**Assignment 1**

**Introduction to Information Technology**

**Name: Abrarul Hakim Akil**

**ID: u3314160**

**Background: Automated Pet Feeder System**

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).

**Challenge PART 1: On the Solving Problem Process**

**Step 1: Understand and Define the Problem**

A local animal shelter is planning for a low-cost, programmable automated pet feeder which is able to 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. The objective is to design a low-cost, programmable pet feeder for a local animal shelter. It must work for both cats and dogs, considering different feeding behaviors. Constraints include limited budget, technical simplicity, and ethical responsibility to ensure animal welfare.

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

**Features:** Scheduled Feeding (e.g. 6am, 11am, 3pm etc.), Food Consumption Monitoring, Alert for an issue.

**Inputs:** Real-time clock, Food level sensor, Weight sensor under bowl, Feed Quantity, Pet type

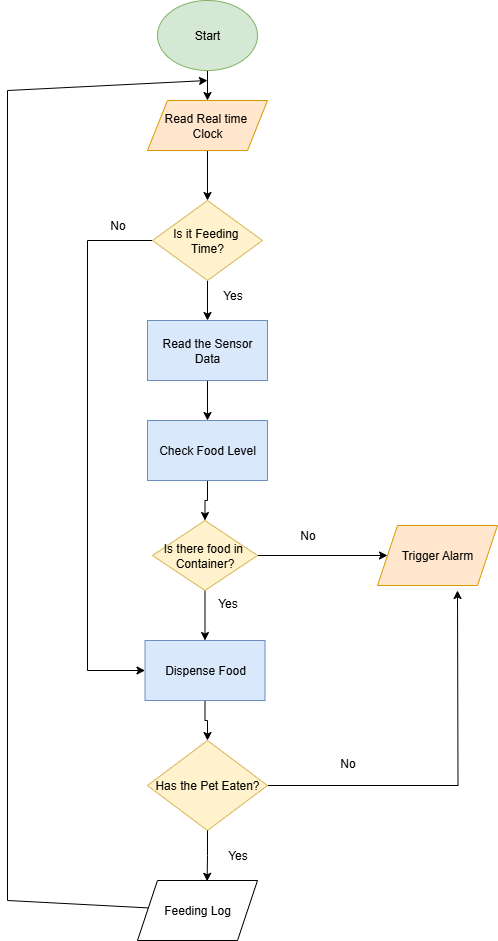
**Outputs:** Servo motor, Alert system, Feeding log

**Assumptions:** Sensors are reliable, Animals eat individually, feeding times are fixed

**Expected Outcome:** Food is dispensed on time, consumption is tracked, staff alerted if issues occur.

**Sketch**: Block diagram — RTC + Sensor → Logic Unit → Servo + Alert + Log System.

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



**Figure: Flowchart of the System**

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

* Read current time from Real Time Clock (input).
* If time matches feeding schedule, then read the sensor data and check the food level
* If there is food in Container then activate servo to dispense food.
* Check sensor to confirm food was dispensed. If not dispensed, trigger the alert.
* Monitor consumption for a set time (e.g., 10 minutes).
* If food not eaten, trigger alert and log issue.
* If food eaten, log success and update feeding history.
* All transactions are stored in a database called “FeedingLog”.

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

* Manual Testing:
* Test servo motor by triggering rotation and observing movement.
* Place weights on load cell to verify accurate sensor readings and stability.

Scenario-Based Testing:

* Schedule feeding time and confirm food is dispensed correctly.
* Monitor bowl to detect pet consumption. If food remains uneaten, system should trigger an alert.
* Check response to pet blocking access and ensure alert logic activates.
* Validate logging of events and sensor data for traceability and performance review.
* Some suggested improvements

1. Adaptive Feeding Schedule
2. Smart Alert System
3. Sensor Health Check
4. Mobile App Integration

Challenge PART 2: On the Use of Technology.

GitHub Link: <https://github.com/AbrarulHakimAkil/u3314160_Assignment-1>

Challenge PART 3: On AI Agent Integration

Use Copilot to assist with at least two or more of the following:

* Refine your logic or Word Code: Ask Copilot to review your Step 4 implementation

and suggest improvements or identify potential issues.

Your implementation outlines a clear sequence for time-based food dispensing, sensor validation, and alert generation. The logic is well-structured and ethically aligned with the shelter’s needs. However, several refinements can enhance reliability, modularity, and fault tolerance:

**✅ Suggested Improvements**

1. Time Synchronization Accuracy Ensure the real-time clock (RTC) is periodically synchronized to avoid drift, especially in long-term deployment.
2. Sensor Redundancy and Validation Introduce cross-verification between the food level sensor and the weight sensor to confirm dispensing success. This reduces false positives in alert generation.
3. Servo Feedback Mechanism Add a passive feedback loop (e.g., limit switch or encoder) to confirm servo movement and detect mechanical failure.
4. **Error Classification in Alerts** Refine alert logic to distinguish between types of issues—e.g., “No food dispensed,” “Food not eaten,” “Sensor failure”—for better staff response.
5. **Dynamic Monitoring Window** Instead of a fixed 10-minute consumption window, consider adjusting based on pet type or historical eating behavior.
6. **Database Logging Granularity** Expand the “FeedingLog” schema to include timestamps, sensor readings, pet ID (if applicable), and alert type for traceability.
7. **Fail-Safe Mechanism** If the system fails to dispense food after multiple attempts, trigger a manual override or escalate the alert priority.
8. **Modular Code Structure** Break down the logic into discrete modules: Time Check, Dispensing, Consumption Monitoring, Alerting, Logging. This simplifies debugging and future upgrades.
9. **Power Stability Check** Monitor voltage levels to ensure consistent servo performance and prevent jitter or incomplete rotation.
10. Ethical Safeguards Include logic to prevent overfeeding or repeated dispensing within short intervals, protecting animal health

* Explore real-world implementation: Use Copilot to discuss how your system could be built using actual hardware (e.g., Arduino, Raspberry Pi).

Real-World Implementation Using Arduino or Raspberry Pi

🟦 1. Microcontroller / Microprocessor

* Arduino Uno: Ideal for simple, low-cost control logic (timing, servo control, sensor input).
* Raspberry Pi: Suitable if you need advanced features like cloud logging, mobile app integration, or camera-based monitoring.

🟩 2. Time Management

* RTC Module (e.g., DS3231): Provides accurate timekeeping for scheduled feeding, even during power loss.

🟨 3. Sensors

* Food Level Sensor: IR or ultrasonic sensor to detect food presence in the container.
* Weight Sensor (Load Cell + HX711 Amplifier): Measures food weight before and after dispensing to confirm consumption.

🟥 4. Actuator

* Servo Motor (e.g., SG90 or MG995): Controls the dispensing mechanism with precise rotation angles.

🟪 5. Alert System

* Buzzer or LED: Local alerts for staff.
* Wi-Fi Module (ESP8266 or built-in Pi Wi-Fi): Sends remote alerts via email or app notification.

🟫 6. Data Logging

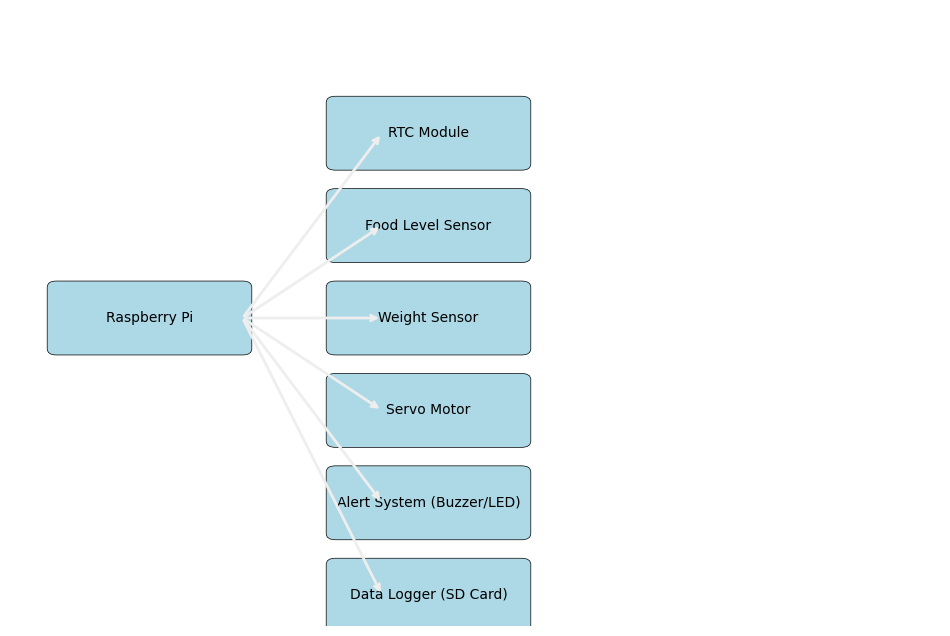
* SD Card Module (Arduino) or Local/Cloud Database (Raspberry Pi): Stores feeding history, sensor readings, and alert logs.

⚪ 7. Power Supply

* 5V Adapter or Battery Pack: Ensure stable power for all components, especially servo motors.

🔷 8. Optional Enhancements

* Touchscreen or LCD Display: For manual override or system status.
* Camera Module (Raspberry Pi): Visual confirmation of pet interaction.
* Mobile App Integration: Via MQTT or HTTP API for remote monitoring and control.



Here’s your **printable wiring diagram** for the automated pet feeder system using Raspberry Pi:

📦 **Included Components**:

* Raspberry Pi (central controller)
* RTC Module (timekeeping)
* Food Level Sensor
* Weight Sensor (load cell)
* Servo Motor (dispensing)
* Alert System (buzzer/LED)
* Data Logger (SD card)

Short Reflection:

I used Copilot to enhance my automated pet feeder system development by refining my concepts and reviewing design procedures and practical construction methods. During my review of Step 4 implementation Copilot recommended implementing servo motor feedback and enhancing alert logic and implementing sensor redundancy. The points enabled me to detect system weaknesses at an early stage which led to improved system reliability.

I requested Copilot to develop a flowchart which presented a detailed sequence of time checks and sensor validation and alert triggers. The system became more straightforward to operate because of this design which also provided better understanding of its decision-making process. The Raspberry Pi hardware design received guidance from Copilot regarding component selection and wiring procedures and system integration. The platform presented real-world examples of homemade feeders which enhanced my ability to visualize the setup.

The final design became more practical and scalable and animal-friendly because of Copilot's structured responses and modular ideas and ethical reminders. The process enhanced my technical solution while demonstrating how digital logic becomes physical systems and why testing and feedback and responsible design matter.