## Prompt

**Review this word code and suggest improvements for reliability and modularity.**

// 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

## Response

Thanks for sharing your word-code for the **Automated Pet Feeder**! Here's a detailed review with suggestions to improve **reliability**, **modularity**, and **clarity**:

**✅ Strengths**

* Clear intent: feeding at scheduled times or via manual override.
* Basic error handling: alerts for low food and uneaten food.
* Logical flow: checks before and after dispensing.

**🔧 Suggested Improvements**

**1. Modularity**

Break the logic into reusable functions or modules. This improves readability and makes testing easier.

**Example:**

FUNCTION check\_dispense\_condition():

RETURN get\_current\_time() IN feeding\_times OR check\_manual\_override\_button()

FUNCTION is\_bin\_empty():

RETURN read\_bin\_sensor() == FALSE

FUNCTION has\_pet\_eaten(initial, final):

RETURN (final - initial) &gt;= eat\_threshold

**2. Reliability Enhancements**

* **Time comparison**: get\_current\_time() should be compared to feeding\_times using a proper time format.
* **Manual override logic**: OR True makes the condition always true. This should be corrected.
* **Sensor validation**: Add checks for sensor errors or null values.
* **Weight comparison**: Use >= or > instead of &lt; and &gt; (HTML entities).

**3. Code Logic Fixes**

* dispense\_condition = get\_current\_time() AND check\_manual\_override\_button() is incorrect. It mixes time and boolean.
  + **Fix**: Separate time check and manual override.
* success = initial\_weight < target\_weight — target\_weight is undefined.
  + **Fix**: Define target\_weight or clarify its source.
* IF success == FALSE: — consider using IF NOT success: for clarity.
* Skip feed AND log event — should be a function or structured action.

**4. Logging and Alerts**

Use consistent logging and alerting mechanisms.

**Example:**

FUNCTION log\_event(event\_type, timestamp):

// Save to system log

FUNCTION send\_alert(message):

// Notify user via app or email

**5. Suggested Structure**

Here’s a modular and reliable version outline:

// Variables

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

dispense\_duration = 5

eat\_threshold = 1.0

target\_weight = 50.0

FUNCTION main\_loop():

WHILE system\_on:

IF should\_dispense():

IF is\_bin\_empty():

send\_alert("Low Food Alert")

ELSE:

IF dispense\_food():

log\_event("Food Dispensed", current\_time())

IF NOT pet\_ate\_food():

send\_alert("NOT EATEN")

ELSE:

log\_event("Dispense Failed", current\_time())

send\_alert("No Food Dispensed")

FUNCTION should\_dispense():

RETURN current\_time() IN feeding\_times OR check\_manual\_override\_button()

FUNCTION dispense\_food():

initial = read\_bowl\_weight()

wait(dispense\_duration)

final = read\_bowl\_weight()

RETURN final &gt; initial

FUNCTION pet\_ate\_food():

wait(600) // 10 minutes

after = read\_bowl\_weight()

RETURN has\_pet\_eaten(final, after)

Would you like me to convert this into a specific programming language (e.g., Python, Arduino C)? Or help you simulate/test it?