Arduino with Load Cell and HX711 Amplifier (Digital Scale)

In this guide, you'll learn how to create a digital scale with the Arduino using a load cell and the HX711 amplifier. First, you'll learn how to wire the load cell and the HX711 amplifier to the Arduino to build a scale. Then, we'll show you how to calibrate the scale, and a simple example to get the weight of objects. Later, we'll also add a display to show the measurements and a button to tare the scale.

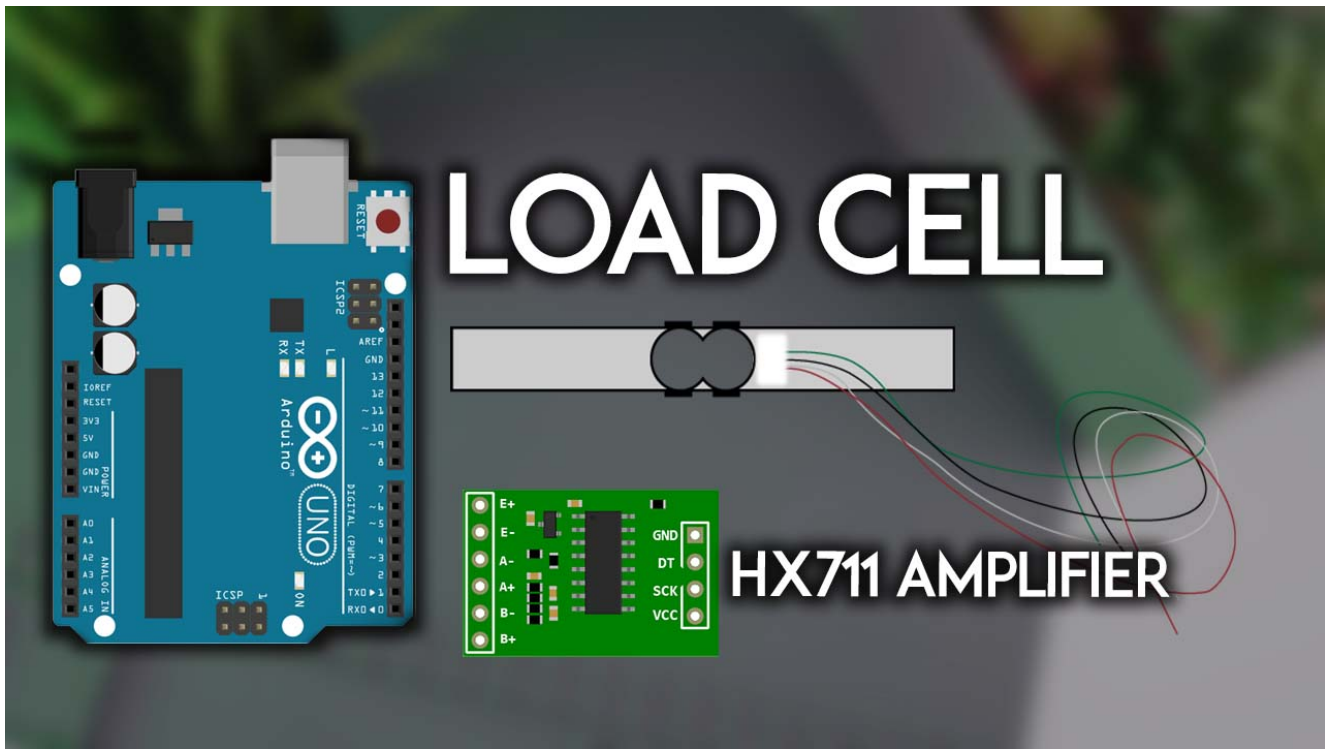
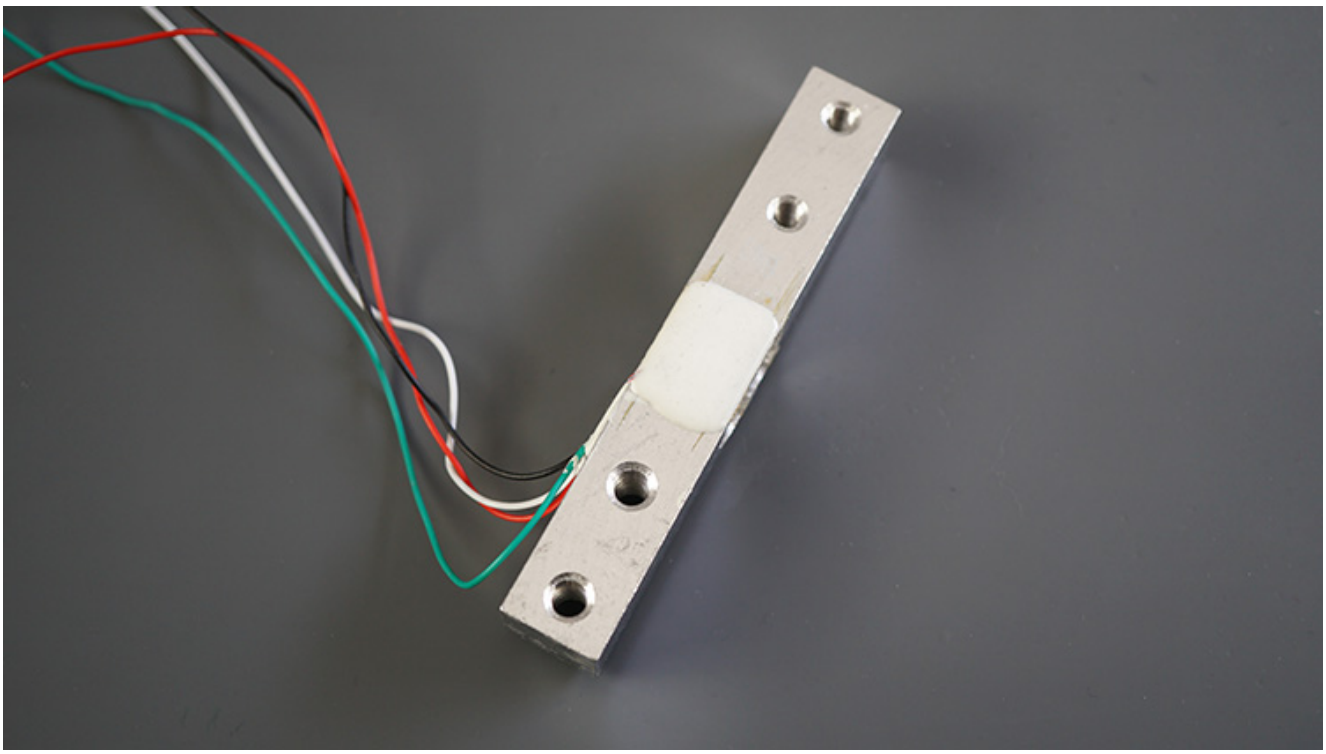


Table of Contents

In this tutorial, we'll cover the following topics:

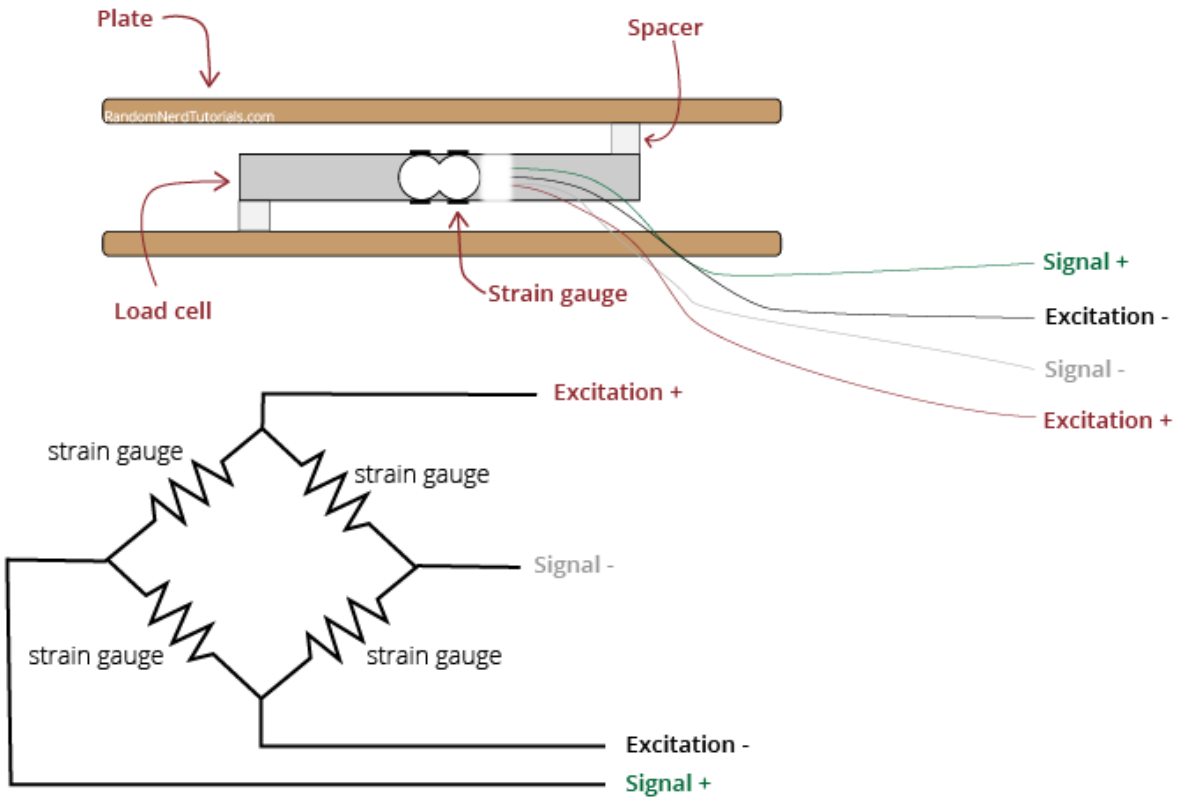
Introducing Load Cells

A load cell converts a force into an electrical signal that can be measured. The electrical signal changes proportionally to the force applied. There are different types of load cells: strain gauges, pneumatic, and hydraulic. In this tutorial, we'll cover strain gauge load cells.



Strain gauge load cells are composed of a metal bar with attached strain gauges (under the white glue in the picture above). A strain gauge is an electrical sensor that measures force or strain on an object. The resistance of the strain gauges varies when an external force is applied to an object, which results in a deformation of the object's shape (in this case, the metal bar). The strain gauge resistance is proportional to the load applied, which allows us to calculate the weight of objects.

Usually, load cells have four strain gauges hooked up in a Wheatstone bridge (as shown below) that allow us to get accurate resistance measurements. For a more detailed explanation of how strain gauges work, read this article.



The wires coming from the load cell usually have the following colors:

- Red : VCC (E+)
- Black : GND (E-)
- White : Output – (A-)
- Green : Output + (A+)

Applications

Strain gauge load cells can be used in a wide variety of applications. For example:

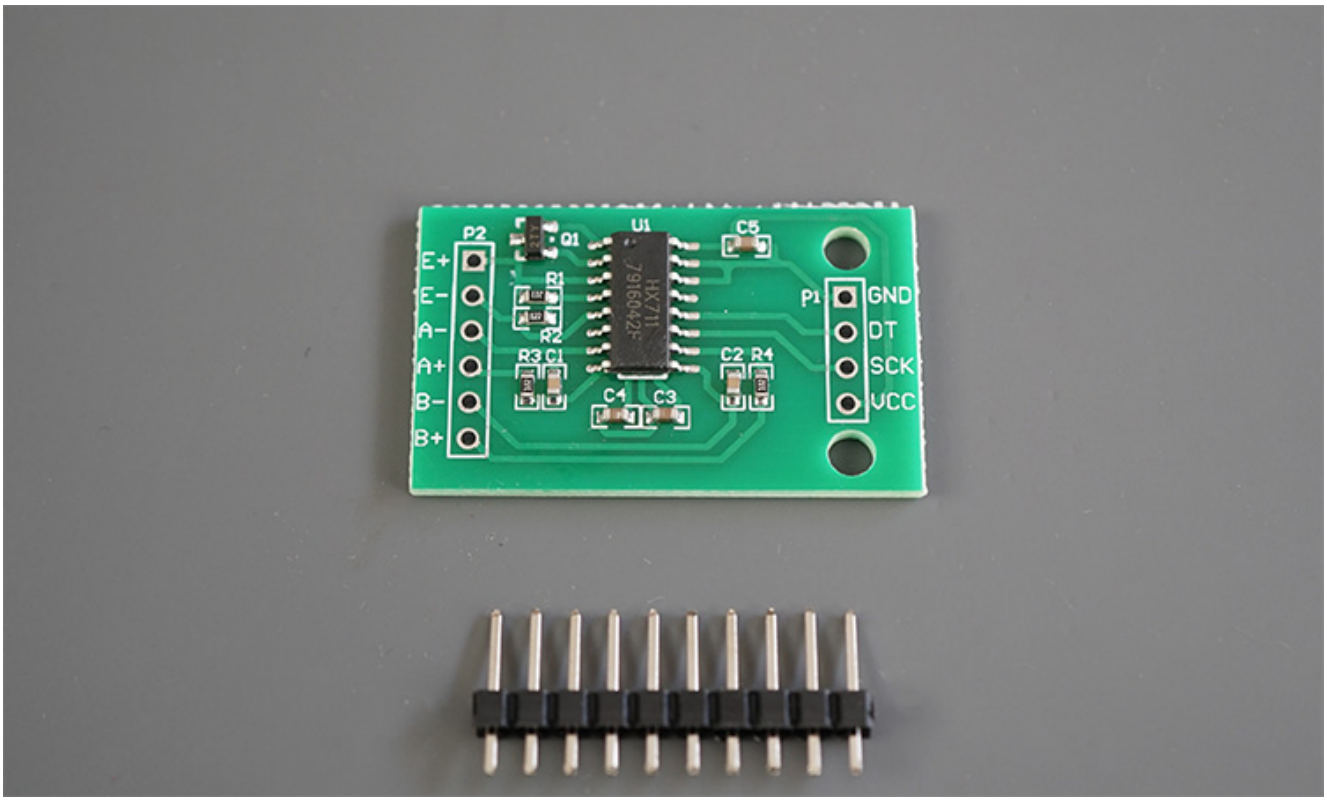
- check if an object's weight changes over time;
- measure the weight of an object;
- detect the presence of an object;

- estimate a container's liquid level;
- etc.

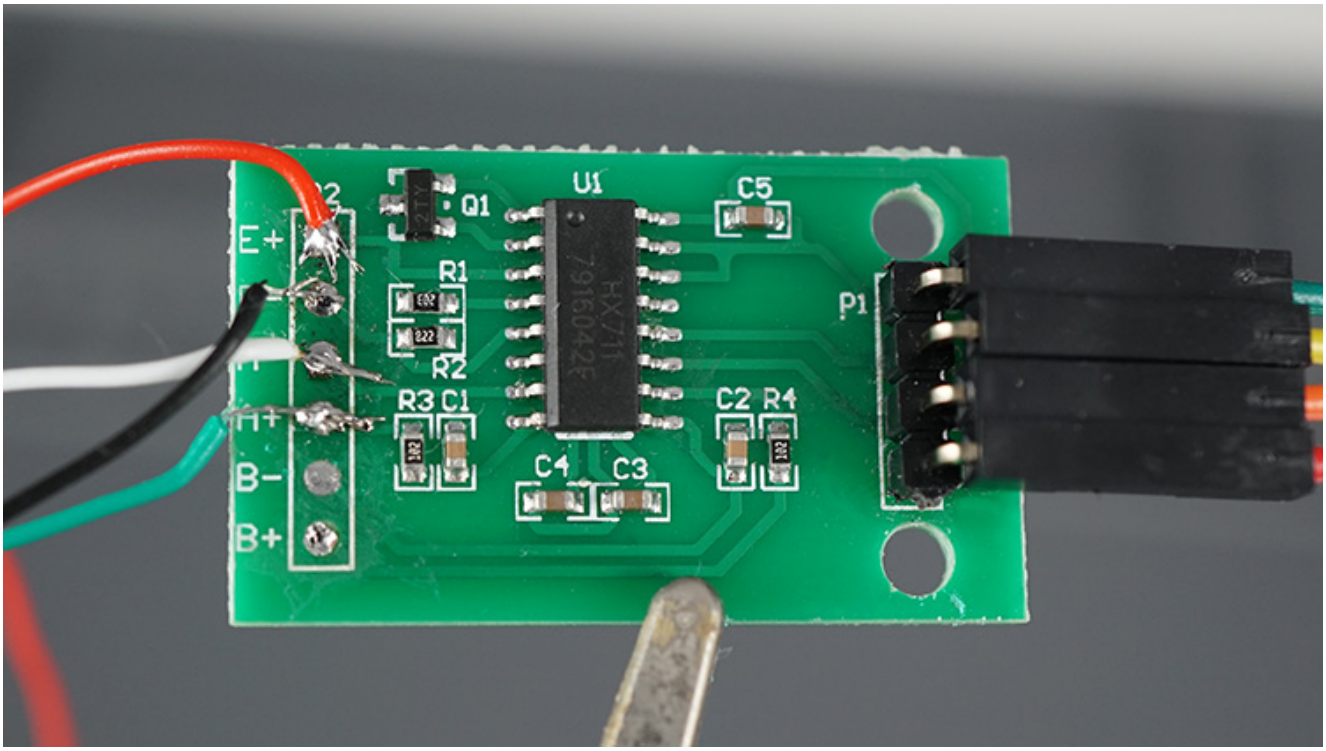
Because the changes in strain when weighting objects are so small, we need an amplifier. The load cell we're using is usually sold together with an HX711 amplifier. So, that's the amplifier we'll use.

HX711 Amplifier

The HX711 amplifier is a breakout board that allows you to easily read load cells to measure weight. You wire the load cell wires on one side, and the microcontroller on the other side. The HX711 communicates with the microcontroller using two-wire interface (Clock and Data).



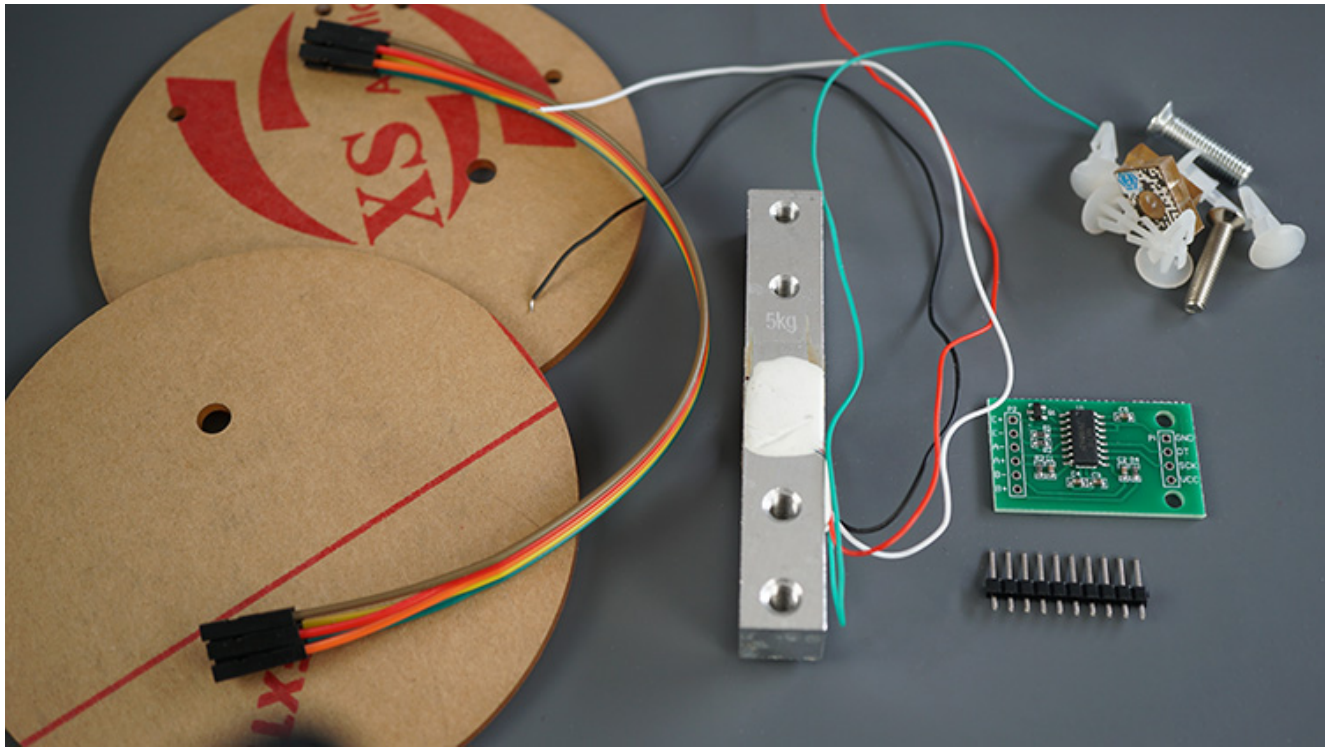
You need to solder header pins on the GND, DT, SCK, and VCC pins to connect to the Arduino. I soldered the load cell wires directly to the E+, E-, A-, and A+ pins. The load cell wires were very thin and fragile, be careful when soldering to not damage the wires.



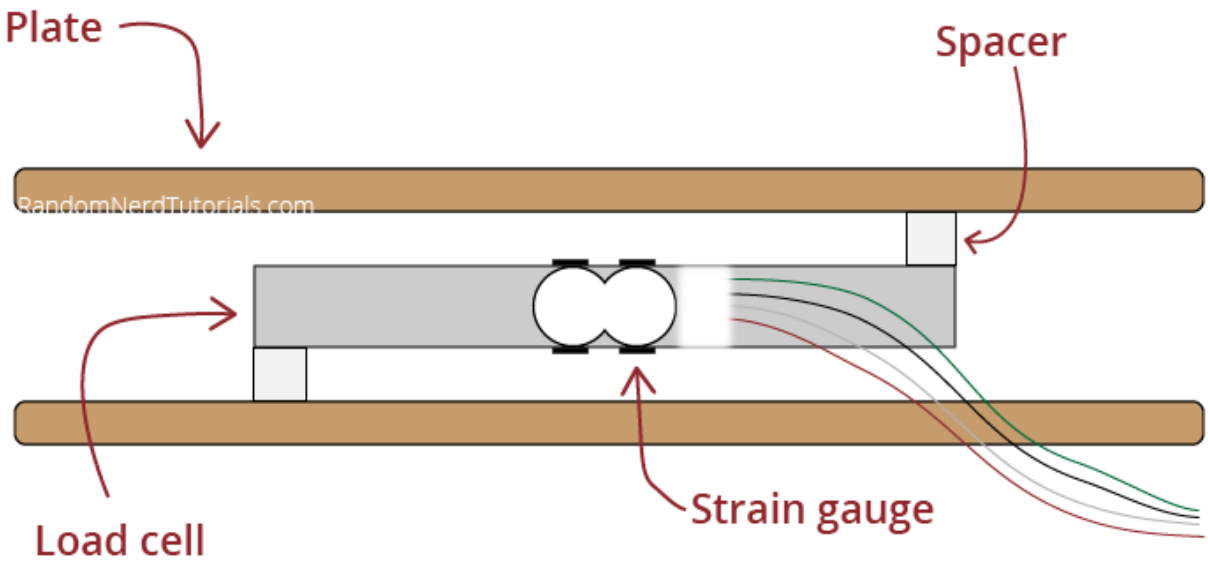
For more information about the HX711 amplifier, you can consult the HX711 datasheet.

Setting Up the Load Cell

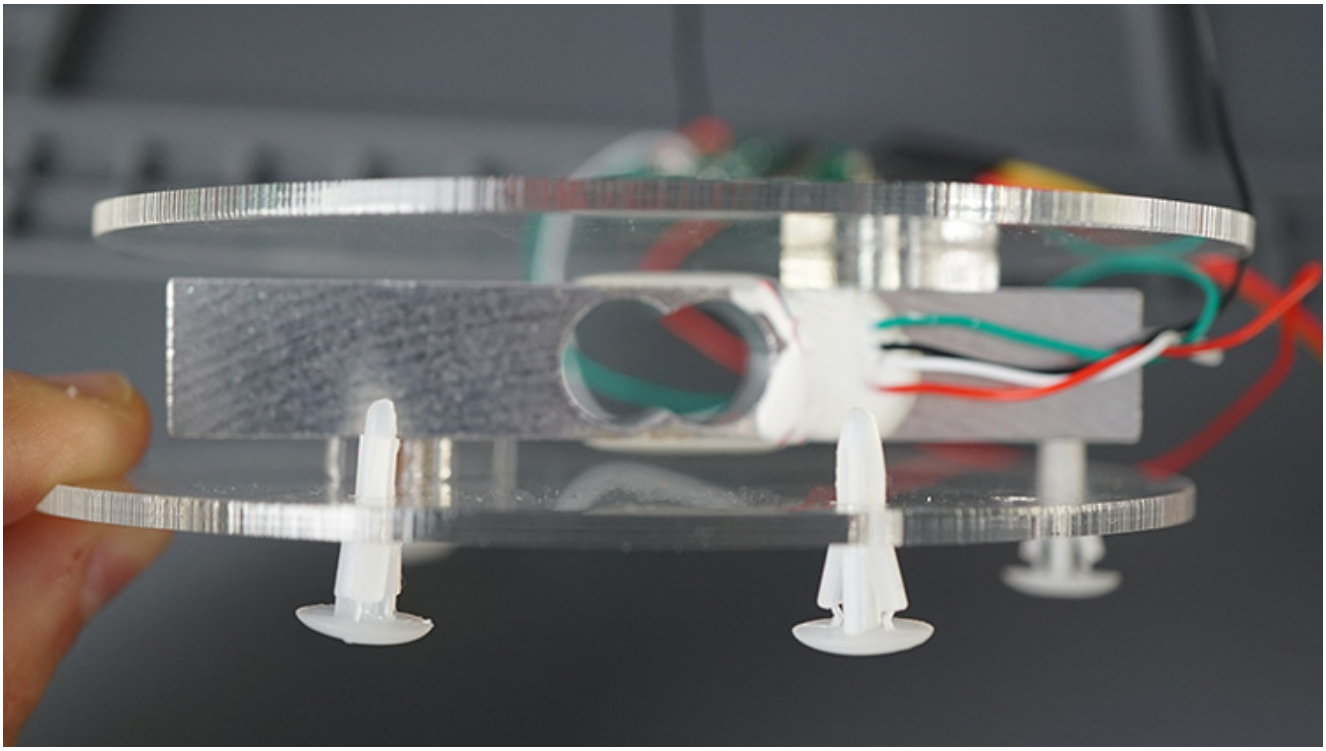
Our load cell kit came with two acrylic plates and some screws to set up the load cell as a scale. You can use wood plates or 3D-print your own plates.



You should attach the plates to the load cell in a way that creates a strain between the opposite ends of the metal bar. The bottom plate holds the load cell, and the upper plate is where you place the objects.

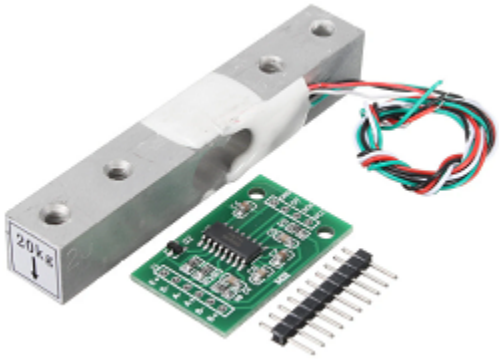


The following figure shows what my load cell with the acrylic plates looks like.



Where to Buy Load Cell with HX711?

You can check the load cell with the HX711 on Maker Advisor to find the best price (with or without acrylic plates included). There are load cells with different measurement ranges. The most common maximum weights are 1kg, 5kg, 10kg, and 20kg.



- Load Cell with HX711 Amplifier

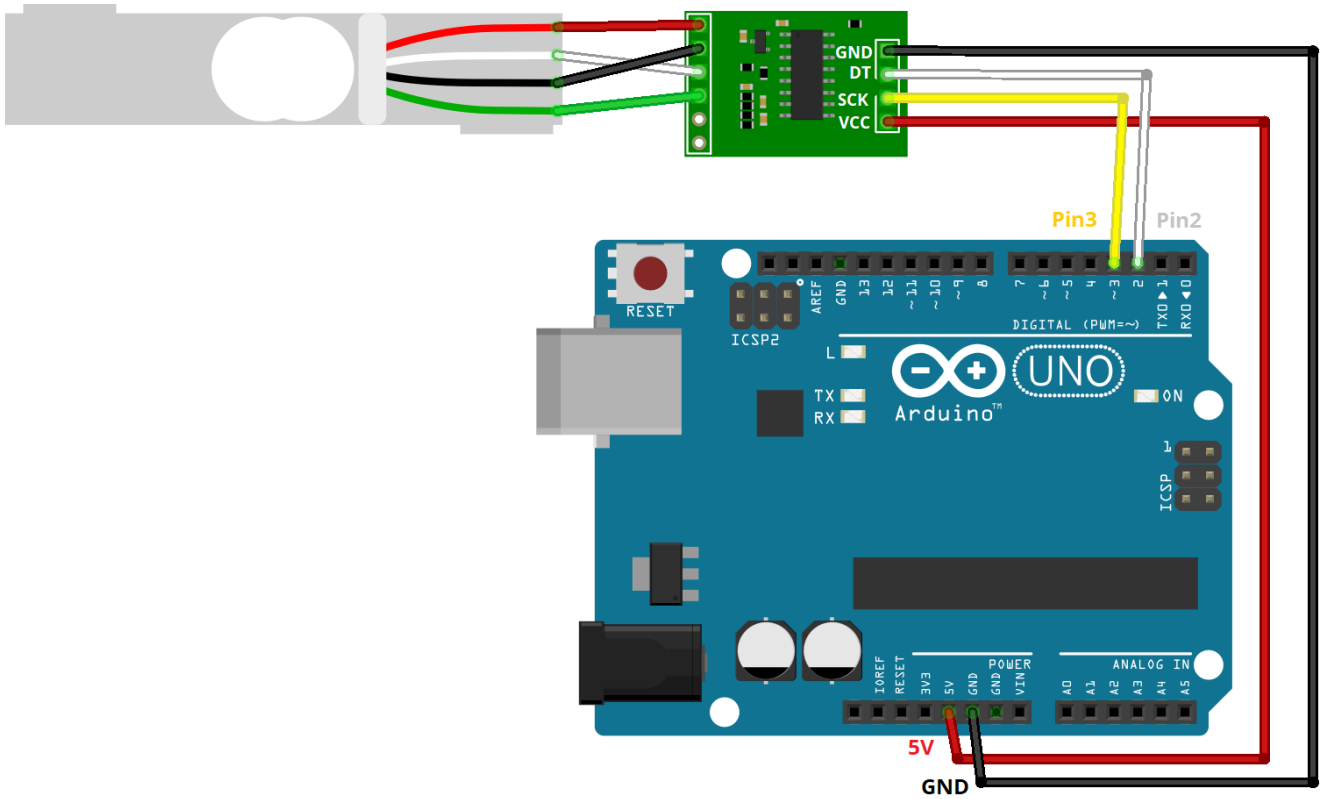
You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Wiring Load Cell and HX711 Amplifier to the Arduino

The HX711 amplifier communicates via two-wire interface. You can connect it to any digital pins of your Arduino board. We're connecting the data pin (DT) to Pin 2 and the clock pin (CLK) to Pin 3 .

Follow the next table or schematic diagram to wire the load cell to the Arduino board.

Load Cell	HX711	HX711	Arduino
Red (E+)	**E+**	**GND**	GND
Black (E-)	**E-**	**DT**	Pin 2
White (A-)	**A-**	**SCK**	Pin 3
Green (A+)	**A+**	**VCC**	5V



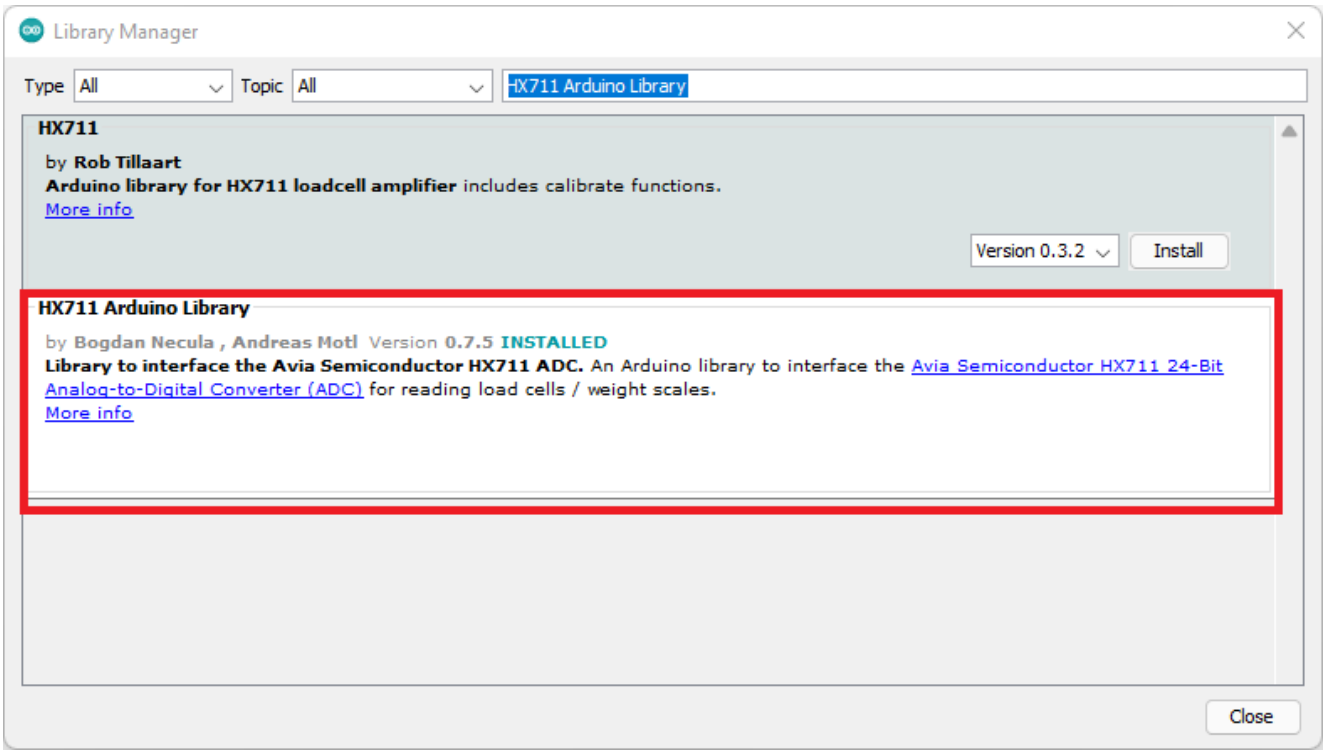
Installing the HX711 Library

There are several different libraries to get measurements from a load cell using the HX711 amplifier. We'll use the HX711 library by bodge. It is compatible with the ESP32, ESP8266, and Arduino.

Arduino IDE

Follow the next instructions to install the library if you're using Arduino IDE.

1. Open Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**.
2. Search for "**HX711 Arduino Library**" and install the library by Bogdan Necula.



Calibrating the Scale (Arduino with Load Cell)

At this time, we assume you have wired the load cell to the HX711 amplifier and the amplifier to the Arduino board. You should also have your scale set up (two plates wired on opposite ends on the load cell), and have installed the HX711 library.

Before getting the weight of objects, you need to calibrate your load cell first by getting the calibration factor. Your calibration factor will be different than mine, so you shouldn't skip this section.

1) Prepare an object with a known weight. I used my kitchen scale and weighed a glass with water (107g).

2) Upload the following code to your Arduino board. We wrote the following code taking into account the instructions to calibrate the load cell provided by the library documentation.

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/ard

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

  The above copyright notice and this permission notice shall be
  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() {
```

View raw code

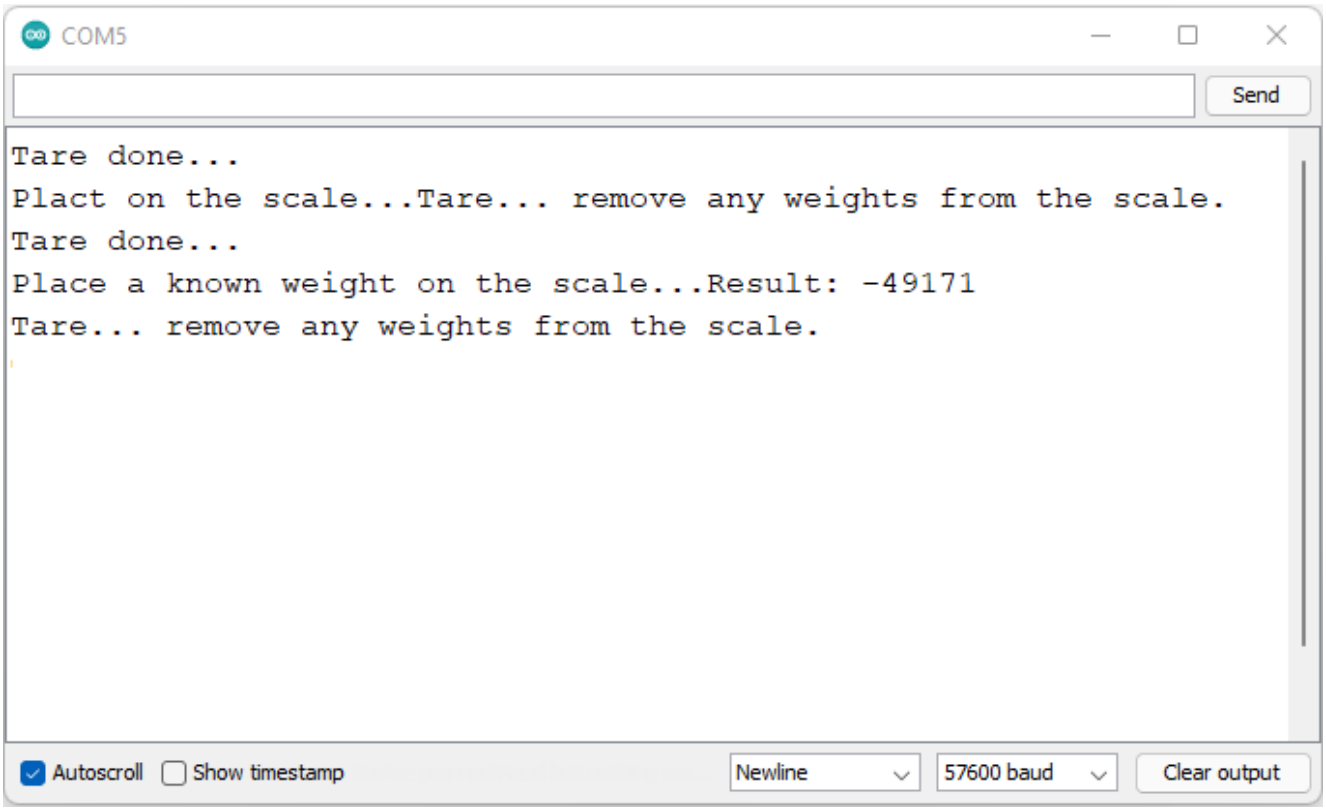
3) After uploading, open the **Serial Monitor at a baud rate of 57600** and then press the Arduino on-board RESET button.

4) Follow the instructions on the Serial Monitor: remove any weights from the scale

(it will tare automatically). Then, place an object with a known weight on the scale and wait until you get a value.

5) Calculate your calibration factor using the formula:

```
calibration factor = (reading)/(known weight)
```



In our case, the reading is -49171. The known weight is 107g, so our calibration factor will be: -49171/107 = -459.542.

```
calibration factor = -49171/107 = -459.542
```

Save your calibration factor because you'll need it later. Yours will be different than ours.

Because the output of the sensor is proportional to the force applied to the load cell, you can calibrate your scale using whatever unit makes sense for you. I used

grams, but you can use pounds, kilograms, or even pieces of cat food (as in this Andreas Spiess video).

Weighting Objects (Arduino with Load Cell)

Now that you know your calibration factor, you can use your load cell to weight objects. Start by weighing objects with a known weight and repeat the calibration process if the values are not accurate.

Copy the following code to your Arduino IDE. Before uploading it to your board, don't forget to insert your calibration factor in line 43/44 of the code. The following code is the example provided by the library that demonstrates the use of most of its functions.

```
/**
 * Complete project details at https://RandomNerdTutorials.com/ar
 *
 * HX711 library for Arduino - example file
 * https://github.com/bogde/HX711
 *
 * MIT License
 * (c) 2018 Bogdan Necula
 *
**/

#include <Arduino.h>
#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);
  Serial.println("HX711 Demo");
  Serial.println("Initializing the scale");

  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
```

View raw code

How the Code Works

Start by including the required library.

```
#include "HX711.h"
```

The following lines define the pins you'll use to connect to the HX711 amplifier. We chose Pin 2 and Pin 3. You can use any other digital pins.

```
const int LOADCELL_DOUT_PIN = 2;
const int LOADCELL_SCK_PIN = 3;
```

Then, create an instance of the HX711 library called `scale` that you'll use later on to get the measurements.

```
HX711 scale;
```

setup()

In the `setup()`, initialize the Serial monitor.

```
Serial.begin(57600);
```

Initialize the load cell by calling the `begin()` method on the `scale` object and passing the digital pins as arguments.

```
scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
```

Then, it calls several methods that you can use to get readings using the library.

- `read()` : gets a raw reading from the sensor
- `read_average(number of readings)` : gets the average of the latest defined number of readings
- `get_value(number of readings)` : gets the average of the last defined number of readings minus the tare weight;
- `get_units(number of readings)` : gets the average of the last defined number of readings minus the tare weight divided by the calibration factor — this will output a reading in your desired units.

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

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

Serial.print("get value: \t\t");
Serial.println(scale.get_value(5));   // print the average of 5 r

Serial.print("get units: \t\t");
Serial.println(scale.get_units(5), 1);  // print the average of 5
// by the SCALE parameter (not set yet)
```

In the following line, don't forget to insert your calibration factor. It uses the `set_scale()` method.

```
scale.set_scale(INSERT YOUR CALIBRATION FACTOR)
```

Then, call the `tare()` method to tare the scale.

```
scale.tare();               // reset the scale to 0
```

After this setup, the scale should be ready to get accurate readings in your desired unit. The example calls the same previous methods so that you can see the difference before and after setting up the scale.

```
Serial.print("read: \t\t");
Serial.println(scale.read());                 // print a raw read

Serial.print("read average: \t\t");
Serial.println(scale.read_average(20));       // print the averag

Serial.print("get value: \t\t");
Serial.println(scale.get_value(5));   // print the average of 5 r

Serial.print("get units: \t\t");
Serial.println(scale.get_units(5), 1);        // print the averag
// by the SCALE parameter set with set_scale
```

loop()

In the `loop()`, the example calls the `get_units()` method in two different ways: to get one single reading (without any parameters) and to get the average of the last 10 readings.

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

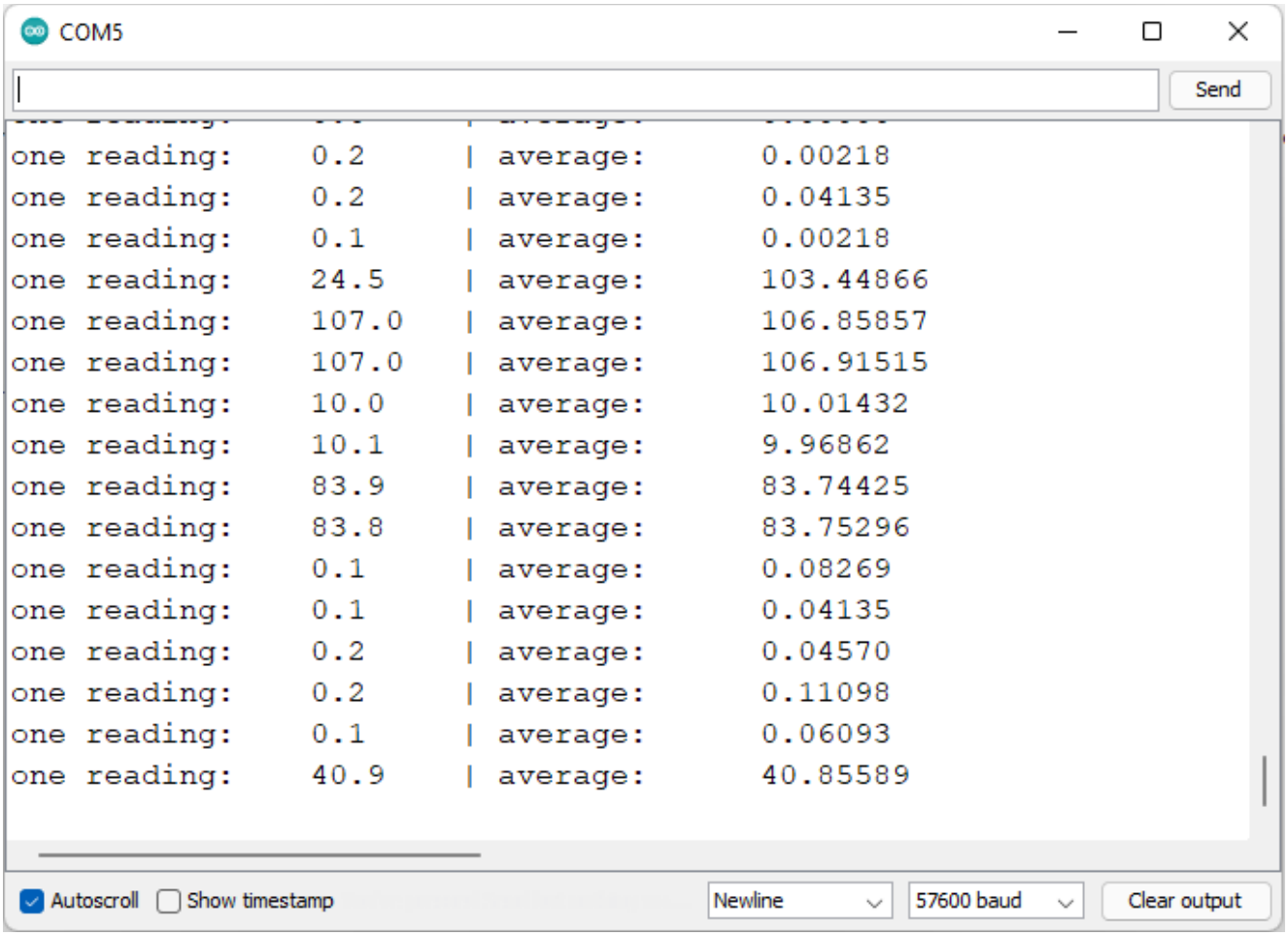
It shuts down the ADC that reads the sensor by using the `power_down()` method. Then, it waits for 5 seconds, powers up the ADC (`power_up()`), and the `loop()` repeats. So, you'll get new readings on the Serial Monitor every 5 seconds.

```
  scale.power_down();  // put the ADC in sleep mode
  delay(5000);
  scale.power_up();
```

Demonstration

Upload the code to your Arduino board. After uploading, open the Serial Monitor at a baud rate of 115200.

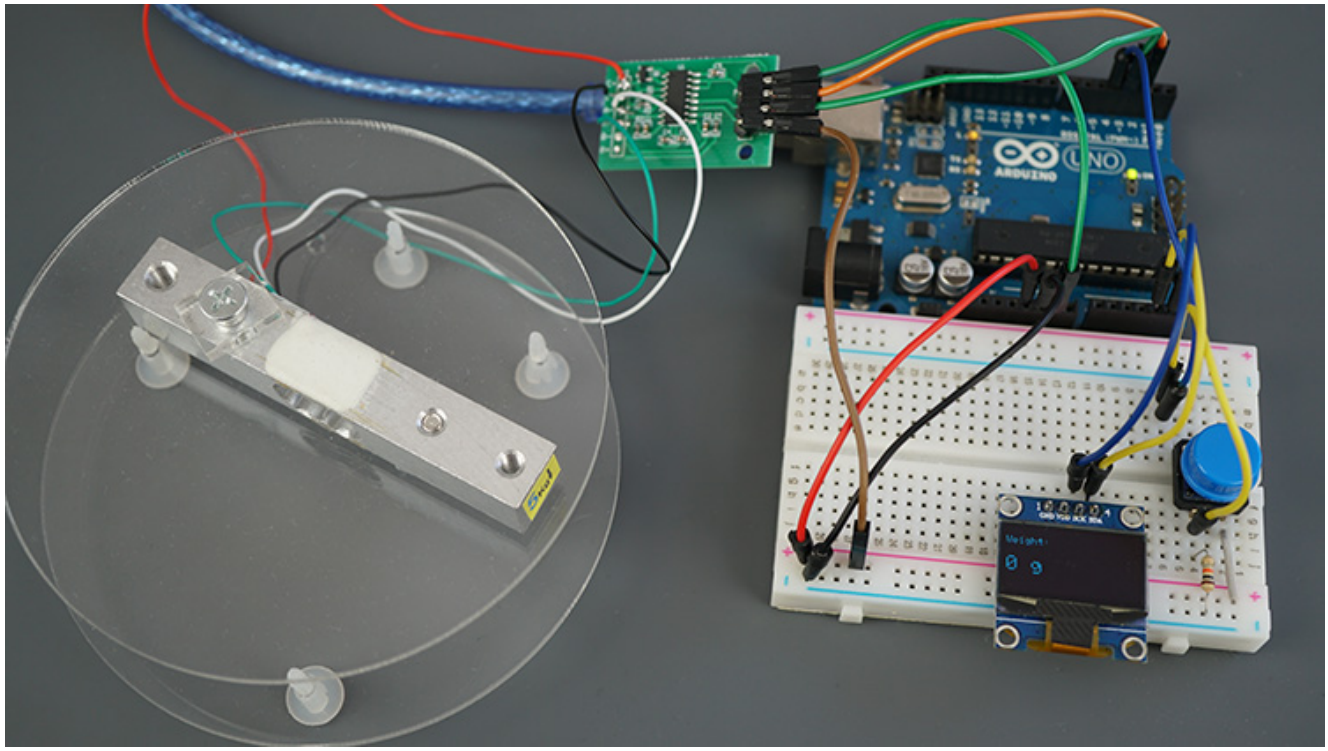
Let the code run a few seconds so that it has time to set up the scale (you'll see the message on the Serial Monitor). Then, place any object on the scale to measure it and you'll get the results on the Serial Monitor.



I experimented with several objects and compared them against the value on my kitchen scale, and the results were the same. So, I can say that my Arduino scale is at least as accurate as my kitchen scale.

Digital Scale with Arduino

In this section, we'll create a simple digital scale with the Arduino. We'll add an OLED display to show the results and a pushbutton to tare the scale.



Parts Required

Here's a list of the parts required for this project:

- Arduino UNO (read Best Arduino starter kits)
- Load Cell with HX711 Amplifier
- I2C SSD1306 OLED Display
- Pushbutton
- 10K Ohm Resistor
- Breadboard
- Jumper Wires

Schematic Diagram

Add an OLED display and a pushbutton to your previous circuit on the following pins:

OLED Display	Arduino
VCC	3.3V or 5V*

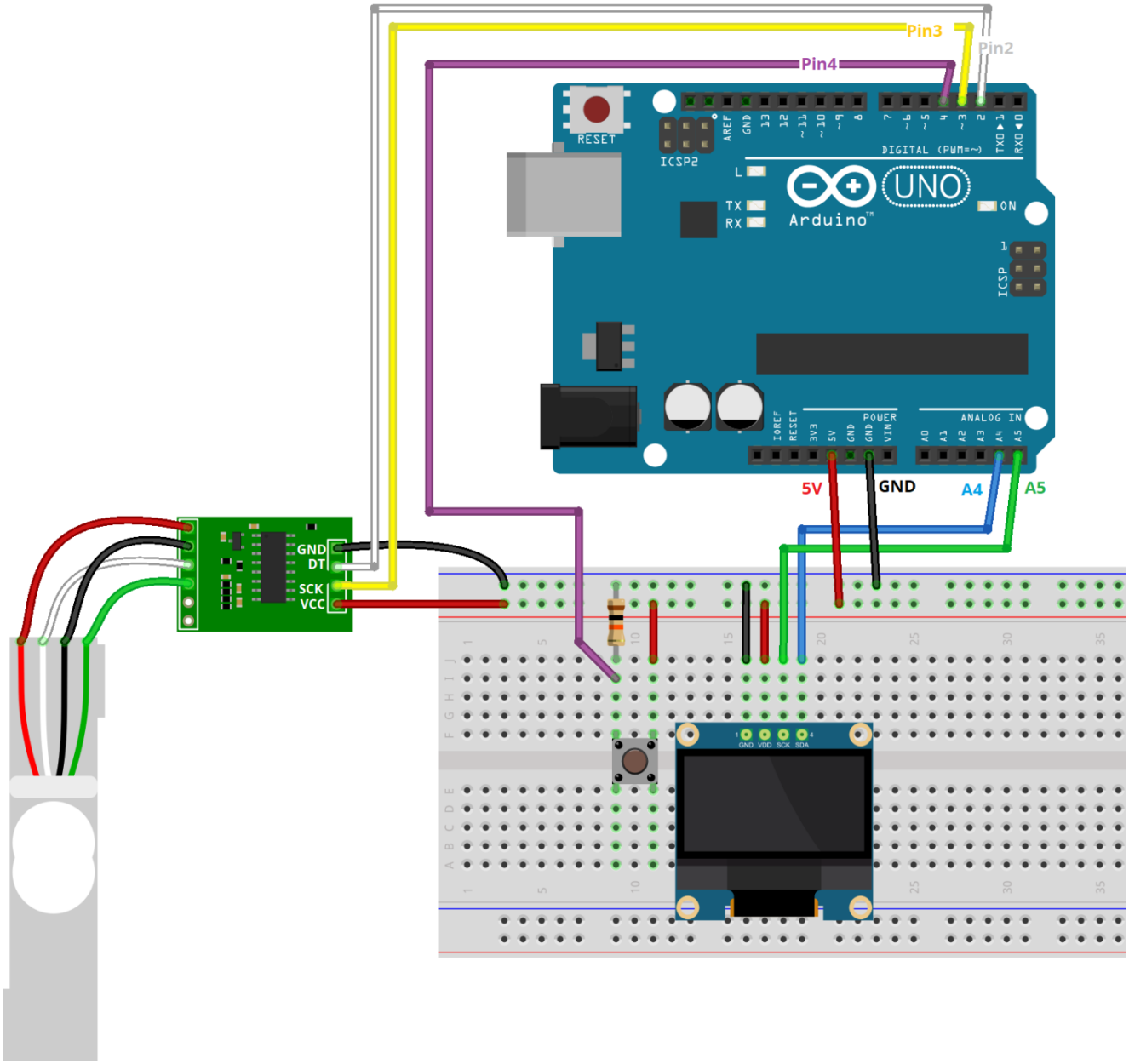
GND	GND
SDA	A4
SCL	A5

*connect to 3.3V or 5V depending on the model.

Not familiar with the OLED display? Read: Guide for I2C OLED Display with Arduino.

Wire the pushbutton via a 10kOhm pull-down resistor to Pin 4 . The other lead of the pushbutton should be connected to 5V. You can use any other Arduino digital pin.

You can follow the next schematic diagram to wire your parts.



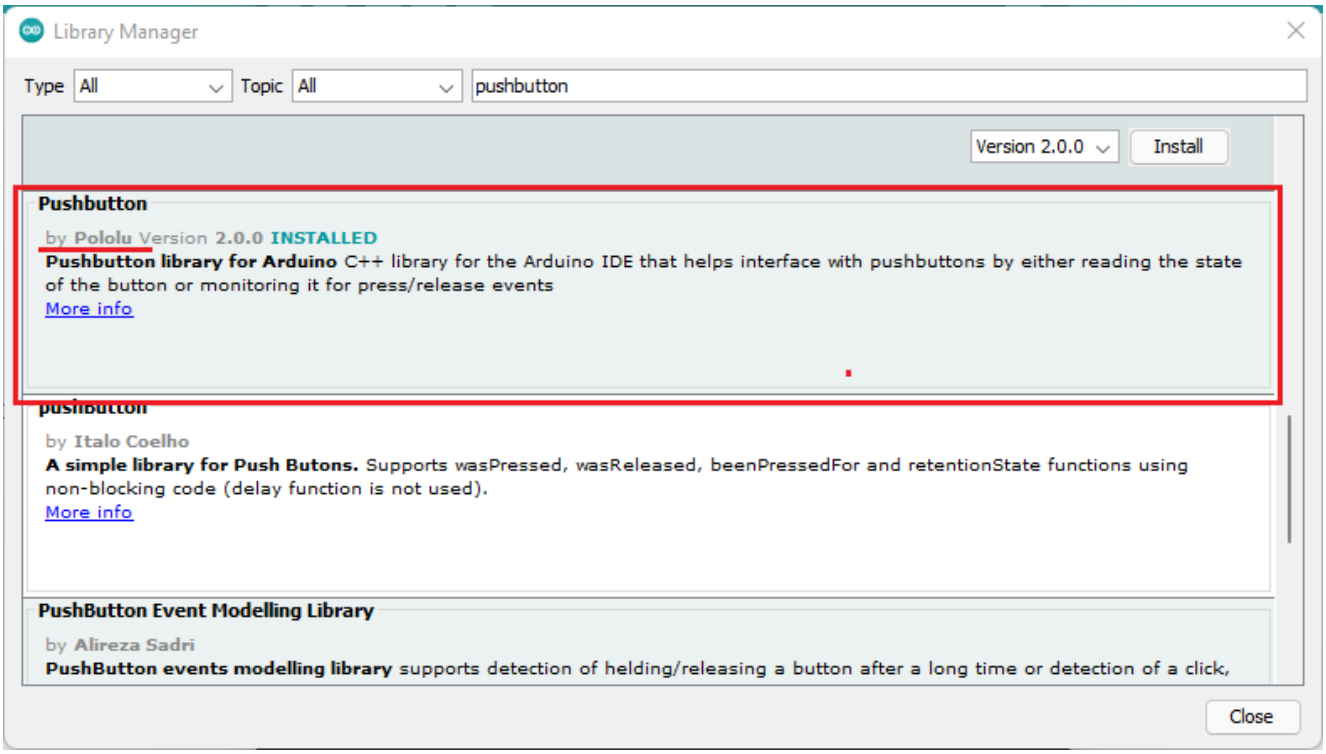
Arduino Digital Scale – Code

For simplicity, we'll handle the pushbutton using a simple library that detects button presses with debouncing (so we don't need to worry about that in our code). To write to the OLED display, we'll use the Adafruit SSD1306 and Adafruit GFX libraries.

Pushbutton Library

There are many libraries with many functionalities to handle pushbuttons. We'll use the pushbutton library by polulu. It is a simple library but comes with everything we need for this project. In your Arduino IDE, go to **Sketch** > **Include**

Library > **Manage Libraries** and search for "**pushbutton**". Install the pushbutton library by polulu.



Alternatively, if you don't want to use the library you can add the debounce code yourself (which is not difficult). For a debounce code example, in the Arduino IDE, you can go to File > Examples > Digital > Debounce.

OLED Libraries

We'll use the following libraries to control the OLED display. Make sure you have these libraries installed:

- Adafruit_SSD1306 library
- Adafruit_GFX library

You can install the libraries using the Arduino Library Manager. Go to **Sketch** > **Include Library** > **Manage Libraries** and search for the library name.

Code

Copy the following code to your Arduino IDE. Before uploading it to the Arduino

board, you need to insert your calibration factor (obtained previously).

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/ard

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

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

// Library HX711 by Bogdan Necula: https://github.com/bogde/HX711
// Library: pushbutton by polulu: https://github.com/pololu/pushb

#include "HX711.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Pushbutton.h>

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

HX711 scale;
int reading;
int lastReading;
```

View raw code

How the Code Works

Start by including the required libraries:

```
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#include "HX711.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Pushbutton.h>
```

Define the pins for the HX711 (load cell)—we're using the same as previous examples:

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

Create an `HX711` instance called `scale`.

```
HX711 scale;
```

The following variables will hold the current weight reading and the last weight reading. We only want to update the OLED display in case there's a new reading, so that's why we need these two variables. Additionally, we don't want to measure decimals of grams which will make the scale too sensitive for our application —that's why these variables are integers. If you need decimals in your measurements, you can define float variables instead.

```
int reading;
int lastReading;
```

Don't forget to replace the next value with your calibration factor. In my case, that line of code looks as follows (my value is negative):

```
#define CALIBRATION_FACTOR -459.542
```

Next, we need to define the OLED width and height:

```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
```

And create an instance of the `Adafruit_SSD1306` library called `display`.

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED
```

Define the GPIO you'll use to read the button and create a `Pushbutton` object called `button` on that pin.

```
#define BUTTON_PIN 4
Pushbutton button(BUTTON_PIN);
```

displayWeight() function

We created a function called `displayWeight()` that accepts as arguments the weight you want to display on the OLED.

```
void displayWeight(int weight){
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  // Display static text
  display.println("Weight:");
  display.display();
```

```
  display.setCursor(0, 30);
  display.setTextSize(2);
  display.print(weight);
  display.print(" ");
  display.print("g");
  display.display();
}
```

Not familiar with the OLED display? Read: Guide for I2C OLED Display with Arduino

setup()

In the `setup()`, initialize the Serial Monitor.

```
Serial.begin(57200);
```

Initialize the OLED display:

```
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println(F("SSD1306 allocation failed"));
  for(;;);
}
delay(2000);
display.clearDisplay();
display.setTextColor(WHITE);
```

And finally, initialize the load cell:

```
Serial.println("Initializing the scale");
scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
```

```
scale.set_scale(CALIBRATION_FACTOR);   // this value is obtained
scale.tare();               // reset the scale to 0
```

loop()

The pushbutton library allows us to wait for an event in case of a pushbutton press or pushbutton release. In this case, we check whether the pushbutton was pushed using the `getSingleDebouncePress()` method and call the `tare()` function if the button was pressed.

```
if (button.getSingleDebouncedPress()){
  Serial.print("tare...");
  scale.tare();
}
```

The HX711 provides a non-blocking method to get readings. It defines a maximum timeout to wait for the hardware to be initialized and doesn't block your code in case the scale gets disconnected or in case of hardware failures.

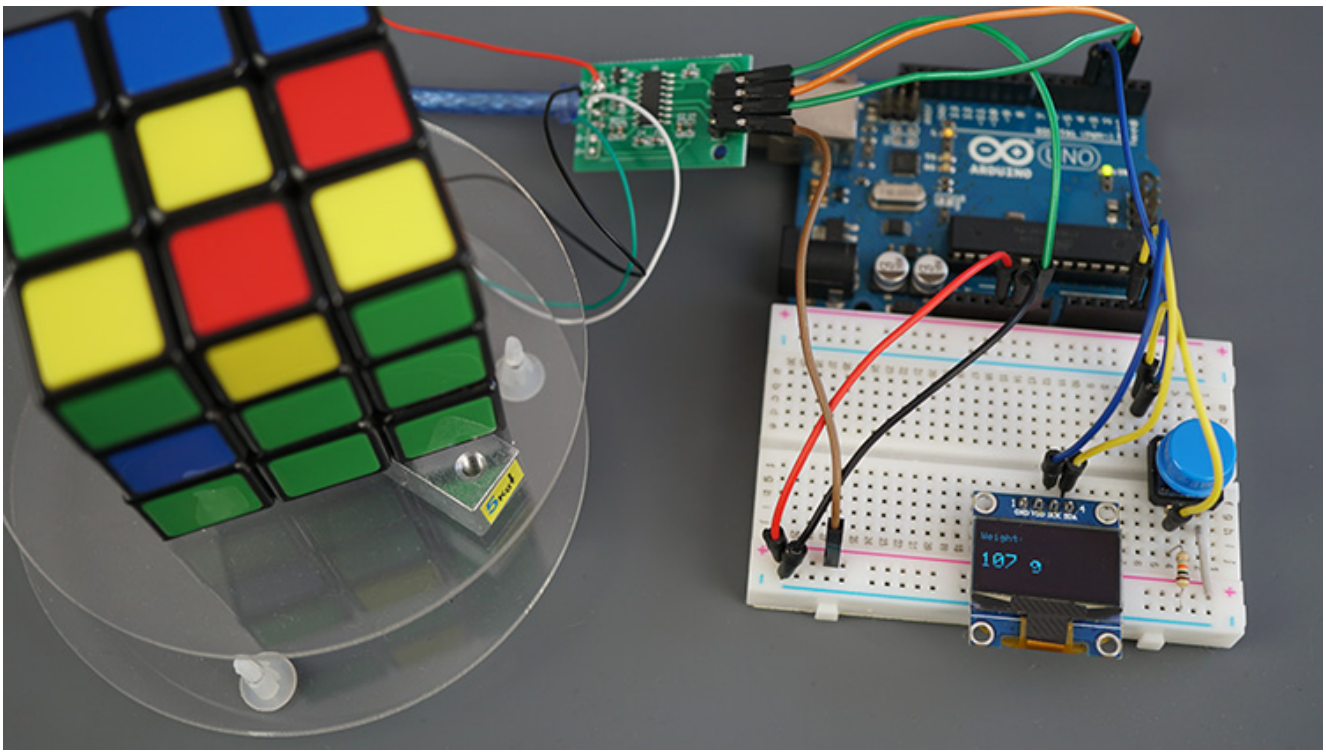
```
if (scale.wait_ready_timeout(200)) {
  reading = round(scale.get_units());
  Serial.print("Weight: ");
  Serial.println(reading);
```

In the `loop()`, we are constantly getting new readings and checking them against the latest reading. If we got a new measurement, we call the `displayWeight()` function to update the OLED display.

```
if (reading != lastReading){
  displayWeight(reading);
}
```

Demonstration

After uploading the code to your board, you can start weighing objects with your load cell. The readings will show up on the OLED display. You can tare the scale by pressing the pushbutton.



Once again, the readings on my Arduino digital scale correspond to the readings on my kitchen scale.

Wrapping Up

In this tutorial, you learned how to interface a strain gauge load cell with the Arduino board using the HX711 amplifier. The output of the load cell is proportional to the force applied. So, you can calibrate it to be used in g, kg, ib, or any other unit that makes sense for your project.

In summary, you learned how to calibrate the scale and how to get the weight of objects. You also learned how to create a simple digital scale with the Arduino using an OLED display to show the measurements and a pushbutton to tare the scale.

We hope you found this tutorial useful to get you started with a load cell. Besides being useful to measure the weight of objects, it can also be useful in many applications like detecting the presence of an object, estimating the level of liquid in a tank, calculating water's evaporation rate, checking if there's food on your pet's bowl, etc.

We have tutorials for other popular sensors that you might find useful:

- Arduino: K-Type Thermocouple with MAX6675 Amplifier (**Temperature Sensor**)
- Arduino with DS18B20: **Temperature Sensor**
- Guide for LM35, LM335, and LM34 **Temperature Sensors** with Arduino
- Arduino with BME680: **Gas, Pressure, Humidity, and Temperature Sensor**
- Arduino with BME280: **Temperature, Humidity, and Pressure Sensor**
- Arduino with DHT11/DHT22: **Temperature, and Humidity Sensor**
- Arduino with BMP388: **Altimeter Sensor**
- Arduino with **Ultrasonic Sensor**
- Arduino Guide for **MPU-6050 Accelerometer and Gyroscope Sensor**
- Arduino with BH1750 **Ambient Light Sensor**
- Arduino with TDS Sensor (**Water Quality**)

Learn more about the Arduino with our resources:

- Arduino Step by step Projects
- Free Arduino projects and tutorials