**AI Integration:**

**Refine your logic or Word Code:**

**Given Prompt:** How can I improve my code further and add weight sensors to check if pets eat properly or not?

**Provided Answer:**

**Required Hardware**

* **HX711 module (for load cell signal amplification)**
* **Load Cell (placed under the food bowl)**
* **Arduino-compatible board**
* **LCD + Servo + Buzzer (already in your setup)**

**Improved Code:**

#include "HX711.h"

HX711 scale ;

#define DT 3

#define SCK 2

float beforeDispenseWeight = 0;

float afterDispenseWeight = 0;

float postMealWeight = 0;

float foodDispensedThreshold = 10.0; // grams

float uneatenThreshold = 5.0; // grams

void setup() {

scale.begin(DT, SCK);

scale.set\_scale(); // Calibrate this value

scale.tare(); // Zero the scale

lcd.begin(16, 2);

pinMode(buzzerPin, OUTPUT);

pinMode(foodSensorPin, INPUT);

}

void loop() {

if (currentTime == feedingTime && digitalRead(foodSensorPin) == HIGH) {

beforeDispenseWeight = scale.get\_units();

feederServo.write(90);

delay(2000);

feederServo.write(0);

delay(3000); // Wait for food to settle

afterDispenseWeight = scale.get\_units();

float dispensedAmount = afterDispenseWeight - beforeDispenseWeight;

if (dispensedAmount >= foodDispensedThreshold) {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Food Dispensed");

delay(600000); // Wait 10 minutes for pet to eat

postMealWeight = scale.get\_units();

float eatenAmount = afterDispenseWeight - postMealWeight;

lcd.setCursor(0, 1);

if (eatenAmount >= uneatenThreshold) {

lcd.print("Meal Eaten");

digitalWrite(buzzerPin, LOW);

} else {

lcd.print("Not Eaten!");

digitalWrite(buzzerPin, HIGH);

}

} else {

lcd.clear();

lcd.print("Dispense Fail");

digitalWrite(buzzerPin, HIGH);

}

} else if (digitalRead(foodSensorPin) == LOW) {

lcd.clear();

lcd.print("Low Food!");

digitalWrite(buzzerPin, HIGH);

} else {

digitalWrite(buzzerPin, LOW);

}

}

**Explore real-world implementation:**

**Given Prompt:** How my system can be built using actual hardware?

**Provided Answer:**

Building Your Smart Pet Feeder with Real Hardware

Core Components

| Component | Purpose | Reference |
| --- | --- | --- |
| Arduino Uno | Main controller for scheduling and I/O | |1 |
| Servo Motor (SG90) | Dispenses food via rotating dispenser flap | | 2 |
| HX711 + Load Cell | Measures bowl weight before/after dispensing | — |
| Food Sensor (IR/Limit) | Detects if there’s food in the hopper | — |
| LCD Display (16×2 I2C) | Shows status (next feed, weight, alerts) | | |
| Buzzer | Audible alert for low food or uneaten meals | — |
| RTC Module (DS3231) | Keeps accurate real-time clock | | |
| Power Supply (5 V) | Powers Arduino, servo, sensors, and display | — |
| Wiring & Connectors | Jumper wires, breadboard or soldered PCB | — |
| Optional 3D-Printed Case | Neat housing, hopper mount, servo support | | |

Assembly Steps

1. Mount the servo motor so its horn can open/close a flap on your food hopper.
2. Place the load cell under your pet’s bowl; secure it on a flat base.
3. Wire the HX711 to the load cell, then connect HX711 to Arduino pins:
   * VCC → 5 V
   * GND → GND
   * DT → D3
   * SCK → D2
4. Hook up the food-level sensor at the hopper’s outlet: signal pin → a digital input (e.g., D4), VCC → 5 V, GND → GND.
5. Connect the I2C LCD:
   * SDA → A4
   * SCL → A5
   * VCC → 5 V
   * GND → GND
6. Wire the DS3231 RTC module:
   * SDA → A4
   * SCL → A5
   * VCC → 5 V
   * GND → GND
7. Attach the buzzer to a digital pin (e.g., D8) with its other lead to GND.
8. Power the system with a stable 5 V supply; ensure the servo has a separate regulator or decoupling capacitor to prevent brown-outs.
9. Secure all wiring inside your housing or on a DIY chassis to prevent pets from chewing or knocking components loose.

Calibration & Testing

* Calibrate the HX711’s scale factor using known weights; use scale.set\_scale(factor) in setup().
* Tare the scale at startup (scale.tare()) so “0 g” matches an empty bowl.
* Test servo endpoints (write(0) and write(90)) to find the exact angles that open and close your hopper.
* Verify RTC accuracy by printing timestamps to Serial and comparing with your phone’s clock.

Optional Enhancements

* Log timestamps and weight readings to an SD card or EEPROM for consumption history.
* Swap delay() for a non-blocking millis() scheduler to keep the UI responsive.
* Add Wi-Fi (ESP8266/ESP32) to view feeding data remotely or adjust schedules via a web interface.
* Design and 3D-print a custom enclosure with snap-fit parts for food safety and aesthetics.