SETUP INSTRUCTIONS - Kamal Smith

Links to things used in this document

https://circuitpython.org/ (lightweight easy to use python library specifically for microprocessors)
https://learn.adafruit.com/welcome-to-circuitpython/the-repl (Circuit python How to)
https://learn.adafruit.com/circuitpython-essentials/circuitpython-essentials (Useful Circuit python info)
https://docs.circuitpython.org/projects/motorkit/en/latest/ (Motorkit library for turning motor)
https://codewith.mu/ (Easy to use IDE for testing, works well with circuitpython)
https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32s3/get-started/index.html (esp32 setup)
https://learn.adafruit.com/adafruit-esp32-s3-feather/factory-reset (To reset board to fac settings)
https://learn.adafruit.com/adafruit-esp32-s3-feather/pinouts (this is not the 4MiB flash board, may be different)
https://components101.com/motors/nema17-stepper-motor (Not actually the right stepper, but one of the only datasheets I could find)

If you plan on using other IDEs you may need the following
https://github.com/adafruit/Adafruit_CircuitPython_Bundle
https://github.com/adafruit/circuitpython-build-tools
You can find requirement.txt files inside the libraries if you need to install certain tools

To place the board into serial mode for flashing

1. While holding the boot button
2. Press Reset
3. Lights should turn off, now you can program things to it.

You can also just plug the device in while holding the boot to put it into serial mode

Flashing the board (First two steps are documented, but may not be necessary)

1. First, you need to get esptool onto your system by following this link https://docs.espressif.com/projects/esptool/en/latest/esp32s3/ . This will instruct you on installing the esptool to your system. (NOTE: “pip install esptool.py” might not work on certain systems, so try “pip install esptool”) there are more detailed instructions in the link if this does not work.

2. Now connect the ESP32S3 to your computer and run the “esptool -p PORT flash_id” with the port being the name of the serial port. (NOTE: I had to unplug the device and

while holding the boot button plug it in to put it in serial mode for this to work. Recheck the port on Windows because it might change!).

3. After this has been done, it's time to put circuitpython on the board. Run “esptool erase_flash” on your command prompt. If it does not automatically detect your boards port, try running “esptool --port PORTNAME erase_flash”. (NOTE: the – is a double dash and on windows the PORTNAME will be something like COM4).

```
C:\Windows\System32>esptool erase_flash
esptool.py v4.8.1
Found 1 serial ports
Serial port COM8
Connecting...
Detecting chip type... ESP32-S3
Chip is ESP32-S3 (QFN56) (revision v0.1)
Features: WiFi, BLE, Embedded Flash 4MB (XMC), Embedded PSRAM 2MB (AP_3v3)
Crystal is 40MHz
MAC: 70:04:1d:cd:2d:90
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 16.3s
Hard resetting via RTS pin...
```

Figure 1: Example of esptool erase_flash

4. Now to deploy the firmware to the device run “esptool –baud 460800 write_flash 0 ESP32_BOARD_NAME-DATE-VERSION.bin” [Feather ESP32-S3 4MB Flash 2MB PSRAM Download] with the .bin file found on this page. (NOTE: pay attention to the .bin file you download. We are using the firmware for 4MiB flash). You will then replace ESP32_BOARD_NAME-DATE-VERSION.bin with the path to the .bin you just downloaded.

```
C:\Windows\System32>esptool --baud 460800 write_flash 0 C:\Users\gamer\Downloads\ESP32_GENERIC_S3-FLASH_4M-20241129-v1.24.1.bin
esptool.py v4.8.1
Found 1 serial ports
Serial port COM8
Connecting...
Detecting chip type... ESP32-S3
Chip is ESP32-S3 (QFN56) (revision v0.1)
Features: WiFi, BLE, Embedded Flash 4MB (XMC), Embedded PSRAM 2MB (AP_3v3)
Crystal is 40MHz
MAC: 70:04:1d:cd:2d:90
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00000000 to 0x00195fff...
Compressed 1659056 bytes to 1086017...
Wrote 1659056 bytes (1086017 compressed) at 0x00000000 in 16.0 seconds (effective 831.0 kbit/s)...
Hash of data verified.
```

Figure 2: Example of flashing using the .bin provided

Side note : There is an online ESP32 factory reset tool that may make things easier, but when I tried using it I never got it to work so instead opted for the manual method. When you get to the final steps when you are loading the .bin file, you need the entire location of the bin file not just the name of the bin file.

MAKING THE MOTOR MOVE

https://learn.adafruit.com/keep-your-circuitpython-libraries-on-devices-up-to-date-with-circup/install-circup (For updating libraries on circuit python)
https://www.adafruit.com/product/2927

https://learn.adafruit.com/adafruit-stepper-dc-motor-featherwing/pinouts
https://learn.adafruit.com/welcome-to-circuitpython/library-file-types-and-frozen-libraries (converting py to .mpy)

https://learn.adafruit.com/adafruit-stepper-dc-motor-featherwing/circuitpython (documentation for featherwing addon)

Below is the wiring for the STEMMA QT header that we are using for the motor to connect by I2C.

- Red - 3.3VDC Power
- Black - Ground
- Blue - I2C SDA Data
- Yellow - I2C SCL Clock

1. Connect your red and black cable to the Logic power pins as shown below. I tested mine using the 3.3V power to the pin closest to the mounting hole, with GND in the other.

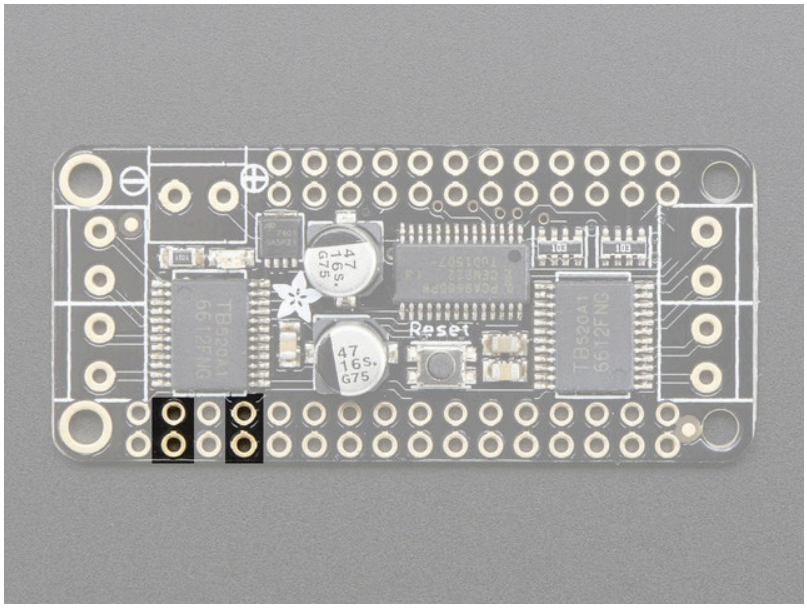

Image 3 : Adafruit showing the logic power pins on the featherwing

NOTE: Pay attention to the reset button, this is the best way to tell the orientation of the featherwing board.

2. Next connect your SCL and SDA lines with SDA (Blue) being the closest to the mounting hole, and SCL (Yellow) being the one next to it.

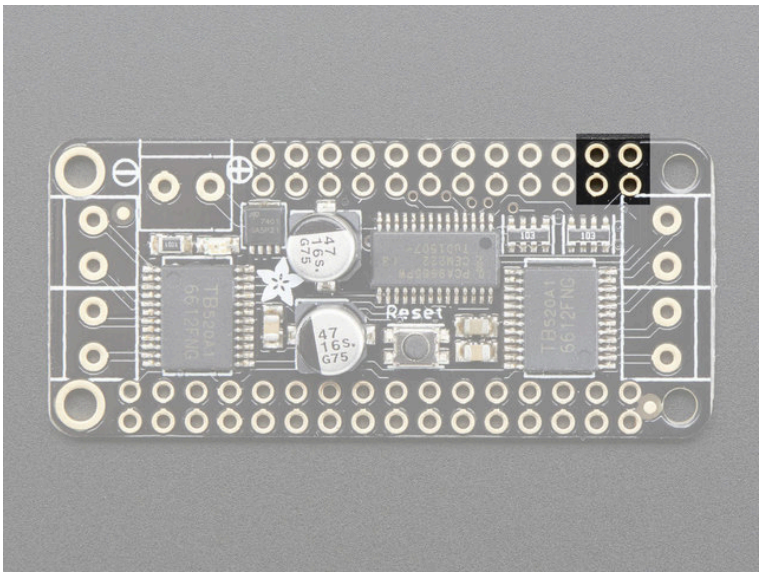

Image 4: Adafruit showing the I2C Data pins on the featherwing

3. Now to power it you will need to provide a Maximum of 12V. For testing, I used a simple 9V battery providing 5V of power. You will need both your power and ground.

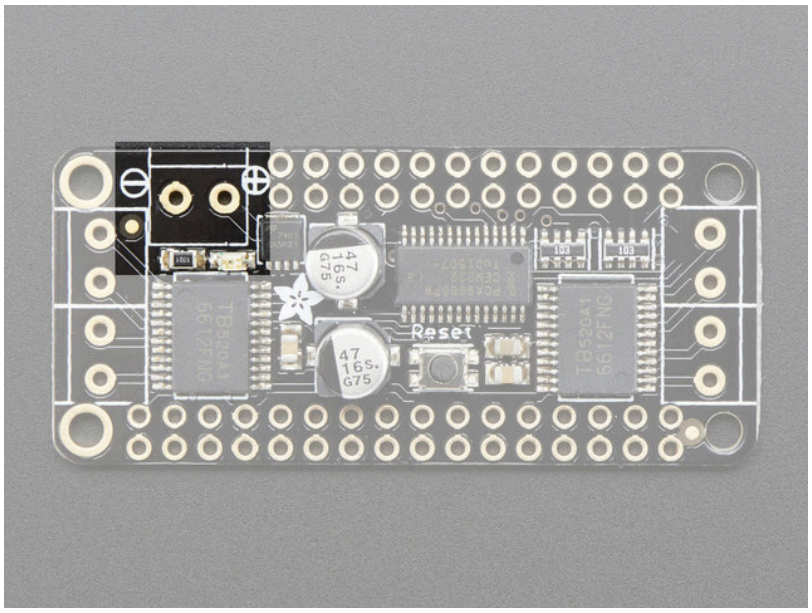

Image 5: Adafruit showing the power pins on the featherwing

4. The Stepper motor is still being tested, but my current working setup is posted below. The orientation of the input is a little twisted in the photo, so pay attention when plugging in.

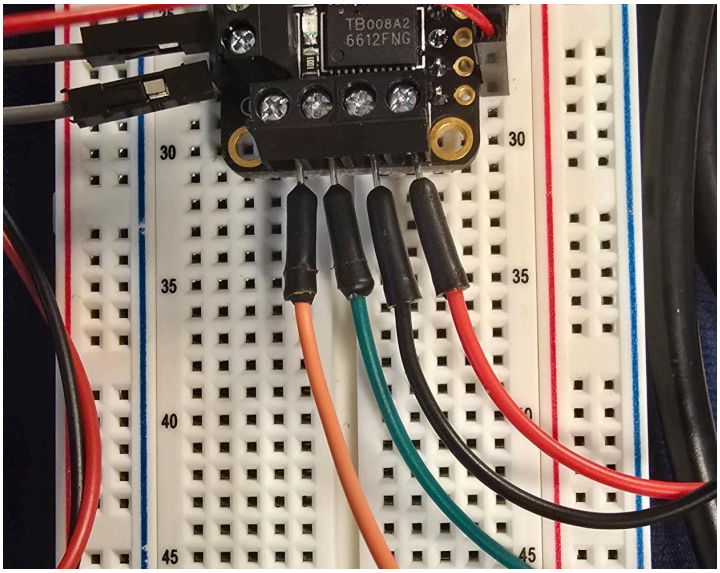

Image 6: Test setup wire input for motor

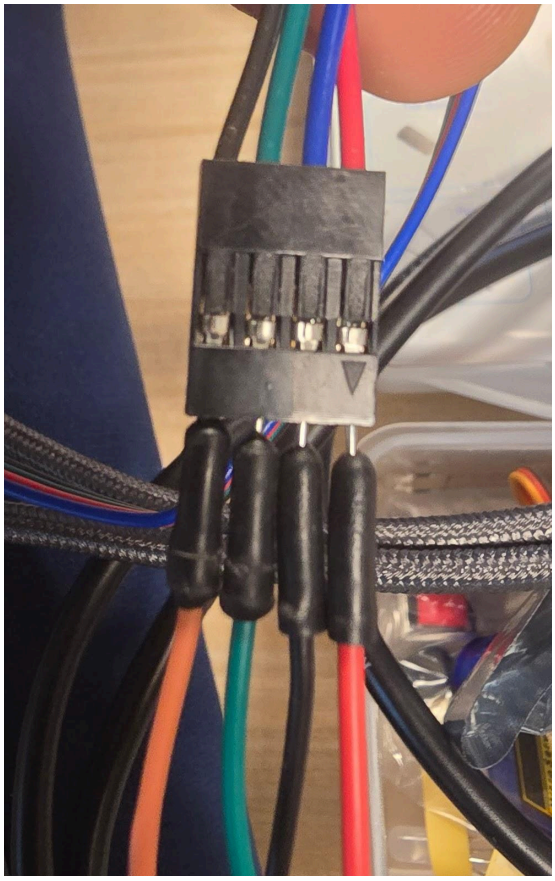

Image 6: Test setup wire input for motor with the orientation arrow

INTERFACING ESP32 TO XE125

https://learn.adafruit.com/introducing-adafruit-stemma-qt (This could help if we are running into issues)
https://www.sparkfun.com/qwiic (Can save us if we absolutely need it and can't get the I2C to connect)

https://docs.circuitpython.org/en/latest/shared-bindings/microcontroller/index.html#microcontroller.Pin (Documentation for Circuit python syntax)

https://learn.adafruit.com/circuitpython-essentials/circuitpython-i2c (I2C Notes)

https://circuitpython.org/libraries (Libraries for the circuit python)

https://learn.adafruit.com/adafruit-qt-py-esp32-s3/i2c

IDE SETUP

For this I am using the online code.circuitpython.org which is an online easy to use IDE for devices with circuit python on it.

1. To connect to the IDE with Circuitpython, you need to make sure your device is seen by Windows as a separate drive/ removable device. If flashed properly, this should work when you plug it in. This will allow you to open the device's files like any other drive.
2. When you connect the device, select the COM port that Windows is detecting. This was COM13 for me.
3. After you have selected the COM port, you will be asked what folder you would like to use. Select the root folder of the CIRCUITPY device that you have plugged in and continue.
4. The code that is currently on the board will show up as “code” and will be a python file. DO NOT CHANGE THE NAME OF THIS. While I am not sure if it actually messes things up, I was having issues whenever I renamed this file.

SIDE NOTE If your code is not saving, try pushing the reset button on the ESP32, this fixed my issues.

Wiring to XE125

Pin Number	Signal	Pin Number	Signal
1	DEBUG_UART_RX[1]	2	VIN_MISC
3	DEBUG_UART_TX[2]	4	MISC_GPIO2_BOOT0
5	WAKE_UP	6	MISC_GPIO1
7	MCU_INT	8	MISC_GPIO0
9	UART_RX[1]	10	GND
11	UART_TX[2]	12	I2C_ADDRESS
13	UART_CTS[3]	14	GND
15	UART_RTS[4]	16	NRESET
17	I2C_SDA	18	SWD_IO
19	I2C_SCL	20	SWD_CLK

For this, we will be focusing on GND, pin 5,7,12,17 and 19. I2C_address can be set high,low or not connected.

XE125_I2C_Address = 0x52

2.1 I²C Address Configuration

The device has a configurable I²C address. The address is selected depending on the state of the **I2C_ADDR** pin according to the following table:

Connected to GND	0x51
Not Connected	0x52
Connected to VIN	0x53

2.2 Usage

The module must be ready before the host starts I²C communication.

The module will enter ready state by following this procedure.

- Set **WAKE_UP** pin of the module HIGH.
- Wait for module to be ready, this is indicated by the **MCU_INT** pin being HIGH.
- Start I²C communication.

The module will enter a low power state by following this procedure.

- Wait for module to be ready, this is indicated by the **MCU_INT** pin being HIGH.
- Set the **WAKE_UP** pin of the module LOW.
- Wait for ready signal, the **MCU_INT** pin, to become LOW.

https://developer.acconeer.com/home/a121-docs-software/

Make Sure the ESP32-S3 and the XE125 share a common ground

Set WAKE_UP pin HIGH and check the MCU PIN before trying I2C processes