**Step 4: Implement the solution in plain English code**

**FUNCTION InitSystem**

schedule ← [08:00, 12:00, 16:00, 20:00] // up to 4 feeds/day

portion\_grams ← 150 // adjustable

low\_storage\_threshold ← 20 // percent

wait\_minutes ← 10 // grace period

fed\_today ← map<time,bool> default false // prevents duplicates

END FUNCTION

**FUNCTION ReadSensors**

storage\_percent = read\_storage\_sensor() // 0–100%

bowl\_weight\_before = read\_bowl\_weight() // grams

time\_now = read\_clock()

RETURN (storage\_percent, bowl\_weight\_before, time\_now)

END FUNCTION

**FUNCTION ShouldFeedNow(time\_now)**

RETURN (time\_now in schedule) AND (fed\_today[time\_now] = false)

END FUNCTION

**FUNCTION Dispense(portion\_grams)**

activate\_servo\_for(portion\_grams) // motor run based on grams

LOG("Dispensed " + portion\_grams + " g at " + time\_now)

END FUNCTION

**FUNCTION PostFeedCheck(wait\_minutes,/bowl\_weight\_before)**

wait(wait\_minutes) // timer

bowl\_weight\_after = read\_bowl\_weight()

IF bowl\_weight\_after < bowl\_weight\_before THEN

LOG("Food eaten within " + wait\_minutes + " min")

ELSE

alert("Uneaten food after " + wait\_minutes + " min")

END IF

END FUNCTION

**FUNCTION UpdateDisplay**

show("Food: " + storage\_percent + “Next: " + next\_schedule\_time())

END FUNCTION

// ---------- Main loop ----------

InitSystem()

LOOP every minute

(storage\_percent, bowl\_before, time\_now) ← ReadSensors()

IF ShouldFeedNow(time\_now) THEN

IF storage\_percent < low\_storage\_threshold THEN

alert("Low food (" + storage\_percent +”). Skipping " + time\_now)

UpdateDisplay()

ELSE

Dispense(portion\_grams)

fed\_today[time\_now] ← true

PostFeedCheck(wait\_minutes, bowl\_before

UpdateDisplay()

END IF

IF is\_midnight(time\_now) THEN

reset(fed\_today) // new day, all slots false

END