ANSU MARIUM SHIBU

WEEKEND ASSIGNMENTS 1

1. Smart Home Temperature Control Problem
   Statement: Design a temperature control system for a smart home. The system should read the current temperature from a sensor every minute and compare it to a user-defined setpoint. Requirements: • If the current temperature is above the setpoint, activate the cooling system. • If the current temperature is below the setpoint, activate the heating system. • Display the current temperature and setpoint on an LCD screen. • Include error handling for sensor failures.

   PSEUDOCODE:

   START
   1. SET setpoint = GetUserInput("Enter desired temp")

   WHILE (true)
   a. READ current_temp from temp_sensor

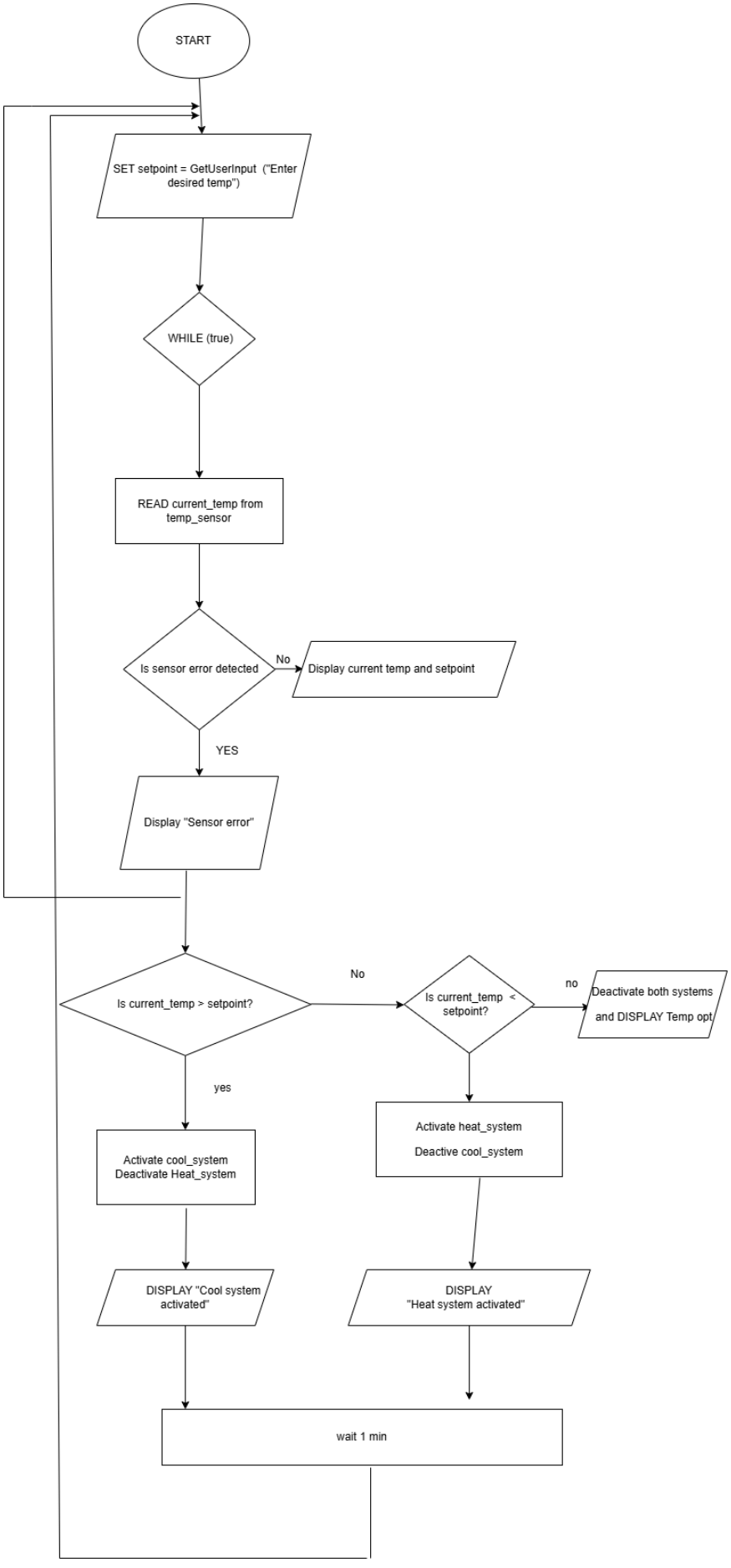
   b. IF (sensor_error)
   DISPLAY "Sensor error"
   CONTINUE to next iteration

   c. DISPLAY "Current Temp: " + current_temp + " Setpoint: " + setpoint on LCD

   d. IF (current_temp > setpoint) THEN
   ACTIVATE cool_system
   DEACTIVATE heat_system
   DISPLAY "Cool system activated" on LCD
   ELSE IF (current_temp < setpoint) THEN
   ACTIVATE heat_system
   DEACTIVATE cool_system
   DISPLAY "Heat system activated" on LCD
   ELSE

```
            DEACTIVATE both cool_system and heat_system
            DISPLAY "Temp  optimal" on LCD
        ENDIF

        e. WAIT 1 minute
    END WHILE

END
```

FLOWCHART:


START
SET setpoint = GetUserInput ("Enter desired temp")
WHILE (true)
READ current_temp from temp_sensor
Is sensor error detected
No
Display current temp and setpoint
YES
Display "Sensor error"
Is current_temp > setpoint?
No
Is current_temp < setpoint?
no
Deactivate both systems and DISPLAY Temp opt
yes
Activate cool_system
Deactivate Heat_system
Activate heat_system
Deactive cool_system
DISPLAY "Cool system activated"
DISPLAY "Heat system activated"
wait 1 min

2. Automated Plant Watering System Problem Statement: Create an automated watering system for plants that checks soil moisture levels and waters the plants accordingly. Requirements: • Read soil moisture level from a sensor every hour. • If moisture level is below a defined threshold, activate the water pump for a specified duration. • Log the watering events with timestamps to an SD card. • Provide feedback through an LED indicator (e.g., LED ON when watering).

PSEUDOCODE:

START

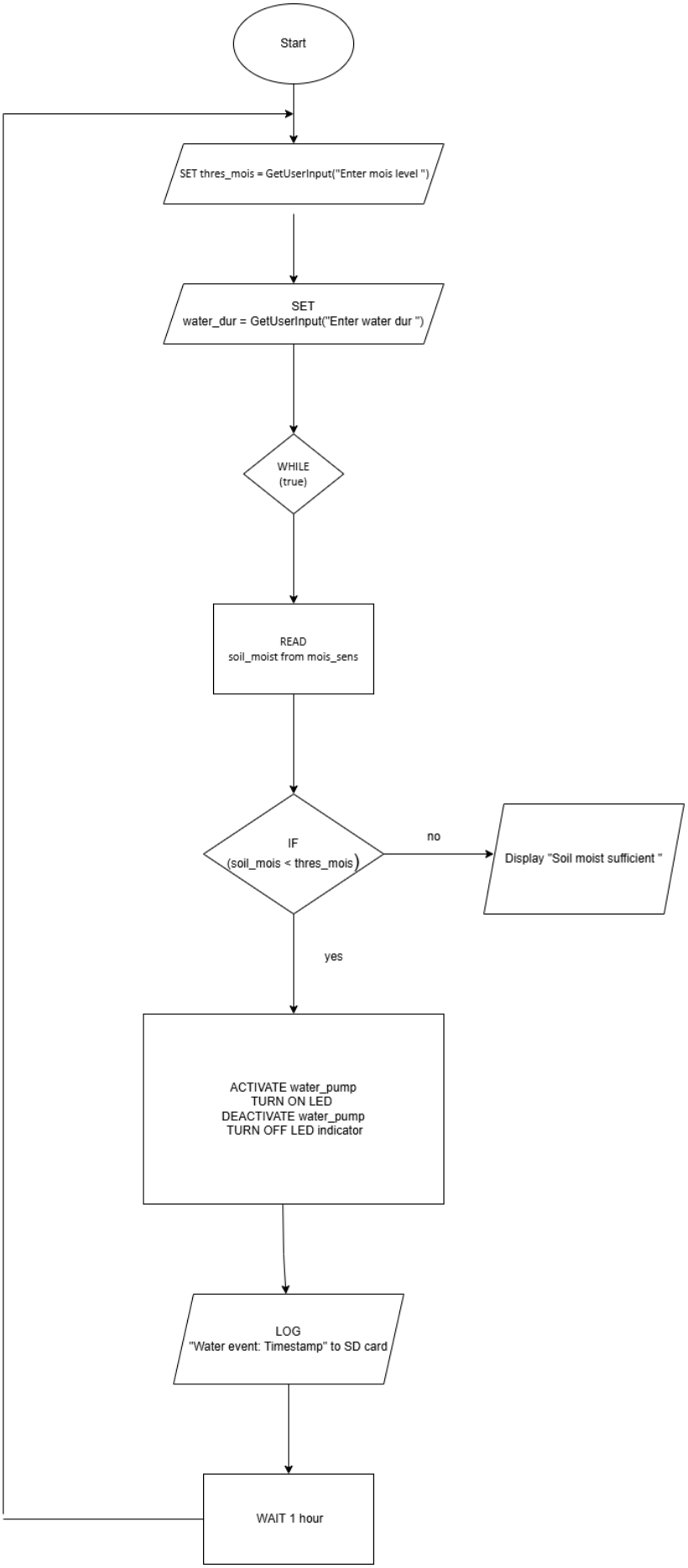
1. SET thres_mois = GetUserInput("Enter mois level ")
2. SET water_dur = GetUserInput("Enter water dur ")

WHILE (true)
a. READ soil_moist from mois_sens

b. IF (soil_mois < thres_mois)
i. ACTIVATE water_pump
ii. TURN ON LED
iii. WAIT for water_dur
iv. DEACTIVATE water_pump
v. TURN OFF LED indicator
vi. LOG "Water event: Timestamp" to SD card
ELSE
i. DISPLAY "Soil moist is sufficient" on LCD

c. WAIT 1 hour
END WHILE
END

FLOWCHART:

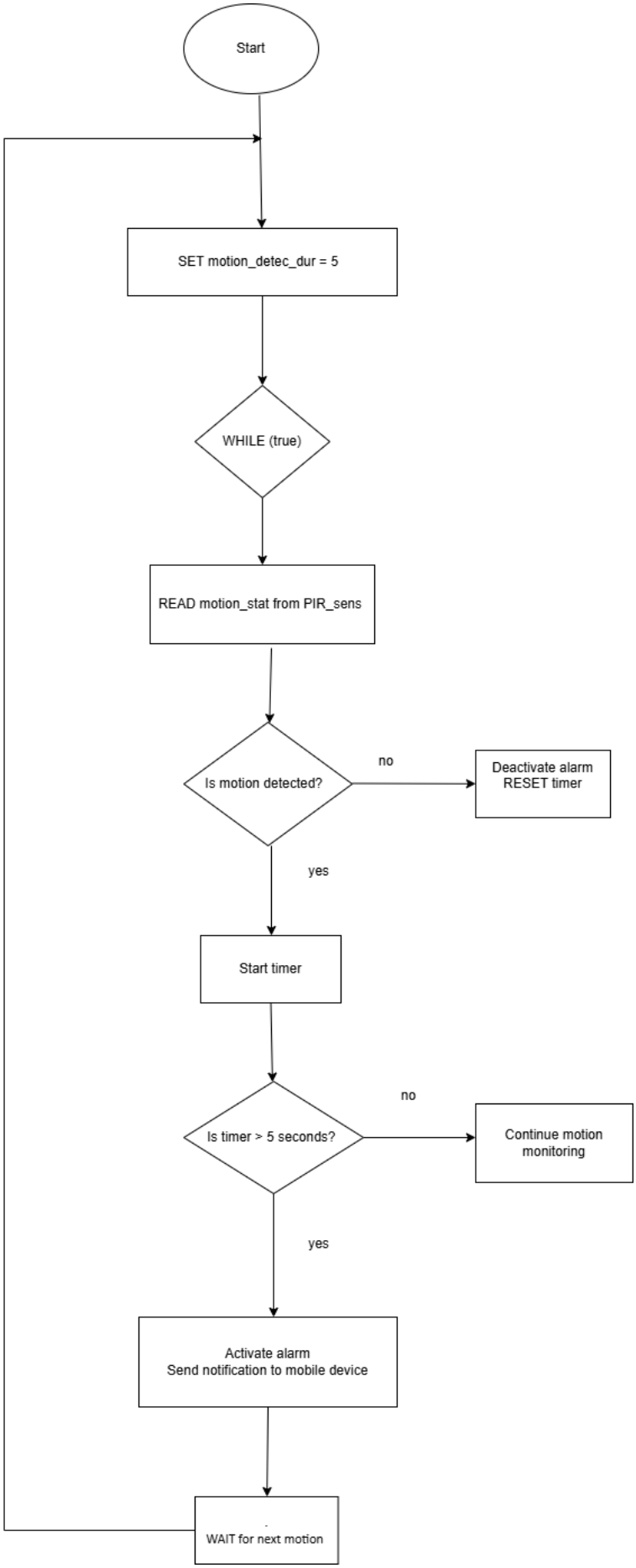

Start
SET thres_mois = GetUserInput("Enter mois level ")
SET
water_dur = GetUserInput("Enter water dur ")
WHILE
(true)
READ
soil_moist from mois_sens
IF
(soil_mois < thres_mois)
no
Display "Soil moist sufficient "
yes
ACTIVATE water_pump
TURN ON LED
DEACTIVATE water_pump
TURN OFF LED indicator
LOG
"Water event: Timestamp" to SD card
WAIT 1 hour

3. Motion Detection Alarm System
Problem Statement:
Develop a security alarm system that detects motion using a PIR sensor.
Requirements:
• Continuously monitor motion detection status.
• If motion is detected for more than 5 seconds, trigger an alarm (buzzer).
• Send a notification to a mobile device via UART communication.
• Include a reset mechanism to deactivate the alarm.

PSEUDOCODE:

```
START
   1. SET motion_detect_dur = 5

   WHILE (true)
      a. READ motion_stat from PIR_sens

      b. IF motion_stat = detect
         i. START timer
         ii. IF timer > motion_detect_dur
            - ACTIVATE alarm
            - SEND notification to mobile device via UART communication
         iii. ELSE
            - CONTINUE monitoring motion

      c. IF motion_stat = not_detected
         i. DEACTIVATE alarm if activated
         ii. RESET timer

      d. WAIT for next motion
   END WHILE
END
```

FLOWCHART:


Start
SET motion_detec_dur = 5
WHILE (true)
READ motion_stat from PIR_sens
Is motion detected?
no
Deactivate alarm
RESET timer
yes
Start timer
Is timer > 5 seconds?
no
Continue motion
monitoring
yes
Activate alarm
Send notification to mobile device
WAIT for next motion

4.Heart Rate Monitor Problem Statement: Implement a heart rate monitoring application that reads data from a heart rate sensor. Requirements: • Sample heart rate data every second and calculate the average heart rate over one minute. • If the heart rate exceeds 100 beats per minute, trigger an alert (buzzer). • Display current heart rate and average heart rate on an LCD screen. • Log heart rate data to an SD card for later analysis.

PSEUDOCODE:

START

1. SET threshold_heart_rate = 100
2. SET heart_rate_data[] as an empty list
3. SET sample_interval = 1 second
4. SET total_samples = 60

WHILE (true)

a. READ current_heart_rate from heart_rate_sensor

b. ADD current_heart_rate to heart_rate_data[]

c. DISPLAY current_heart_rate on LCD

d. IF current_heart_rate > threshold_heart_rate

i. ACTIVATE alert

ii. DISPLAY "ALERT: High heart rate" on LCD

e. IF number of samples collected >= total_samples

i. CALCULATE average_heart_rate = SUM(heart_rate_data[]) / total_samples

ii. DISPLAY average_heart_rate on LCD

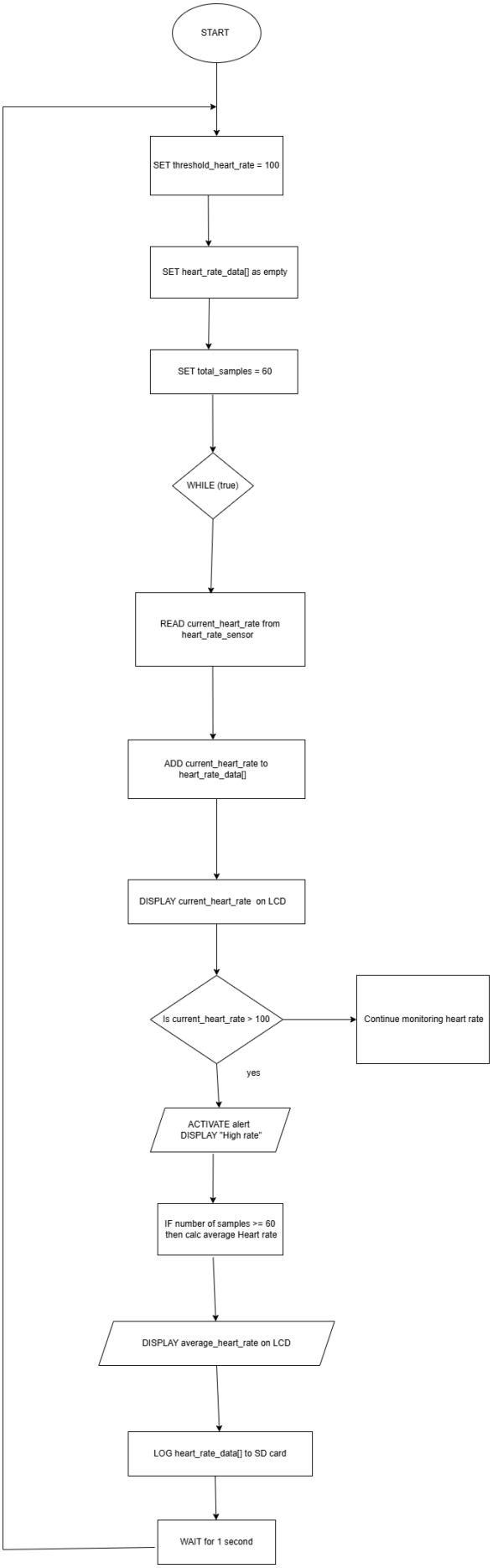
iii. LOG heart_rate_data[]on SD card

f. WAIT for 1 second

END WHILE

END

FLOWCHART:


START
SET threshold_heart_rate = 100
SET heart_rate_data[] as empty
SET total_samples = 60
WHILE (true)
READ current_heart_rate from heart_rate_sensor
ADD current_heart_rate to heart_rate_data[]
DISPLAY current_heart_rate on LCD
Is current_heart_rate > 100
Continue monitoring heart rate
yes
ACTIVATE alert
DISPLAY "High rate"
IF number of samples >= 60 then calc average Heart rate
DISPLAY average_heart_rate on LCD
LOG heart_rate_data[] to SD card
WAIT for 1 second

5. LED Control Based on Light Sensor Problem Statement: Create an embedded application that controls an LED based on ambient light levels detected by a light sensor. Requirements: • Read light intensity from the sensor every minute. • If light intensity is below a certain threshold, turn ON the LED; otherwise, turn it OFF. • Include a manual override switch that allows users to control the LED regardless of sensor input. • Provide status feedback through another LED (e.g., blinking when in manual mode).

PSEUDOCODE:

START

1. SET light_thre = 500
2. SET light_sens_pin to read light inte
3. SET led_pin to control LED
4. SET override_pin to control override switch
5. SET feedback_pin to provide feedback

WHILE (true)

a. READ light_inten from light_sens_pin

b. IF override_switch is ON (override_pin is HIGH)

i. TURN ON main LED (led_pin) if override switch is ON

ii. BLINK feedback LED (feedback_pin) to indicate manual mode

iii. DISPLAY "Manual Override Mode

iv. CONTINUE to next iteration

c. IF light_intensity < light_thres

i. TURN ON main LED (led_pin)

ii. TURN OFF feedback LED

iii. DISPLAY "Low Light: LED ON"

ELSE

i. TURN OFF main LED

ii. TURN OFF feedback LED

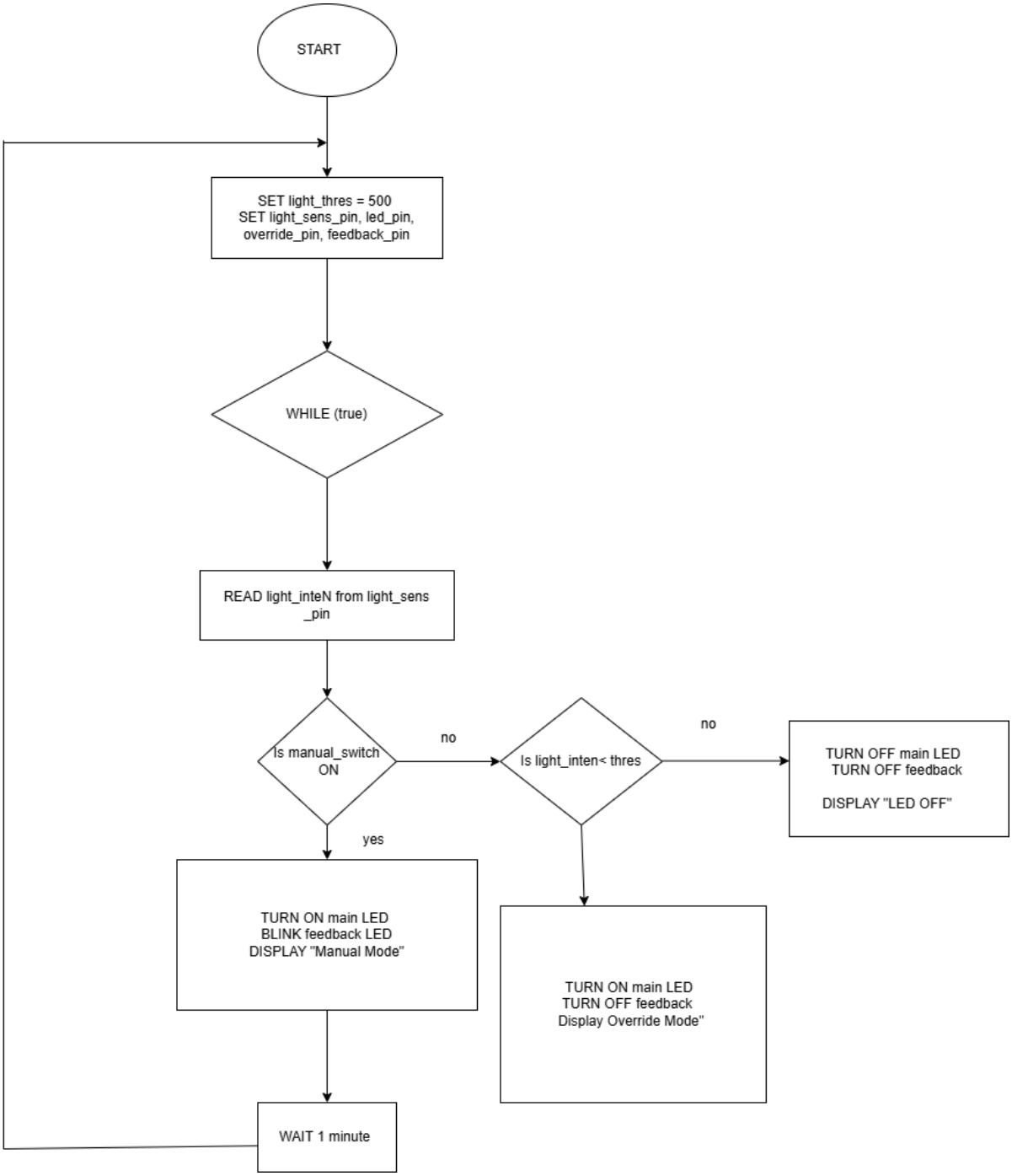
iii. DISPLAY " LED OFF"

d. WAIT 1 minute

END WHILE

END

FLOWCHART:



6. Digital Stopwatch Problem Statement: Design a digital stopwatch application that can start, stop, and reset using button inputs. Requirements: • Use buttons for Start, Stop, and Reset functionalities. • Display elapsed time on an LCD screen in hours, minutes, and seconds format. • Include functionality to pause and resume timing without resetting. • Log start and stop times to an SD card when stopped.

PSEUDOCODE:

START

1. SET elaps_time = 0
2. SET stopwatch_state = "paused"
3. SET start_time = 0
4. SET stop_time = 0

5. WAIT for user input

WHILE (true)

a. READ button inputs

b. IF Start button pressed

i. IF stopwatch is pause

- SET start_time = current_time
- SET stopwatch_state = "running"
- DISPLAY "Started" on LCD

c. IF Stop button pressed

i. IF stopwatch is running

- SET stop_time = current_time
- SET stopwatch_state = "paused"
- DISPLAY "Stopped" on LCD
- Log start_time and stop_time to SD card

d. IF Reset button pressed

i. RESET elaps_time, start_time, and stop_time to 0

ii. SET stopwatch_state = "paused"

iii. DISPLAY "Reset" on LCD

e. IF stopwatch_state = "running"

i. UPDATE elaps_time = current_time - start_time

ii. DISPLAY elaps_time on LCD

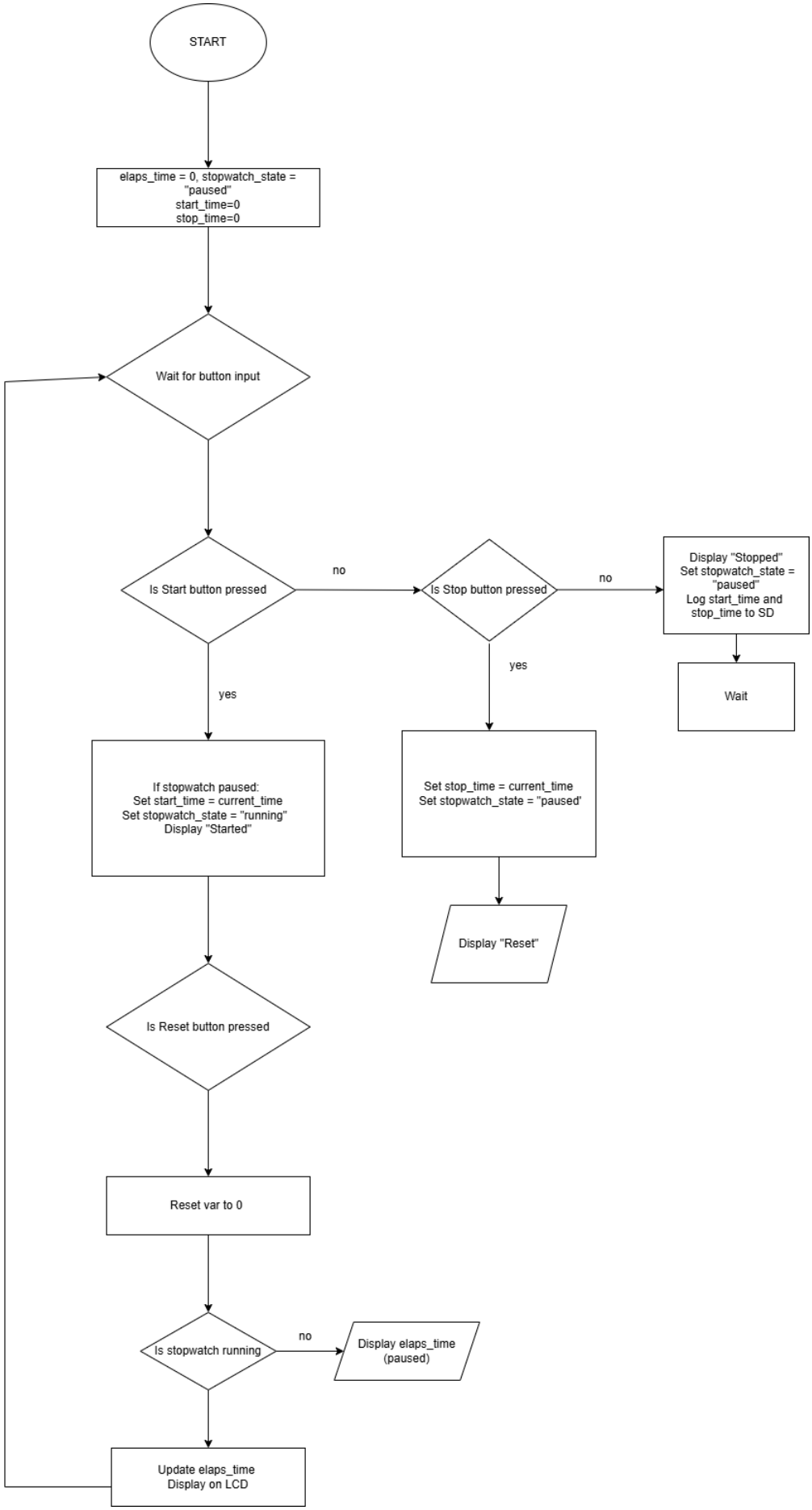
f. IF stopwatch_state = "paused"

i. DISPLAY elaps_time on LCD

g. WAIT

END

FLOWCHART:


START
elaps_time = 0, stopwatch_state = "paused"
start_time=0
stop_time=0
Wait for button input
Is Start button pressed
no
Is Stop button pressed
no
Display "Stopped"
Set stopwatch_state = "paused"
Log start_time and stop_time to SD
Wait
yes
yes
If stopwatch paused:
Set start_time = current_time
Set stopwatch_state = "running"
Display "Started"
Set stop_time = current_time
Set stopwatch_state = "paused'
Display "Reset"
Is Reset button pressed
Reset var to 0
Is stopwatch running
no
Display elaps_time (paused)
Update elaps_time
Display on LCD

7. Temperature Logging System Problem Statement: Implement a temperature logging system that records temperature data at regular intervals. Requirements: • Read temperature from a sensor every 10 minutes. • Store each reading along with its timestamp in an array or log file. • Provide functionality to retrieve and display historical data upon request. • Include error handling for sensor read failures.

PSEUDOCODE:

START

1. Initialize an empty array

2. Set interval to 10 minutes

WHILE (true)

a. READ temp from temp_sens

b. IF (sens_error_detecT)

DISPLAY "Sensor error"

CONTINUE to next iteration

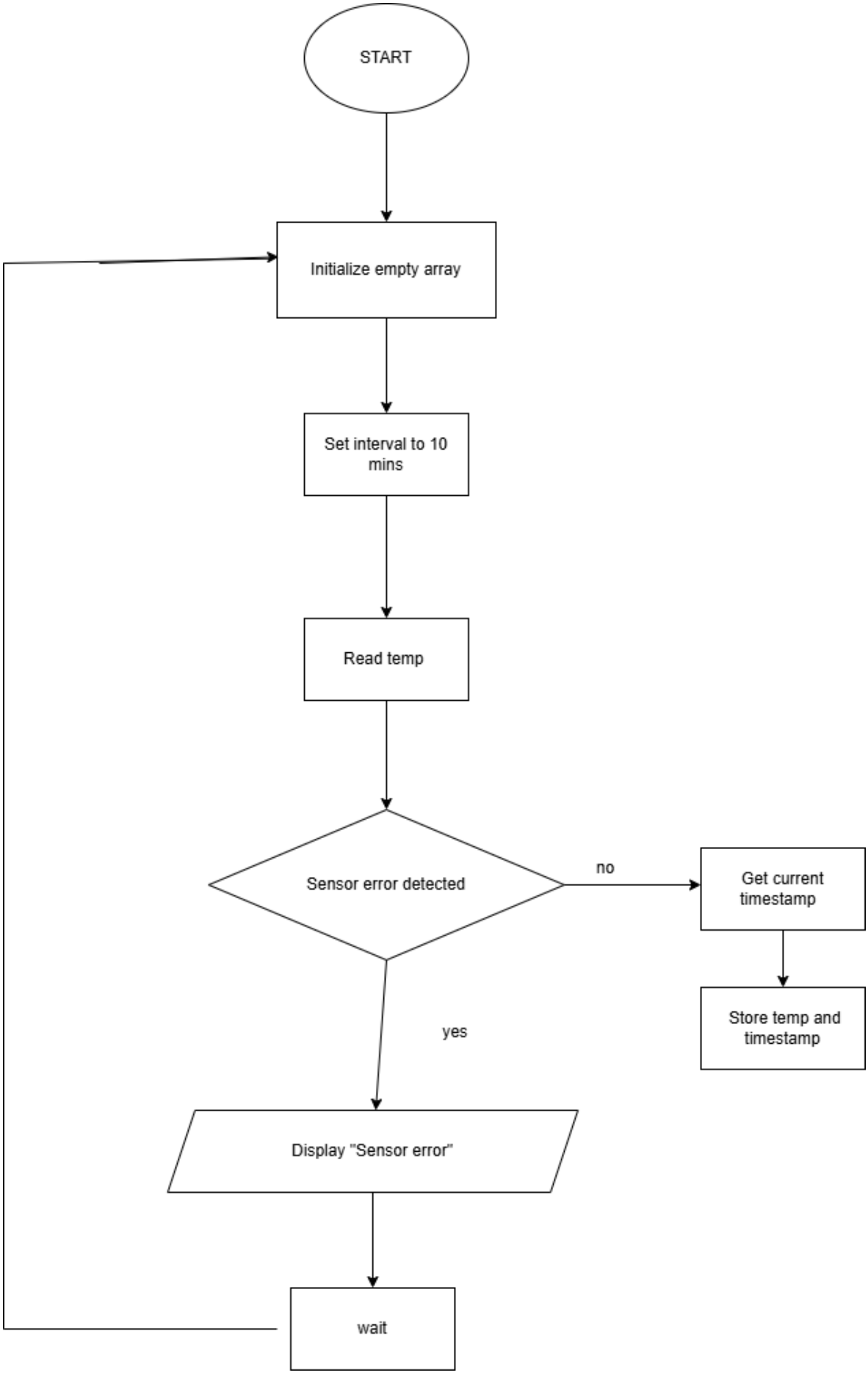
c. GET current_timestamp

d. STORE temp and current_timestamp in log

e. WAIT

END WHILE

FLOWCHART:


START
Initialize empty array
Set interval to 10 mins
Read temp
Sensor error detected
no
Get current timestamp
Store temp and timestamp
yes
Display "Sensor error"
wait

8. Bluetooth Controlled Robot Problem Statement: Create an embedded application for controlling a robot via Bluetooth commands. Requirements: • Establish Bluetooth communication with a mobile device. • Implement commands for moving forward, backward, left, and right. • Include speed control functionality based on received commands. • Provide feedback through LEDs indicating the current state (e.g., moving or stopped).

PSEUDOCODE:

START

1. Initialize Bluetooth mod
2. Initialize LEDs for feedback

WHILE (true)

a. READ command from Bluetooth

b. IF (command == "FORWARD")

SET direction to forward

SET speed according to command

TURN ON "moving" LED

c. ELSE IF (command == "BACKWARD")

SET direction to backward

SET speed according to command

TURN ON "moving" LED

d. ELSE IF (command == "LEFT")

SET direction to turn left

SET speed according to command

TURN ON "moving" LED

e. ELSE IF (command == "RIGHT")

SET direction to turn right

SET speed according to command

TURN ON "moving" LED

f. ELSE IF (command == "STOP")

STOP

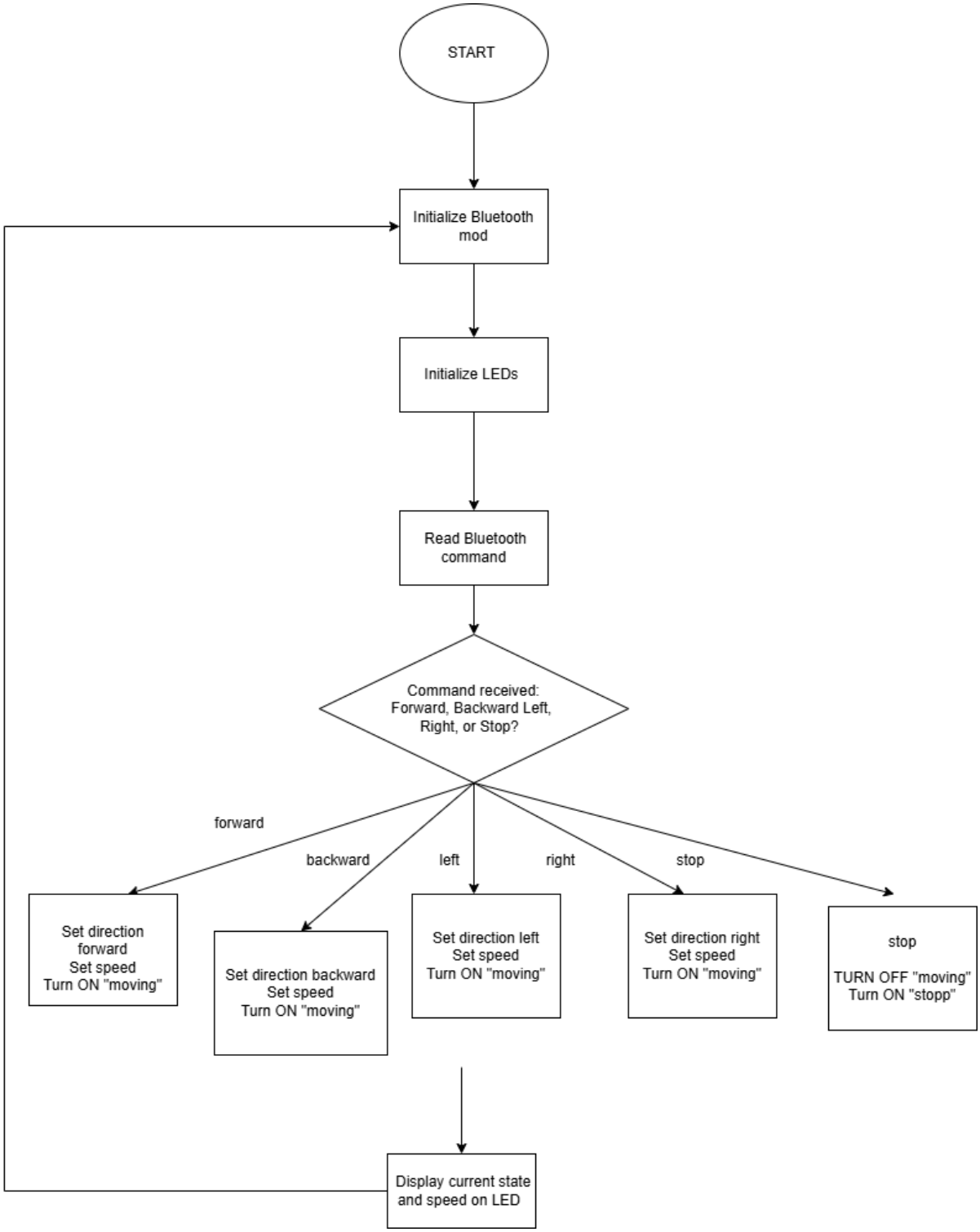
TURN OFF "moving" LED

TURN ON "stopp" LED

g. DISPLAY current state and speed on LED

END WHILE

FLOWCHART:



9. Battery Monitoring System Problem Statement: Develop a battery monitoring system that checks battery voltage levels periodically and alerts if voltage drops below a safe threshold. Requirements: • Measure battery voltage every minute using an ADC (Analog-to-Digital Converter). • If voltage falls

below 11V, trigger an alert (buzzer) and log the event to memory. • Display current voltage on an LCD screen continuously. • Implement power-saving features to reduce energy consumption during idle periods.

PSEUDOCODE:

START

1. Initialize ADC, LCD, and buzzer

LOOP (every minute):

a. READ battery_voltage using ADC

b. DISPLAY battery_voltage on LCD

c. IF (battery_voltage < 11V) THEN

ACTIVATE buzzer

LOG "Low voltage event"

ELSE

DEACTIVATE buzzer

ENDIF

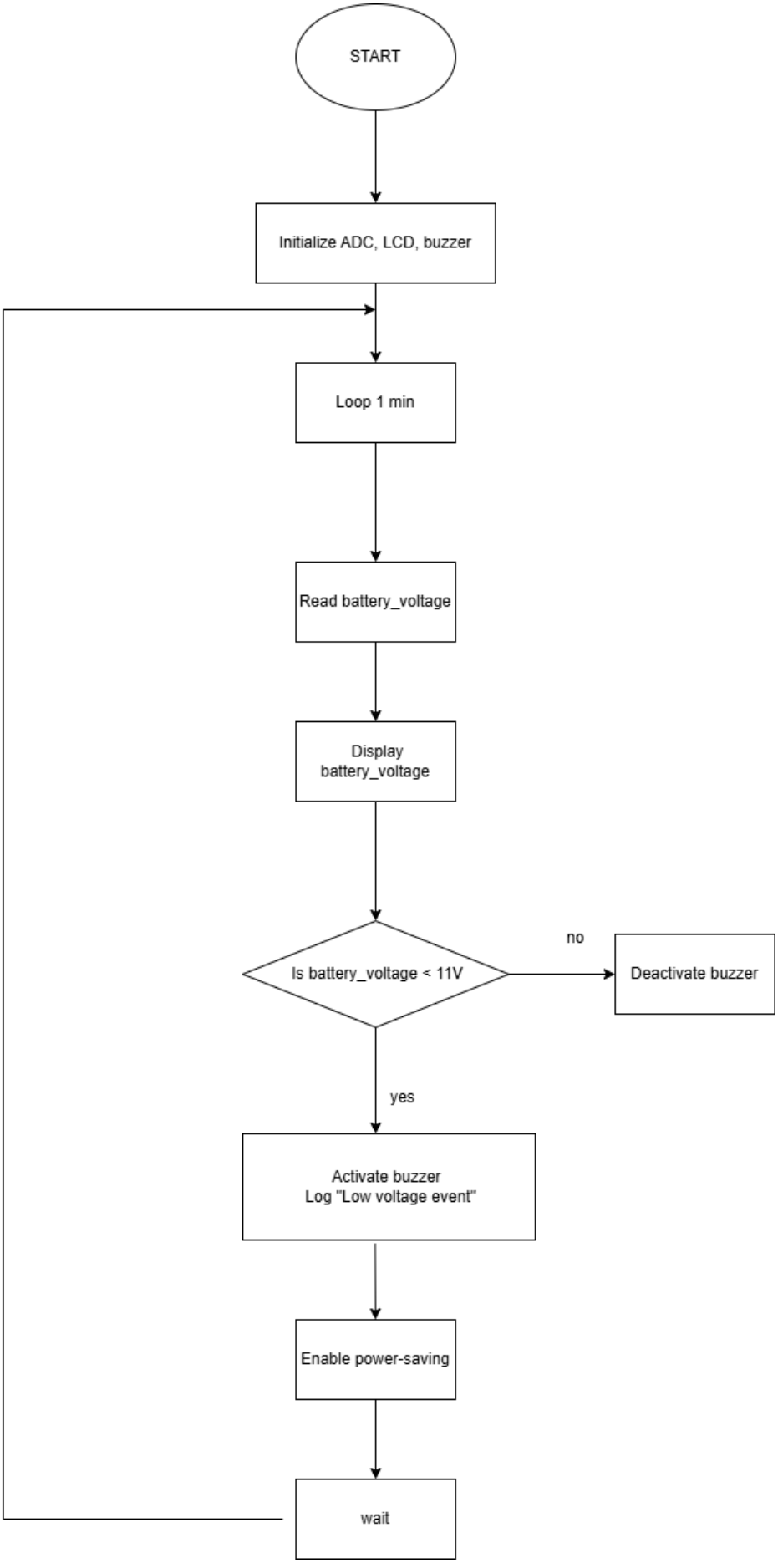
d. ENABLE power-saving mode during idle time

WAIT

REPEAT

END

FLOWCHART:


START
Initialize ADC, LCD, buzzer
Loop 1 min
Read battery_voltage
Display
battery_voltage
Is battery_voltage < 11V
no
Deactivate buzzer
yes
Activate buzzer
Log "Low voltage event"
Enable power-saving
wait

10. RFID-Based Access Control System Problem Statement: Design an access control system using RFID technology to grant or deny access based on scanned RFID tags. Requirements: • Continuously monitor for RFID tag scans using an RFID reader. • Compare scanned tags against an authorized list stored in memory. • Grant access by activating a relay if the tag is authorized; otherwise, deny access with an alert (buzzer). • Log access attempts (successful and unsuccessful) with timestamps to an SD card.

PSEUDOCODE:

START

1. Initialize RFID reader, relay, buzzer, and SD card for logging

LOOP:

a. WAIT for RFID tag scan

b. READ scanned_tag from RFID reader

c. IF (scanned_tag is in authorized_list) THEN

ACTIVATE relay to grant access

LOG "Access Granted" to SD card

ELSE

ACTIVATE buzzer for access denial

LOG "Access Denied" to SD card

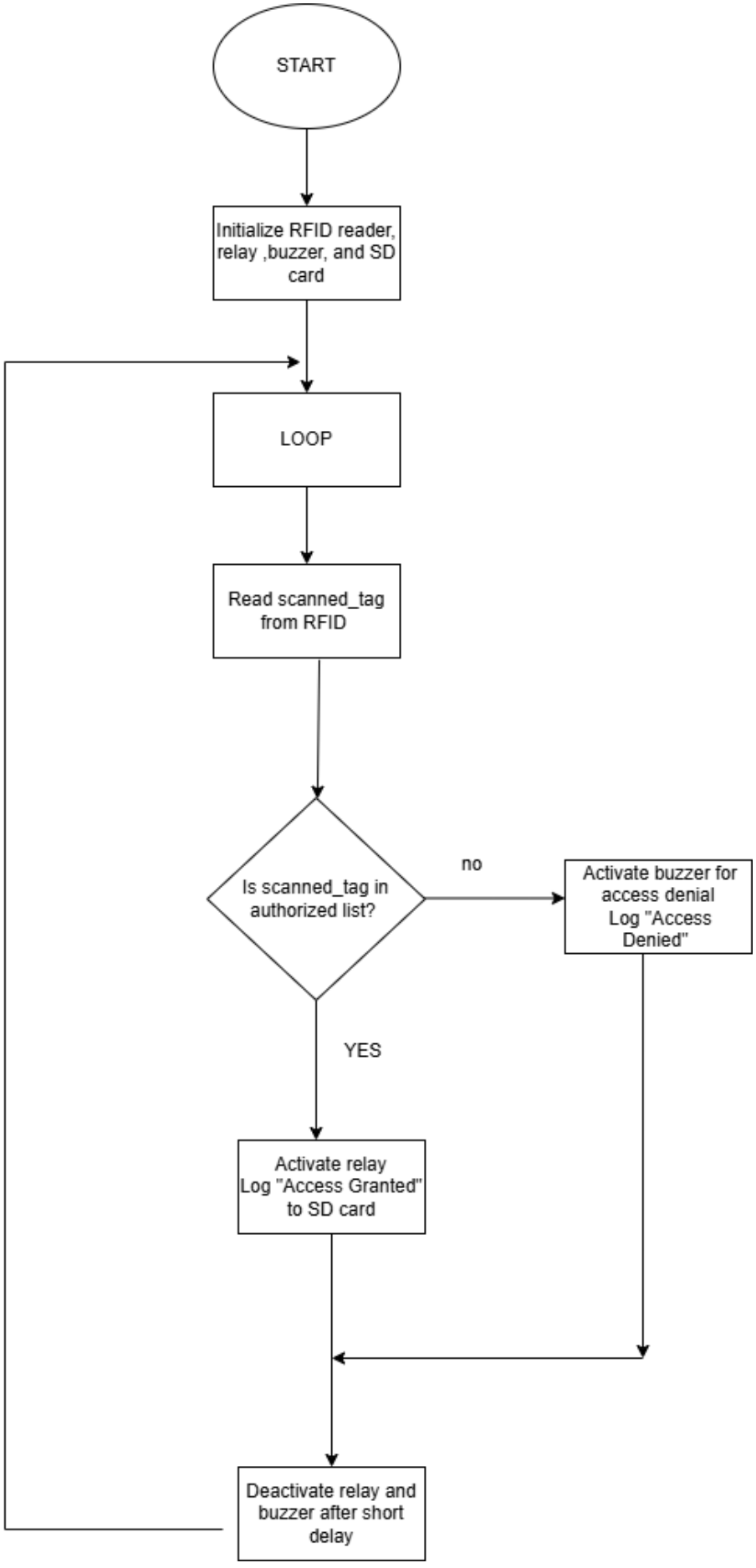
ENDIF

d. DEACTIVATE relay and buzzer after a short delay

REPEAT

END

FLOWCHART:


START
Initialize RFID reader, relay ,buzzer, and SD card
LOOP
Read scanned_tag from RFID
Is scanned_tag in authorized list?
no
Activate buzzer for access denial
Log "Access Denied"
YES
Activate relay
Log "Access Granted" to SD card
Deactivate relay and buzzer after short delay