# **Automated Pet Feeder System**

**Background:** A local animal shelter is looking for a low-cost, programmable automated pet feeder that can:

- Dispense food for cats and dogs at scheduled times.

- Monitor whether food has been consumed or the amount of food that has been

consumed.

- Alert staff if there’s an issue (e.g., no food dispensed, food not eaten).

They want a solution that could eventually be implemented using low-cost components (like a servo motor and sensors), but your task is to design and simulate the logic and behaviour of the system first.

**Step 1:** **Understand and Define the Problem (Analyse)**

**Goal:** Design an automated pet feeder for cats and dogs and monitor food availability to ensure pets are fed reliably. 

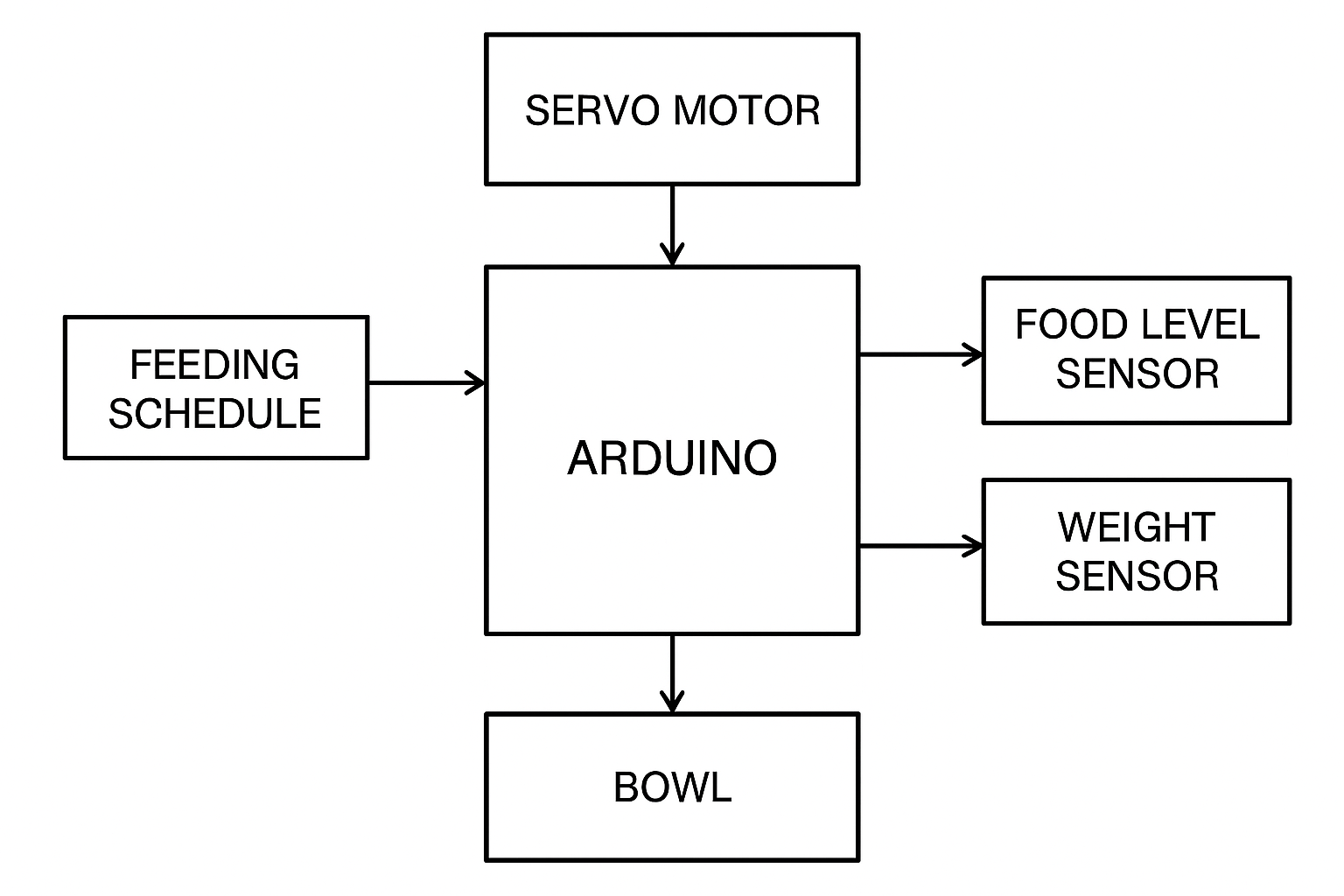
**Constraints:** The automated feeder must be low cost, reliable, safe for pets, and implementable using Arduino and affordable components like servo motors and sensors.

**Feeder's Functions and features:**

* The feeder is designed for cats and dogs.
* Portion sizes are adjustable, generally ranging from 50 grams to 300 grams per feeding.
* The system supports up to four feeding times per day (e.g., 8 AM, 12 PM, 6 PM, 9 PM).
* Sensors include a food level sensor inside the dispenser and a weight sensor under the bowl.
* Alerts notify staff or owners if food runs low or if food is not eaten within 10 minutes after dispensing with a buzzer.

**Feeder’s Specifications:**

* **Pet type:** Cats and dogs
* **Portion size:** Adjustable, 50 g – 300 g per feeding
* **Feeding times:** Up to 4 scheduled feedings per day (e.g., 08:00, 12:00, 18:00(9 PM), 21:00(9 PM))
* **Sensors:**
  + Food level sensor (inside dispenser)
  + Weight sensor (under bowl)
* **Alerts:**
  + Low food level alert
  + Uneaten food alert (if food remains after 10 minutes)
  + If weight doesn’t change after 10 minutes of dispensing food



DISPLAY

**Step 2:** **Organise and Describe the Data**

**Input Types**

* **Real-time clock to keep track of current time.**
* **Food level sensor inside the dispenser to detect remaining food.**
* **Weight sensor under the bowl to monitor food consumption.**
* **Manual feed request input from user (e.g., button press).**

**Expected Outputs:**

* **Servo motor control to rotate and dispense food.**
* **Alert system (Buzzer) to notify low food levels or feeding errors.**
* **Display output to show food level status and next scheduled feeding time.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Type** | **Units** | **Sample Values** | **Notes /Constraints** |
| **Feeding time** | **Input** | **Time** | **08:00, 12:00, 18:00, 21:00** | **Up to 4 scheduled feedings per day** |
| **Food level status** | **Input** | **Boolean** | **TRUE (means food available), FALSE** | **Alert if food level below 10%** |
| **Bowl weight**  **(weight sensor)** | **Input** | **Grams** | **0g to 300g** | **Detects if pet has eaten (weight increase)** |
| **Uneaten Food** | **Output** | **Buzzer** | **Buzzer sound** | **Triggers** **if no weight decrease after 10 min** |
| **Servo motor control** | **Output** | **ON/OFF** | **ON (will dispense food), OFF** | **Dispenses a fixed portion of food** |
| **Alert signal** | **Output** | **Buzzer** | **Buzzer sound** | **Triggers on low food or feeding error** |
| **Display info** | **Output** | **Text** | **“Food: 50% (Remaining food)”, “Next feed: 18:00(next feeding time)”** | **Updated in real-time** |

**Step 3:** **Plan the Solution (Design the Algorithm)**

**Algorithm:**

IF Feeding Time:{

Check current time.

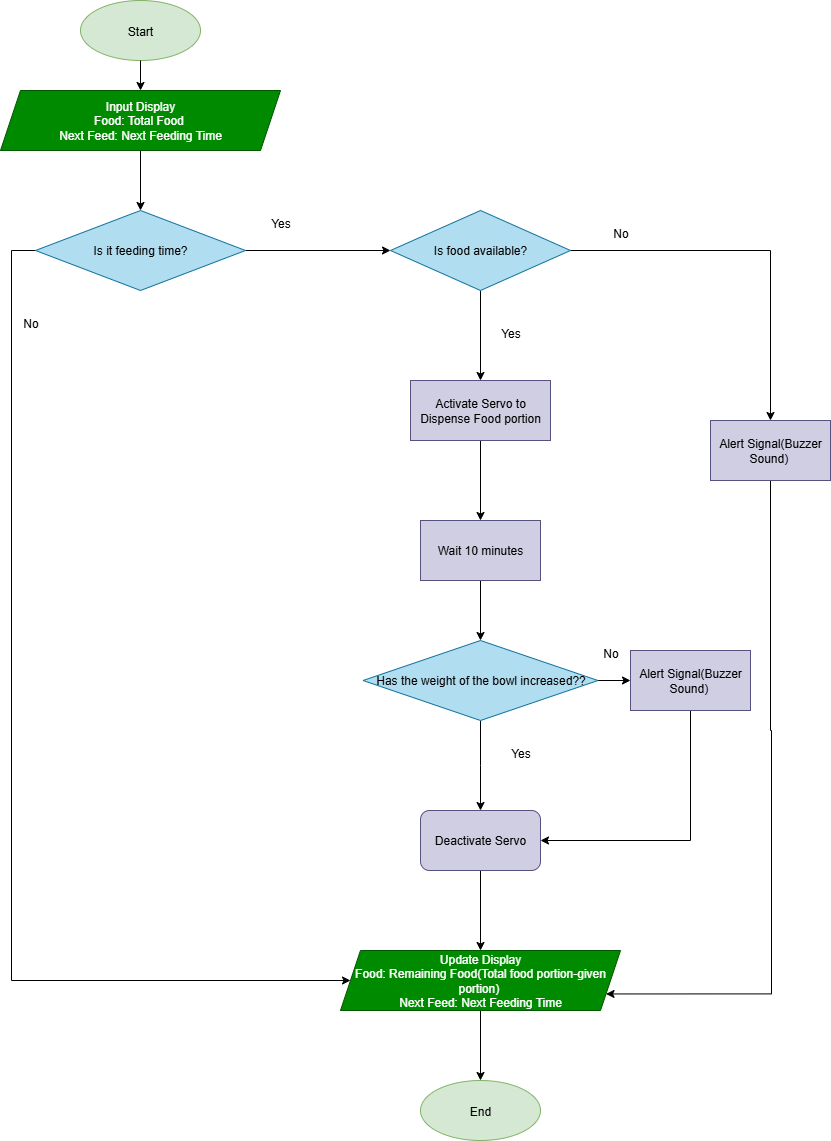
IF{ current time matches any scheduled times:   
  - IF { food level sensor detects food available == TRUE:   
    - Activate servo to dispense set portion.   
    - Wait 10 minutes.   
    - IF bowl weight has increased:   
      - Feeding successful. }  
    - ELSE: {  
      - Trigger alert (“Food not dispensed or pet not eating”). } }  
  - ELSE: {  
    - Trigger alert (“Low Food Level”). }

Show next feed time and current food level on display. }

-ELSE:{

Show next feed time and current food level on display.

}

**Flowchart:** 

**Step 4:** **implement the Solution (Word Coding)**

**Code:**

**Platform: Arduino UNO**

**Language: Arduino C/C++**

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

  // the feeding time is correct and there’s enough food available

    feederServo.write(90);     // Dispense food portion

    delay(2000);               // Wait 2 seconds

    feederServo.write(0);      // Reset servo position

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print("Next: (The Next Feeding Time)");  // Replace with actual next feeding time

    lcd.setCursor(0, 1);

    lcd.print("Food: (Remaining Food Percent)"); // Replace with actual food percent

    digitalWrite(buzzerPin, LOW);  // Make sure buzzer is off during feeding

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

    lcd.clear();

    lcd.print("Low Food!");

    digitalWrite(buzzerPin, HIGH); // Turn buzzer on for low food alert

} else {

    digitalWrite(buzzerPin, LOW);  // Turn buzzer off if no alert  }

**Step 5: Test and Refine the Solution (Debug and Verify)**

**Sample Scenarios:**

**- Pet eats as expected**

**- Pet does not eat**

**- Food bin is empty**

**- Compare output with expectations**

**- Suggest improvements**

**1. Pet Eats as Expected**

**Scenario:**

* It's feeding time (currentTime == feedingTime).
* Food bin is not empty (foodSensorPin == HIGH).

**Expected Patterns:**

* Servo dispenses food.
* LCD shows next feeding time and remaining food percent.
* Buzzer is off.

**Test Output:**

* Servo rotates to 90°, waits, then returns to 0°.
* LCD line 1: "Next: (The Next Feeding Time)"
* LCD line 2: "Food: (Remaining Food Percent)"
* Buzzer is silent.

**Result:**

* **Works as intended for basic feeding.**

**2. Pet Does Not Eat**

**Scenario:**

* It's feeding time.
* Food bin is not empty.
* Pet does not eat (no sensor detects food is eaten).

**Expected Patterns:**

* System behaves exactly as above.
* **Limitation:** The code does not detect or care if the pet eats or not.

**Test Output:**

* Same as "Pet Eats as Expected."

**Result:**

* **Limitation:** The system cannot detect whether the pet has actually eaten.
* **Improvement:** To detect this, you would need a separate "food taken" sensor or a weight sensor.

**3. Food Bin is Empty**

**Scenario:**

* At any time (feeding or not).
* Food bin is empty (foodSensorPin == LOW).

**Expected Patterns:**

* LCD shows "Low Food!"
* Buzzer is ON.

**Test Output:**

* LCD: "Low Food!"
* Buzzer sounds.

**Result:**

* **Works as intended for low food warning.**

**4. Compare Output with Expectations**

* **Feeding event:** Works as intended if food is present.
* **Low food:** LCD and buzzer alert you.
* **Pet eating detection:** **Not implemented**—system can't tell if food is actually eaten.
* **Display:** Both "Next" and "Food" messages show correctly on 2 lines.
* **Buzzer:** Properly managed.

**5. Suggestions for Improvement**

**Detect if pet eats**

* Add a mechanism (e.g., weight sensor, IR sensor in bowl) to check if food was removed after dispensing.

**6. System Refinements**

* **Stable for basic auto-feeding and low food alerting.**
* **Not capable of monitoring if pet actually eats.**
  + Add hardware if this is a requirement.
* **Easy to read and manage display.**
* **Buzzer logic is sound.**

| **Scenario** | **Expected Output** | **Actual Output** | **Meets Expectation?** | **Note/Improvement** |
| --- | --- | --- | --- | --- |
| Pet eats as expected | Dispense, update LCD, buzzer off | Yes | Yes | - |
| Pet does not eat | Dispense, update LCD, buzzer off | Yes | Yes (but limited) | Can't detect if pet ate |
| Food bin is empty | LCD "Low Food!", buzzer on | Yes | Yes | - |

**Summary Table:**

**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.

Reflection:

I began by asking Copilot how can I improve my code further and add weight sensors to check if pets eat properly or not?. It responded with a clear, commented Arduino C++ snippet that used an HX711 load cell to measure before-and-after weights, threshold checks for dispensing success, and a ten-minute eat-or-alert routine. That example helped me improve my previous code and implement weight sensors .

Next, I asked how my system can be built using actual hardware?. Copilot answered with a detailed parts list like Arduino Uno, SG90 servo, HX711 module, DS3231 RTC, IR food sensor, 16×2 I2C LCD, buzzer—and step-by-step wiring instructions. It even suggested using Fritzing for PCB layouts and adding Wi-Fi for remote control.

The information provided by Copilot helped me improve my code and enhanced my knowledge about building automated machines and codes behind it.