ASSIGNMENT

1. Smart Home Temperature Control

Problem Statement:

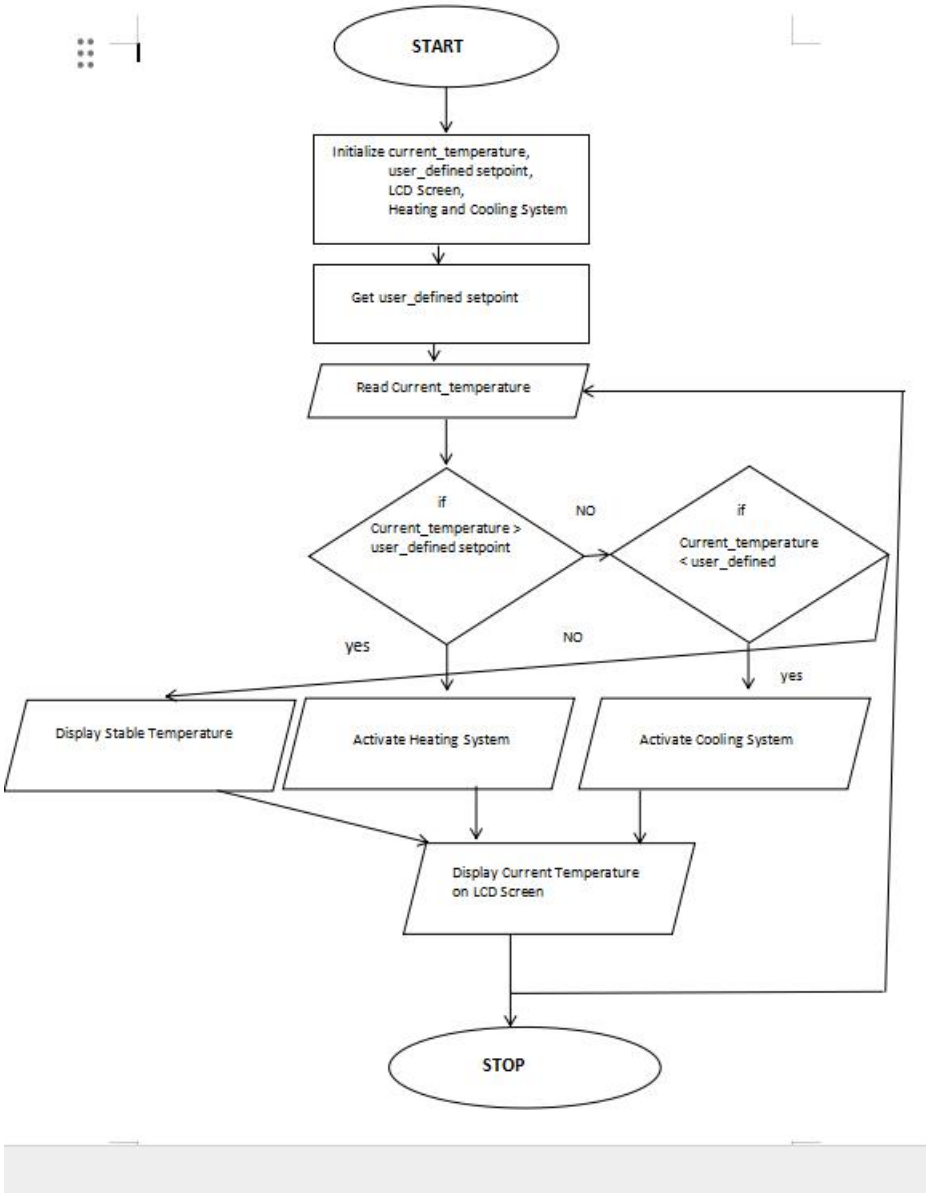
Design a temperature control system for a smart home. The system should read the current temperature from a sensor every minute and compare it to a user-defined setpoint.

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
Initialize Variables:
        current_temperature=0
        user_defined setpoint=0
        error_flag=FALSE
Input user_defined setpoint
Loop
        Current_temperature=read_temperature_sensor()
        If current_temperature=ERROR_VALUE
                Then error_flag=TRUE
                Display"Sensor error.Please check the sensor!!"
        Else
                If current_temperature > user_defined setpoint:
                        Display Current Temperature and Setpoint on LCD
                        "Activate Cooling System"
                Else If current_temperature < user_defined setpoint:
                        Display Current Temperature and Setpoint on LCD
                        "Activate Heating System"
                Else
                        Display Current Temperature and Setpoint on LCD
                        Maintain Current Temperature
                End if
```



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).

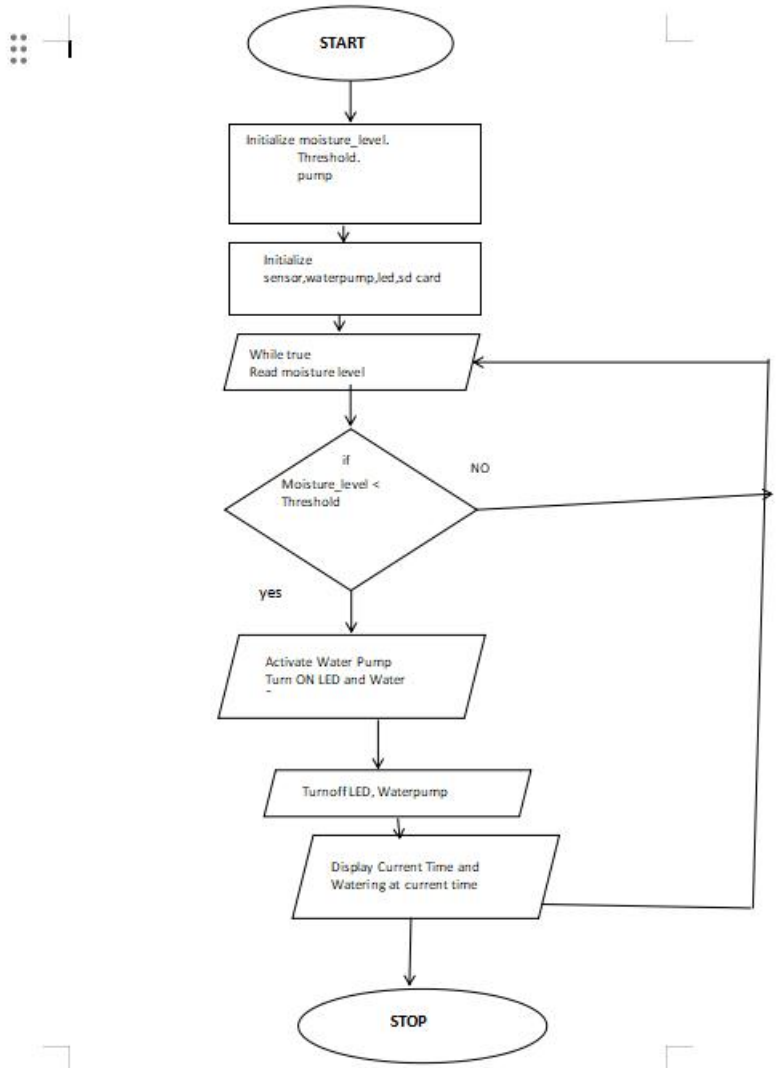
Pseudocode:

Start

Initialize Variables

moistureLevel=0,

```
        threshold=200,
        pump=4000;
Initialize the sensor, waterpump, Led, SD card
While True (do)
         moistureLevel=Read moistureLevel from sensor
        Display moistureLevel
        If moisture level < threshold
        then Print "Activate the water pump for a specified duration"
        Turn LED and Water pump ON
         Wait
         Turn water pump and LED OFF
         currentTime = Get currentTime
        Print "Watering at current time:"
        Display Current Time
        End if
End while
End
```



3. Motion Detection Alarm System

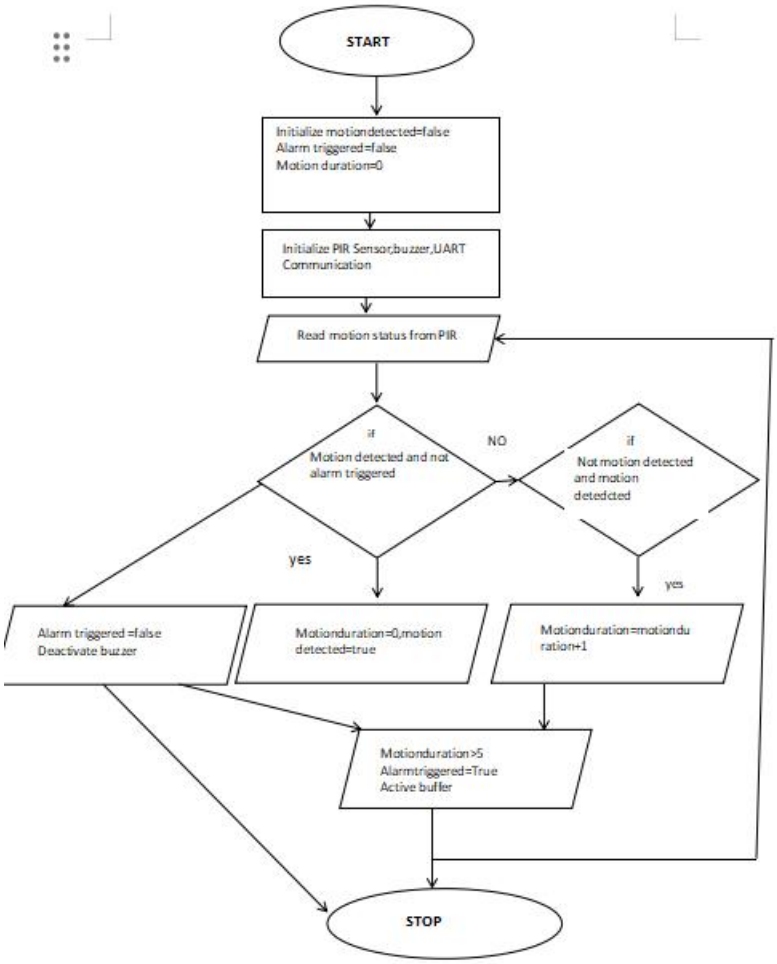
Problem Statement: Develop a security alarm system that detects motion using a PIR sensor.

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
Initialize variables:
motionDetected = FALSE
alarmTriggered = FALSE
motionDuration = 0
Initialize hardware: Initialize PIR sensor Initialize buzzer Initialize UART communication
Read motion status from PIR sensor
IF motion detected AND NOT alarmTriggered
THEN motionDuration = 0
motionDetected = TRUE
ELSE IF NOT motion detected AND motionDetected
THEN motionDuration = motionDuration + 1
IF motionDuration >= 5
THEN alarmTriggered = TRUE
Activate buzzer Send notification via UART
END IF
END IF
IF alarmTriggered AND NOT motionDetected
THEN alarmTriggered = FALSE
Deactivate buzzer
END IF
Wait for a short interval
END LOOP END



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

Initialize the system

- Setup heart rate sensor
- Setup LCD display
- Setup SD card for logging
- Initialize heartRateArray[60] to store heart rate data for 1 minute
- Initialize heartRateSum to 0

WHILE true

FOR each second in 60 seconds

i. Read currentHeartRate from heart rate sensor

ii. Store currentHeartRate in heartRateArray

iii. Update heartRateSum = heartRateSum + currentHeartRate

iv. IF currentHeartRate > 100 THEN

- Trigger buzzer

v. Display currentHeartRate on LCD

vi. Log currentHeartRate to SD card

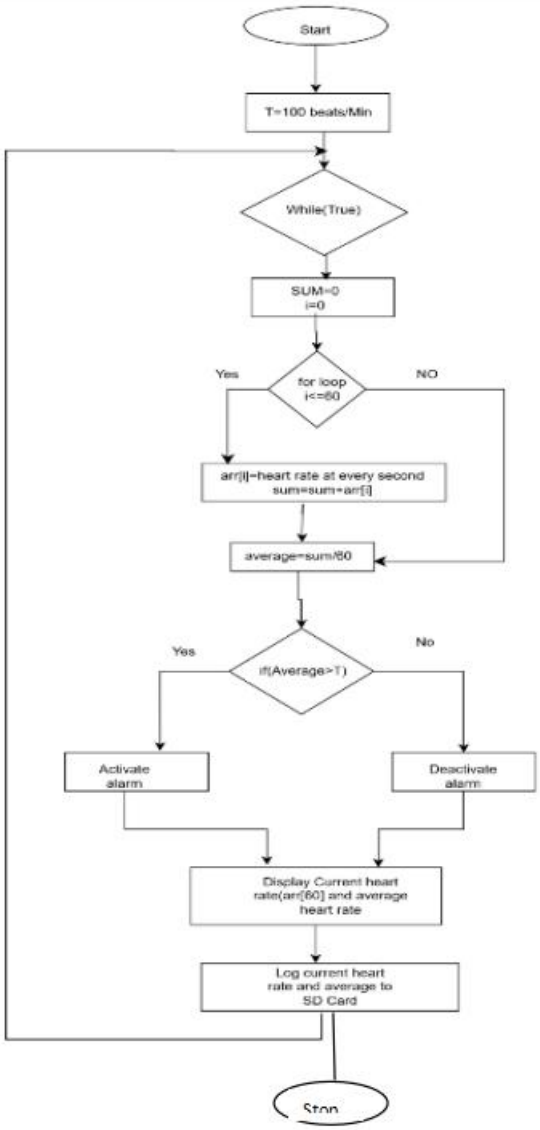
vii. WAIT for 1 second

b. Calculate averageHeartRate = heartRateSum / 60

c. Display averageHeartRate on LCD

d. Reset heartRateArray and heartRateSum for the next minute

ENDWHILE



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.

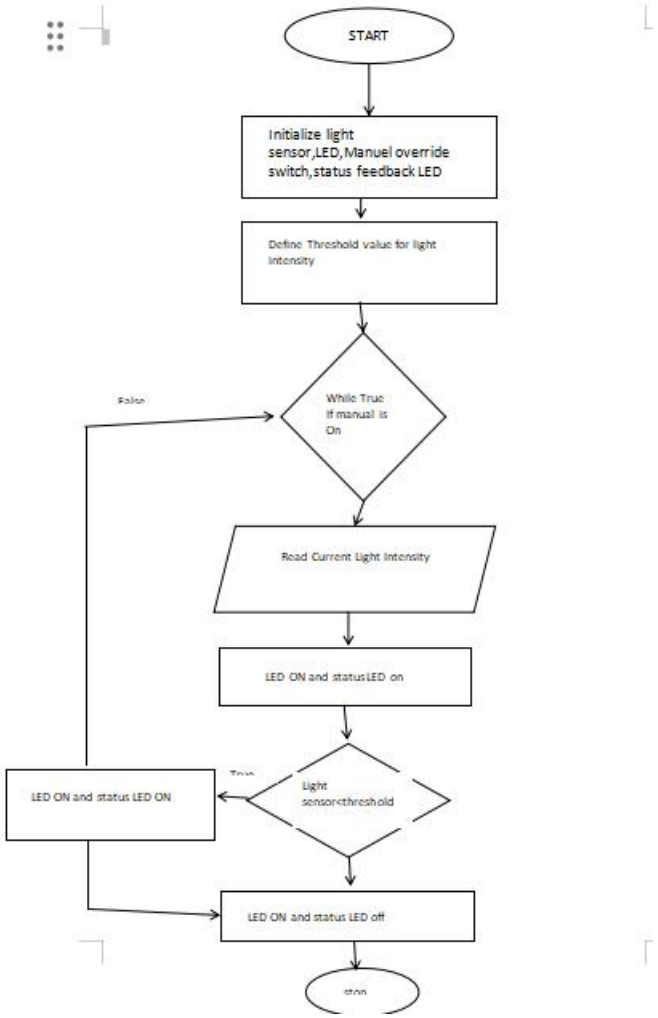
• Provide status feedback through another LED (e.g., blinking when in manual mode).

Pseudocode

Initialize the system
- Setup light sensor,LED,Manuel override switch
- Setup status feedback LED

Define threshold value for light intensity
WHILE true
a. Read current light intensity from the sensor
b. IF manual override switch is ON THEN
i. Turn ON status feedback LED (blinking)
ii. IF manual switch is ON THEN
- Turn ON the LED
iii. ELSE
- Turn OFF the LED
c. ELSE
i. Turn OFF status feedback LED
ii. IF light intensity < THRESHOLD THEN
- Turn ON the LED
iii. ELSE
- Turn OFF the LED

d. WAIT
ENDWHILE



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
Initialize LCD display, Start button, Stop button, Reset button, and SD card
Set elapsed_time to 0 seconds.
Set stopwatch_running = False and start_time to None
If Start button is pressed:
        If stopwatch_running is False:
                Set start_time to current time
                Set stopwatch_running to True
```

Log start_time to SD card

If Stop button is pressed:

If stopwatch_running is True:

Set stopwatch_running to False

Log current time as stop time to SD card

If Reset button is pressed:

Set elapsed_time to 0 Display "00:00:00" on LCD

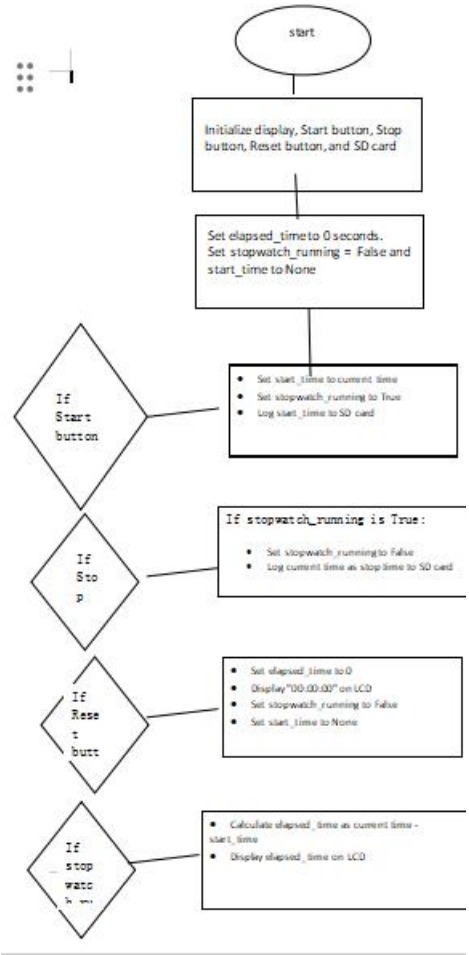
Set stopwatch_running to False

Set start_time to None

If stopwatch_running is True:

Calculate elapsed_time as current time - start_time Display elapsed_time on the LCD

Wait before the next loop



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.

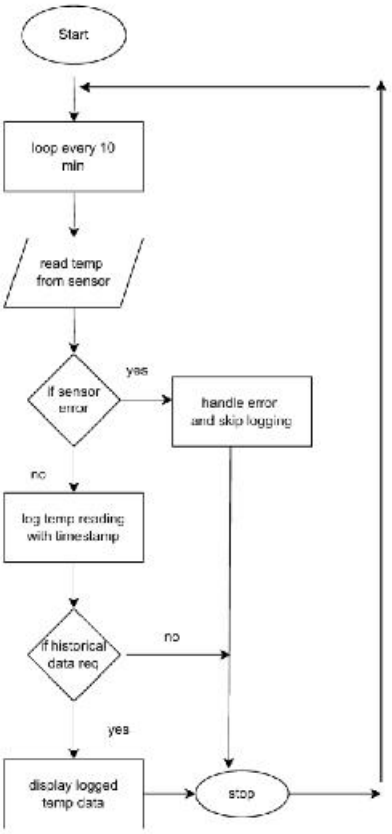
1. Initialize the system
   - Setup temperature sensor
   - Setup storage (array or log file)
   - Initialize temperatureArray for storing temperature data
   - Initialize error handling mechanism

2. WHILE true
   a. Every 10 minutes:
      i. Read currentTemperature from temperature sensor
      ii. IF sensor read fails THEN
         - Log error message
         - CONTINUE to the next iteration

      iii. Get currentTimestamp
      iv. Store currentTemperature and currentTimestamp in temperatureArray

   b. WAIT for 10 minutes

3. Provide functionality to retrieve and display historical data
   a. Upon user request:
      i. Retrieve temperature data and timestamps from temperatureArray
      ii. Display the historical data
ENDWHILE



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

1. Initialize the system
   - Setup Bluetooth communication module
   - Setup motor control for movement
   - Setup speed control
   - Setup LEDs for feedback

2. WHILE true
   a. Listen for Bluetooth commands
   b. Decode received command

   c. IF command == "Forward" THEN

i. Move robot forward
ii. Set LED to indicate moving forward

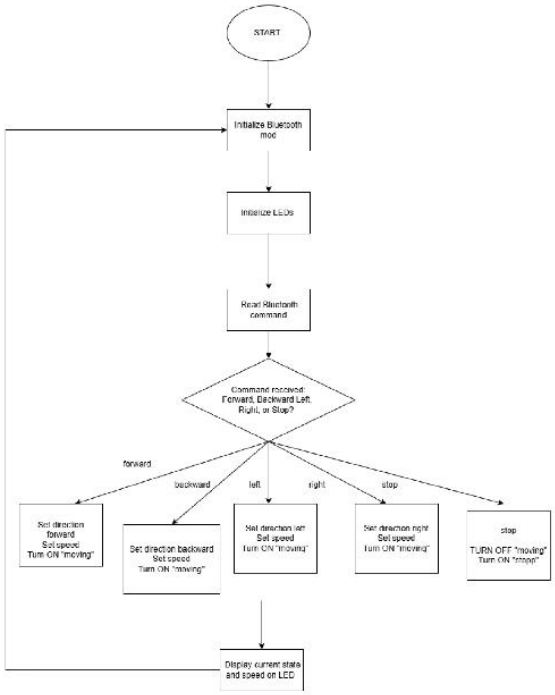
d. ELSE IF command == "Backward" THEN
i. Move robot backward
ii. Set LED to indicate moving backward

e. ELSE IF command == "Left" THEN
i. Turn robot left
ii. Set LED to indicate turning left

f. ELSE IF command == "Right" THEN
i. Turn robot right
ii. Set LED to indicate turning right

g. ELSE IF command == "Stop" THEN
i. Stop the robot
ii. Set LED to indicate stopped

h. ELSE IF command includes speed control THEN
i. Adjust robot speed accordingly
ENDWHILE



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.

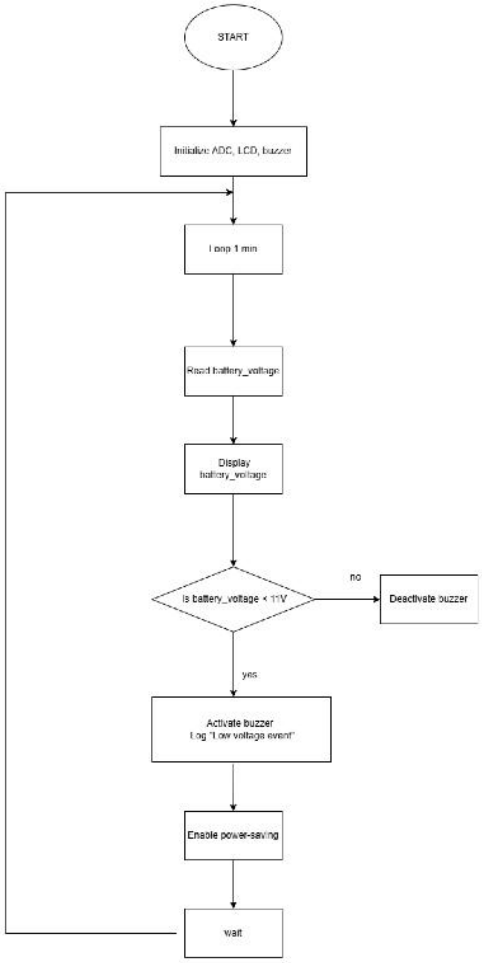
Pseudocode

1. Initialize the system
   - Setup ADC for voltage measurement
   - Setup buzzer for alerts
   - Setup LCD display for voltage display
   - Initialize memory for event logging
   - Implement power-saving mode

2. WHILE true
   a. Measure battery voltage using ADC
   b. Display currentVoltage on the LCD
   c. IF currentVoltage < 11V THEN
      i. Trigger buzzer alert
      ii. Log the event to memory
   d. Enter power-saving mode

   e. WAIT for 1 minute

ENDWHILE



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

1. Initialize the system
   - Setup RFID reader
   - Setup relay for granting access
   - Setup buzzer for denial alerts
   - Setup SD card for logging access attempts
   - Initialize list of authorized RFID tags in memory

2. WHILE true
   a. Monitor for RFID tag scans
   b. IF RFID tag is scanned THEN
      i. Read scannedTagID from RFID reader

ii. Get currentTimestamp
iii. IF scannedTagID is in authorized list THEN
- Activate relay to grant access
- Log "Access Granted" with scannedTagID and currentTimestamp to SD card
iv. ELSE
- Trigger buzzer alert to deny access
- Log "Access Denied" with scannedTagID and currentTimestamp to SD card
ENDWHILE