Weight Measurement System

EE537 Report

Fall 2024

By

Neeraj Kummaragunta

Sai Prasanna Thimmapuram

Abstract: This project implements a weight measurement system using an HX711 load cell amplifier, an ST7735S LCD, and an Arduino microcontroller. The system features functionalities for tare and calibration, real-time weight display, and saving measured weights to an SD card. A Bluetooth module allows wireless updates, while a buzzer provides auditory feedback. It incorporates idle shutdown for energy efficiency, offering a reliable solution for precise weight measurements in various applications.

Section A:

1. ***Project Description:*** This project is a weight measurement system built using an Arduino microcontroller, HX711 load cell amplifier, ST7735S LCD, HC-08 Bluetooth module, and an SD card module. The system allows precise weight measurement, tare, and calibration functionalities. It displays real-time weight on the LCD screen and provides auditory feedback using a buzzer. Users can save measured weights to an SD card for record-keeping. The system also includes a Bluetooth interface for wireless communication. It is energy-efficient and incorporates an idle shutdown feature to save power.

   Key Criteria for successful outcome:
   - Accurate and stable weight measurement
   - Push Button controlled Tare and Calibration
   - Real-time display of weight on the LCD screen.
   - Successful saving of weight data to the SD card.
   - Reliable Bluetooth communication for updates.
   - Idle shutdown after prolonged inactivity to save power.

   Significant Components List:

   - **Arduino Microcontroller**: Core processing and control unit.
   - **HX711 Load Cell Amplifier**: Ensures precise weight measurement.
   - **ST7735S LCD (80x160)**: Displays weight and system messages.
   - **HC-08 Bluetooth Module**: Enables wireless communication.
   - **SD Card Module**: Stores recorded weight data for later use.
   - **Buzzer**: Provides auditory feedback for user actions.

2. ***Constraint Analysis:***

This weight measurement system is influenced by various constraints that must be carefully managed to ensure successful implementation and functionality.

1. Hardware Constraints
   - Processing Power: The Arduino microcontroller has limited computational capabilities, restricting complex operations and requiring efficient coding to optimize resources.
   - Power Supply: The system must operate within the available power budget, especially for components like the buzzer, LCD, and Bluetooth module, which may consume significant energy.
2. Sensor Accuracy and Calibration
   - The HX711 load cell's precision depends on proper calibration and stable environmental conditions. Inconsistent measurements due to noise in the system can affect reliability.
3. Interface Constraints
   - The ST7735S LCD has a limited resolution (80x160), restricting the amount of information that can be displayed. Clear and concise UI design is essential to maximize usability.
   - The Bluetooth module (HC-08) has a limited range, which may affect wireless communication reliability over larger distances.
4. Software and Functional Constraints
   - The system must provide consistent real-time weight readings while ensuring timely saving of data to the SD card. This requires balancing tasks like weight measurement, display updates, and data storage.
   - Interrupt-driven events (e.g., button presses) must be handled efficiently to avoid delays or missed inputs.
5. Environmental Constraints
   - External factors like vibrations or unstable surfaces may affect weight readings, necessitating measures to filter noise and stabilize the scale.

By recognizing these constraints, the system design emphasizes simplicity, energy efficiency, and reliable performance within the technical and environmental boundaries.

3. ***Hardware Interface:***

The hardware interface of the weight measurement system connects multiple components to the Arduino microcontroller for seamless operation. The **HX711 load cell amplifier** is interfaced via digital data and clock pins to measure weight precisely. The **ST7735S LCD** uses SPI communication to display real-time weight and system messages. The **HC-08 Bluetooth module** is connected to serial pins for wireless communication. The **SD card module**, controlled via SPI, stores weight data. A **buzzer** connected to a GPIO pin provides auditory feedback. Push buttons handle user inputs for tare, calibration, and saving data. All components are powered by a 5V supply.

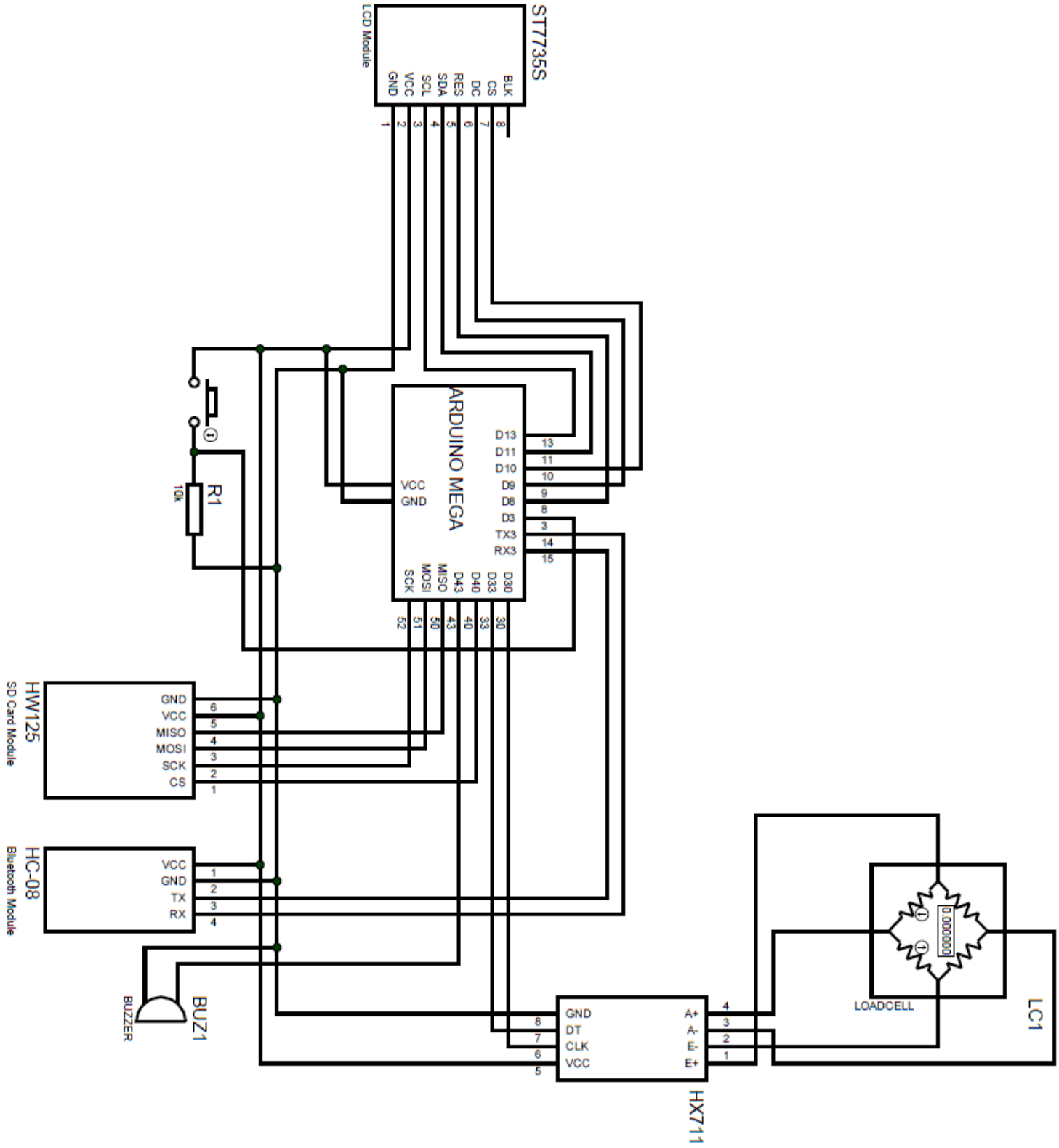

Figure 1. Circuit Schematic of Hardware Interface

4. ***Software Interface:***

The software interface of the weight measurement system is designed to control the system's functionality and interact with the user. The **Arduino IDE** is used to write and upload the code to the microcontroller. The program interfaces with several hardware components, using the **HX711 library** to read weight data from the load cell amplifier and the **UTFT library** for controlling the ST7735S LCD to display weight, calibration, and status messages. The **SoftwareSerial library** is utilized to communicate with the HC-08 Bluetooth module, enabling wireless data transfer. The **SD library** manages data logging, saving the measured weight to an SD card. The software continuously checks button presses (to trigger tare, calibration, or data saving), processes weight readings and updates the LCD display accordingly. Additionally, the buzzer's tone is controlled through digital output to provide auditory feedback, alerting the user about actions like tare or data saving. The software also ensures efficient power usage with idle shutdown.

5. ***Power Constraints and Efficiency:***

The power consumption of the various modules used in this project are given below:

Component	Power Consumption
Arduino Mega	0.25 W
HX711 Load Cell Amplifier	7.5 mW
ST7735S LCD Display (80x160 IPS)	7.5 mW
HC-08 Bluetooth Module	0.06 W
Buzzer	0.1 W
SD Card Module	0.2 W

Table 1. Component Power Consumption

The total Power Consumption for this system comes to approximately 0.7 W when all components are active.

6. ***Cost Constraints:***

The approximate costs of each module are given below:

Component	Cost($)
Arduino Mega 2560	43.56
HX711 Load Cell Amplifier	7.88
ST7735S LCD Display (80x160 IPS)	4.68
HC-08 Bluetooth Module	10.99
Buzzer	0.95
SD Card Module	4.00

Table 2. Component Costs

Total Cost: $72.06

7. ***Component Selection Rationale:***

All parts for this project were received from the instructor, but the reasoning behind most parts was understood.

- The Arduino microcontroller provides a simple, cost-effective platform with sufficient processing power and flexibility for controlling all system components.
- The HX711 load cell amplifier was chosen for its precision and compatibility with the load cell, enabling accurate weight measurements.
- The ST7735S LCD was selected for its compact size (80x160 resolution) and low power consumption, which is ideal for portable use.

Section B:

1. ***Detailed Project Description:***

Significant system components:

Component 1: **Arduino Microcontroller**

- **Model**: Arduino Uno
- **Made by**: Arduino
- **Explanation**: The Arduino Uno is a popular microcontroller board based on the ATmega328P. It serves as the central control unit, processing input from the load cell, LCD, and other components. It also manages tasks like data storage, user input processing, and communication with the Bluetooth module. Its open-source nature and ease of use make it ideal for prototyping and educational projects.

Component 2: **HX711 Load Cell Amplifier**

- **Model**: HX711
- **Made by**: HiLetgo
- **Explanation**: The HX711 is a precision 24-bit analog-to-digital converter (ADC) designed for use with load cells. It amplifies the low voltage signals from the load cell, making them readable by the Arduino. It's widely used in weighing systems due to its high accuracy and compatibility with Arduino, making it essential for precise weight measurement in this project.

Component 3: **ST7735S LCD Display**

- **Model**: ST7735S 80x160
- **Made by**: Adafruit
- **Explanation**: This 0.96-inch LCD screen uses IPS technology and has a resolution of 80x160 pixels. It is used to display weight readings, system status, and messages to the user. Compact size and low power consumption make it perfect for portable applications, while its bright color output enhances readability in different lighting conditions.

Component 4: **HC-08 Bluetooth Module**

- **Model**: HC-08
- **Made by**: HC Technology

- **Explanation**: The HC-08 is a Bluetooth Low Energy (BLE) module, ideal for wireless communication between the weight measurement system and a smartphone or computer. It operates on 3.3V and communicates via UART (serial communication). It is easy to use and integrates well with Arduino, enabling remote monitoring of the weight data.

Component 5: **SD Card Module**

- **Model**: Generic SD Card Module
- **Made by**: Various manufacturers (e.g., DSD Tech, Arduino)
- **Explanation**: The SD card module interfaces with the Arduino to read from and write to an SD card. It is used in this project to store recorded weight measurements, ensuring data persistence for future analysis or record-keeping. It operates over the SPI protocol and provides a simple method to expand the Arduino's storage capacity.

Component 6: **Buzzer**

- **Model**: Passive Buzzer
- **Made by**: Various manufacturers
- **Explanation**: A simple passive buzzer is used to provide auditory feedback to the user, signalling actions like tare completion or data saving. It is connected to a digital I/O pin on the Arduino and activated with a tone, providing non-visual feedback to complement the LCD display.

2. ***Project Development:***

The project was divided into many parts and each part was tested first, followed by integration into the main program, then debugging to make sure components interface perfectly.

Step 1: Weight Measurement

The HX711 Load cell Amplifier and the corresponding load sensors were all soldered appropriately, and then the HX711 Library was used to understand the capabilities of said IC. It was coded basically to get reading of different weights, Once that was accomplished, taring and calibration functions were introduced in the code. Serial input was used to test these functions. Weight output was being displayed on the Arduino IDE Serial Monitor.

Step 2: Button Controlled Tare and Calibration

A push button was added to the project, to detect user input for when they were ready to Tare and Calibrate the system. Code was modified to run on input from push button instead of Serial Communication.

Step 3: LCD Module Integration

The LCD module was tested with a basic graphics test which helped with the understanding of the various commands in its UTFT library. The commands were then integrated into the main code, so as to display the user prompts and weight on the LCD module instead of the Serial Monitor.

Step 4: SD Card module

The SD card module was then tested to check functionality, and once the SD Library was understood, the module was integrated into the system, taking input from a push button to detect user intent to save weight data.

Step 5: Buzzer

A buzzer was added to help the user understand that push button input was detected.

Step 6: Timeout System

To save power, an idle shutdown feature was implemented. If no significant weight change occurred for 2 minutes, the system powers down, providing energy efficiency.

Step 7: Bluetooth Module

The Bluetooth module was lastly integrated into the code, so as to display user prompts and weight data not only on the LCD display but also on a device connected with Bluetooth.

3. ***Conclusion and Future Improvements:***

The smart weight measurement system successfully met the project objectives of weight measurement, data storage, and wireless communication. The HX711 load cell amplifier provided stable weight readings, while the Arduino microcontroller efficiently handled the input from the load cell. The system displayed real-time weight on the ST7735S LCD and allowed users to save weight data to an SD card. The HC-08 Bluetooth module enabled wireless communication, and the buzzer provided essential auditory feedback for user interactions. Power consumption was managed effectively with an idle shutdown feature. The project demonstrated good accuracy, reliable performance, and ease of use.

Further improvements to the smart weight measurement system could include enhancing accuracy by incorporating a more precise load cell with better resolution. Furthermore, integrating a mobile app or cloud-based platform for real-time monitoring and remote data storage would provide more convenience. Lastly, enhancing the user interface with more detailed feedback and error handling would improve usability.

Section C:

Part Name	Quantity	Description	Vendor Link/Name
Arduino Mega 2560	1	The CPU of the system	Amazon
Breadboard	1	Used to connect Components	Amazon
HX711 Amplifier with Load Cells	1	The weight sensors	https://www.amazon.com/Wishiot-Weighing-Half-Bridge-Pressure-Amplifier/dp/B0CWLHXHMD?th=1
ST7735S LCD Display (80x160 IPS)	1	The Display Unit	Amazon
HC-08 Bluetooth Module	1	Used for Communication with Bluetooth Devices	Amazon
Buzzer	1	To alert user that button has been pressed	Amazon
SD Card Module	1	Used for Data Storage	Amazon
Push Button	1	To detect input from User	Amazon
Resistor (10k)	1	Used in the push button input system	Amazon
M-M Wires	30	For making connections between different modules	Amazon

Table 3. Detailed Part List

Name	Link
HX711	https://github.com/bogde/HX711
UTFT	https://cdn.shopify.com/s/files/1/2386/9605/files/TFT80160.zip?2174
SD	https://github.com/arduino-libraries/SD
SoftwareSerial	https://github.com/arduino/ArduinoCore-avr/tree/master/libraries/SoftwareSerial

Table 4. List of Libraries used in code

Program Code:

```
#include "HX711.h" // Include the HX711 library for interfacing with the load cell
#include <UTFT.h> // Include the UTFT library for managing the LCD display
#include <SD.h> // Include the SD library to handle SD card operations
#include <SoftwareSerial.h> // Include the SoftwareSerial library for bluetooth

#define buzzer 43

// Create a SoftwareSerial object for communication with the HC-08 Bluetooth module
SoftwareSerial hc08(15,14); // RX pin, TX pin;

HX711 scale; // Declare an HX711 object to manage the load cell amplifier

// Define pins for the HX711 module
uint8_t dataPin = 33;
uint8_t clockPin = 30;

const int chipSelect = 40; // Define the chip select pin for the SD card module
File myFile;// File object to manage files on the SD card

// Create a UTFT object to control the LCD screen with specific pin configurations
UTFT myGLCD(ST7735S_4L_80160,11,13,10,8,9);

extern uint8_t BigFont[];// Declare an external font array for the LCD display

long count = 0; // General-purpose counter
float scale1; // Variable to store the scale factor of the HX711
uint32_t offset; // Variable to store the tare offset value
int bread = 0; // Variable to store button states

// Function declaration for generating a buzz sound
void buzz();

void setup()
{
 Serial.begin(115200);  // Start Serial communication at 115200 baud for debugging
 Serial.println();      // Print a newline to format the output on Serial Monitor
 hc08.begin(9600);      // Start communication with HC-08 Bluetooth module at 9600 baud

 pinMode(buzzer, OUTPUT); // Set the buzzer pin as an output

 while (!Serial);  // Wait for Serial to initialize

  // Initialize the SD card and check if it's working
 Serial.print("Initializing SD card...");
 hc08.write("Initializing SD card...");
 hc08.write("\n");

 if (!SD.begin(chipSelect)) { // Check if the SD card initialization fails
   Serial.println("initialization failed. Things to check:");
```

```
    Serial.println("1. is a card inserted?");
    Serial.println("2. is your wiring correct?");
    Serial.println("3. did you change the chipSelect pin to match your shield or module?");
    Serial.println("Note: press reset button on the board and reopen this Serial Monitor after fixing your
issue!");
    while (true); // Halt the program if the SD card is not initialized
  }

  Serial.println("initialization done.");
  hc08.write("initialization done.\n");

  pinMode(3, INPUT); // Set button pin (pin 3) as an input

  randomSeed(analogRead(0)); // Seed the random number generator with analog noise
  myGLCD.InitLCD();          // Initialize the LCD display
  myGLCD.setFont(BigFont);   // Set the font for the LCD display
  myGLCD.clrScr();           // Clear the screen

  // Initialize the HX711 scale
  scale.begin(dataPin, clockPin);
  Serial.print("UNITS: ");
  Serial.println(scale.get_units(10)); // Display the initial weight value on Serial Monitor
  myGLCD.print("Scale", LEFT, 1);
  myGLCD.print("Ready", LEFT, 10000);
  delay(3000);

  // Prompt user to empty the scale for taring
  myGLCD.print("Empty" , LEFT, 1);
  myGLCD.print("the scale", LEFT, 10000);
  delay(3000);
  myGLCD.clrScr();
  myGLCD.print("press" , LEFT, 1);
  myGLCD.print("Button", LEFT, 10000);
  myGLCD.print("to tare" , LEFT, 20000);
  Serial.println("\nEmpty the scale, press button to continue");
  hc08.write("Empty the scale, press button to continue\n");

  // Wait for button press to perform taring
  here:
  bread = digitalRead(3); // Read the button state
  if(bread == HIGH){ //if button is pressed
    myGLCD.clrScr();
    scale.tare(20); // Perform taring using 20 readings
    myGLCD.print("Taring...", LEFT, 10000);
    hc08.write("Taring\n");
    buzz(); //generate sound to show button is pressed
    myGLCD.clrScr();
    offset = scale.get_offset(); // Store the tare offset value
    myGLCD.print("Tare", LEFT, 1);
    myGLCD.print("Done", LEFT, 10000);
    hc08.write("Tare Done\n");
```

```
}
else{
  goto here; // Keep checking for button press if not pressed
}
delay(3000);

// Prompt user to place a calibration weight
Serial.print("UNITS: ");
Serial.println(scale.get_units(10));
bread = 0;
myGLCD.clrScr();
myGLCD.print("Put 2.5" , LEFT, 1);
myGLCD.print("lbs Weight" , LEFT, 10000);
myGLCD.print("on the scale" , LEFT, 20000);
delay(3000);
myGLCD.clrScr();
myGLCD.print("press" , LEFT, 1);
myGLCD.print("Button to", LEFT, 10000);
myGLCD.print("callibrate" , LEFT, 20000);
Serial.println("\nPut 2.5lbs Weight on the scale, press button to continue");
hc08.write("Put 2.5lbs Weight on the scale, press button to continue\n");
uint32_t weight = 1134; // Set the calibration weight (2.5 lbs in grams)

// Wait for button press to perform calibration
there:
bread = digitalRead(3);
if(bread == HIGH){
  myGLCD.clrScr();
  scale.calibrate_scale(weight, 20); // Calibrate the scale with the specified weight
  myGLCD.print("Callibrating", LEFT, 1);
  myGLCD.print("...", LEFT, 10000);
  hc08.write("Calibrating...\n");
  buzz();
  scale1 = scale.get_scale(); // Store the scale factor
  myGLCD.clrScr();
  myGLCD.print("Callibration", LEFT, 1);
  myGLCD.print("done", LEFT, 10000);
  hc08.write("Calibration Done\n");
  delay(3000);
}
else{
  goto there; // Keep checking for button press if not pressed
}

// Finalize scale configuration
scale.set_offset(offset); // Apply the stored offset
scale.set_scale(scale1); // Apply the calculated scale factor
myGLCD.clrScr();
Serial.print("UNITS: ");
Serial.println(scale.get_units(10));
```

```
  // Notify user that the scale is calibrated
  myGLCD.print("Scale is " , LEFT, 1);
  myGLCD.print("calibrated," , LEFT, 10000);
  delay(3000);
  myGLCD.clrScr();
  myGLCD.print("press" , LEFT, 1);
  myGLCD.print("Button to", LEFT, 10000);
  myGLCD.print("continue" , LEFT, 20000);
  Serial.println("\nScale is calibrated, press button to continue");
  hc08.write("Scale is calibrated, press button to continue\n");
  bread=0; // Reset the button state

  //wait for button press
  wait:
  bread = digitalRead(3);
  if(bread == HIGH){
    buzz();
  }
  else{
    goto wait;
  }
  myGLCD.clrScr();
}

void loop()
{
  static long lastWeight = 0; // Variable to store the last measured weight
  static unsigned long lastChangeTime = millis(); // Timestamp of the last significant weight change
  long currentWeight; // Variable to store the current weight

  // Initial instruction to the user
  myGLCD.clrScr();
  myGLCD.print("Press button", LEFT, 1);
  myGLCD.print("to save weight", LEFT, 10000);
  hc08.write("Press button to save Weight\n");
  delay(5000); // Display message for 5 seconds

  // Start continuous weight display
  while (true) {
    // Clear previous display
    myGLCD.clrScr();
    myGLCD.setColor(255, 255, 255);
    myGLCD.print("weight=", LEFT, 1);
    hc08.write("Weight = ");
    // Get current weight
    currentWeight = scale.get_units(5);
    myGLCD.printNumI(currentWeight, LEFT, 10000);
    myGLCD.print("g", RIGHT, 10000);
    Serial.print("UNITS: ");
    Serial.println(currentWeight);
    hc08.write(currentWeight);
```

```
hc08.write(" g\n");
// Check if the weight changed significantly
if (abs(currentWeight - lastWeight) > 10) {
  lastChangeTime = millis(); // Reset the timer if weight changes
  lastWeight = currentWeight;
}

// Check if button is pressed to save weight
int bread = digitalRead(3); // Button for saving weight
if (bread == HIGH) {
  buzz();
  myFile = SD.open("test.txt", FILE_WRITE);
  if (myFile) {
    Serial.print("Writing to test.txt...");
    myFile.println(currentWeight);
    // close the file:
    myFile.close();
    Serial.println("done.");
  } else {
  // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
  Serial.println("Weight saved.");
  hc08.write("Weight Saved\n");
  myGLCD.clrScr();
  myGLCD.print("Weight", LEFT, 1);
  myGLCD.print("Saved!", LEFT, 10000);
  delay(2000); // Show saved message for 2 seconds
}

// Delay between readings
delay(2000);

// Save offset and scale settings
scale.set_offset(offset);
scale.set_scale(scale1);

// Check if 2 minutes have passed without significant weight change
if (millis() - lastChangeTime > 120000) { // 2 minutes = 120000 ms
  hc08.write("Turning off");
  buzz();
  buzz();
  buzz();
  myGLCD.clrScr();
  myGLCD.print("No change", LEFT, 1);
  myGLCD.print("in weight.", LEFT, 10000);
  myGLCD.print("Turning off", LEFT, 20000);
  delay(5000); // Show message for 5 seconds
  myGLCD.clrScr();
  myGLCD.print("Goodbye!", LEFT, 1);
  delay(2000); // Show goodbye message for 2 seconds
```

```
      myGLCD.clrScr(); // Turn off the display
      while (1); // End the program
    }
  }
}

void buzz(){
  pinMode(buzzer, OUTPUT);
  tone(buzzer,3);
  delay(150);
  tone(buzzer,6);
  delay(150);
  tone(buzzer,10);
  delay(150);
  noTone(buzzer);
}
```

References

Bogde. (n.d.). *HX711: Arduino library for the HX711 24-bit ADC for weigh scales.* GitHub. Retrieved December 2, 2024, from https://github.com/bogde/HX711

PMD Way. (n.d.). *Using the 0.96" 80 x 160 full color IPS LCD module with Arduino.* Retrieved December 2, 2024, from https://pmdway.com/blogs/arduino-tutorials/using-the-0-96-80-x-160-full-color-ips-lcd-module-with-arduino

Arduino Libraries. (n.d.). *SD: Arduino library for SD card functionality.* GitHub. Retrieved December 2, 2024, from https://github.com/arduino-libraries/SD