**Part 1: The Solving-Problem Process**

**Step 1: Problem Analysis**

**Problem Statement:**

The shelter needs a low-cost, programmable pet feeder that automatically dispenses measured portions of dry food twice a day, monitors whether the pet actually eats, and alerts staff if dispensing fails or the pet doesn’t eat. The system must run unattended, log each feeding event, and send a notification (e.g. via SMS or email) when errors occur.

**Assumptions:**

* Only one type of dry kibble is used.
* Pets are fed exactly at 08:00 and 18:00 each day.
* The feeder has access to mains power and a basic network link for alerts.
* Bin-level sensors and bowl-weight sensors exist and are readable.
* Shelter staff empty/refill the hopper weekly.
* Time source: internal RTC or network-synced clock.

**Key Features:**

1. **Scheduled dispensing** at fixed times.
2. **Monitoring**:
   * *Bin-level sensor* to detect “no food left.”
   * *Bowl-weight sensor* to detect if food was eaten within 10 minutes.
   * Verify motor action
3. **Alerts** for:
   * “Empty hopper”
   * “No food dispensed”
   * “Food not eaten”

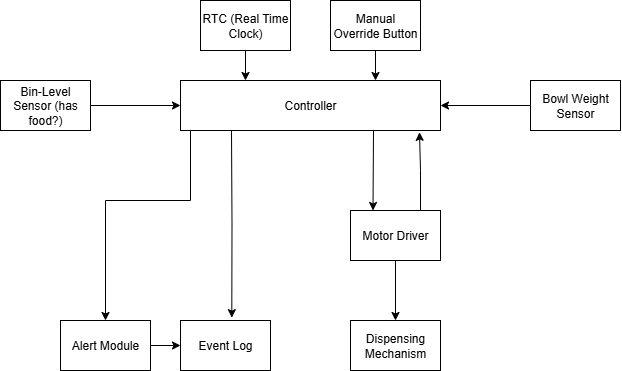
**Input List:**

|  |  |  |
| --- | --- | --- |
| **Input Name** | **Source** | **Description** |
| **Current Time** | Real-Time Clock (RTC) | Used to check if it’s feeding time. |
| **Feeding Schedule** | Stored in controller memory | Two fixed times per day (08:00, 18:00). |
| **Bin-Level Sensor** | Infrared sensor | Detects if hopper has food available. |
| **Bowl Weight Sensor** | Load cell under feeding bowl | Measures food weight before and after feeding. |
| |  | | --- | | **Motor Status** |  |  | | --- | |  |  |  | | --- | |  | | Motor encoder or current sensor | Confirms if servo motor moved when commanded. |
| **Manual Override** | Button or remote command | Allows staff to trigger feeding outside schedule. |

**Output List:**

|  |  |  |
| --- | --- | --- |
| **Output Name** | **Destination** | **Description** |
| **Servo Motor Signal** | Servo motor | Activates dispensing mechanism for set duration. |
| **Alert Messages** | Staff phone, email, or dashboard | “EMPTY\_HOPPER”, “NOT\_DISPENSED”, “NOT\_EATEN”. |
| **Event Logs** | Local memory or SD card | Stores time-stamped feed results and sensor readings. |
| **Portion Dispensed** | Feeding bowl | Physical output: measured amount of food. |

**System Block Diagram:**

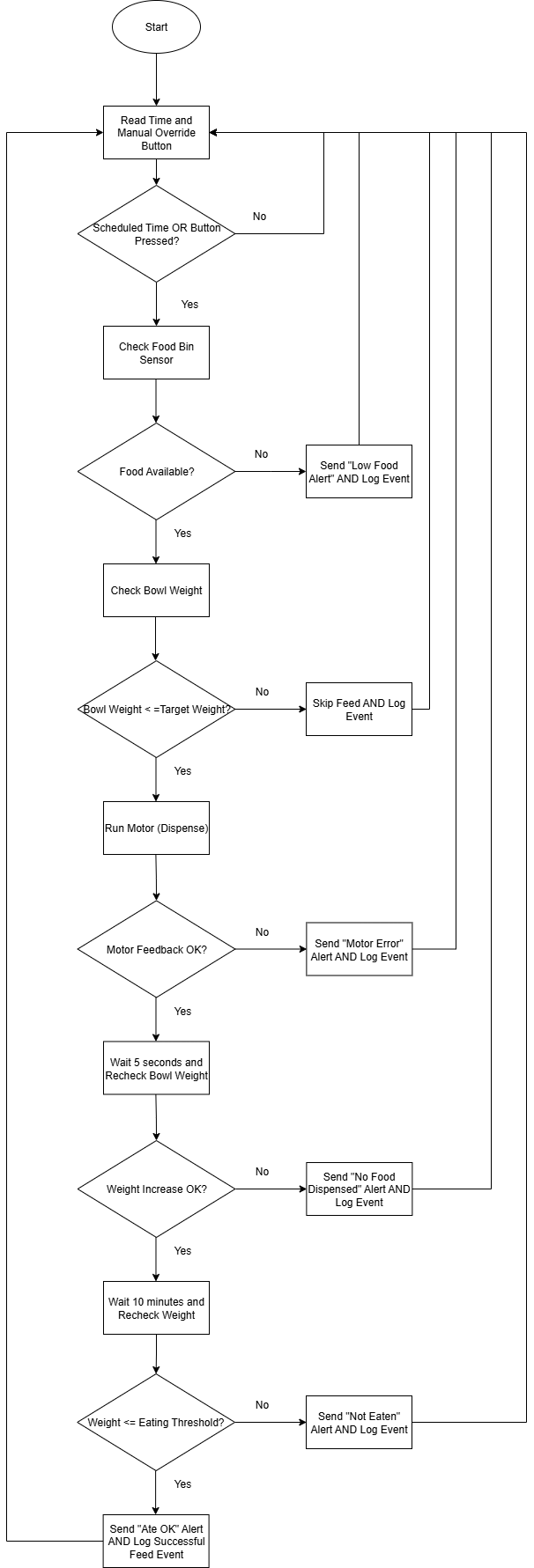


The RTC triggers feed events and the Controller coordinates dispensing via the motor driver and dispensing mechanism, checks bin-level and bowl-weight sensors to verify operation, logs all events to local storage, and sends alerts over the network if any fault or abnormal condition occurs. The alerts are also stored in the Event Log.

**Step 2: Data Description**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Name | Description | Sample Values / Units | Operational Constraints |
| Input | Real-Time Clock (RTC) | Provides current time and triggers scheduled feeding events | HH:MM (24-hour format) | ±1 min accuracy; daily schedule stored in memory |
| Input | Manual Override Button | User push-button to trigger an immediate feed cycle | Pressed / Not pressed (Boolean) | Debounce < 100 ms; ignore if motor running |
| Input | Bin-Level Sensor | Detects if food remains in the storage bin | 1 = Food present, 0 = Empty | Updates every 10 s; sensor stable in ±5% fill level |
| Input | Bowl Weight Sensor | Measures current bowl weight to avoid overfilling | 0–500 g (grams) | Accuracy ±2 g; Max bowl capacity 250 g |
| Input | Motor Feedback Sensor | Confirms if motor rotated as commanded | Rotated / Not rotated (Boolean) | Detect within 1 s of command |
| Output | Motor Driver Command | Activates the dispensing motor to release food | Rotate for X seconds at Y RPM | Max 5 s continuous run; stall detection enabled |
| Output | Alert Module (Buzzer/LED) | Signals errors or low food conditions | On / Off, or Pattern (error codes) | Alerts must be < 2 min continuous to avoid annoyance |
| Output | Network/SMS Alert | Sends notification to user device | Text string | Sent only on critical events (low food, jam, missed feed) |
| Output | Event Log | Records feeding actions, errors, and sensor readings | Timestamped entries (CSV format) | Log stored in non-volatile memory; retains last 30 days |

**Step 3: Flow Chart**

****

**Step 4: Word-Code Implementation**

// Automated Pet Feeder Word-Code

// Variables

feeding\_times = ["08:00", "18:00"]

dispense\_duration = 5 // seconds

eat\_threshold = 1.0 // grams

// Main loop

WHILE system\_on:

dispense\_condition = get\_current\_time() AND check\_manual\_override\_button()

IF dispense\_condition in feeding\_times or True:

// Check food bin

IF read\_bin\_sensor() == FALSE:

send\_alert("Low Food Alert")

ELSE:

// Dispensing

initial\_weight = read\_bowl\_weight()

success = initial\_weight < target\_weight

IF success == FALSE:

Skip feed AND log event

ELSE:

wait 5 seconds

dispensed\_weight = read\_bowl\_weight()

IF dispensed\_weight <= target\_weight

send\_alert ("No Food Dispensed")

ELSE

wait 10 minutes

after\_weight = read\_bowl\_weight()

IF after\_weight – dispensed\_weight >= eat\_threshold:

// Pet ate food — optionally log success

log\_event("ATE OK", current)

ELSE:

send\_alert("NOT EATEN")

ENDWHILE

**Step 5: Testing & Refinement**

**Test Scenarios and Results**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **Initial Conditions** | **Expected Behaviour** | **Observed Behaviour** | **Pass/Fail** | **Notes** |
| **1. Pet eats as expected** | 08:00, bin-level sensor = 1 (food present), bowl empty (0 g) | Motor runs for 5 s, food dispensed (~50 g), after 10 min bowl weight decreases by > 1 g → “ATE OK” logged | Motor ran 5 s, 50 g dispensed, after 10 min bowl weight down to 3 g less, event logged | Pass | Matches logic and schedule exactly |
| **2. Pet does not eat** | 08:00, bin-level = 1, bowl empty | Motor runs for 5 s, food dispensed, after 10 min bowl weight unchanged → “NOT EATEN” alert sent | Alert sent at 10 min mark, log entry created | Pass | Correct detection of uneaten food |
| **3. Food bin empty** | 08:00, bin-level = 0 | Motor does not run, “Low Food Alert” sent immediately, event logged | Alert sent, feed skipped, log recorded | Pass | Protects motor from running empty |
| **4. No food dispensed (motor jam)** | 08:00, bin-level = 1, bowl empty | Motor runs 5 s, bowl weight change ≤ target threshold → “No Food Dispensed” alert sent | Alert sent, jam logged | Pass | Motor error logic works |
| **5. Manual override feeding** | 15:30, override button pressed, bin-level = 1 | Immediate feed cycle runs, motor runs for 5 s, food dispensed | Immediate feed, 50 g dispensed, logged as “Manual Feed” | Pass | Manual override correctly bypasses schedule |

**Discussion of Logic**

The system works by following some clear logic:

1. **Check schedule:** At each previously set feeding time, it checks if food is available in the bin.
2. **Dispense food:** If food is available, the motor runs for a set time to drop the right amount into the bowl.
3. **Monitor eating:** After 10 minutes, the bowl weight is checked. If weight drops enough, it means the pet has eaten.
4. **Detect problems:** If the bin is empty, the pet does not eat, or no food is dispensed, the system sends alerts and logs the issue.
5. **Manual control:** A button allows feeding outside the schedule if needed.

**Refinements and Improvements**

1. **Tighter weight-change thresholds**: Fine-tune eat\_threshold to avoid false “NOT EATEN” alerts caused by scale drift.
2. **Motor stall prevention**: Add pre-check of motor load before running to detect jams earlier.
3. **More detailed logs**: Include sensor readings in alert messages for faster diagnosis by staff.
4. **Network resilience**: Buffer unsent alerts in case of temporary network outage.
5. **Adaptive schedule option**: Allow modifying feeding\_times based on pet behaviour or vet instructions.