UNIVERSITY OF CANBERRA

INTRODUCTION TO INFORMATION TECHNOLOGY (8936)

Assignment 1: The Solving Problem Process

Name: BIPIN KC

ID: U3309931

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).

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 behavior of the system first.

Step 1: Understand and define the problem (Analyze)

Here, we have the requirement for the local animal shelter which needs an Automated Pet Feeder System that can dispense food for cats and dogs at scheduled times. And the system should be able to monitor whether food has been consumed or the amount of food that has been consumed.

Additionally, staff should be alerted if there's an issue like no food consumed or dispensed from the system.

Features Required:

- **Scheduled Food Dispensing –** The feeder must release food at programmed times (e.g., 8:00 AM, 6:00 PM).
- **Food Consumption Monitoring –** Detect whether the food is eaten and measure the amount consumed.
- **Error Detection & Alerts –** Notify staff if:
    - Food was not dispensed.
    - Food is not eaten after a set time.
    - Food bin is empty.

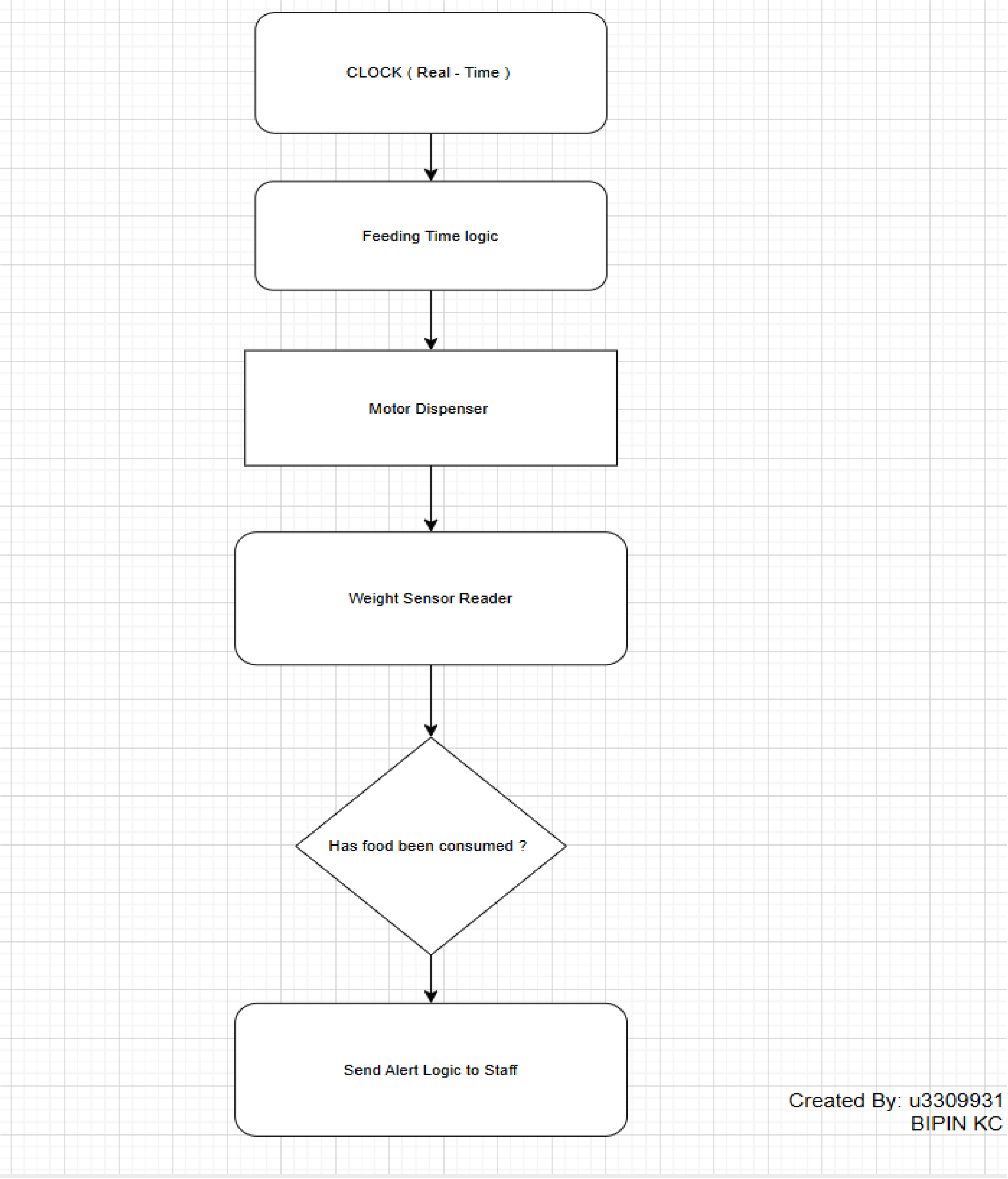
Inputs / Outputs:

Input	Output
Bowl weight sensor data	Display status on LCD
Food Level Sensor	Motors to dispense food
Clock	SMS / Email alert to Staff
Feeding Times	

Pre-Assumptions for the Project:

- There is no change in the feeding time for the pets, it is fixed schedule.
- Weight sensor under the bowl accurately measures the pet food.
- Enough memory to store the feeding data.

Simple Block Diagram: Created using Draw.io:



Step 2: Organize and Describe the Data

Type	Symbol	Meaning	Active Logic
Input	FTR	Feeding Time Reached	1 = Current time matches schedule
Input	FBS	Food Bin Status	1 = Food is present 0 = otherwise
Input	WCAF	Weight Change after Feeding	1= If consumed 0 = if no change
Output	DMA	Dispense Motor Activation	1 = rotate to serve 0 = Do not rotate
Output	ANFC	Alert: No food consumed	1 = Send Alert 0 = No alert
Output	ABE	Alert: Bin has been Empty	1 = Send Alert 0 = No Alert

Truth Table:

FTR (Feeding Time Reached)	**FBS** (Food Bin Status)	**WCAF** (Weight Change after Reading)	**DMA** (Dispense Motor Activation)	**ANFC** (Alert: No food consumed)	**ABE** (Alert: Bin has been empty)
0	0	X	0	0	0
0	1	X	0	0	0
1	0	X	0	0	1
1	1	0	1	1	0
1	1	1	1	0	0

Here,

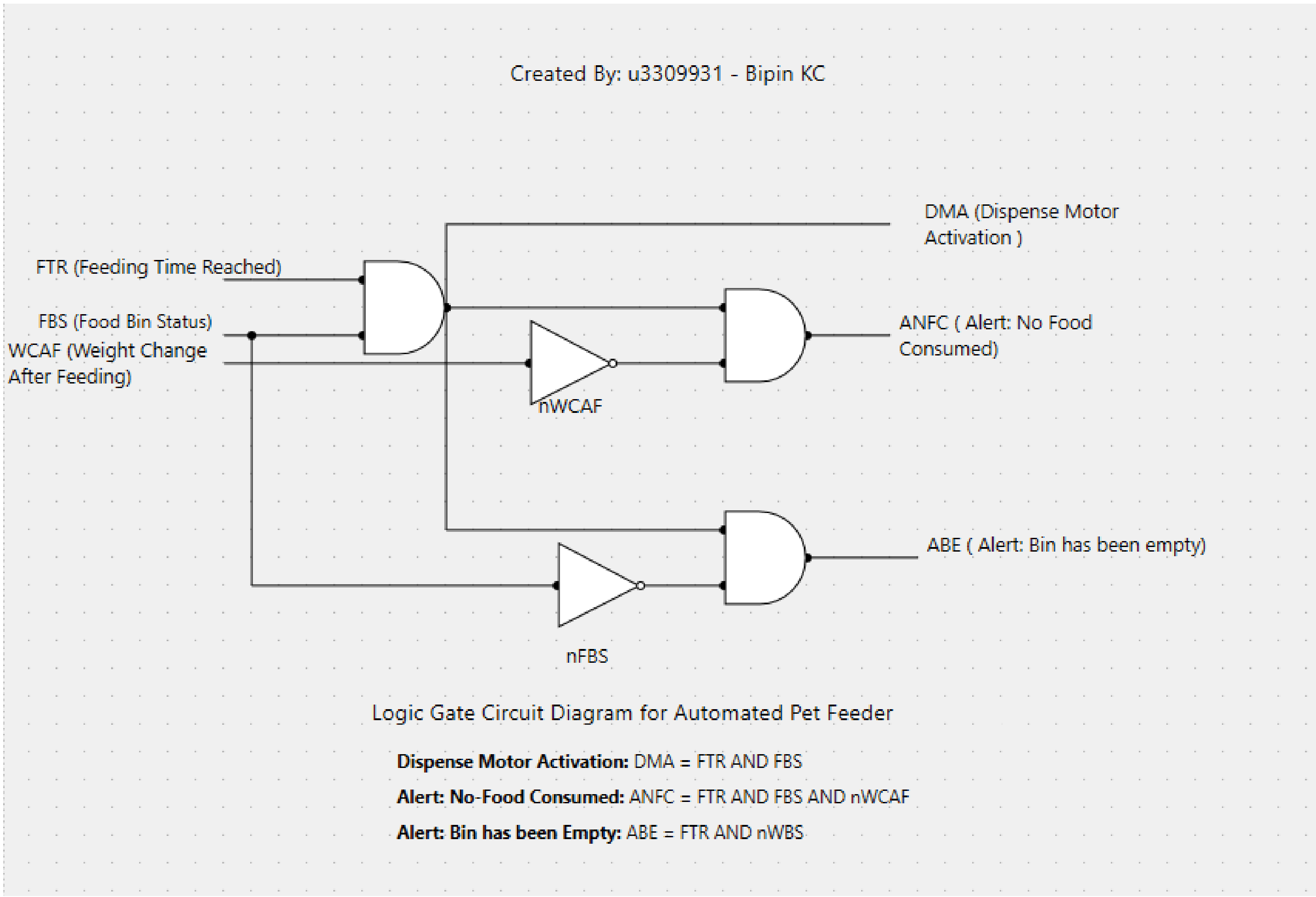
Dispense Motor Activation: DMA = FTR AND FBS

Alert: No-Food Consumed: ANFC = FTR AND FBS AND $\overline{WCAF}$

Alert: Bin has been Empty: ABE = FTR AND $\overline{FBS}$

LOGIC GATE DIAGRAM:

This diagram is created using Logic Circuit windows app, as suggested in Tutorial class.


Created By: u3309931 - Bipin KC
FTR (Feeding Time Reached)
FBS (Food Bin Status)
WCAF (Weight Change After Feeding)
DMA (Dispense Motor Activation)
nWCAF
ANFC (Alert: No Food Consumed)
ABE (Alert: Bin has been empty)
nFBS

Logic Gate Circuit Diagram for Automated Pet Feeder

Dispense Motor Activation: DMA = FTR AND FBS

Alert: No-Food Consumed: ANFC = FTR AND FBS AND nWCAF

Alert: Bin has been Empty: ABE = FTR AND nWBS

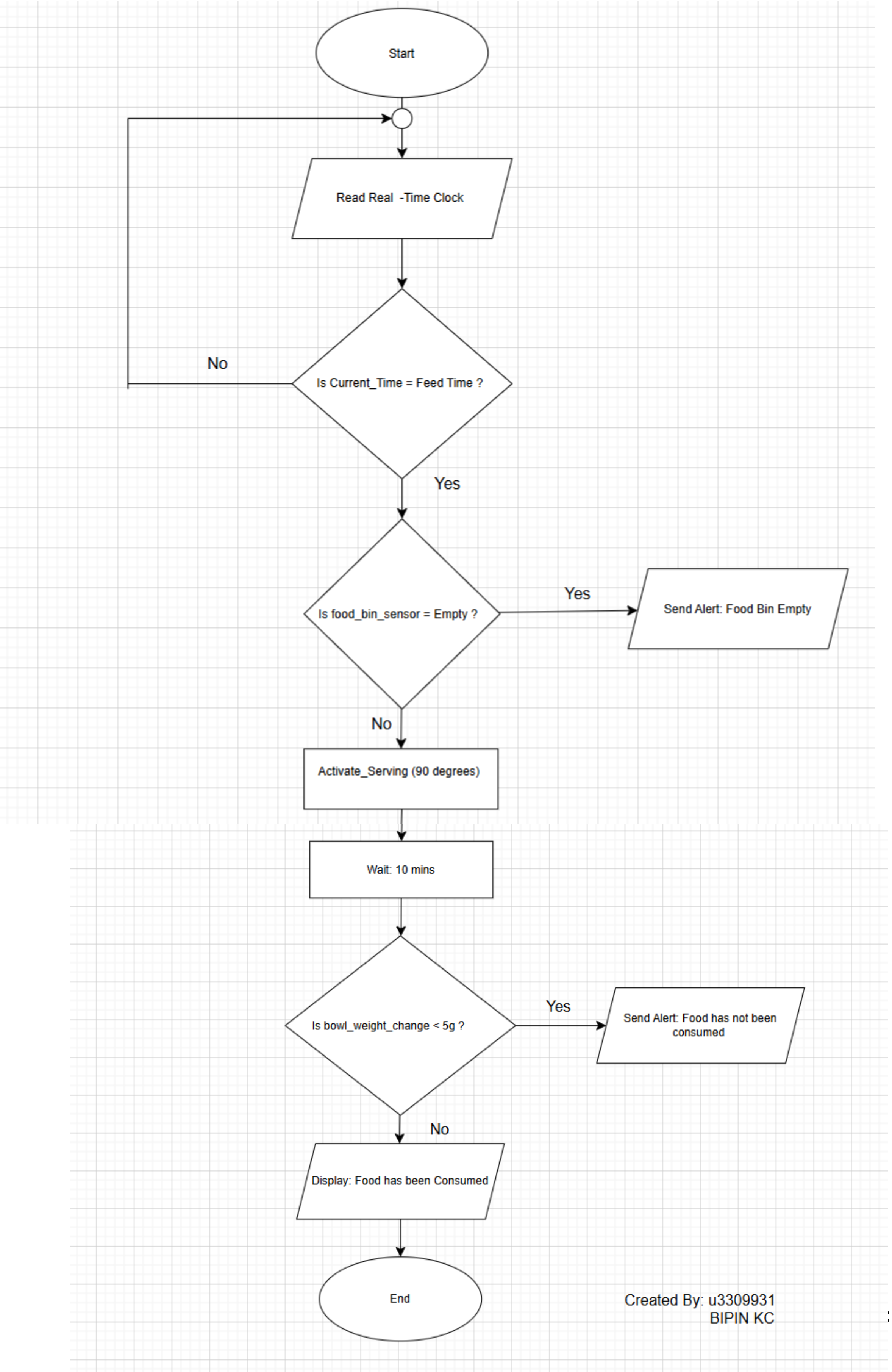
Step 3: Plan the Solution (Design the Algorithm)

PSUEDOCODE:

Assume that pet feeder system sends the alert, if the bowl weight changes to less than 5gm.

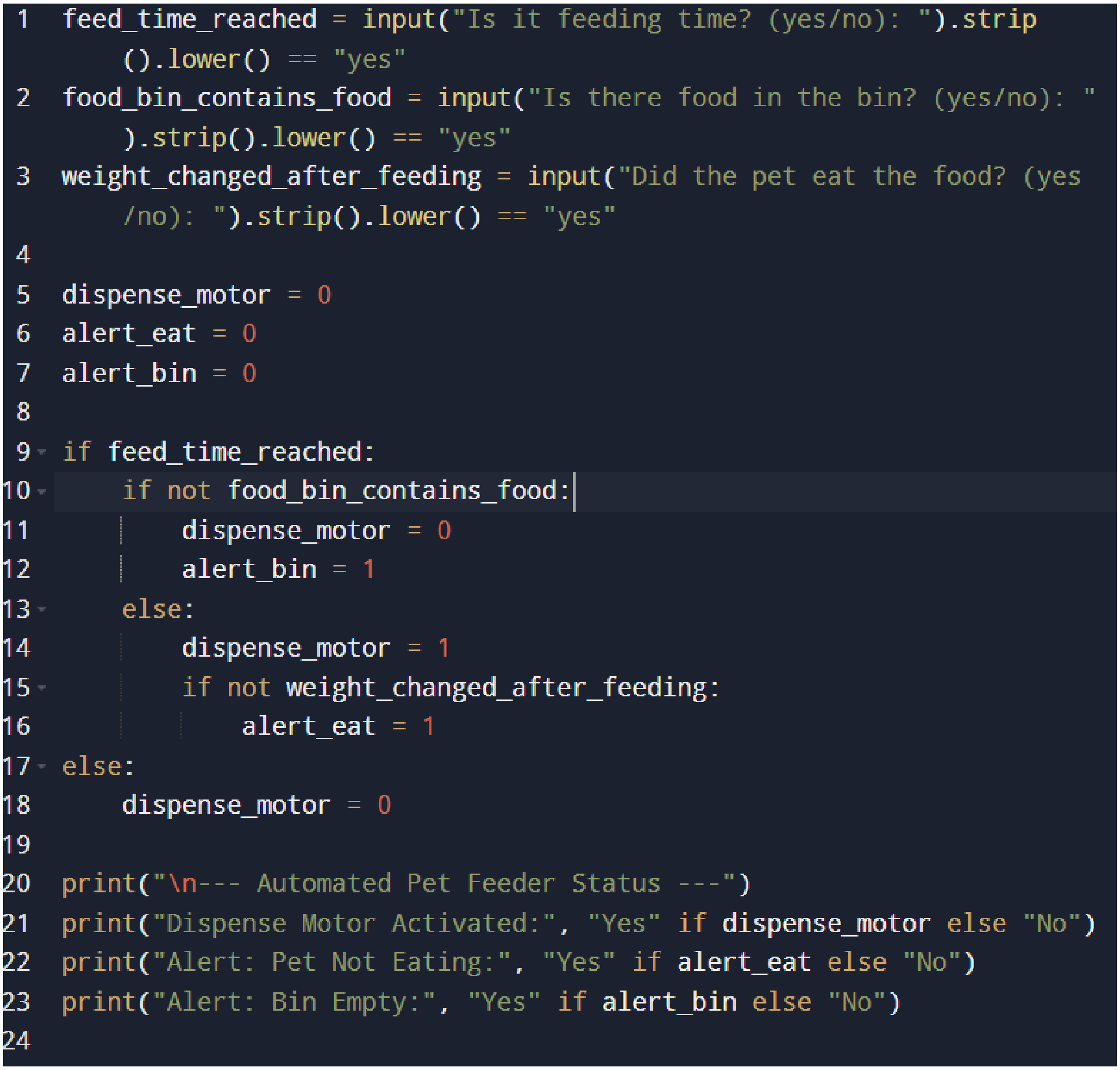
```
BEGIN
SET feed_times = [08:00, 18:00]

LOOP forever:
  current_time = GET_TIME_FROM_RTC()

  IF current_time IN feed_times THEN
    IF food_bin_sensor == EMPTY THEN
      SEND_ALERT("Food bin empty")
    ELSE
      ACTIVATE_SERVING(90 degrees)
      WAIT(10 minutes)
      IF bowl_weight_change < 5g THEN
        SEND_ALERT("Food has not been consumed")
      ELSE
        LOG ("Food consumed" + current_time)
      ENDIF
    ENDIF
  ENDIF
ENDLOOP
END
```

Flowchart:


Start
Read Real -Time Clock
Is Current_Time = Feed Time ?
No
Yes
Is food_bin_sensor = Empty ?
Yes
Send Alert: Food Bin Empty
No
Activate_Serving (90 degrees)
Wait: 10 mins
Is bowl_weight_change < 5g ?
Yes
Send Alert: Food has not been consumed
No
Display: Food has been Consumed
End
Created By: u3309931
BIPIN KC

Step 4: Implement the Solution (Word Coding)

Here, I implemented simple python file to simulate the above scenario.

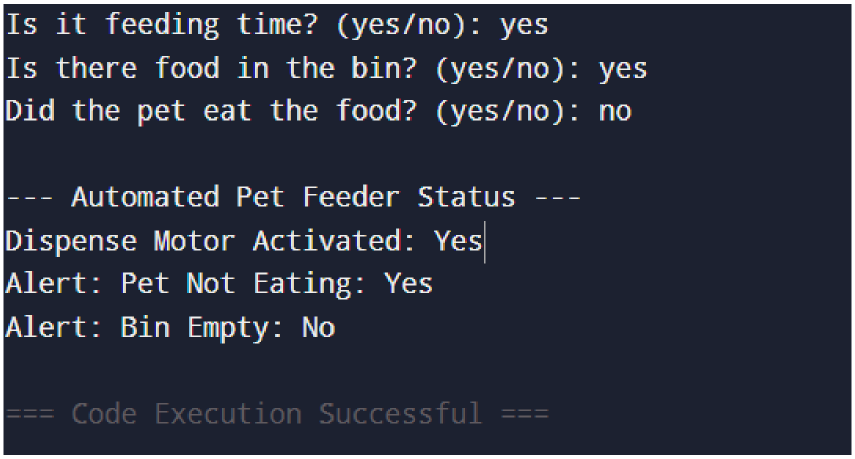


```python
feed_time_reached = input("Is it feeding time? (yes/no): ").strip().lower() == "yes"
food_bin_contains_food = input("Is there food in the bin? (yes/no): ").strip().lower() == "yes"
weight_changed_after_feeding = input("Did the pet eat the food? (yes/no): ").strip().lower() == "yes"

dispense_motor = 0
alert_eat = 0
alert_bin = 0

if feed_time_reached:
    if not food_bin_contains_food:
        dispense_motor = 0
        alert_bin = 1
    else:
        dispense_motor = 1
        if not weight_changed_after_feeding:
            alert_eat = 1
else:
    dispense_motor = 0

print("\n--- Automated Pet Feeder Status ---")
print("Dispense Motor Activated:", "Yes" if dispense_motor else "No")
print("Alert: Pet Not Eating:", "Yes" if alert_eat else "No")
print("Alert: Bin Empty:", "Yes" if alert_bin else "No")

```

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

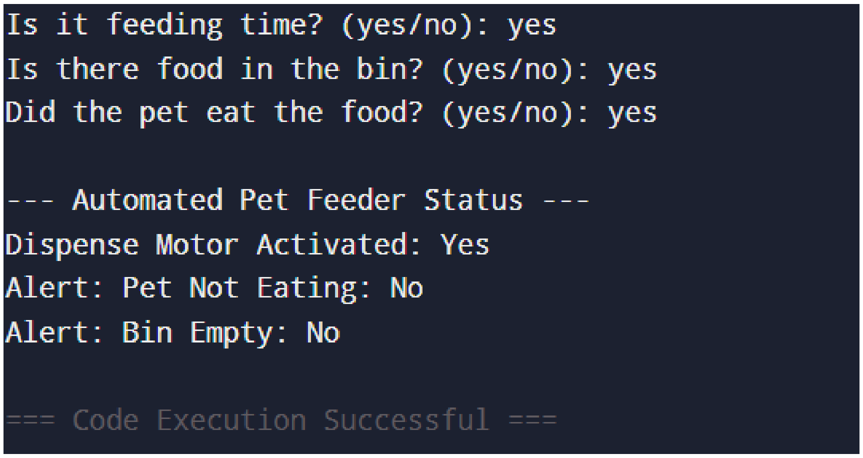
Test Scenarios:

- Pet does not eat:



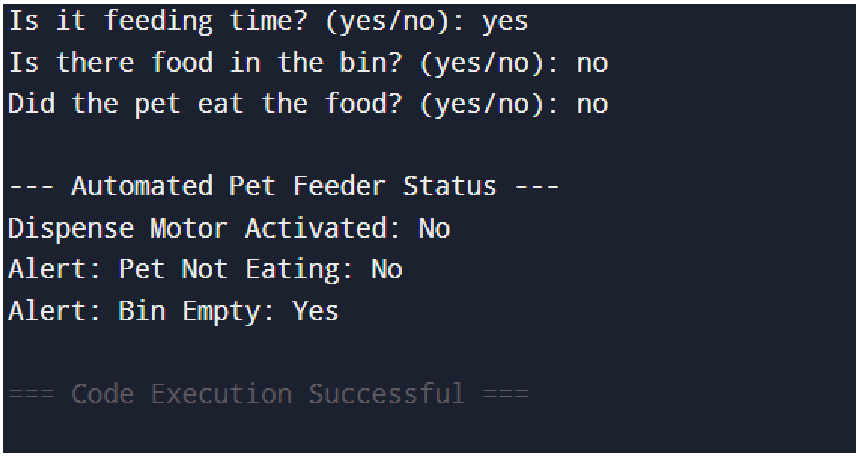
```
Is it feeding time? (yes/no): yes
Is there food in the bin? (yes/no): yes
Did the pet eat the food? (yes/no): no

--- Automated Pet Feeder Status ---
Dispense Motor Activated: Yes
Alert: Pet Not Eating: Yes
Alert: Bin Empty: No

=== Code Execution Successful ===
```

- Pet eats as expected:



```
Is it feeding time? (yes/no): yes
Is there food in the bin? (yes/no): yes
Did the pet eat the food? (yes/no): yes

--- Automated Pet Feeder Status ---
Dispense Motor Activated: Yes
Alert: Pet Not Eating: No
Alert: Bin Empty: No

=== Code Execution Successful ===
```

- Food bin is empty:



```
Is it feeding time? (yes/no): yes
Is there food in the bin? (yes/no): no
Did the pet eat the food? (yes/no): no

--- Automated Pet Feeder Status ---
Dispense Motor Activated: No
Alert: Pet Not Eating: No
Alert: Bin Empty: Yes

=== Code Execution Successful ===
```

Possible Improvements:

- We can develop a mobile app for remote monitoring for the consumption of food.
- It can also further be developed to add a camera and visually verify feeding.
- AI integration for generating reports about feeding habits by pets.

AI AGENT INTEGRATION:

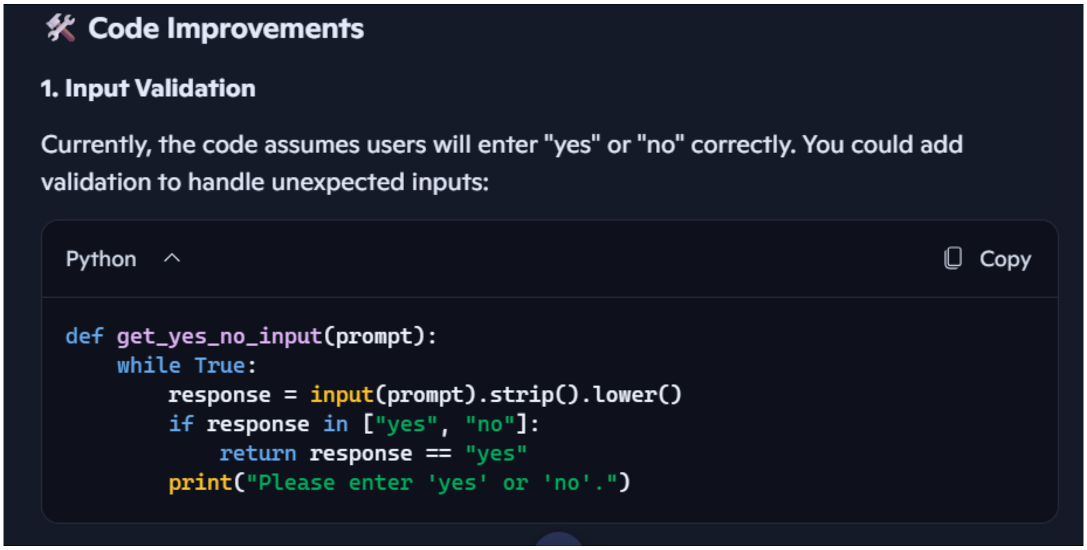
Here, I am using Microsoft Copilot as an AI agent.

I asked Copilot first to review my Step 4 implementation of code and to suggest some improvements.

Prompt Asked:

PS: I provided my code in Step 4.
This is a code for automated pet feeder system. Suggest improvements or identify potential issues.

Response:

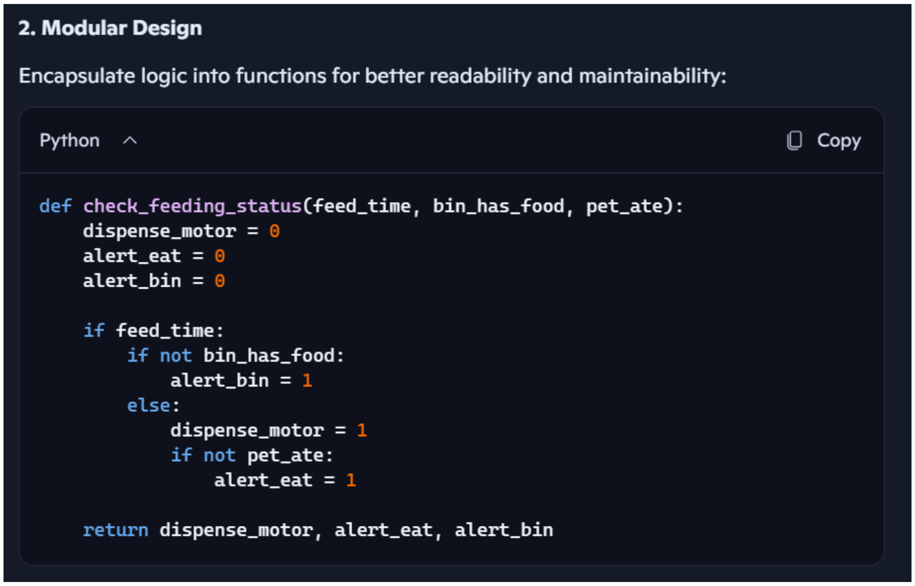


Code Improvements

1. Input Validation

Currently, the code assumes users will enter "yes" or "no" correctly. You could add validation to handle unexpected inputs:

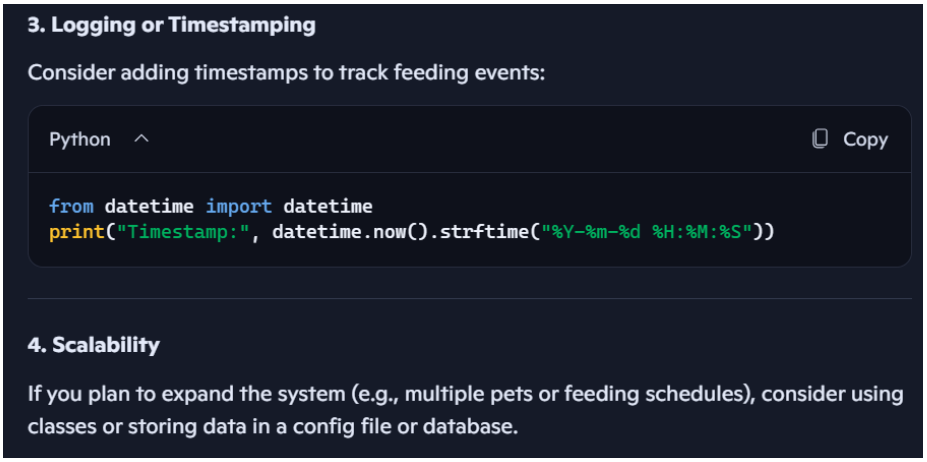
```python
def get_yes_no_input(prompt):
    while True:
        response = input(prompt).strip().lower()
        if response in ["yes", "no"]:
            return response == "yes"
        print("Please enter 'yes' or 'no'.")
```



2. Modular Design

Encapsulate logic into functions for better readability and maintainability:

```python
def check_feeding_status(feed_time, bin_has_food, pet_ate):
    dispense_motor = 0
    alert_eat = 0
    alert_bin = 0

    if feed_time:
        if not bin_has_food:
            alert_bin = 1
        else:
            dispense_motor = 1
            if not pet_ate:
                alert_eat = 1

    return dispense_motor, alert_eat, alert_bin
```



3. Logging or Timestamping

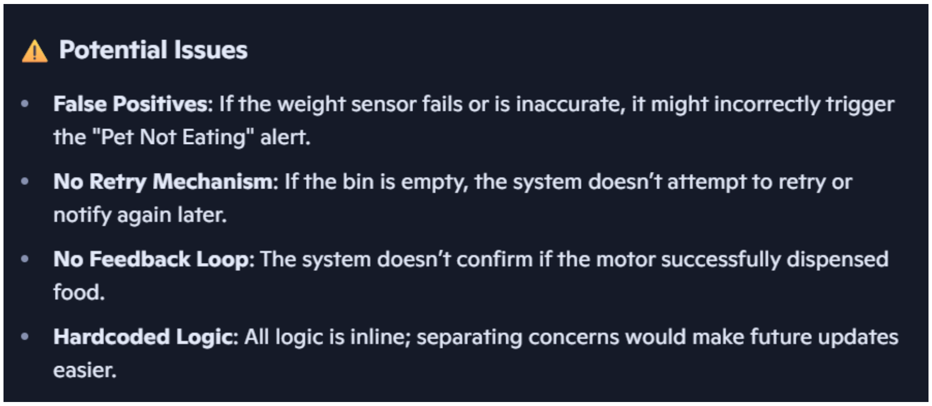
Consider adding timestamps to track feeding events:

```python
from datetime import datetime
print("Timestamp:", datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
```

---

4. Scalability

If you plan to expand the system (e.g., multiple pets or feeding schedules), consider using classes or storing data in a config file or database.



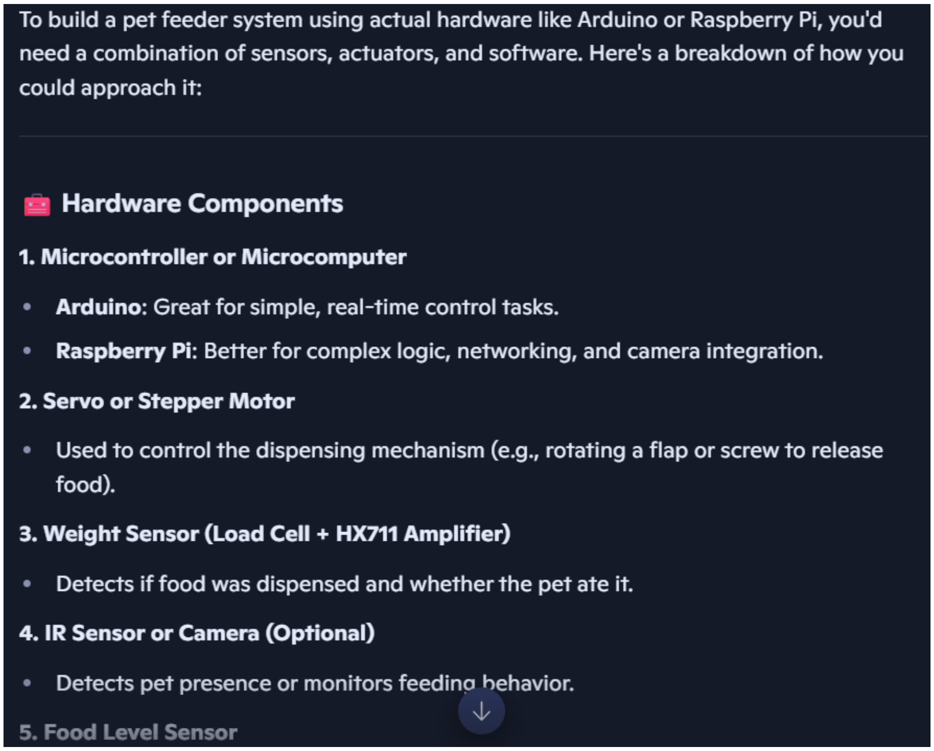
Potential Issues

- **False Positives**: If the weight sensor fails or is inaccurate, it might incorrectly trigger the "Pet Not Eating" alert.
- **No Retry Mechanism**: If the bin is empty, the system doesn't attempt to retry or notify again later.
- **No Feedback Loop**: The system doesn't confirm if the motor successfully dispensed food.
- **Hardcoded Logic**: All logic is inline; separating concerns would make future updates easier.

Prompt Asked:

How can a pet feeder system be built using actual hardware (eg: arduino, raspberry pi)

Response:



To build a pet feeder system using actual hardware like Arduino or Raspberry Pi, you'd need a combination of sensors, actuators, and software. Here's a breakdown of how you could approach it:

🧰 Hardware Components

1. Microcontroller or Microcomputer

- **Arduino**: Great for simple, real-time control tasks.
- **Raspberry Pi**: Better for complex logic, networking, and camera integration.

2. Servo or Stepper Motor

- Used to control the dispensing mechanism (e.g., rotating a flap or screw to release food).

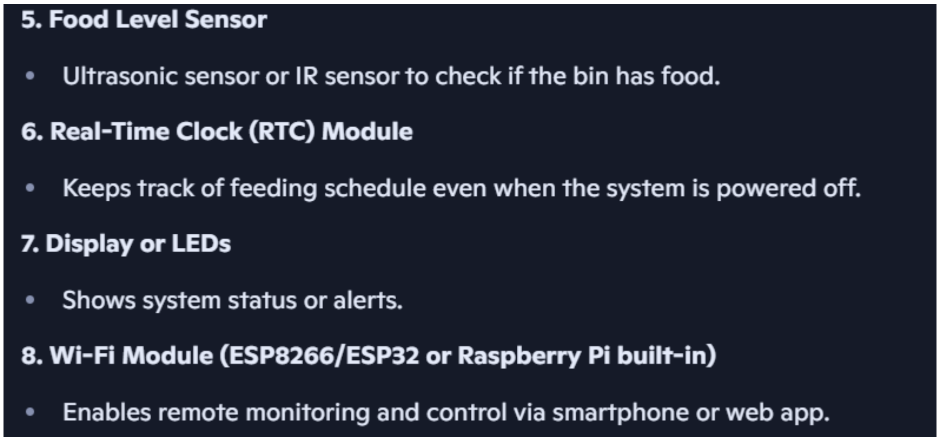
3. Weight Sensor (Load Cell + HX711 Amplifier)

- Detects if food was dispensed and whether the pet ate it.

4. IR Sensor or Camera (Optional)

- Detects pet presence or monitors feeding behavior.

5. Food Level Sensor



5. Food Level Sensor

- Ultrasonic sensor or IR sensor to check if the bin has food.

6. Real-Time Clock (RTC) Module

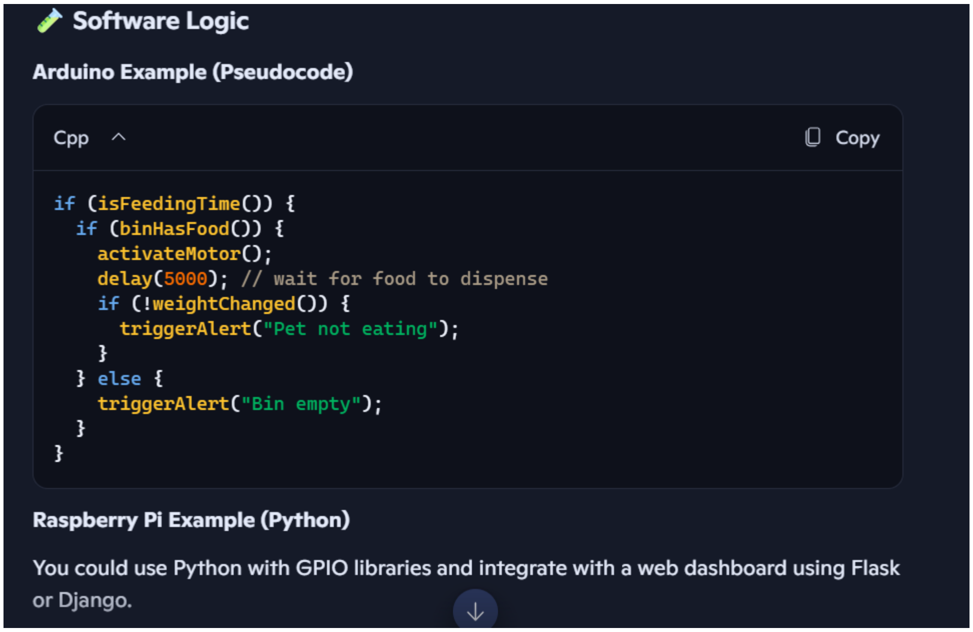
- Keeps track of feeding schedule even when the system is powered off.

7. Display or LEDs

- Shows system status or alerts.

8. Wi-Fi Module (ESP8266/ESP32 or Raspberry Pi built-in)

- Enables remote monitoring and control via smartphone or web app.



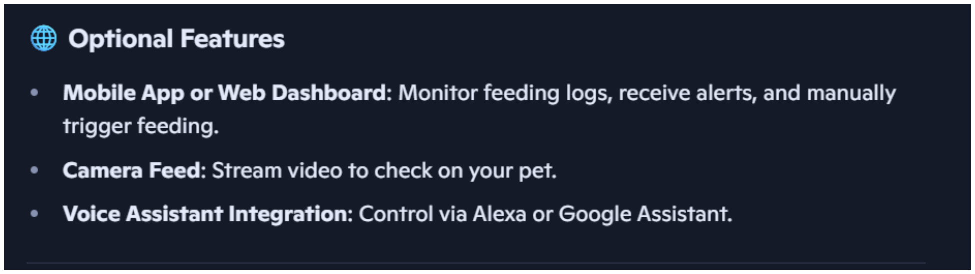
Software Logic

Arduino Example (Pseudocode)

```cpp
if (isFeedingTime()) {
  if (binHasFood()) {
    activateMotor();
    delay(5000); // wait for food to dispense
    if (!weightChanged()) {
      triggerAlert("Pet not eating");
    }
  } else {
    triggerAlert("Bin empty");
  }
}
```

Raspberry Pi Example (Python)

You could use Python with GPIO libraries and integrate with a web dashboard using Flask or Django.



🌐 Optional Features

- **Mobile App or Web Dashboard:** Monitor feeding logs, receive alerts, and manually trigger feeding.
- **Camera Feed:** Stream video to check on your pet.
- **Voice Assistant Integration:** Control via Alexa or Google Assistant.

Conclusion:

Microsoft Copilot is a great tool for providing insights on how to better refine projects. I asked for the above prompts and got valuable feedback about the development of the Automated Pet Feeder Project. Copilot suggested to me that I should focus more on user-friendly coding and use better approach for maintainability. I also suggest that we can incorporate database system to store large amounts of data. Ultimately, it provided me with a guide for how to build a pet feeder system using Arduino as well.

Finally, copilot helped me to understand that there are multiple ways to solve a problem. Also, there is always room for improvement on the code level. It also provided me with a suggestion to add Camera Feed or AI integration to our system as well.