09-20-24

Blink an LED with an ESP32

Components

- ESP32-S3
- 220Ω resistor
- LED

Notes

- Wokwi Project
- I used this tutorial to blink a singular LED on Wokwi

Use ESP32 to interface with HX711 and Load Cell in Wokwi

Components

- ESP32-S3
- HX711
- Load Cell

Notes

Wokwi Project

- I used the Example Arduino Sketch from the Hobby Components forum post by Andrew Davies
    - I prefer this to using the HX711 Arduino library as I eventually want to learn how to implement this in ESP-IDF using C.
        - The HX711 datasheet contains a reference driver in both ASM and C

09-23-24

Interface with eInk Display using ESP32 on Wokwi

Details

- Status: Incomplete
- Hardware
  - ESP32-S3
  - ePaper 2.9"
- Software
  - Wokwi
  - C++

Notes

- I cannot find an example Wokwi project using the Adafruit EDP Library that works properly
  - The error seems to be with the GXEPD2 library
    - ```
      Error installing GxEPD2: Library install failed: testing local archive integrity: testing archive size: fetched archive size differs from size specified in index: 176128 != 4501132
      ```
- I'll need to learn how to upload custom libraries
  - It might be easier if I set up Wokwi for VS Code

Start Project Schematic

Details

- Status: Incomplete
- Software
  - KiCAD

Notes

- We needed to make a custom symbol within KiCAD
  - This can be found within
- I cannot figure out what electrical type MISO and MOSI should be
  - Related Forum Post
- I think Adafruit really wants us to use a breakout board to interface with the eInk display
      ```

    - Without it, we'd have to solder headers onto the display if we wanted to use it on a breadboard

Misc

```
// 2.9" Tricolor Featherwing or Breakout with UC8151D chipset
// ThinkInk_290_Tricolor_Z13 display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS,
// EPD_BUSY, EPD_SPI);
```

- I found this on the Adafruit EDP Library. This might give insight on to what data pins actually matter.

09-24-24

Start Project Schematic

Details

- Status: Version 001
- Hardware
  - ESP32-S3
  - HX711
  - Custom UC8151D
  - Custom Strain Gauge
  - MCP72831

Wiring Guide

Load Cell Wiring Guide to HX711

```
- E+ (red wire) stands for Excitation Positive of the load cell
    - Connects to the Analog Voltage Supply Pin (AVDD/VIN) of the HX711
- E- (black ire) stands for Excitation Negative of the load cell
    - Connects to the Analog Ground (AGND/GND) of the HX711
- A+ (white wire) stands for Signal Positive of the load cell
    - Connects to the Positive Input for channel A (INA+ or A+) of the XH711
- A- (green wire) stands for Signal Negative of the load cell
    - Connect to the Negative Input for channel A (INA- or A-) of the HX711

*** Important: Since no datasheet came with the acquired load cell. Assume
manufacturer followed this convention.
    1. Red Wire (E+): Positive excitation
    2. Black Wire (E-): Negative excitation
    3. White Wire (A+): Positive signal output
    4. Green Wire (A-): Negative signal output
```

HX711 to ESP32-S3

```
- HX711 SCK (Serial Clock) or PD_SCK
    - Connect to any GPIO
- HX711 DATA (Data Output) or DOUT
    - Connect to any GPIO
- HX711 E+ (Excitation Positive)
    - Connect to AVDD/VIN
- HX711 E- (Excitation Negative)
    - Connect to GND
```

ESP32 to E-ink Display

```
- Power Connections:
    - VIN → Connect to 5V (or 3.3V, depending on your display's requirement)
    - 3.3V → Connect to VDD3P3 on the ESP32-S3
    - GND → Connect to GND on the ESP32-S3

- SPI Connections:
    - SCK → Connect to SCK of ESP32-S3 (GPIO36 on the ESP32-S3 Feather)
    - MISO → Connect to MISO of ESP32-S3 (GPIO37 on the ESP32-S3 Feather)
    - MOSI → Connect to MOSI of ESP32-S3 (GPIO35 on the ESP32-S3 Feather)
    - ECS (Chip Select) → Connect to any GPIO
    - D/C (Data/Command) → Connect to any GPIO
    - SRCS (Source Select) → Connect to any GPIO
    - SDCS (Data Chip Select) → Connect to any GPIO
    - RST (Reset) → Connect to any GPIO
    - BUSY → Connect to any GPIO
    - ENA (Enable) → Connect to any GPIO

*** Necessary Pins for E-ink Display
    - SCK: Required for clocking the data into the E-ink display.
    - MISO: Typically used for data feedback from the E-ink display to the ESP32-S3. Not always necessary unless display requires it for certain operations (like reading status).
    - MOSI: Essential for sending data from the ESP32-S3 to the E-ink display.
    - ECS: This is important to tell the E-ink display when to listen to the data being sent. Without this, the display may not respond correctly.
    - D/C : Important for distinguishing between data and command instructions sent to the display. It helps the display interpret what type of data it is receiving.
    - RST: Often required to initialize the display. Without this, may not be able to reset the display or clear it properly.

*** Optional Pinks for E-ink Display
    - BUSY: Indicates when the display is busy processing data. It's useful for synchronizing operations but may not be strictly necessary for all applications.
    - SRCS: Might be used to select between different sources for data or power. Depending on your display.
    - SDCS: Similar to ECS, this pin is likely used to select the data source. If display has both ECS and SDCS, check the documentation to see if both are necessary, as sometimes they can serve similar purposes.
    - ENA: Typically enables or disables the display. It may be required to power on the display or enable certain functions. It's a common pin in various display modules.
```

Notes

- ESP32-S3 has a lot more features than your standard ESP32. I am wondering if we should build this with an ESP32 instead of mounting on a ESP32-S3.

Interface with eInk display using ESP32-S3 IRL

Details

- Status: Incomplete
- Hardware:
  - Adafruit ESP32-S3 Feather
  - R: 220 Ω
  - LED (Red)

Soldering Headers

- In order to breadboard with these components, we had to solder headers onto the ESP32-S3, EYESPI Breakout Board, and HX711 breakout board
  - HX711
    - Might need to reflow some solder joints

Notes

```
A fatal error occurred: Could not open COM6, the port doesn't exist
Failed uploading: uploading error: exit status 2
```

In order to solve this error, I have attempted the following:

- Uninstalled and reinstalled Arduino IDE
- Downloaded and installed CP210x USB to UART Bridge VCP Drivers from here
- Restarted my computer multiple times
- Swapped USB-C cable

10-02-24

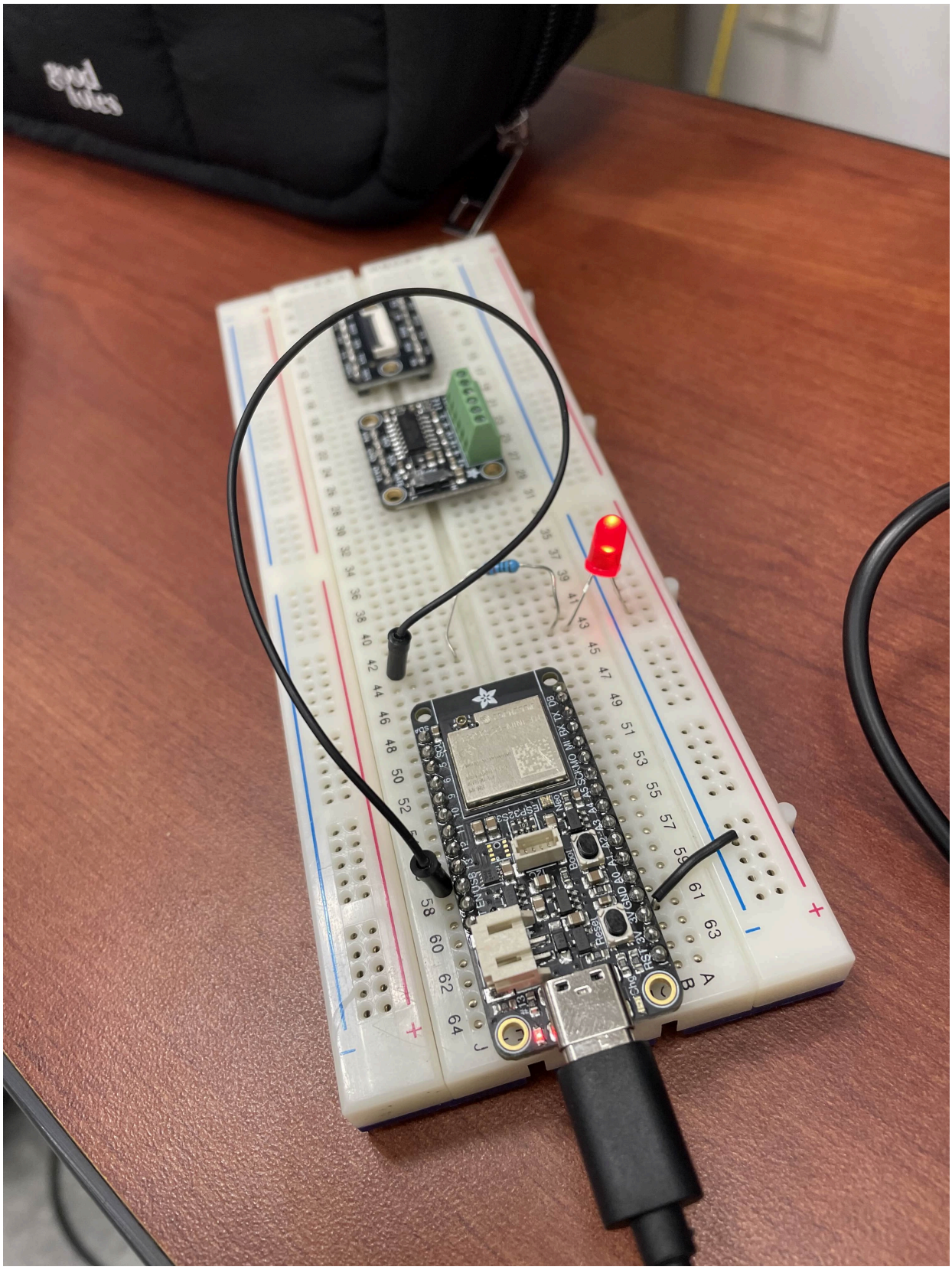
Blink LED IRL

Details

- Hardware
  - Adafruit Feather ESP32-S3 8MB No PSRAM
  - LED (Red)
  - 220 Ω Resistor
- Software
  - Arduino IDE
  - C++

Notes

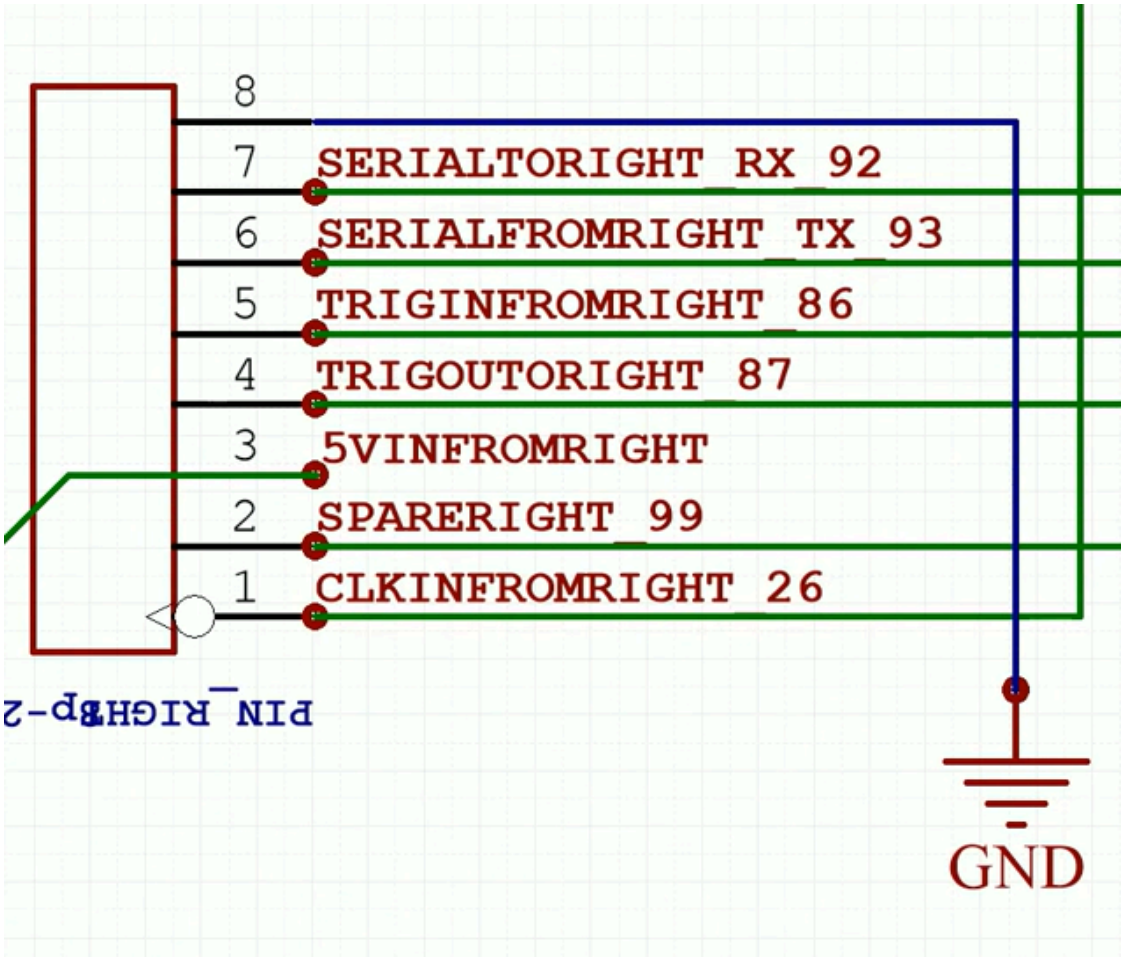
After many long hours of troubleshooting, I was finally able to blink an LED following a factory reset using this tutorial. Huzzah!



10-08-24

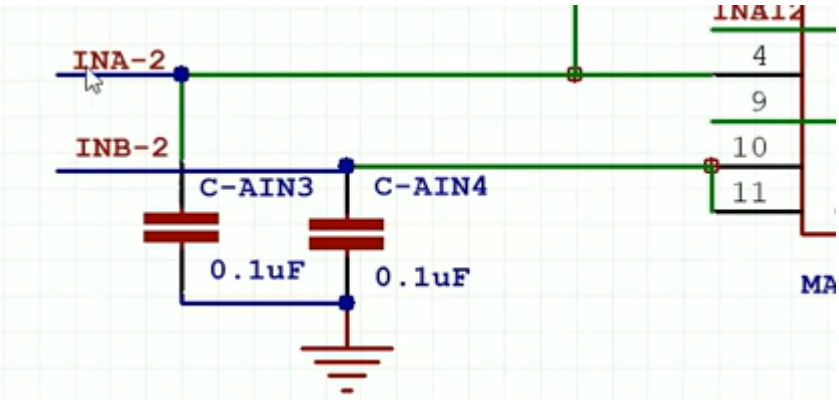
Create a Nice Readable Schematic

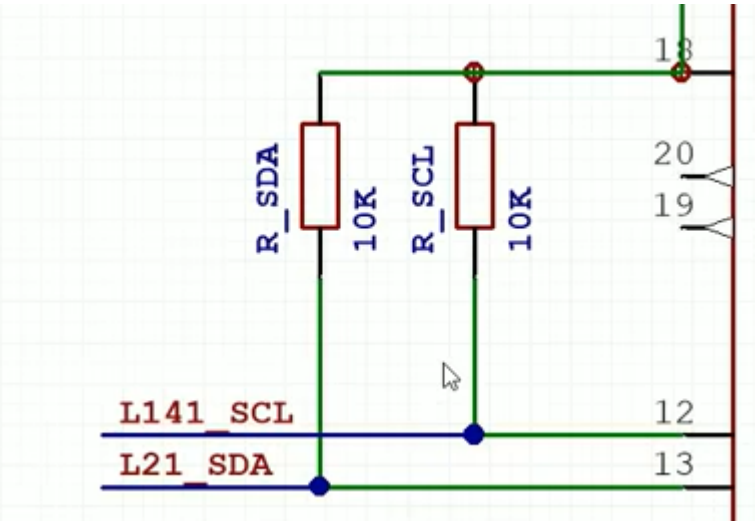
Rules/Conventions

- Signals should flow from left to right
- PWR on top, GND always down
    - 

- Lines should never be on an angle
    - Up, down, left, right
- Never, **ever** have lines going through your chip symbols
- Annotations should be horizontal
    - The person shouldn't have to tilt their head to read them
- Designators and their values can either be together on one side, or on opposite sides of a symbol
    - Just make sure you are consistent
- Set pins to snap grid when creating symbols
- No need to show footprints on schematic
- Net name should be on top of netlist/lines
- Don't assign nets to things that have ports (+3V3 net when you have a 3V3 port)

- Avoid crossing wires if possible
    - Edit symbol to move pins even if it doesn't match the physical layout
- Don't label nets if you are not going to use them
- Signify if a symbol is going out to prevent confusion (someone might think it's a completed circuit)





- Don't run wires through the schematic if you don't have to
- Connectors don't have to be next to the components; however, you must properly label the nets
- Net names should be meaningful and nonredundant
- Either change the symbol to all match the physical in mapping or group them up by functionality so that they support signal flow
    - If you edit for symbol flow, make sure that the inputs are on the left and outputs are on the right
- Be consistent with schematic symbols i.e. for capacitors, resistors
- Do not use decimal points for values in your schematic
    - 0.01uF > 100nF or 100n
- Use ports for outputs
- Label sections with notes
- Make LEDS and other symbols vertical
- Don't create four way junctions off a single node

- Designators can be renamed if you truly need to but keep it short and sweet
- Use A4 schematic size since we're yankees

10-11-24

Interface with eInk display using ESP32-S3 IRL

- Installed "adafruit_EPD", "adafruit GFX", and "adafruit ImageReader" library from arduino
- From https://github.com/adafruit/Adafruit_EPD, I downloaded and moved the src folder to the arduino library folder

So far the example code I copied has finally compiled. Next goal is to try and change image in the display. However, will need to verify the pin connections again.

10-13-24

Interface with eInk display using ESP32-S3 IRL

Example Code for E-Ink from Adafruit

```
#include "Adafruit_ThinkInk.h"

#define EPD_DC 10
#define EPD_CS 5
#define EPD_BUSY -1 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 11
#define EPD_RESET 9  // can set to -1 and sh\are with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI

ThinkInk_290_Tricolor_Z13 display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in red/black/white");
  display.begin(THINKINK_TRICOLOR);
}

void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 144) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Tri");
  display.setTextColor(EPD_RED);
  display.print("Color");
  display.display();

  delay(15000);

  Serial.println("Color rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 3, 0, display.width() / 3,
                   display.height(), EPD_BLACK);
  display.fillRect((display.width() * 2) / 3, 0, display.width() / 3,
                   display.height(), EPD_RED);
  display.display();

  delay(15000);
```

```
  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(15000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }

  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_RED);
  }
  display.display();

  delay(15000); # 15 second delay before going into the next display
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}

- Note when this code ran
    - Multiple flickering inbetween display changes

- Other issues
    - The process can be confusing to have the ESP32 execute the code to the E-Ink
        - Things that worked
            - Have to select the USB port again after it compiles
            - Have to press the reset button on the ESP32

- EYESPI Breakout
    - https://learn.adafruit.com/adafruit-eyespi-breakout-board/pinouts
```

10-15-24

Interface with HX711 and Load Cell IRL

Wiring from Load Cell to HX711

- Red wire to E+
- Black wire to E-
- Green wire to B+
- White wire t0 B-

Wiring from HX711 to ESP32-S3

- VIN (Red wire) to 3V
- GND (Black wire) to GND
- DATA (Yellow wire) to GPIO8/A5
- SCK (Blue wire) to GPIO14/A4

Library

- Adafruit_HX711 library

HX711 Schematics

https://learn.adafruit.com/adafruit-hx711-24-bit-adc/downloads

Example Code

```
#include "Adafruit_HX711.h"

// Define the pins for the HX711 communication
const uint8_t DATA_PIN = 2;  // Can use any pins!
const uint8_t CLOCK_PIN = 3; // Can use any pins!

Adafruit_HX711 hx711(DATA_PIN, CLOCK_PIN);

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

  // wait for serial port to connect. Needed for native USB port only
  while (!Serial) {
    delay(10);
  }

  Serial.println("Adafruit HX711 Test!");

  // Initialize the HX711
```

```
  hx711.begin();

  // read and toss 3 values each
  Serial.println("Tareing....");
  for (uint8_t t=0; t<3; t++) {
    hx711.tareA(hx711.readChannelRaw(CHAN_A_GAIN_128));
    hx711.tareA(hx711.readChannelRaw(CHAN_A_GAIN_128));
    hx711.tareB(hx711.readChannelRaw(CHAN_B_GAIN_32));
    hx711.tareB(hx711.readChannelRaw(CHAN_B_GAIN_32));
  }
}

void loop() {
  // Read from Channel A with Gain 128, can also try CHAN_A_GAIN_64 or
CHAN_B_GAIN_32
  // since the read is blocking this will not be more than 10 or 80 SPS (L or H
switch)
  int32_t weightA128 = hx711.readChannelBlocking(CHAN_A_GAIN_128);
  Serial.print("Channel A (Gain 128): ");
  Serial.println(weightA128);

  // Read from Channel A with Gain 128, can also try CHAN_A_GAIN_64 or
CHAN_B_GAIN_32
  int32_t weightB32 = hx711.readChannelBlocking(CHAN_B_GAIN_32);
  Serial.print("Channel B (Gain 32): ");
  Serial.println(weightB32);
}
```

Issues

- Serial monitor always outputs zero even after applying pressure to the strain gauge

10-16-24

Setup Wokwi for VS Code

I was able to setup Wokwi for VS Code using Wokwi's Guide. I'm interested in exploring PlatformIO as an IDE for programming the physical ESP32-S3 itself.

10-22-24

Interface with HX711 and Load Cell IRL

Adafruit Example Code

```
#include <Arduino.h>
#include "HX711.h"

// HX711 circuit wiring
const int LOADCELL_DOUT_PIN = 9;
const int LOADCELL_SCK_PIN = 8;

HX711 scale;

void setup() {
  Serial.begin(57600);
  Serial.println("HX711 Demo");
  Serial.println("Initializing the scale");

  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);

  Serial.println("Before setting up the scale:");
  Serial.print("read: \t\t");
  Serial.println(scale.read());      // print a raw reading from the ADC

  Serial.print("read average: \t\t");
  Serial.println(scale.read_average(20));   // print the average of 20 readings from
the ADC

  Serial.print("get value: \t\t");
  Serial.println(scale.get_value(5));   // print the average of 5 readings from the
ADC minus the tare weight (not set yet)

  Serial.print("get units: \t\t");
  Serial.println(scale.get_units(5), 1);  // print the average of 5 readings from
the ADC minus tare weight (not set) divided
            // by the SCALE parameter (not set yet)

  scale.set_scale(765);
  //scale.set_scale(-471.497);                       // this value is obtained by
calibrating the scale with known weights; see the README for details
  scale.tare();               // reset the scale to 0

  Serial.println("After setting up the scale:");

  Serial.print("read: \t\t");
  Serial.println(scale.read());                 // print a raw reading from the ADC

  Serial.print("read average: \t\t");
```

```
  Serial.println(scale.read_average(20));       // print the average of 20 readings
from the ADC

  Serial.print("get value: \t\t");
  Serial.println(scale.get_value(5));   // print the average of 5 readings from the
ADC minus the tare weight, set with tare()

  Serial.print("get units: \t\t");
  Serial.println(scale.get_units(5), 1);        // print the average of 5 readings
from the ADC minus tare weight, divided
            // by the SCALE parameter set with set_scale

  Serial.println("Readings:");
}

void loop() {
  Serial.print("one reading:\t");
  Serial.print(scale.get_units(), 1);
  Serial.print("\t| average:\t");
  Serial.println(scale.get_units(10), 5);

  delay(5000);
}
```

Random Nerd Tutorials Code

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/arduino-load-cell-
hx711/

  Permission is hereby granted, free of charge, to any person obtaining a copy
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be included in all
  copies or substantial portions of the Software.
*/

// Calibrating the load cell
#include "HX711.h"

// HX711 circuit wiring
const int LOADCELL_DOUT_PIN = 2;
const int LOADCELL_SCK_PIN = 3;

HX711 scale;

void setup() {
  Serial.begin(57600);
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
}
```

```
void loop() {

  if (scale.is_ready()) {
    scale.set_scale();
    Serial.println("Tare... remove any weights from the scale.");
    delay(5000);
    scale.tare();
    Serial.println("Tare done...");
    Serial.print("Place a known weight on the scale...");
    delay(5000);
    long reading = scale.get_units(10);
    Serial.print("Result: ");
    Serial.println(reading);
  }
  else {
    Serial.println("HX711 not found.");
  }
  delay(1000);
}

//calibration factor will be the (reading)/(known weight)
```

Notes

- The test code from adafruit library was not working as intended
    - The output is either 0 or some constant value that does not change no matter how much pressure is applied to the gauge

10-29-24

Interface with OLED display IRL

Components

- SPI OLED Display 128x32
- Arduino Uno

I followed this Adafruit Guide to test the SPI OLED on an Arduino Uno.

This code can be found in here

Adafruit ESP32 Feather

- Load Cell: Successfully calibrated and now measures weight with acceptable tolerance
- OLED Display: Functions as intended when tested with example code
- Integration: Preliminary code to connect the load cell and OLED display
    - Code found here
    - Result found here
- Power/Wiring issues
    - OLED display or HX711 does not receive power at certain points along the power rail
        - This was evident when rewiring peripherals
        - As of right now, this setup works
- Our known weight is a 60.3 grams plier
    - The result is acceptable, as can be seen from this image

10-30-24

Update schematic to include OLED display

General

- Added OLED (SSD1306) datasheet to documentation branch

Eagle File to KiCAD

- Why does Adafruit only offer Eagle files :c

Symbol Creation

- Figured out that we don't need it anyways
    - Added 08x02 connectors to socket display

Make a Nice Readable Schematic

LiPo Battery Charger

- Replaced SparkFun ESP32 Thing circuit with prettier circuit from Adafruit ESP32-S3 Feather

Power and Filtering

- Changed custom JST connector to boring KiCAD 01x02 connector
    - I should place this on the left side to have it function clearly as an input

Load Cell

- Edited symbol so no wires crisscross

ADC

- Added No Connect flags

Microcontroller

- Added No Connect flags

Display

- Made wiring nice and pretty
- Added text
    - I should add more text about the actual display

11-09-24

Update "Stage 2: ADC" based on "Figure 4. Reference PCB Board Schematic"

General

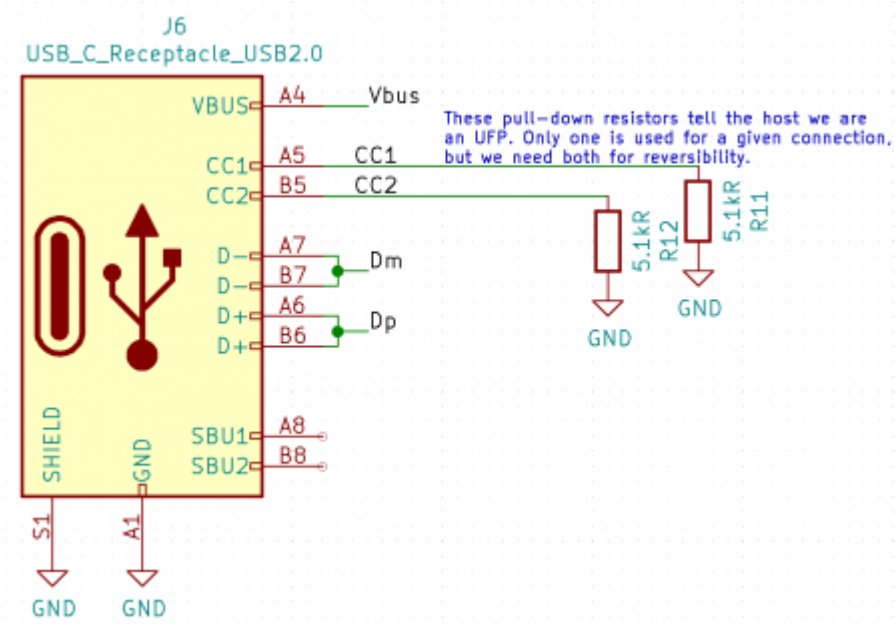
- Updated Stage 1 and Stage 2 based on Figure 4 on HX711 datasheet

11-12-24

Add USB-C receptacle to schematic

General

- Added USB-C receptable
- Read this article to learn more about reasoning behind design



11-13-24

Add pushbuttons to prototype

- Added three button to do three different functionalities
  - Tare feature to zero the scale after putting weight when button is pressed
  - Timer function that starts to count and show on the OLED display when the button is pressed
  - Wake up
- Added an active buzzer for sound effects

Configure Deep Sleep mode

- Added a deep sleep mode by following this website
- Currently, the threshold weight for deep sleep is <1g

Create flow rate display algorithm

- Although this works, we still need to figure out a better method of testing

Run tests on breadboard prototype

- Tested the core functionalities of our prototype for our midterm presentation
  - More details can be found in our testing document
- Scale
  - Accuracy Test
    - Benchmarked three different objects to determine the scale's accuracy
  - Precision Test
    - Weighed the same object multiple times to determine consistency
- Timer
  - Ran a known timer at the same time to determine accuracy
- Deep sleep timer
  - Tested deep sleep timer by letting an object rest for more than 30 seconds, removing it, and then waiting