# Table of Contents

## Overview

## Pinouts
- Connecting ribbon cable to floppy drive
- Inputs (to microcontroller)
- Outputs (to floppy drive)
- Optional 5V Connection

## Archiving Floppies with GreaseWeazle Compatible Firmware
- Upload the firmware to the Feather RP2040
- Determine the serial port name
- Install & Use FluxEngine
- Install & Use GreaseWeazle

## Examining IMG Files
- Examining IMG files

## Fluxengine Documentation

## GreaseWeazle documentation

## USB Floppy Drive
- Upload the firmware to the Feather RP2040
- Use the USB Floppy Drive

## Using Arduino adafruit_floppy
- MFM (DOS/Windows) Floppy Decoding
- Pins & Construction
- Grabbing raw flux data
- Decoding MFM flux data

## Arduino API documentation

## Using CircuitPython: adafruit_floppy

## CircuitPython API documentation

## Downloads
- Schematic and Fab Print
Overview

Relive the days when storage was counted out in kilobytes, not gigabytes, using the Adafruit Floppy FeatherWing on a Feather board. This is perfect for interfacing with 34-Pin IDC Connector floppy drives that were ubiquitous on PC's. This 'Wing has level shifting and a ready-to-plug connector that works with 3.5" or 5.25" floppy disks for reading or writing!

Floppy disks have an interesting data transfer method where raw bit transitions are measured and converted to data. This data streams pretty fast from the disk drive, so you'll want to use a fast microcontroller that has large SRAM storage and ideally, a peripheral to DMA the data signal. For that reason, we only have this FeatherWing working with the Feather M4 (🔗) or Feather RP2040 (🔗). ESP32, ATmega, nRF52, etc have not been ported to our support library (🔗)!

Floppy disk drives require 5V power and logic. For the logic level shifting, we have a small boost converter on the 'Wing that will give 5V logic levels out the 34-pin connector. It will also shift the incoming signals down to a Feather-safe 3.3V.

For floppy drive powering you will likely need a separate external 5V power supply that can provide 5V at 2 A. You can't power most drives off of USB power, they have big motors that require more current than USB can provide!

©Adafruit Industries
There is code and there are examples for three use cases:

- **Reading floppy disks and storing the raw bit patterns and/or converted data format for data archiving**. For this use case we recommend using the 'Wing with a Feather programmed to act as a greaseweazle or fluxengine client device. Then run those command line programs on your computer to have the Feather send the raw track data to the computer for analysis, bit-repair and format interpretation. You can also write some formats via those two programs, for use with older computers. There is a 'Write Disable' switch on the board specifically for this use case so that it is not possible for the write-enable pin to accidentally be activated.

- **Reading/Writing FAT formatted floppy disks from Arduino, CircuitPython or a computer that is connected via USB**. Floppy disks can still be used to save, read data from! This is great for retro-themed projects or if you want to write data to floppy disks for them to be used by IBM PC compatibles or other devices that are expecting FAT-formatted floppy disks.

- **Cool floppy music sequencing!"**
Each order comes with one FeatherWing with the 2x17 header attached and level shifting circuitry built in. Before it can be used to read/write floppies, you will need to:

- Solder pin headers on to attach to a Feather board.
- Attach an M4 or RP2040 Feather (link) as the driver/brains.
- Find a 34-pin IDC Floppy Drive that works
- Connect a 34-pin DC cable (link) between the 'Wing and the Floppy drive.
- Provide 5V power to the floppy drive, as much as 2 Amps. 5.25" disk drives will also need a 12V power supply (link)!
- Find some floppy disks to read/write

Many of these items are no longer sold, and will require some sleuthing to locate!

Adafruit Floppy FeatherWing with 34-Pin IDC Connector
Relive the days when storage was counted out in kilo-bytes not giga-bytes, using the Adafruit Floppy FeatherWing on a Feather board - perfect for interfacing...
https://www.adafruit.com/product/5679
Rainbow 34-pin Dual Row IDC Floppy Ribbon Cable - 30cm long
Often used for floppy drives, 34 pin IDC cables have a 2x17 array of pins. This cable will let you connect one port...
https://www.adafruit.com/product/5453

4-pin AT / ATX / IDE Molex to Berg Floppy Drive Power Cable
For big projects, with big power needs, an AT or ATX power supply is a handy place to grab 5V or 12VDC. Only problem is that these supplies have connectors that only plug into...
https://www.adafruit.com/product/425

Adafruit Feather RP2040
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather...
https://www.adafruit.com/product/4884

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
1 x Power Supply
+5/+12V power supply with Molex and/or Berg Connector

Compatible power supplies can be bought inexpensively as part of a "USB to IDE" kit. See this one as an example, they can be found at many retailers for reasonable prices.

Optional for 5.25" floppies:

1 x Compatible 5.25" Drive
5.25" drive with 34-pin card-edge connector

1 x 34-pin IDC to card-edge connector
For connecting a 5.25" floppy drive

Pinouts

The Adafruit Floppy Featherwing is intended for use with the Feather RP2040 or Feather M4. Note that most of the views below show the bottom side of the FeatherWing. A bunch of pins are required, but note that the standard I2C and SPI pins are left free so you’ll be able to incorporate displays or sensors into your project if needed!

To keep easy access to the RP2040’s BOOT button consider putting this FeatherWing side by side with the Feather RP2040 on a Feather Doubler, Tripler, or Quadrupler.
Connecting ribbon cable to floppy drive

The Adafruit 34-pin rainbow ribbon connector is polarized, so insert it noting the location of the "key" and aligning it with the notch on the Floppy FeatherWing. The brown wire is pin 1, and the other pins can be "counted off" according to the standard resistor code: red is 2, orange is 3, and so on; black is 10, and then the next brown is 11. This makes last wire yellow, because there are 34 wires in all.

Then, do the same at the drive. However, note that some drives lack a keying notch; and others have a notch but it does not match modern convention! (the author’s own Sony MFP520-1 with 9411 date code is such a drive) In the case that the drive's polarization notch is missing or incorrectly positioned, you may have to carefully use a tool to shave off the polarizing tab from the cable, or find an alternate non-polarized cable.

Therefore, it's important to check whether the cable is installed correctly before applying any power. Use a multimeter in continuity mode to check that GND on the floppy drive is connected to GND on the FeatherWing.
Only once everything else is connected should you provide power to the floppy drive on its dedicated +5/+12V power connector. Remember that floppies were not designed for hot-plugging of their power or data connections.

**Inputs (to microcontroller)**

All inputs include pull-up resistors & a level shifter to 3.3V.

CH: Media change. When HIGH, the drive indicates a media change operation is in progress. When LOW a media change operation is not in progress. This is connected to D5 on the common Feather pinout, and pin 34 of the IDC connector. This pin is also known as "media ready". Based on some floppy drives your author tested, this signal is not particularly useful but is included for the sake of completeness.
RD: Read Data. Alternates between HIGH and LOW according to the flux stored on the floppy disk. The specific interpretation of the pulses depends on the floppy disk format. This is connected to D9 on the common Feather pinout.

Trk: Track 0 (home) sense. Pulled LOW when the disk head is moved to the "track zero", or home position. HIGH when not at the home position. This is connected to D10 on the common Feather pinout, and pin 26 on the IDC connector.

Pr: Write Protect Sense. Pulled LOW when the disk is write protected (e.g., by the sliding tab on a 3.5" floppy disk). HIGH when the disk is not write protected. This is connected to D11 on the common Feather pinout, and pin 28 on the IDC connector.
Idx: Gives a LOW pulse once per revolution of the floppy (or, for the unusual "hard sectored" floppy, once for each sector mark). Connected to A5 (aka D25) on the common Feather pinout, and pin 8 on the IDC connector.

Outputs (to floppy drive)

All outputs include a pull-up resistor and 5V level shifter. They may be driven as open collector pins or as push-pull pins.

Sel: Select signal. When LOW, "selects" (activates) the floppy disk. Connected to A0 on the common Feather pinout and pin 12 ("Select1") on the IDC connector.

En: Enable signal. When LOW enables the selected floppy drive spindle motor. Connected to pin A2 on the common Feather pinout and pin 16 on the IDC connector.

Pulling these two pins LOW together selects a drive connected with a straight-through cable, or to the un-twisted portion of a multi-drive connector.

There is no connection for the Select0/2/3 pins of the IDC connector. Consequently, it's only possible to select a drive on the un-twisted part of the cable. So if you have
an old floppy drive cable, make sure you plug the floppy drive into the correct position.

Dir: Direction. Selects whether a step pulse moves inward to a smaller track number (LOW) or outward to a higher track number (HIGH). Connected to pin A3 on the standard Feather pinout and pin 20 on the IDC connector.

Stp: Step. Each LOW going pulse moves one step. Depending on the drive & format, more than one step may be required in order to move by one full track.

Sid: Side select. HIGH to select "Side 0" of the floppy, LOW to select "Side 1". Connected to D6 on the common Feather pinout and pin 32 of the IDC connector.
Den: Density selection. Some drives use this pin to select whether to write at "high density" (pin pulled LOW) or "double density" (pin pulled HIGH). Connected to A1 on the common Feather pinout and pin 2 of the IDC connector.

Write Enable: Depending on its position, this switch either isolates (NoWr) or connects (WrOK) the WG signal from the microcontroller to the floppy drive.

WG: Write Gate. When pulled LOW, puts the drive in writing mode. If the Write Enable switch is in the position labeled NoWr then this is physically disconnected from the microcontroller and it is not possible to write to a floppy (or, with buggy code, to accidentally erase a floppy!) Connected to D12 on the common Feather pinout and pin 24 on the IDC connector.

WD: Write Data. When WG is pulled low (and the Write Enable switch is in the WrOK position and the inserted disk is not write protected), data on this pin is written to the disk. The specific encoding of the data on this pin depends on the floppy format being written. Connected to D13 on the common feather pinout and pin 22 on the IDC connector.
Optional 5V Connection

The screw terminal is connected to the Feather's USB5V line and to GND. Depending on the USB power supply to the Feather, this can be used to power some low-power laptop floppy disk drives, but should not be used to power desktop floppy drives.

Archiving Floppies with GreaseWeazle Compatible Firmware

Adafruit provides a GreaseWeazle compatible firmware for the Adafruit Floppy FeatherWing paired with the Adafruit Feather RP2040.

This firmware can be used together with GreaseWeazle or FluxEngine software on a host computer to archive the data from floppies onto a modern PC. They each have slightly different capabilities, so it's great that they both work with the same device firmware.

Upload the firmware to the Feather RP2040

Grab the latest release from GitHub. Pick the file named "feather_rp2040_tinyusb.zip" and unzip it.

Plug in the Feather and then place it in bootloader mode by holding the BOOT button and clicking RESET.
Drag the GreaseWeazle UF2 file to the RPI-RP2 volume. It will automatically restart with the new firmware.

Determine the serial port name

Depending on your operating system, the device will be assigned a serial port name. On Windows, a serial port name is a string like "COM6" or "COM33"; on Mac and Linux it's a string starting "/dev/". Check out this guide page for more info on finding the serial port name: "COM / Serial Port Name ()"

When you run FluxEngine or GreaseWeazle, you'll need to manually specify the serial port name.

Install & Use FluxEngine

Grab the appropriate installer (exe or pkg) from the releases page on GitHub ()

After installing it, insert a floppy disk and close the door. In a terminal and run the FluxEngine command to check that everything's working (change COM6 to the serial device name that's right for your computer):

```
C:\fluxengine&gt; fluxengine rpm --usb.greaseweazle.port=COM6
Using GreaseWeazle on serial port COM6
Rotational period is 198.445 ms (302.351 rpm)
```

FluxEngine supports a wide range of disk formats. Check out the official FluxEngine documentation () for detailed information on how to use it. To whet your appetite, here's how to use FluxEngine to read the content of a 1.44MB 3.5" DOS/Windows disk to an image file:

```
C:\fluxengine&gt; fluxengine read --usb.greaseweazle.port=COM6 ibm
```

Install & Use GreaseWeazle

Windows users can grab a compiled version from the releases page on GitHub (). Others can install the source with the command pipx install git+https://github.com/keirf/greaseweazle or by cloning the git repo and running its make command.

After installing it, open a terminal and run the gw command to check that everything's working:
GreaseWeazle also supports a wide range of formats. There's an official greaseweazle documentation wiki for more information on how to use it. To get you started, here's how to read the content of a 1.44MB 3.5" DOS/Windows disk to an image file (change COM6 to the serial device name that's right for your computer):

C:\greaseweazle&gt; gw read --device COM6 --format ibm.1440 output.img

Examining IMG Files

Examining IMG files

Once you have imaged the IBM PC floppy, you can safely put it away and look at the data on the image you’ve made.

If you want a known-good FAT12 formatted IBM PC 1.44MB disk image to practice with, this zip download contains one image from fluxengine. Note you have to uncompress the zip and remove the image file within!

IBM_PC_FAT12_1_44MB.zip

Windows

Even though IMG files are 'byte for byte' disk images, Windows doesn't have native ability to mount it as a floppy disk like CD-ROM ISOs. Not sure why, but that's how it is.

You can use https://www.7-zip.org/ instead, which happens to have support for FAT12 format images.

We recommend you download the .msi version as we need that installer to have the context window show up.
In the FluxEngine folder, right click on the img file created, select 7-Zip and Open archive (or you can Extract files...)

You can then interact with the files as you like:

Fluxengine Documentation

GreaseWeazle documentation

USB Floppy Drive

Adafruit provides a USB Mass Storage Device firmware for the Adafruit Floppy FeatherWing paired with the Adafruit Feather RP2040.
With this firmware installed, you can directly read a DOS/Window (FAT)-formatted 1.44MB floppy disk on standard computers.

Under the covers, the RP2040 is reading the raw flux and performing MFM decoding before it sends the decoded data to your PC.

Upload the firmware to the Feather RP2040

Grab the [latest release from GitHub](https://github.com/Adafruit/Adafruit Feather RP2040) (pick the file named "feather_rp2040_tinyusb.zip" and unzip it).

Plug in the Feather and then place it in bootloader mode by holding the BOOT button and clicking RESET.

Drag the msd_test.uf2 file to the RPI-RP2 volume. It will automatically restart with the new firmware.

Use the USB Floppy Drive

Insert a floppy drive and then click the device's reset button. It will connect to your computer as a USB floppy disk. Note that the floppy disk is read-only, and only 1.44MB PC-format disks can be used.

Using Arduino adafruit_floppy

Adafruit has an Arduino library & example code ready to go with this FeatherWing. The library is optimized for the Adafruit Feather RP2040 microcontroller board.

This library needs to be installed using the Arduino Library Manager. From the Arduino “Sketch” menu, select “Include Library” then “Manage Libraries...”.

Type “floppy” in the search field to quickly find the library — Adafruit_Floppy:
If prompted to install additional libraries, say "yes"; Adafruit_Floppy relies on Adafruit BusIO and SdFat - Adafruit Fork.

You have to make specific selections in the Tools menu:

- Board: "Adafruit Feather RP2040"
- CPU Speed: "200 MHz (Overclock)"
- Optimize: "Optimize Even More (-O3)"
- USB Stack: "Adafruit TinyUSB"
After restarting the Arduino software, you should see a new example folder called Adafruit_Floppy, and inside, an example called floppy_capture_track_test. Upload the sketch to your Arduino and open the serial monitor. Place a disk in your floppy drive, and reset the board. Your floppy should "spin up" and you should see information about the flux data on your floppy's "track zero". For a formatted floppy, the "flux bins" (a histogram of times between adjacent flux transitions) should cluster preferentially into 3 to 4 different groups of values with large gaps between them. For example, here’s the frequency information from an MFM-formatted (DOS) floppy. The peaks correspond to 4, 6, and 8µs, or 96, 144, and 192 counts at 24MHz:

```
[12:22:39] its time for a nice floppy transfer!
[12:22:40] Sample freqency 24.00MHz
[12:22:40] Waiting for index pulse...Found!
[12:22:40] Seeking track...Going to track 0
[12:22:40] done!
[12:22:40] Captured 45282 flux transitions
[12:22:40] -------
[12:22:40] 93: 130
[12:22:40] 94: 1460
[12:22:40] 95: 6257
[12:22:40] 96: 13469
[12:22:40] 100: 302
```
Because the floppy is inherently an analog device, the exact counts in each flux bin will vary slightly from run to run.

**MFM (DOS/Windows) Floppy Decoding**

There are a lot of floppy formats, but one ubiquitous one at the end of the floppy era was the IBM PC compatible MFM floppy. The example "mfm_test" will decode the content of such a floppy disk and display it on the serial monitor in hex and ASCII.

```
Seeking track 8 head 1
Going to track 8
Captured 9 sectors
Validity: VVVVVVVVV
00000000 65 76 65 72 61 6c 20 6c 6f 77 65 72 20 63 6f 75 | everal lower cou
00000000 6f 70 69 6e 69 6f 6e 73 2c 20 69 6e 63 | rt opinions, inc
00000000 6c 75 64 69 6e 67 20 0d 0a 74 68 65 20 6f 70 69 | luding the opi
00000000 6e 69 6f 6e 73 20 69 6e 20 41 6d 65 72 61 64 61 | nions in Amerada
00000000 20 48 65 73 73 20 61 6e 64 20 4c 54 56 20 53 65 | Hess and LTV Se
00000000 63 75 72 69 74 69 65 73 20 74 68 61 74 20 61 72 | curities that ar
00000000 65 20 73 6f 20 0d 0a 66 72 65 71 75 65 6e 74 6c | e so frequentl
00000000 79 20 63 69 74 65 64 20 69 6e 20 73 75 70 70 6f | y cited in suppo
00000000 72 74 20 6f 66 20 74 68 65 20 62 72 6f 61 64 65 | rt of the broade
00000000 72 20 73 63 6f 70 65 20 66 6f 72 20 74 68 65 20 | r scope for the
00000000 0d 0a 41 6d 65 72 61 64 61 20 48 65 73 73 20 6d 61 | ection that the
00000000 79 20 62 65 20 6e 6f 20 6d 61 79 20 62 65 20 6e 6f | may be no m
```

For full details of the Adafruit Floppy library, see the [API documentation](https://www.example.com). Here are some key details, though:

©Adafruit Industries
Pins & Construction

There are a lot of pins, so we put a standard definition block at the top of the examples for you to cut & paste into your code:

```c
// these are for Adafruit Feather RP2040
#define DENSITY_PIN A1 // IDC 2
#define INDEX_PIN 25   // IDC 8
#define SELECT_PIN A0  // IDC 12
#define MOTOR_PIN A2   // IDC 16
#define DIR_PIN A3     // IDC 18
#define STEP_PIN 24    // IDC 20
#define WRDATA_PIN 13  // IDC 22
#define WRGATE_PIN 12  // IDC 24
#define TRK0_PIN 10    // IDC 26
#define PROT_PIN 11    // IDC 28
#define READ_PIN 9     // IDC 30
#define SIDE_PIN 8     // IDC 32
#define READY_PIN 7    // IDC 34

Adafruit_Floppy floppy(DENSITY_PIN, INDEX_PIN, SELECT_PIN,
                         MOTOR_PIN, DIR_PIN, STEP_PIN,
                         WRDATA_PIN, WRGATE_PIN, TRK0_PIN,
                         PROT_PIN, READ_PIN, SIDE_PIN, READY_PIN);
```

Usually, you'll start with the sequence: `begin`, `select`, `spin_motor`, and `goto_track`:

```c
if (!floppy.begin()) {
    Serial.println("Failed to initialize floppy interface");
    while (1) yield();
}

floppy.select(true);
if (! floppy.spin_motor(true)) {
    Serial.println("Failed to spin up motor & find index pulse");
    while (1) yield();
}

Serial.print("Seeking track...");
if (! floppy.goto_track(0)) {
    Serial.println("Failed to seek to track");
    while (1) yield();
}
```

Grabbing raw flux data

You can capture flux data to an array:

```c
// outside any function
// WARNING! there are 150K max flux pulses per track!
uint8_t flux_transitions[MAX_FLUX_PULSE_PER_TRACK];

// inside loop() or other function
int32_t index_pulse_offset;
```
Decoding MFM flux data

Or use an `Adafruit_MFM_Floppy` object to get block-level access to a MFM-formatted floppy disk:

```c
uint32_t captured_flux = floppy.capture_track(flux_transitions, sizeof(flux_transitions), &index_pulse_offset, true);
```

Arduino API documentation

Using CircuitPython: adafruit_floppy

Adafruit has a CircuitPython library called adafruit_floppy for working with raw flux and with MFM-encoded data.

In this example using the library, information about the raw flux on track 0 of the inserted floppy is printed.

To use with CircuitPython, you need to first install a few libraries, 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, open the directory examples/ and then click on the directory that matches the version of CircuitPython you're using and copy the contents of that directory to your CIRCUITPY drive.

Your CIRCUITPY drive should now look similar to the following image:
import board
import adafruit_floppy

D24 = getattr(board, "D24") or getattr(board, "A4")
D25 = getattr(board, "D25") or getattr(board, "A5")

floppy = adafruit_floppy.MFMFloppy(
    densitypin=board.A1,
    indexpin=D25,
    selectpin=board.A0,
    motorpin=board.A2,
    directionpin=board.A3,
    steppin=D24,
    track0pin=board.D10,
    protectpin=board.D11,
    rddatapin=board.D9,
    sidepin=board.D6,
    readypin=board.D5,
)

floppy.selected = True
floppy.spin = True
print("Seek track 8")
floppy.track = 8
print("Seek track 0")
floppy.track = 0
print("Read partial track raw flux data")
buf = bytearray(30000)
n_read = floppy.flux_readinto(buf)
print("read", n_read)
buckets = [0] * 256
for b in buf:
    buckets[b] += 1
oi = -1
for i, bi in enumerate(buckets):
    if bi > 10:
        if i != oi + 1:
            print("---")
        oi = i
    print(f"{i:3} {bi:5}"

First, necessary modules are imported. Then, some conditional lines are used to adapt between boards that call the pins "A4/A5" and boards that call the pins "D24/
D25". After that, an MFMFloppy object is created that refers to all of the appropriate pins:

```python
import board
import adafruit_floppy

D24 = getattr(board, "D24") or getattr(board, "A4")
D25 = getattr(board, "D25") or getattr(board, "A5")

floppy = adafruit_floppy.MFMFloppy(
    densitypin=board.A1,
    indexpin=D25,
    selectpin=board.A0,
    motorpin=board.A2,
    directionpin=board.A3,
    steppin=D24,
    track0pin=board.D10,
    protectpin=board.D11,
    rddatapin=board.D9,
    sidepin=board.D6,
    readypin=board.D5,
)

The next steps are to select and spin the floppy disk:

```python
floppy.selected = True
floppy.spin = True
print("Seek track 8")
floppy.track = 8
print("Seek track 0")
floppy.track = 0
```

Finally, raw flux data can be captured and printed:

```python
print("Read partial track raw flux data")
buf = bytearray(30000)
n_read = floppy.flux_readinto(buf)
print("read", n_read)
buckets = [0] * 256
for b in buf:
    buckets[b] += 1
oi = -1
for i, bi in enumerate(buckets):
    if bi > 10:
        if i != oi + 1:
            print("---")
        oi = i
        print(f"{i:3} {bi:5}"
```

Typical output:

```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Seek track 8
Seek track 0
Read partial track raw flux data
read 30000
---
 94   121
 95   1122
```
CircuitPython API documentation

Downloads

- IDC Connector Diagram ()
- EagleCAD PCB Files on GitHub ()
- Fritzing Object in the Adafruit Fritzing Library ()
Schematic and Fab Print