NASA TechRise: Basic Sensor How-To

As you brainstorm your ideas, you can use this “Basic Sensor How-To" to help you think through the steps you might take to begin building a payload.

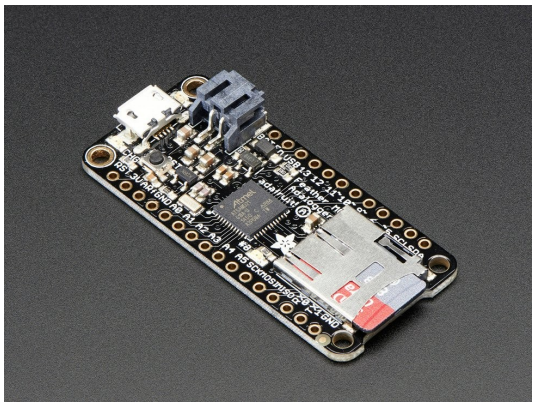
Let's say we want to record temperature and humidity during a weather balloon flight and usethat data to create graphs. What is the basic hardware and software necessary to do that?

Hardware

To collect data, we need a sensor that responds to the conditions we want record (temperature & humidity in this case), a microcontroller to read the sensor, and a storage device to store the data.

Microcontroller with SD card reader/writer

Some microcontrollers have built-in SD card readers.

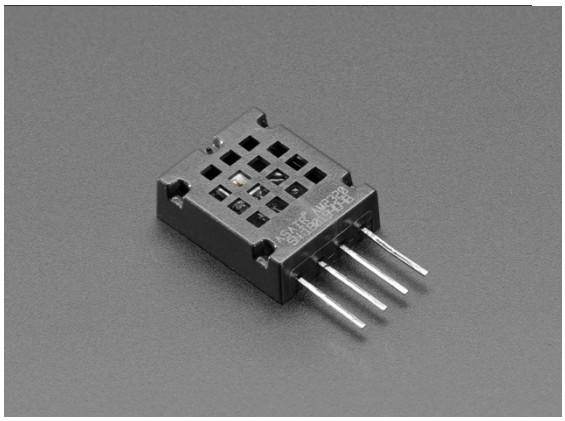


This simple datalogger microcontroller is a useable choice.

Adafruit Feather M0 Adalogger
https://www.adafruit.com/product/2796

Temperature and Humidity Sensor

Any simple temperature and humidity sensor should work...



This one is low power and uses 12C, so only 4 wires are needed to connect to an MCU. AM2320 Digital Temperature and Humidity Sensor
https://www.adafruit.com/product/3721

Additional Hardware

- Half sized

  breadboard

- Jumper wires
- 2X 10k ohm resistors (pull-up for I2C)

Hardware Setup

*Note: The microcontroller board (Adafruit Feather M0 Adalogger) used in this example doesrequire soldering. For a complete guide on how to setup a board like this go here: https://learn.adafruit.com/adafruit-feather-m0-adalogger/assembly

Assumptions:

1. The Feather microcontroller has pins soldered to its underside
2. We will use the USB port on the microcontroller as the primary power supply.This will provide 5V to the microcontroller.
3. We will use the 3V pin on the microcontroller to power the sensor
4. The sensor requires two pull-up resistors connected to its SDA and SCL pins (pins 2 and 4)

Microcontroller pin-out

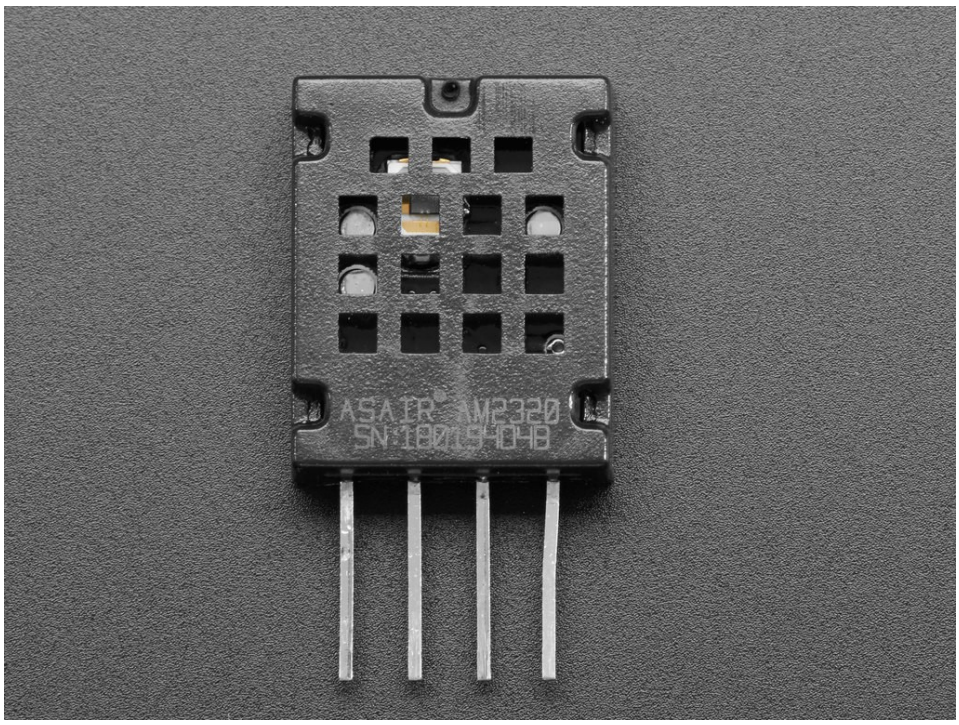
Microcontrollers use metal pins to connect to power, ground, and other devices. Each board will have a set of pin "names" that are either unique or based on a standard (like Arduino UNO). For this simple experiment we will need to connect to a sensor that uses 4 pins labeled 3V, GND,

SCL and SDA. For a complete list and descriptions of the pins on the microcontroller in this lesson go here:

https://learn.adafruit.com/adafruit-feather-m0-adalogger/pinouts

Sensor pin-out

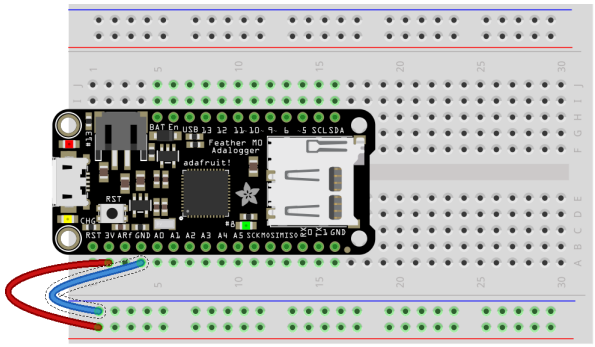
Every electrical sensor will have pins that connect back to a microcontroller. The "names" of these pins are known as the pin-out. Some sensors might be simple on/off devices like motion sensors (digital) others might create a variable voltage like light sensitive diodes or resistors (analog) and still others will use a communications protocol (like I2C or SPI). The sensor in this example is I2C which is a universal 2-wire serial interface. The pins on our devices are labeled 1-4 from left to right and the "names" of those pins, in order, are VCC (power), SDA (Serial-data), GND (ground), and SCL (Serial-clock). For an in depth look at how I2C works, check out this tutorial from Sparkfun.com: https://learn.sparkfun.com/tutorials/i2c/all. For a complete guideon the sensor in this experiment go here: https://learn.adafruit.com/adafruit-am2320-temperature-humidity-i2c-sensor



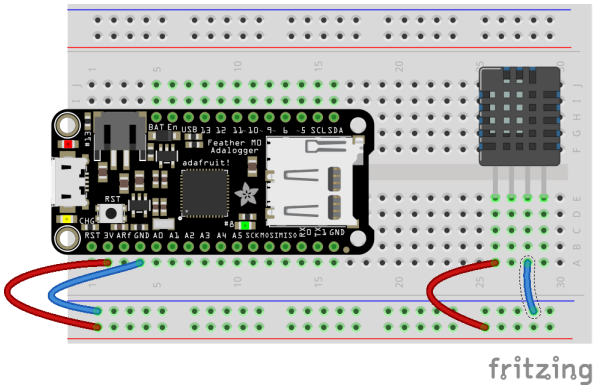
From left to right: VDD (voltage), SDA (serial data), GND (ground), SCL (serial clock)

Breadboard wiring

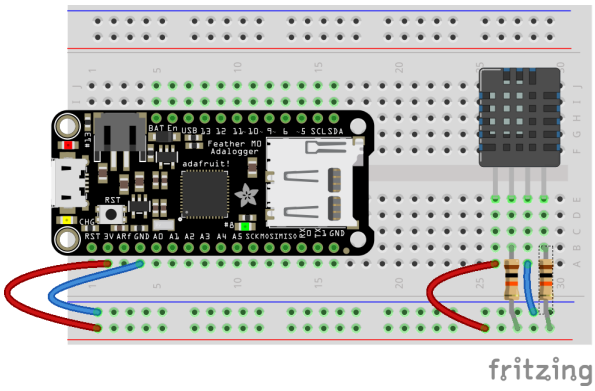
Step 1: Press the microcontroller into a breadboard with the USB port on the edge of the board and use a red wire to connect the pin labeled3V to the red rail on the bottom and a blue wire to connect the pin labeled GND to the blue rail on the bottom.



Step 2: Connect the sensor on the board as shown with the front grid facing the bottom of the breadboard and connect a red wire to the 1stpin on the left of the sensor and the red power rail at the bottom. Then connect a bluewire to the 3rd pin from the left to the blue ground rail on the bottom of the breadboard.



Step 3: Connect a 10k ohm resistor to the 2nd pin from the left on the sensor to the red power rail on the bottom and a second 10k ohm resistor to the 4th pin from the left also to the bottom red power rail. These are called pull-up resistors.



Step 4: Connect a blue wire from the SDA pin on thetop right of the microcontroller to the 2nd pinfrom the left on the sensor. Then connect a yellow wire from the SCL pin, 2nd from the top right on the microcontroller to the 4th pinfrom the left on the sensor. This is the data and clock connections for the sensor.

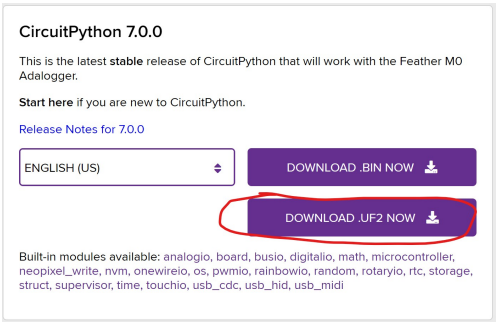
Step 5: Connect the USB cord to the microcontroller at the micro-USB port, and then connect to your computer for power supply.

Step 6: Make sure there is a micro-SD card (8gb or less) inserted in the SD card slot on the microcontroller and that is it! That is everything we need to program this microcontroller to read data from our I2C temperature and humidity sensor and save it to the SD card.

Software

To read our temperature and humidity sensor and save data we will need to write a bit of codeon the microcontroller. This example will use CircuitPython to do this. Software setup on our microcontroller is easy.

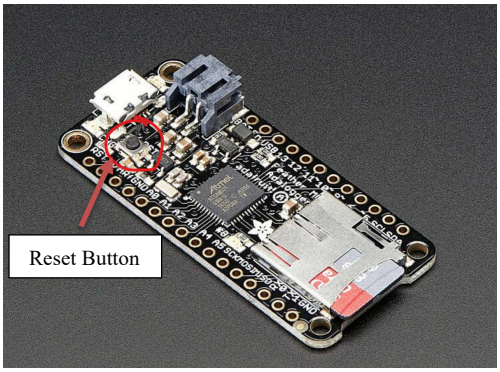
CircuitPython setup



You will need to download a .uf2 file from the CircuitPython website here: https://circuitpython.org/board/feather_m0_adalogger/

Use the latest release .uf2 file on the top right.

Once it is downloaded, plug the microcontroller into a micro USB cable and plug the other end into your computer.



Double click the small reset button on the board. A new drive called FEATHERBOOT will appear on your computer.

Simply drag the .uf2 file you downloaded into that drive. The microcontroller will restart and a new drive called CIRCUITPY will appear.

Libraries

Next you will need to download the CircuitPython library bundle here: https://github.com/adafruit/Adafruit_CircuitPython_Bundle/releases/download/20210921/adafruit-circuitpython-bundle-7.x-mpy-20210921.zip and extract the zip file onto your computer.

Copy the following .mpy files from the folder /lib/ in the zip file to the /lib folder on theCIRCUITPY drive.

- adafruit_am2320.mpy
- adafruit_sdcard.mpy
- adafruit_bus_device.mpy

Code

Now you can open Mu editor or your python editor of choice and open the code.py file from the CIRCUITPY drive.

Imports

We will need to import several modules at the top of our program.

The modules we need are listed here:

```
import time
import board
import busio
import adafruit_am2320
import adafruit_sdcard
import storage
from digitalio import DigitalInOut
```

Time will give us access to time.sleep() so we can pause the program, and time.monotonic() so we can keep track of the passage of time. Please note that the time is time elapsed in seconds. Board allows us to access the names of the pins on the board. Busio will give us access to the I2C and SPI devices we need for the sensor and sd card reader. adafruit_am2320 is the library for our sensor and adafruit_sdcard is the library for the sd card reader. Storage will allow us to access the sd card's file system, open a file, and save data to that file. Digitalio will allow us to set a CS pin for the SPI bus and a red led to blinkwhen we are writing data to file on the sd card.

Setup

Next we will initialize the I2C bus that the sensor is connected to and use the I2C object to initialize the sensor itself:

```
# initialize the I2c bus
i2c = busio.I2C(board.SCL, board.SDA)
# initialize the AM2320 sensor
temp_hum = adafruit_am2320.AM2320(i2c)
```

Then we will initialize the SPI bus that the SD card reader is connected to and create the digital CS (chip select) pin for the card reader as well. (On the Feather M0 Adalogger the CS pin is board.D5):

```
# initialize SPI bus and set CS pin
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
# sd card reader CS pin on Feather M0 Adalogger is connected to board.D4
cs = DigitalInOut(board.D4)
```

Next, we can initialize the the sd card reader itself using the SPI object and CS pin we defined. Then we will mount the filesystem of the storage device:

```
# initialize sdcard and storage
sdcard = adafruit_sdcard.SDCard(spi, cs)
vfs = storage.VfsFat(sdcard)
# mount the storage device
storage.mount(vfs, "/sd")
```

Finally, we can set the onboard D13 pin as an output so we can flash the onboard led when writing to the SD card:

```
# set the onboard red LED as an output so we can flash it when writing data
led = DigitalInOut(board.D13)
led.switch_to_output()
```

Main Loop

With setup completed we can start the main loop and add the code we need to read the temperature and humidity, print it out to serial, and save it to the SD card.

At the top of the loop we can save the current time in a variable called this_time so we can later use it to put a time stamp on our data:

```
# start the main loop
while True:
    # read time from time.monotonic() and save it for this loop
    this_time = time.monotonic()
```

Next, we can easily read the temperature and humidity into variables named t and h respectively, and print all three values to serial:

```
    # read the sensor values and print them into the serial monitort
    = temp_hum.temperature
    h = temp_hum.relative_humidity
    print("Time is:", this_time)
```

```
    print("Temp is:", t)
    print("Humidity is:", h)
```

Then with a few lines of code we can turn on the red led to indicate that data is being written, and format the data into a string to write to the SD card:
(for more on how Python's 'string'.format() function works, check out this tutorial: https://www.w3schools.com/python/ref_string_format.asp)

```
    # turn on the red led to indicate that data is being written to the sd card
    led.value = True
    print("writing data to the SD Card...", '\n')
    # format a data string as a list of comma separated values
    data_string = '{:n},{:-},{:n}\n'.format(this_time, t, h)
```

Finally, we will open a file called data_log.txt (don't worry about creating it ahead of time, if the file does not exist, the open command will create it automatically). The open command opens the file from the SD card so we can write our string into the file, turn off the led, then sleep for a second before the code will loop and do this all over again as long as the controller is powered on:

```
    # save the data to the data_log.txt file
    with open("/sd/data_log.txt", "a") as dl:
        dl.write(data_string)
        dl.flush()
    # turn off the red led until next time
    led.value = False
    # sleep for 1s before taking another reading
    time.sleep(1)
    # loop repeats
```

And that is all that is necessary to save data to a txt file in CircuitPython.

The full code.py file is here:

```
import time
import board
import busio
import adafruit_am2320
import adafruit_sdcard
import storage
from digitalio import DigitalInOut

# initialize the I2c bus
i2c = busio.I2C(board.SCL, board.SDA)
# initialize the AM2320 sensor
temp_hum = adafruit_am2320.AM2320(i2c)
```

```
# initialize SPI bus and set CS pin
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
# sd card reader CS pin on Feather M0 Adalogger is connected to board.D4
cs = DigitalInOut(board.D4)

# initialize sdcard and storage
sdcard = adafruit_sdcard.SDCard(spi, cs)
vfs = storage.VfsFat(sdcard)
# mount the storage device
storage.mount(vfs, "/sd")

# set the onboard red LED as an output so we can flash it when writing data
led = DigitalInOut(board.D13)
led.switch_to_output()

# start the main loop
while True:
    # read time from time.monotonic() and save it for this loop
    this_time = time.monotonic()

    # read the sensor values and print them into the serial monitor
    t = temp_hum.temperature
    h = temp_hum.relative_humidity
    print("Time is:", this_time)
    print("Temp is:", t)
    print("Humidity is:", h)

    # turn on the red led to indicate that data is being written to the sd card
    led.value = True
    print("writing data to the SD Card...", '\n')
    # format a data string as a list of comma separated values
    data_string = '{:n},{:-},{:n}\n'.format(this_time, t, h)

    # save the data to the data_log.txt file
    with open("/sd/data_log.txt", "a") as dl:
        dl.write(data_string)
        dl.flush()
    # turn off the red led until next time
    led.value = False
    # sleep for 1s before taking another reading
    time.sleep(1)
    # loop repeats
```

Output

So, what does this program give us? Essentially it creates lines of text in a file called data_log.txt that start with a time stamp followed by a comma with a temperature value followed by a comma and then a relative humidity value. This is commonly known as a CSV or Comma Separated Value format and it is a simple and typical way that data is stored so it can be converted into a spread sheet.

In order for us to use the data we have stored we only need to add one thing to our data_log.txt file: a header. A header is just a set of labels of each value in each line of comma separated

data.

If we open data_log.txt it will something like this:

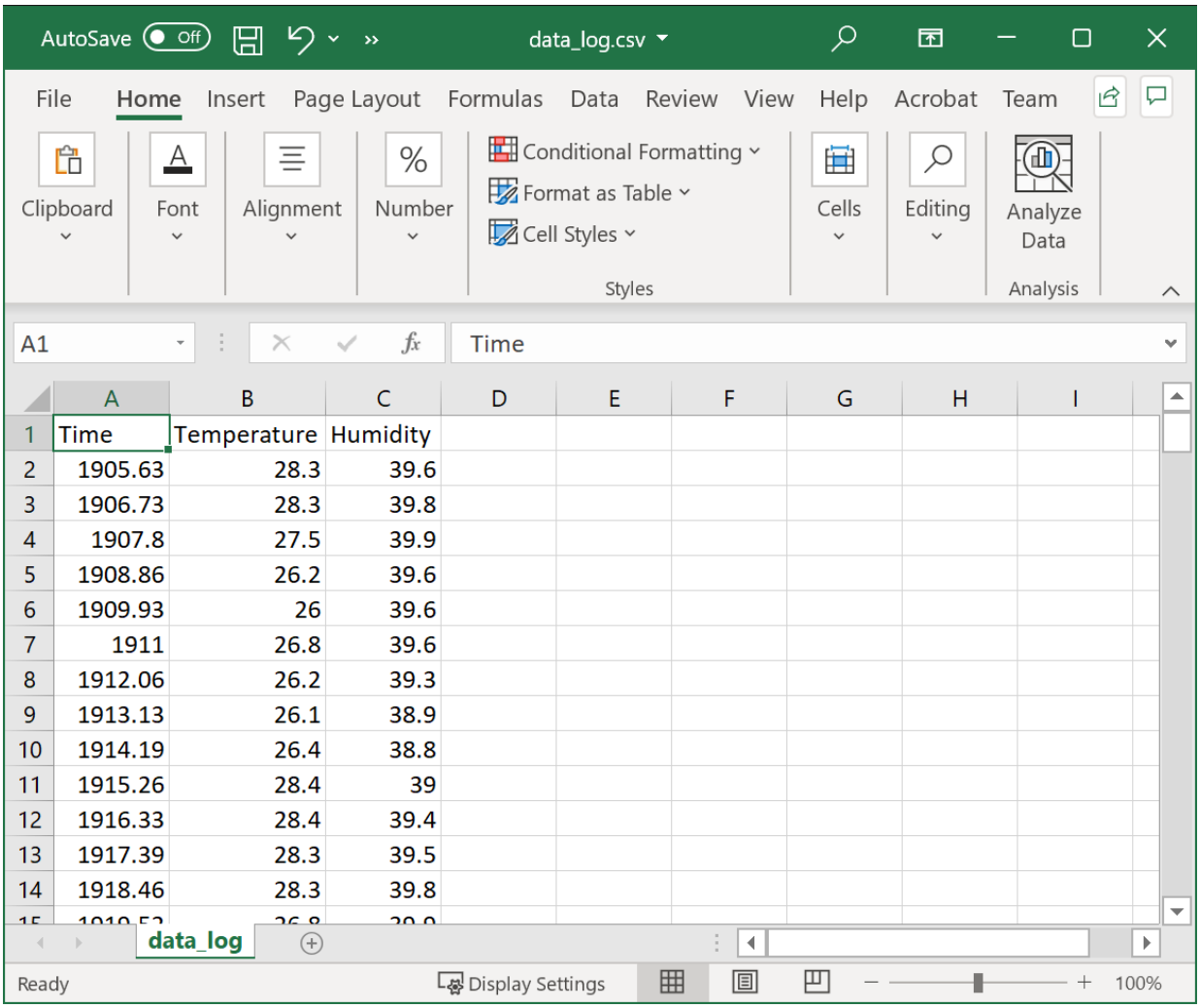
```
1905.63,28.3,39.6
1906.73,28.3,39.8
1907.8,27.5,39.9
1908.86,26.2,39.6
1909.93,26,39.6
1911,26.8,39.6
1912.06,26.2,39.3
1913.13,26.1,38.9
1914.19,26.4,38.8
1915.26,28.4,39
1916.33,28.4,39.4
1917.39,28.3,39.5
1918.46,28.3,39.8
1919.52,26.8,39.9
1920.59,28.3,40.1
1921.65,28.3,40.1
1922.72,28.5,40.1
1923.79,28.3,40.1
1924.85,26.8,39.9
1925.92,26.2,39.8
...
```

Each line is one of the data_string lines that we created in our program. They are all the same in that each one is Time,Temperature,Humidity. So, to create a csv file all we need to do is add a line to the top with those labels like this:

```
Time,Temperature,Humidity
1905.63,28.3,39.6
1906.73,28.3,39.8
1907.8,27.5,39.9
1908.86,26.2,39.6
1909.93,26,39.6
1911,26.8,39.6
1912.06,26.2,39.3
1913.13,26.1,38.9
1914.19,26.4,38.8
1915.26,28.4,39
1916.33,28.4,39.4
1917.39,28.3,39.5
1918.46,28.3,39.8
1919.52,26.8,39.9
1920.59,28.3,40.1
1921.65,28.3,40.1
1922.72,28.5,40.1
1923.79,28.3,40.1
1924.85,26.8,39.9
1925.92,26.2,39.8
...
```

Then we can save the file and rename it to data_log.csv

We can then import that csv file into google sheets or open it in Excel and we will see neatly formatted data that can be turned into graphs and charts!



	A	B	C
1	Time	Temperature	Humidity
2	1905.63	28.3	39.6
3	1906.73	28.3	39.8
4	1907.8	27.5	39.9
5	1908.86	26.2	39.6
6	1909.93	26	39.6
7	1911	26.8	39.6
8	1912.06	26.2	39.3
9	1913.13	26.1	38.9
10	1914.19	26.4	38.8
11	1915.26	28.4	39
12	1916.33	28.4	39.4
13	1917.39	28.3	39.5
14	1918.46	28.3	39.8