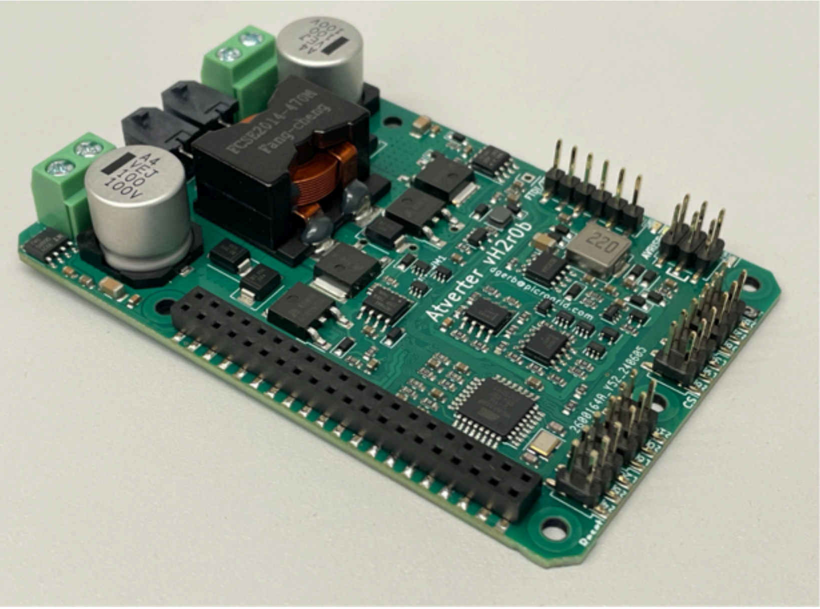
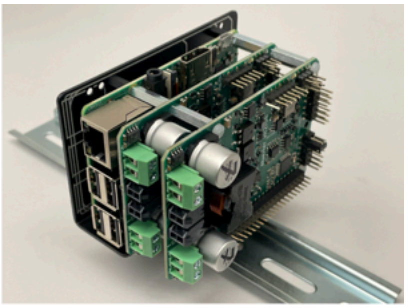
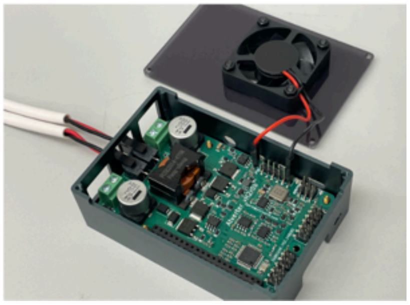
AtverterH v2r1

Atverter Hobbyist - 48V 5A Programmable DC-DC Converter







Description

The Atverter is a programmable DC-DC converter, powered by the ATmega328p microcontroller. It is a bidirectional 4-switch buck-boost converter topology, with terminals rated for 60V and 5A. The Atverter can be programmed like an Arduino Uno and controlled as a Raspberry Pi shield.

Datasheet Contents

Features

- Onboard Atmega328p that is programmable like an Arduino Uno
- Controllable from a Raspberry Pi 40-pin header
- Up to 96% efficiency measured
- Slim design that fits in a Raspberry Pi fan case

- Easily stacked and DIN rail mountable - up to 6 Atverters per Pi
- Voltage and current sensing per terminal. Temperature sensing on switches
- Hardware safety shut-off when voltage exceed 65V or current exceeds 6.5A
- LED indicators for quick on-site debugging
- Molex Microfit snap ports allow rapid deployment and reconfiguration

Applications

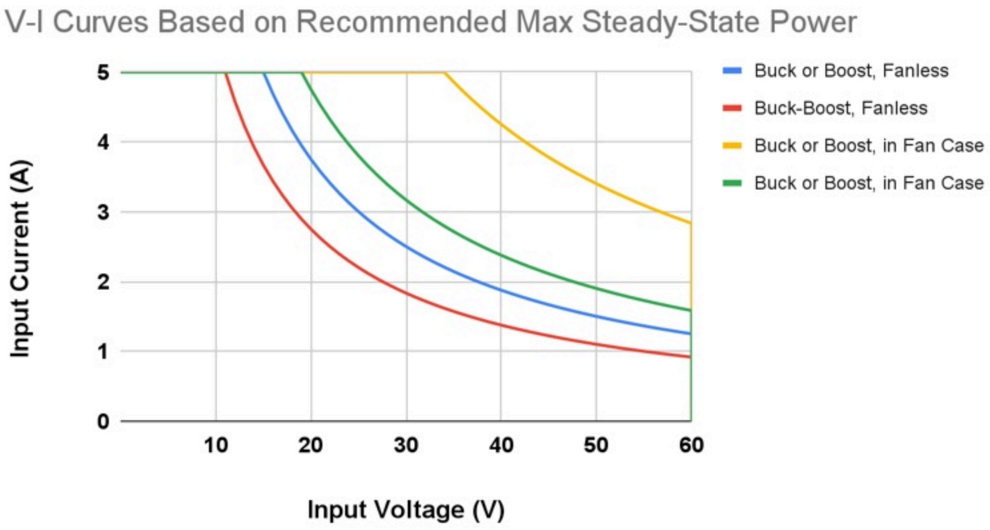
- Open source projects
- Controllable power supply
- DC microgrids and off-grid
- Behind-the-meter power transfer
- Renewable generation and MPPT
- Resilience and battery charge control
- DC bus stabilization
- Controllable LED driver
- Controllable DC motors

Absolute Max Ratings

Steady-State Rating	**Recommended Max**	**Absolute Max**
Terminal Voltage	60 V	65 V
Terminal Current	5 A	6.5 A
Measured Switch Temperature	80 C	90 C
Input Power[1] - Buck or Boost, Fanless	75 W	100 W
Input Power[2] - Buck-Boost, Fanless	55 W	65 W
Input Power[3] - Buck or Boost, in Fan Case	170 W	
Input Power[4] - Buck-Boost, in Fan Case	95 W	105 W

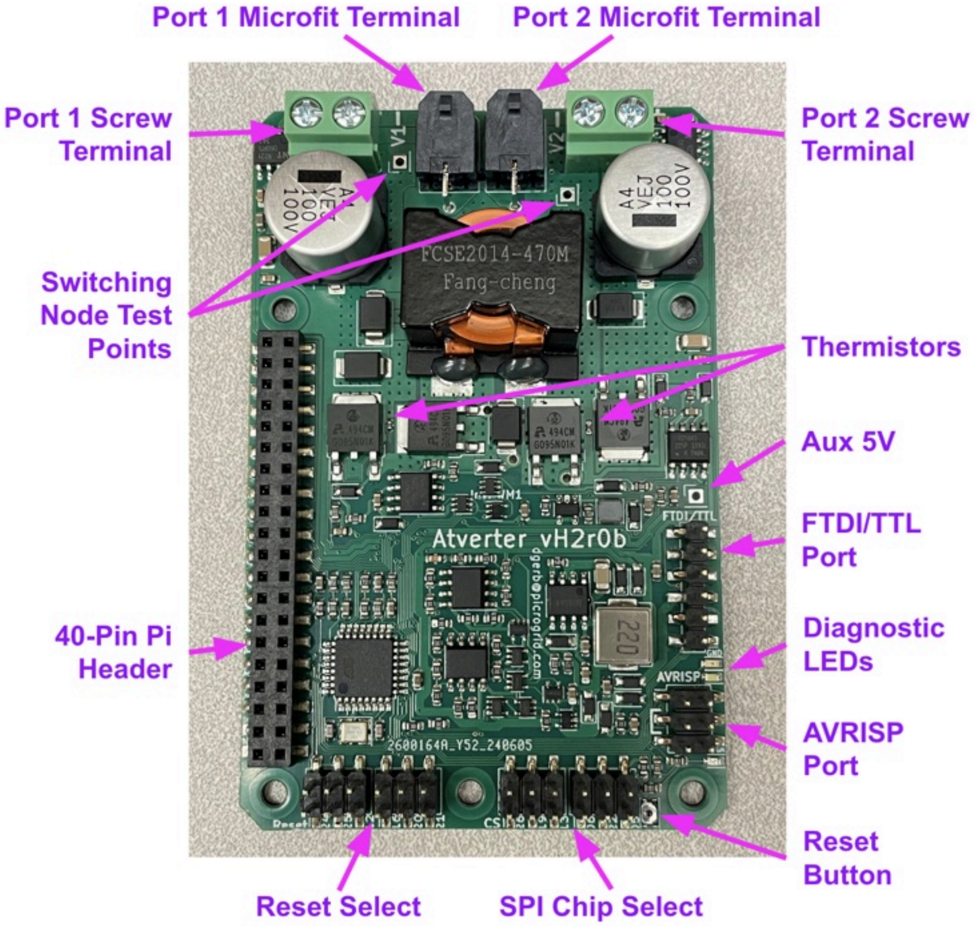
[1] Buck mode, 48V input, 50% duty; temperature does not exceed max within 5 minutes
[2] Buck-boost mode, 24V input, 50% duty; temperature does not exceed max within 5 minutes
[3] Buck mode, 56V input, 60% duty; temperature does not exceed max within 5 minutes
[4] Buck-boost mode, 36V input, 50% duty; temperature does not exceed max within 5 minutes

Power ratings are measured based on whether the switch temperature exceeds the recommended/absolute max within a 5-minute period. The following V-I curves show the recommended steady-state operating limits on input voltage and current.



Board Diagram

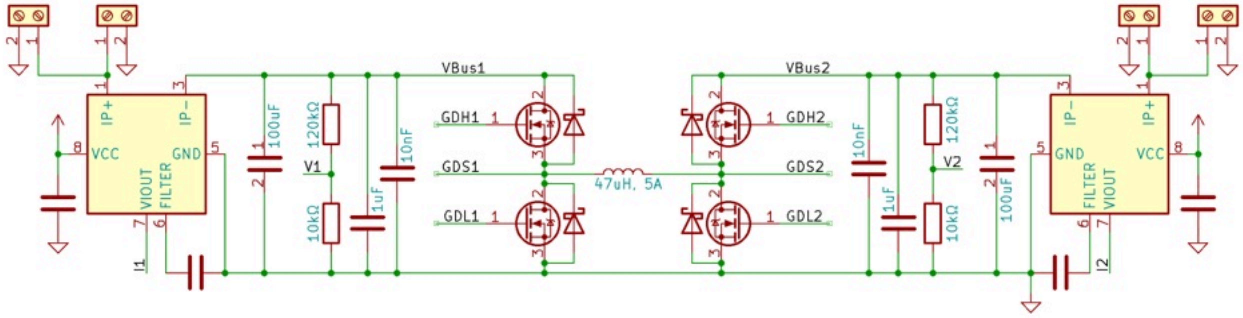
The following image shows where the terminals, connectors, and features are located:



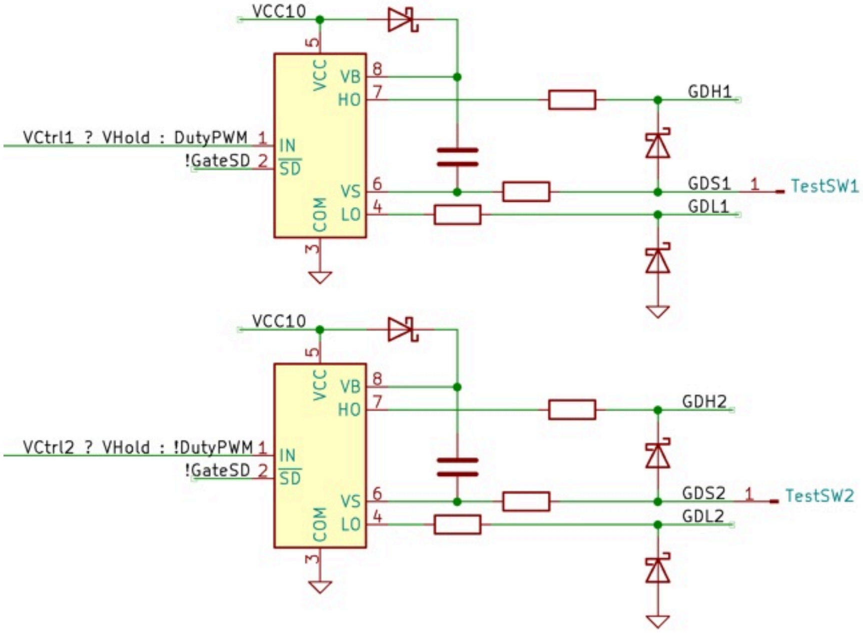
Application Schematics

The following application schematics are provided for educational and debugging purposes:

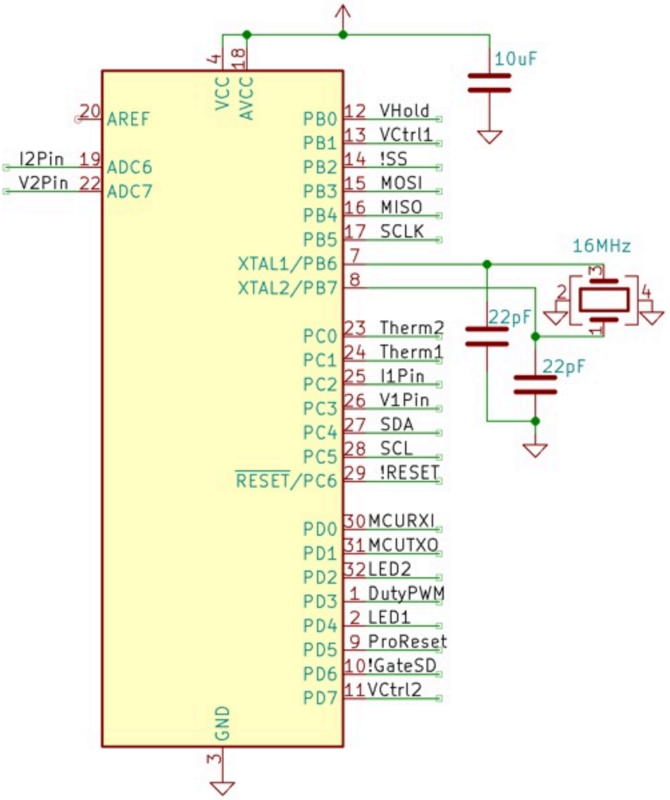
Four-Switch Buck-Boost Power Stage



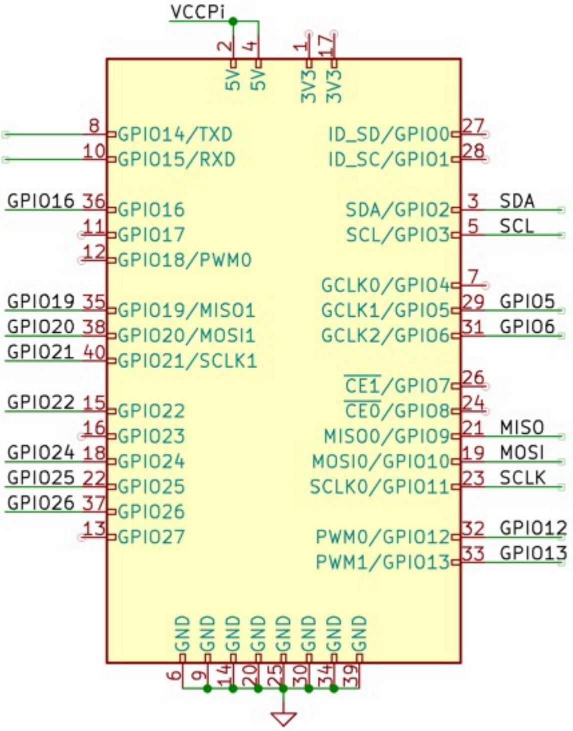
Gate Drivers



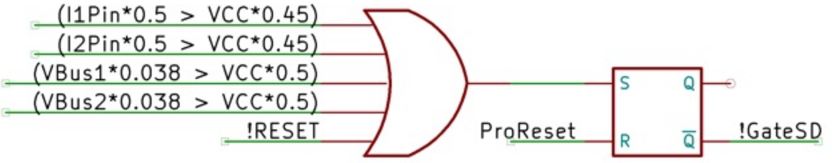
ATmega328P Pin Assignment



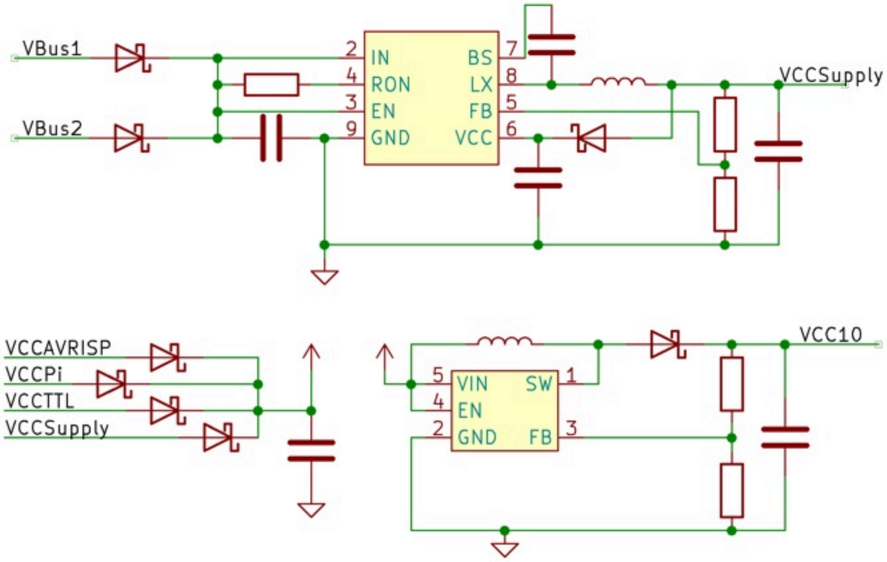
Pi Header Pin Assignment



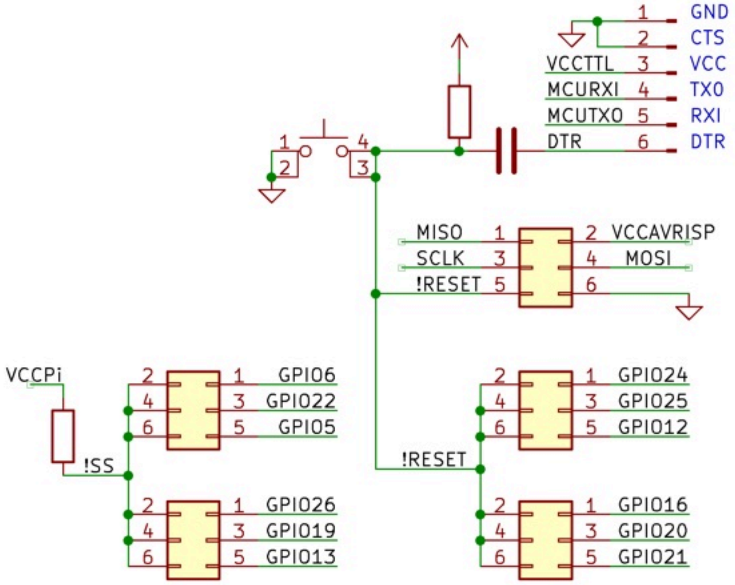
Hardware Protection Circuit



Onboard Power Supply



Reset, Programming, Chip Select



How to Program

There are three ways to load your C++ code onto the Picrogrid board:

- USB/TTL cable
- AVRISP programmer
- Raspberry Pi SPI pins

Please refer to the Picrogrid GitHub repository for any code references:
https://github.com/dgerb/Picrogrid

Flashing the Picrogrid board with the USB/TTL cable

You can use an FTDI USB/TTL cable to flash the Picrogrid board similar to how you would upload code to an Arduino Uno. This method is recommended when you are already using the Arduino serial console for diagnostics. This method requires that the Arduino bootloader is already installed on the ATmega. For blank ATmegas (i.e. the first flash of a new Picrogrid Board), refer to one of the other two methods to flash the bootloader. This method has been tested with a Sparkfun 5V FTDI cable.

First, plug in the FTDI USB/TTL cable, noting that the black wire corresponds to the GND marking on the board.

In one of the AtverterHExamples Github subdirectories, open an example program (such as Blink) in the Arduino IDE. Once open, select the Tools tab, and ensure the following settings:

- Board: "Arduino Uno" (found under Arduino AVR Boards)
- Port: "/dev/cu.usbserial-AB0JTGZW" (this is what shows up on my mac, will be different for others)
- Programmer: "AVRISP mkII"

You should be able to compile and upload the example. If you selected Blink, you should see flashing lights after a few seconds.

Flashing the Picrogrid board with the AVRISP programmer

You can use an AVRISP programmer to flash the execution code or a boot loader. This method is recommended when you need to flash a new board for the first time, but do not wish to use the board with a Raspberry Pi. The method has been tested with the HiLetgo USBTinyISP programmer.

First, plug in the AVRISP programmer, noting that the extrusion on the side of the 6-pin connector should point away from the edge of the board.

In one of the AtverterHExamples Github subdirectories, open an example program (such as Blink) in the Arduino IDE. Here, you can do one of two things: (a) burn the boot loader and upload code via a FTDI USB/TTL cable, or (b) upload code via the programmer, knowing that this does not upload a boot loader.

(a) To burn the boot loader, select: Tools > Burn Bootloader
(b) To upload code via the programmer, select: Sketch > Upload Using Programmer

Note that uploading code via the AVRISP programmer does not necessarily upload a bootloader. Sometimes, you might have to burn the bootloader first so as to properly synchronize the clock frequency.

Flashing the Picrogrid board with the Raspberry Pi

The final method of flashing the ATmega is through the SPI pins on a Raspberry Pi. This method is the most complicated, and is only recommended if the board will connect to a Raspberry Pi and may require firmware code updates after installation. First, properly mount the Atverter on a Raspberry Pi.

Open the Arduino IDE with the code and settings of "Board: Arduino Uno". Click the “Verify” button. Once successfully verified/compiled, go to "Sketch > Export Compiled Binary". This will export the compiled code as two hex files in the project directory. One of these files includes a boot loader, which you will probably want. You need the boot loader to be able to flash the Atverter with the USB/TTL cable.

Open a terminal and copy the binary file to the Pi. Here is an example SCP terminal command from my mac:

- scp Blink.ino.with_bootloader.standard.hex pi@raspberrypi.local:Desktop

SSH into the Pi. The terminal command on my mac is (for Windows, use PuTTY):

- ssh pi@raspberrypi.local

Once connected to the Pi, use AVRDude to flash the Atmega over SPI. If you haven't done so already, you will need to set up the Pi as an AVR programmer, following the instructions in the RaspberryPi/Setup Github subdirectory:

Confirm that the board's Reset Jumper corresponds with the listed reset pin in the /usr/local/etc/avrdude.conf file for the selected programmer (typically linuxgpio).

In this example, the avrdude command is:

- sudo avrdude -c linuxgpio -p atmega328p -v -U flash:w:Desktop/Blink.ino.with_bootloader.standard.hex:i

If there are problems, you can check connectivity with the Atmega using this command:

- sudo avrdude -c linuxgpio -p atmega328p -v

Mounting Guide

Here are two recommended options to mount and deploy the Atverter:

- For multi-board stacking with a Raspberry Pi, use a DIN rail mounting system
- For standalone converter applications, use a Raspberry Pi case with a fan

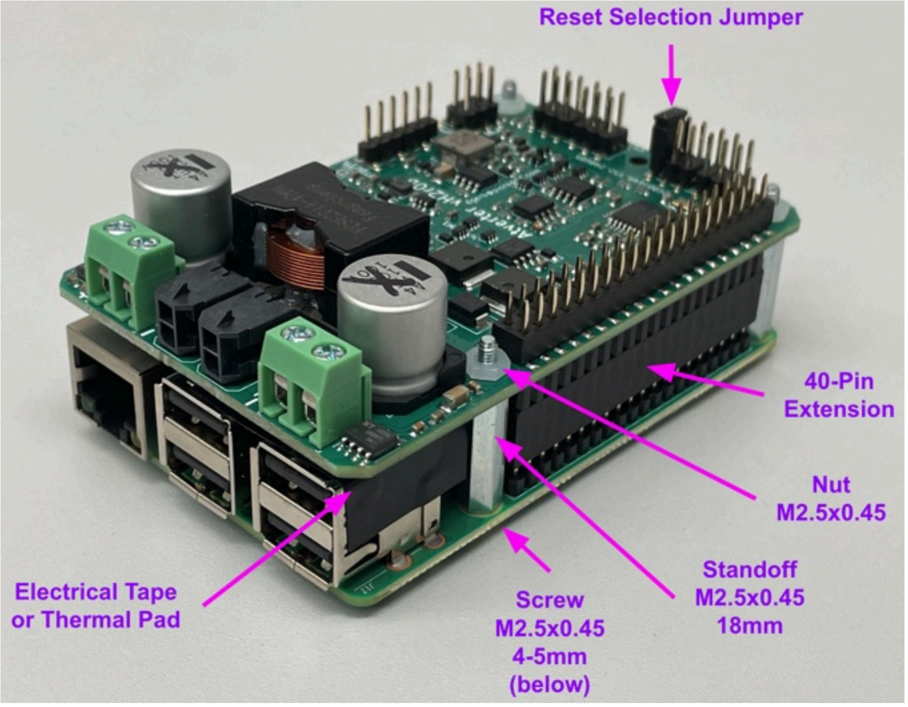
Mounting on a Raspberry Pi

The Atverter is compatible with a Raspberry Pi, enabling remote tertiary control and firmware updates. We can provide an optional mounting kit, which includes:

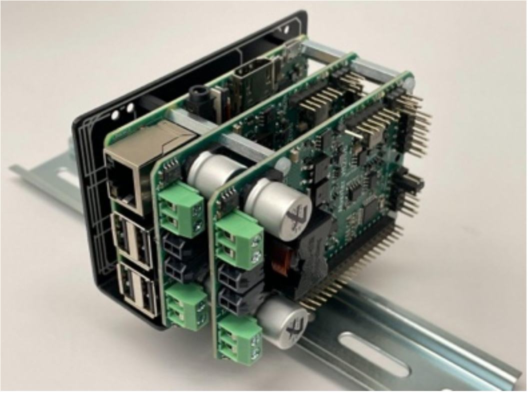
- 4x Standoff: M2.5x0.45, 18mm height
- 4x Screw: M2.5x0.45, 4-5mm height
- 4x Nut: M2.5x0.45 (can optionally can use a 4-6mm standoff for ease of fastening)
- 2x Jumper, 2.54mm pitch, 1x2
- 1x 40-pin Header, 2.54mm pitch, 2x20
- 1x Thermal Pad

The Atverter can be mounted on the Raspberry Pi through the following steps:

1. Slide the Jumper onto the Reset and CS pins that correspond to the intended Pi GPIO
2. Slide the male side of the 40-pin Header through the Atverter's 40-pin receptacle from the bottom
3. Fasten each Standoff to the four corner mounting holes of the Atverter with the Nuts. To mount on a Raspberry Pi Zero, use the mounting hole between the reset and CS header
4. Stick the Thermal Pad or some electrical tape on top of the right (Pi 3B, 5) or left (Pi 4) most USB port, so as to ensure no electrical contact with the Atverter's screw terminal
5. Slide the female side of the 40-pin Header onto the Raspberry Pi
6. Fasten the Raspberry Pi to the Standoffs from the bottom using the Screws



DIN Rail Mounting



DIN rails are the industry standard for mounting electronics in enclosures and cabinets. Many types of Raspberry Pi mounting systems will work. We recommend the Chunzehui mounting system. The Din-R-Plate can also be used, though may require hole tapping.

To stack multiple Atverters on a Raspberry Pi, refer to the guide for Mounting on a Raspberry Pi, above. Depending on the application, we may recommend an additional DIN rail mounting system on the opposite side of a large stack of Atverters.

Pi Case Mounting

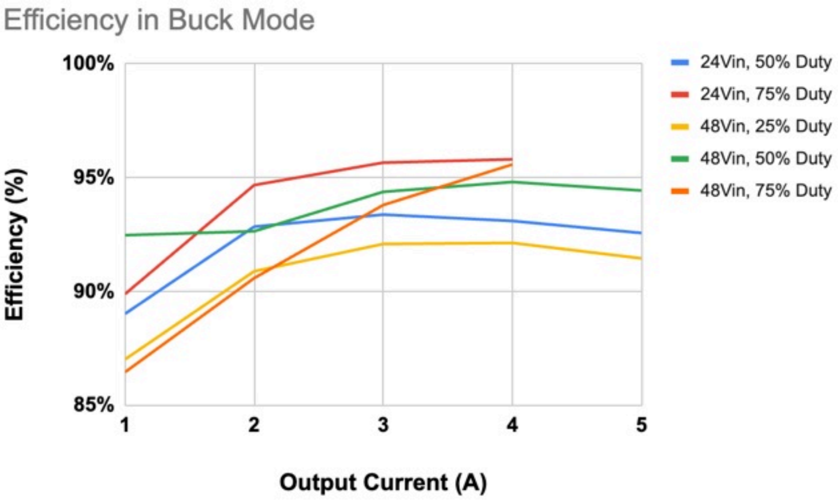
For standalone applications, the Atverter can be housed in one of several existing Raspberry Pi fan cases. We have tested and recommend the GeeekPi Pi 4 Case case for a single Atverter. Follow the product's mounting instructions, treating the Atverter like a Pi. Connect the fan's black wire to the GND pin of the FTDI/TTL header. Solder a 1x1 header into the Aux VCC hole and attach the red wire.

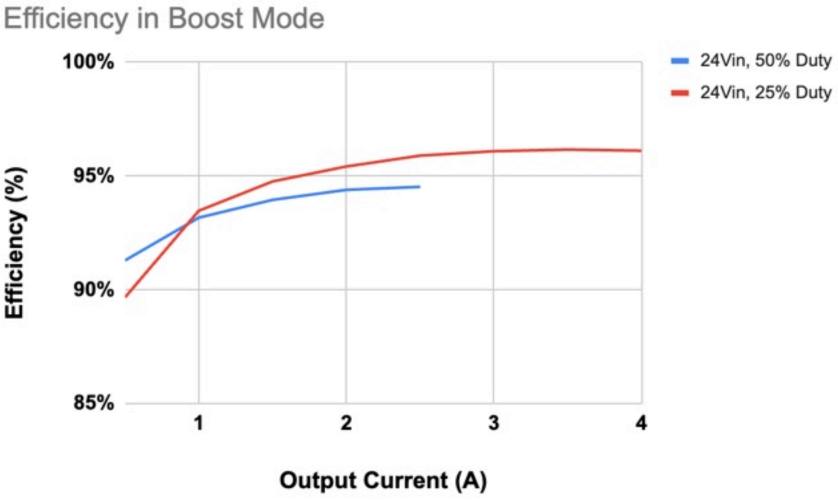
The GeeekPi Pi 4 High Case can house an Atverter mounted on a Pi, though we had to use nuts to raise the roof for the fan header. It is also possible this case may only work with the screw terminals.

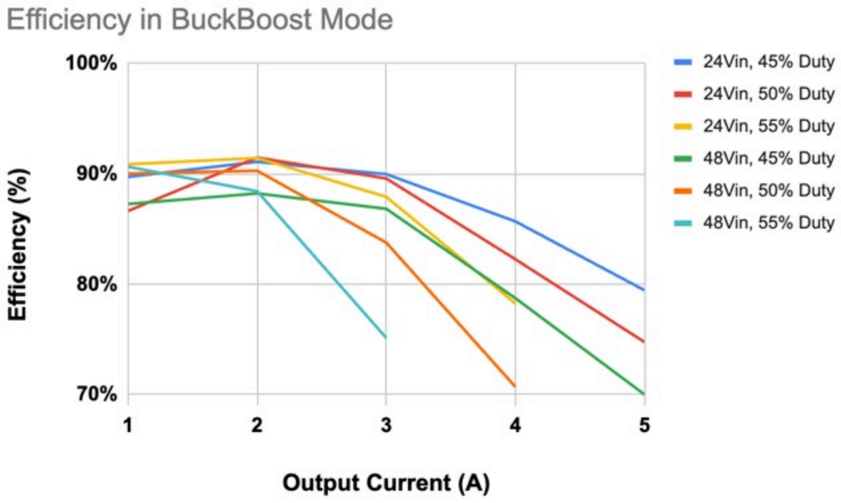
Several other fan cases may also work, but have not been tested. We expect the GeeekPi Pi 5 Case and GeeekPi Pi 5 High Case might work with some modifications to the fan cable. We also expect the iUniker Pi 4 Case and MazerPi Pi 4 Case may work with some modifications to the case itself.

Efficiency

The efficiency curves are shown below for Atverter, operating in buck, boost, and buck-boost modes with 24V or 48V input and several relevant duty cycles. Data was measured operating the Atverter with open-loop control at a fixed duty cycle, running a modified version of the example 2_OpenLoopBuckBoost. Power consumed by the controller is accounted for.

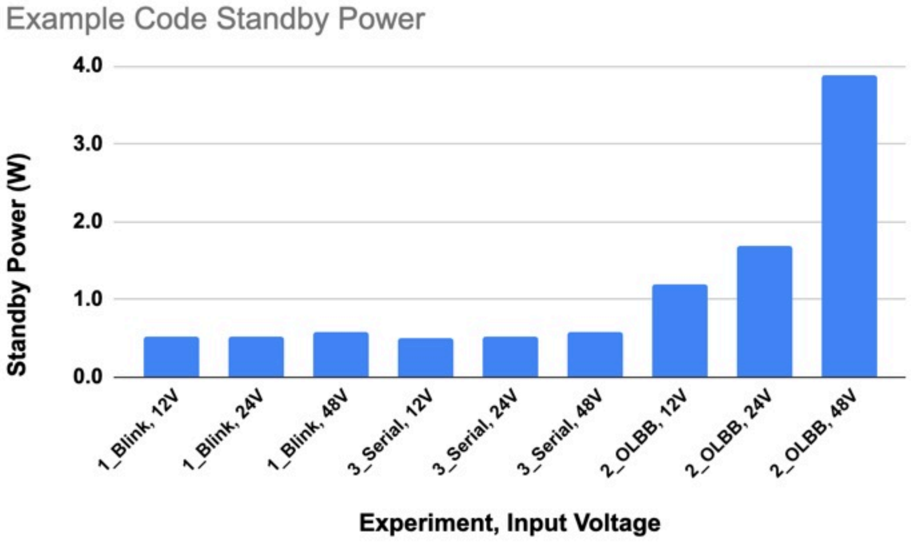






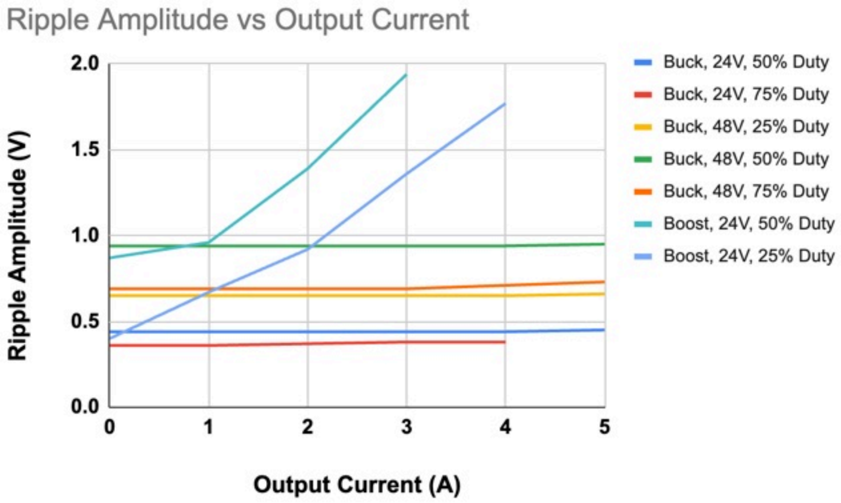
Standby Power

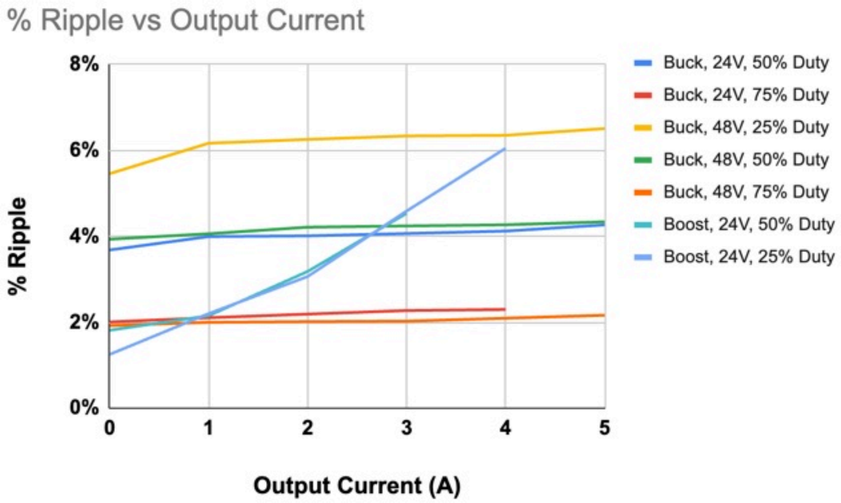
The standby power varies depending on what code is running. We measure standby power, running several example programs with 12V, 24V, and 48V input. Initializing the PWM greatly increases standby consumption.



Output Ripple

The ripple amplitude and percent ripple are given below for buck and boost operation at 24V and 48V input with several relevant duty cycles. Data was measured operating the Atverter with open-loop control at a fixed duty cycle, running a modified version of the example 2_OpenLoopBuckBoost.


Ripple Amplitude vs Output Current


% Ripple vs Output Current