Write the Pseudocode and Flowchart for the problem statements mentioned below:

---

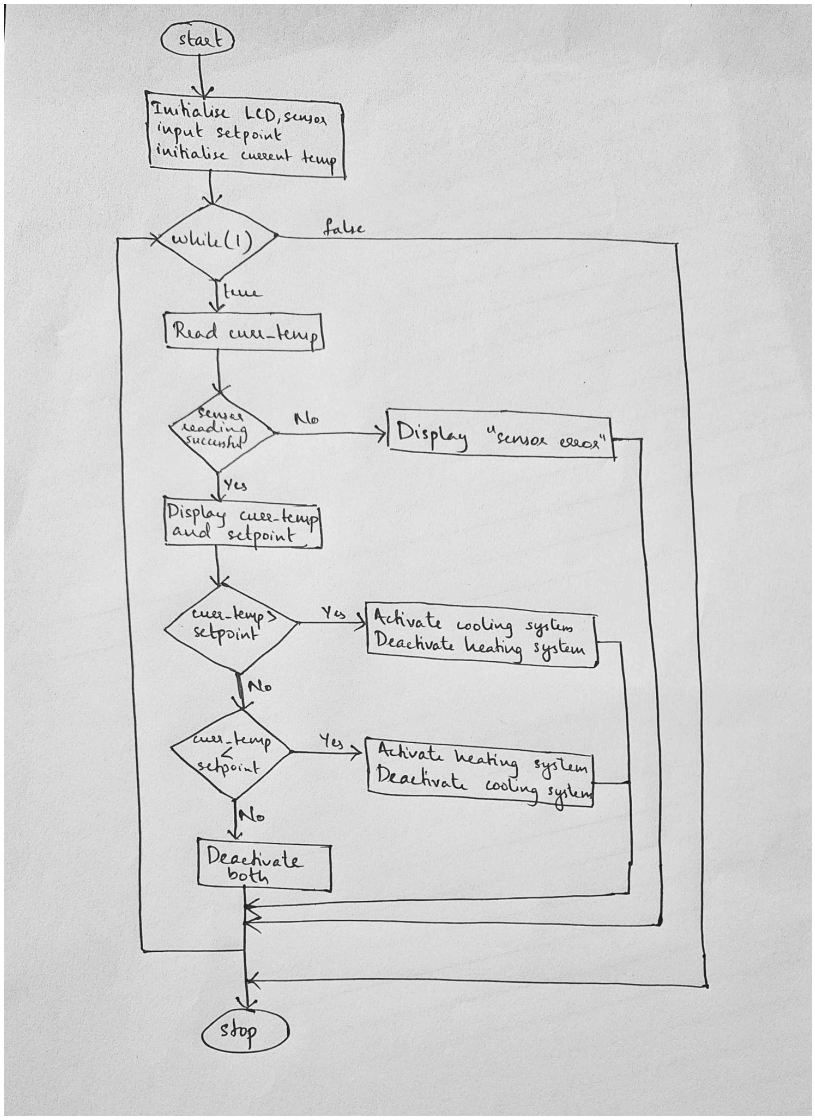
1. Smart Home Temperature Control

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:

```
Initialize LCD, sensor
Input setpoint
Initialise current temperature=curr_temp
While(monitor every 1 min):
    Read curr_temp
    If(sensor_reading is successful):
        Display curr_temp and setpoint
        If(curr_temp > setpoint):
            Activate cooling system
            Deactivate heating system
        Else if(curr_temp < setpoint):
            Activate heating system
            Deactivate cooling system
        Else:
            Deactivate both
    Else:
        Display "Sensor Error"
End while
```

Flowchart:



2. Automated Plant Watering System

Problem Statement:

Create an automated watering system for plants that checks soil moisture levels and waters the plants accordingly.

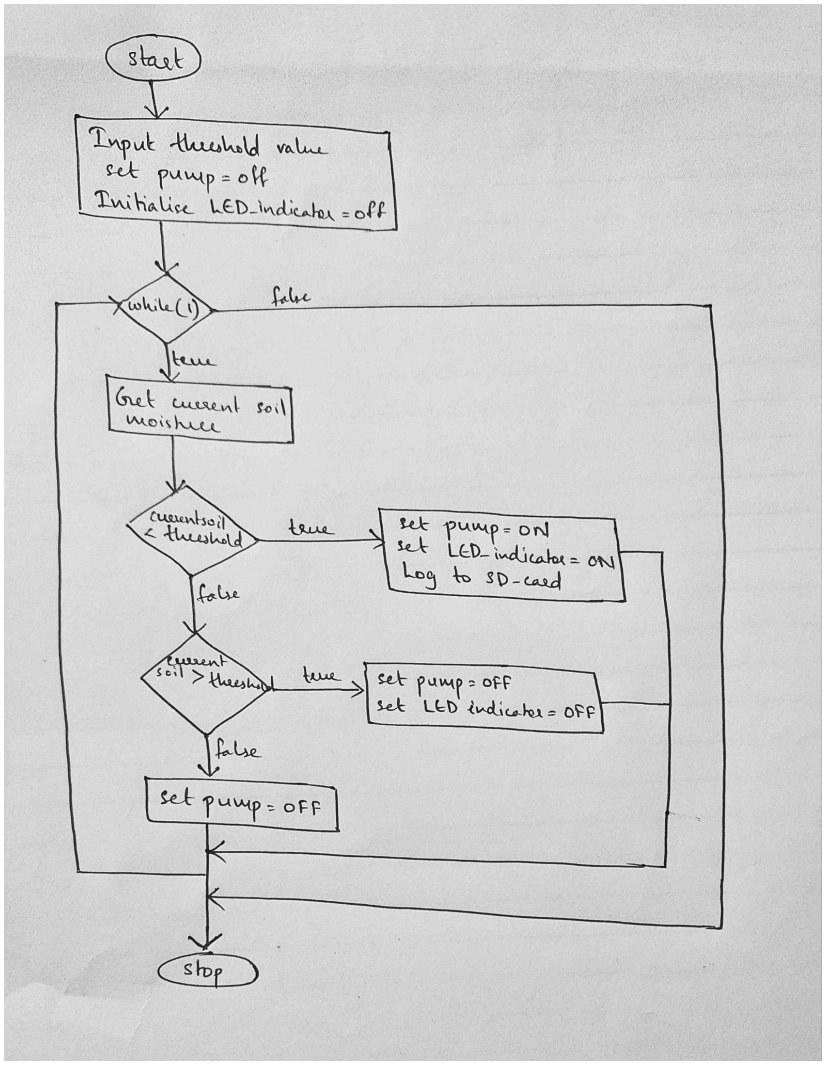
Requirements:

• 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:

input the threshold value=threshold
set pump=off
Initialise LED_indicator=off
while(monitor soil moisture every 1 hour):
    Get current soil moisture= currsoil
     If (currsoil<threshold):
        Set pump=on
        Set LED_indicator=on
        Log to SD card
    Else if(currsoil>threshold):
        Set pump=off
        Set LED_indicator=off
    Else
        set pump=off
 End while

Flowchart:



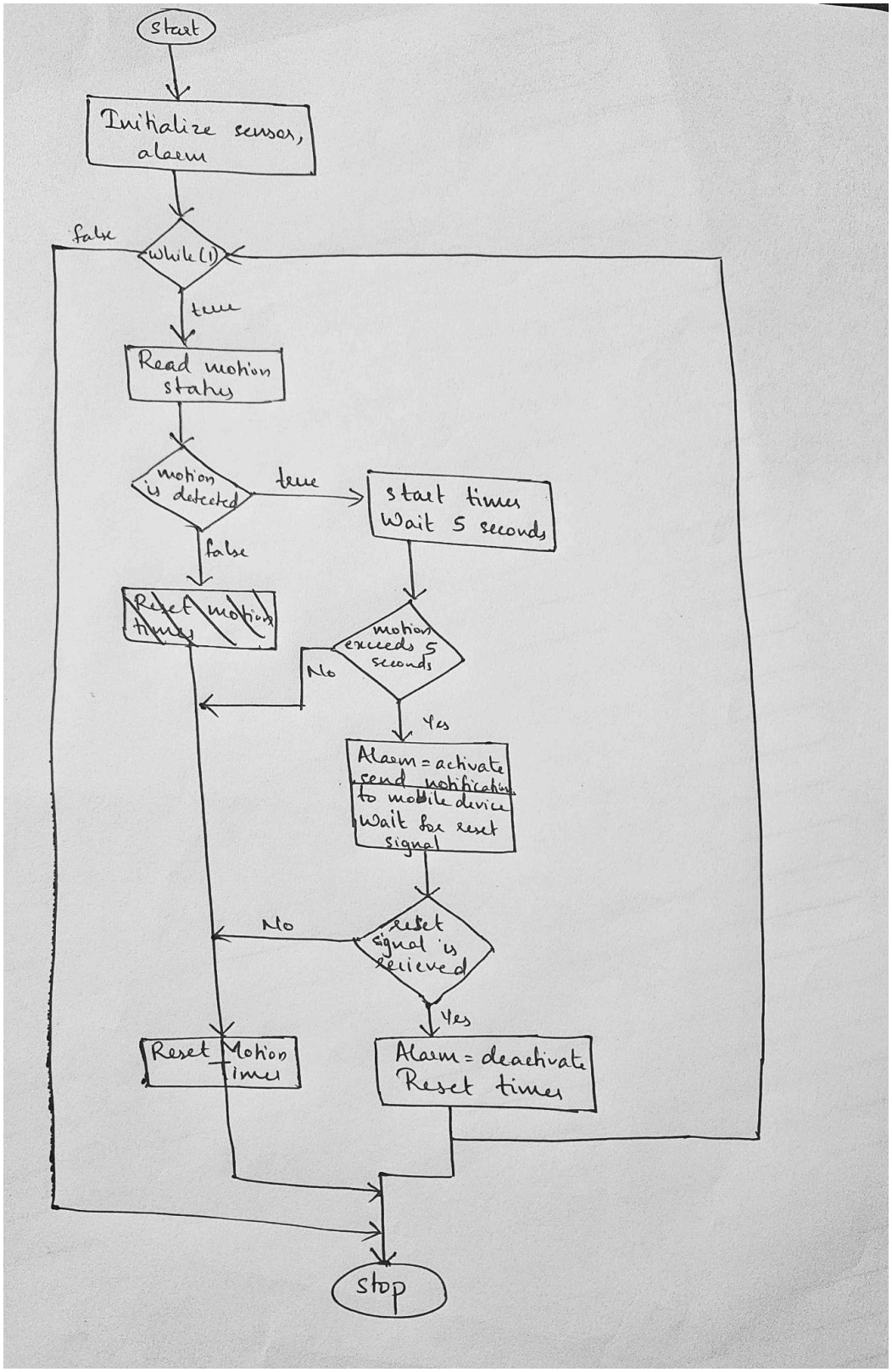
3. Motion Detection Alarm System

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:

```
Initialize sensor
Initialize alarm
While (True):
    Read motion status
    If (motion is detected):
        Start timer
        Wait 5 seconds
        If (motion exceeds 5 seconds):
            Alarm=activate
            Send notification to mobile device via UART
            Wait for reset signal
            If (reset signal is received):
                Alarm=deactivate
                Reset timer
            End If
        End If
    Else:
        Reset motion timer
End while
```

Flowchart:


start
Initialize sensor, alarm
false
while (1)
true
Read motion status
motion is detected
true
start timer Wait 5 seconds
false
Reset motion timer
motion exceeds 5 seconds
No
Yes
Alarm = activate send notification to mobile device Wait for reset signal
No
reset signal is recieved
Yes
Reset Motion timer
Alarm = deactivate Reset timer
stop

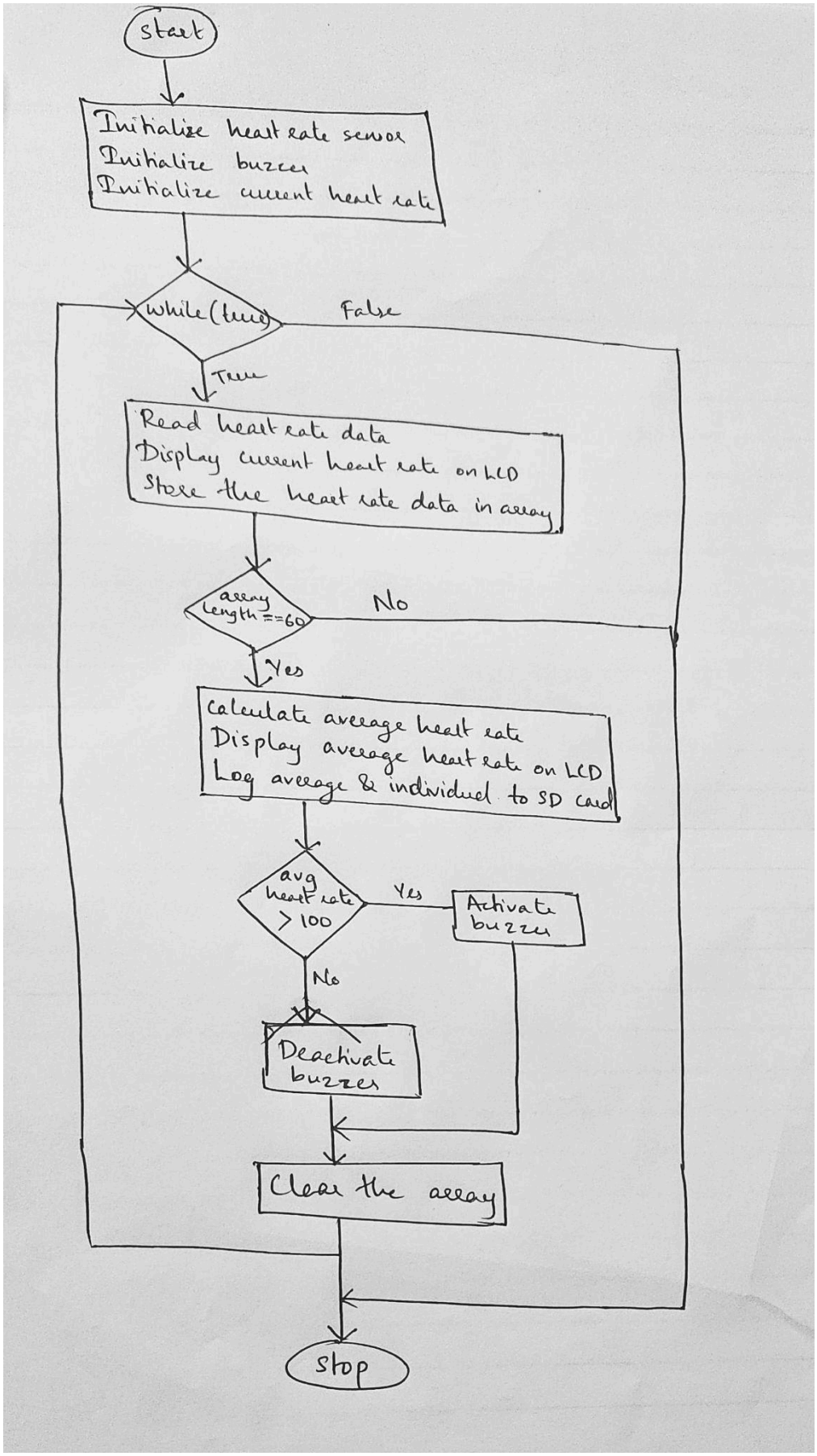
4. Heart Rate Monitor

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 heart rate sensor
Initialize buzzer
Initialize current heart rate=curr_rate
While(Every 1 Second):
   Read heart rate data
   Display current heart rate on LCD
   Store the heart rate data in an array
   If (array contains 60 seconds of data):
      Calculate average heart rate over the last 60 seconds
      Display average heart rate on LCD
      Log average heart rate and individual readings to SD card
      If (average heart rate > 100):
         Activate buzzer
      Else
         Deactivate buzzer
      Clear the array
   End If
End while
```

Flowchart:


start
Initialize heart rate sensor
Initialize buzzer
Initialize current heart rate
while (true)
False
True
Read heart rate data
Display current heart rate on LCD
Store the heart rate data in array
array length == 60
No
Yes
Calculate average heart rate
Display average heart rate on LCD
Log average & individual to SD card
avg heart rate > 100
Yes
Activate buzzer
No
Deactivate buzzer
Clear the array
stop

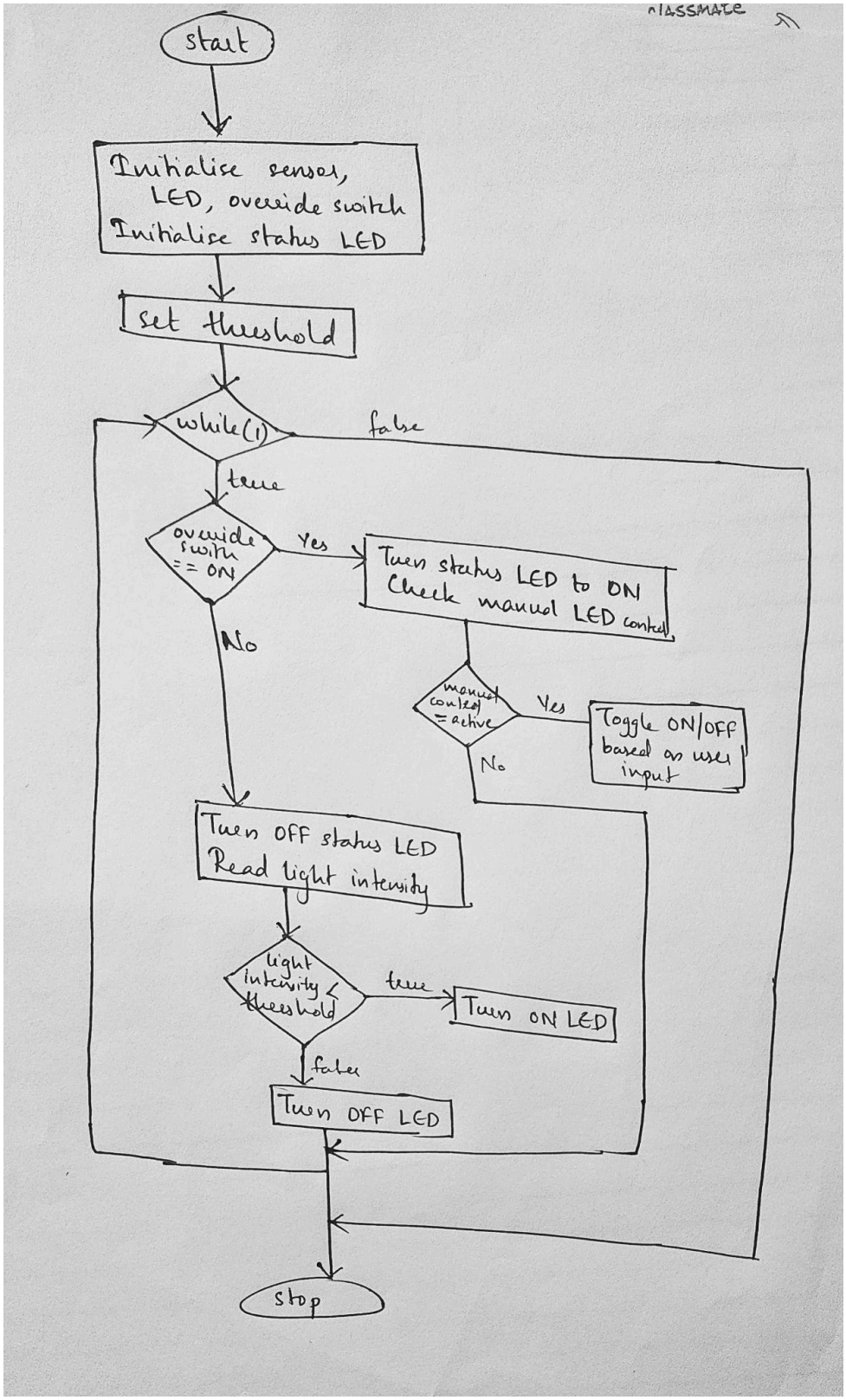
5. LED Control Based on Light Sensor

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 sensor
Initialize LED
Initialize override switch
Initialize status LED for feedback
Set threshold
while (Every 1 Minute):
    If override switch is ON:
        Turn status LED to ON
        If manual control is active, toggle LED ON/OFF based on user input
    Else:
        Turn off status LED
        Read light intensity from sensor
        If (light intensity < threshold):
            Turn ON LED
        Else:
            Turn OFF LED
End while
```

Flowchart:


start
Initialise sensor, LED, override switch
Initialise status LED
set threshold
while(1)
false
true
override switch == ON
Yes
Turn status LED to ON
Check manual LED control
No
manual control = active
Yes
Toggle ON/OFF based on user input
No
Turn OFF status LED
Read light intensity
light intensity < threshold
true
Turn ON LED
false
Turn OFF LED
stop

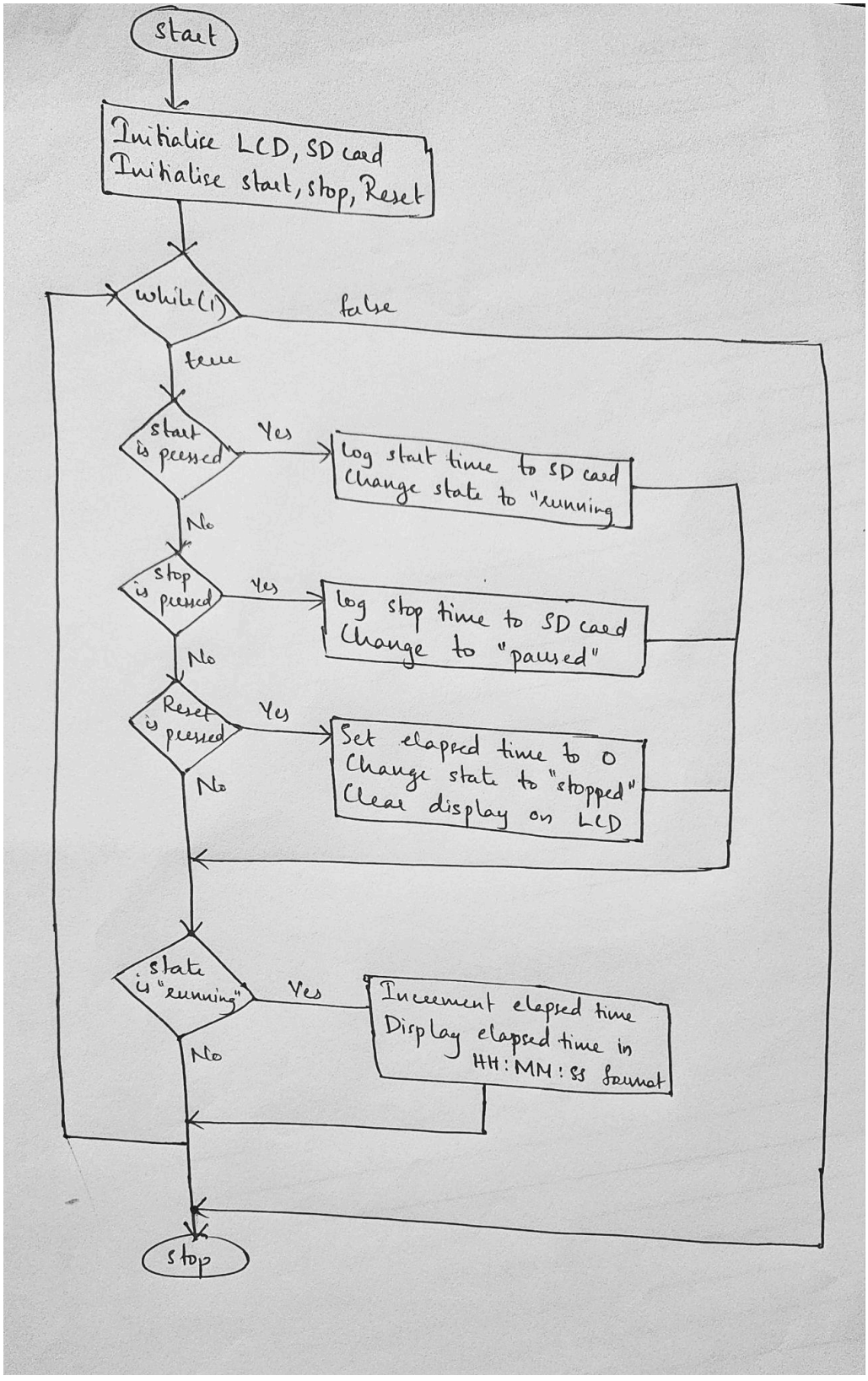
6. Digital Stopwatch

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:

```
Initialize LCD
Initialize SD card
Initialize Start, Stop, Reset
while(true):
   If(Start is pressed):
         Log start time to SD card
         Change state to "running"
   Else If(Stop is pressed):
         Log stop time to SD card
         Change state to "paused"
   Else If(Reset is pressed):
      Set elapsed time to 0
      Change state to "stopped"
      Clear display on LCD
   If(state is "running"):
      Increment elapsed time by 1 second
      Display elapsed time on LCD in HH:MM:SS format
End while
```

Flowchart:


start
Initialise LCD, SD card
Initialise start, stop, Reset
while(1)
false
true
start is pressed
Yes
log start time to SD card
Change state to "running"
No
stop is pressed
Yes
log stop time to SD card
Change to "paused"
No
Reset is pressed
Yes
Set elapsed time to 0
Change state to "stopped"
Clear display on LCD
No
state is "running"
Yes
Increement elapsed time
Display elapsed time in
HH:MM:SS format
No
stop

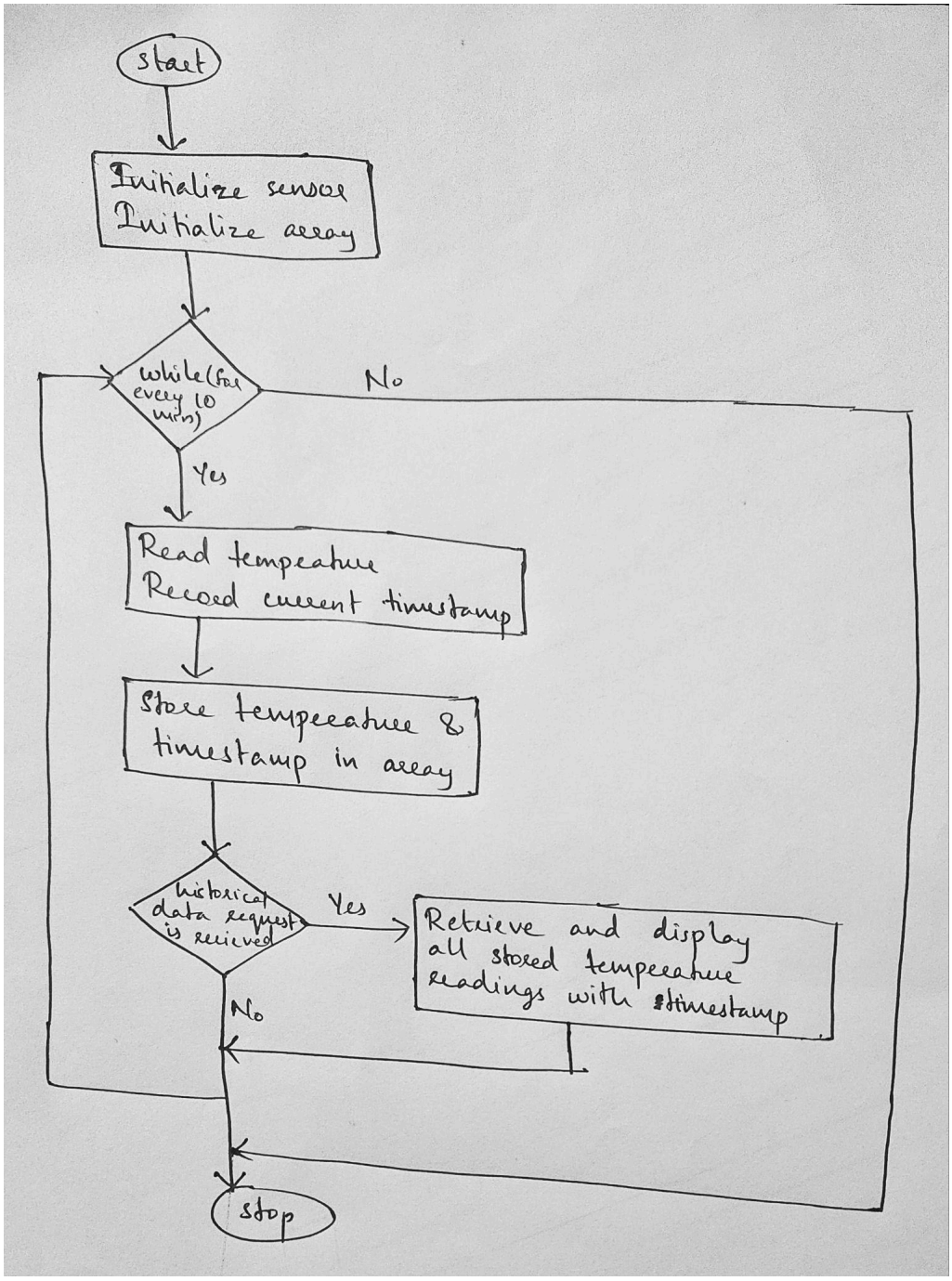
7. Temperature Logging System

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:

```
Initialize sensor
Initialize array
While(for every 10 minutes):
    Read temperature
    Record current timestamp
    Store temperature and timestamp in array
    If(historical data request is received):
        Retrieve and display all stored temperature readings with timestamps
End while
```

Flowchart:



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.

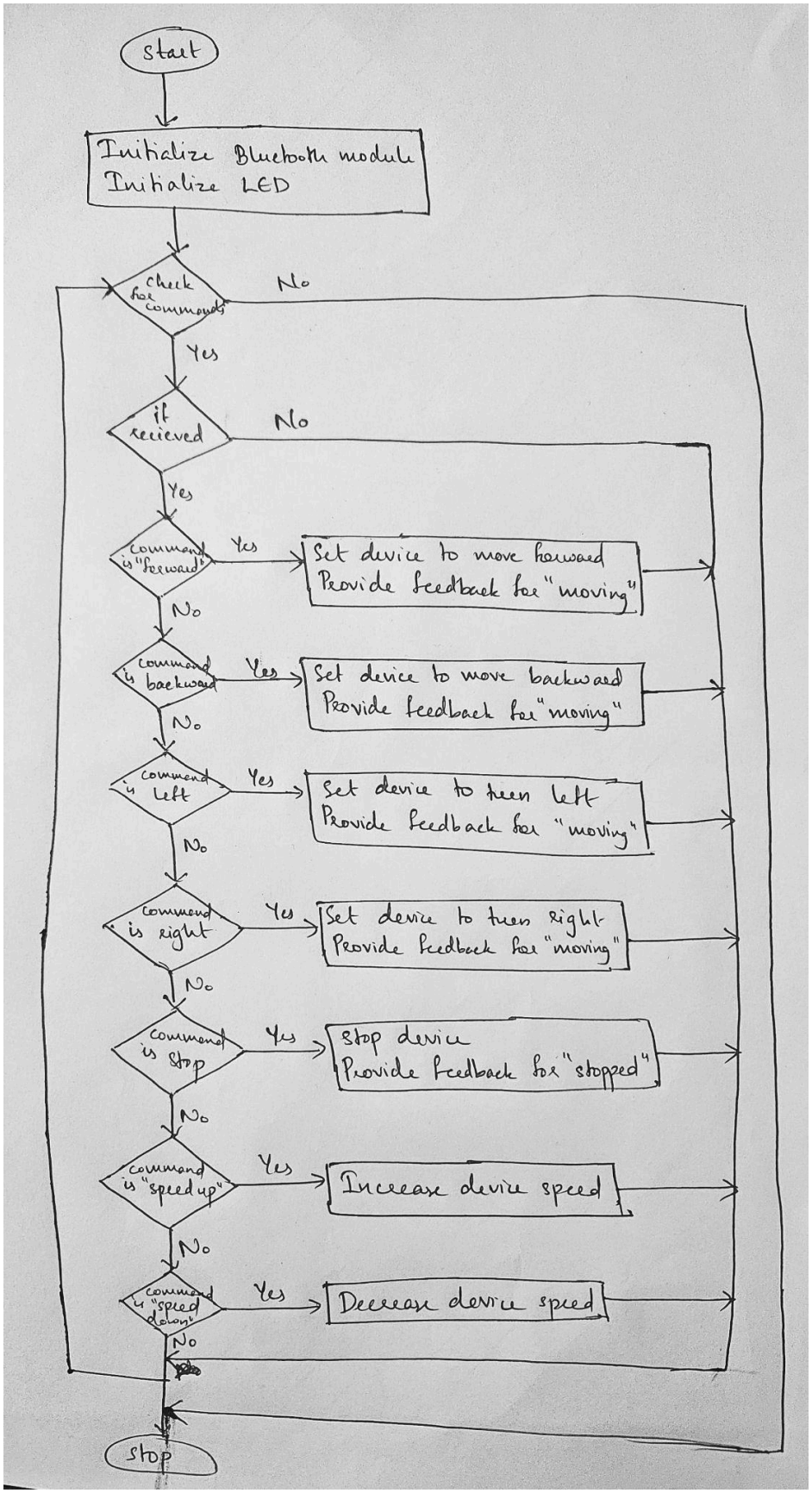
• Implement commands for moving forward, backward, left, and right.

• Include speed control functionality based on received commands.
• Provide feedback through LEDs indicating the current state (e.g., moving or stopped).

Pseudocode:

```
Initialize Bluetooth module
Initialize LED for feedback
while(check if there's any bluetooth commands):
    If (command received):
        If (command is "FORWARD"):
            Set device to move forward
            Providing feedback through LED for "Moving"
        Else if(command is "BACKWARD"):
            Set device to move backward
            Providing feedback through LED for "Moving"
        Else if (command is "LEFT"):
            Set device to turn left
            Providing feedback through LED for "Moving"
        Else if(command is "RIGHT"):
            Set device to turn right
            Providing feedback through LED for "Moving"
        Else if (command is "STOP"):
            Stop device
            Providing feedback through LED for "Stopped"
        Else if (command is "SPEED_UP"):
            Increase device speed
        Else if (command is "SLOW_DOWN"):
            Decrease device speed
End While
```

Flowchart:


start
Initialize Bluetooth module
Initialize LED
Check for commands
No
Yes
if recieved
No
Yes
Command is "forward"
Yes
Set device to move forward
Provide feedback for "moving"
No
Command is backward
Yes
Set device to move backward
Provide feedback for "moving"
No
Command is left
Yes
Set device to turn left
Provide feedback for "moving"
No
Command is right
Yes
Set device to turn right
Provide feedback for "moving"
No
Command is stop
Yes
Stop device
Provide feedback for "stopped"
No
Command is "speed up"
Yes
Increase device speed
No
Command is "speed down"
Yes
Decrease device speed
No
stop

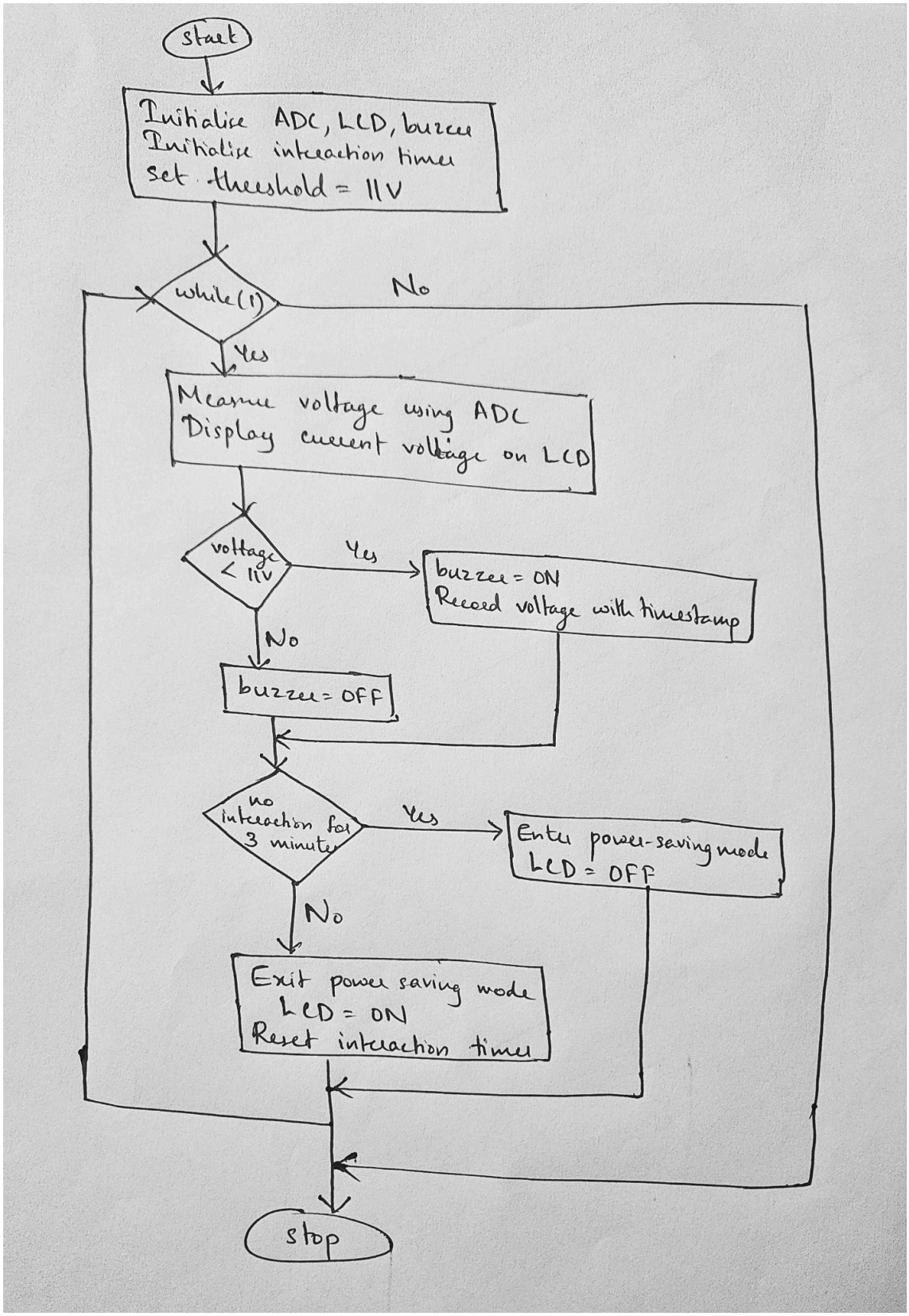
9. Battery Monitoring System

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:

```
Initialize ADC
Initialize LCD, buzzer
Initialize interaction timer
Set voltage_threshold=11V
While(every 1 minute):
   Measure voltage using ADC
   Display current voltage on LCD
   If(voltage < 11V):
      buzzer=ON
      Record voltage with timestamp in memory
   Else:
      buzzer=OFF
   If (no interaction for 3 minutes):
      Enter power-saving mode
      LCD=OFF
   Else:
      Exit power-saving mode
      LCD=ON
      Reset interaction timer
End While
```

Flowchart:


start
Initialize ADC, LCD, buzzer
Initialize interaction timer
set threshold = 11V
while(1)
No
Yes
Measure voltage using ADC
Display current voltage on LCD
voltage < 11V
Yes
buzzer = ON
Record voltage with timestamp
No
buzzer = OFF
no interaction for 3 minutes
Yes
Enter power-saving mode
LCD = OFF
No
Exit power saving mode
LCD = ON
Reset interaction timer
stop

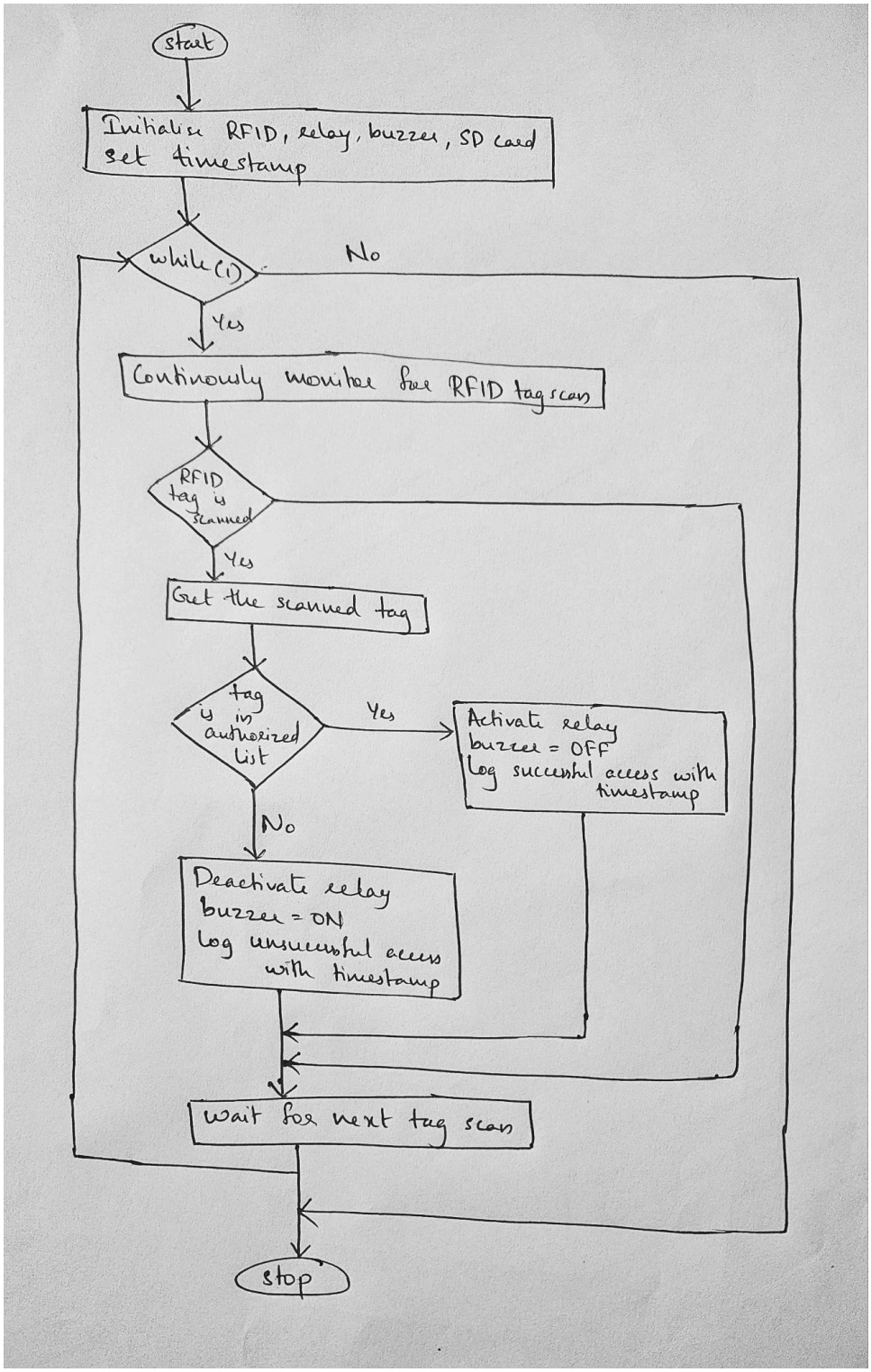
10. RFID-Based Access Control System

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:

```
Initialize RFID reader, relay, buzzer
Initialize SD card for logging
Set timestamp
While(true):
   Continuously monitor for RFID tag scan
   If (RFID tag is scanned):
      Get the scanned tag ID
      If (scanned tag is in authorized list):
         Activate relay
         buzzer=OFF
         Log successful access attempt with timestamp on SD card
         Trigger “access granted” LED
      Else:
         Deactivate relay
         buzzer=ON
         Log unsuccessful access attempt with timestamp on SD card
         Trigger “access denied” LED
   Wait for the next tag scan
End while
```

Flowchart:


start
Initialise RFID, relay, buzzer, SD card
set timestamp
while (1)
No
Yes
Continously monitor for RFID tag scan
RFID tag is scanned
Yes
Get the scanned tag
tag is in authorized list
Yes
Activate relay
buzzer = OFF
log successful access with timestamp
No
Deactivate relay
buzzer = ON
log unsuccessful access with timestamp
wait for next tag scan
stop