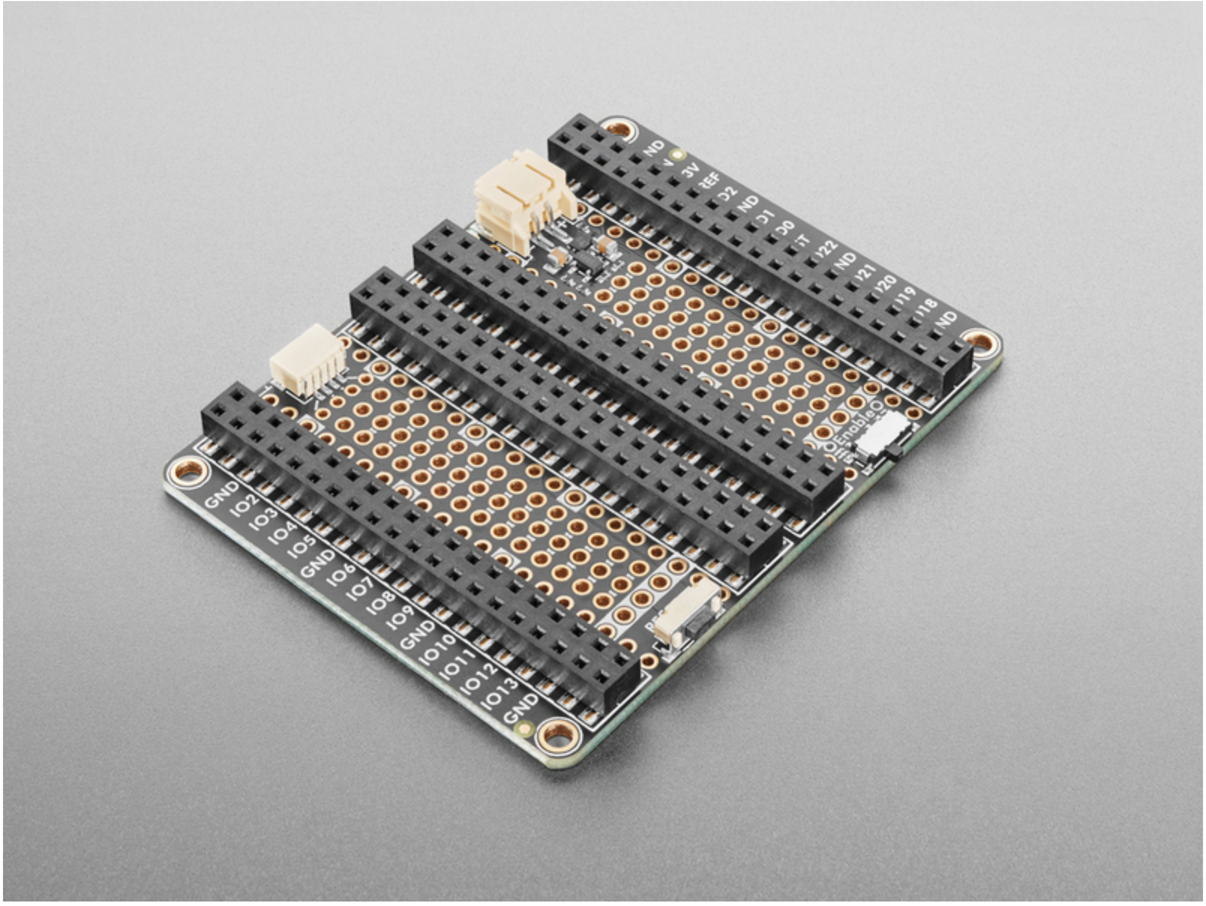
Adafruit Proto Doubler PiCowbell

Created by Liz Clark



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

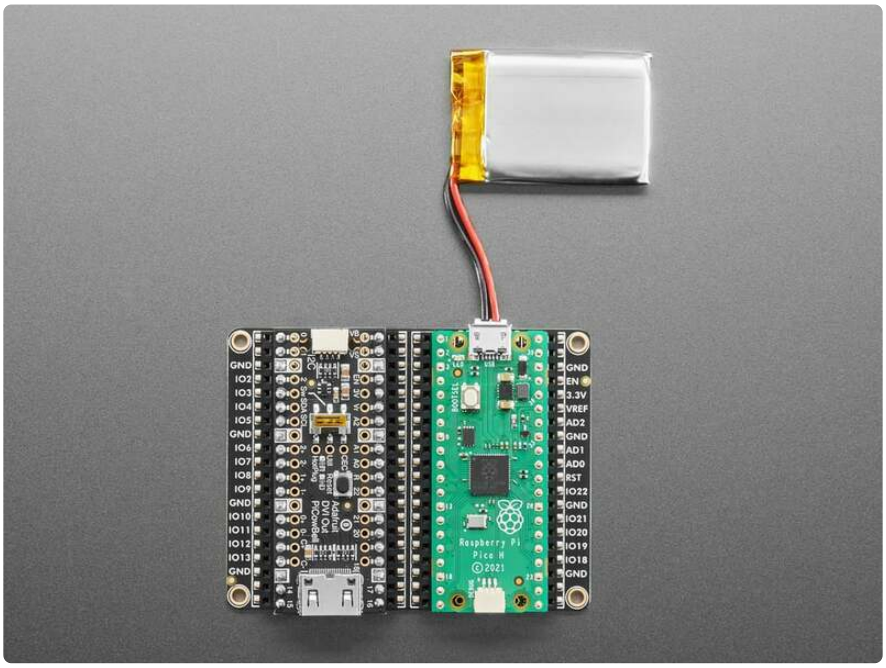
Last updated on 2024-06-03 03:59:23 PM EDT

Table of Contents

Overview

The **Adafruit Proto Doubler PiCowBell** is intended to be treated like a mini solder-less proto plate to simplify programming and sensor 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. 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!

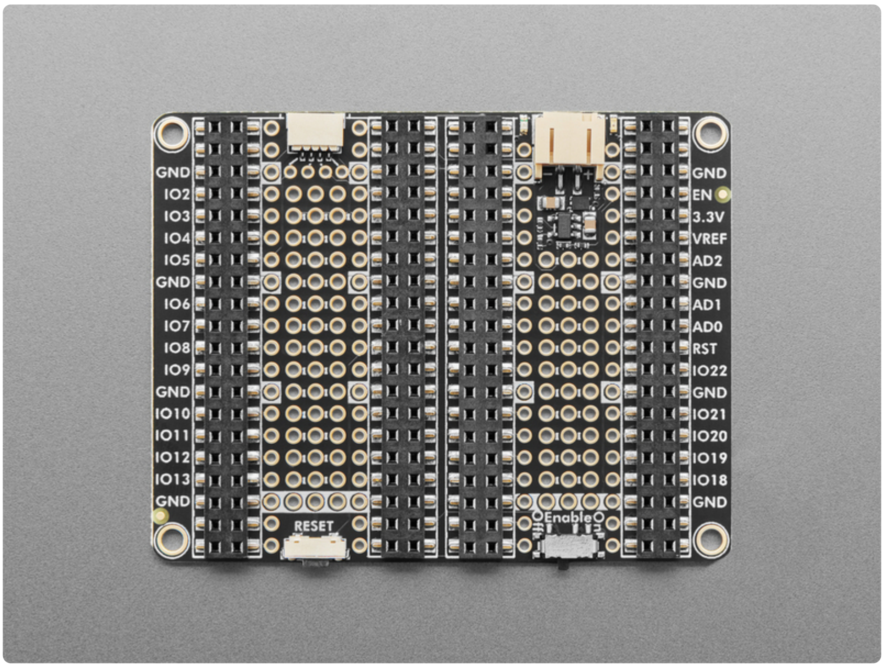
The Doubler is like if we used a 'Clone Tool' on the Proto Under Plate PiCowbell (http://adafru.it/5905) - you get two slots side-by-side which means you can easily add accessories to your Pico or Pico W without soldering on any stacking headers.



LiPoly Battery Charging Support

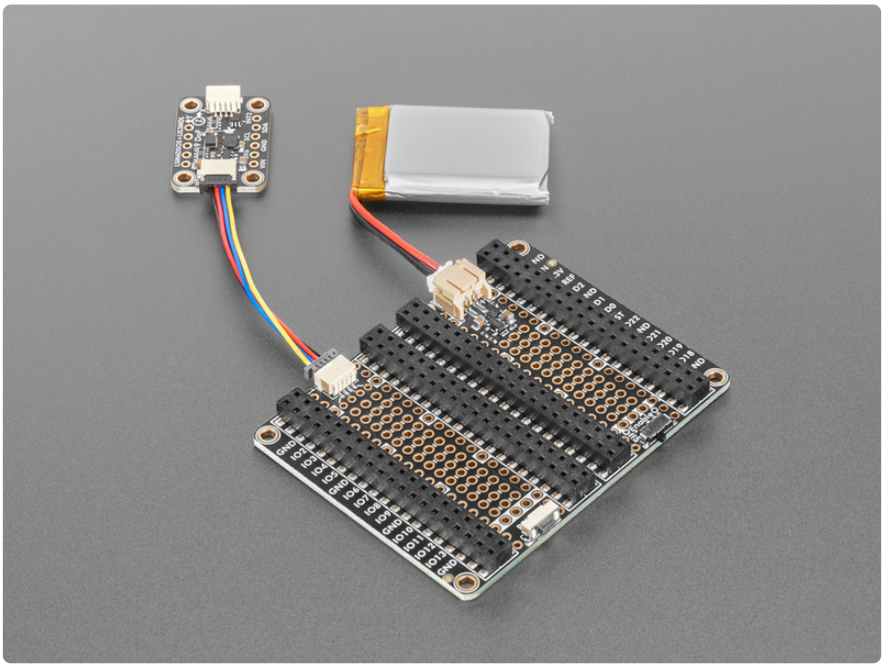
The Doubler also gives you Lipoly/LiIon support circuitry so you can take your Pico project on the go. Use any 3.7V/4.2V single-cell Lithium battery with a JST 2-PH connector in the correct polarity. The battery will automagically charge with the Pico is plugged into USB, and switch over seamlessly to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

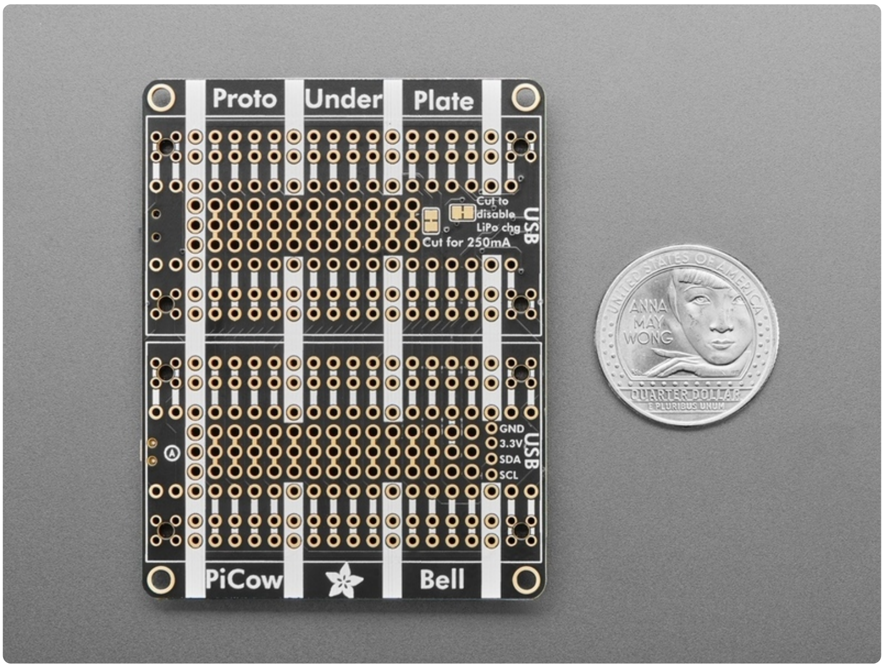


Double Socket Headers

This board has double sockets already soldered on, you can plug in your Pico and 'Bell boards directly in, and get access to a prototyping area underneath as well as 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's also 4 mounting holes for easy attachment to an underplate.

The dual socket headers we use are 'hollow' - that means you can connect through the back if desired. Specifically, you can use Pico Stacking Headers (http://adafru.it/5582) to plug through to 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.

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



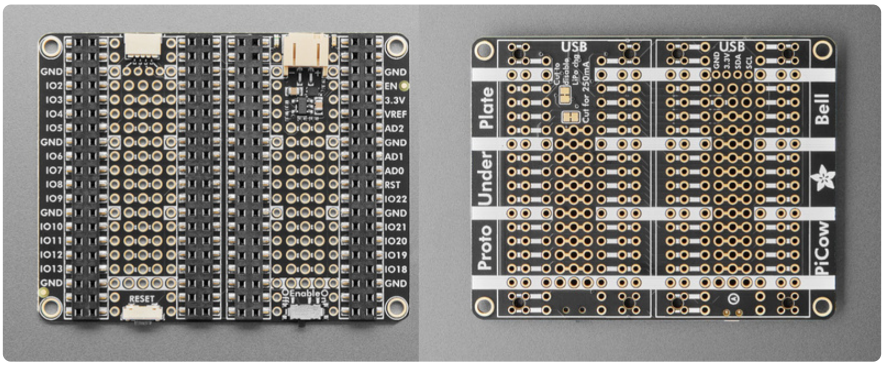
Features:

- **Two 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 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

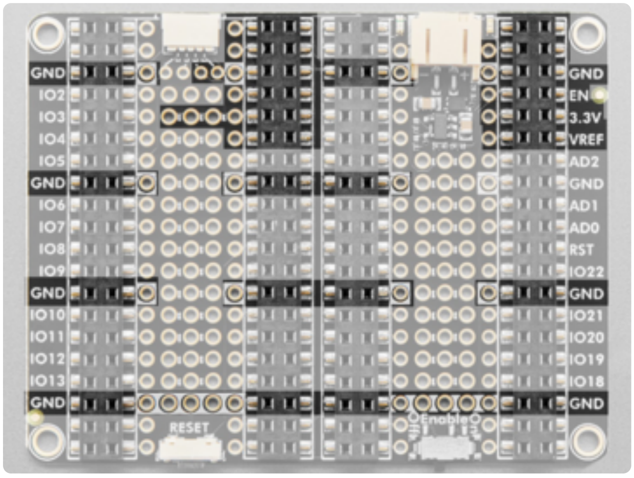
We do not have I2C pullups on the board, but your Qwiic/QT breakout board or accessory likely already has them onboard. If using the Philhower Arduino core, the Wire peripheral is already set up to use IO4 and IO5. If using CircuitPython or MicroPython, you'll need to let the code know to look at 4+5 for SDA+SCL pins.

Pinouts



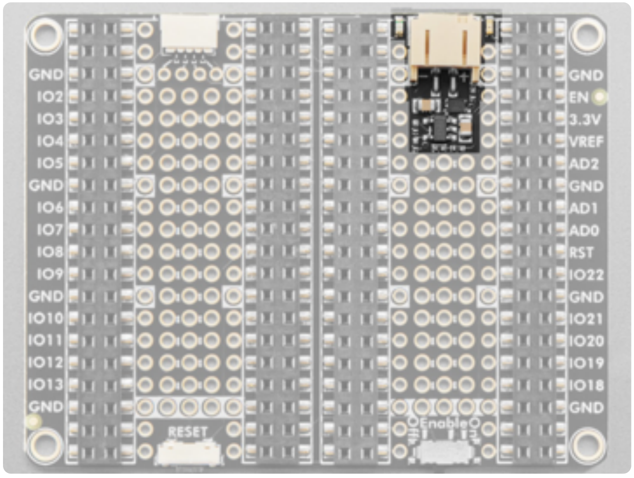
Check that your battery has the correct polarity for the Doubler PiCowbell! Otherwise you could damage or destroy the PiCowbell and anything plugged into it!

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 Doubler 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.**

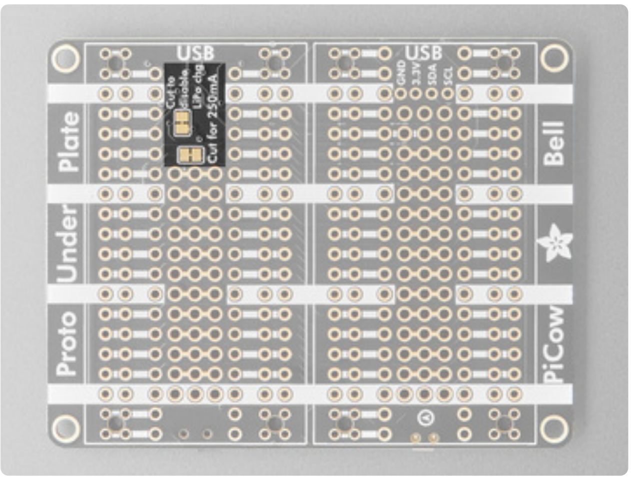
The Doubler also gives you Lipoly/LiIon support circuitry:



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.

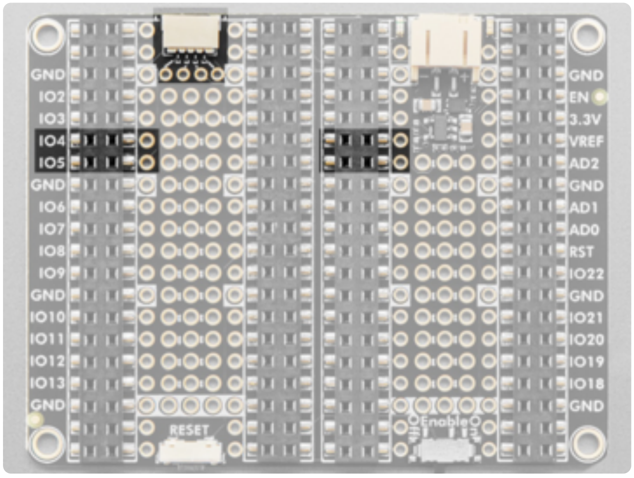


Charge Rate Jumper - On the back of the Doubler is the charge rate jumper. It is labeled **Cut for 250mA** on the board silk. By default the charge rate is 500mA for use with 500mAh+ sized batteries. You can cut a jumper to reduce the charge rate to 250mA if using 250mAh to 500mAh battery. **Batteries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

Alkaline/NiMH Battery Pack Jumper - On the back of the Doubler is the alkaline/NiMH battery pack jumper. It is labeled **Cut to disable LiPo chg** 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



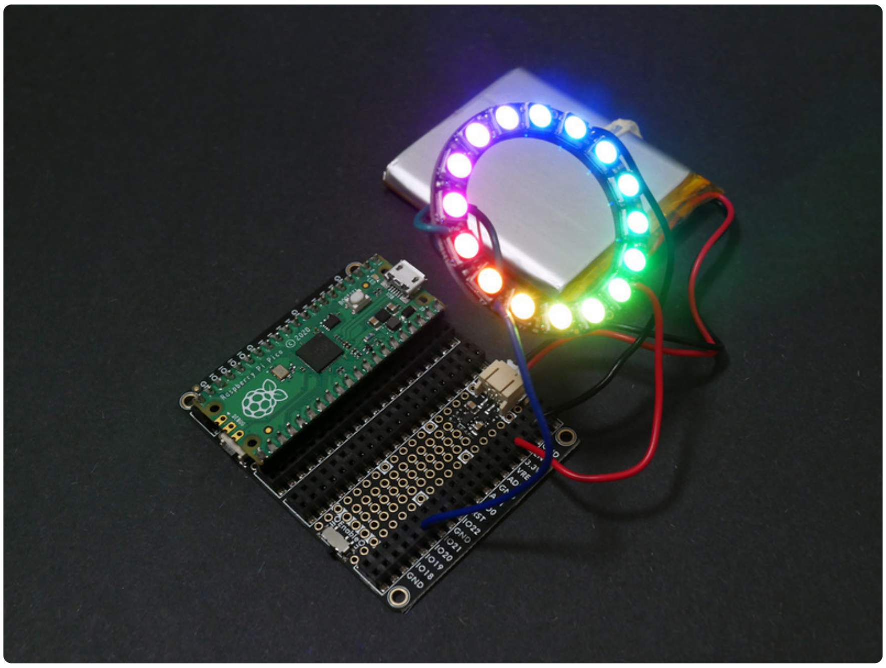
SCL - I2C clock pin on the PiCowBell. It is connected to your Pico I2C clock line, which is **GPIO5**. This connection is shared with the STEMMA QT port on the end of the board.

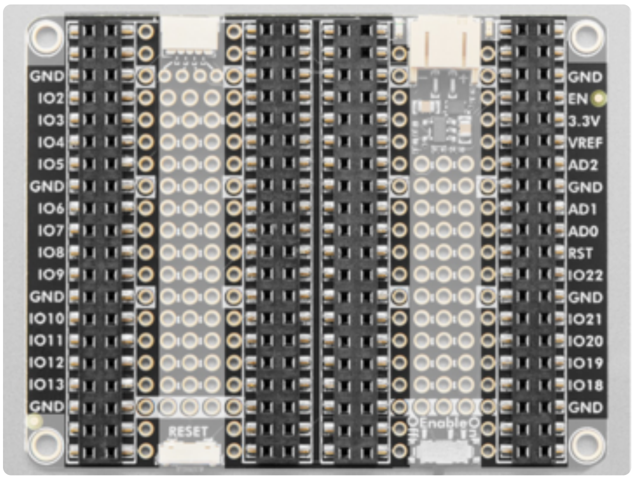
SDA - I2C data pin on the PiCowBell. It is connected to your microcontroller I2C data line, which is **GPIO4**. This connection is shared with the STEMMA QT port on the end of the board.

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.

Socket Headers and Duplicate GPIO Hole Pads

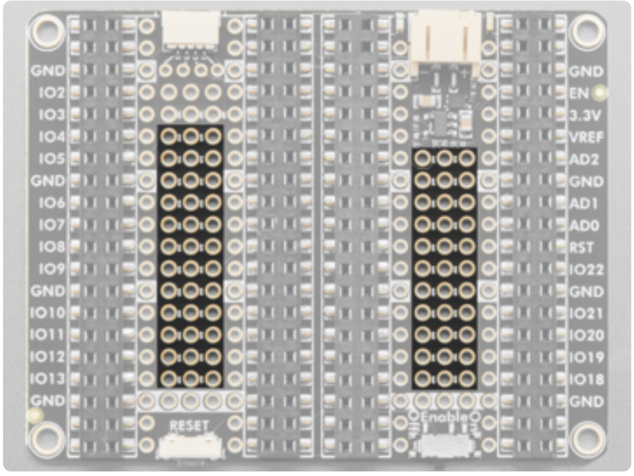




The PiCowbell has two sets of two 2x20 slim socket headers to plug in your Pico and have an extra row of sockets for each pin. On the second set of headers you can plug in an accessory for your Pico. There are also duplicate hole pads next to each pin for solder-jumpering:

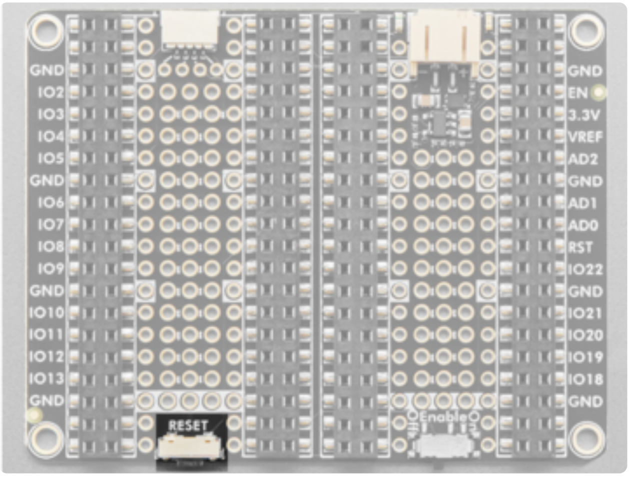
IO0-IO15, **IO16-IO22**, **Reset**, **A0-A2**, **VR**, **3V**, **EN**, **VS** and **VB**. Ground pins that have a duplicate hole pad are highlighted in white rectangles on the board silkscreen.

Proto Areas



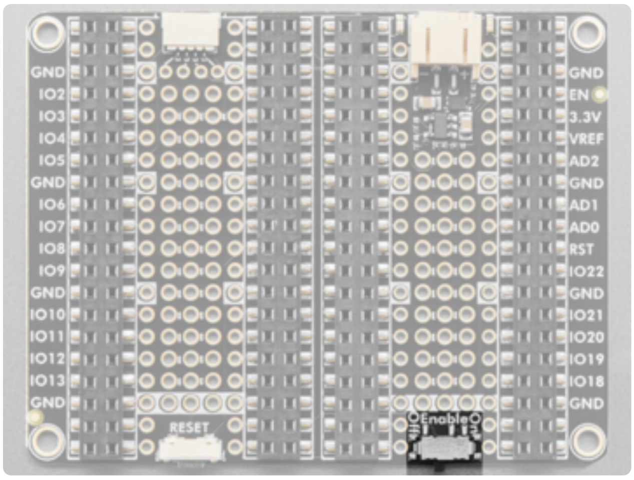
Between the sets of socket headers, you'll find the **proto areas**. These areas on the board are made up of 12 **3 hole-connected strips on the left** and 11 **3 hole-connected strips on the right**. You can cut the traces between the holes, but they're intended to be treated like mini-mini breadboards.

Reset Button



At the bottom left edge of the board 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 Doubler PiCowbell** with CircuitPython to monitor the voltage of an attached lipoly battery with the analogio (https://adafru.it/19lE) core module. This module allows you to easily write Python code for accessing basic analog inputs and outputs.

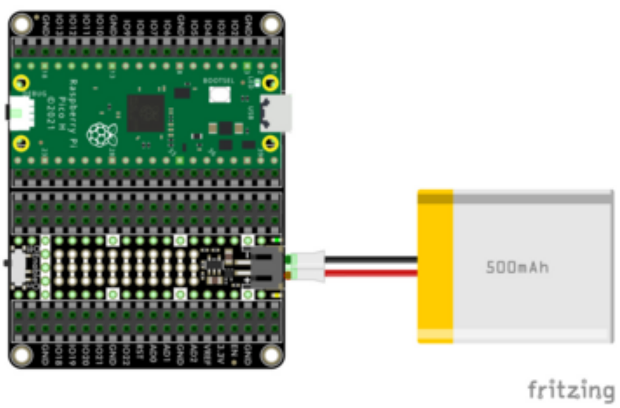
Raspberry Pi Pico ADC3 and VSYS

The Raspberry Pi Pico can use its internal ADC3 pin (GPIO29) to monitor the voltage on VSYS (https://adafru.it/19lc). 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 Doubler PiCowbell exactly as shown below. Then, plug in a supported lipoly battery to the PiCowbell JST 2-PH port. Here's an example of connecting a Pico to the PiCowbell with a lipoly battery.



Connect the Pico with plug headers into the Proto Doubler PiCowbell. It should be plugged in with the Pico USB port pointing towards the STEMMA QT port and the **USB** text on the PiCowbell board silk.

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

CircuitPython Usage

To use with CircuitPython, you need to update **code.py** with the example script.

In the example below, click the **Download Project Bundle** button below to download the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **code.py** file to your **CIRCUITPY** drive.

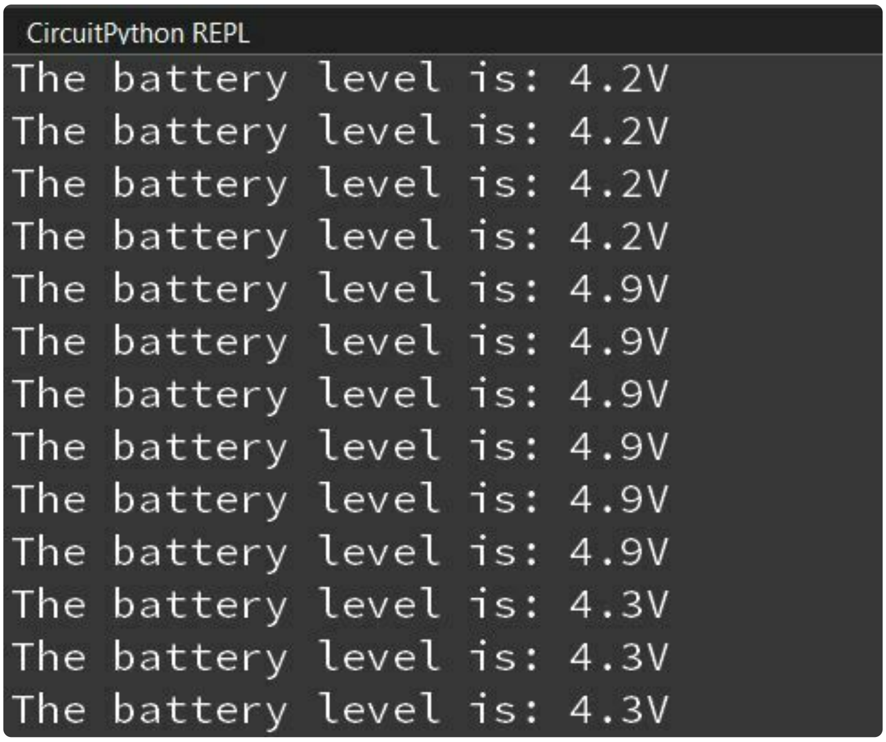
No additional libraries are needed in the /lib folder for this example.

Example Code

If running CircuitPython: 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: 2024 ladyada for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
from analogio import AnalogIn
from digitalio import DigitalInOut, Direction

led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT

analog_in = AnalogIn(board.A3)

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

while True:
    led.value = True
    print(f"The battery level is: {get_vsys(analog_in):.1f}V")
    led.value = False
    time.sleep(5)
```

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 4.2V. The battery was unplugged, reading 4.9V from the USB power and then plugged back in, reading 4.3V.



Arduino

Using the Proto Doubler PiCowbell with Arduino to monitor the voltage of an attached lipoly battery involves plugging a Pico or Pico W board into the PiCowbell, attaching a compatible battery to the JST2-PH port and running the provided example code.

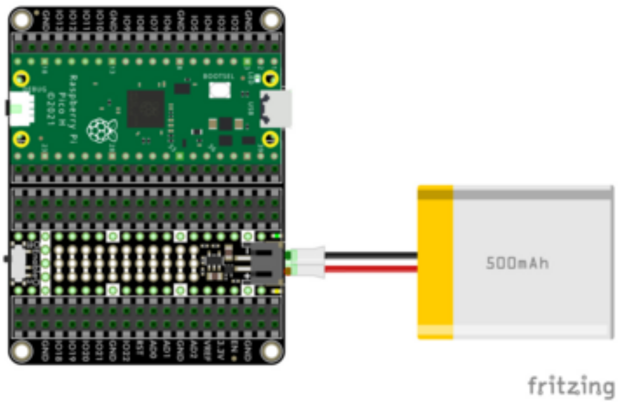
Raspberry Pi Pico ADC3 and VSYS

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 Doubler PiCowbell exactly as shown below. Then, plug in a supported lipoly battery to the PiCowbell JST 2-PH port. Here's an example of connecting a Pico to the PiCowbell with a lipoly battery.



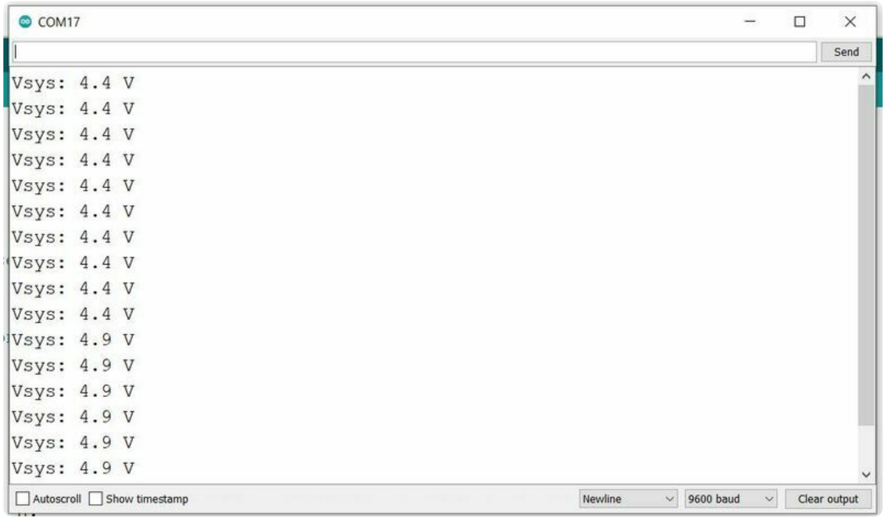
Connect the Pico with plug headers into the Proto Doubler PiCowbell. It should be plugged in with the Pico USB port pointing towards the STEMMA QT port and the **USB** text on the PiCowbell board silk.

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

Example Code

```
// SPDX-FileCopyrightText: 2024 Limor Fried for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#define LED    LED_BUILTIN

void setup() {
  Serial.begin(115200);
  // while (!Serial)  delay(1);  // wait for serial port
  pinMode(LED, OUTPUT);
  delay (100);
  Serial.println("PiCowbell Doubler Battery Monitor");

}

void loop() {
  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);
  delay(5000);
}
```

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, a battery was plugged in and was providing a reading of approximately 4.4V. After the battery was unplugged, the reading changed to 4.9V from the USB power.

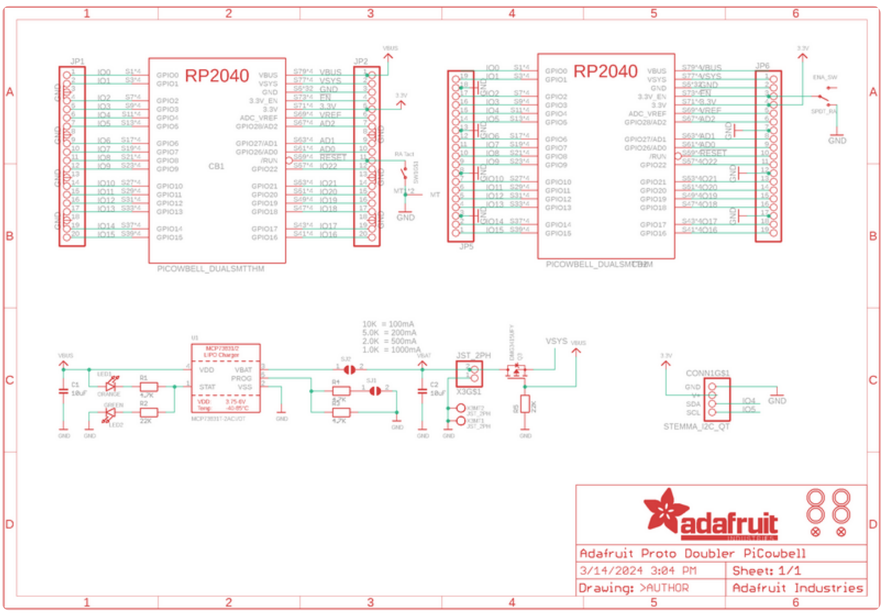


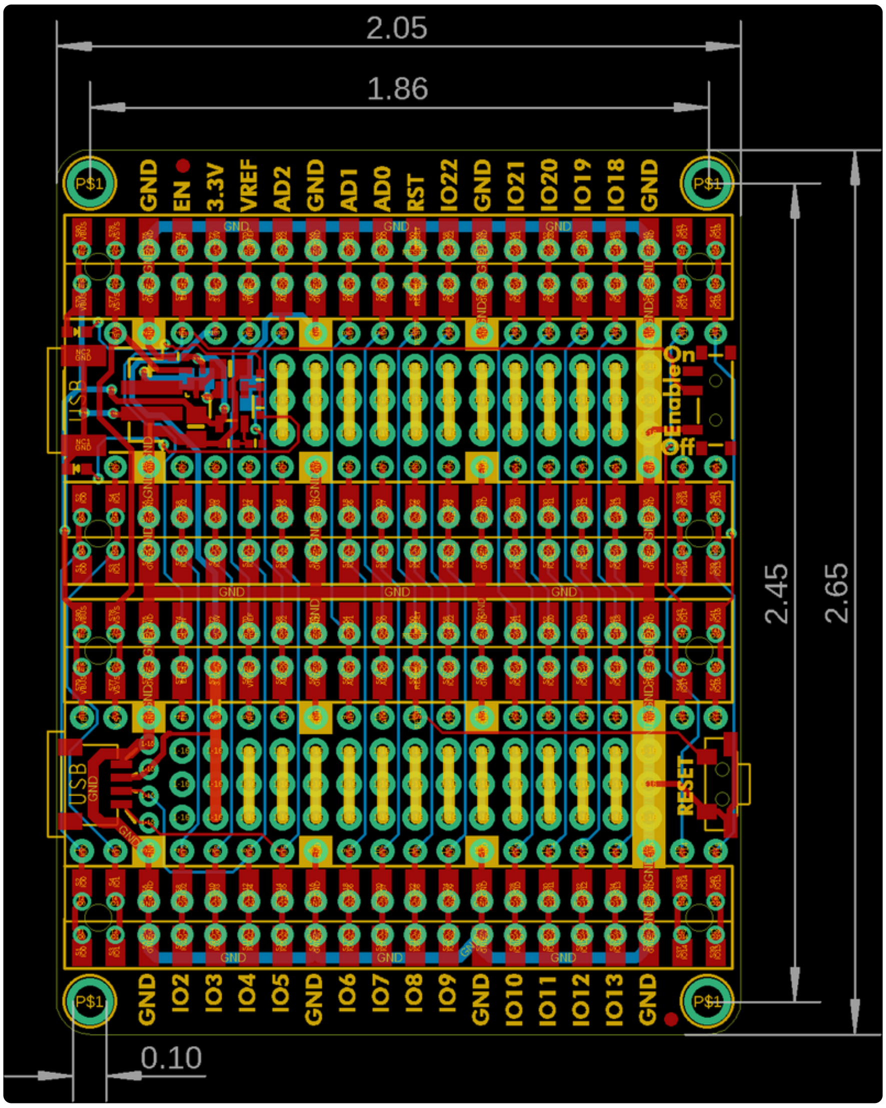
Downloads

Files

- EagleCAD PCB files on GitHub (https://adafru.it/19Id)
- 3D Models on GitHub (https://adafru.it/19IF)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/19Ie)

Schematic and Fab Print




2.05
1.86
2.45
2.65
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