Clue Coffee Scale
Created by Jan Goolsbey

https://learn.adafruit.com/clue-coffee-scale

Last updated on 2023-08-29 04:24:14 PM EDT
# Table of Contents

## Overview
- Parts

## Features and Operation

## Build the Coffee Scale
- Attach Plates
- Mount Scale to Drip Tray Grid
- Wire Prep
- Clue Connection
- Clue Mount
- Battery Pack

## CircuitPython on CLUE
- Set up CircuitPython Quick Start!

## Code the Coffee Scale
- Text Editor
- Download the Project Bundle
- code.py

## Code Details
- Import and Set Defaults
- Calibration Ratio
- Instantiate the Sensor and Display
- Display the Background Bitmap Image
- Define and Display the Text Labels and Graphic Elements
- Helpers
- Activate the Sensor and Prepare to Loop
- The Primary Code Loop

## Calibrate the Load Cell
- Calibrator
- Calibrator Code Listing
Overview

Consistently brewing that crowning cup of coffee involves attention to the precise addition of ingredients and control of the brew ratio, tempered with practiced technique. Perhaps it's time to apply a little more science to the art.

The Clue Coffee Scale can accurately measure the weight of the ground beans or the extraction output, getting closer to an exact and repeatable approach for creating your preferred brew. To measure the contents of a container, the scale can be zeroed with a push of a button to subtract the container's tare weight.

The scale uses a load cell sensor device that consists of a very sensitive resistance array (a strain gauge) attached to a metal bending beam. When connected to an exciter voltage, the resistance array provides a differential voltage output signal proportional to the torque of the mass placed on the load cell. The signal voltage is very low so a sensitive amplifier and high-resolution analog to digital converter (ADC) are used to provide a microcontroller-compatible measurement.

The Adafruit NAU7802 24-Bit ADC - STEMMA QT () board has everything needed to connect a load cell and provide measurements for the Clue. Add a CircuitPython program, and the Clue Coffee Scale is born.

John Park built a version of the coffee scale for use with his primo espresso station. Check out the Build the Coffee Scale section of this guide for the step-by-step approach he used for his custom scale. A trenta-sized thank you goes to John for creating a practical and beautiful working scale.
This guide will show how to build, calibrate, and code the scale. Your perfect cup of Java awaits.

A load cell is a precision device particularly when it is paired with a sensitive high-resolution ADC such as the NAU7802. However, because of component variations and software accuracy limitations, the scale SHOULD NOT BE USED for safety, health, scientific, or commercial weight measurement applications.

Parts

Adafruit CLUE - nRF52840 Express with Bluetooth LE
Do you feel like you just don't have a CLUE? Well, we can help with that - get a CLUE here at Adafruit by picking up this sensor-packed development board. We wanted to build some...
https://www.adafruit.com/product/4500
Clear Acrylic Enclosure + Hardware Kit for Adafruit CLUE
Here is a chic minimalist enclosure for your CLUE board! This case has been laser-cut specifically to accommodate the TFT display, tactile buttons, and...
https://www.adafruit.com/product/4675

Strain Gauge Load Cell - 4 Wires - 1Kg
A strain gauge is a type of electronic sensor used to measure force or strain (big surprise there). They are made of an insulating flexible backing with a metallic...
https://www.adafruit.com/product/4540

Adafruit NAU7802 24-Bit ADC - STEMMA QT / Qwiic
If you are feeling the stress and strain of modern life a Wheatstone bridge and you want to quantify it, this handy breakout will do the job, no sweat! The Adafruit...
https://www.adafruit.com/product/4538

STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long
This 4-wire cable is a little over 100mm / 4" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...
https://www.adafruit.com/product/4210

©Adafruit Industries
<table>
<thead>
<tr>
<th>Item</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 x USB cable</td>
<td><a href="https://www.adafruit.com/product/592">https://www.adafruit.com/product/592</a></td>
</tr>
<tr>
<td>USB A to Micro-B - 3 foot long</td>
<td></td>
</tr>
<tr>
<td>1 x Clear Adhesive Squares</td>
<td><a href="https://www.adafruit.com/product/4813">https://www.adafruit.com/product/4813</a></td>
</tr>
<tr>
<td>6 pack - UGlu Dashes</td>
<td></td>
</tr>
<tr>
<td>1 x Aluminum Extrusion Corner Brace Support for 20x20</td>
<td><a href="https://www.adafruit.com/product/1155">https://www.adafruit.com/product/1155</a></td>
</tr>
<tr>
<td>1 x Coupling Plate</td>
<td><a href="https://www.adafruit.com/product/1216">https://www.adafruit.com/product/1216</a></td>
</tr>
<tr>
<td>3 Holes - 20x20 Aluminum Extrusion</td>
<td></td>
</tr>
<tr>
<td>1 x Cross-Plate</td>
<td><a href="https://www.adafruit.com/product/1220">https://www.adafruit.com/product/1220</a></td>
</tr>
<tr>
<td>for 2020 Aluminum Extrusion</td>
<td></td>
</tr>
<tr>
<td>1 x Wire Ferrule Kit</td>
<td><a href="https://www.adafruit.com/product/5131">https://www.adafruit.com/product/5131</a></td>
</tr>
<tr>
<td>800 pieces</td>
<td></td>
</tr>
<tr>
<td>3 x AAA Battery Holder with On/Off Switch and 2-Pin JST</td>
<td><a href="https://www.adafruit.com/product/727">https://www.adafruit.com/product/727</a></td>
</tr>
<tr>
<td>This battery holder connects 3 AAA batteries together in</td>
<td></td>
</tr>
<tr>
<td>series for powering all kinds of projects. We spec’d</td>
<td></td>
</tr>
<tr>
<td>these out because the box is slim, and 3 AAA’s add up to</td>
<td></td>
</tr>
<tr>
<td>about...</td>
<td></td>
</tr>
</tbody>
</table>

©Adafruit Industries
Features and Operation

- The Clue Coffee Scale continuously displays the load cell weight measurement in both grams and ounces.
- The five-division graduated scale in the center of the display is calibrated in grams and shows the measured weight relative to the maximum value of 100 grams (default).
- The indicator bubble travels up and down the graduated scale, pointing to the currently measured value in grams. The interior of the indicator bubble will turn red if the measured value is outside of the graduated scale range.
- The Clue A button is used to zero or tare the scale. Press the button until a beep is heard, then release. The interior of the bubble and the Clue's NeoPixel will turn red while zeroing/taring. A second beep is heard when the scale completes the zero/tare process.
Build the Coffee Scale

This is one specific implementation of the scale for use on an espresso machine. Your build will vary depending on the target machine.
Attach Plates

In an attempt to make a minimal scale to integrate with the espresso machine's drip tray, only two pieces of aluminum, the load cell, and a few fasteners are used.

The fixed end of the load cell (the side with the wires) is tapped with M5 screw holes, while the weighing end that can flex is tapped with M4.

Using an M4 screw, attach the large cross-plate to the underside of the load cell. (Note the sticker on the end of the cell with the load rating and arrow.)

Then, attach the 3-hole coupling plate with an M5 screw to the underside of the load cell at the fixed end.
Mount Scale to Drip Tray Grid

Use M3 screws, spacers, and nuts to attach the coupling plate to the drip tray grid as shown.

The spacers are necessary to give the load cell clearance to flex under load as shown in the fourth picture here.
Wire Prep

If you like, you can add some short lengths of heat shrink tubing to collect the four wires, as well as add small ferrules to their ends -- this makes it easier to connect them to the screw terminals on the ADC board.

These wires are so thin you may want to solder them to the inside of the ferrules at the tip rather than crimp them.
Clue Connection
Add the Clue case to the clue, but leave off one screw in the upper right corner.

Use a longer M2.5 screw and nut to connect the ADC board to the back of the Clue.

Then, connect the Clue and ADC via STEMMA QT cable.
Clue Mount
Mount the 2020 corner brace to the front of the drip tray grid using an M3 screw, nut, and washer or spacer.

Then mount the Clue to the brace using the remaining M3 nylon screw and nut from the acrylic case kit.
Battery Pack

Connect the AAA battery pack to the Clue. Then, use the adhesive square to connect the battery pack lid to the drip tray's side.

This will allow you to pull out the battery pack to change the batteries when needed.
CircuitPython on CLUE

CircuitPython is a derivative of MicroPython designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the CIRCUITPY flash drive to iterate.

The following instructions will show you how to install CircuitPython. If you've already installed CircuitPython but are looking to update it or reinstall it, the same steps work for that as well!
Set up CircuitPython Quick Start!

Follow this quick step-by-step for super-fast Python power :)

Download the latest version of CircuitPython for CLUE from circuitpython.org

Click the link above to download the latest version of CircuitPython for the CLUE.

Download and save it to your desktop (or wherever is handy).

Plug your CLUE into your computer using a known-good USB cable.

A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

Double-click the Reset button on the top (magenta arrow) on your board, and you will see the NeoPixel RGB LED (green arrow) turn green. If it turns red, check the USB cable, try another USB port, etc. Note: The little red LED next to the USB connector will pulse red. That's ok!

If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!
You will see a new disk drive appear called CLUEBOOT.

Drag the adafruit-circuitpython-clue-etc.uf2 file to CLUEBOOT.

The LED will flash. Then, the CLUEBOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

If this is the first time you're installing CircuitPython or you're doing a completely fresh install after erasing the filesystem, you will have two files - boot_out.txt, and code.py, and one folder - lib on your CIRCUITPY drive.

If CircuitPython was already installed, the files present before reloading CircuitPython should still be present on your CIRCUITPY drive. Loading CircuitPython will not create new files if there was already a CircuitPython filesystem present.

That's it, you're done! :)

©Adafruit Industries
Code the Coffee Scale

First make sure you are running the latest version of Adafruit CircuitPython for your board.

Text Editor

Adafruit recommends using the Mu editor for using your CircuitPython code with the Feather. You can get more info in this guide.

Alternatively, you can use any text editor that saves text files.

Download the Project Bundle

Your Clue Coffee Scale project will use a specific set of CircuitPython libraries, a bitmap image, fonts, a calibration method, and the code.py file. To get everything you need, click on the Download Project Bundle link below, and uncompress the .zip file.

Connect your CLUE to your computer via a known good data+power cable. Look for the CIRCUITPY drive which pops up in your Finder or File Explorer (depending on your operating system).

Drag the contents of the uncompressed bundle directory onto your Clue board's CIRCUITPY drive, replacing any existing files or directories with the same names, and adding any new ones that are necessary.

The Project Bundle contains the following folders and files:

- clue_scale_bkg.bmp  the background bitmap image
- clue_scale_calibrator.py  the load cell calibration program code
- code.py  the main Coffee Scale program code
- fonts folder:
  - Helvetica-Bold-24.bdf  font file
  - OpenSans-16.bdf  font file
  - OpenSans-9.bdf  font file
lib folder containing these required libraries:

- adafruit_apds9960
- adafruit_bitmap_font
- adafruit_bmp280
- adafruit_clue
- adafruit_display_shapes
- adafruit_display_text
- adafruit_lis3mdl
- adafruit_lsm6ds
- adafruit_register
- adafruit_sht31d
- cedargrove_nau7802
- neopixel
- simpleio

The cedargrove_nau7802 library driver can also be found in the CircuitPython Community Bundle. For the curious, NAU7802 driver API information is available in Cedar Grove's Github repository.

code.py

The main CircuitPython code for the Coffee Scale is contained in the project zip folder as code.py. Copy this and all the other bundle files to the main (root) folder of the CIRCUITPY drive that appears when your Clue is connected to your computer via a known good USB cable.

Before using the scale, you'll need to update code.py with a calibration ratio that is unique to the load cell attached to the NAU7802 Stemma breakout. See the guide section, Calibrate the Load Cell for instructions. You should only have to measure and record the load cell calibration ratio once.

```python
# SPDX-FileCopyrightText: 2023 Jan Goolsbey for Adafruit Industries
# SPDX-License-Identifier: MIT
#
# clue_scale_code.py
# 2023-01-13 v1.2.1
#
# Clue Scale - Single Channel Version
# Adafruit NAU7802 Stemma breakout example

# import clue_scale_calibrator  # Uncomment to run calibrator method

import time
import board
from simpleio import map_range
from adafruit_clue import clue
from adafruit_display_shapes.circle import Circle
```
from adafruit_display_text.label import Label
from adafruit_bitmap_font import bitmap_font
import displayio
from cedargrove_nau7802 import NAU7802

clue.pixel.brightness = 0.2  # Set NeoPixel brightness
clue.pixel[0] = clue.YELLOW  # Set status indicator to yellow (initializing)

# Set Scale Defaults
MAX_GR = 100  # Maximum (full-scale) display range in grams
DEFAULT_GAIN = 128  # Default gain for internal PGA
SAMPLE_AVG = 5  # Number of sample values to average
SCALE_NAME_1 = "COFFEE"  # 6 characters maximum
SCALE_NAME_2 = "SCALE"  # 6 characters maximum

"""Enter the calibration ratio for the individual load cell in-use. The ratio is
composed of the reference weight in grams divided by the raw reading. For
example, a raw reading of 215300 for a 100 gram weight results in a calibration
ratio of 100 / 215300. Use the clue_scale_single_calibrate method to obtain the
raw value.
FYI: A US dime coin weighs 2.268 grams or 0.079 ounces."""
CALIB_RATIO = 100 / 215300  # load cell serial#4540-02

# Instantiate the Sensor and Display
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
# microcontroller
nau7802 = NAU7802(i2c, address=0x2A, active_channels=1)
display = board.DISPLAY
scale_group = displayio.Group()

FONT_0 = bitmap_font.load_font("/fonts/Helvetica-Bold-24.bdf")
FONT_1 = bitmap_font.load_font("/fonts/OpenSans-16.bdf")
FONT_2 = bitmap_font.load_font("/fonts/OpenSans-9.bdf")

# Display the Background Bitmap Image
bkg = displayio.OnDiskBitmap("/clue_scale_bkg.bmp")
_background = displayio.TileGrid(bkg, pixel_shader=bkg.pixel_shader, x=0, y=0)
scale_group.append(_background)

# Define and Display the Text Labels and Graphic Elements
# Place the project name on either side of the graduated scale
scale_name_1 = Label(FONT_1, text=SCALE_NAME_1, color=clue.CYAN)
scale_name_1.anchor_point = (0.5, 0.5)
scale_name_1.anchored_position = (40, 96)
scale_group.append(scale_name_1)

scale_name_2 = Label(FONT_1, text=SCALE_NAME_2, color=clue.CYAN)
scale_name_2.anchor_point = (0.5, 0.5)
scale_name_2.anchored_position = (199, 96)
scale_group.append(scale_name_2)

# Define the zeroing button graphic
zero_button_circle = Circle(14, 152, 14, fill=None, outline=clue.RED, stroke=2)
scale_group.append(zero_button_circle)

zero_button_label = Label(FONT_1, text="Z", color=clue.RED)
zero_button_label.x = 8
zero_button_label.y = 150
scale_group.append(zero_button_label)

# Place tickmark labels next to the graduated scale
for i in range(-1, 6):
    tick_value = Label(FONT_2, text=str((MAX_GR) // 5 * i), color=clue.CYAN)
    if i == -1:
        tick_value.anchor_point = (1.0, 1.1)
    elif i == 5:
        tick_value.anchor_point = (1.0, 0.0)
    else:
        tick_value.anchor_point = (0.5, 0.5)
    scale_group.append(tick_value)
else:
    tick_value.anchor_point = (1.0, 0.5)
tick_value.anchored_position = (99, 201 - (i * 40))
scale_group.append(tick_value)

# Place the grams and ounces labels and values near the bottom of the display
grams_label = Label(FONT_0, text="grams", color=clue.BLUE)
grams_label.anchor_point = (1.0, 0)
grams_label.anchored_position = (80, 216)
scale_group.append(grams_label)

ounces_label = Label(FONT_0, text="ounces", color=clue.BLUE)
ounces_label.anchor_point = (1.0, 0)
ounces_label.anchored_position = (230, 216)
scale_group.append(ounces_label)

grams_value = Label(FONT_0, text="0.0", color=clue.WHITE)
grams_value.anchor_point = (1.0, 0.5)
grams_value.anchored_position = (80, 200)
scale_group.append(grams_value)

ounces_value = Label(FONT_0, text="0.00", color=clue.WHITE)
ounces_value.anchor_point = (1.0, 0.5)
ounces_value.anchored_position = (230, 200)
scale_group.append(ounces_value)

# Define the moveable indicator bubble
indicator_group = displayio.Group()
bubble = Circle(120, 200, 10, fill=clue.YELLOW, outline=clue.YELLOW, stroke=3)
indicator_group.append(bubble)
scale_group.append(indicator_group)
display.show(scale_group)

# Helpers
def zero_channel():
    """Prepare internal amplifier settings and zero the current channel. Use
    after power-up, a new channel is selected, or to adjust for measurement
    drift. Can be used to zero the scale with a tare weight.
    The nau7802.calibrate function used here does not calibrate the load cell,
    but sets the NAU7802 internals to prepare for measuring input signals."""
    nau7802.calibrate("INTERNAL")
    nau7802.calibrate("OFFSET")

def read(samples=1):
    """Read and average consecutive raw samples; return averaged value."""
    sample_sum = 0
    sample_count = samples
    while sample_count > 0:
        if nau7802.available():
            sample_sum = sample_sum + nau7802.read()
            sample_count -= 1
    return int(sample_sum / samples)

# Activate the Sensor
# Enable the internal analog circuitry, set gain, and zero
nau7802.enable(True)
nau7802.gain = DEFAULT_GAIN
zero_channel()

# Play "welcome" tones
clue.play_tone(1660, 0.15)
clue.play_tone(1440, 0.15)

# The Primary Code Loop
# Read sensor, move bubble, and display values
while True:
    clue.pixel[0] = clue.GREEN  # Set status indicator to green (ready)

    # Read the raw scale value and scale for grams and ounces
    value = read(SAMPLE_AVG)
    mass_grams = round(value * CALIB_RATIO, 1)
    mass_ounces = round(mass_grams * 0.03527, 2)
    grams_value.text = f"{mass_grams:.1f}"  
    ounces_value.text = f"{mass_ounces:.2f}"  
    print(f" {mass_grams:.1f} grams   {mass_ounces:.2f} ounces")

    # Reposition the indicator bubble based on grams value
    min_gr = (MAX_GR // 5) * -1  # Minimum display value
    bubble.y = int(map_range(mass_grams, min_gr, MAX_GR, 240, 0)) - 10
    if mass_grams > MAX_GR or mass_grams < min_gr:
        bubble.fill = clue.RED
    else:
        bubble.fill = None

    # Check to see if the zeroing button is pressed
    if clue.button_a:
        # Zero the sensor
        clue.pixel[0] = clue.RED  # Set status indicator to red (stopped)
        bubble.fill = clue.RED  # Set bubble center to red (stopped)
        clue.play_tone(1660, 0.3)  # Play "button pressed" tone
        zero_channel()

        while clue.button_a:
            # Wait until the button is released
            time.sleep(0.1)

            clue.play_tone(1440, 0.5)  # Play "reset completed" tone
            bubble.fill = None  # Set bubble center to transparent (ready)

Let's take a walk through the code and look in more detail how each section works within code.py.
The main module, code.py, prepares and operates the Coffee Scale. It consists of the following major sections:

- Import and Set Defaults
- Calibration Ratio
- Instantiate the Sensor and Display
- Display the Background Bitmap Image
- Define and Display the Text Labels and Graphic Elements
- Helpers
- Activate the Sensor and Prepare to Loop
- The Primary Code Loop

**Import and Set Defaults**

This section imports all the needed modules and libraries, including the NAU7802 sensor driver. After importing, the first visible task of this section is to turn the Clue NeoPixel to the color yellow to indicate that the Coffee Scale is initializing. During operation, the indicator LED will glow green when operating normally or red when zeroing the scale.

Next, the scale defaults are specified. These are constants with names that are capitalized to help identify them as constants. These are values that you can change to alter the operation of the scale.

- **MAX_GR** is the full-scale value of the scale in grams. The tick mark values adjacent to the graduated scale graphic are automatically derived from **MAX_GR**, which can be any positive integer value. Since the graduated scale graphic is divided into tenths of full-scale with five tick marks, it's best to choose an integer **MAX_GR** value that will display nicely such as 1000, 500, 250, 100, 50, 25, 10, or 5.

- **DEFAULT_GAIN** is the gain setting of the NAU7802's internal ADC pre-amplifier. Normally, this is set to the highest value of 128.

- **SAMPLE_AVG** specifies the number of measurements that are averaged when the load cell is measured. The measurement value will be more stable with a high **SAMPLE_AVG** value. The display will update more slowly with a high value; it's a trade-off between stability and speed.

- **SCALE_NAME_1** is the text string that appears on the display to the left of the graduated scale. This should be no longer than 6 characters. Upper case letters will look the best.
**SCALE NAME 2** is the text string that appears on the display to the right of the graduated scale. This should be no longer than 6 characters. Upper case letters will look the best on the display.

```python
# import clue_scale_calibrator  # Uncomment to run calibrator method
import time
import board
from simpleio import map_range
from adafruit_clue import clue
from adafruit_display_shapes.circle import Circle
from adafruit_display_text.label import Label
from adafruit_bitmap_font import bitmap_font
import displayio
from cedargrove_nau7802 import NAU7802
clue.pixel.brightness = 0.2  # Set NeoPixel brightness
clue.pixel[0] = clue.YELLOW  # Set status indicator to yellow (initializing)
# Set Scale Defaults
MAX_GR = 100  # Maximum (full-scale) display range in grams
DEFAULT_GAIN = 128  # Default gain for internal PGA
SAMPLE_AVG = 5  # Number of sample values to average
SCALE_NAME_1 = "COFFEE"  # 6 characters maximum
SCALE_NAME_2 = "SCALE"  # 6 characters maximum
```

### Calibration Ratio

**CALIB_RATIO** is the factor that is used to convert the NAU7802's raw measurement value into grams. This ratio is updated after running the calibrator method. See the guide section, Calibrate the Load Cell for instructions. You should only have to measure and record the load cell calibration ratio once.

```
"""Enter the calibration ratio for the individual load cell in-use. The ratio is composed of the reference weight in grams divided by the raw reading. For example, a raw reading of 215300 for a 100 gram weight results in a calibration ratio of 100 / 215300. Use the clue_scale_single_calibrate method to obtain the raw value.
FYI: A US dime coin weighs 2.268 grams or 0.079 ounces."""
CALIB_RATIO = 100 / 215300  # load cell serial#4540-02
```

### Instantiate the Sensor and Display

The NAU7802 Stemma board is connected to the Clue's I2C bus and lives at address 42 (hexadecimal 2A). The NAU7802 24-bit ADC (analog to digital converter) chip is a dual-channel device, but only the first channel is available on the Stemma board; **active_channels** is set to 1.

Next, the Clue's integrated display is instantiated along with the primary displayio graphics group layer, **scale_group** which will contain the background bitmap image and other group layers.
The three font objects for display titles, labels, and measurements are listed as constants that will be used when display labels are defined.

```python
# Instantiate the Sensor and Display
nau7802 = NAU7802(board.I2C(), address=0x2A, active_channels=1)
display = board.DISPLAY
scale_group = displayio.Group()
FONT_0 = bitmap_font.load_font("/fonts/Helvetica-Bold-24.bdf")
FONT_1 = bitmap_font.load_font("/fonts/OpenSans-16.bdf")
FONT_2 = bitmap_font.load_font("/fonts/OpenSans-9.bdf")
```

Display the Background Bitmap Image

The display background containing the graduated scale is loaded from the Clue root directory and is appended as the first item in the primary displayio group layer. All other graphics objects such as labels and the indicator bubble will be put on layers above the background.

```python
# Display the Background Bitmap Image
bkg = displayio.OnDiskBitmap("/clue_scale_bkg.bmp")
_background = displayio.TileGrid(bkg, pixel_shader=bkg.pixel_shader, x=0, y=0)
scale_group.append(_background)
```
Define and Display the Text Labels and Graphic Elements

This code section defines the graphic elements that will be placed on the display in front of the background graphic. First, the scale name text and zeroing button graphics are appended to the primary displayio group.

The subsection that starts with `for i in range..` is the code that steps through the graduated scale’s tick marks, creating a value label calculated using the `MAX_GR` constant and appending them to the displayio group. Measurement values and units labels added next.

Lastly, the displayio `indicator_group` and it's floating indicator bubble is created and added to the primary displayio group. The `indicator_group` becomes the front-most graphics layer of the display.

The indicator bubble is a yellow circle that travels up and down the graduated scale, pointing to the measured value. The center of the circle will normally be transparent, but will appear yellow or red depending on the scale's current status; yellow when initializing, red when zeroing.

```python
# Define and Display the Text Labels and Graphic Elements
# Place the project name on either side of the graduated scale
scale_name_1 = Label(FONT_1, text=SCALE_NAME_1, color=clue.CYAN)
scale_name_1.anchor_point = (0.5, 0.5)
scale_name_1.anchored_position = (40, 96)
scale_group.append(scale_name_1)

scale_name_2 = Label(FONT_1, text=SCALE_NAME_2, color=clue.CYAN)
scale_name_2.anchor_point = (0.5, 0.5)
scale_name_2.anchored_position = (199, 96)
```
scale_group.append(scale_name_2)

# Define the zeroing button graphic
zero_button_circle = Circle(14, 152, 14, fill=None, outline=clue.RED, stroke=2)
scale_group.append(zero_button_circle)

zero_button_label = Label(FONT_1, text="Z", color=clue.RED)
zero_button_label.x = 8
zero_button_label.y = 150
scale_group.append(zero_button_label)

# Place tickmark labels next to the graduated scale
for i in range(-1, 6):
    tick_value = Label(FONT_2, text=str((MAX_GR) // 5 * i), color=clue.CYAN)
    if i == -1:
        tick_value.anchor_point = (1.0, 1.1)
    elif i == 5:
        tick_value.anchor_point = (1.0, 0.0)
    else:
        tick_value.anchor_point = (1.0, 0.5)
    tick_value.anchored_position = (99, 201 - (i * 40))
scale_group.append(tick_value)

# Place the grams and ounces labels and values near the bottom of the display
grams_label = Label(FONT_0, text="grams", color=clue.BLUE)
grams_label.anchor_point = (1.0, 0)
grams_label.anchored_position = (80, 216)
scale_group.append(grams_label)

ounces_label = Label(FONT_0, text="ounces", color=clue.BLUE)
ounces_label.anchor_point = (1.0, 0)
ounces_label.anchored_position = (230, 216)
scale_group.append(ounces_label)

grams_value = Label(FONT_0, text="0.0", color=clue.WHITE)
grams_value.anchor_point = (1.0, 0.5)
grams_value.anchored_position = (80, 200)
scale_group.append(grams_value)

ounces_value = Label(FONT_0, text="0.00", color=clue.WHITE)
ounces_value.anchor_point = (1.0, 0.5)
ounces_value.anchored_position = (230, 200)
scale_group.append(ounces_value)

# Define the moveable indicator bubble
indicator_group = displayio.Group()
bubble = Circle(120, 200, 10, fill=clue.YELLOW, outline=clue.YELLOW, stroke=3)
indicator_group.append(bubble)
scale_group.append(indicator_group)
display.show(scale_group)

Helpers

These two helper functions work with the NAU7802 driver class to zero the scale and to read the load cell's current raw value.

The `zero_channel()` helper sets up and zeros the NAU7802's internal amplifiers and ADC (analog to digital converter) to prepare it to receive signals. This function is used when the scale is first powered-up as well as when manually zeroed by pressing the Clue's button A.
The load cell's raw measurement value is obtained by the `read()` helper. This helper accepts an integer parameter that specifies the number of samples to be averaged each time the helper is called, defaulting to 1 sample (no averaging). The helper returns the averaged raw measurement value. Since the NAU7802's internal ADC has 24-bit accuracy, the raw value can range from about -8.3M to +8.3M.

```python
# Helpers
def zero_channel():
    """Prepare internal amplifier settings and zero the current channel. Use after power-up, a new channel is selected, or to adjust for measurement drift. Can be used to zero the scale with a tare weight. The nau7802.calibrate function used here does not calibrate the load cell, but sets the NAU7802 internals to prepare for measuring input signals."""
    nau7802.calibrate("INTERNAL")
    nau7802.calibrate("OFFSET")

def read(samples=1):
    """Read and average consecutive raw samples; return averaged value."""
    sample_sum = 0
    sample_count = samples
    while sample_count > 0:
        if nau7802.available():
            sample_sum = sample_sum + nau7802.read()
            sample_count -= 1
    return int(sample_sum / samples)
```

### Activate the Sensor and Prepare to Loop

This is where the NAU7802 ADC is enabled and calibrated for use. Before calibrating and zeroing, the internal sensor amplifier's gain is set to the default value. Once completed, the Clue will chirp some welcoming notes.

```python
# Activate the Sensor
# Enable the internal analog circuitry, set gain, and calibrate/zero
nau7802.enable(True)
nau7802.gain = DEFAULT_GAIN
zero_channel()

# Play "welcome" tones
clue.play_tone(1660, 0.15)
clue.play_tone(1440, 0.15)
```

### The Primary Code Loop

The primary loop is the operational process of the scale. The loop indicates the scale's status on the Clue NeoPixel; green when the scale is operating, red when it's busy zeroing.
After setting the NeoPixel to green, the load cell raw value is measured and converted to grams and ounces. The converted values are formatted and placed into the corresponding display labels and is printed in the REPL.

The grams measurement value is used to position the on-screen indicator bubble along the graduated scale graphic using the `map_range()` function. Also, if the grams measurement value falls outside of the minimum or maximum range, the bubble is "parked" at the extreme position and its interior color is changed from transparent to red.

Finally, the Clue A button is watched to see if it has been pressed. When pressed, the Clue will play a sound and will begin zeroing the scale. During the zeroing process, the NeoPixel and the center of the bubble are set to a red color. When zeroing is completed, the code will wait until the button is released before playing a completion sound, setting the bubble interior to transparent, and returning to normal operation.

```python
# The Primary Code Loop
# Read sensor, move bubble, and display values
while True:
    clue.pixel[0] = clue.GREEN  # Set status indicator to green (ready)

    # Read the raw scale value and scale for grams and ounces
    value = read(SAMPLE_AVG)
    mass_grams = round(value * CALIB_RATIO, 1)
    mass_ounces = round(mass_grams * 0.03527, 2)
    grams_value.text = f"{mass_grams:5.1f}"
    ounces_value.text = f"{mass_ounces:5.2f}"
    print(f" {mass_grams:5.1f} grams   {mass_ounces:5.2f} ounces")

    # Reposition the indicator bubble based on grams value
    min_gr = (MAX_GR // 5) * -1  # Minimum display value
    bubble.y = int(map_range(mass_grams, min_gr, MAX_GR, 240, 0)) - 10
    if mass_grams > MAX_GR or mass_grams < min_gr:
        bubble.fill = clue.RED
    else:
        bubble.fill = None

    # Check to see if the zeroing button is pressed
    if clue.button_a:
        # Zero the sensor
        clue.pixel[0] = clue.RED  # Set status indicator to red (stopped)
        bubble.fill = clue.RED  # Set bubble center to red (stopped)
        clue.play_tone(1660, 0.3)  # Play "button pressed" tone
        zero_channel()

        while clue.button_a:
            # Wait until the button is released
            time.sleep(0.1)

            clue.play_tone(1440, 0.5)  # Play "reset completed" tone
            bubble.fill = None  # Set bubble center to transparent (ready)
```

©Adafruit Industries
Calibrate the Load Cell

Calibrator

Before putting the scale to use at your coffee-making station, use a known weight to record the unique resistance characteristics of the load cell sensor. This calibration should only need to be performed once.

The calibration process reads and averages a handful of raw measurement samples then prints the averaged value in the serial output window. The printed raw measurement value that corresponds to a reference weight will be used within the scale's CircuitPython code to calculate the grams and ounces displayed on the scale's screen.

Remember to mount the load cell so that the fixed end is securely fastened and the weighing end is free to flex with no restrictions.

1. To begin the calibration process, remove all weights from the load cell.

2. The next step is to edit code.py to activate the calibrator method. Using Mu or your favorite text editor (Atom is shown in the example), open the code.py file stored in the root directory of the Clue board and also open a serial output window.
3. Remove the left-most comment hashmark and the following space character from line 10 of the file.

4. After making the change to line 10, save the file. The calibration method will automatically run when the code.py file is saved.

A notice that the calibrator is ready will print to the serial output window.
5. Place a reference weight on the measurement end of the load cell. Once a few RAW VALUE measurements are taken, choose one that looks typically like a median value. In this example, the 100 gram reference weight is producing a raw measurement value of approximately 215300.

If you need to re-zero the load cell to get a fresh measurement, remove any weights then press and hold the Clue A button until you hear a confirming beep. Release the button and the calibrator will start the internal zeroing process. You'll hear a confirming tone when zeroing is complete.

After zeroing the NAU780, the raw value measurement will likely hover within +/- 50 counts of zero. That's expected and within the accuracy expectations for the device.

6. It's time to edit the code.py file in your editor once again to update the CALIB_RATIO constant on line 38. The ratio is the reference weight in grams divided by the raw value you noted during calibration. In this example, it's \( \frac{100}{215300} \).

Don't save the code.py file just yet. There's one more step to go.

7. Place a comment hashmark and a space at the beginning of line 10 to "comment out" the line so that the calibrator method won't be imported the next time code.py executes.
8. Save the file and close the editor. The code.py program will automatically begin. The Clue Coffee Scale is now ready to go!

Calibrator Code Listing

Here's the code listing for the calibrator method. The calibrator is also a good example of a fundamental approach to initiate and read the raw values of a load cell connected to the NAU7802 ADC STEMMA QT board.

```python
# SPDX-FileCopyrightText: 2022 Jan Goolsbey for Adafruit Industries
# SPDX-License-Identifier: MIT
#
# clue_scale_calibrator.py
# 2023-01-13 v1.1.1
#
# Clue Scale Calibrator - Single Channel Version
# Adafruit NAU7802 Stemma breakout example

import time
import board
from adafruit_clue import clue
from cedargrove_nau7802 import NAU7802

clue.pixel.brightness = 0.2  # Set NeoPixel brightness
clue.pixel[0] = clue.YELLOW  # Set status indicator to yellow (initializing)

SAMPLE_AVG = 3  # Number of sample values to average
DEFAULT_GAIN = 128  # Default gain for internal PGA

# Instantiate 24-bit load sensor ADC
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a microcontroller
nau7802 = NAU7802(i2c, address=0x2A, active_channels=1)

def zero_channel():
    """Initiate internal calibration and zero the current channel. Use after power-up, a new channel is selected, or to adjust for measurement drift. Can be used to zero the scale with a tare weight."""
    nau7802.calibrate("INTERNAL")
    nau7802.calibrate("OFFSET")

def read(samples=100):
    # Read and average consecutive raw sample values; return average raw value
    sample_sum = 0
    sample_count = samples
    while sample_count > 0:
        if nau7802.available():
            sample_sum = sample_sum + nau7802.read()
            sample_count -= 1
    return int(sample_sum / samples)

# Activate the NAU780 internal analog circuitry, set gain, and calibrate/zero
nau7802.enable(True)
au7802.gain = DEFAULT_GAIN  # Use default gain
zero_channel()  # Calibrate and zero

print("-----------------------------")
print(" NAU7802 SINGLE CHANNEL CALIBRATOR")
```
print("-----------------------------------")
print("Place a calibration weight on the")
print("load cell.")
print("To re-zero the load cell, remove")
print("any weights then press and hold A.")
print("====================================")
print("")

# Play "welcome" tones
clue.play_tone(1660, 0.15)
clue.play_tone(1440, 0.15)

# Main loop: Read sample and display value
while True:
    clue.pixel[0] = clue.GREEN  # Set status indicator to green
    # Read the raw value; print raw value, gain setting, and % of full-scale
    value = read(SAMPLE_AVG)
    print(f"CHAN_{nau7802.channel:1.0f} RAW VALUE: {value:7.0f}\n")
    print(f"GAIN: x{DEFAULT_GAIN} full-scale: {(value / ((2**23) - 1)) * 100:3.2f}%")
    print("====================================")
    time.sleep(0.1)
    if clue.button_a:
        # Zero and recalibrate the NAU780
        clue.play_tone(1660, 0.3)  # Play "button pressed" tone
        clue.pixel[0] = clue.RED  # Set status indicator to red (stopped)
        zero_channel()
        while clue.button_a:
            # Wait until button is released
            time.sleep(0.1)
            print("RECALIBRATED")
        clue.play_tone(1440, 0.5)  # Play "reset completed" tone