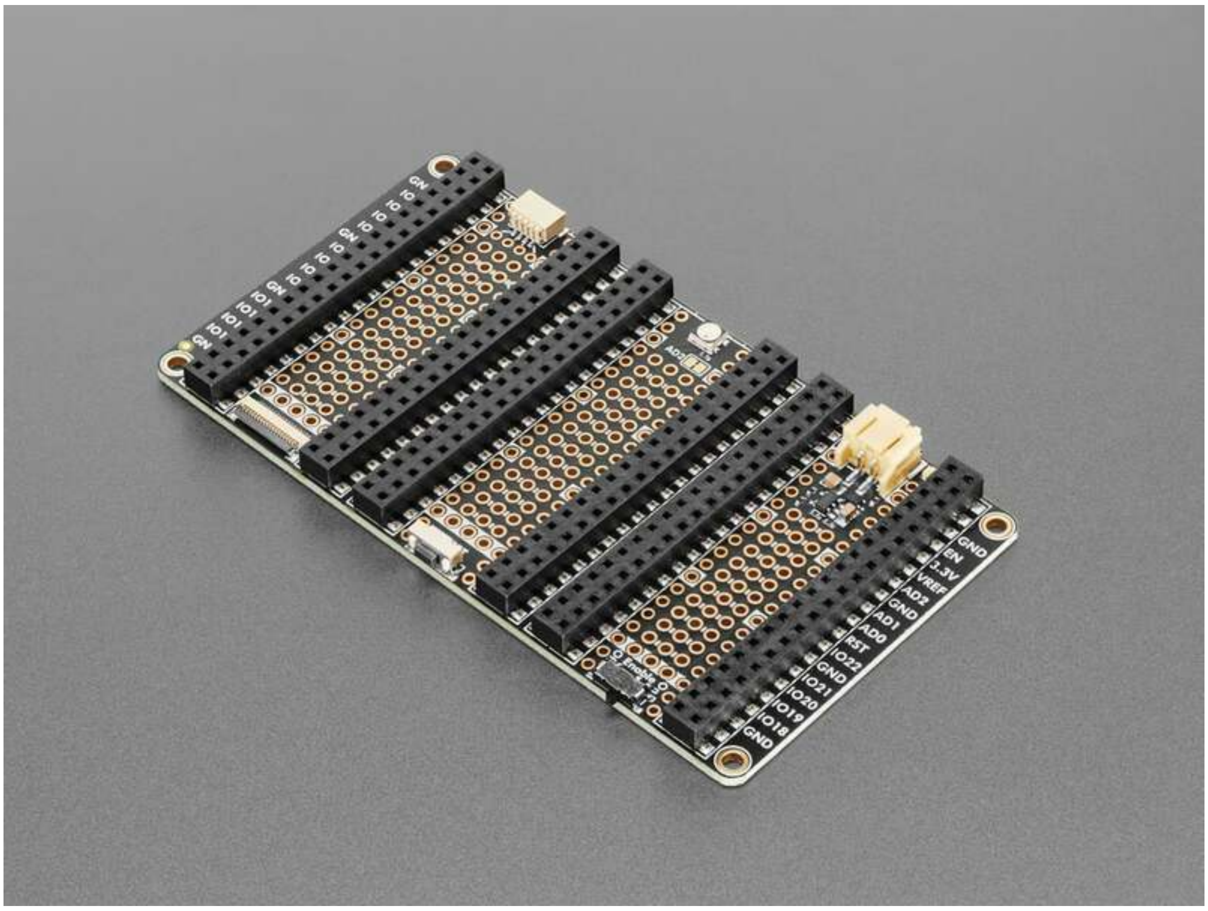
Adafruit Proto Tripler PiCowbell

Created by Liz Clark

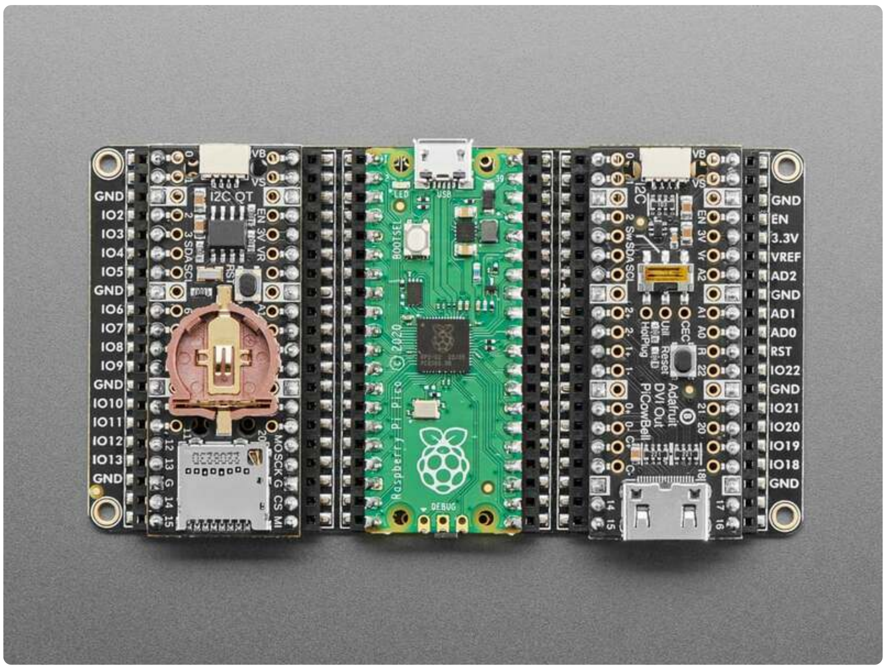


https://learn.adafruit.com/adafruit-proto-tripler-picowbell

Last updated on 2024-06-03 04:01:03 PM EDT

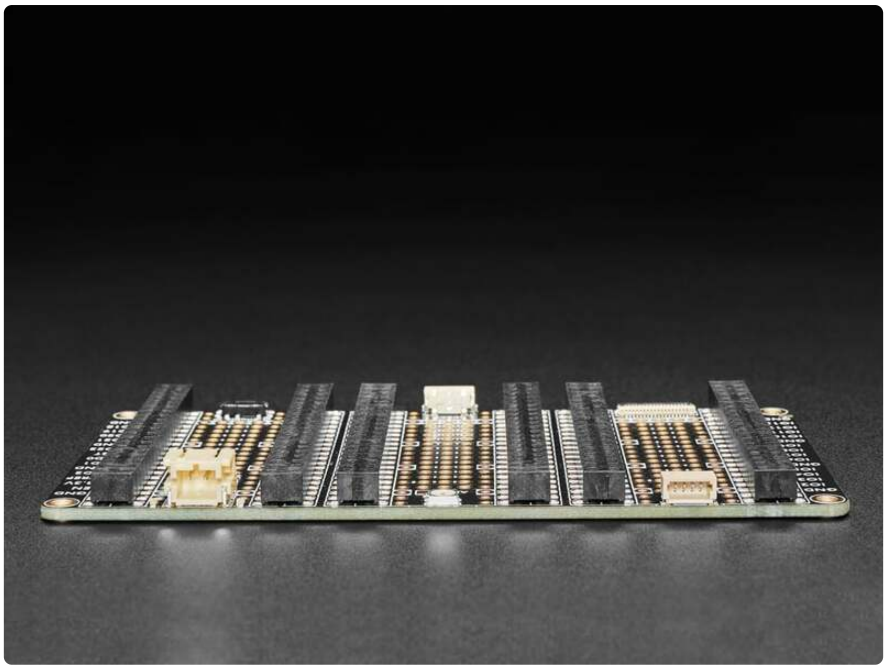
Table of Contents

Overview



The **Adafruit Proto Tripler PiCowBell** is intended to be treated like a mini solder-less proto plate to simplify programming and sensor or display connectivity for your Raspberry Pi Pico board. Reset button? Yes! STEMMA QT / Qwiic connector for fast I2C? Indeed. Battery with recharging and on/off switch? Affirmative. Built in NeoPixel? Bien sur! EYE SPI connector? Truly! All plug-and-play, so no soldering necessary when used with a Pico H (http://adafru.it/5525) or Pico WH (http://adafru.it/5544)? Here you go!

The Tripler is like if we used a 'Clone Tool' twice on the Proto Under Plate PiCowbell (http://adafru.it/5905) - you get three slots side-by-side, which means you can easily add two 'bell accessories to your Pico or Pico W without soldering on any stacking headers.



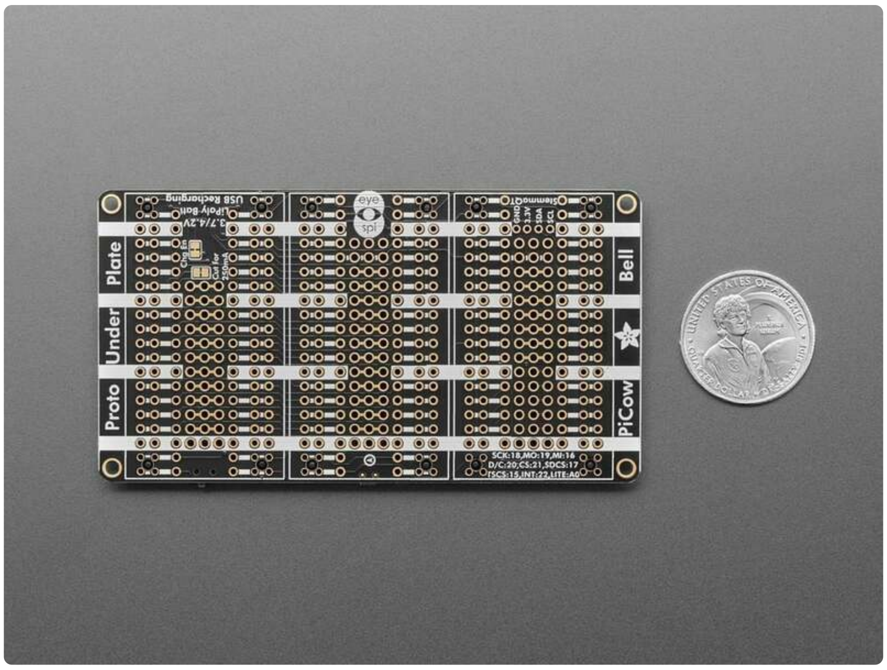
LiPoly Battery Charging Support

The Tripler also gives you Lipoly/LiIon support circuitry to take your Pico project on the go. Use any 3.7V/4.2V single-cell Lithium battery with a JST 2-PH connector in the correct polarity. The battery will automagically charge when the Pico is plugged into the USB and switch over seamlessly to battery when USB power is removed.

By default, the charge rate is 500mA for use with 500mAh+ sized batteries. You can cut a jumper to reduce the charge rate to 250mA if using 250mAh to 500mAh battery. So you don't have to unplug the battery often, a slide switch is connected to the Pico Enable pin, so you can disable the Pico's 3.3V power supply completely - this is as close as we can get to 'turning off' the Pico.

Alkaline/NiMH Battery Pack Support

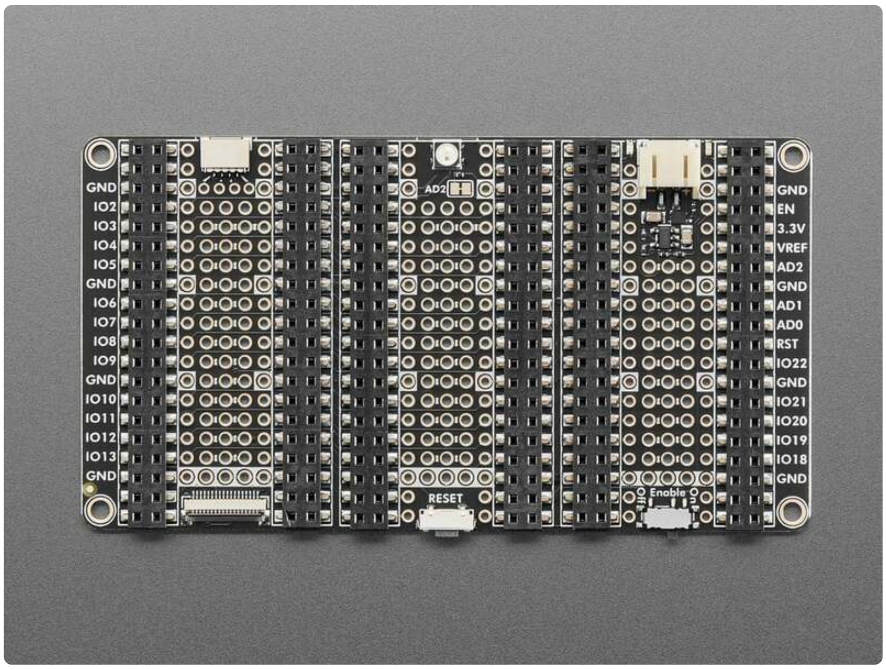
You can also cut a different jumper to disconnect the charger completely from the battery, which means you can use 3xAA (http://adafru.it/3287) or 3xAAA battery packs (http://adafru.it/727) for alkaline or NiMH battery usage. Of course, you will need an external NiMH charger in that case.



STEMMA QT + EYESPI Connectors

Add sensors and displays in an instant with a 4-Pin JST SH connector that is Qwiic / STEMMA QT compatible for just about any I2C sensor or device we stock. We do not have I2C pullups on the board, but your Qwiic/QT breakout board or accessory likely already has them onboard. If using the Philhower Arduino core, the Wire peripheral is already set up to use IO4 and IO5. If using CircuitPython or MicroPython, you'll need to let the code know to look at 4+5 for SDA+SCL pins.

For colorful high-resolution output, connect any of our EYESPI displays with an 18-pin FPC cable. We connect most of the pins for SPI displays with SD card and touchscreen support. Note that we didn't wire through the Busy or Reset pins so E-Ink displays aren't recommended (they sometimes require those pins for deep-sleep and wakeup)

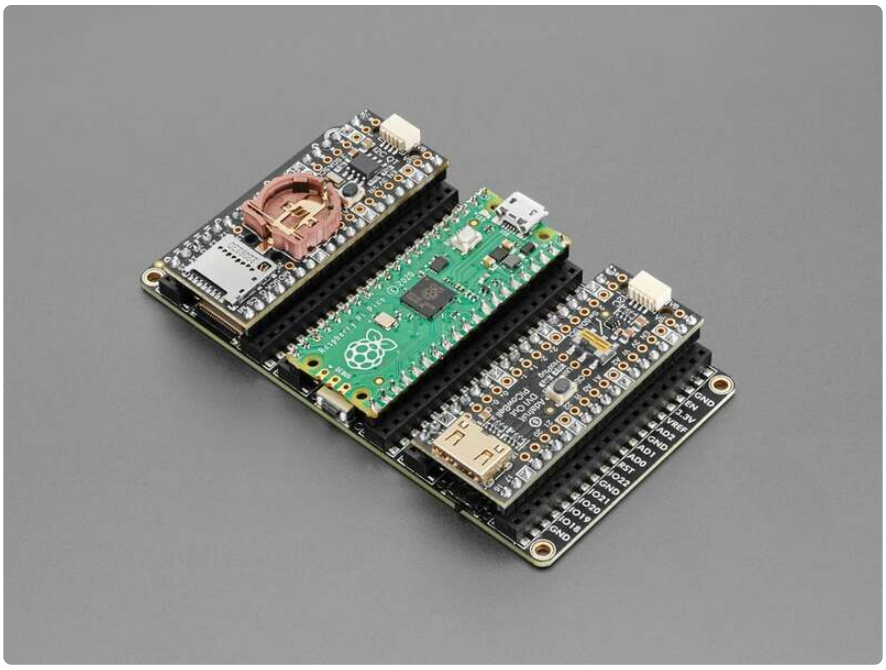


Double-row Socket Headers

This board has double-row sockets already soldered on; you can plug in your Pico and 'Bell boards directly and access a prototyping area underneath and a second row of socket pins. The second row means you can connect wires just by poking them into the header, either directly to LEDs or buttons or to jumper to a breadboard. So you don't have to look up or count pins; each socket is nicely labeled. There are also 4 mounting holes for easy attachment to an underplate.

The double-row socket headers we use are 'hollow,' meaning you can connect through the back if desired. Specifically, you can use Pico Stacking Headers to plug into some other device that is expecting Pico pins. Also, in theory, if you soldered pin headers on the 'wrong side' of a Pico, you could plug it up through the bottom.

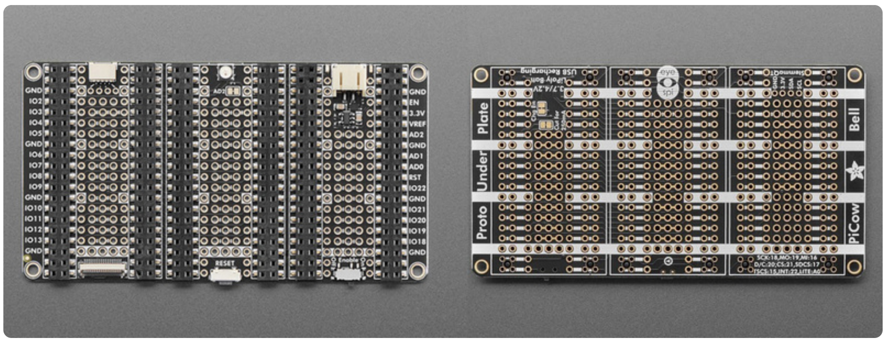
If you're not mounting this board to something else, we recommend picking up a set of rubber feet (http://adafru.it/550) to protect your desk.



Features:

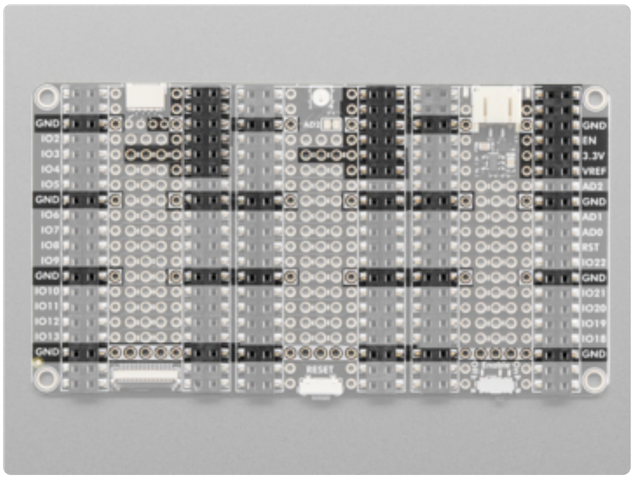
- **Three 2x20 slim socket headers** - plug in your Pico and desired 'Bell and have an extra row of sockets for each pin!
- **JST PH connector** for LiIon/LiPoly or Alkaline/NiMH battery usage - be sure to cut the "disable charge" jumper to use Alkaline/NiMH batteries.
- **LiPoly/LiIon charging circuitry** for any 3.7V/4.2V nominal 1-cell battery. Default rate is 500mA, can be set to 250mA by cutting the charge-rate jumper on bottom. Two indicator LEDs let you know if battery is charging (orange) or complete (green)
- **Slide switch** connected to the Pico Enable pin, which will disable the 3.3V power supply.
- **Reset button** that sticks out at the end
- **JST SH connector for I2C / Stemma QT / Qwiic** connection. Or can use it for plain GPIO wiring if you don't have any I2C devices to attach. Provides 3V, GND, IO4 (SDA), and IO5 (SCL) There is an extra set of 4 breakout holes next to the JST SH if you want more I2C connections or want to re-assign the I2C port.
- **3 hole-connected strips are** in the center areas. You can cut the traces between the holes, but they're intended to be treated like a mini-mini breadboard
- **Nearly every pad on the Pico has a duplicate hole pad** next to it for solder-jumpering
- **The ground pads have white silkscreen rectangles** to easily identify, plus one long ground strip near the reset button
- **One long strip of connected holes for 3.3V power**
- Gold-plated pads for easy soldering

Pinouts



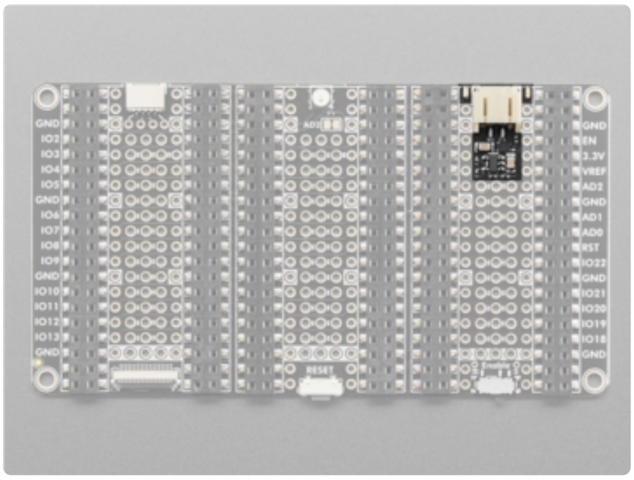
Check that your battery has the correct polarity for the Tripler PiCowbell! Otherwise you could damage or destroy the PiCowbell and anything plugged into it!

Power



VB (VBUS) - This is the micro-USB input voltage, connected to the micro-USB port on the Raspberry Pi Pico. It is nominally 5V.
VS (VSYS) - This is the main system input voltage. It can range from 1.8V to 5.5V and is used to generate the 3.3V needed for the RP2040 and the GPIO pins.
EN (3V3_EN) - This connects to the enable pin on the Raspberry Pi Pico, and is pulled high (to VSYS) via a 100kΩ resistor. The **slide switch** at the bottom right edge of the Tripler is connected to this pin.
3.3V - This is the 3.3V output from the Raspberry Pi Pico. There is a long strip of connected holes for 3.3V power, noted by a line of white on the board silk.
VREF (ADC_VREF) - This is the ADC power supply and reference voltage. It is generated on the Raspberry Pi Pico by filtering the 3.3V supply. It can be used with an external reference when ADC performance is required.
GND - This is the common ground for power and logic. All **GND pins are highlighted in white rectangles on the silk.**

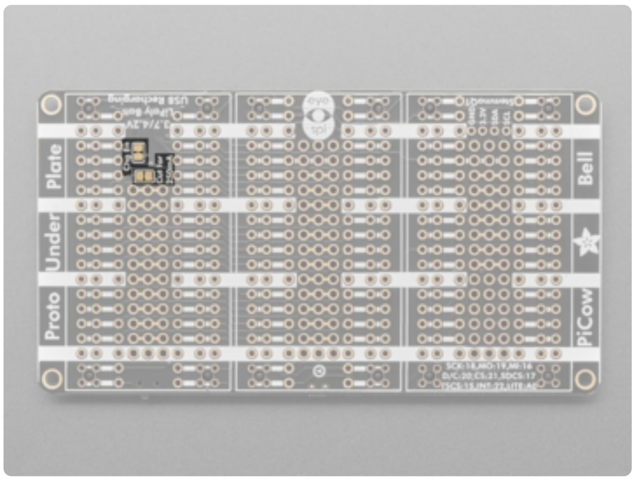
The Tripler also gives you Lipoly/LiIon support circuitry:



LiPoly connector/charger - You can plug in any 500mAh or larger 3.7V/4.2V single-cell battery into the **JST 2-PH port with the correct polarity** to both power your Pico and charge the battery. The battery will charge from the USB power when USB is plugged in. If the battery is plugged in and USB is plugged in, the Pico will power itself from USB and it will charge the battery up.

Orange Indicator LED - To the right of the JST 2-PH port is the orange charge indicator LED. When the battery is charging, the orange LED will be lit. When charging is complete, the LED will turn off. If there's no battery plugged in while the board is powered by USB, the LED may be dimly lit - this is expected!

Green Indicator LED - To the left of the JST 2-PH port is the green charge indicator LED. While a battery is charging, the green LED will be off. When the battery is fully charged, or If there's no battery plugged in, the green LED will be lit.

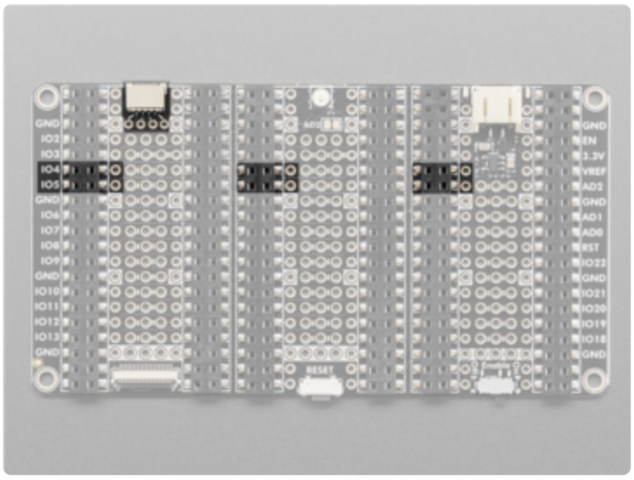


Charge Rate Jumper - On the back of the Tripler is the charge rate jumper. It is labeled **Cut for 250mA** on the board silk. By default the charge rate is 500mA for use with 500mAh+ sized batteries. You can cut a jumper to reduce the charge rate to 250mA if using 250mAh to 500mAh battery. **Batteries that are smaller than 250mAh are not supported.**

Alkaline/NiMH Battery Pack Jumper - On the back of the Tripler is the alkaline/NiMH battery pack jumper. It is labeled **Chg En** on the board silk. If you cut this jumper, it will disconnect the charger circuitry completely from the JST 2-PH port. This means you can use 3xAA (http://adafru.it/3287) or 3xAAA battery packs (http://adafru.it/727) for alkaline or NiMH battery usage.

If your battery is smaller than 500mAh, you need to cut the charge rate jumper. Batteries that are smaller than 250mAh are not supported.

I2C Logic



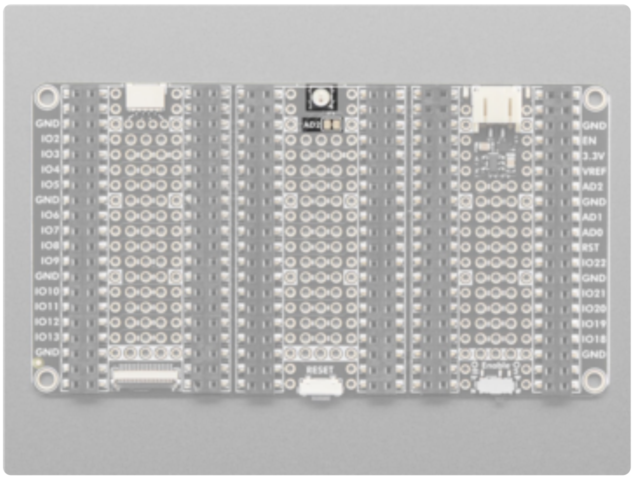
SCL - I2C clock pin on the PiCowBell. It is connected to your Pico I2C clock line, which is **GPIO5**. This connection is shared with the STEMMA QT port and the EYESPI connector on the board.

SDA - I2C data pin on the PiCowBell. It is connected to your microcontroller I2C data line, which is **GPIO4**. This connection is shared with the STEMMA QT port and the EYESPI connector on the board.

STEMMA (https://adafru.it/Ft4)QT **(https://adafru.it/Ft4)** - These connectors allow you to connect to dev boards with STEMMA QT connectors or to other things with various associated accessories (https://adafru.it/JRA). The port is located on the end of the PiCowBell.

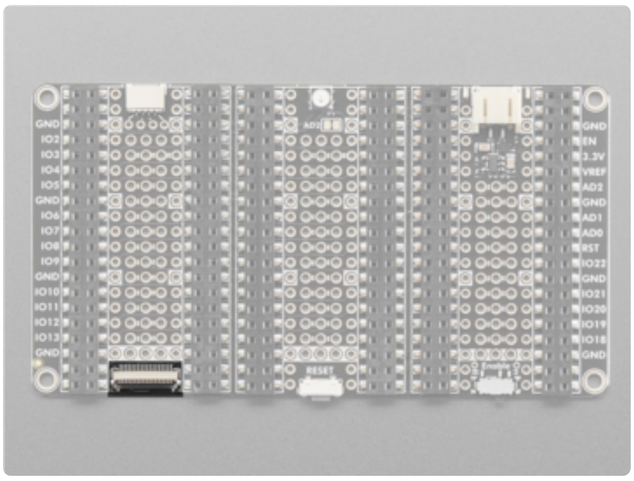
STEMMA QT Breakout Holes - There is an extra set of 4 breakout holes directly below the STEMMA QT port if you want more I2C connections or want to re-assign the I2C port.

NeoPixel and NeoPixel Jumper



At the top of the board in the middle section is one **RGB NeoPixel LED**. Its data pin is connected to GPIO **AD2**. There is a jumper below it labeled **AD2**. You can cut this jumper to disconnect the NeoPixel from **AD2**.

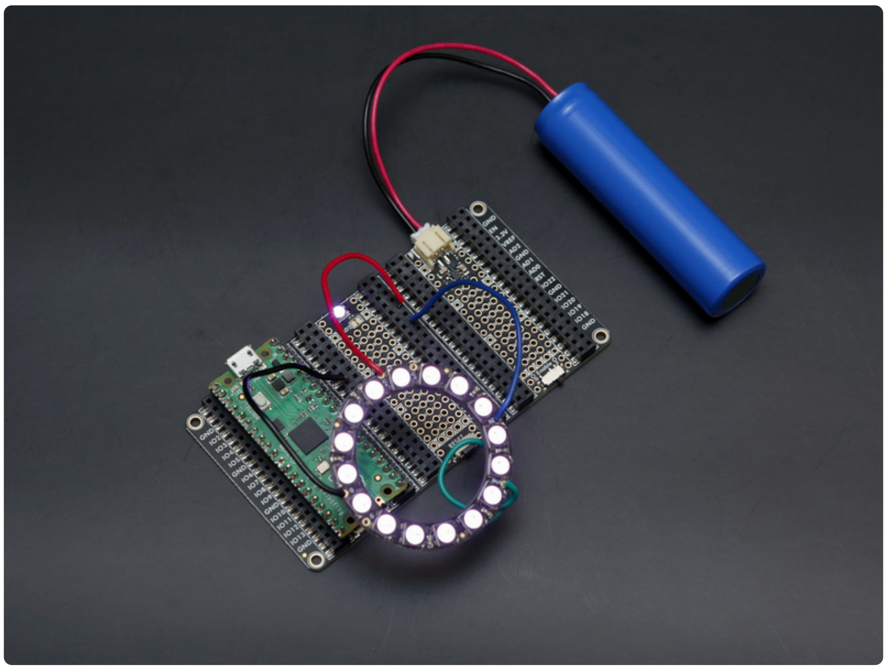
EYESPI Connector

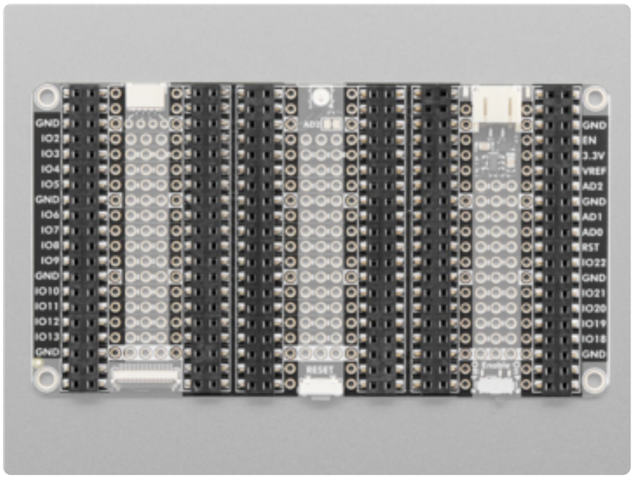


At the bottom of the board, opposite the STEMMA QT port, is the **EYESPI connector**. This connector allows you to connect EYESPI-compatible displays using an EYESPI cable, with no soldering or jumper wires needed. The EYESPI pins are connected to the following GPIO:

Vin - 3.3V
Backlight - AD0
GND - GND
MISO - GPIO16
MOSI - GPIO19
SCK - GPIO18
Display CS - GPIO21
DC - GPIO20
INT - GPIO22
SDA - GPIO4
SCL - GPIO5
Touchscreen CS - GPIO15
SD card CS - GPIO17

Socket Headers and Duplicate GPIO Hole Pads

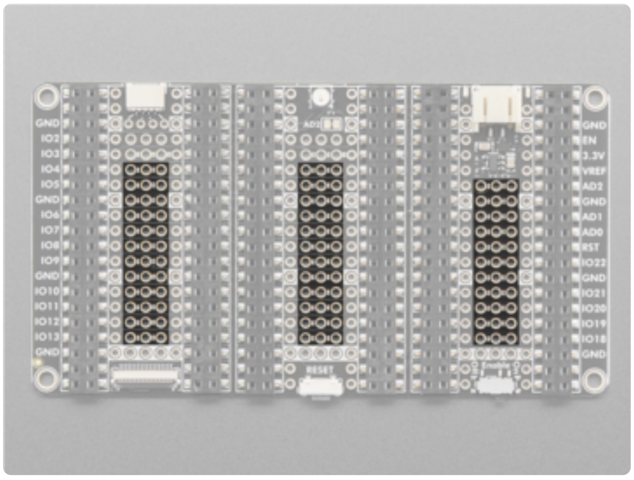




The PiCowbell has three sets of two 2x20 slim socket headers to plug in your Pico and have extra rows of sockets for each pin. On the second and third sets of headers you can plug in an accessory for your Pico. There are also duplicate hole pads next to each pin for solder-jumpering:

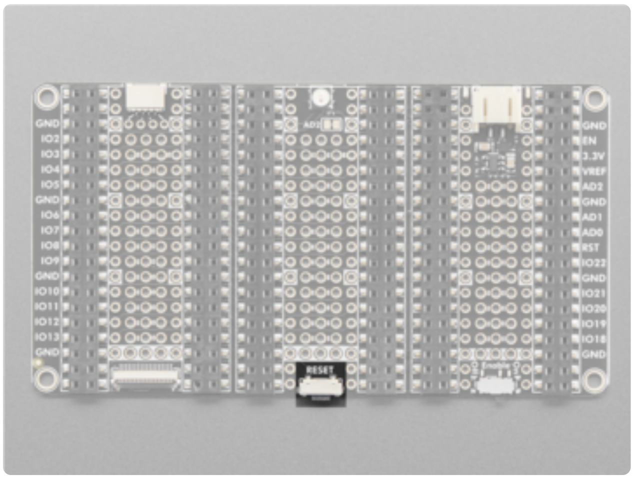
IO0-IO15, **IO16-IO22**, **Reset**, **A0-A2**, **VR**, **3V**, **EN**, **VS** and **VB**. Ground pins that have a duplicate hole pad are highlighted in white rectangles on the board silkscreen.

Proto Areas



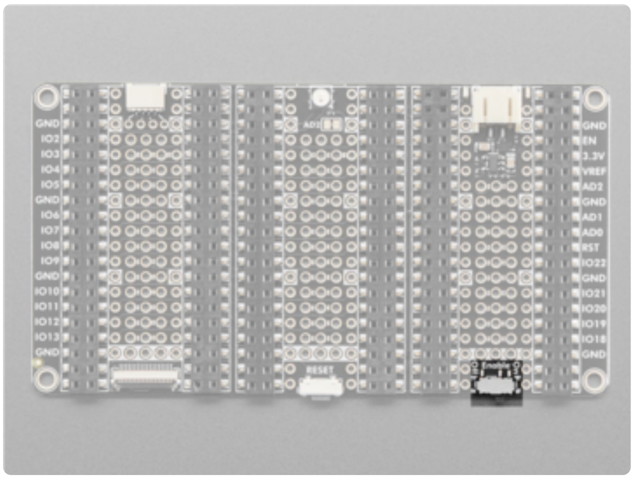
Between the sets of socket headers, you'll find the **proto areas**. These areas on the board are made up of 12 **3 hole-connected strips on the left** and **in the middle**, and 11 **3 hole-connected strips on the right**. You can cut the traces between the holes, but they're intended to be treated like mini-mini breadboards.

Reset Button



At the bottom edge of the board in the middle section is the reset button. It is routed to the **reset pin on the PiCowbell** and is labeled **RESET** on the board silk. You can press it to restart your program.

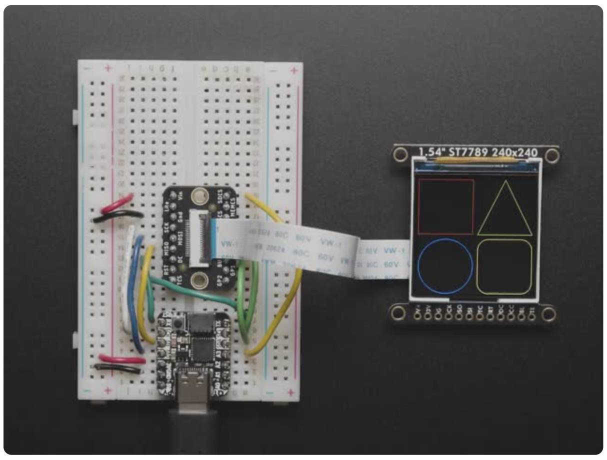
Enable Switch



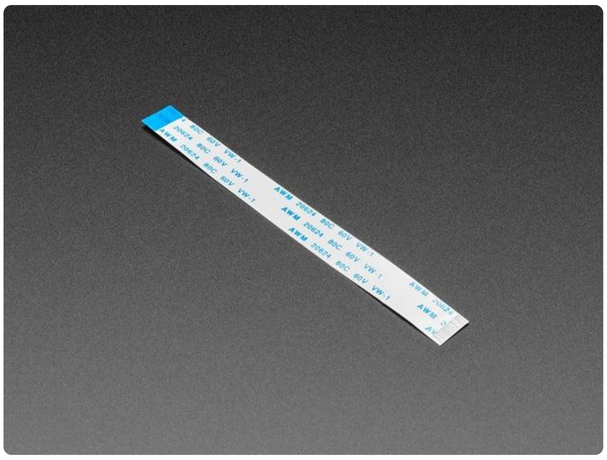
At the bottom right edge of the board is the enable switch. It is routed to the **enable pin on the PiCowbell** and is labeled **Enable** on the board silk. Moving the switch to the **Off** position will disable the 3.3V power supply. Moving the switch to the **On** position will enable the 3.3V power supply. The switch does not disconnect USB power. Disabling 3.3V is as close as we can get to 'turning off' the Pico.

CircuitPython

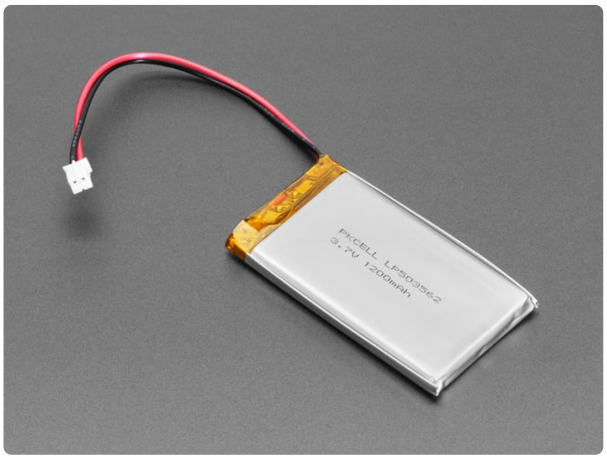
It's easy to use the **Proto Tripler PiCowbell** with CircuitPython to monitor the voltage of an attached lipoly battery with the analogio (https://adafru.it/19lE) core module and connect to an external display with the EYESPI connector and displayio (https://adafru.it/18KB) core module. These modules allow you to easily write Python code for accessing basic analog inputs and outputs and writing to a display.



Adafruit 1.54" 240x240 Wide Angle TFT LCD Display with MicroSD
We've been looking for a display like this for a long time - it's only 1.5" diagonal but has a high density 220 ppi, 240x240 pixel display with full-angle viewing. It...
https://www.adafruit.com/product/3787



EYESPI Cable - 18 Pin 100mm long Flex PCB (FPC) A-B type
Connect this to that when a 18-pin FPC connector is needed. This 25 cm long cable is made of a flexible PCB. It's A-B style which means that pin one on one side will match...
https://www.adafruit.com/product/5239



Lithium Ion Polymer Battery - 3.7v 1200mAh
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/258

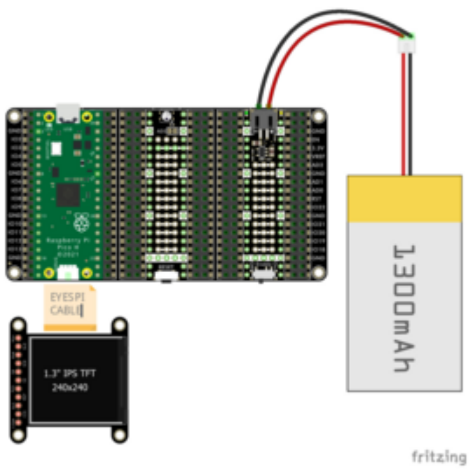
Raspberry Pi Pico ADC3 and VSYS

The Raspberry Pi Pico can use its internal ADC3 pin (GPIO29) to monitor the voltage on VSYS (https://adafru.it/19Ic). As a result, you can read the voltage currently being supplied to VSYS (aka the voltage from your battery) with this calculation in CircuitPython:

```
((ADC3 value * 3) * 3.3) / 65535
```

CircuitPython Microcontroller Wiring

Plug a Pico or Pico W into your Proto Tripler PiCowbell exactly as shown below. Then, plug in a supported lipoly battery to the PiCowbell JST 2-PH port. Connect an external display with an EYESPI connector to the EYESPI connector on the PiCowbell with an EYESPI cable. Here's an example of connecting a Pico to the PiCowbell with a lipoly battery and 1.54" 240x240 display.



Connect the Pico with plug headers into the Proto Tripler PiCowbell. It should be plugged in with the Pico USB port pointing towards the STEMMA QT port.

Then, plug in a lipoly battery with matching polarity to the JST 2-PH port on the PiCowbell.

Finally, attach the EYESPI display to the EYESPI connector with an EYESPI cable. For reference on plugging in an EYESPI cable, you can follow along with this Learn Guide page. (https://adafru.it/18eU)

Plugging in an EYESPI Cable

https://adafru.it/18eU

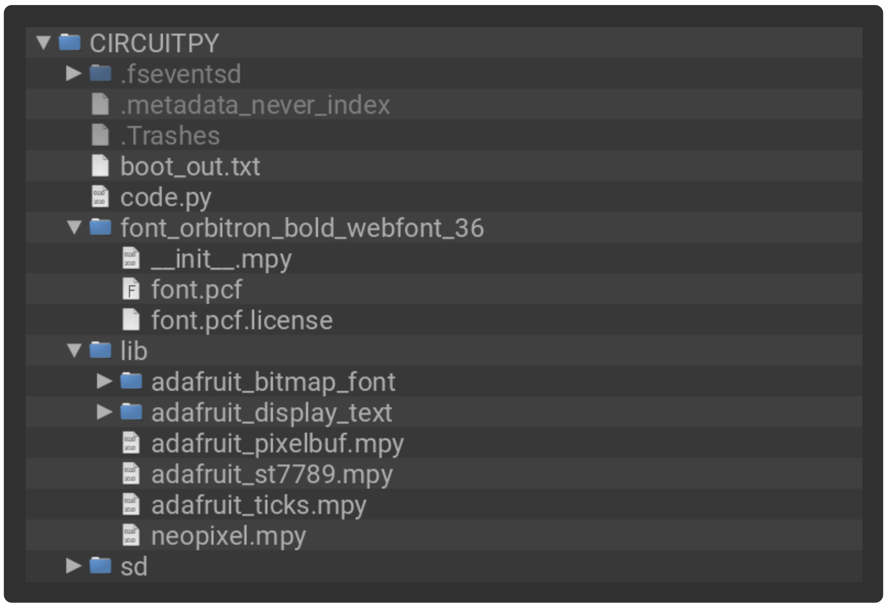
CircuitPython Usage

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_ST7789** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder**, **font folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and files:

- **adafruit_bitmap_font/**
- **adafruit_display_text/**
- **adafruit_pixelbuf.mpy**
- **adafruit_st7789.mpy**
- **adafruit_ticks.mpy**
- **neopixel.mpy**

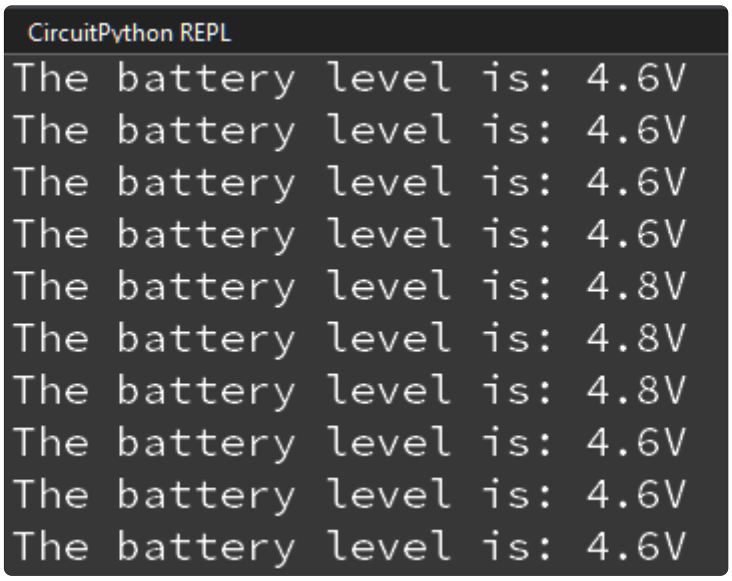


Example Code

Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!

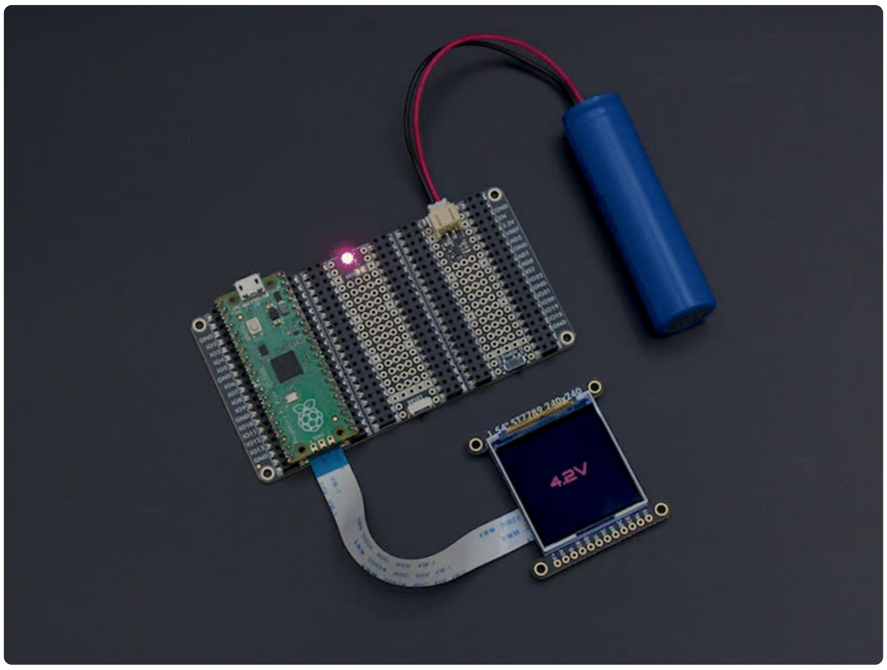
```
# SPDX-FileCopyrightText: Copyright (c) 2024 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import busio
import board
from analogio import AnalogIn
import adafruit_st7789
import displayio
import neopixel
from rainbowio import colorwheel
from adafruit_display_text import label
from adafruit_ticks import ticks_ms, ticks_add, ticks_diff
from font_orbitron_bold_webfont_36 import FONT as orbitron_font

displayio.release_displays()
spi = busio.SPI(clock=board.GP18, MOSI=board.GP19)
display_bus = displayio.FourWire(spi, command=board.GP20, chip_select=board.GP21,
reset=None)
display = adafruit_st7789.ST7789(display_bus, width=240, height=240, rowstart=80,
rotation=0)

group = displayio.Group()
text = label.Label(orbitron_font, text="0V", color=0xFF0000)
text.anchor_point = (0.5, 0.5)
text.anchored_position = (display.width / 2, display.height / 2)
group.append(text)
display.root_group = group

analog_in = AnalogIn(board.A3)

def get_vsys(pin):
    return ((pin.value * 3) * 3.3) / 65535

pixel_pin = board.A2
num_pixels = 1
pixel = neopixel.NeoPixel(pixel_pin, num_pixels,
                                   brightness=0.1, auto_write=True)
hue = 0
pixel.fill(colorwheel(hue))

bat_clock = ticks_ms()
bat_timer = 5000
neo_clock = ticks_ms()
neo_timer = 100

while True:
    if ticks_diff(ticks_ms(), bat_clock) >= bat_timer:
        print(f"The battery level is: {get_vsys(analog_in):.1f}V")
        text.text = f"{get_vsys(analog_in):.1f}V"
        text.color = colorwheel(hue)
        bat_clock = ticks_add(bat_clock, bat_timer)
    if ticks_diff(ticks_ms(), neo_clock) >= neo_timer:
        hue = (hue + 7) % 256
        pixel.fill(colorwheel(hue))
        neo_clock = ticks_add(neo_clock, neo_timer)
```

Every 5 seconds, the analog reading from **pin A3** on the Pico is passed to the `get_vsys()` function. This function calculates the voltage on VSYS and prints the reading to the serial monitor. The screenshot below shows a battery initially plugged in at 4.6V. The battery was unplugged, reading 4.8V from the USB power and then plugged back in, reading 4.6V.



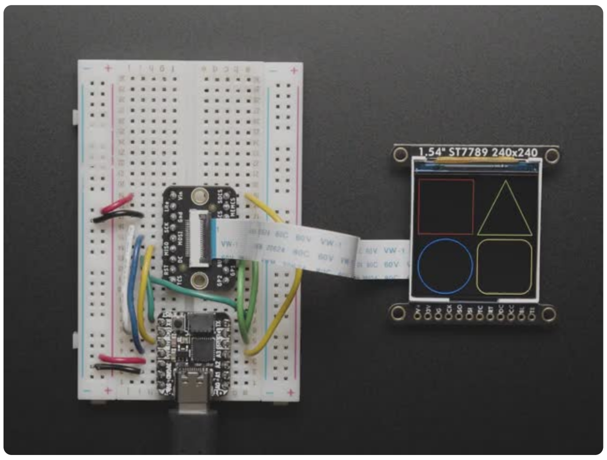
```
The battery level is: 4.6V
The battery level is: 4.6V
The battery level is: 4.6V
The battery level is: 4.6V
The battery level is: 4.8V
The battery level is: 4.8V
The battery level is: 4.8V
The battery level is: 4.6V
The battery level is: 4.6V
The battery level is: 4.6V
```

The display will show the voltage reading as well and will update every 5 seconds. The color of the font will change to match the current hue of the NeoPixel. The onboard NeoPixel will show a rainbow swirl animation.



Arduino

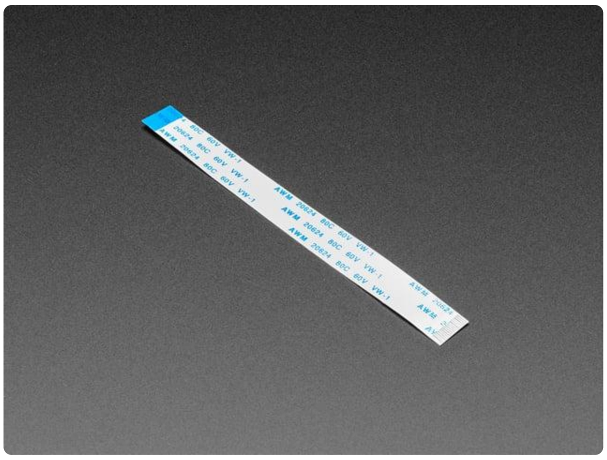
Using the Proto Tripler PiCowbell with Arduino to monitor and display the voltage of an attached lipoly battery involves plugging a Pico or Pico W board into the PiCowbell, attaching a compatible battery to the JST 2-PH port, attaching an EYESPI display to the EYESPI connector and running the provided example code.



Adafruit 1.54" 240x240 Wide Angle TFT LCD Display with MicroSD

We've been looking for a display like this for a long time - it's only 1.5" diagonal but has a high density 220 ppi, 240x240 pixel display with full-angle viewing. It...

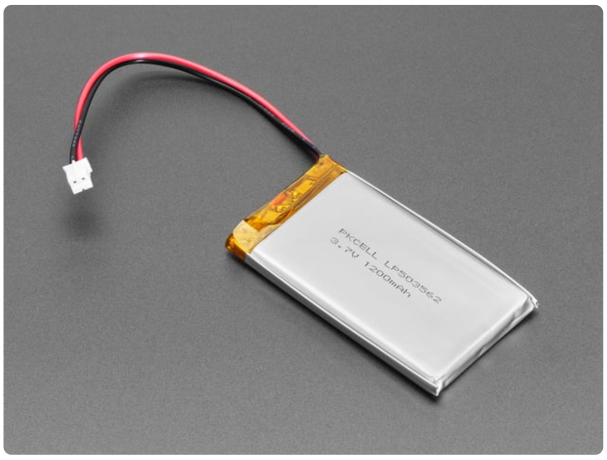
https://www.adafruit.com/product/3787



EYESPI Cable - 18 Pin 100mm long Flex PCB (FPC) A-B type

Connect this to that when a 18-pin FPC connector is needed. This 25 cm long cable is made of a flexible PCB. It's A-B style which means that pin one on one side will match...

https://www.adafruit.com/product/5239



Lithium Ion Polymer Battery - 3.7v 1200mAh

Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...

https://www.adafruit.com/product/258

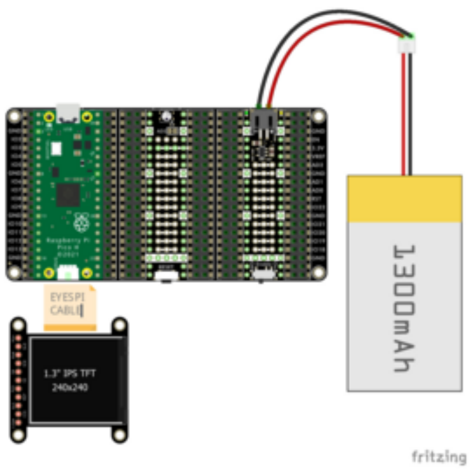
Raspberry Pi Pico ADC3 and VSYS

The Raspberry Pi Pico can use its internal ADC3 pin (GPIO29) to monitor the voltage on VSYS (https://adafru.it/19Ic). As a result, you can read the voltage currently being supplied to VSYS (aka the voltage from your battery) with this calculation in Arduino:

`((ADC3 value * 3) * 3.3) / 1023.0`

Wiring

Plug a Pico or Pico W into your Proto Tripler PiCowbell exactly as shown below. Then, plug in a supported lipoly battery to the PiCowbell JST 2-PH port. Finally, attach an EYESPI display to the EYESPI connector. Here's an example of connecting a Pico to the PiCowbell with a lipoly battery.



Connect the Pico with plug headers into the Proto Tripler PiCowbell. It should be plugged in with the Pico USB port pointing towards the STEMMA QT port.

Then, plug in a lipoly battery with matching polarity to the JST 2-PH port on the PiCowbell.

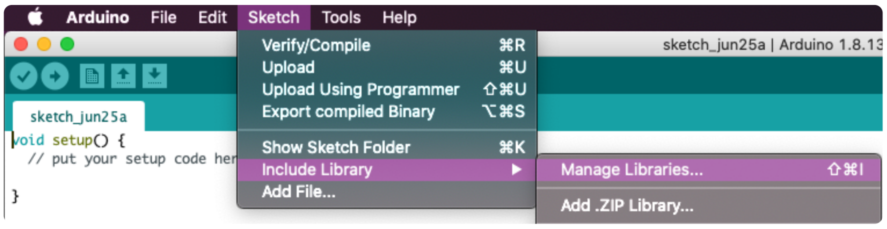
Finally, attach the EYESPI display to the EYESPI connector with an EYESPI cable. For reference on plugging in an EYESPI cable, you can follow along with this Learn Guide page. (https://adafru.it/18eU)

Plugging in an EYESPI Cable

https://adafru.it/18eU

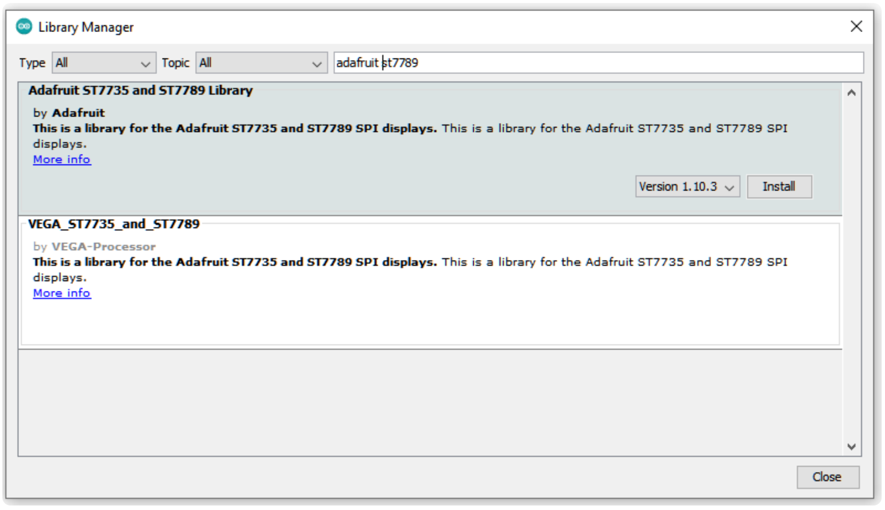
Library Installation

You can install the **Adafruit ST7735 and ST7789** library for Arduino using the Library Manager in the Arduino IDE.



```
void setup() {
  // put your setup code her
}
```

Click the **Manage Libraries ...** menu item, search for **Adafruit_ST7789**, and select the **Adafruit ST7735 and ST7789** library:


Library Manager

Type All Topic All adafruit st7789

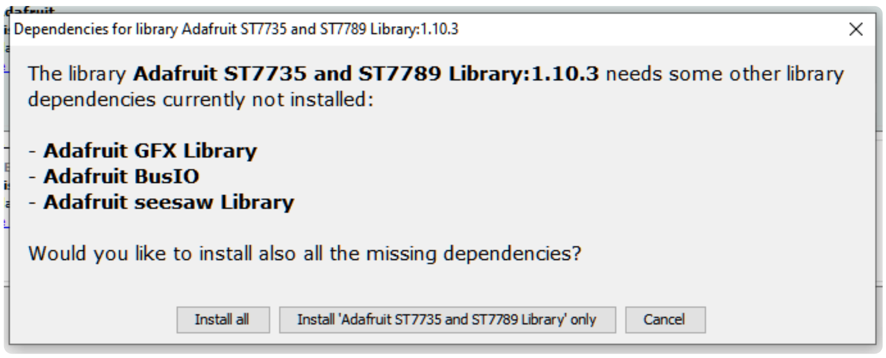
Adafruit ST7735 and ST7789 Library
by **Adafruit**
This is a library for the Adafruit ST7735 and ST7789 SPI displays. This is a library for the Adafruit ST7735 and ST7789 SPI displays.
More info

Version 1.10.3 Install

VEGA_ST7735_and_ST7789
by **VEGA-Processor**
This is a library for the Adafruit ST7735 and ST7789 SPI displays. This is a library for the Adafruit ST7735 and ST7789 SPI displays.
More info

Close

If asked about dependencies, click "Install all".


Dependencies for library Adafruit ST7735 and ST7789 Library:1.10.3

The library **Adafruit ST7735 and ST7789 Library:1.10.3** needs some other library dependencies currently not installed:

- **Adafruit GFX Library**
- **Adafruit BusIO**
- **Adafruit seesaw Library**

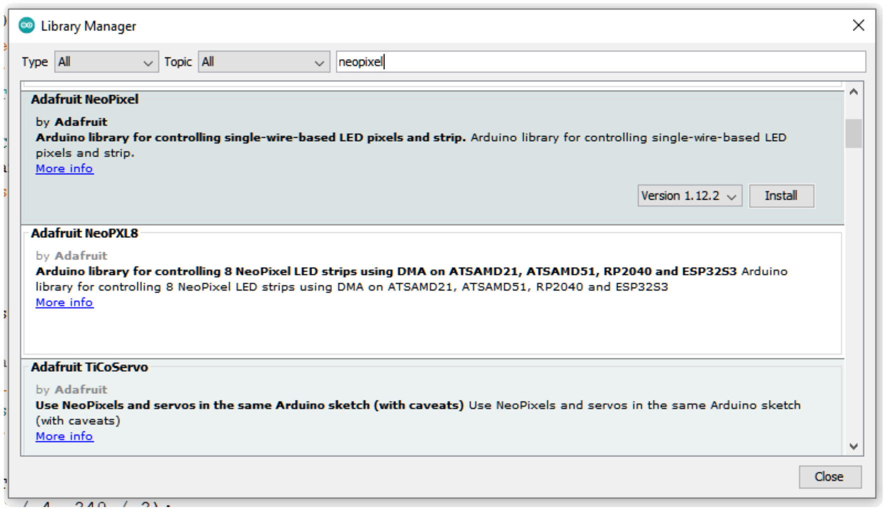
Would you like to install also all the missing dependencies?

Install all | Install 'Adafruit ST7735 and ST7789 Library' only | Cancel

If the "Dependencies" window does not come up, then you already have the dependencies installed.

If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Then, search for **Adafruit NeoPixel**, and select the **Adafruit NeoPixel** library. This library has no additional dependencies.



Adafruit NeoPixel
by Adafruit
Arduino library for controlling single-wire-based LED pixels and strip. Arduino library for controlling single-wire-based LED pixels and strip.
More info

Adafruit NeoPXL8
by Adafruit
Arduino library for controlling 8 NeoPixel LED strips using DMA on ATSAMD21, ATSAMD51, RP2040 and ESP32S3 Arduino library for controlling 8 NeoPixel LED strips using DMA on ATSAMD21, ATSAMD51, RP2040 and ESP32S3
More info

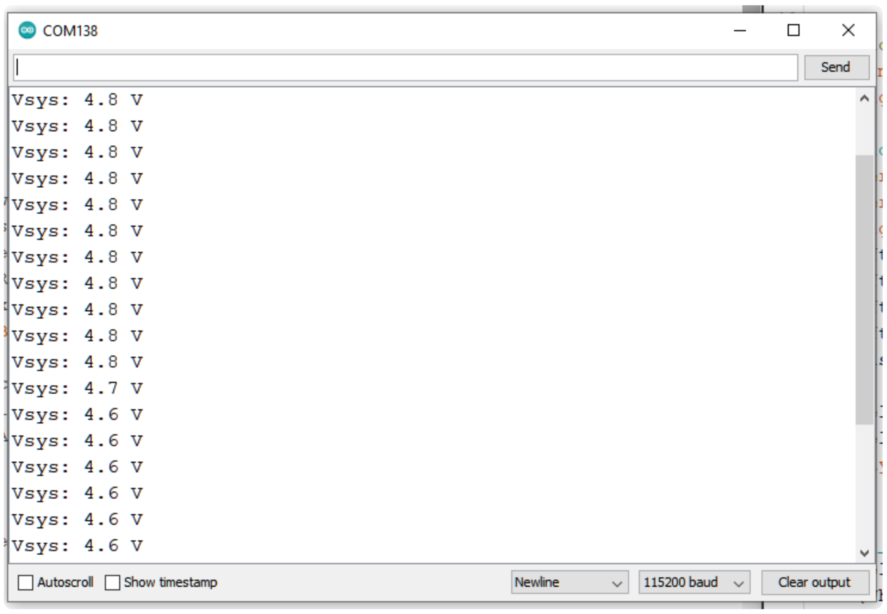
Adafruit TiCoServo
by Adafruit
Use NeoPixels and servos in the same Arduino sketch (with caveats) Use NeoPixels and servos in the same Arduino sketch (with caveats)
More info

The NeoPixel library has no additional dependencies.

Example Code

```
// SPDX-FileCopyrightText: 2024 Liz Clark for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#include <Adafruit_NeoPixel.h>
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_ST7789.h> // Hardware-specific library for ST7789
#include <SPI.h>
#include <Fonts/FreeSansBold24pt7b.h>

#define LED    LED_BUILTIN
#define TFT_CS        21
#define TFT_RST       -1
#define TFT_DC        20
#define NEO_PIN       A2

uint32_t Wheel(byte WheelPos);

Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_RST);

Adafruit_NeoPixel pixel = Adafruit_NeoPixel(1, NEO_PIN, NEO_GRB + NEO_KHZ800);

unsigned long lastMillis = 0;
uint8_t j = 0;

void setup() {
  Serial.begin(115200);
  //while (!Serial)  delay(1);  // wait for serial port
  tft.init(240, 240);
  pinMode(LED, OUTPUT);
  delay (100);
  Serial.println("PiCowbell Tripler Demo");
  tft.setFont(&FreeSansBold24pt7b);
  pixel.begin();
  pixel.setBrightness(50);
  pixel.show(); // Initialize all pixels to 'off'
  lastMillis = millis();

}

void loop() {
  if (millis() > (lastMillis + 5000)) {
    digitalWrite(LED, HIGH);
    // get the on-board voltage
    float vsys = analogRead(A3) * 3 * 3.3 / 1023.0;
    Serial.printf("Vsys: %0.1f V", vsys);
    Serial.println();
    digitalWrite(LED, LOW);
    tft.fillScreen(ST77XX_BLACK);
    tft.setCursor(240 / 4, 240 / 2);
    tft.setTextColor(ST77XX_WHITE);
    tft.printf("%0.1f V", vsys);
    lastMillis = millis();
  }
  pixel.setPixelColor(0, Wheel(j++));
  pixel.show();
  delay(20);
}

uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
    return pixel.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  }
  if(WheelPos < 170) {
    WheelPos -= 85;
    return pixel.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
  WheelPos -= 170;
  return pixel.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
}
```

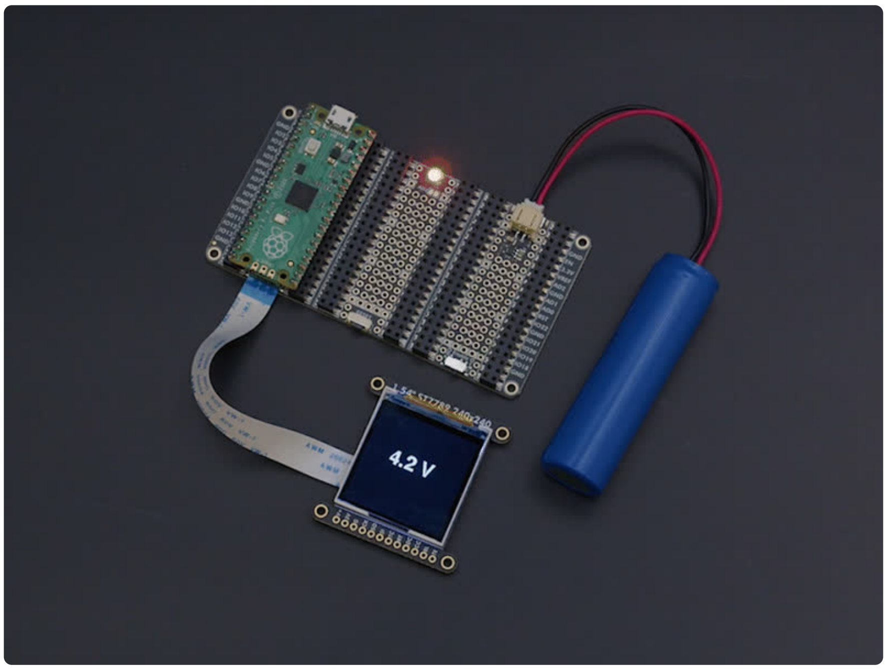
Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the voltage calculation printed to the monitor. In the screenshot below, the reading shows 4.8V from the USB power. Then, a battery was plugged in and was providing a reading of approximately 4.6V.



```
Vsys: 4.8 V
Vsys: 4.8 V
Vsys: 4.8 V
Vsys: 4.8 V
Vsys: 4.8 V
Vsys: 4.8 V
Vsys: 4.8 V
Vsys: 4.8 V
Vsys: 4.8 V
Vsys: 4.8 V
Vsys: 4.8 V
Vsys: 4.7 V
Vsys: 4.6 V
Vsys: 4.6 V
Vsys: 4.6 V
Vsys: 4.6 V
Vsys: 4.6 V
Vsys: 4.6 V
```

Autoscroll: unchecked; Show timestamp: unchecked; Newline; 115200 baud; Clear output

The display will show the voltage reading as well and will update every 5 seconds. The onboard NeoPixel will show a rainbow swirl animation.

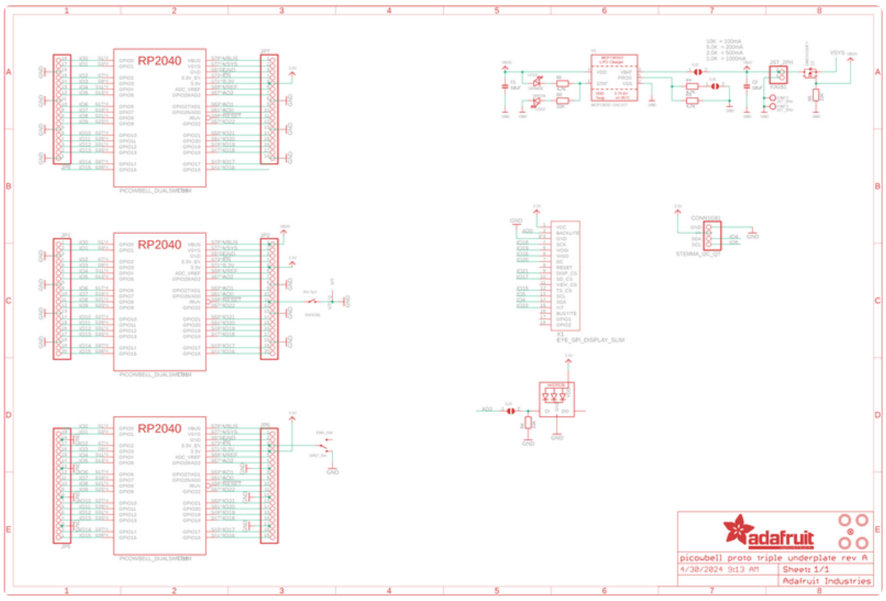


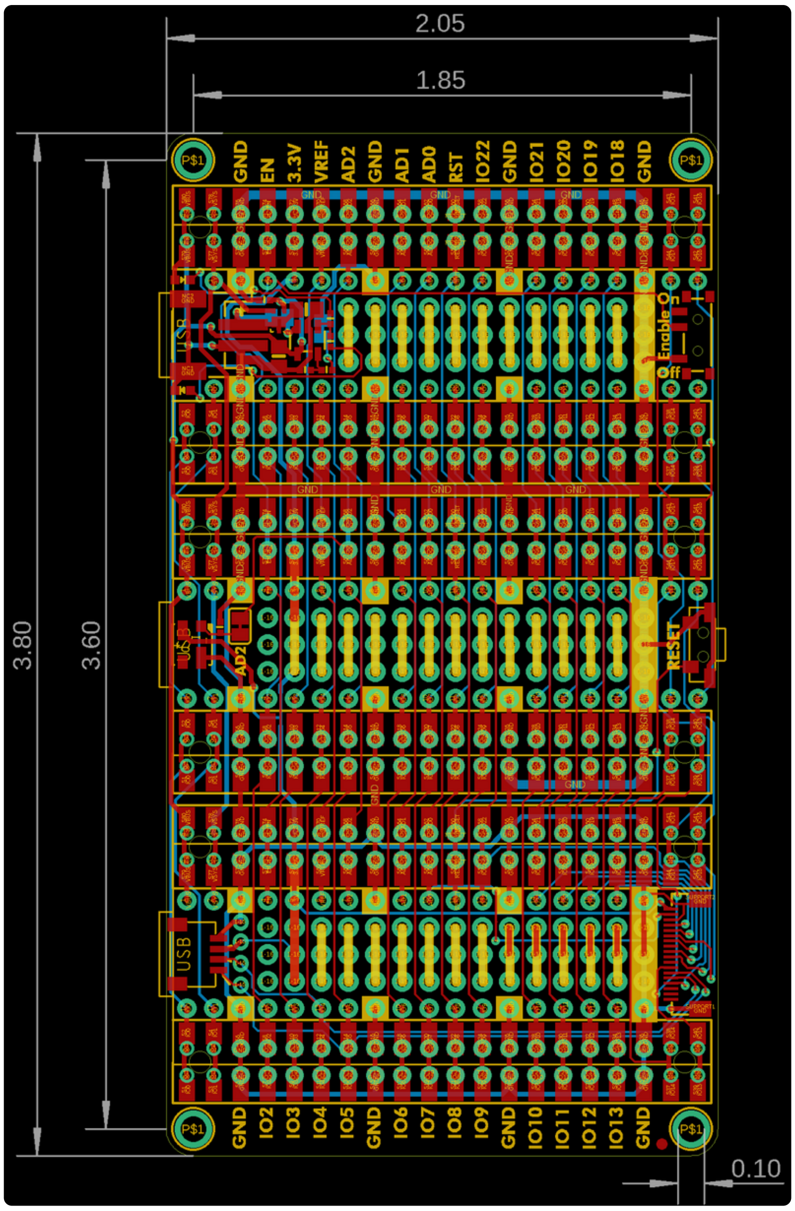
Downloads

Files

- EagleCAD PCB files on GitHub (https://adafru.it/1a10)
- 3D models on GitHub (https://adafru.it/1a16)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1a11)

Schematic and Fab Print




2.05
1.85
3.80
3.60
0.10
GND
EN
3.3V
VREF
AD2
GND
AD1
AD0
RST
IO22
GND
IO21
IO20
IO19
IO18
GND
GND
IO2
IO3
IO4
IO5
GND
IO6
IO7
IO8
IO9
GND
IO10
IO11
IO12
IO13
GND
USB
RESET
Enable
Off