**Assignment - 1 The Solving Problem Process**

**Step-1: Understand and Define the Problem (Analyze)**

**Problem Statement:** The animal shelter requires a low-cost, programmable automated pet feeder to dispense dry and wet food for cats and dogs at scheduled times, monitor consumption, and alert staff if food is not dispensed or remains uneaten. The system should use affordable components (e.g., servo motors, weight sensors) and support manual feeding.

**Features:**

* Dispense dry and wet food at user-defined times (e.g., 8 AM, 6 PM).
* Measure food consumption using a weight sensor under the bowl.
* Alert staff via buzzer/LED if:
* Food fails to dispense (e.g., servo jam).
* Food remains uneaten for 2 hours.
* Log consumption data with timestamps using a real-time clock (RTC).
* Manual feed button for staff intervention.
* Support for one pet per kennel, with scalability.

**Inputs:**

* Current time (from RTC).
* Food bowl weight (grams, from weight sensor).
* Manual feed button (digital: pressed/not pressed).
* Feeding schedule (e.g., 8 AM dry, 6 PM wet)
* Servo status (dispense successful/failed).

**Outputs:**

* Servo motor signals (dispense dry/wet food).
* Buzzer/LED alerts for issues.
* Log entries (e.g., "[timestamp]: Dispensed [food type], [weight]g").

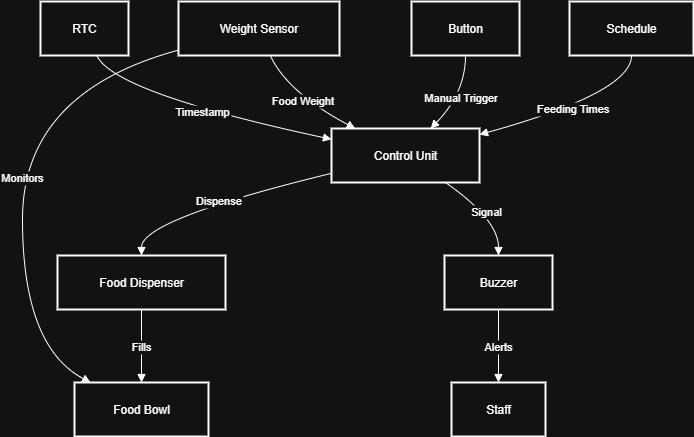
**Assumptions:**

* One bowl per kennel, shared for dry and wet food.
* Pets eat from the bowl within 2 hours.
* Limited memory (logs store 24 hours of data).
* Reliable power supply.

**Limitations:**

* Weight sensor accuracy (±5 grams).
* Servo motor may jam, requiring detection.
* No wireless connectivity (local logging).

**Block Diagram:**

****

**1**Block Diagram

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

**Data Table** (correlating Data, Inputs, Outputs):

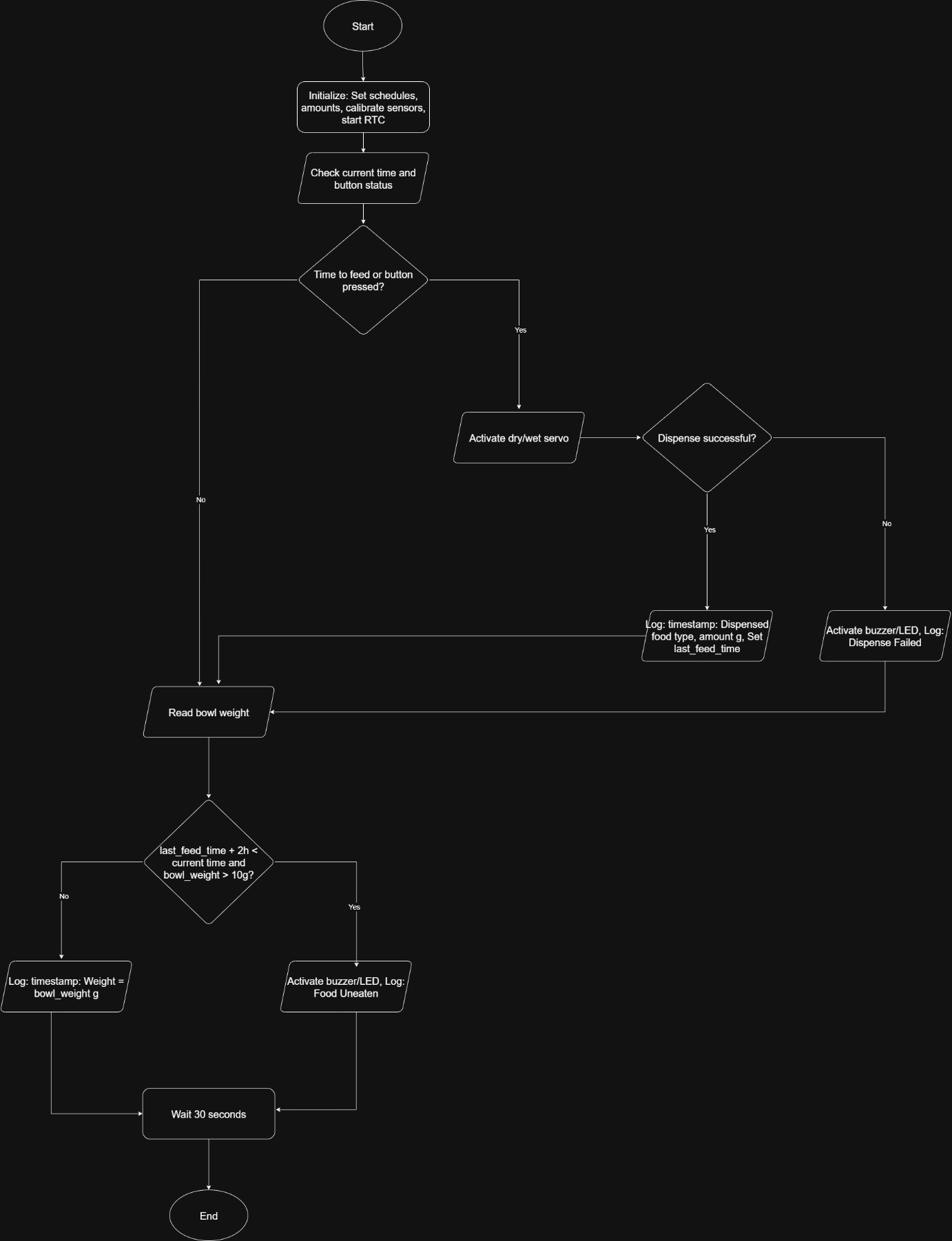
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data** | **Type** | **Example** | **Input Source** | **Output Destination** | **Limits** |
| current\_time | datetime | 2025-08-07 22:07:00 | RTC (I2C signal) | Log (EEPROM) | 24-hour format |
| bowl\_weight | float | 100.0 g | Weight sensor (Analog) | Log (EEPROM) | 0–500 grams |
| button\_status | boolean | True | Button (Digital HIGH/LOW) | Servo motors (PWM) | True/False |
| dry\_schedule | array | ["08:00", "12:00"] | Control Unit (pre-set) | Servo motors (PWM) | Up to 3 times/day |
| wet\_schedule | array | ["18:00"] | Control Unit (pre-set) | Servo motors (PWM) | Up to 2 times/day |
| dry\_amount | float | 100.0 g | Control Unit (pre-set) | Servo motors (PWM) | Fixed amount |
| wet\_amount | float | 150.0 g | Control Unit (pre-set) | Servo motors (PWM) | Fixed amount |
| uneaten\_threshold | integer | 2 hours | Control Unit (pre-set) | Buzzer/LED (Digital HIGH) | In hours |
| last\_feed\_time | datetime | 2025-08-07 08:00:00 | RTC (I2C signal) | Log (EEPROM) | 24-hour format |
| last\_weight | float | 100.0 g | Weight sensor (Analog) | Log (EEPROM) | 0–500 grams |
| log\_entry | string | "[2025-08-07 08:00:00]: Gave Dry, 100g" | Control Unit (generated) | Log (EEPROM) | Short text |

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

**Algorithm:**

1. Set up times, amounts, and limits.
2. Check time or button.
3. Give food if needed.
4. If food doesn’t come, buzz.
5. If food comes, save time.
6. Check weight.
7. If food sits too long, buzz.
8. Save weight.
9. Wait, repeat.

Flowchart:



2Flowchart\_Ass\_1.drawio

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

BEGIN

SET dry\_schedule = ["08:00", "12:00"]

SET wet\_schedule = ["18:00"]

SET dry\_amount = 100 grams

SET wet\_amount = 150 grams

SET uneaten\_threshold = 2 hours

CALIBRATE weight\_sensor

INITIALIZE button

START RTC

SET last\_feed\_time = null

SET last\_weight = read\_weight()

WHILE system is running

# Check if it’s time to feed or button is pressed

SET current\_time = read\_RTC()

SET button\_status = read\_button()

SET is\_feed\_time = (current\_time in dry\_schedule OR current\_time in wet\_schedule)

IF is\_feed\_time OR button\_status = True THEN

# Give the right type of food

IF current\_time in dry\_schedule THEN

SET food\_type = "Dry"

SET amount = dry\_amount

GIVE dry\_food

ELSE IF current\_time in wet\_schedule THEN

SET food\_type = "Wet"

SET amount = wet\_amount

GIVE wet\_food

ELSE

SET food\_type = "Dry (Manual)"

SET amount = dry\_amount

GIVE dry\_food

ENDIF

# Check if food came out

IF servo\_status = Failed THEN

TURN\_ON buzzer\_led

SET log\_entry = "[current\_time]: Food didn't come out"

ELSE

SET log\_entry = "[current\_time]: Gave [food\_type], [amount]g"

SET last\_feed\_time = current\_time

ENDIF

SAVE log\_entry

ENDIF

# Check if food is eaten

SET bowl\_weight = read\_weight()

IF bowl\_weight > 0 THEN

IF last\_feed\_time is not null AND last\_feed\_time + uneaten\_threshold < current\_time AND bowl\_weight > 10 grams THEN

TURN\_ON buzzer\_led

SET log\_entry = "[current\_time]: Food not eaten"

ELSE

SET log\_entry = "[current\_time]: Weight = [bowl\_weight]g"

ENDIF

SAVE log\_entry

SET last\_weight = bowl\_weight

ELSE

SET log\_entry = "[current\_time]: Bad Weight"

SAVE log\_entry

ENDIF

WAIT 30 seconds

ENDWHILE

END

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Test cases** | **Input** | **Output** | **Result** |
| Scheduled Feeding | Time = 08:00, button not pressed, servo successful, weight decreases by 90g | Dispensed Dry, 100g", no alert | Pass |
| Manual Feeding | Button pressed, servo successful, weight decreases by 95g | Dispensed Dry (Manual), 100g", no alert | Pass |
| Dispense Failure | Time = 12:00, servo fails | Buzzer/LED activates, log: "Dispense Failed | Pass |
| Uneaten Food | Weight = 100g, last\_feed\_time = 2.5 hours ago | Buzzer/LED activates, log: "Food Uneaten | Pass |
| Invalid Weight | Weight = -5g | Ignore invalid weight, log error | Refined to validate weight > 0g |

**Refinement:**

* Added weight validation (ignore values ≤ 0g).
* Reduced weight check frequency to every 10 minutes to save power, with 30-second loop delay.

**PART 3: AI Agent Integration**

I used Microsoft Copilot, as required. What I Did:

1. **Simplified Flowchart:**
   * Prompt: “Help me make a very simple flowchart for a pet feeder that’s easy to understand.”
   * Copilot suggested fewer steps, like combining time and button checks, and using clear words like “Give food.”
   * Result: Made flowchart simpler in Step 3.
2. **Refined Word Code:**
   * Prompt: “Check my word code for a pet feeder and suggest fixes: [pasted code].”
   * Copilot suggested checking for bad weights (e.g., negative values).
   * Result: Added bowl\_weight > 0 check in Step 4.

**Github Url for Step-2: https://github.com/parth23704/IIT.git**