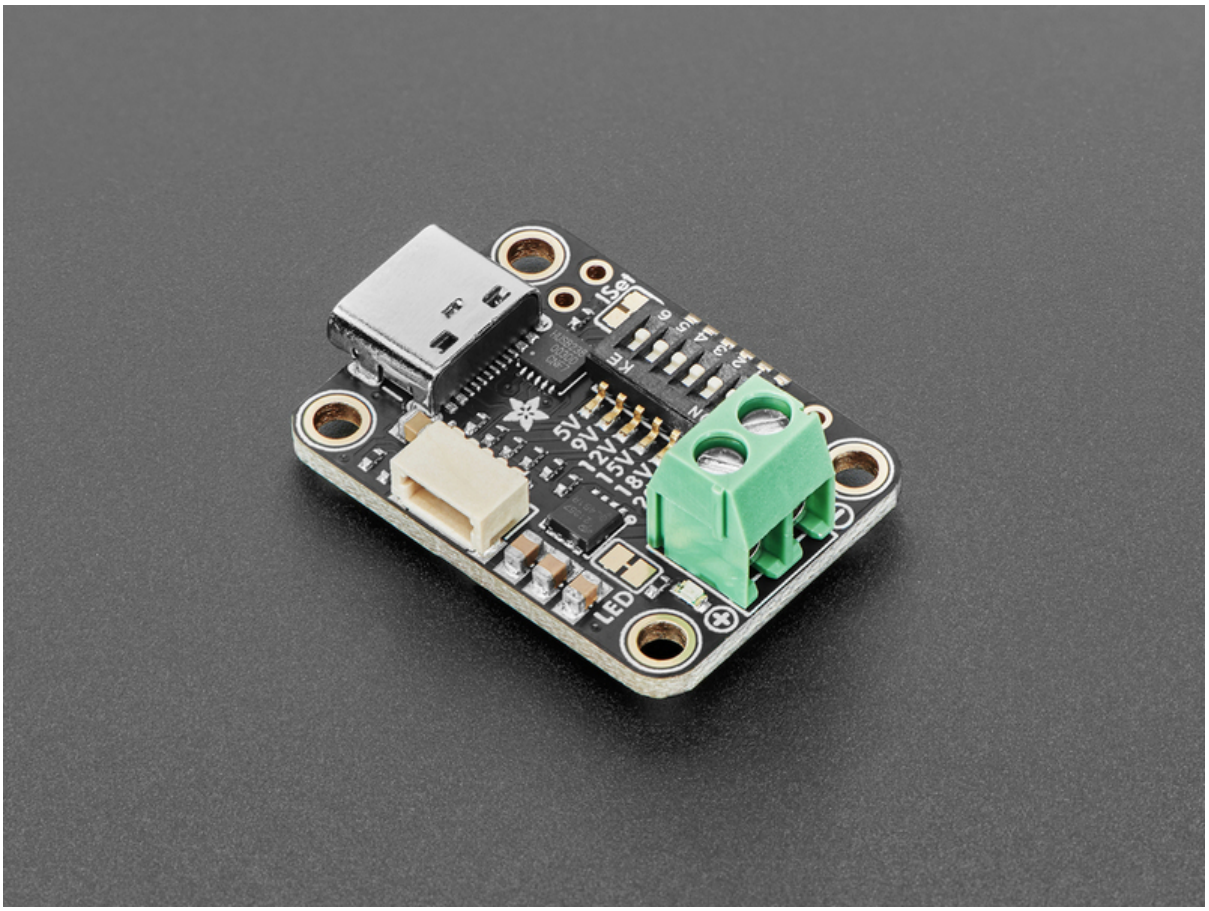
Adafruit USB Type C Power Delivery Switchable Breakout

Created by Liz Clark

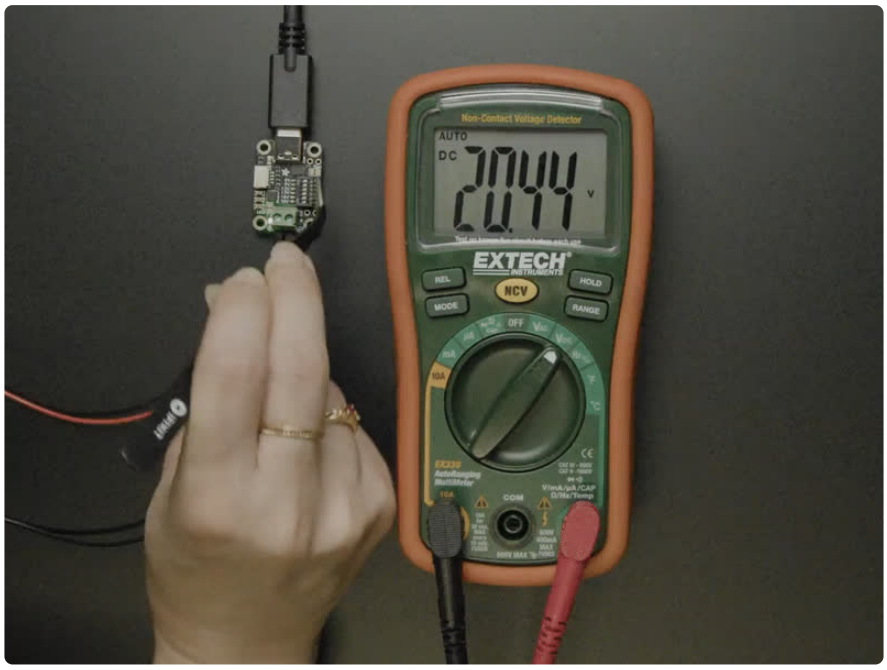


https://learn.adafruit.com/adafruit-usb-type-c-power-delivery-switchable-breakout

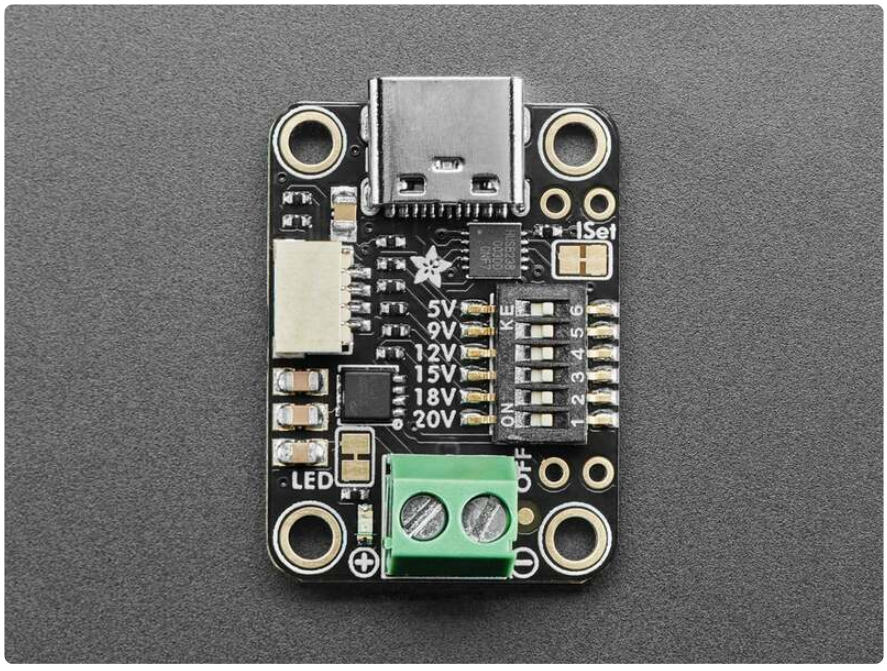
Last updated on 2024-08-22 10:34:25 AM EDT

Table of Contents

Overview

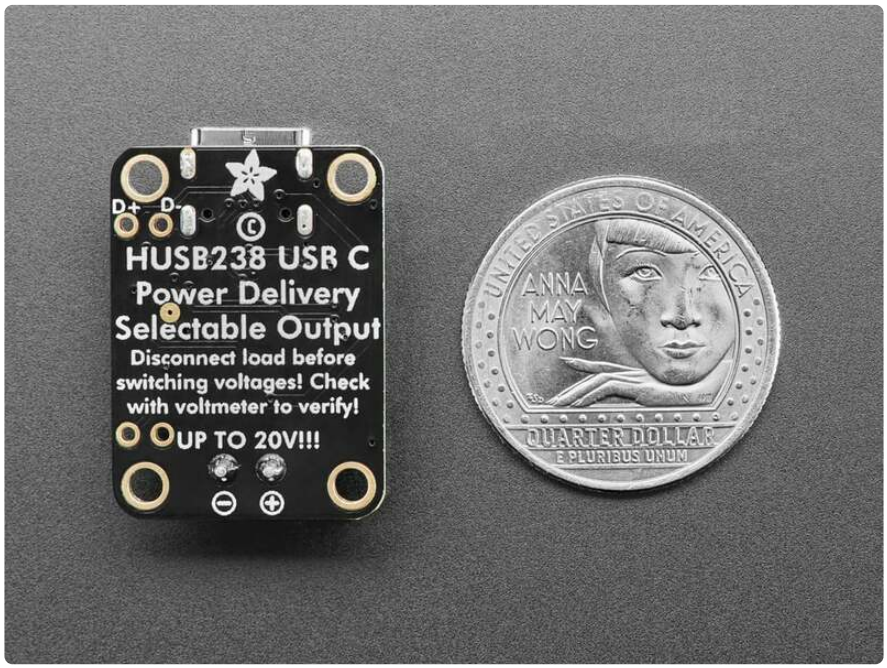


The HUSB238 USB PD sink chip is neat in that you can either use switches (really, resistor selection) to set the desired PD voltage or use I2C for dynamic querying and setting. We already stock a simple Adafruit USB Type C Power Delivery Dummy Breakout **(http://adafru.it/5807)** board around the HUSB238, but that one requires soldering closed jumpers to select the voltage. For folks who want a no-soldering-required board, this **Adafruit USB Type C Power Delivery Dummy - I2C or Switchable** is ready for instant gratification. No soldering required!

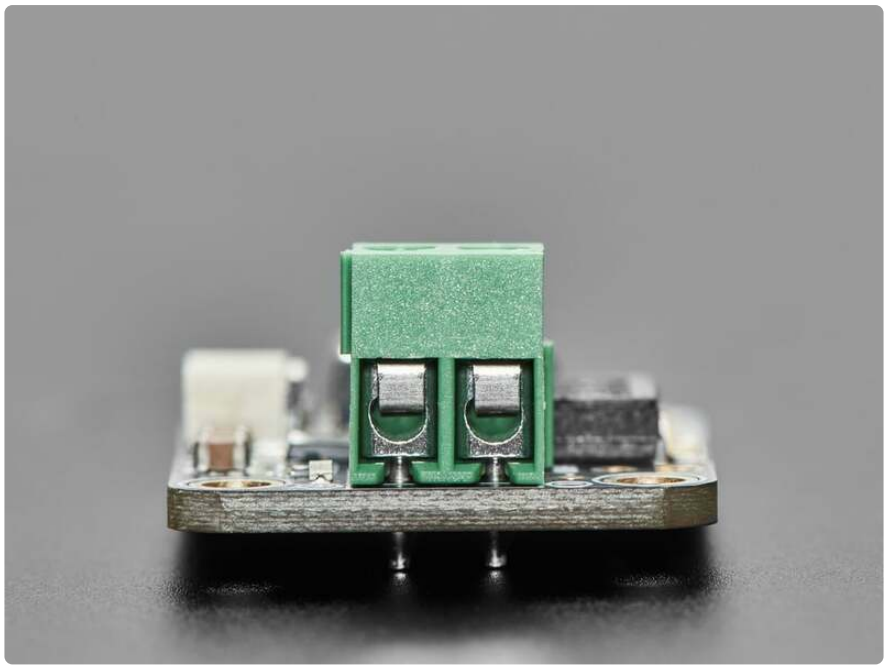


Compared to the basic Adafruit USB Type C Power Delivery Dummy Breakout **(http://adafru.it/5807)** we've done a renovation to keep the same great schematic and also add some oft-requested features:

- **No soldering required!** Switches allow on-the-fly voltage changes. Terminal block can be used to provide power to your robot, display, LEDs, etc.
- **Green LED** output lets you know that the terminal block power is on - can be disabled if you prefer
- **Stemma QT for I2C control** plug-and-play with Qwiic/QT
- **USB Data lines** available on solderable pads
- **On/Off switch** - you can disconnect the internal pass FET by connecting a switch between two onboard pads. When the switch is closed the output will disconnect / turn off.
- We use the **HUSB328 PDL003A E-Marker variant** of the chip, a little more expensive, but it can request 5A power.

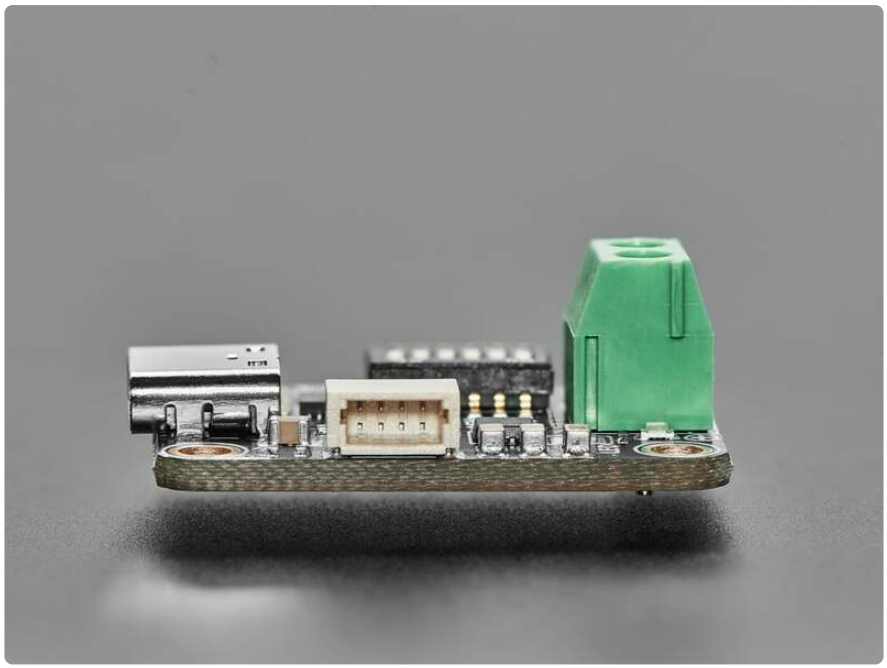


It's perfect for USB Type C wall adapters providing multiple voltages. The standard offerings are 5V, 9V, 12V, 15V, 18V, and 20V. This HUSB238 breakout plugs into the USB C cable and negotiates the PD request and commands over the CC lines. For example, we can ask what voltages are available and pick the highest. Or if you need a specific voltage, it will select that one.



This breakout will be handy for projects where you need a lot more than 5V @ 2A power: this adapter can give up to 20V at 5A - yes you can get 100W over USB C! - and you could buck that down to get a ton of current at 5V or 12V if that's needed. Or use it to convert a DC or battery-powered device into a USB C powered one!

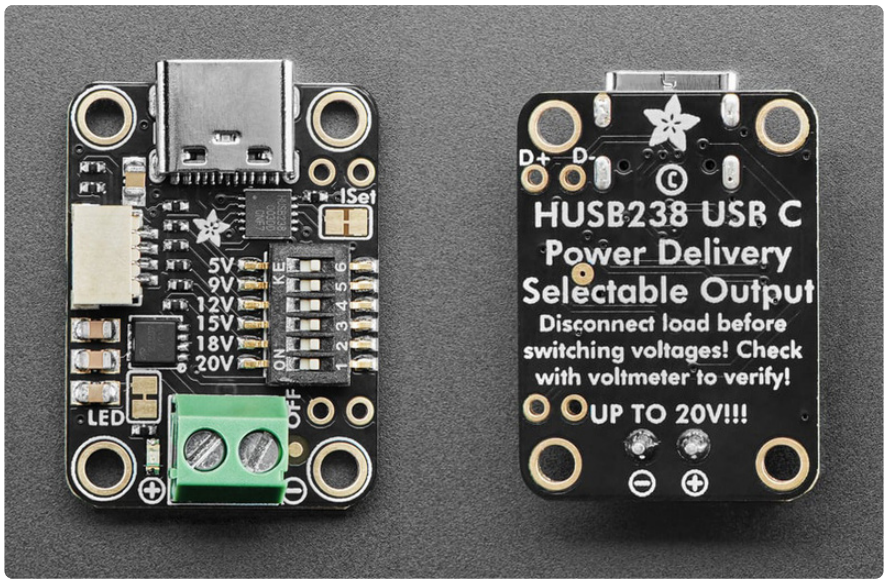
DIP switch-configured usage is simple: simply unplug the USB PD so you don't accidentally select too high a voltage for your device. Then switch ON the voltage you want: 5V, 9V, 12V, 15V, 18V, or 20V. You'll get that voltage and as much current as the adapter will provide. No microcontroller or microcomputer is required!



I2C-configured usage is also available via a Stemma QT port. Use the Arduino library and example code (https://adafru.it/192c) to query the USB Type C PD source for available voltages and currents and select the desired voltage dynamically. When configuring over I2C, the jumper settings are used on startup until the I2C commands come over.

The STEMMA QT connector (https://adafru.it/JqB) lets you make solderless connections between your development board and the breakout or chain it with a wide range of other sensors and accessories using a **compatible cable** (https://adafru.it/JnB). **QT Cable is not included**, but we have a variety in the shop (https://adafru.it/17VE).

Pinouts



The default I2C address is **0x08**.

Power Output Terminal Block

- **+** - This is the voltage output pin. This pin will output the voltage selected from the HUSB238.
- **-** - Common ground for power and logic.

USB Type C Port and Data Pins

At the top of the board is the USB type C port. You'll use this port to plug into a USB C PD wall adapter with a USB C cable. The USB data pins are broken out on the board: Data Plus (labeled **D+** on the board silk) and Data Minus (labeled **D-** on the board silk). These pins are located to the right of the USB type C port.

DIP Switches

On the right side of the board is the 1x6 DIP switch. These switches let you select the voltage that is requested via USB PD. The available voltages are labeled on the board silk. To select a voltage, flip the switch to **ON** next to the board silk label for the voltage you need. The following voltages are available: **5V**, **9V**, **12V**, **15V**, **18V**, **20V**.

If none of the switches are turned **ON**, then the highest voltage available from the USB PD supply will be selected, which could be up to 20V.

If none of the switches are turned ON, then the highest voltage available from the USB PD supply will be selected, which could be up to 20V.

I2C

Connect the STEMMA QT connector to a microcontroller or microcomputer board that has a separate power supply. When configuring over I2C, the selected switch settings are used on startup until the I2C commands come over.

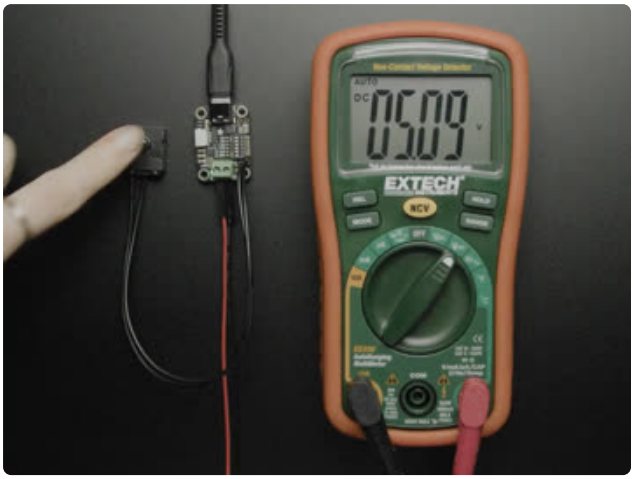
- STEMMA QT **(https://adafru.it/Ft4)** - This port is located on the left side of the board. These connectors allow you to connect to dev boards with **STEMMA QT** (Qwiic) connectors or to other things with various associated accessories (https://adafru.it/Ft6) with connections for GND (black wire), power (red wire), SDA (blue wire) and SCL (yellow wire).

When configuring over I2C, the switch settings are used on startup until the I2C commands come over.

ISet Jumper

- **ISet** - Towards the top right corner of the board, above the DIP switches, is the ISet jumper. This jumper is connected to the **ISET** pin on the HUSB238. The **ISET** pin controls the current available on the board. When the jumper is closed, 1.25A is selected. If the jumper is cut (open), then 3.25A is selected.

On/Off Switch



OFF - directly below the DIP switches are two pins that can be used as an on/off switch. These pins are labeled **OFF** on the board silk. You can disconnect the internal pass FET by connecting a switch between these two onboard pads. When the switch is closed the output will disconnect / turn off.

Power LED and Jumper

- **Power LED -** In the lower left corner, to the left of the terminal block, on the front of the board, is the power LED. It is a green LED.
- **LED jumper** - This jumper is located on the front of the board, above the green LED, and is labeled **LED** on the board silk. Cut the trace on this jumper to cut power to the power LED.

CircuitPython and Python

It's easy to use the **HUSB238** with Python or CircuitPython, and the Adafruit_CircuitPython_HUSB238 (https://adafru.it/192d) module. This module allows you to easily write Python code to control the power delivery chip.

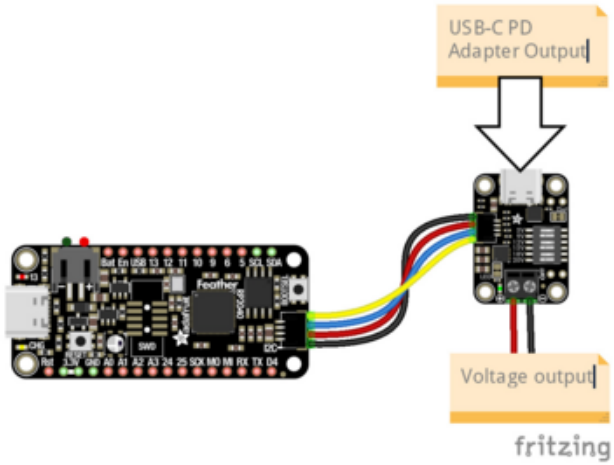
You can use this driver with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

It's important to note that when you are controlling the breakout over I2C, the DIP switch settings on the board are used on startup until the I2C commands come over.

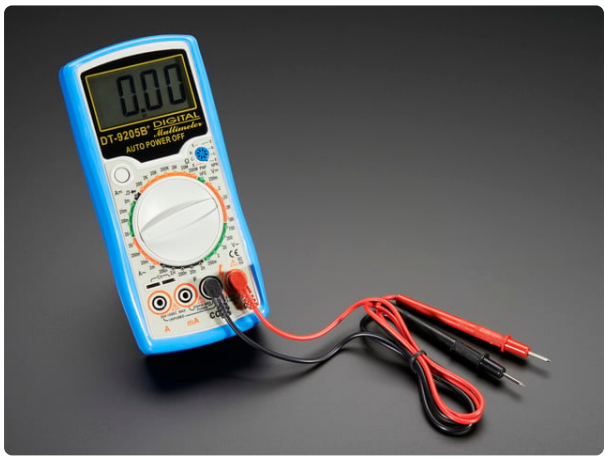
When configuring over I2C, the DIP switch settings are used on startup until the I2C commands come over.

CircuitPython Microcontroller Wiring

First wire up the breakout to your board exactly as follows. For testing, you can connect the + and - outputs from the breakout to a multimeter with alligator clips. You'll set the multimeter to read DC voltage (labeled with a "V" and one dashed and one solid line). The following is the breakout wired to a Feather RP2040 using a STEMMA QT cable:



USB C PD power supply to **breakout USB C port**
Board STEMMA GND to **breakout STEMMA GND (black wire)**
Board STEMMA SCL to **breakout STEMMA SCL (yellow wire)**
Board STEMMA SDA to **breakout STEMMA SDA (blue wire)**
Breakout + to **multimeter positive (red wire)**
Breakout - to **multimeter negative (black wire)**



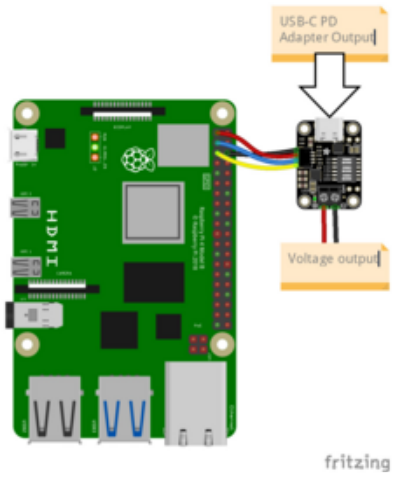
Digital Multimeter - Model 9205B+
This massive multimeter has everything but the kitchen sink included. It's a great addition to any workbench or toolbox. It's low cost, simple to use, and has a big clear...
https://www.adafruit.com/product/2034

Python Computer Wiring

Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

For testing, you can connect the + and - outputs from the breakout to a multimeter with alligator clips. You'll set the multimeter to read DC voltage (labeled with a "V" and one dashed and one solid line).

Here's the Raspberry Pi wired with I2C using STEMMA QT cable:



USB C PD power supply to **breakout USB C port**
Pi GND to **breakout STEMMA GND (black wire)**
Pi 3.3V to **breakout STEMMA VIN**
Pi SCL to **breakout STEMMA SCL (yellow wire)**
Pi SDA to **breakout STEMMA SDA (blue wire)**
Breakout + to **multimeter positive (red wire)**
Breakout - to **multimeter negative (black wire)**

Python Installation of HUSB238 Library

You'll need to install the **Adafruit_Blinka** library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

Once that's done, from your command line run the following command:

- `pip3 install adafruit-circuitpython-husb238`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython Usage

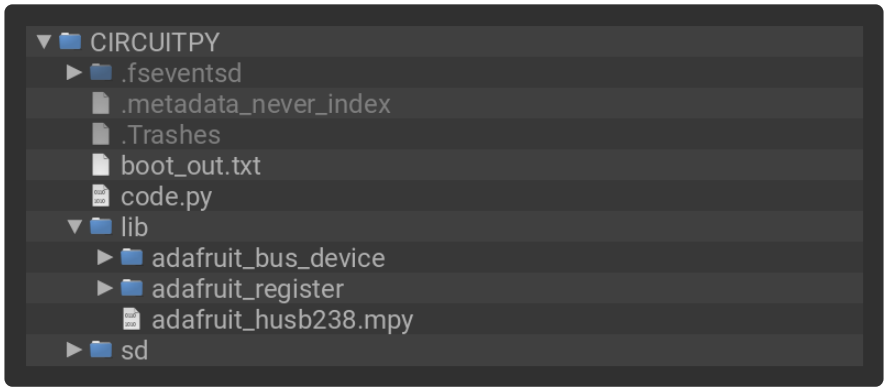
To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_HUSB238** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

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** and the **code.py** file to your **CIRCUITPY** drive.

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

- **adafruit_bus_device/**
- **adafruit_register/**
- **adafruit_husb238.mpy**



Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

```
python3 code.py
```

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!

If running Python: The console output will appear wherever you are running Python.

```
# SPDX-FileCopyrightText: Copyright (c) 2023 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT
"""
Simple test for the HUSB238.
Reads available voltages and then sets each available voltage.
Reads the set voltage and current from the attached PD power supply.
"""
import time
import board
import adafruit_husb238
```

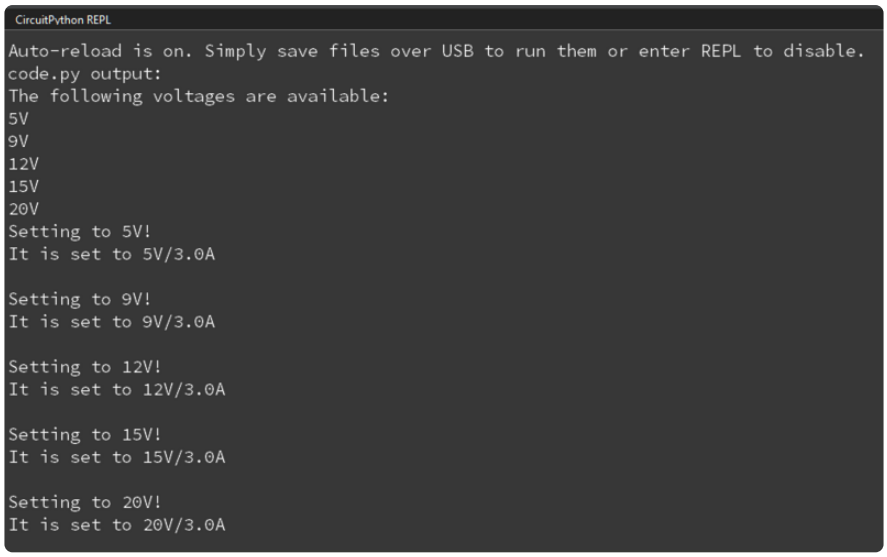
```
i2c = board.I2C()

# Initialize HUSB238
pd = adafruit_husb238.Adafruit_HUSB238(i2c)
voltages = pd.available_voltages
print("The following voltages are available:")
for i, volts in enumerate(voltages):
    print(f"{volts}V")

v = 0

while True:
    while pd.attached:
        print(f"Setting to {voltages[v]}V!")
        pd.voltage = voltages[v]
        print(f"It is set to {pd.voltage}V/{pd.current}A")
        print()
        v = (v + 1) % len(voltages)
        time.sleep(2)
```

For this example, it's best to test with the output from the breakout connected to a multimeter in DC voltage mode. DC voltage mode is labeled with a V and two lines, one dashed and one solid. For information on using a multimeter, check out this guide. (https://adafru.it/192A)

Multimeters Learn Guide

https://adafru.it/192A

In the example, the HUSB238 is instantiated over I2C. Then, the available voltages are read from the attached USB-C PD power supply. In the loop, the available voltages are set, one by one, by the HUSB238. You'll see these output on your multimeter. The voltage and current are read from the PD supply and are printed to the serial console.



Python Docs

Python Docs (https://adafru.it/192d)

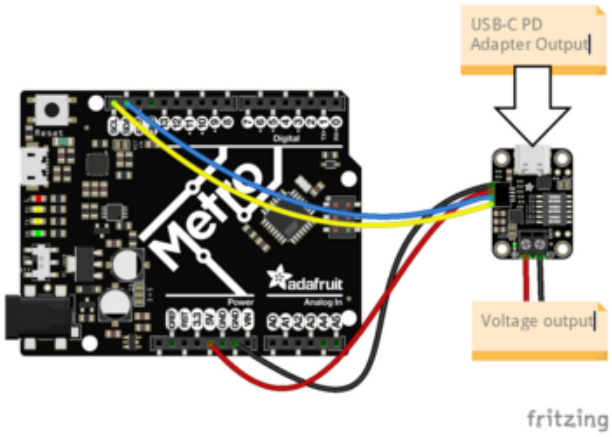
Arduino

Using the HUSB238 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the Adafruit_HUSB238 (https://adafru.it/192c) library, plugging in a USB C PD power supply to the breakout and running the provided example code. It's important to note that when you are controlling the breakout over I2C, the DIP switch settings on the board are used on startup until the I2C commands come over.

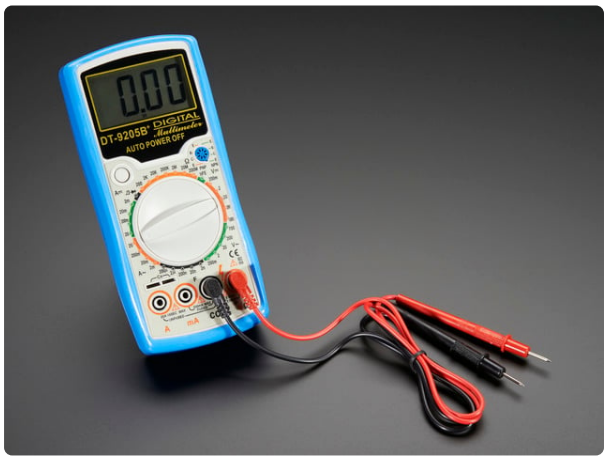
When configuring over I2C, the DIP switch settings are used on startup until the I2C commands come over.

Wiring

Here is an Adafruit Metro wired up to the breakout. For testing, you can connect the + and - outputs from the breakout to a multimeter with alligator clips. You'll set the multimeter to read DC voltage (labeled with a "V" and one dashed and one solid line).



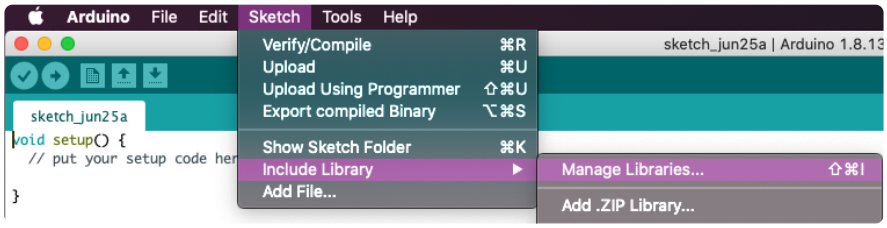
USB C PD power supply to **breakout USB C port**
Board GND to **breakout STEMMA GND (black wire)**
Board 5V to **breakout STEMMA VIN (red wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**
Breakout + to **multimeter positive (red wire)**
Breakout - to **multimeter negative (black wire)**



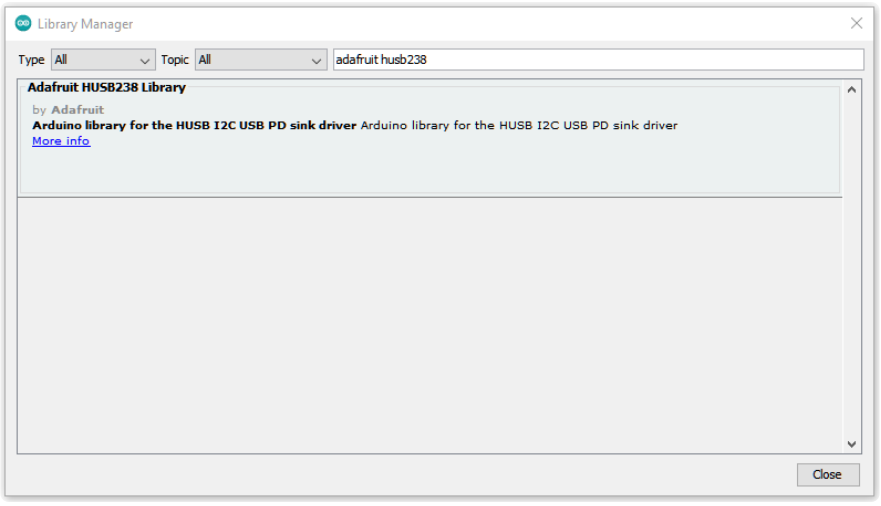
Digital Multimeter - Model 9205B+
This massive multimeter has everything but the kitchen sink included. It's a great addition to any workbench or toolbox. It's low cost, simple to use, and has a big clear...
https://www.adafruit.com/product/2034

Library Installation

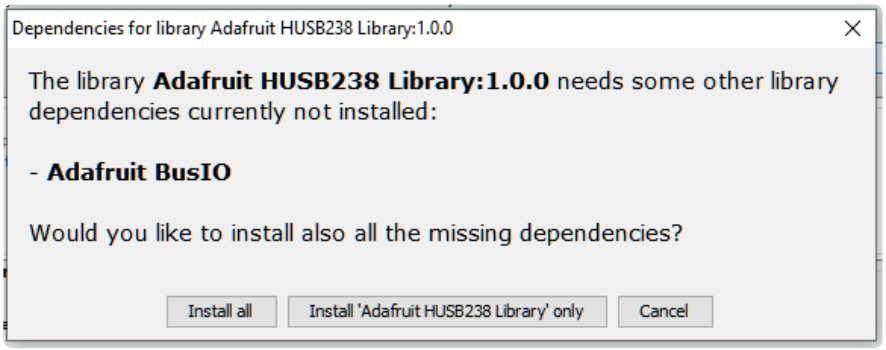
You can install the **Adafruit_HUSB238** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit_HUSB238**, and select the **Adafruit HUSB238** library:



If asked about dependencies, click "Install all".



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!

Simple Test

```
#include <Wire.h>
#include "Adafruit_HUSB238.h"

Adafruit_HUSB238 husb238;

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);
  Serial.println("Adafruit HUSB238 Test Sketch");

  // Initialize the HUSB238
  if (husb238.begin(HUSB238_I2CADDR_DEFAULT, &Wire)) {
    Serial.println("HUSB238 initialized successfully.");
  } else {
    Serial.println("Couldn't find HUSB238, check your wiring?");
    while (1);
  }
}
```

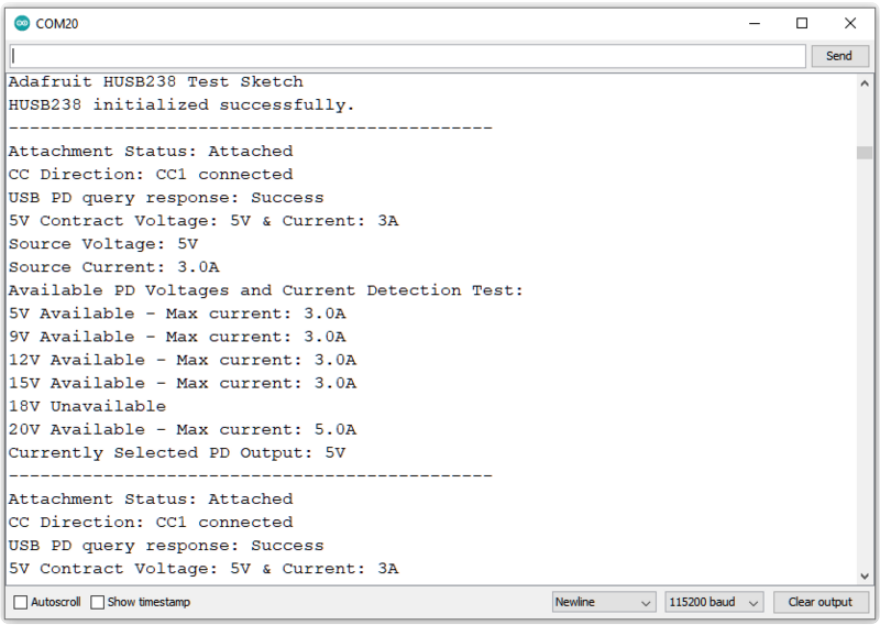
```
void loop() {
  delay(1000);  // Add a delay to prevent flooding the serial output
  Serial.println(F("---------------------------------------------"));

  // Determine whether attached or unattached
  bool attached = husb238.isAttached();
  Serial.print("Attachment Status: ");
  Serial.println(attached ? "Attached" : "Unattached");

  if (! attached) return;

  // Test getCCStatus function
  bool ccStatus = husb238.getCCdirection();
  Serial.print("CC Direction: ");
  Serial.println(ccStatus ? "CC1 connected" : "CC2 Connected");

  // Check if we can get responses to our PD queries!
  HUSB238_ResponseCodes pdResponse = husb238.getPDResponse();
  Serial.print("USB PD query response: ");
  switch (pdResponse) {
    case NO_RESPONSE:
      Serial.println("No response");
      break;
    case SUCCESS:
      Serial.println("Success");
      break;
    case INVALID_CMD_OR_ARG:
      Serial.println("Invalid command or argument");
      break;
    case CMD_NOT_SUPPORTED:
      Serial.println("Command not supported");
      break;
    case TRANSACTION_FAIL_NO_GOOD_CRC:
      Serial.println("Transaction fail");
      break;
    default:
      Serial.println("Unknown response code");
      break;
  }

  if (pdResponse != SUCCESS)
    return;

  // Is there a default 5V 'contract' voltage available
  bool contractV = husb238.get5VContractV();
  Serial.print("5V Contract Voltage: ");
  Serial.print(contractV ? "5V" : "Other");

  // How much current can we get?
  HUSB238_5VCurrentContract contractA = husb238.get5VContractA();
  Serial.print(" & Current: ");
  switch (contractA) {
    case CURRENT5V_DEFAULT:
      Serial.println("Default current");
      break;
    case CURRENT5V_1_5_A:
      Serial.println("1.5A");
      break;
    case CURRENT5V_2_4_A:
      Serial.println("2.4A");
      break;
    case CURRENT5V_3_A:
      Serial.println("3A");
      break;
    default:
      Serial.println("Unknown current");
      break;
  }
```

```
// What is the actual voltage being output right now?
HUSB238_VoltageSetting srcVoltage = husb238.getPDSrcVoltage();
Serial.print("Source Voltage: ");
switch (srcVoltage) {
  case UNATTACHED:
    Serial.println("Unattached");
    break;
  case PD_5V:
    Serial.println("5V");
    break;
  case PD_9V:
    Serial.println("9V");
    break;
  case PD_12V:
    Serial.println("12V");
    break;
  case PD_15V:
    Serial.println("15V");
    break;
  case PD_18V:
    Serial.println("18V");
    break;
  case PD_20V:
    Serial.println("20V");
    break;
  default:
    Serial.println("Unknown voltage setting");
    break;
}

// What is the max current available right now?
HUSB238_CurrentSetting srcCurrent = husb238.getPDSrcCurrent();
Serial.print("Source Current: ");
printCurrentSetting(srcCurrent);
Serial.println();

// What voltages and currents are available from this adapter?
Serial.println("Available PD Voltages and Current Detection Test:");
for (int i = PD_SRC_5V; i <= PD_SRC_20V; i++) {
  bool voltageDetected = husb238.isVoltageDetected((HUSB238_PDSelection)i);

  switch ((HUSB238_PDSelection)i) {
    case PD_SRC_5V:
      Serial.print("5V");
      break;
    case PD_SRC_9V:
      Serial.print("9V");
      break;
    case PD_SRC_12V:
      Serial.print("12V");
      break;
    case PD_SRC_15V:
      Serial.print("15V");
      break;
    case PD_SRC_18V:
      Serial.print("18V");
      break;
    case PD_SRC_20V:
      Serial.print("20V");
      break;
    default:
      continue;
  }
  Serial.print(voltageDetected ? " Available" : " Unavailable");

  // Loop over currents if voltage is detected
  if (voltageDetected) {
    HUSB238_CurrentSetting currentDetected =
```

```
husb238.currentDetected((HUSB238_PDSelection)i);
      Serial.print(" - Max current: ");
      printCurrentSetting(currentDetected);
    }
    Serial.println();
  }

  // Override whatever the jumpers on the board say, and get a specific voltage!
  husb238.selectPD(PD_SRC_5V);  // Select 5V
  // Uncomment one of the following lines to select a different PD:
  // husb238.selectPD(PD_SRC_9V);  // Select 9V
  // husb238.selectPD(PD_SRC_12V);  // Select 12V
  // husb238.selectPD(PD_SRC_15V);  // Select 15V
  // husb238.selectPD(PD_SRC_18V);  // Select 18V
  // husb238.selectPD(PD_SRC_20V);  // Select 20V

  // Perform the actual PD voltage request!
  husb238.requestPD();

  // Test getSelectedPD function
  HUSB238_PDSelection selectedPD = husb238.getSelectedPD();
  Serial.print("Currently Selected PD Output: ");
  switch (selectedPD) {
    case PD_NOT_SELECTED:
      Serial.println("Not Selected");
      break;
    case PD_SRC_5V:
      Serial.println("5V");
      break;
    case PD_SRC_9V:
      Serial.println("9V");
      break;
    case PD_SRC_12V:
      Serial.println("12V");
      break;
    case PD_SRC_15V:
      Serial.println("15V");
      break;
    case PD_SRC_18V:
      Serial.println("18V");
      break;
    case PD_SRC_20V:
      Serial.println("20V");
      break;
    default:
      Serial.println("Unknown");
      break;
  }
}


void printCurrentSetting(HUSB238_CurrentSetting srcCurrent) {
  switch (srcCurrent) {
    case CURRENT_0_5_A:
      Serial.print("0.5A ");
      break;
    case CURRENT_0_7_A:
      Serial.print("0.7A ");
      break;
    case CURRENT_1_0_A:
      Serial.print("1.0A ");
      break;
    case CURRENT_1_25_A:
      Serial.print("1.25A ");
      break;
    case CURRENT_1_5_A:
      Serial.print("1.5A ");
      break;
    case CURRENT_1_75_A:
```

```
      Serial.print("1.75A ");
      break;
    case CURRENT_2_0_A:
      Serial.print("2.0A ");
      break;
    case CURRENT_2_25_A:
      Serial.print("2.25A ");
      break;
    case CURRENT_2_50_A:
      Serial.print("2.50A ");
      break;
    case CURRENT_2_75_A:
      Serial.print("2.75A ");
      break;
    case CURRENT_3_0_A:
      Serial.print("3.0A ");
      break;
    case CURRENT_3_25_A:
      Serial.print("3.25A ");
      break;
    case CURRENT_3_5_A:
      Serial.print("3.5A ");
      break;
    case CURRENT_4_0_A:
      Serial.print("4.0A ");
      break;
    case CURRENT_4_5_A:
      Serial.print("4.5A ");
      break;
    case CURRENT_5_0_A:
      Serial.print("5.0A ");
      break;
    default:
      break;
  }
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the HUSB238 recognized over I2C by the code. It will set the voltage to 5V over I2C. Then, it will query the attached PD adapter to see which voltage and current combinations are available and print the currently set voltage.



Test All Voltages Example

For this example, it's best to test with the output from the breakout connected to a multimeter in DC voltage mode. DC voltage mode is labeled with a V and two lines, one dashed and one solid. For information on using a multimeter, check out this guide. (https://adafru.it/192A)

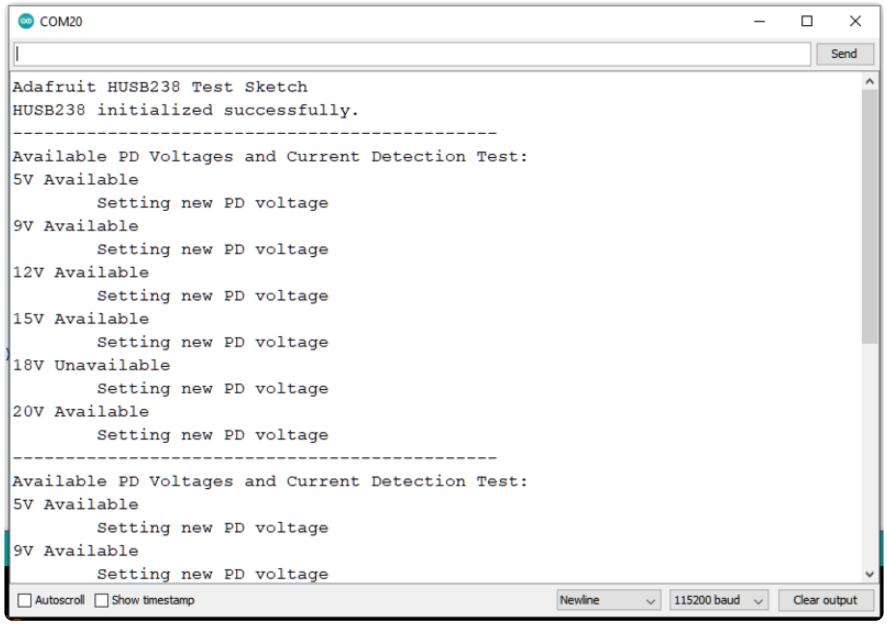
Multimeters Learn Guide

https://adafru.it/192A

```
#include <Wire.h>
#include "Adafruit_HUSB238.h"

Adafruit_HUSB238 husb238;

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);
  Serial.println("Adafruit HUSB238 Test Sketch");

  // Initialize the HUSB238
  if (husb238.begin(HUSB238_I2CADDR_DEFAULT, &Wire)) {
    Serial.println("HUSB238 initialized successfully.");
  } else {
    Serial.println("Couldn't find HUSB238, check your wiring?");
    while (1);
```

```
  }
}

void loop() {
  delay(1000);  // Add a delay to prevent flooding the serial output
  Serial.println(F("---------------------------------------------"));

  if (! husb238.isAttached())
    return;

  if (husb238.getPDResponse() != SUCCESS)
    return;

  // What voltages and currents are available from this adapter?
  Serial.println("Available PD Voltages and Current Detection Test:");
  for (int i = PD_SRC_5V; i <= PD_SRC_20V; i++) {
    bool voltageDetected = husb238.isVoltageDetected((HUSB238_PDSelection)i);

    switch ((HUSB238_PDSelection)i) {
      case PD_SRC_5V:
        Serial.print("5V");
        break;
      case PD_SRC_9V:
        Serial.print("9V");
        break;
      case PD_SRC_12V:
        Serial.print("12V");
        break;
      case PD_SRC_15V:
        Serial.print("15V");
        break;
      case PD_SRC_18V:
        Serial.print("18V");
        break;
      case PD_SRC_20V:
        Serial.print("20V");
        break;
      default:
        continue;
    }
    Serial.println(voltageDetected ? " Available" : " Unavailable");

    Serial.println("\tSetting new PD voltage");
    // Change to that voltage
    husb238.selectPD((HUSB238_PDSelection)i);
    // Perform the actual PD voltage request!
    husb238.requestPD();

    delay(2000);
  }
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the HUSB238 recognized over I2C by the code. Then, it will try setting a new voltage one after the other. As the Serial Monitor updates, you should see the same voltage read by your multimeter.



Arduino Docs

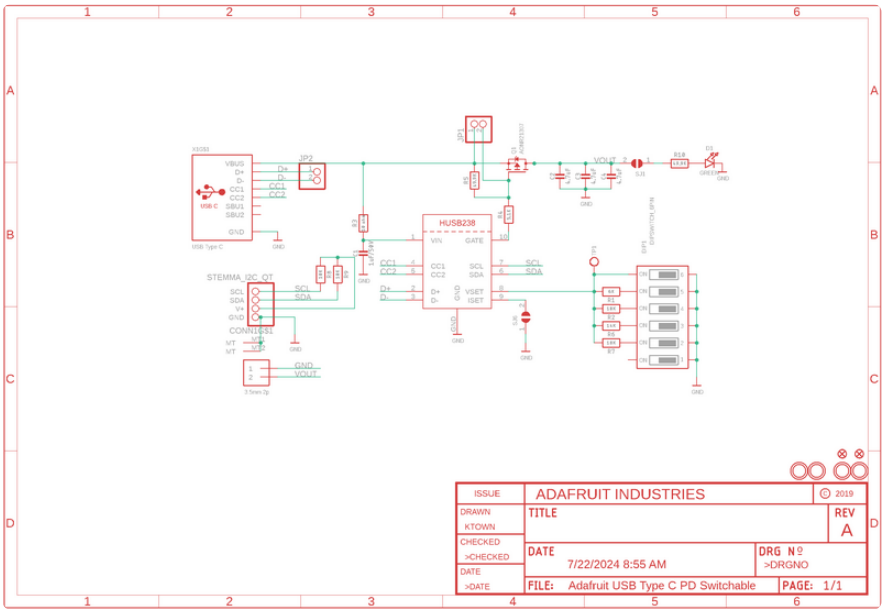
Arduino Docs (https://adafru.it/192D)

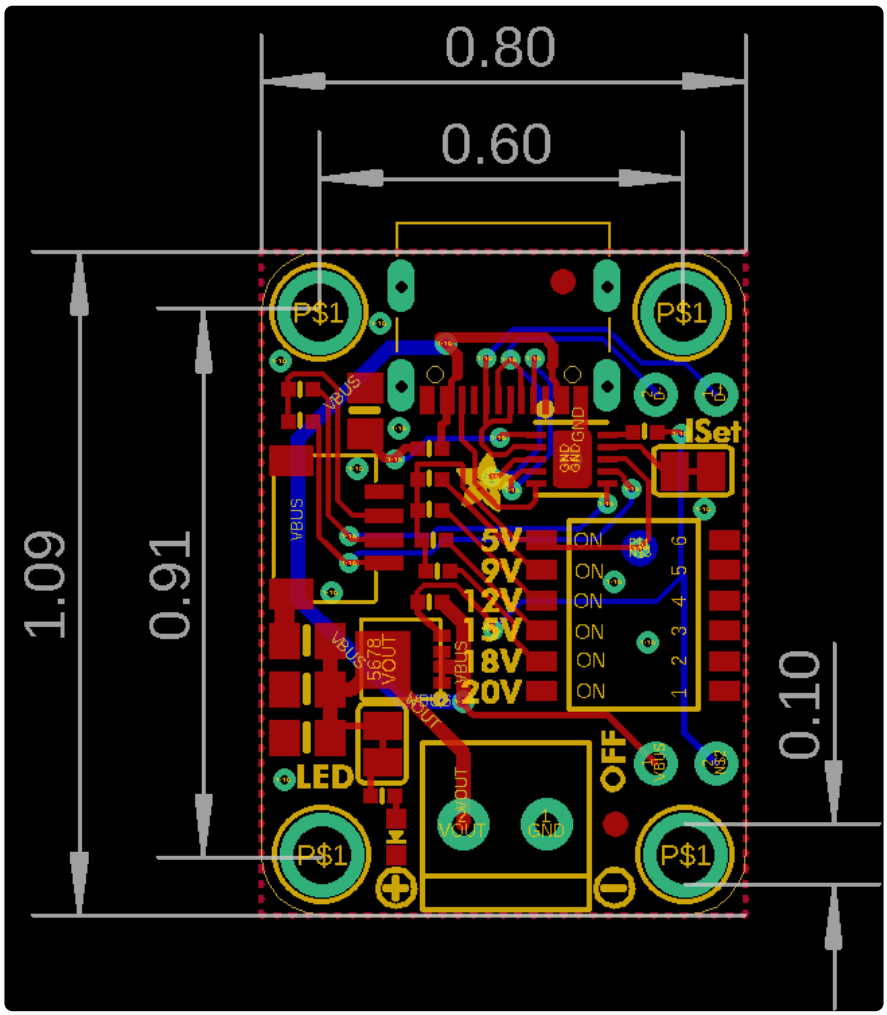
Downloads

Files

- HUSB238 Datasheet (https://adafru.it/192E)
- EagleCAD PCB Files on GitHub (https://adafru.it/1a4D)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1a4E)
- 3D Models on GitHub (https://adafru.it/1a6i)

Schematic and Fab Print





3D Model