## Appendix X: Instructions for Construction

Instructions for Constructing Weight Wizard

The steps below are instructions that should be followed when using this product.

The following is a list of materials needed to construct the Weight Wizard:

Table x: Materials Used for Constructing the Weight Wizard

| Item | Description | Qty |
|---|---|---|
| ESP-32 | Built in USB-to-Serial converter, automatic bootloader reset, Lithium Ion/Polymer charger, GPIOs, a dual-core ESP32 chip, 4 MB of SPI Flash, tuned antenna, WiFi *and* Bluetooth Classic/LE support | 2 |
| Battery | Output ranges from 4.2V when completely charged to 3.7V, a capacity of 500mAh for a total of about 1.9 Wh, pre-attached with a 2-pin JST-PH connector as shown and include the necessary protection circuitry | 2 |
| Amplifier | A small breakout board for the HX711 IC that allows you to easily read load cells to measure weight | 2 |
| Display | 8X32 256 LEDs. 32 horizontal LED and 8 vertical LED. Each 5050SMD LED is individually addressable. High quality, The LED material is the best, Using pure gold wire. | 1 |
| Scale | Malama Digital Body Weight Bathroom Scale, Weighing Scale with Step-On Technology, LCD Backlit Display, 400 lbs Accurate Weight Measurements, Silver | 1 |
| Nylon plastic | Nylon is a strong, stiff engineering plastic with outstanding bearing and wear properties. Nylon is frequently used to replace metal bearings and bushings often eliminating the need for external lubrication. | 30 x 30 x 0.5 (cm) 450 $cm^3$ |
| Wood | Wood has aesthetic appeal, versatility, and durability, making it suitable for enhancing the visual appeal and supporting the weight of the base of the display. | 13 x 4.5 x 0.25 (in) 14.625 $in^3$ |

Note: See Bill of Materials (Appendix B) for prices and part numbers.

The following tools are needed to construct the Weight Wizard:

- Vertical band saw
- Hot glue
- Ruler
- Pencil
- Soldering station
- USB-micro to USB-A transfer cable

The following steps are required in order to construct the Wireless Weight Display:

1. Selecting and Disassembling the Bathroom Scale:

- Purchase the Malama Digital Body Weight Bathroom Scale, though other standard bathroom scales can be used as well.
- Carefully disassemble the scale to expose the four load sensors. This is typically done by removing the screws or fasteners from the bottom of the scale.
- *While disassembling, ensure the load sensors and their connections are not damaged.

2. Preparing the Load Sensors:

- Each load sensor has wires attached to it. Identify each sensor's wires and carefully detach the wires.
- Create a Wheatstone bridge circuit by connecting the four white wires from the sensors together and the four black wires together. The Wheatstone bridge is essential for accurate measurement of the load.

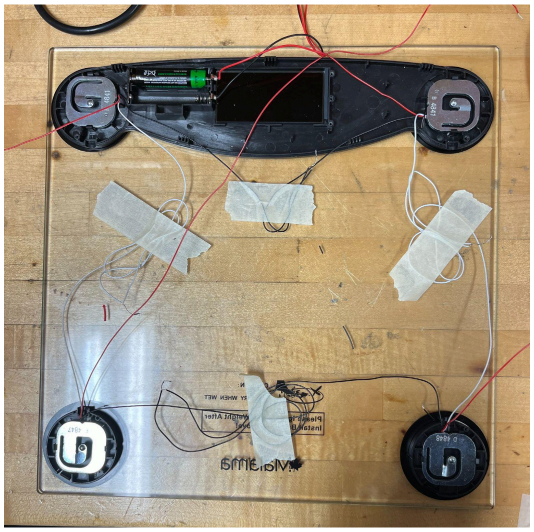

Figure **x**: Creating Wheatstone bridge circuit

3. Amplifying the Signal:

- Connect the red wires from the sensors to a load amplifier. This device amplifies the small signal generated by the load sensors, making it readable by the microcontroller.
- Connect the output of the load amplifier to an ESP32 board microcontroller. The ESP32 will process the signal and enable wireless communication.

4. Setting Up the Arduino IDE for Programming the ESP32:

- Download and install the Arduino IDE from the official Arduino website.
- Open the Arduino IDE, go to File -> Preferences.
- In the "Additional Board Manager URLs" field, enter the URL for the ESP32 board manager (this can be found on the ESP32 GitHub repository or the manufacturer's website).
- Open the Board Manager by going to Tools -> Board -> Boards Manager, search for ESP32, and install the latest version.
- Select the correct ESP32 board from Tools -> Board.

5. Installing the HX711 Library:

- In the Arduino IDE, go to Sketch -> Include Library -> Manage Libraries.
- Search for the HX711 library and install it.

6. Calibrating the Scale:

- Load an example sketch for the HX711 library.
- Define the load cell factor (this will require calibration using known weights).
- Adjust the code to include the WiFi credentials and any other settings specific to your project.
- Upload the sketch to the ESP32 board.

7. Testing and Troubleshooting:

- After uploading the code, test the scale with known weights to ensure accuracy.
- If the readings are off, recalibrate the load cell factor.
- Ensure the wireless connection is stable and the ESP32 is transmitting data correctly.

8. Final Assembly:

- Once the calibration is verified, reassemble the scale.

- Secure the ESP32 and any additional circuitry inside the scale. The wires should not interfere with the load cells themselves, and the wires should be properly and orderly layered such that it is easy for any engineer to understand the wiring scheme upon glance.

9. Building the Display Housing:

- Cut two rectangles from nylon plastic, ensuring they have differing widths but equal lengths. This plastic is chosen for its durability and transparency.
- Use hot glue to attach these rectangles at right angles to each other. This configuration maximizes surface area and will serve as the main body of the display housing.
- For the base and sides of the housing, use inch-thick wood. Cut pieces to the required size to cover the bottom and two sides of the plastic structure.

10. Assembling the Light Display Board:

- Inside this housing, place a 256-light display board. The transparency of the nylon plastic will allow the light from the display to be visible.
- Secure the display board in place, ensuring it is aligned correctly and firmly attached within the housing.

11. Wiring the Second ESP32 for the Display:

- Wire the second ESP32 board to the light display board. Ensure proper connections for power and data transfer.
- Double-check all connections for stability and correctness.

12. Programming the ESP32 for Display Communication:

- In the Arduino IDE, write a sketch to program the ESP32 connected to the display.
- This involves writing a function to receive weight data wirelessly from the ESP32 on the scale and implementing a communication protocol (such as MQTT or HTTP) to transmit data between the two ESP32 boards.

```
#include the necessary libraries:
#include <WiFi.h>
#include <PubSubClient.h>

#Set up the WiFi and MQTT server details:
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
const char* mqtt_server = "your_MQTT_server_address";
#Initialize the WiFi and MQTT client:
WiFiClient espClient;
PubSubClient client(espClient);

#Connect to the WiFi network and MQTT server:
void setup() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  client.setServer(mqtt_server, 1883);
}

#Subscribe to the MQTT topic that the scale ESP32 publishes the weight data #on:
void reconnect() {
  while (!client.connected()) {
    if (client.connect("displayESP32")) {
      client.subscribe("scale/weightData");
    }
    delay(5000);
  }
}

#Handle the received message:
void callback(char* topic, byte* message, unsigned int length) {
  String messageTemp;

  for (int i = 0; i < length; i++) {
    messageTemp += (char)message[i];
  }

  if (String(topic) == "scale/weightData") {
    displayWeight(messageTemp);
  }
}
```

- *Ensure the ESP32 is configured to connect to the same network as the ESP32 on the scale for seamless data transfer.

13. Displaying the Weight Data:

- Code the ESP32 to convert the received weight data into a format suitable for the light display board.
- This involves writing a function to program the display board to show the weight readings in real-time as the scale measures weight.
- *Test the communication between the scale and the display. Adjust the network settings or code as necessary for reliable data transfer.

```
#In the main loop, keep checking and maintaining the MQTT connection:
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```

14. Displaying the Weight Data with Color Indicators:

- In your Arduino IDE sketch, add functions to control the display board. This involves writing a function to display text on the board in different colors:

```
void displayText(String text, uint32_t color) {
   // Code to display the text on the board in the specified color.
   // Implementation depends on your hardware specifics.
}

uint32_t RED = 0xFF0000;
uint32_t YELLOW = 0xFFFF00;
uint32_t GREEN = 0x00FF00;

Write the displayWeight function to include logic for changing the color based on the weight
threshold:
void displayWeight(String weightStr, float redThreshold) {
   float weight = weightStr.toFloat();
   uint32_t displayColor;
```

```
    if (weight >= redThreshold) {
        displayColor = RED;  // Weight exceeds the threshold, display in red.
    } else if (weight >= redThreshold * 0.9) {
        displayColor = YELLOW;  // Weight is close to the threshold, display in yellow.
    } else {
        displayColor = GREEN;  // Normal weight, display in green.
    }

    displayText(weightStr, displayColor);  // Display the weight with the appropriate color.
}
```

Modify the MQTT callback function to include the weight threshold when calling displayWeight:

```
void callback(char* topic, byte* message, unsigned int length) {
    String messageTemp;

    for (int i = 0; i < length; i++) {
        messageTemp += (char)message[i];
    }

    if (String(topic) == "scale/weightData") {
        float redThreshold = 100.0;  // Define the red color threshold.
        displayWeight(messageTemp, redThreshold);
    }
}
```

15. Loop to Keep the Connection Alive:

- In the main loop, maintain the MQTT connection. This involves writing a function to continuously check for new weight data.

```
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}
```

16. Finalizing the Display Unit:

- Assemble the display unit with the ESP32 board connected to the light display board inside the nylon plastic housing.
- Attach the wooden base and sides to complete the housing.
- *Test the entire setup to ensure the display accurately reflects the weight readings and changes colors appropriately based on the set thresholds.

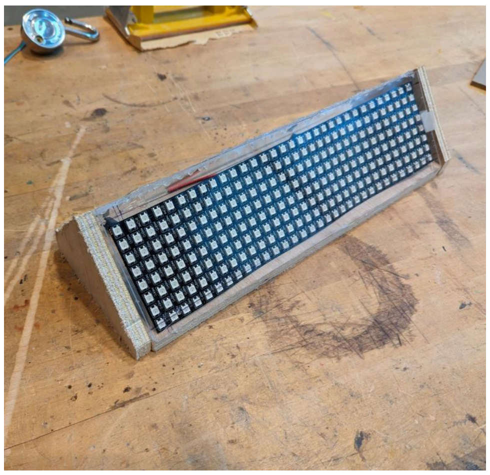

Figure **x**: Assembling the display unit and housing