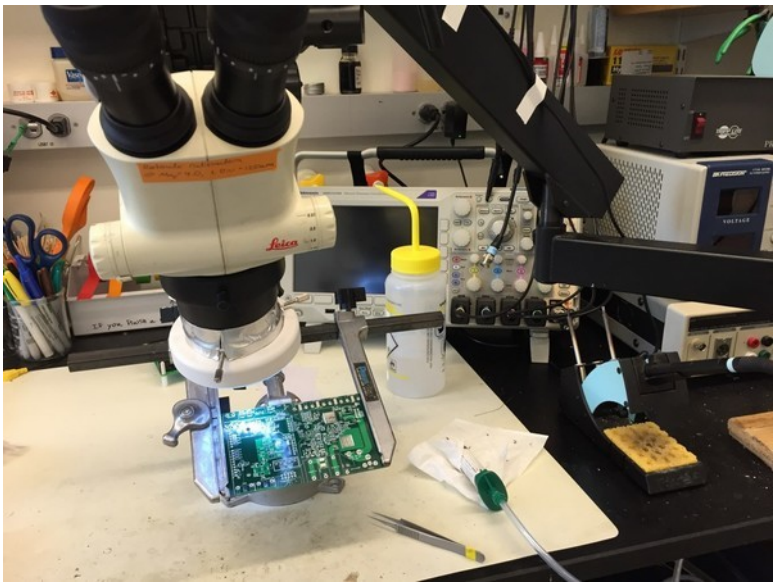


Figure 15: Instead of populating components on a table, holding the PCB using a PanaVise can be helpful.

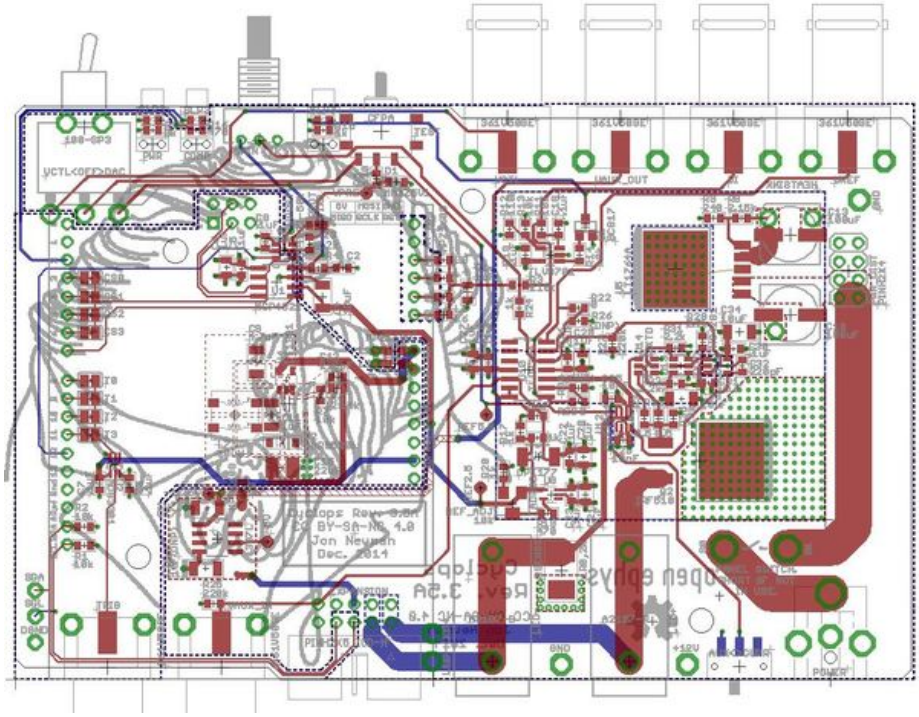


Figure 16: The cyclops PCB design in CadSoft EAGLE

You can then the `information` tool to get detailed information on each component, e.g. to ensure you are placing the correct value resistor or capacitor.

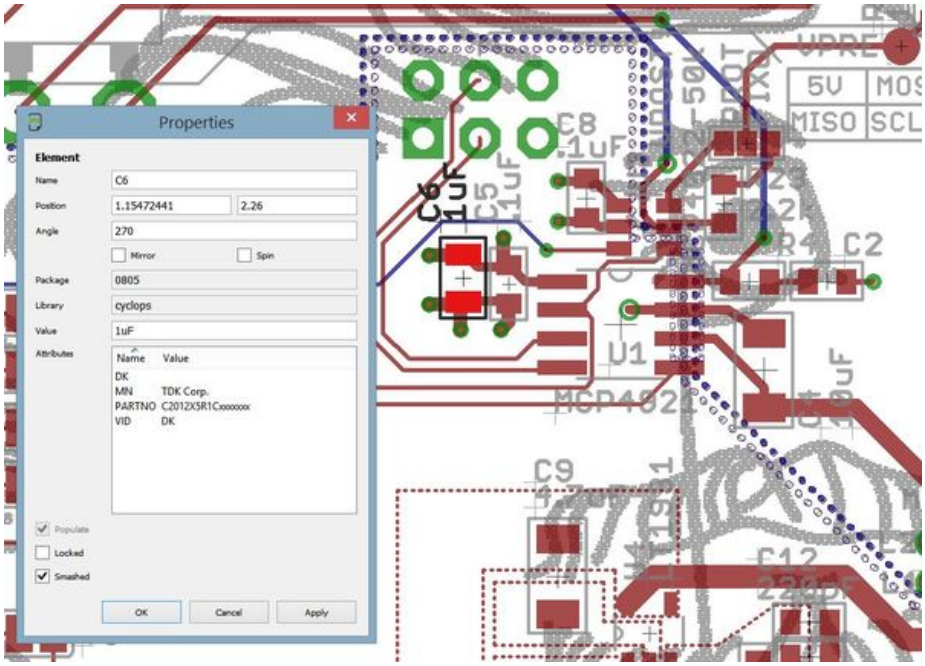


Figure 17: By selecting the information tool and clicking the cross at the center of a component, you can pull up detailed info (e.g. part number)

3. After cleaning the surface of the board with isopropyl acholhol or similar, apply solder paste to each of the pads. For an excellent series of tips on effective methods for dispensing solder paste, see Mike's video on the subject. Do not apply solder paste to through-holes or the pads shown outlined in red in the following image. These will be hand soldered later in the assembly process.

   The correct amount of solder paste to apply is 'enough'. Each component contact should sit in a small amount of paste, but blobs of paste that envelop the component pad or pin may later result in a short. The following images show examples of good and bad solder placement.

   If you need to pause at any point, you should store place the PCB in the fridge to prevent the flux in the solder paste from breaking down.

4. Populate all **top-side surface mount** components on the board. There is a single surface mount switch on the back of the board that will be hand soldered later. Additionally, all through hole components (e.g. power jack, BNC connectors, etc) will be populated later. Start by placing the

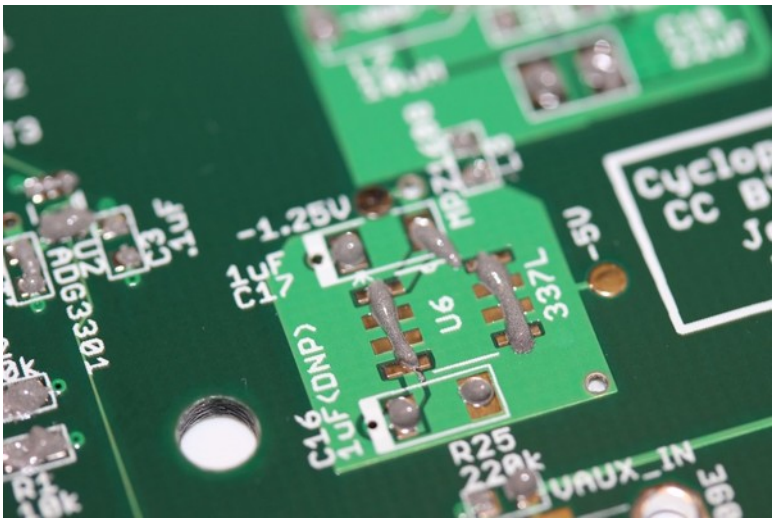


Figure 18: Good solder placement.

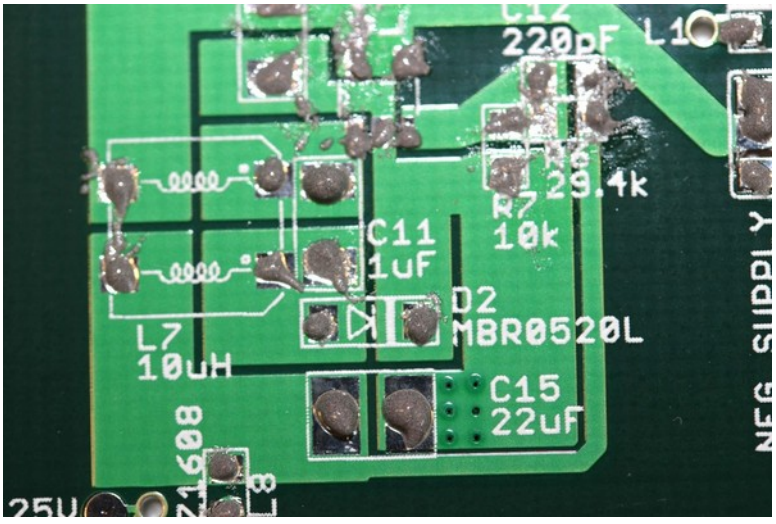


Figure 19: Bad solder placement. Too much paste!

integrated circuits (ICs). Use the stereoscope or loupe to ensure that pads are making contact with the pins of the placed components. Precise component alignment is not necessary. Components will self-align during the reflow process.

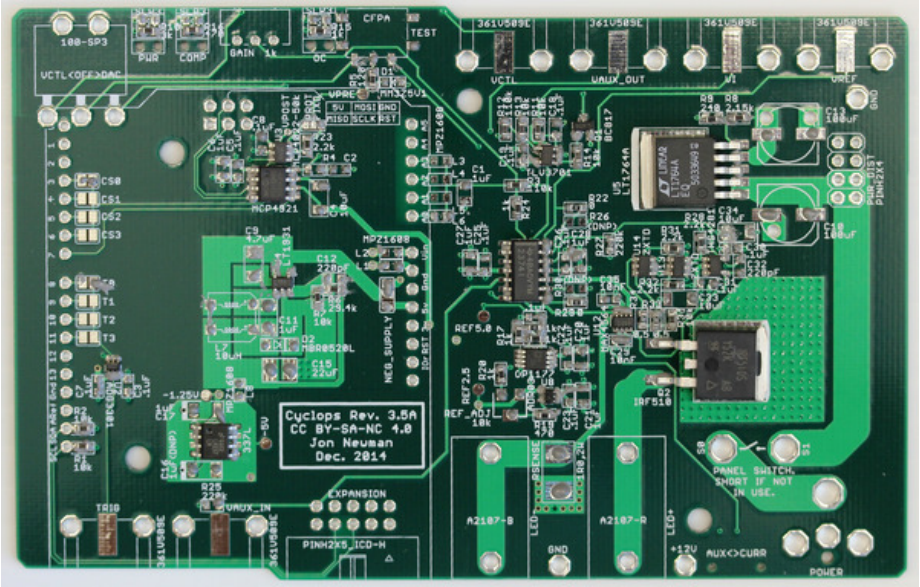


Figure 20: Integrated circuit population.

5. After placing the ICs, place the passive components (resistors, capacitors, inductors, diodes, and ferrite chips).

6. Next, reflow solder the board. We use a homemade reflow oven constructed from a toaster oven, Arduino board, reflow oven control shield, and mains relay. You can make a similar one, use a commercial reflow oven, or use the hot air station. Reflow the solder paste on the board using your oven or hot air gun as described in the links above.

7. After the solder has cooled, examine solder pads using the stereoscope of loupe for solder bridges between pins, solder that has not melted, or pads lacking a decent solder joint. Fix any issues using a standard soldering iron If there are solder bridges present, get rid of them using some solder wick before moving on. Solder through-hole components in place using a standard soldering iron. A low cost reflow oven can be made form a toaster oven as shown here. This link also contains useful information on the basics of the reflow soldering process,

8. Each board has an address (0 through 3) that is defined by two solder jumpers and the location of a ferrite chip. This allows cyclops boards

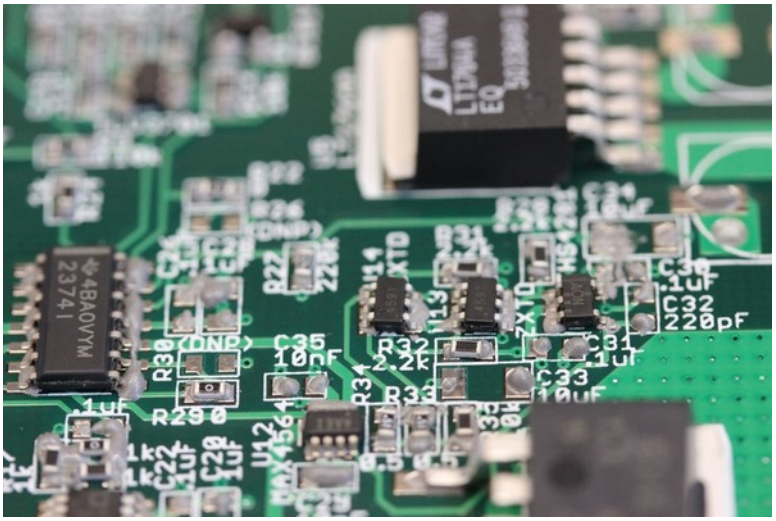


Figure 21: Zoomed view of integrated circuit placement.

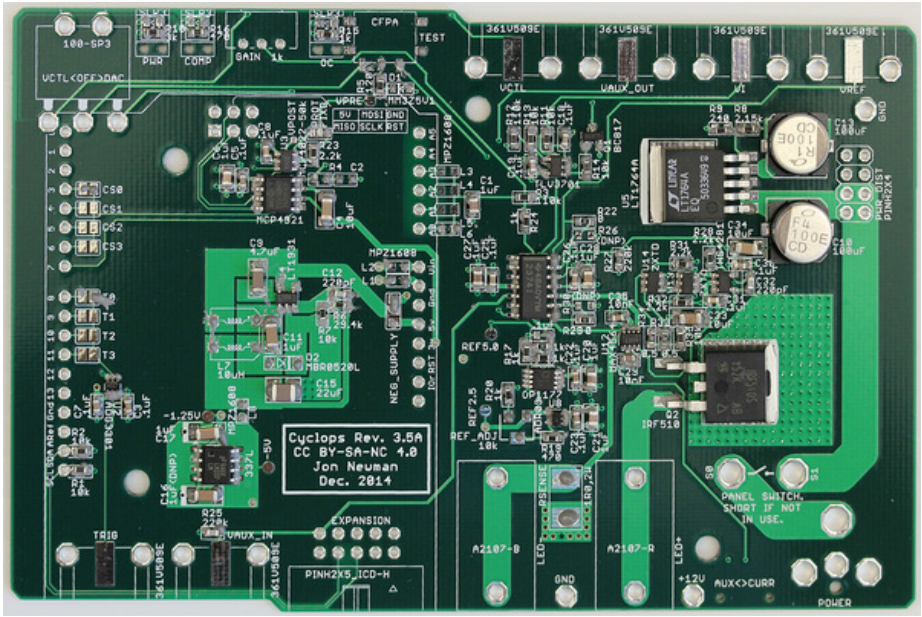


Figure 22: Board following resistor and capacitor population.

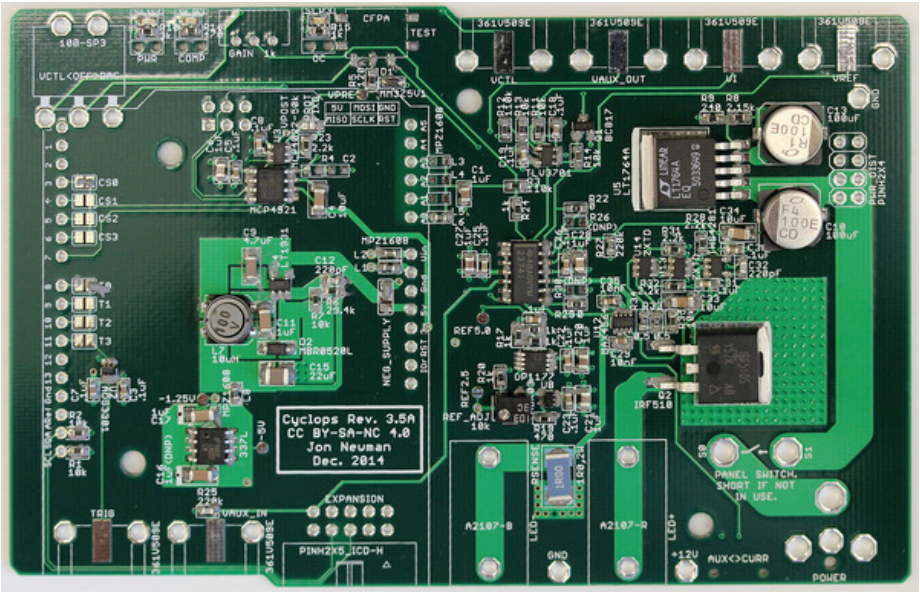


Figure 23: Board with all top-side surface mount components installed.

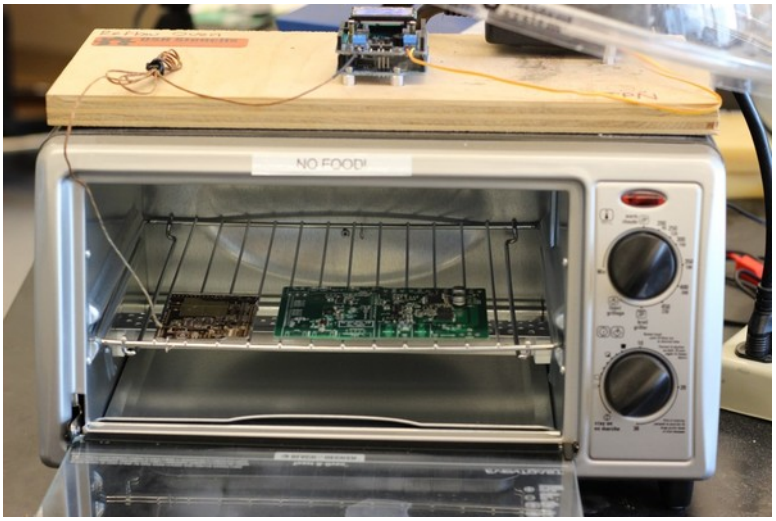


Figure 24: Homemade reflow oven with the populated board inside.

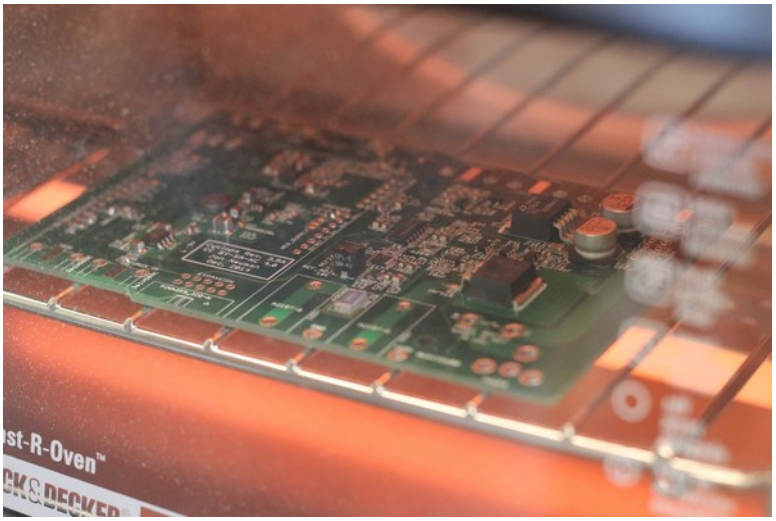


Figure 25: The board is shown after the reflow temperature has been reached. Reflow will occur at different temperatures depending on the specification of the solder paste you are using.

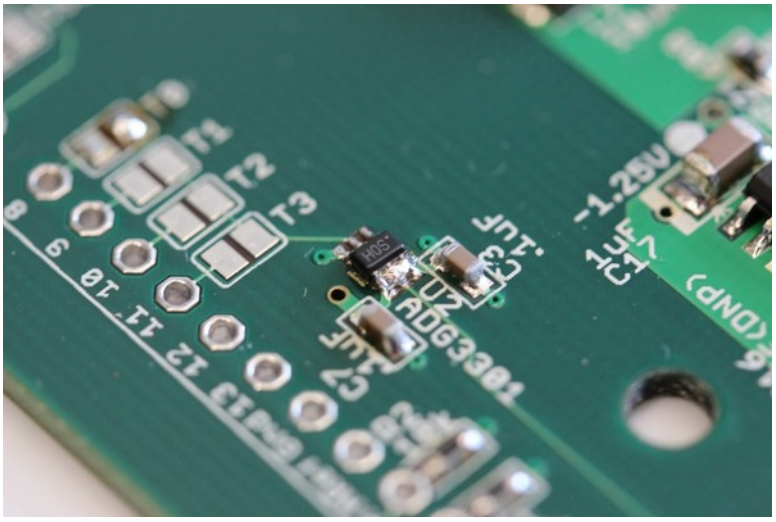


Figure 26: Example of a short between IC pins. This must be resolved before moving forward.

Figure 27: Dipping the copper braid in flux will make the solder wick much more redily.

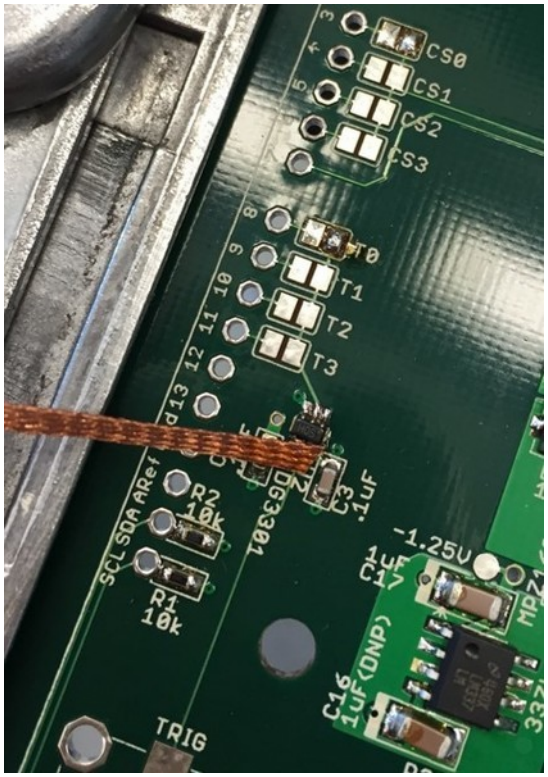


Figure 28: Place the copper braid over the solder blob and then press with the soldering iron. You should see the excess solder wick up the copper braid.

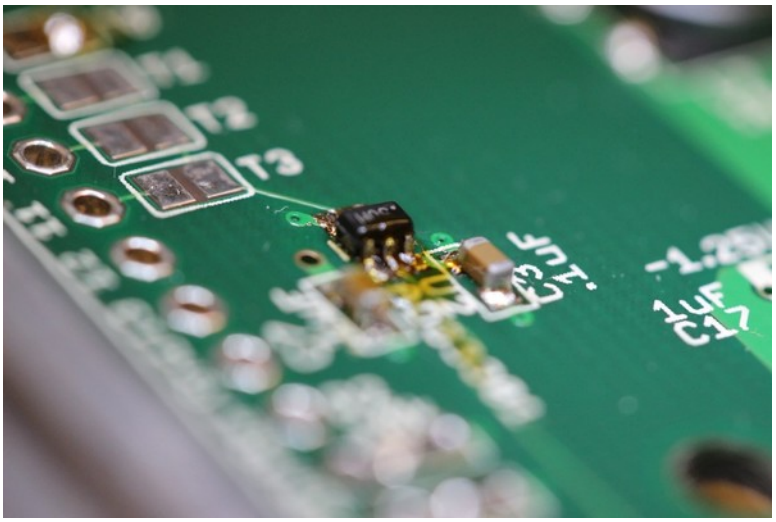


Figure 29: Often you will not have to re-apply solder after this process because there will be enough left over to maintain a good electrical contact. Once you are satisfied the flux residue can be cleaned using isopropyl alcohol

to be stacked to share a power supply while being driven by a common microcontroller. For each board that will share a microcontroller, a unique address must be specified and the solder jumpers and ferrite chip must be soldered in appropriate positions to reflect this address. See the picture below to better understand this addressing scheme.

TODO: Pictures of circled 0 ohm jumpers

9. Flip the board over and install the final surface mount component, the `AUX<>CURR` and `BW LIM` slide switches, by hand soldering.

10. Next, populate all electromechanical components. This can be soldered in place with a standard soldering iron and a large chisel tip.

    Note: The the barrel power jack (name: `POWER`, value: PJ-063BH on the schematic) should be mounted on the **bottom** of the board. It fits on both the top and the bottom, and will properly supply the board with power if mounted on the top. However, if the barrel jack is mounted on the top side of the board, it will not fit inside the enclosure.

11. If you are not planning on putting the PCB in an enclosure, jumper the solder points for the power switch together using AWG 20 (~1.8 mm diameter) braided copper wire or thicker.

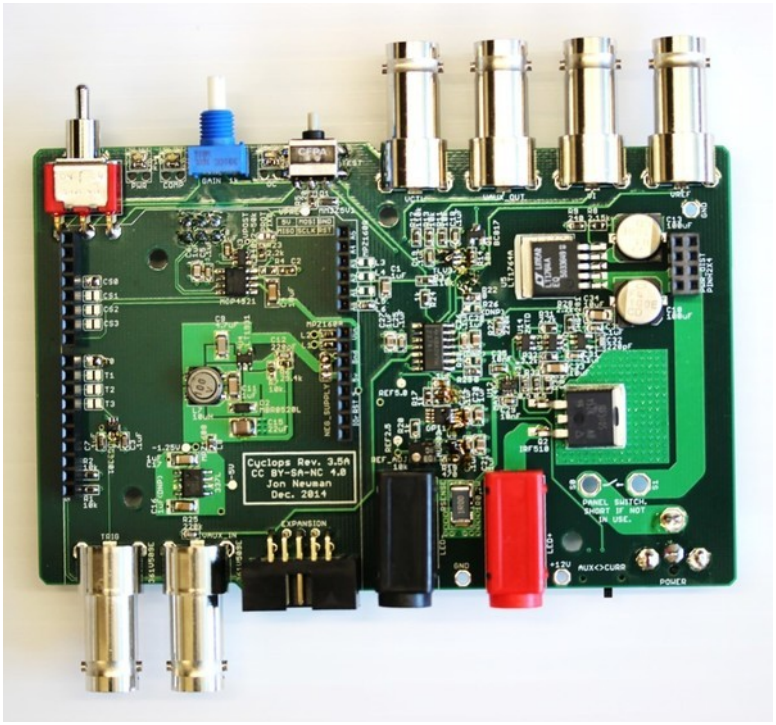


Figure 30: Top side of board following electromechanical component installation.

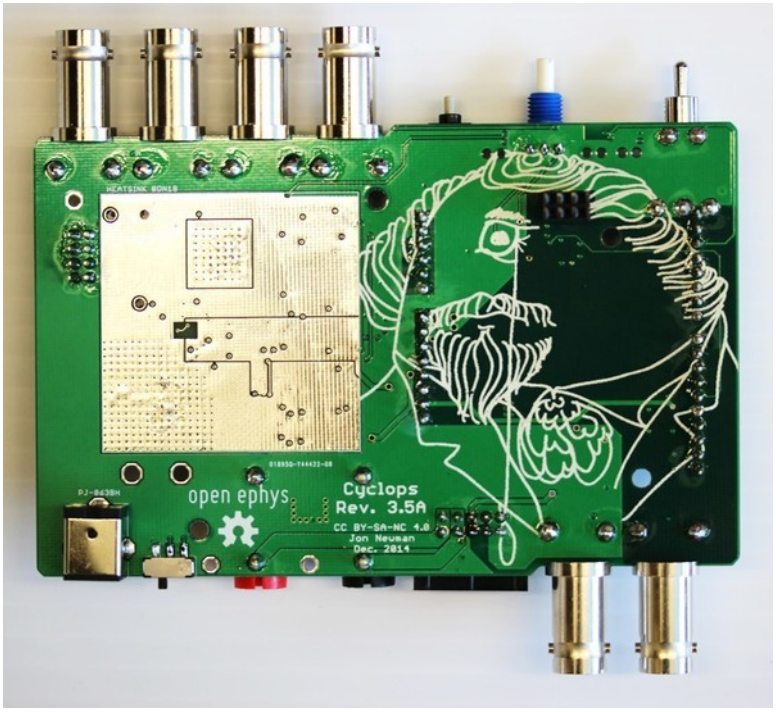


Figure 31: Bottom side of board following electromechanical component installation.

Device Assembly Manual

Required Parts and Tools

Before starting, please ensure you have the following components. Suggested part numbers and suppliers are provided on this google spreadsheet. In the lists below, numbers (#XX) correspond to the labels on the image below.

1. Assembled Cyclops PCB (#6, 1x)
2. Extruded aluminum enclosure (#1,1x)
3. Enclosure panels (front: #20, 1x; rear: #21, 1x)
4. Rocker switch (#18, 1x)
5. 18 AWG silicone insulated hookup wire (#17, 2x)
6. Shrink tube (#19, 2x)
7. Gain knob (#7, 1x), jam nut (#8, 1x), and toothed washer (#9, 1x)
8. Button cover (#10, 1x)
9. Waterjet-cut aluminum heatsink (#13, 1x)
10. 6mm length, M3 thread-forming screws (#11, 11x)
11. Nylon washers (#12, 3x)
12. Silicon thermal compound (#4)
13. Thermal tape (#5)
14. Teensy 3.2 MCU board (#15, 1x)
15. 0.1" sockets for Teensy 3.1 (#16, 2x 1x14 pin; 1x 1x2 pin; 1x 2x7 pin SMD)

You will also need the following tools.

1. Scissors
2. Soldering iron
3. Hot air gun or lighter
4. T10 Torx key or driver (#2)
5. XX Allen key or driver (#3)
6. Teensy soldering jig made from breadboard with male headers (#14)

Additionally, to use your device once it is assembled, you will need:

1. An LED to drive [Required]

- See the LED section for options.

1. A power supply [Required]

- Any wall-wart, battery pack, or bench-top power supply providing 15-20 VDC that can source >=2A will work.

- The power jack is **center positive** (but reversing this will not ruin the device, it just wont turn on).
- Look around and see if you have a wall-wart laying around the lab that meets the specs. They are pretty common and the jack is likely to fit.
- Digikey part number 1470-3096-ND or equivalent is a good option for those that need to buy a supply.

1. An M8, 4-pin connector [Optional]

- An inexpensive, standard, non-insane, 2x5 header (`EXPANSION A`) provides access to most of the internal signals on the Cyclops board, e.g. for driving an LED or providing auxiliary feedback using an amplified photodiode etc.
- An expensive, strange, moderately insane, M8 expansion connector that is used by two major companies in the optics space (`EXPANSION B`) is left unpopulated. You can populate this port if you wish to drive Thorlabs or Doric LED modules using the Cyclops. For instructions, see this the complete manual.

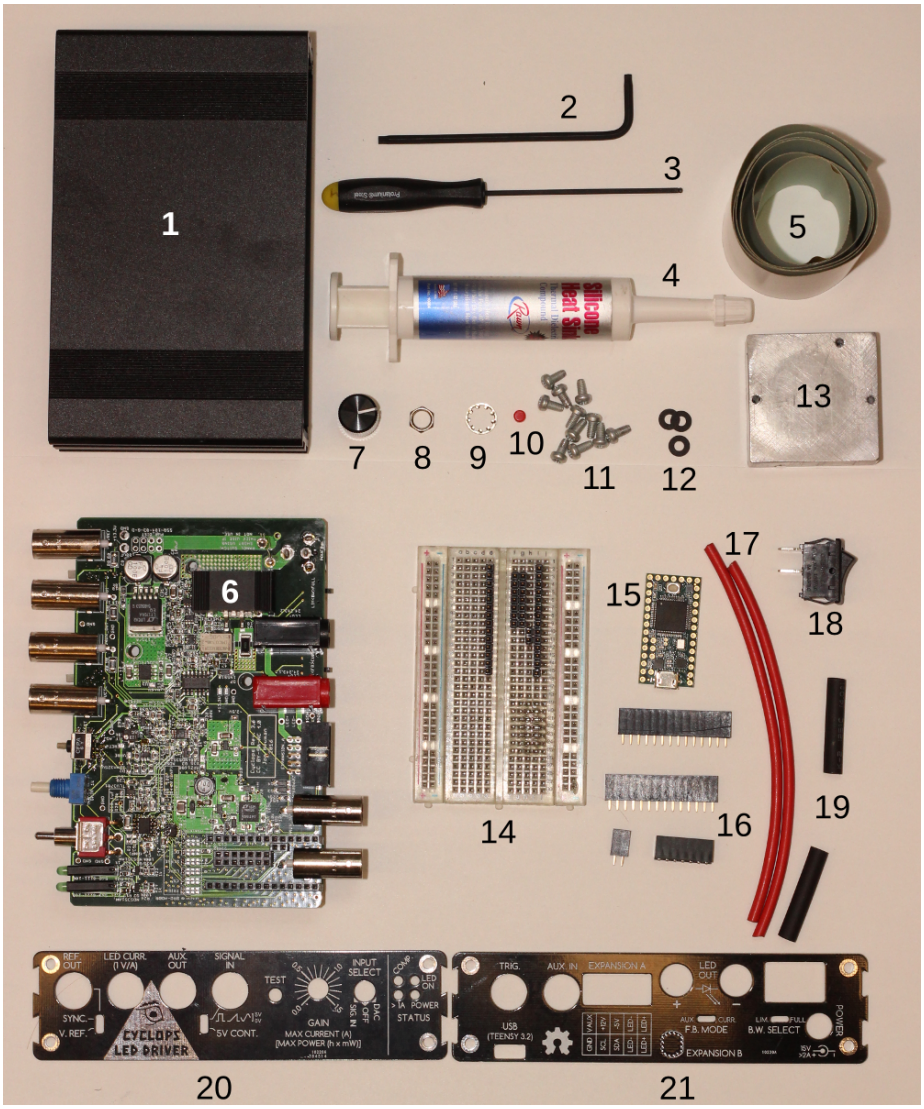


Figure 32: Cyclops device parts and tools

Device Assembly Instructions

1. Assemble the teensy 3.2

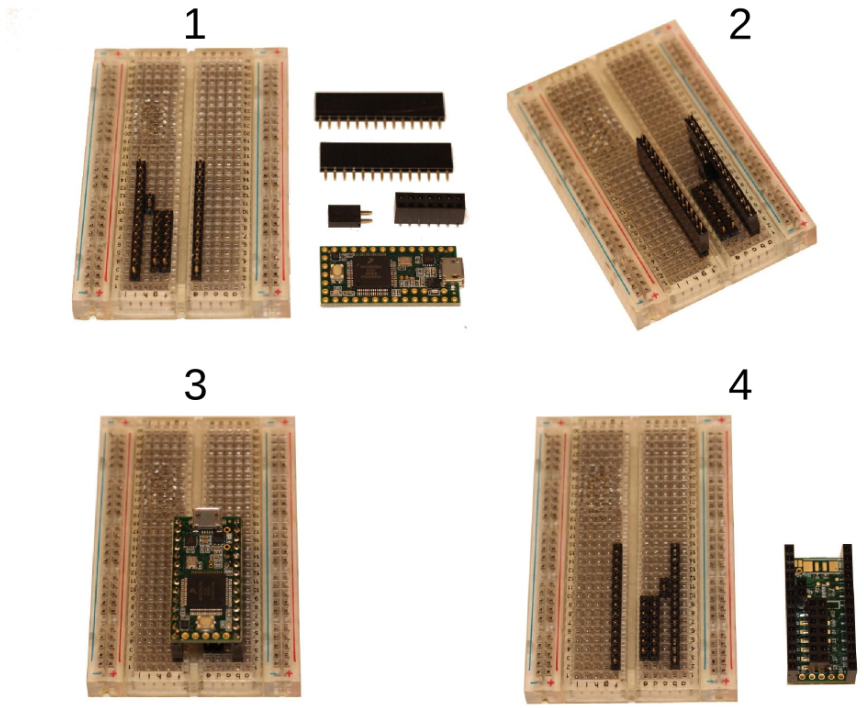


Figure 33: Teensy MCU assembly

- Solder the surface mount, 2x7 pin header to the bottom of the teensy. It is important to do this first because its very hard to get to after the other headers are installed.
- Put the single row headers on the soldering jig (panel 2, above), followed by the Teensy (panel 3, above). The microcontroller chip should be facing up.
- Solder the headers in place.

2. Assemble the power switch

- Strip about 1 cm of insulation from each end of the hookup wires
- Thread the stripped portion of the wire halfway through each of the switch's solder terminals. Fold the wire back, so that the stripped part is touching both sides of the terminal.

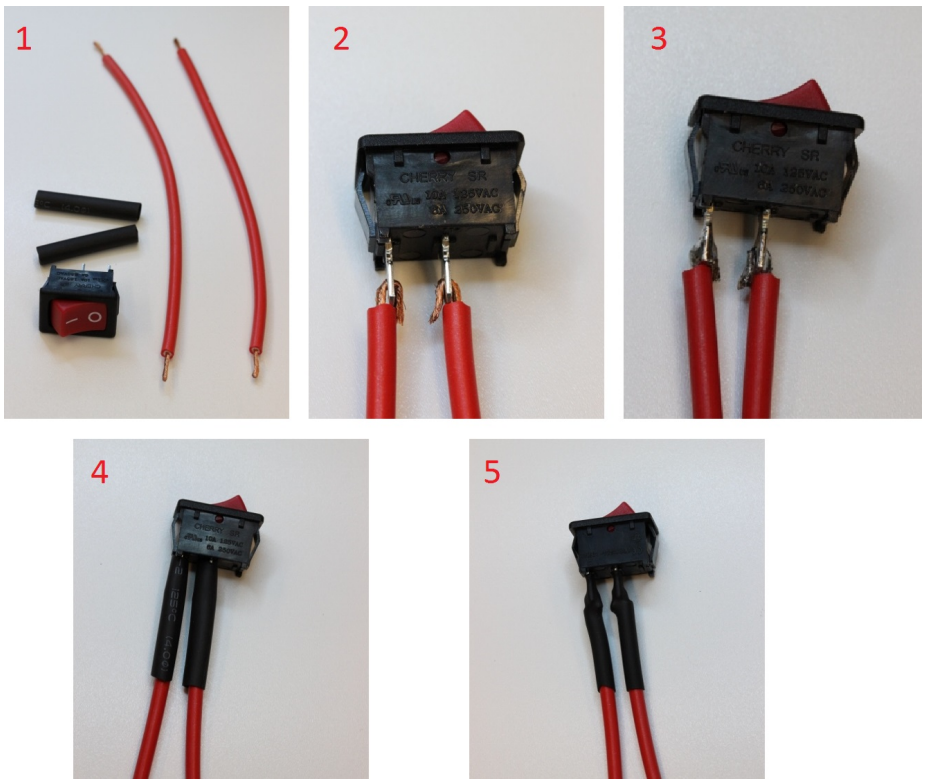


Figure 34: Power switch assembly

- Solder the hookup wire to the terminals. Make sure the solder flows between the wires' copper braid and the switch terminals.
- Slide the heat shrink from the back of each wire, over the solder joints. Hit then with a hot air gun or pass a lighter underneath them to shrink them into place over the solder joints.

3. Prepare the back panel

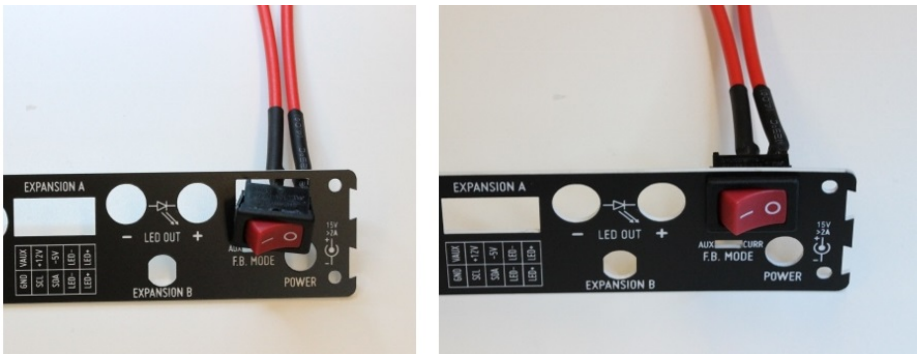


Figure 35: Power switch installation

- Press the power switch into position on the back panel. Orient the switch so that the 'on' symbol (-) is toward the middle of the panel and the 'off' symbol (o) is toward the outside. This will make the panel easier to mount on the enclosure.
- The switch will snap into place.

4. Install the teensy on the cyclops

- Press the previously assembled teensy onto the 0.1" pitch male headers. It should be *upside down* on the bottom of the cyclops PCB.

5. Put thermally conductive tape on the heatsink

- Cut a square of thermal tape from the roll
- Peal off the backing and press the tape on the back of the heatsink.
- Poke holes through the tape to expose the bolt holes in the heatsink.

6. Install the heatsink on the PCB

- Flip the PCB so that the bottom is exposed.

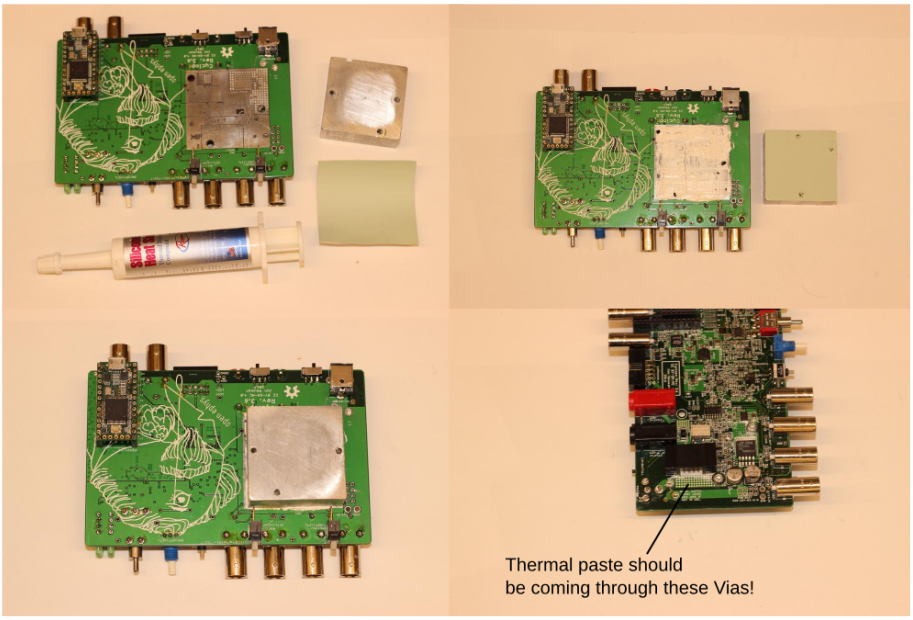


Figure 36: Heatsink installation

- Locate the large white square surrounding the exposed copper pad indicating the heatsink mounting location
- Apply an even layer of thermal paste to the area.
- Remove the paper backing from the tape on the heatsink to expose the adhesive surface
- Press the heatsink into place on the PCB
- Filp the PCB so the top is up and locate the three holes used to mount the heatsink to the PCB. Thead three self-tapping screws through a nylon washer and then into the heatsink itself. Hand tighten until the heat sink is firmly head against the back of the PCB. You should see thermal compount coming through the numerious vias in the PCB. This is disired because it provides a low thermal resistance path between the PCB and the heatsink.

6. Solder the power switch to the PCB.

- Solder the power switch to the two indicated terminals on the PCB.
- Make sure you do this **after** you have installed the switch in the rear panel.
- The wires can be soldered to either solder point, orientation does not matter since this is a SPST switch.

7. Install the front panel on the enclosure

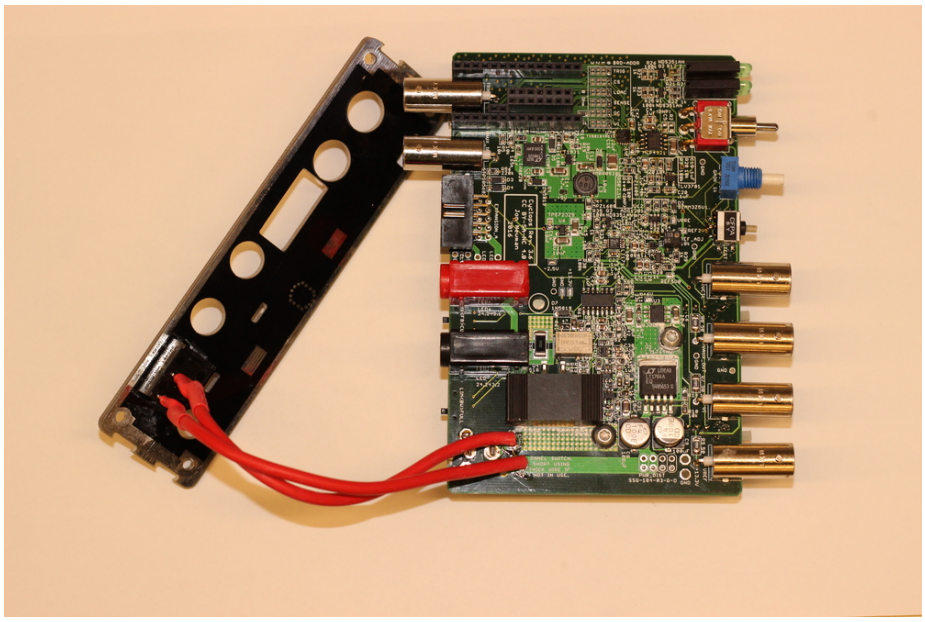


Figure 37: Power switch soldering

Figure 38: Front panel installation

- Using 4 of the 11 self-trapping screws, install the front panel on the enclosure.
- **Note:** The orientation of the enclosure matters. The top of the enclosure is indicated by rows of decorative lines as indicated in the above figure.

8. Install the PCB in the enclosure.

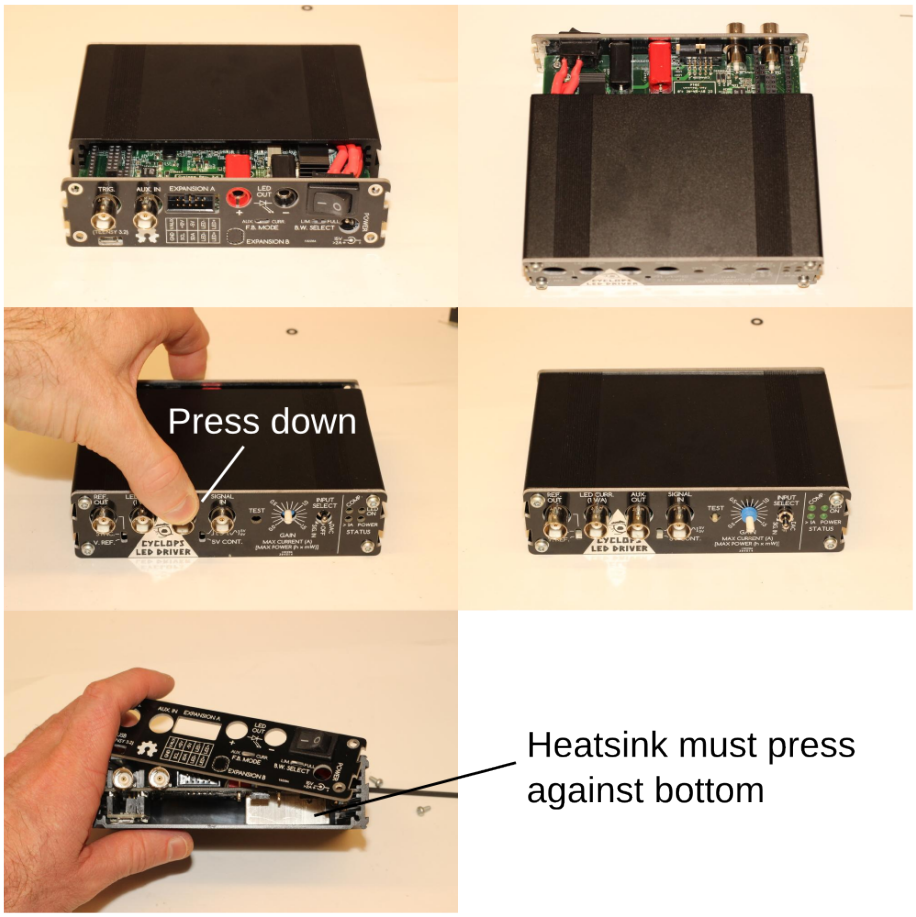


Figure 39: Front panel installation

- Slide the PCB into the box using the **middle** mounting rail.
- **Note:** The heatsink should be making firm contact with the bottom of the enclosure. If it is not, you should add thermal paste to the bottom of the heatsink to fill the gap. The heatsink must have a low thermal resistance path to the enclosure to dissipate heat during operation.

- **Note:** When the BNC connectors come through the front panel, you will need to push them down a bit, slightly flexing the PCB to get the panel mount controls through the front panel.

9. Install the rear panel.

- Loop the two hook up wires and push them into the enclosure on top of the PCB.
- Use the panel the push the remaining wire into the enclosure.
- Use the remaining 4 screws to install the rear panel.

10. Install gain knob and button cover.

Figure 40: Gain knob installation

- Slip the toothed washer over the gain dial.
- Tighten the jam nut on the gain dial's threads until just past finger tight using a pair of needle nose pliers.
- **Note:** Do not over-tighten using a combination spanner or you will strip the threads.
- Turn the dial on the front panel *fully counter-clockwise*.
- Slip the gain knob over the dial with the tick mark pointed slightly below the 0 position.

- Tighten the set screw on the side of the knob to lock it into place using an Allen key.
- When you turn the knob fully clockwise, the tick mark should be pointing close to the 1.5A position.
- Press the red cover over the `TEST` switch until it snaps into place.

Congratulations, you are the proud owner of a high-precision, high-power, high-speed LED driver that will make commercial drives feel a bit ridiculous for costing so much and very self-conscious about their performance characteristics. For further usage instructions, performance specs, theory of operations, etc, etc, please refer to the complete manual located on the repository. As stated previously:

LED

There are several things to consider when determining the type of LED you wish to drive with the Cyclops and the configuration of the LED.

- Will optical stimulation be used in-vivo or through a microscope?
- Will it be performed on freely behaving animals?
- Do you need to perform bilateral stimulation?
- Will you need to incorporate an amplified photodiode into your stimulator to measure optical power or use use optical feedback to produce ultra precise light waveforms?

The answers to these questions will determine the type of LED you use, how it is coupled to the preparation (e.g. collimated for the back aperture of your microscope or, fiber coupled for in-vivo stimulation), and weather or not it needs to be commutated in some way. The following provide a few simple options for LED configurations, but there are many more to consider for your experiments.

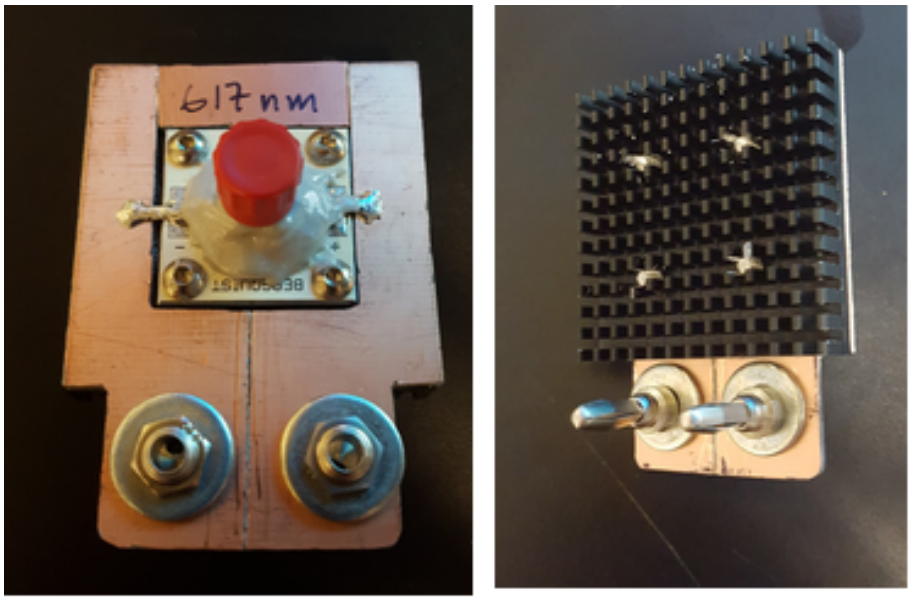


Figure 41: A simple 'pigtailed' LED module. A female SMA connector is cemented over a lens-free high power LED to form a near optimal butt-coupling between the fiber and the LED die.

Fiber-coupled LED

DIY Solution

Anders Asp has contributed the following PDF document containing detailed instructions for fabricating a bilateral, commutated fiber-coupled LED for use in freely moving animals that works with the Cyclops driver:

Bilateral fiber-coupled LED

Thorlabs fiber-coupled LED modules

The cyclops can be used to drive Thorlabs fiber-coupled LED modules. You will need to install the M8 4-position connector in expansion port B to drive these LEDs.

TODO: Pictures/instructions for M8 installation process in Thorlabs configuration

Doric LED fiber-coupled modules

The cyclops can be used to Drive Doric fiber-coupled LED modules. You will need to install the M8 4-position connector in expansion port B to drive these LEDs.

TODO: Pictures/instructions for M8 installation process in Doric configuration

Microscope mounted LEDs

The cyclops can be used to Drive Thorlabs collimated LEDs for microscope-based stimulation. You will need to install the M8 4-position connector in expansion port B to drive these LEDs. See Thorlab fiber-coupled LED instructions for instructions.

Quality Control Procedure

The following procedure can be performed on assembled boards to ensure functionality.

Setup

- Insert alligator clip across power switch solder points
- Insert device into PCB clamp
- Power from 15V, 1.5A capable bench-top power supply.
- [ ] Power indicator LED turns on.

DC Levels

- Using a multimeter, probe the 12V, 2.5V, -5V, and -1.25V test points
- [ ] 12V good
- [ ] -5V good
- [ ] -1.25V good
- [ ] While probing the 2.5V test point, use a ESD-safe screwdriver on the trimpot to get exactly 2.5V.
- [ ] Seal the pot with a dab of hot-glue.

Dynamic characteristics

- Set MDO3000's AFG to produce 1-5V, 100 Hz, 10% duty cycle square wave.
- Insert LED/amplified photodiode test fixture into banana sockets, IDC connector, and AUX BNC port.
- Insert AFG output of MDO3000 output into VCTL BNC port of device
- Bring CURR output of device to Ch1 of MDO3000
- Bring VREF output to MD3000
- Triggering on VREF Channel set scope to measure rise and fall times
- Bring front panel potentiometer to 50% position.
- Input switch to EXT source
- [ ] Examine wave shape and rise/fall times in **current** FB mode. Rise/fall times < 300 ns. No ringing on waveform.
- [ ] Examine wave shape and rise/fall times in **optical** FB mode. Rise/fall times < 300 ns. No ringing on waveform.
- [ ] **Return FB switch to curr position**
- [ ] Return input switch to OFF (middle) position

Overcurrent indication

- Bring gain potentiometer to full on position
- Briefly tap on the TEST button.
- [ ] Ensure that the >1A indicator LED turned on during pulse.
- [ ] **Return gain potentiometer to zero position**

Finish

- [ ] Remove all power connectors.
- [ ] Remove alligator clip.
- [ ] Initial and serial number the board using sharpie on the large power trace on the right side of the board.
- [ ] Enter board serial number into the spreadsheet.

License

Hardware Licensing

Software Licensing

References

[1] J.P. Newman, M.-f. Fong, D.C. Millard, C.J. Whitmire, G.B. Stanley, S.M. Potter. S.M. Potter. Optogenetic feedback control of neural activity. *eLife* (4:e07192) 2015. doi: 10.7554/eLife.07192

[2] T. Tchumatchenko*, J.P. Newman*, M.-f. Fong, S.M. Potter. Delivery of time-varying stimuli using ChR2. (* - equal contributions, co-first authors) *Front. Neural Circuits* (7:184) 2013. doi: 10.3389/fncir.2013.00184