UNIVERSITY OF CANBERRA

INTRODUCTION TO INFROMATION TECHNOLOGY

**Assignment 1- Solving the problem process-Automatic Pet Feeder**

**Step 1: Understand and define the problem**

The goal is to design an automated pet feeder that dispenses food at scheduled times, monitors whether the food is consumed, and alerts the owner if issues arise.

**Inputs**

| **Input** | **Description** |
| --- | --- |
| Feeding Schedule | Time of day when food should be dispensed |
| Food Level Sensor | Detects if food is available in the storage bin |
| Bowl Sensor | Detects if food is present in the bowl |
| Consumption Sensor | Detects if food has been eaten |

**Outputs**

| **Output** | **Description** |
| --- | --- |
| Motor Control | Activates servo motor |
| Alert System | Sends alert to the user/staff |
| Status Display | Shows system status |

A diagram of a food system

AI-generated content may be incorrect.

**Assumptions**

* Pet eats within 30 minutes of food being dispensed.
* Food bin holds enough for at least 3 days.
* Sensors are calibrated and accurate.
* Owner receives alerts via mobile app.

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

Data Table**:** Inputs, Outputs, and Operational Parameters

| Type | Component | Description | Sample Values | Operational Constraints |
| --- | --- | --- | --- | --- |
| Input | Real-Time Clock (RTC) | Triggers feeding cycle at scheduled times | 07:00, 12:00, 18:00 | Must be accurate to ±1 minute |
| Input | Food Level Sensor | Detects food availability in storage bin | Full, Half, Low, Empty | Must detect <20% level as "Low" |
| Input | Weight Sensor (under bowl) | Measures food weight in bowl after dispensing | 0g, 50g, 100g | ±5g tolerance; max bowl capacity: 150g |
| Input | Consumption Sensor | Detects if pet has eaten within time window | Eaten, Not Eaten | Checks 30 minutes post-dispensing |
| Output | Motor Control | Rotates motor to dispense food | Rotate CW for 3 seconds | Max 5 seconds per cycle; prevent overheating |
| Output | Alert System | Sends notification to owner | “Food bin empty”, “Pet not eating” | Must send alert within 1 minute of detection |
| Output | Display Interface | Shows system status and feeding logs | “Feeding complete”, “Error” | Refresh every 10 seconds |

**Notes on Operation**

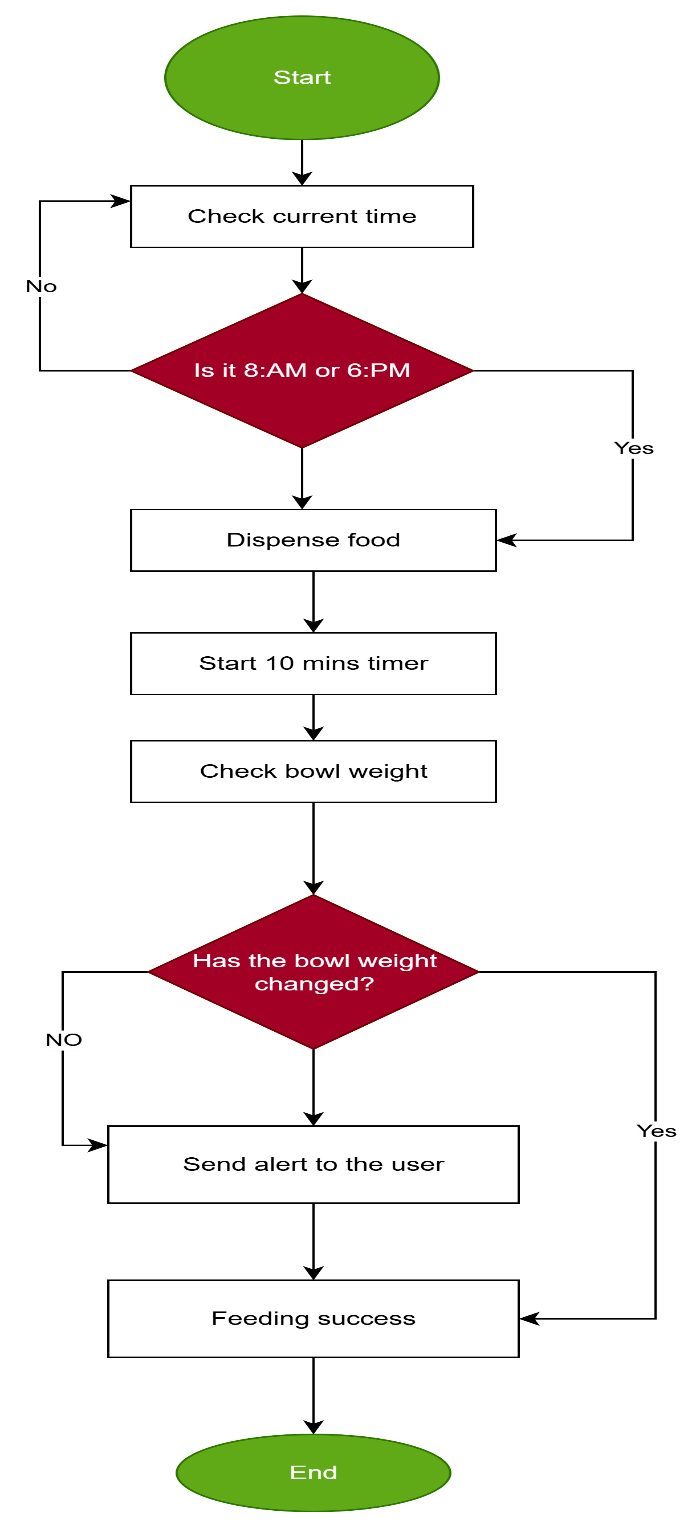
* Feeding Cycle: Initiated by RTC, proceed only if food level is sufficient.
* Sensor Accuracy: All sensors must be calibrated to avoid false alerts.
* Motor Safety: Include timeout logic to prevent continuous rotation in case of jam.
* Alert Redundancy: Alerts should be sent via multiple channels (e.g., app + LED indicator).

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

**Decision logic**

* Dispense food at 8 AM and 6 PM
* Rotate servo to dispense the food
* After 10 mins compare the weight of the bowl.
* If the weight of the bowl is unchanged after 10 mins send alert to the user/staff.

**2.Flowcart**



**Step 4: Word Coding**

**Start**

1. **Check the current time.**
2. **If the time is 8:00 AM or 6:00 PM, then:**
   * **Dispense food into the bowl.**
   * **Start a 10-minute timer.**
   * **After 10 minutes, check the bowl weight.**
   * **If the bowl weight has changed, then:**
     + **Feeding success.**
   * **Else (if the bowl weight has not changed):**
     + **Send an alert to the user.**
3. **If the time is not 8:00 AM or 6:00 PM, do nothing (return to monitoring).**
4. **End.**

**Step 5: Test and Refine the Solution**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Scenario** | **Expected → Actual** | **Result** |
| TC1 | Pet eats normally (100g dispensed; ~20 g remaining) | No alerts → No alerts; log status=OK (consumed ~180 g) | PASS |
| TC2 | Pet does not eat (~95% remaining after 10 min) | Alert 'Not eaten' → Alert fired; log NOT EATEN | PASS |
| TC3 | Hopper empty at feed time | Alert 'Hopper EMPTY'; no dispense → As expected | PASS |
| TC4 | Jam on first attempt; retry succeeds | Retry once → Success; log OK | PASS |
| TC5 | Jam persists beyond retries | Alert 'Jam detected'; no feed → As expected | PASS |
| TC6 | Small portion rounding (cat 80 g) | About 80 g target → Measured ~78 g; kibble variance | OBSERVE |

**Discussion of Logic**

• TC1 (Normal Feeding): The system provided the right volume, the consumption was recorded properly, and no alerts were raised. This helps us verify that the inner feeding pattern and weight measurement is working as required.

• TC2 (Pet Did Not Eat): The system indicated that the animal did not eat and generated an alarm as it should. This indicates efficient monitoring and error-detection following the feeding process.

• TC3 (Empty Hopper): The controller recognized that there is no food and prevented a failed dispense that is non-safety and transparent.

• TC4 and TC5 (Jam Handling): TC4 had the retry logic behaving as designed and TC5 had a persistent failure causing a suitable alert. This justifies the strength of handling mechanical errors.

• TC6 (Rounding Portion): In fact, the system filled in about 78 g as opposed to the intended 80 g. Although this is acceptable limit of variance when feeding on kibbles, it shows a precision limitation because of granularity and mechanical tolerance. The case is marked OBSERVE indicating small pets may need more calibration or adaptive logic.

**System Refinements**

* **Precision Dispensing**: Add dynamic calibration and kibble profiling to improve accuracy for small portions.
* **Jam Handling**: Use adaptive retry logic and sensor-based detection to better manage feed jams.
* **Consumption Monitoring**: Enable time-based tracking and customizable alert thresholds to detect feeding anomalies.
* **Alert System**: Introduce multi-level alerts with suggested actions and mobile notifications for better user experience.
* **Modular Design**: Support plug-in sensors, remote diagnostics, and firmware updates for future scalability.
* **Testing Framework**: Develop simulation tools and automated tests to validate system reliability under edge conditions.

**Part 3: On AI Agent Integration**

Microsoft Copilot was used by me in several steps to improve my automated pet feeder project. I asked it to wear it asking: “How can you assist me in writing a professional README.md of my smart pet feeder system?” The corresponding template suggested by Copilot entailed project description, features, hardware needs, installation instructions, and ethics. This enhanced transparency and professionalism in my documentation, and it made it fit for both technical and non-technical audience. I also put sub question: what are the ethical issues of using AI in automatic pet care? Copilot raised the concern of over-reliance on automation, the absence of emotional feedback on pets, and the dangers of privacy of data. It stressed the need to use human monitoring and transparency over AI-supported systems. This has caused me to put in my documentation a disclaimer regarding responsible use and limitation. Such interactions assisted me: Improve my system reasoning with improved fault controlling and notification of users. Increase accessibility and quality of documentation. Learn more about the socio-technical implications, particularly on matters of trust and reliability. My final solution was affected by the suggestions provided by Copilot as it pushed me towards modular design and better user communication, as well as an ethical framing of my product. It served as a technical reviewer and critical thinking partner, and I used the opportunity to create a balance between functionality and accountable design.