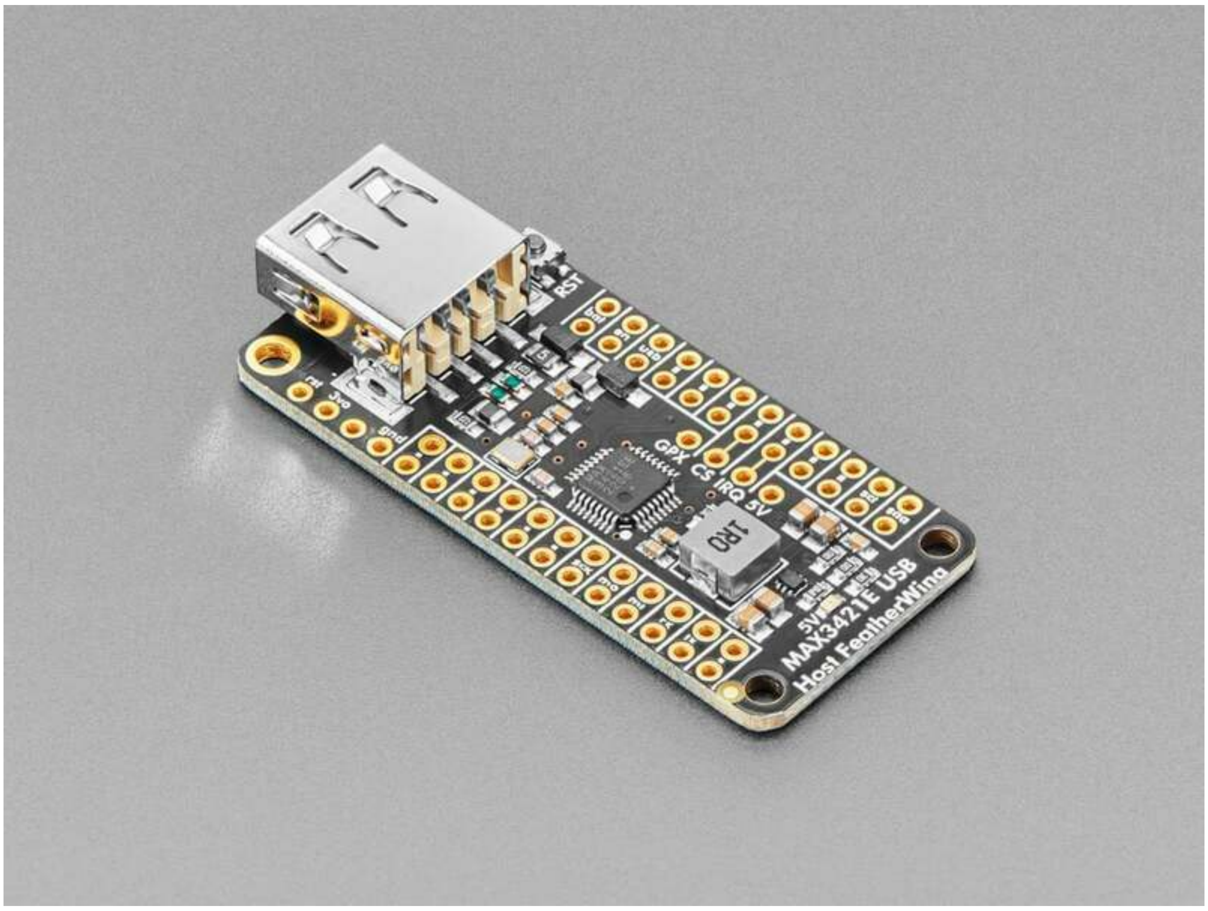
Adafruit USB Host FeatherWing with MAX3421E

Created by Liz Clark

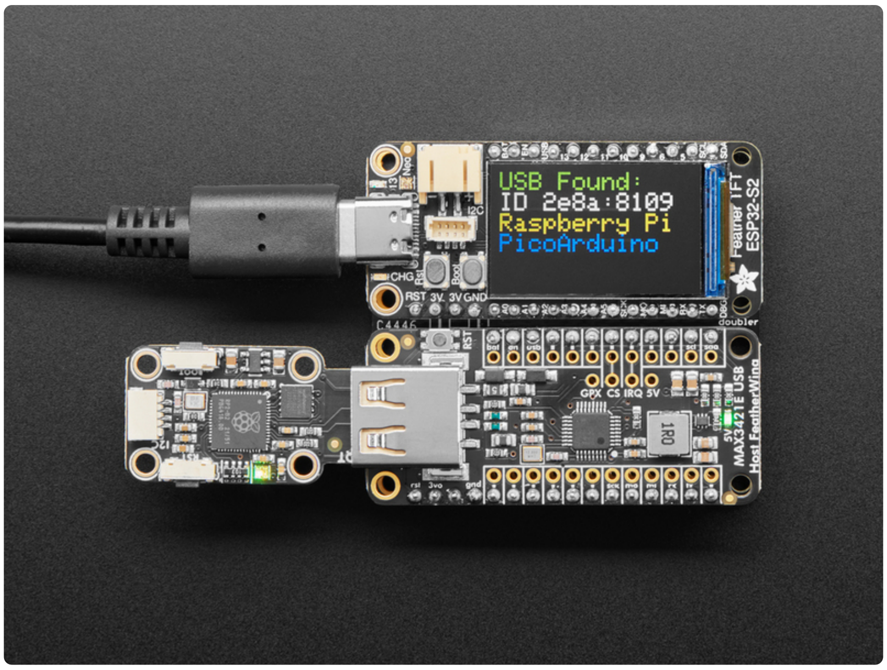


https://learn.adafruit.com/adafruit-usb-host-featherwing-with-max3421e

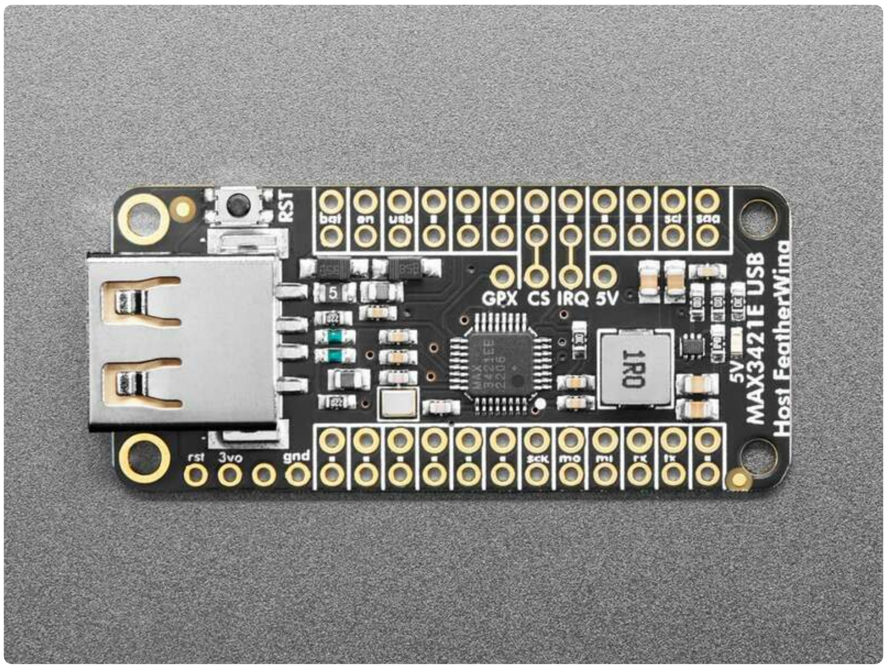
Last updated on 2024-01-26 10:41:48 AM EST

Table of Contents

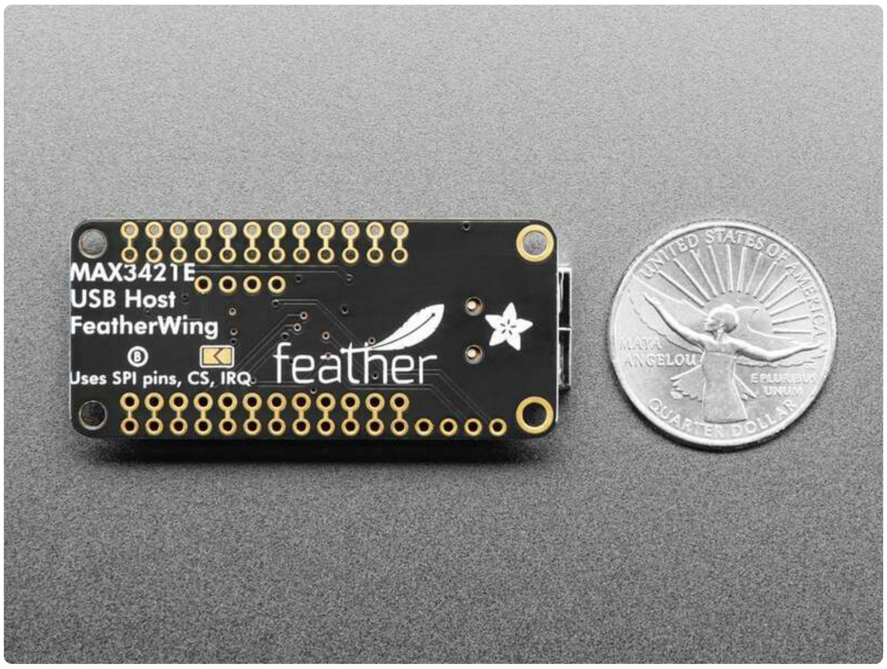
Overview



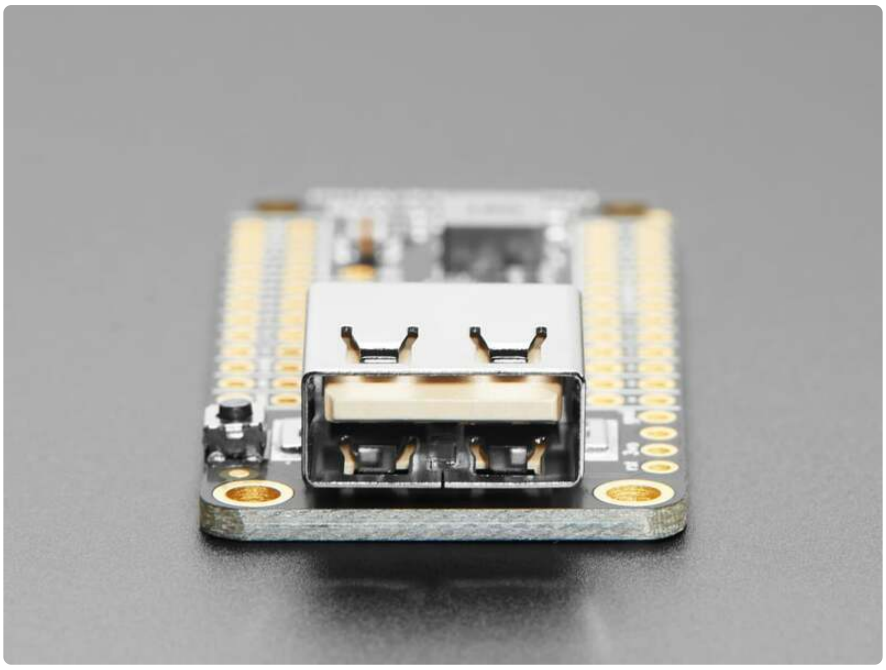
Lots of microcontrollers these days have USB ports on them, to program or debug, act like a keyboard or disk drive, or simply send data between a computer and your firmware. But did you know that you can also add a USB Host port? That means that your microcontroller project can have a keyboard or mouse or disk drive plugged into it - which opens up a huge ecosystem of common off-the-shelf devices that you can now integrate. The **Adafruit USB Host FeatherWing** makes it easy to add USB Host support, especially now that TinyUSB supports it in the Arduino library (https://adafru.it/EWc) as a 'native' interface for host support.



This 'Wing uses the MAX3421E - a tried and true USB Host chip. It uses SPI plus an IRQ pin to send data to just about any USB device. Note that because the chip is older, and you're limited to the SPI port speed, you're not going to get blazing 480Mbps high speed data transfer. But for basic HID interfacing, or even reading/writing to a Mass Storage device, it does work quite well. There's a famous USB Host Library that can be used (https://adafru.it/19be), and it's specialty is **AVR** support, but also seems to support nRF52 and ESP32. We personally recommend using the TinyUSB Arduino library (https://adafru.it/EWc) - however the trade-off is that the chip must have TinyUSB support already which means it's great for **RP2040, ESP32 or S2 or S3, nRF52840, SAMD21/51** chips. Between the two libraries, just make sure your desired Feather mainboard is supported before purchasing!

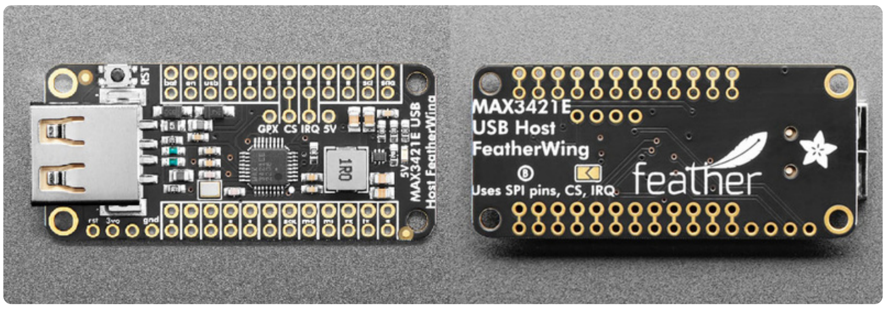


Next to the MAX3421E we have a 5V 1A booster with 500mA fuse, which can provide a nice clean 5V from the USB or Battery power supply. An enable pin is available to power cycle when desired.



It comes as an assembled 'Wing with some header. Solder on the header and plug into a Feather Mainboard to expand its capabilities! Don't forget, you need driver support for the MAX3421E (see above for chips that are known working) and unless you're using a generic mouse, keyboard, CDC serial or USB mass storage device you will also need a USB driver that knows how to talk to the device - and writing a driver is non-trivial.

Pinouts



USB-A Port

On the left end of the FeatherWing is the USB type A port for connecting USB devices to. This USB-A port can be used for sending or receiving data from an attached USB device with the MAX3421E.

MAX3421E Control Pins

The MAX3421E USB Host chip on the FeatherWing uses SPI plus an IRQ pin to send data to just about any USB device.

- **MOSI** - This is the SPI MOSI (**M**icrocontroller **O**ut / **S**erial **I**n) pin.
- **MISO** - This is the SPI MISO (**M**icrocontroller **I**n **S**erial **O**ut) pin.
- **SCK** - This is the **S**PI clo**ck** input pin.
- **CS** - This is the **c**hip **s**elect pin. It is connected to **pin 10** on the FeatherWing. It is also broken out towards the middle of the board.
- **IRQ** - This is the interrupt pin for the MAX3421E. It is connected to **pin 9** on the FeatherWing and is broken out towards the middle of the board.

GPX Pin

- **GPX** - Towards the center of the FeatherWing is the general purpose multiplexed push-pull output pin for the MAX3421E. This is for advanced users. The internal MAX3421E signal that appears on GPX is programmable by writing to the GPXB and GPXA bits of the PINCTL (R17) register and the SEPIRQ bit of the MODE (R27) register.

GPX indicates one of five signals:

GPXB	GPXA	GPX PIN
0	0	OPERATE (Default State)
0	1	VBUS_DET
1	0	BUSACT/INIRQ*
1	1	SOF

1. OPERATE: This signal goes high when the MAX3421E is able to operate after a power-up or RES reset. OPERATE is active when the RES input is high and the

internal power-on-reset (POR) is not asserted. OPERATE is the default GPX output.
2. VBUS_DET: VBUS_DET is the VBCOMP comparator output. This allows the user to directly monitor the VBUS status.
3. BUSACT: USB BUS activity signal (active high). This signal is active whenever there is traffic on the USB bus. The BUSACT signal is set whenever a SYNC field is detected. BUSACT goes low during bus reset or after 32-bit times of J-state.
4. INIRQ: When the SEPIRQ bit of the MODE (R27) register is set high, the BUSACT signal is removed from the INT output and GPX is used as an IRQ output pin dedicated to GPIN interrupts if GPX[B:A] = 10. In this mode, GPIN interrupts appear only on the GPX pin, and do not appear on the INT output pin.
5. SOF: A square wave with a positive edge that indicates the USB start-of-frame.

5V Enable Pin and Jumper

- **5V** - Towards the middle of the FeatherWing is a pin labeled **5V**. This is the **5V enable** output from the 5V 1A booster. If you tie this pin to ground, it can disable 5V power to the USB-A port, effectively turning off your connected USB device.
- **5V Enable Jumper** - On the back of the FeatherWing is an open jumper. If you solder this jumper closed you will connect the **5V enable** pin to **GPOUT0** (general purpose output 0) on the MAX3421E. This output is controllable over SPI, which means you can then enable or disable 5V to the USB-A port in software.

5V LED

- **5V LED** - Towards the right end of the FeatherWing is a green LED labeled 5V on the FeatherWing silk. It indicates whether the 5V power from the 5V 1A booster is enabled.

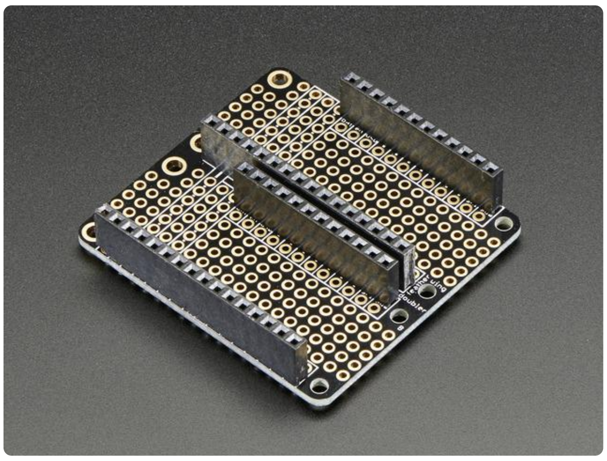
Reset Button

- **RST** - The reset button, located in the top left corner on the FeatherWing, is connected to the reset pin.

Arduino Library Install and Simple Test

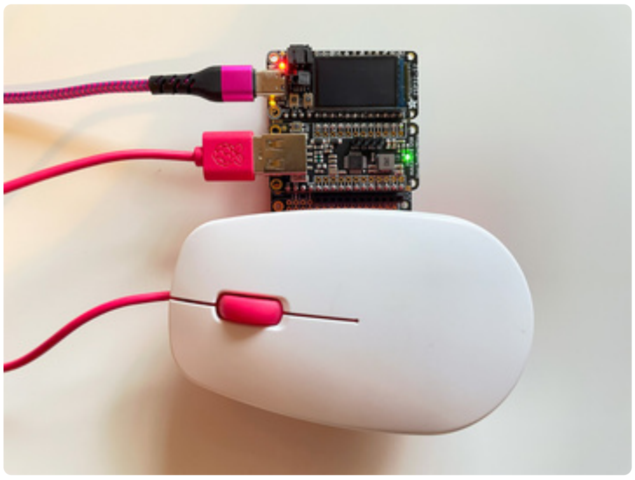
Using the USB Host FeatherWing with Arduino involves connecting your FeatherWing to an Arduino-compatible Feather microcontroller that has TinyUSB support (**ESP32, ESP32-S2 or ESP32-S3, nRF52840, SAMD21/51, RP2040** chips), plugging in a USB device to the USB-A port on the USB Host FeatherWing, installing the Adafruit_TinyUSB_Arduino (https://adafru.it/EWc) library, and running the provided example code.

For the RP2040, if you want to use the MAX3421E as the host driver, do not include 'pio_usb.h' in your sketch.



FeatherWing Doubler - Prototyping Add-on For All Feather Boards
This is the FeatherWing Doubler - a prototyping add-on and more for all Feather boards. This is similar to our https://www.adafruit.com/product/2890

Hardware



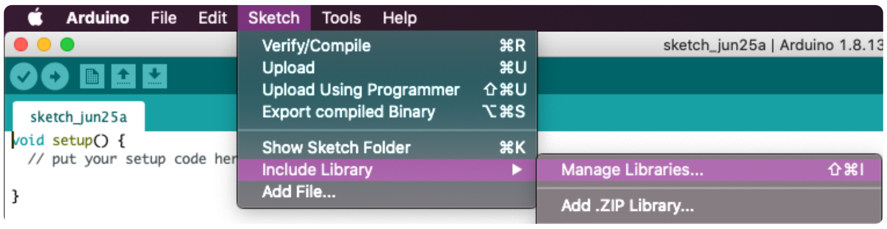
Plug in a TinyUSB supported Feather and the USB Host FeatherWing to a FeatherWing Doubler. Then, plug a USB device into the USB-A port on the USB Host FeatherWing.

The USB-A port is **only** for connecting USB devices. You cannot program the Feather with that port.

The USB-A port on the USB Host FeatherWing is only for USB devices. You cannot program the Feather with that port.

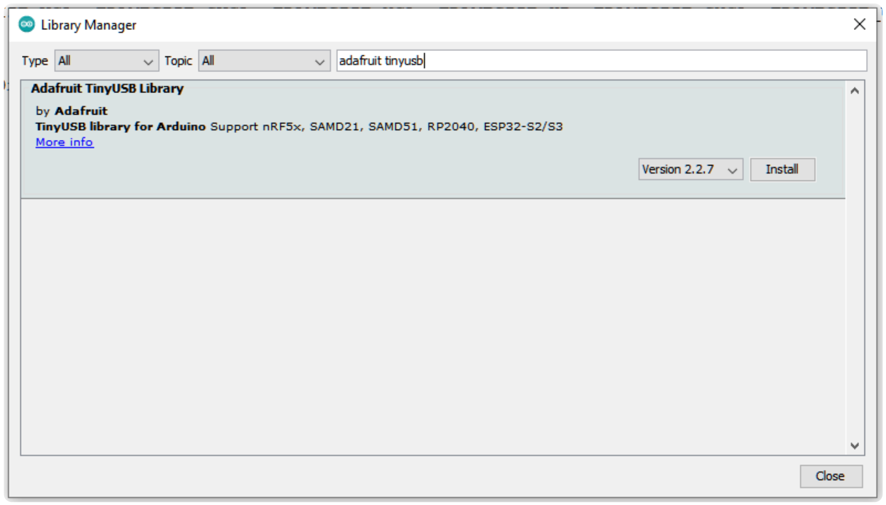
Library Installation

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



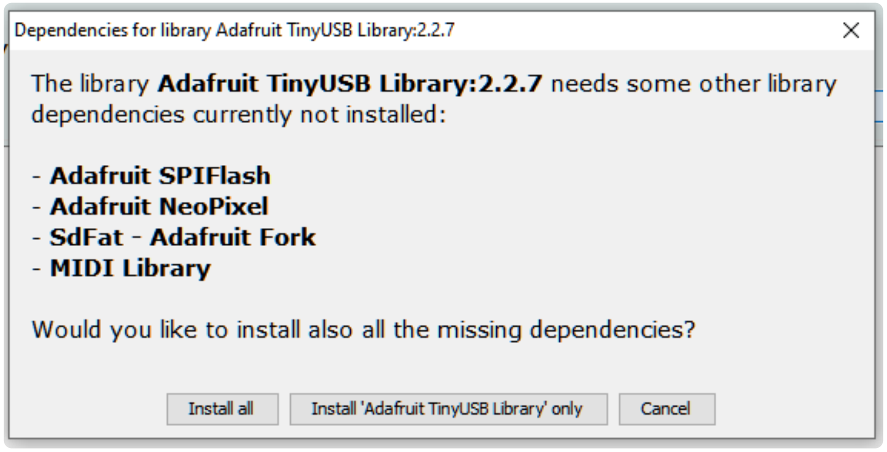
```
void setup() {
  // put your setup code her
}
```

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



Adafruit TinyUSB Library
by **Adafruit**
TinyUSB library for Arduino Support nRF5x, SAMD21, SAMD51, RP2040, ESP32-S2/S3
More info

If asked about dependencies, click "Install all".



Dependencies for library Adafruit TinyUSB Library:2.2.7

The library **Adafruit TinyUSB Library:2.2.7** needs some other library dependencies currently not installed:

- **Adafruit SPIFlash**
- **Adafruit NeoPixel**
- **SdFat - Adafruit Fork**
- **MIDI Library**

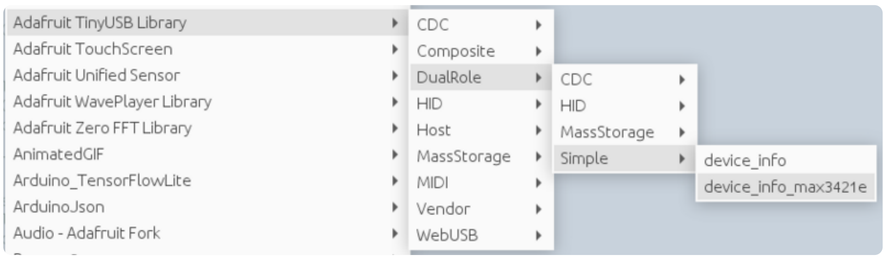
Would you like to install also all the missing dependencies?

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!

Device Info MAX3421E Example

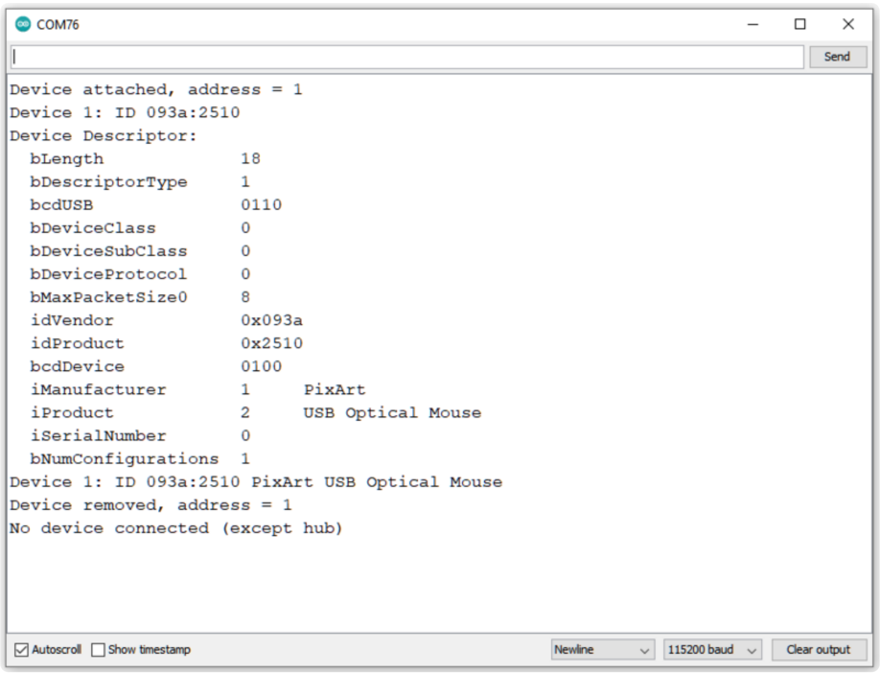
Navigate to the **Adafruit TinyUSB Library Examples** and select **DualRole -> Simple -> device_info_max3421e**



```
/*********************************************************************
 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!

 MIT license, check LICENSE for more information
 Copyright (c) 2019 Ha Thach for Adafruit Industries
 All text above, and the splash screen below must be included in
 any redistribution
*********************************************************************/

/* This example demonstrates use of both device and host, where
 * - Device run on native usb controller (roothub port0)
 * - Host run on MAX3421E controller (roothub port1) tested with:
 *   - SAMD21, SAMD51, nRF52840, ESP32S2, ESP32S3, ESP32
 *   - RP2040: "pio_usb.h" must not be included, otherwise pio-usb will be used as
host controller
 *
 * Requirements:
 * - SPI instance, CS pin, INT pin are correctly configured
 */

/* Host example will get device descriptors of attached devices and print it out:
 *    Device 1: ID 046d:c52f
      Device Descriptor:
        bLength             18
        bDescriptorType     1
        bcdUSB              0200
        bDeviceClass        0
        bDeviceSubClass     0
        bDeviceProtocol     0
        bMaxPacketSize0     8
        idVendor            0x046d
        idProduct           0xc52f
        bcdDevice           2200
        iManufacturer       1     Logitech
        iProduct            2     USB Receiver
        iSerialNumber       0
        bNumConfigurations  1
 *
 */
#include "Adafruit_TinyUSB.h"
#include "SPI.h"

// USB Host using MAX3421E: SPI, CS, INT
#if defined(ARDUINO_METRO_ESP32S2)
  Adafruit_USBH_Host USBHost(&SPI, 15, 14);
#elif defined(ARDUINO_ADAFRUIT_FEATHER_ESP32_V2)
  Adafruit_USBH_Host USBHost(&SPI, 33, 15);
```

```
#else
  // Default CS and INT are pin 10, 9
  Adafruit_USBH_Host USBHost(&SPI, 10, 9);
#endif

// Language ID: English
#define LANGUAGE_ID 0x0409

typedef struct {
  tusb_desc_device_t desc_device;
  uint16_t manufacturer[32];
  uint16_t product[48];
  uint16_t serial[16];
  bool mounted;
} dev_info_t;

// CFG_TUH_DEVICE_MAX is defined by tusb_config header
dev_info_t dev_info[CFG_TUH_DEVICE_MAX] = { 0 };

//--------------------------------------------------------------------+
// setup() & loop()
//--------------------------------------------------------------------+
void setup() {
  Serial.begin(115200);

  // init host stack on controller (rhport) 1
  USBHost.begin(1);

//  while ( !Serial ) delay(10);   // wait for native usb
  Serial.println("TinyUSB Dual: Device Info Example with MAX3421E");
}

void loop() {
  USBHost.task();
  Serial.flush();
}

//--------------------------------------------------------------------+
// TinyUSB Host callbacks
//--------------------------------------------------------------------+
void print_device_descriptor(tuh_xfer_t *xfer);

void utf16_to_utf8(uint16_t *temp_buf, size_t buf_len);

void print_lsusb(void) {
  bool no_device = true;
  for (uint8_t daddr = 1; daddr < CFG_TUH_DEVICE_MAX + 1; daddr++) {
    // TODO can use tuh_mounted(daddr), but tinyusb has an bug
    // use local connected flag instead
    dev_info_t *dev = &dev_info[daddr - 1];
    if (dev->mounted) {
      Serial.printf("Device %u: ID %04x:%04x %s %s\r\n", daddr,
                    dev->desc_device.idVendor, dev->desc_device.idProduct,
                    (char *) dev->manufacturer, (char *) dev->product);

      no_device = false;
    }
  }

  if (no_device) {
    Serial.println("No device connected (except hub)");
  }
}

// Invoked when device is mounted (configured)
void tuh_mount_cb(uint8_t daddr) {
  Serial.printf("Device attached, address = %d\r\n", daddr);

  dev_info_t *dev = &dev_info[daddr - 1];
  dev->mounted = true;

  // Get Device Descriptor
  tuh_descriptor_get_device(daddr, &dev->desc_device, 18, print_device_descriptor,
0);
}

/// Invoked when device is unmounted (bus reset/unplugged)
void tuh_umount_cb(uint8_t daddr) {
  Serial.printf("Device removed, address = %d\r\n", daddr);
  dev_info_t *dev = &dev_info[daddr - 1];
  dev->mounted = false;

  // print device summary
  print_lsusb();
}

void print_device_descriptor(tuh_xfer_t *xfer) {
  if (XFER_RESULT_SUCCESS != xfer->result) {
    Serial.printf("Failed to get device descriptor\r\n");
    return;
  }

  uint8_t const daddr = xfer->daddr;
  dev_info_t *dev = &dev_info[daddr - 1];
  tusb_desc_device_t *desc = &dev->desc_device;

  Serial.printf("Device %u: ID %04x:%04x\r\n", daddr, desc->idVendor, desc-
>idProduct);
  Serial.printf("Device Descriptor:\r\n");
  Serial.printf("  bLength             %u\r\n"     , desc->bLength);
  Serial.printf("  bDescriptorType     %u\r\n"     , desc->bDescriptorType);
  Serial.printf("  bcdUSB              %04x\r\n"   , desc->bcdUSB);
  Serial.printf("  bDeviceClass        %u\r\n"     , desc->bDeviceClass);
  Serial.printf("  bDeviceSubClass     %u\r\n"     , desc->bDeviceSubClass);
  Serial.printf("  bDeviceProtocol     %u\r\n"     , desc->bDeviceProtocol);
  Serial.printf("  bMaxPacketSize0     %u\r\n"     , desc->bMaxPacketSize0);
  Serial.printf("  idVendor            0x%04x\r\n" , desc->idVendor);
  Serial.printf("  idProduct           0x%04x\r\n" , desc->idProduct);
  Serial.printf("  bcdDevice           %04x\r\n"   , desc->bcdDevice);

  // Get String descriptor using Sync API
  Serial.printf("  iManufacturer       %u     ", desc->iManufacturer);
  if (XFER_RESULT_SUCCESS ==
      tuh_descriptor_get_manufacturer_string_sync(daddr, LANGUAGE_ID, dev-
>manufacturer, sizeof(dev->manufacturer))) {
    utf16_to_utf8(dev->manufacturer, sizeof(dev->manufacturer));
    Serial.printf((char *) dev->manufacturer);
  }
  Serial.printf("\r\n");

  Serial.printf("  iProduct            %u     ", desc->iProduct);
  if (XFER_RESULT_SUCCESS ==
      tuh_descriptor_get_product_string_sync(daddr, LANGUAGE_ID, dev->product,
sizeof(dev->product))) {
    utf16_to_utf8(dev->product, sizeof(dev->product));
    Serial.printf((char *) dev->product);
  }
  Serial.printf("\r\n");

  Serial.printf("  iSerialNumber       %u     ", desc->iSerialNumber);
  if (XFER_RESULT_SUCCESS ==
      tuh_descriptor_get_serial_string_sync(daddr, LANGUAGE_ID, dev->serial,
sizeof(dev->serial))) {
    utf16_to_utf8(dev->serial, sizeof(dev->serial));
    Serial.printf((char *) dev->serial);
  }
  Serial.printf("\r\n");
  Serial.printf("  bNumConfigurations  %u\r\n", desc->bNumConfigurations);

  // print device summary
  print_lsusb();
}

//--------------------------------------------------------------------+
// String Descriptor Helper
//--------------------------------------------------------------------+

static void _convert_utf16le_to_utf8(const uint16_t *utf16, size_t utf16_len,
uint8_t *utf8, size_t utf8_len) {
  // TODO: Check for runover.
  (void) utf8_len;
  // Get the UTF-16 length out of the data itself.

  for (size_t i = 0; i < utf16_len; i++) {
    uint16_t chr = utf16[i];
    if (chr < 0x80) {
      *utf8++ = chr & 0xff;
    } else if (chr < 0x800) {
      *utf8++ = (uint8_t) (0xC0 | (chr >> 6 & 0x1F));
      *utf8++ = (uint8_t) (0x80 | (chr >> 0 & 0x3F));
    } else {
      // TODO: Verify surrogate.
      *utf8++ = (uint8_t) (0xE0 | (chr >> 12 & 0x0F));
      *utf8++ = (uint8_t) (0x80 | (chr >> 6 & 0x3F));
      *utf8++ = (uint8_t) (0x80 | (chr >> 0 & 0x3F));
    }
    // TODO: Handle UTF-16 code points that take two entries.
  }
}

// Count how many bytes a utf-16-le encoded string will take in utf-8.
static int _count_utf8_bytes(const uint16_t *buf, size_t len) {
  size_t total_bytes = 0;
  for (size_t i = 0; i < len; i++) {
    uint16_t chr = buf[i];
    if (chr < 0x80) {
      total_bytes += 1;
    } else if (chr < 0x800) {
      total_bytes += 2;
    } else {
      total_bytes += 3;
    }
    // TODO: Handle UTF-16 code points that take two entries.
  }
  return total_bytes;
}

void utf16_to_utf8(uint16_t *temp_buf, size_t buf_len) {
  size_t utf16_len = ((temp_buf[0] & 0xff) - 2) / sizeof(uint16_t);
  size_t utf8_len = _count_utf8_bytes(temp_buf + 1, utf16_len);

  _convert_utf16le_to_utf8(temp_buf + 1, utf16_len, (uint8_t *) temp_buf, buf_len);
  ((uint8_t *) temp_buf)[utf8_len] = '\0';
}
```

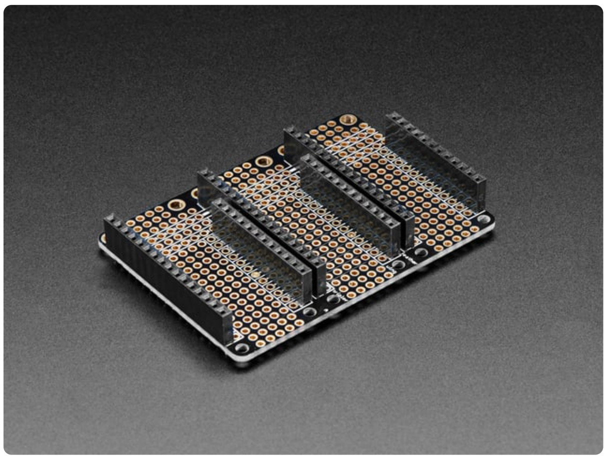
Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. As you plug and unplug your USB device to the USB Host FeatherWing, you will see the device info print out to the Serial Monitor.



```
Device attached, address = 1
Device 1: ID 093a:2510
Device Descriptor:
  bLength             18
  bDescriptorType     1
  bcdUSB              0110
  bDeviceClass        0
  bDeviceSubClass     0
  bDeviceProtocol     0
  bMaxPacketSize0     8
  idVendor            0x093a
  idProduct           0x2510
  bcdDevice           0100
  iManufacturer       1     PixArt
  iProduct            2     USB Optical Mouse
  iSerialNumber       0
  bNumConfigurations  1
Device 1: ID 093a:2510 PixArt USB Optical Mouse
Device removed, address = 1
No device connected (except hub)
```

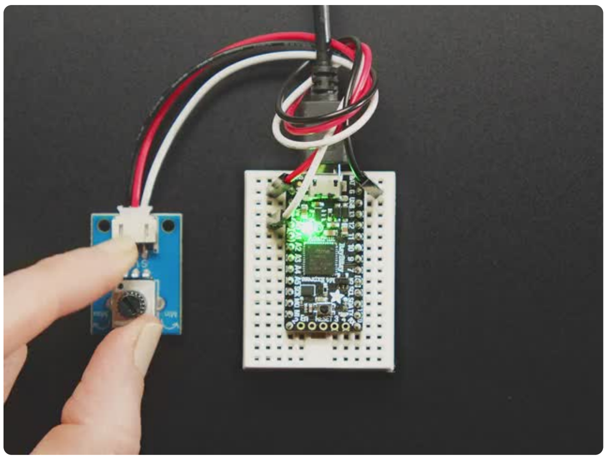
Arduino Data Logger Example

In the data logger example, you'll log data from a potentiometer connected to pin A0 on your TinyUSB supported Feather to a USB drive plugged into the USB Host FeatherWing.



FeatherWing Tripler Mini Kit - Prototyping Add-on For Feathers

This is the FeatherWing Tripler - a prototyping add-on and more for all Feather boards. This is similar to our https://www.adafruit.com/product/3417



STEMMA Wired Potentiometer Breakout Board - 10K ohm Linear

For the easiest way possible to measure twists, turn to this STEMMA potentiometer breakout (ha!). This plug-n-play pot comes with a JST-PH 2mm connector and a matching

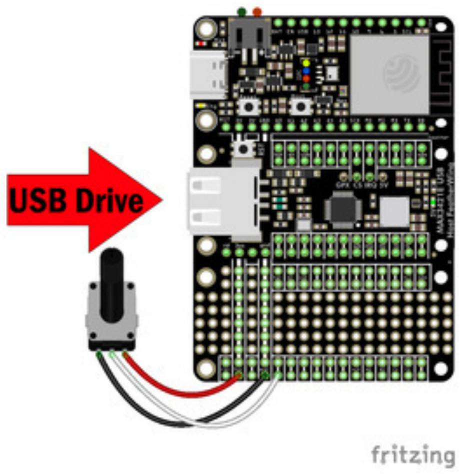
https://www.adafruit.com/product/4493

STEMMA JST PH 2mm 3-Pin to Male Header Cable - 200mm

This cable will let you turn a JST PH 3-pin cable port into 3 individual wires with high-quality 0.1" male header plugs on the end. We're carrying these to match up with our...

https://www.adafruit.com/product/3893

Wiring



Plug a TinyUSB supported Feather and USB Host FeatherWing into a FeatherWing Tripler

Connect a potentiometer to the FeatherWing Tripler

Potentiometer ground to **Feather GND**

Potentiometer wiper to **Feather A0**

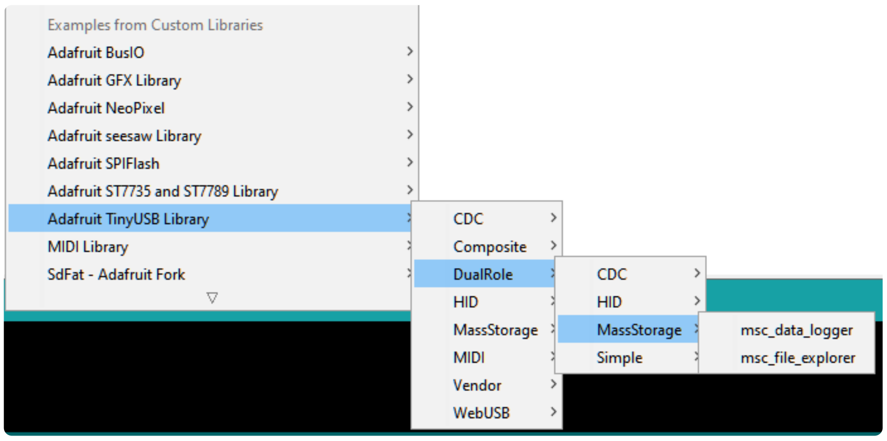
Potentiometer voltage to **Feather 3.3V**

Plug a USB drive into the USB Host FeatherWing

Data Logger Example

Load the example code onto your Feather after following the library installation instructions on the Arduino Library Install page (https://adafru.it/19bf).

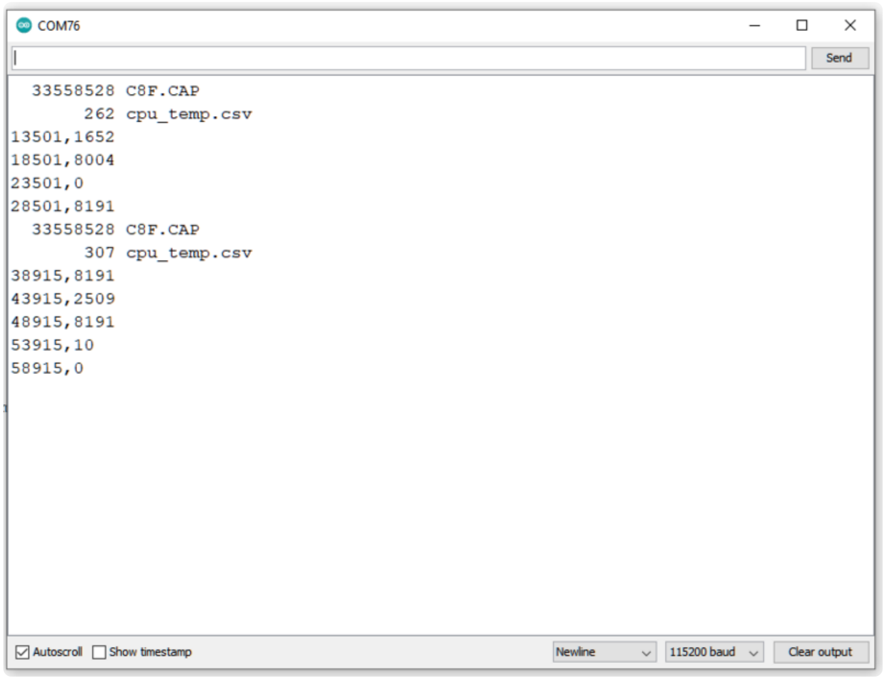
Navigate to the **Adafruit TinyUSB Library Examples** and select **DualRole - MassStorage - msc_data_logger**



```
/*********************************************************************
 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!

 MIT license, check LICENSE for more information
 Copyright (c) 2019 Ha Thach for Adafruit Industries
 All text above, and the splash screen below must be included in
 any redistribution
*********************************************************************/


/* This example demonstrates use of both device and host, where
 * - Device run on native usb controller (roothub port0)
 * - Host depending on MCUs run on either:
 *   - rp2040: bit-banging 2 GPIOs with the help of Pico-PIO-USB library (roothub
port1)
 *   - samd21/51, nrf52840, esp32: using MAX3421e controller (host shield)
 *
 * Requirements:
 * - For rp2040:
 *   - [Pico-PIO-USB](https://github.com/sekigon-gonnoc/Pico-PIO-USB) library
 *   - 2 consecutive GPIOs: D+ is defined by PIN_USB_HOST_DP, D- = D+ +1
 *   - Provide VBus (5v) and GND for peripheral
 *   - CPU Speed must be either 120 or 240 Mhz. Selected via "Menu -> CPU Speed"
 * - For samd21/51, nrf52840, esp32:
 *   - Additional MAX2341e USB Host shield or featherwing is required
 *   - SPI instance, CS pin, INT pin are correctly configured in usbh_helper.h
 */

/* Example sketch read analog pin (default A0) and log it to LOG_FILE on the msc
device
 * every LOG_INTERVAL ms. */

// nRF52 and ESP32 use freeRTOS, we may need to run USBhost.task() in its own
rtos's thread.
// Since USBHost.task() will put loop() into dormant state and prevent followed
code from running
// until there is USB host event.
#if defined(ARDUINO_NRF52_ADAFRUIT) || defined(ARDUINO_ARCH_ESP32)
  #define USE_FREERTOS
#endif

// SdFat is required for using Adafruit_USBH_MSC_SdFatDevice
#include "SdFat.h"

// USBHost is defined in usbh_helper.h
```

```
#include "usbh_helper.h"

#define LOG_FILE        "cpu_temp.csv"
#define LOG_INTERVAL    5000

// Analog pin for reading
const int analogPin = A0;

// USB Host MSC Block Device object which implemented API for use with SdFat
Adafruit_USBH_MSC_BlockDevice msc_block_dev;

// file system object from SdFat
FatVolume fatfs;
File32 f_log;

// if file system is successfully mounted on usb block device
volatile bool is_mounted = false;

void data_log(void) {
  if (!is_mounted) {
    // nothing to do
    return;
  }

  static unsigned long last_ms = 0;
  unsigned long ms = millis();

  if ( ms - last_ms < LOG_INTERVAL ) {
    return;
  }

  // Turn on LED when start writing
  digitalWrite(LED_BUILTIN, HIGH);

  f_log = fatfs.open(LOG_FILE, O_WRITE | O_APPEND | O_CREAT);

  if (!f_log) {
    Serial.println("Cannot create file: " LOG_FILE);
  } else {
    int value = analogRead(analogPin);

    Serial.printf("%lu,%d\r\n", ms, value);
    f_log.printf("%lu,%d\r\n", ms, value);

    f_log.close();
  }

  last_ms = ms;
  Serial.flush();
}

#ifdef USE_FREERTOS

#ifdef ARDUINO_ARCH_ESP32
  #define USBH_STACK_SZ 2048
#else
  #define USBH_STACK_SZ 200
#endif

void usbhost_rtos_task(void *param) {
  (void) param;
  while (1) {
    USBHost.task();
  }
}

#endif

void setup() {
  Serial.begin(115200);

  pinMode(LED_BUILTIN, OUTPUT);

#if defined(CFG_TUH_MAX3421) && CFG_TUH_MAX3421
  // init host stack on controller (rhport) 1
  // For rp2040: this is called in core1's setup1()
  USBHost.begin(1);
#endif

#ifdef USE_FREERTOS
  // Create a task to run USBHost.task() in background
  xTaskCreate(usbhost_rtos_task, "usbh", USBH_STACK_SZ, NULL, 3, NULL);
#endif

//  while ( !Serial ) delay(10);   // wait for native usb
  Serial.println("TinyUSB Host MassStorage Data Logger Example");
}

#if defined(CFG_TUH_MAX3421) && CFG_TUH_MAX3421
//--------------------------------------------------------------------+
// Using Host shield MAX3421E controller
//--------------------------------------------------------------------+
void loop() {
#ifndef USE_FREERTOS
  USBHost.task();
#endif
  data_log();
}

#elif defined(ARDUINO_ARCH_RP2040)
//--------------------------------------------------------------------+
// For RP2040 use both core0 for device stack, core1 for host stack
//--------------------------------------------------------------------+
void loop() {
  data_log();
}

//------------- Core1 -------------//
void setup1() {
  // configure pio-usb: defined in usbh_helper.h
  rp2040_configure_pio_usb();

  // run host stack on controller (rhport) 1
  // Note: For rp2040 pico-pio-usb, calling USBHost.begin() on core1 will have most
of the
  // host bit-banging processing works done in core1 to free up core0 for other
works
  USBHost.begin(1);
}

void loop1() {
  USBHost.task();
}

#endif

//--------------------------------------------------------------------+
// TinyUSB Host callbacks
//--------------------------------------------------------------------+
bool write_complete_callback(uint8_t dev_addr, tuh_msc_complete_data_t const
*cb_data) {
  (void) dev_addr;
  (void) cb_data;

  // turn off LED after write is complete
  // Note this only marks the usb transfer is complete, device can take longer to
actual
  // write data to physical flash
  digitalWrite(LED_BUILTIN, LOW);

  return true;
}

extern "C"
{

// Invoked when device is mounted (configured)
void tuh_mount_cb(uint8_t daddr) {
  (void) daddr;
}

/// Invoked when device is unmounted (bus reset/unplugged)
void tuh_umount_cb(uint8_t daddr) {
  (void) daddr;
}

// Invoked when a device with MassStorage interface is mounted
void tuh_msc_mount_cb(uint8_t dev_addr) {
  // Initialize block device with MSC device address
  msc_block_dev.begin(dev_addr);

  // For simplicity this example only support LUN 0
  msc_block_dev.setActiveLUN(0);
  msc_block_dev.setWriteCompleteCallback(write_complete_callback);
  is_mounted = fatfs.begin(&msc_block_dev);

  if (is_mounted) {
    fatfs.ls(&Serial, LS_SIZE);
  } else {
    Serial.println("Failed to mount mass storage device. Make sure it is formatted
as FAT");
  }
}

// Invoked when a device with MassStorage interface is unmounted
void tuh_msc_umount_cb(uint8_t dev_addr) {
  (void) dev_addr;

  // unmount file system
  is_mounted = false;
  fatfs.end();

  // end block device
  msc_block_dev.end();
}

}
```

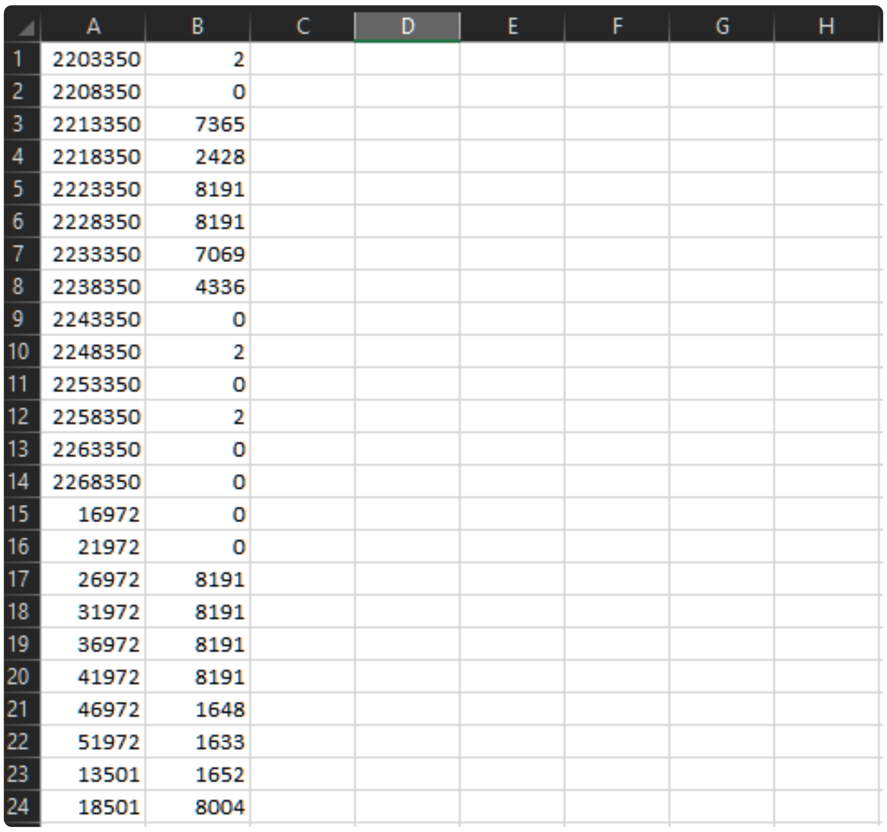
After uploading the sketch to your board, open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. After you plug in your USB drive to the USB Host FeatherWing, you will the files currently on the drive print out to the Serial Monitor. As you turn the potentiometer, you'll see the values alongside a timestamp print to the Serial Monitor.



```
  33558528 C8F.CAP
       262 cpu_temp.csv
13501,1652
18501,8004
23501,0
28501,8191
  33558528 C8F.CAP
       307 cpu_temp.csv
38915,8191
43915,2509
48915,8191
53915,10
58915,0
```

Newline | 115200 baud

The values are saved to a file called **cpu_temp.csv**. If you open the CSV file on your computer, you'll see the logged values.

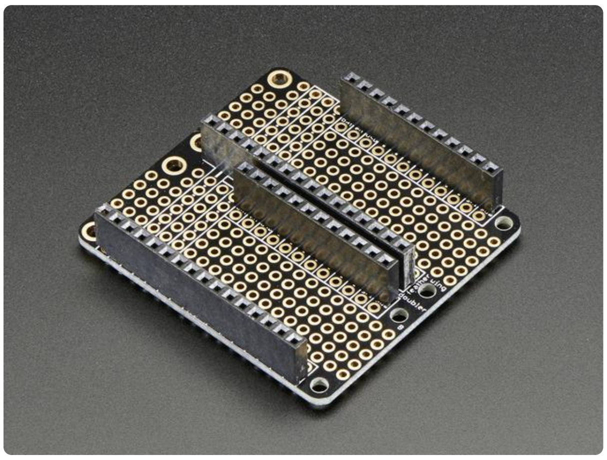


	A	B	C	D	E	F	G	H
1	2203350	2						
2	2208350	0						
3	2213350	7365						
4	2218350	2428						
5	2223350	8191						
6	2228350	8191						
7	2233350	7069						
8	2238350	4336						
9	2243350	0						
10	2248350	2						
11	2253350	0						
12	2258350	2						
13	2263350	0						
14	2268350	0						
15	16972	0						
16	21972	0						
17	26972	8191						
18	31972	8191						
19	36972	8191						
20	41972	8191						
21	46972	1648						
22	51972	1633						
23	13501	1652						
24	18501	8004						

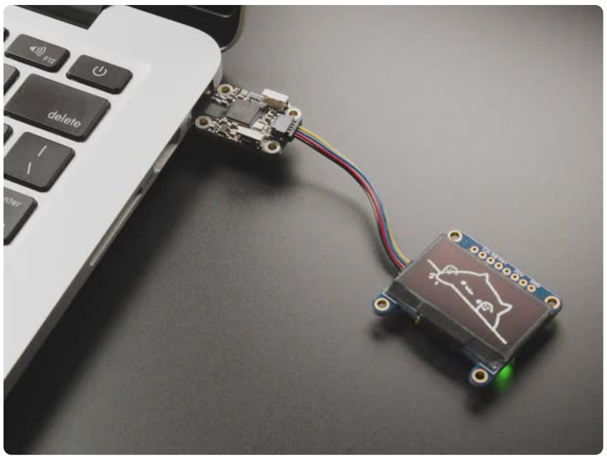
Arduino Serial Host Bridge Example

In the serial host bridge example, you'll see serial messages printed out from another microcontroller plugged into your USB Host FeatherWing.

Right now this example does not run on ESP32 family boards.



FeatherWing Doubler - Prototyping Add-on For All Feather Boards
This is the FeatherWing Doubler - a prototyping add-on and more for all Feather boards. This is similar to our
https://www.adafruit.com/product/2890



Adafruit Trinkey QT2040 - RP2040 USB Key with Stemma QT
It's half USB Key, half Adafruit QT Py, and a lotta RP2040...it's Trinkey QT2040, the circuit board with an RP2040 heart and Stemma QT legs....
https://www.adafruit.com/product/5056

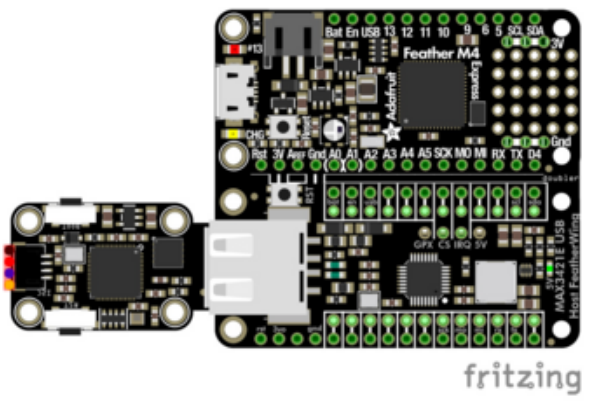
Simple Serial Printing Sketch

First, load a sketch onto the microcontroller that you'll be plugging into the USB Host FeatherWing that prints something to the Serial Monitor. This quick example will print `hello world from another arduino!` every second.

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  Serial.println("hello world from another arduino!");
  delay(1000);
}
```

Hardware

Next, you can prepare the hardware for the demo.



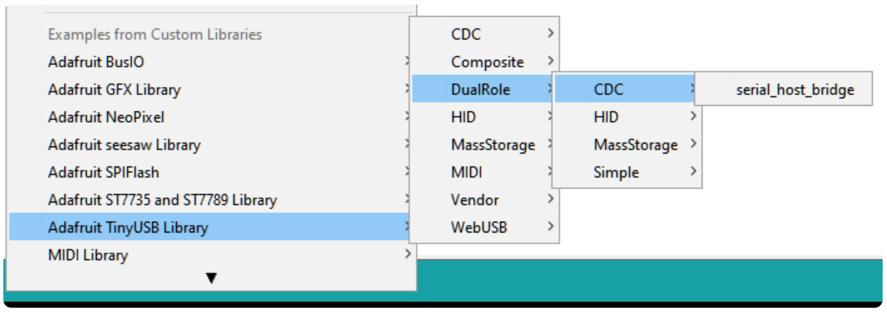
Plug a non-ESP32 TinyUSB supported Feather and USB Host FeatherWing into a FeatherWing Doubler

Plug a microcontroller running an Arduino program that prints data to the Serial Monitor into the USB Host FeatherWing. A Trinkey QT2040 is shown in the Fritzing diagram.

Serial Host Bridge Example

Load the example code onto your Feather after following the library installation instructions on the Arduino Library Install page (https://adafru.it/19bf).

Navigate to the **Adafruit TinyUSB Library Examples** and select **DualRole - CDC - serial_host_bridge**

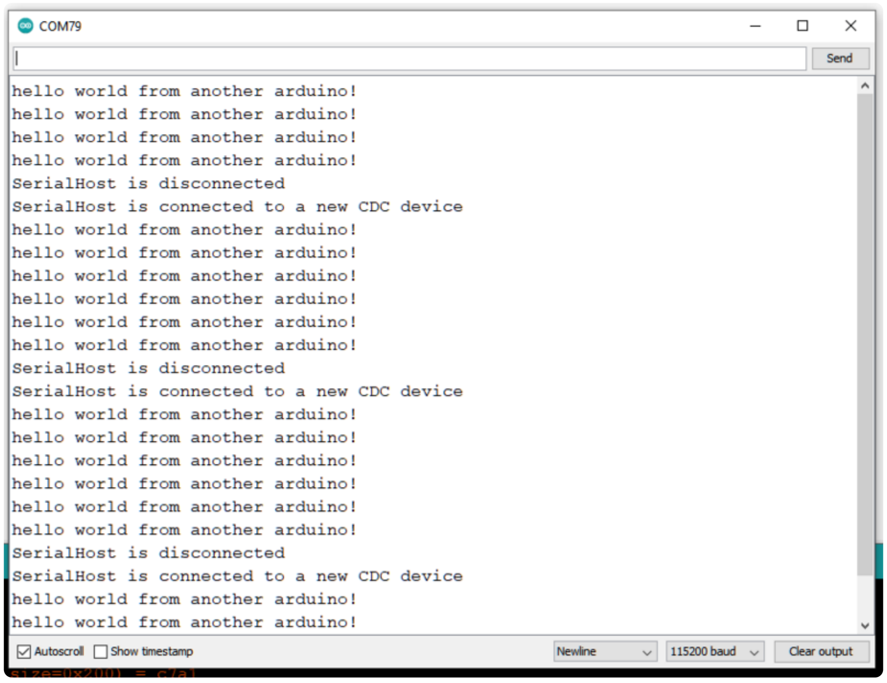


```
/*********************************************************************
 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!

 MIT license, check LICENSE for more information
 Copyright (c) 2019 Ha Thach for Adafruit Industries
 All text above, and the splash screen below must be included in
 any redistribution
*********************************************************************/

/* This example demonstrates use of both device and host, where
 * - Device run on native usb controller (roothub port0)
 * - Host depending on MCUs run on either:
 *   - rp2040: bit-banging 2 GPIOs with the help of Pico-PIO-USB library (roothub
port1)
 *   - samd21/51, nrf52840, esp32: using MAX3421e controller (host shield)
 *
 * Requirements:
 * - For rp2040:
 *   - [Pico-PIO-USB](https://github.com/sekigon-gonnoc/Pico-PIO-USB) library
 *   - 2 consecutive GPIOs: D+ is defined by PIN_USB_HOST_DP, D- = D+ +1
 *   - Provide VBus (5v) and GND for peripheral
 *   - CPU Speed must be either 120 or 240 Mhz. Selected via "Menu -> CPU Speed"
 * - For samd21/51, nrf52840, esp32:
 *   - Additional MAX2341e USB Host shield or featherwing is required
 *   - SPI instance, CS pin, INT pin are correctly configured in usbh_helper.h
 */

/* This example demonstrates use of Host Serial (CDC). SerialHost (declared below)
is
 * an object to manage an CDC peripheral connected to our USB Host connector. This
example
 * will forward all characters from Serial to SerialHost and vice versa.
 */

// nRF52 and ESP32 use freeRTOS, we may need to run USBhost.task() in its own
rtos's thread.
// Since USBHost.task() will put loop() into dormant state and prevent followed
code from running
// until there is USB host event.
#if defined(ARDUINO_NRF52_ADAFRUIT) || defined(ARDUINO_ARCH_ESP32)
  #define USE_FREERTOS
#endif

// USBHost is defined in usbh_helper.h
#include "usbh_helper.h"

// CDC Host object
Adafruit_USBH_CDC SerialHost;

// forward Seral <-> SerialHost
void forward_serial(void) {
  uint8_t buf[64];

  // Serial -> SerialHost
  if (Serial.available()) {
    size_t count = Serial.read(buf, sizeof(buf));
    if (SerialHost && SerialHost.connected()) {
      SerialHost.write(buf, count);
      SerialHost.flush();
    }
  }

  // SerialHost -> Serial
  if (SerialHost.connected() && SerialHost.available()) {
    size_t count = SerialHost.read(buf, sizeof(buf));
    Serial.write(buf, count);
    Serial.flush();
  }
}

#if defined(CFG_TUH_MAX3421) && CFG_TUH_MAX3421
//--------------------------------------------------------------------+
// Using Host shield MAX3421E controller
//--------------------------------------------------------------------+

#ifdef USE_FREERTOS

#ifdef ARDUINO_ARCH_ESP32
  #define USBH_STACK_SZ 2048
#else
  #define USBH_STACK_SZ 200
#endif

void usbhost_rtos_task(void *param) {
  (void) param;
  while (1) {
    USBHost.task();
  }
}
#endif
```

```
void setup() {
  Serial.begin(115200);

  // init host stack on controller (rhport) 1
  USBHost.begin(1);

  // Initialize SerialHost
  SerialHost.begin(115200);

#ifdef USE_FREERTOS
  // Create a task to run USBHost.task() in background
  xTaskCreate(usbhost_rtos_task, "usbh", USBH_STACK_SZ, NULL, 3, NULL);
#endif

//  while ( !Serial ) delay(10);   // wait for native usb
  Serial.println("TinyUSB Host Serial Echo Example");
}

void loop() {
#ifndef USE_FREERTOS
  USBHost.task();
#endif

  forward_serial();
}

#elif defined(ARDUINO_ARCH_RP2040)
//--------------------------------------------------------------------+
// For RP2040 use both core0 for device stack, core1 for host stack
//--------------------------------------------------------------------+

//------------- Core0 -------------//
void setup() {
  Serial.begin(115200);
  // while ( !Serial ) delay(10);   // wait for native usb
  Serial.println("TinyUSB Host Serial Echo Example");
}

void loop() {
  forward_serial();
}

//------------- Core1 -------------//
void setup1() {
  // configure pio-usb: defined in usbh_helper.h
  rp2040_configure_pio_usb();

  // run host stack on controller (rhport) 1
  // Note: For rp2040 pico-pio-usb, calling USBHost.begin() on core1 will have most 
of the
  // host bit-banging processing works done in core1 to free up core0 for other 
works
  USBHost.begin(1);

  // Initialize SerialHost
  SerialHost.begin(115200);
}

void loop1() {
  USBHost.task();
}

#endif

//--------------------------------------------------------------------+
// TinyUSB Host callbacks
//--------------------------------------------------------------------+
extern "C" {
```

```
// Invoked when a device with CDC interface is mounted
// idx is index of cdc interface in the internal pool.
void tuh_cdc_mount_cb(uint8_t idx) {
  // bind SerialHost object to this interface index
  SerialHost.mount(idx);
  Serial.println("SerialHost is connected to a new CDC device");
}

// Invoked when a device with CDC interface is unmounted
void tuh_cdc_umount_cb(uint8_t idx) {
  SerialHost.umount(idx);
  Serial.println("SerialHost is disconnected");
}

}
```

After uploading the sketch to your board, open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. After you plug in your additional microcontroller to the USB Host FeatherWing, you see it mount and then print to the Serial Monitor thru the USB Host FeatherWing.



```
hello world from another arduino!
hello world from another arduino!
hello world from another arduino!
hello world from another arduino!
SerialHost is disconnected
SerialHost is connected to a new CDC device
hello world from another arduino!
hello world from another arduino!
hello world from another arduino!
hello world from another arduino!
hello world from another arduino!
hello world from another arduino!
SerialHost is disconnected
SerialHost is connected to a new CDC device
hello world from another arduino!
hello world from another arduino!
hello world from another arduino!
hello world from another arduino!
hello world from another arduino!
hello world from another arduino!
SerialHost is disconnected
SerialHost is connected to a new CDC device
hello world from another arduino!
hello world from another arduino!
```

Arduino Docs

Arduino Docs (https://adafru.it/19bg)

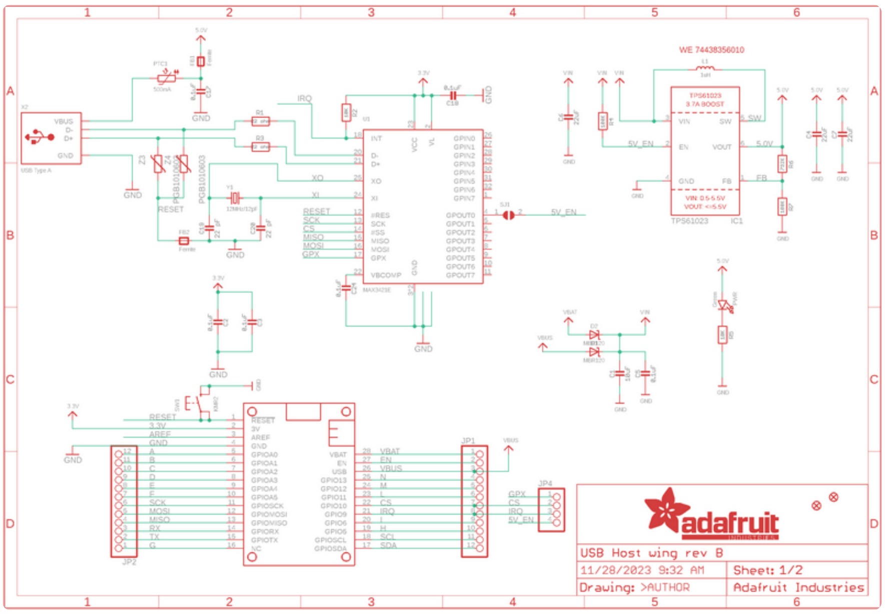
Downloads

Files

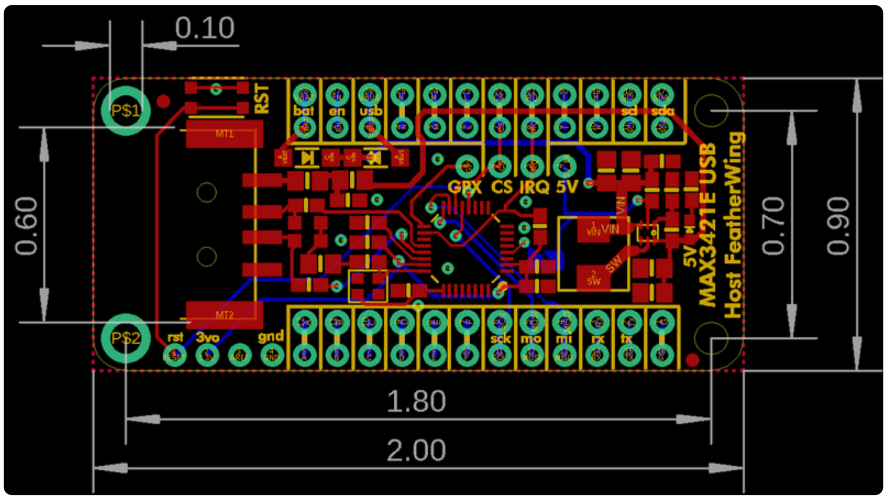
- MAX3421E Datasheet (https://adafru.it/19bh)
- EagleCAD PCB Files on GitHub (https://adafru.it/19bj)

- [Fritzing object in the Adafruit Fritzing Library](https://adafru.it/19bm) (https://adafru.it/19bm)

Schematic and Fab Print



USB Host wing rev B
11/28/2023 9:32 AM
Sheet: 1/2
Drawing: >AUTHOR
Adafruit Industries



0.10
0.60
0.70
0.90
1.80
2.00
MAX3421E USB Host FeatherWing