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.

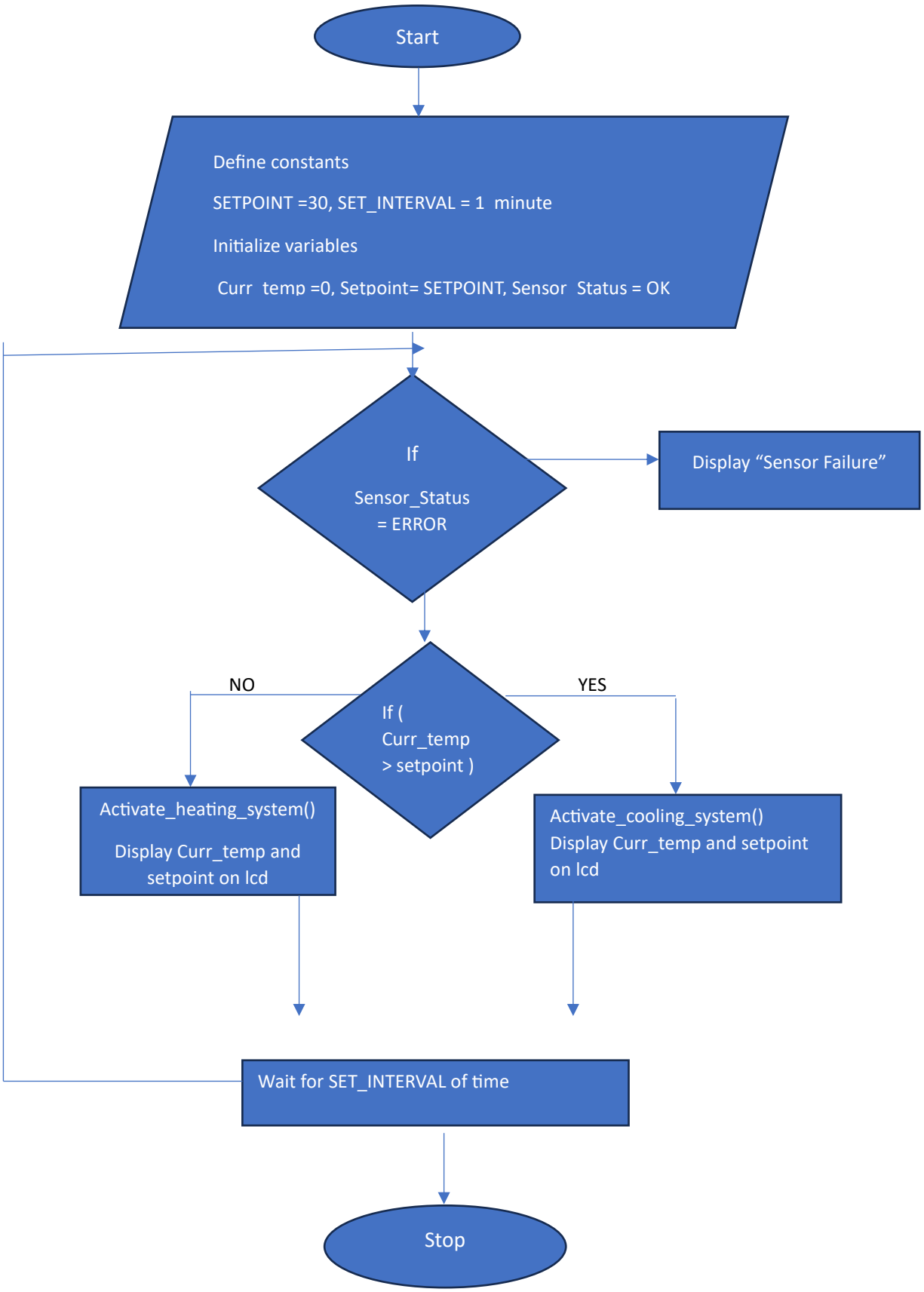
Requirements:

- 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

```
//Define constants

SETPOINT =30

SET_INTERVAL = 1 minute

//Initialise variables

Curr_temp = 0

Setpoint= SETPOINT

Sensor_Status = OK

//main loop

Curr_temp = read_temparature_sensor()     // read_temparature_sensor()     function read temperature from sensor

If  (Sensor_Status == ERROR)

   Print(“Sensor Failure)

If ( Curr_temp > setpoint )

     Activate_cooling_system()  // Activate_cooling_system()  fuction to activate the cooling system

     Display Curr_temp and setpoint on lcd

Else if( Curr_temp < setpoint)

    Activate_heating_system()  //    Activate_heating_system() fuction to activate the heating  system

    Display Curr_temp and setpoint on lcd

Wait for SET_INTERVAL of time
```

FLOWCHART


Start
Define constants
SETPOINT =30, SET_INTERVAL = 1 minute
Initialize variables
Curr temp =0, Setpoint= SETPOINT, Sensor Status = OK
If
Sensor_Status
= ERROR
Display “Sensor Failure”
NO
YES
If (
Curr_temp
> setpoint)
Activate_heating_system()
Display Curr_temp and
setpoint on lcd
Activate_cooling_system()
Display Curr_temp and setpoint
on lcd
Wait for SET_INTERVAL of time
Stop

2. Automated Plant Watering System

Problem Statement:

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

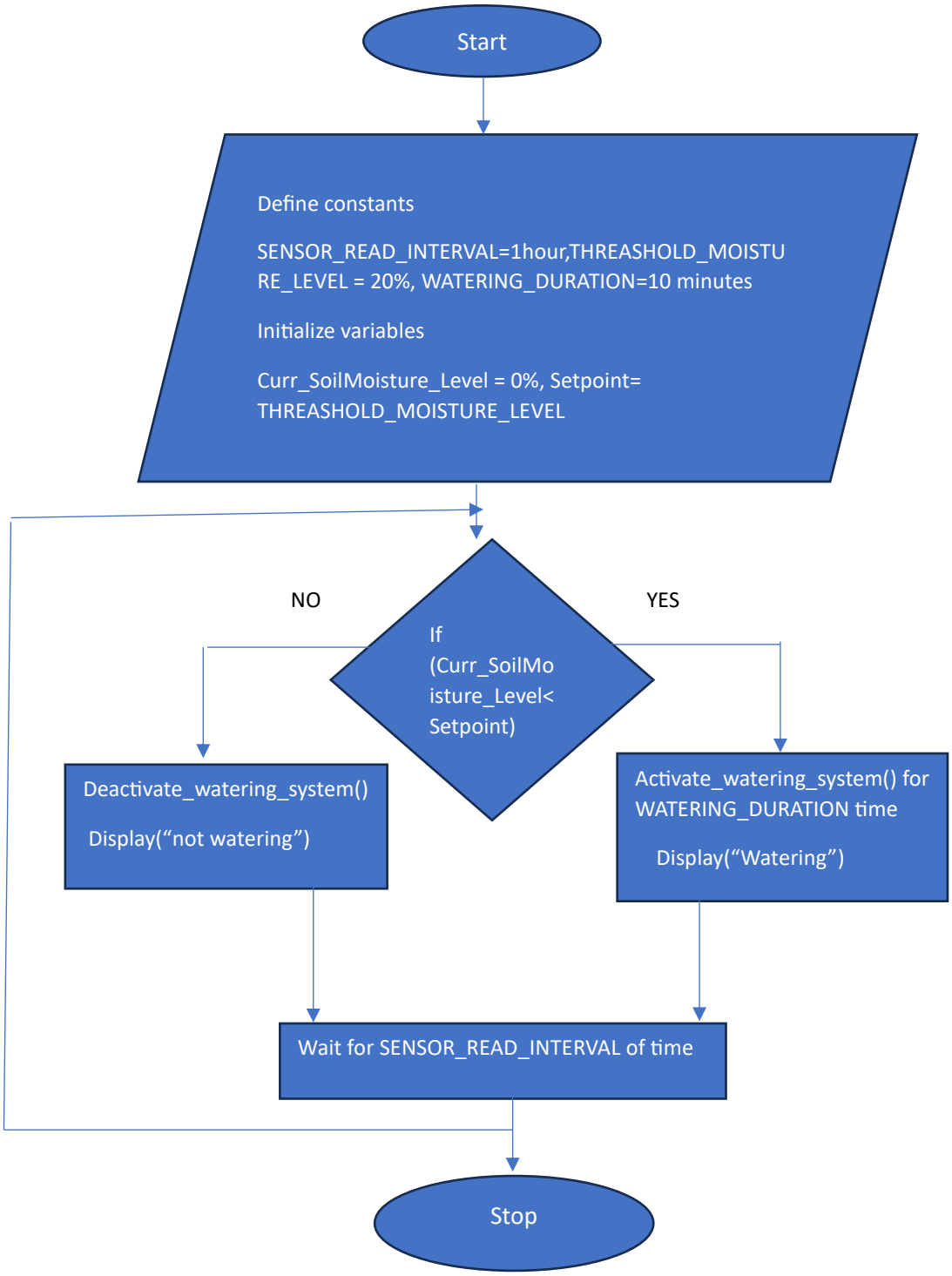
Requirements:

- 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

```
//Define constants
SENSOR_READ_INTERVAL=1hour
THREASHOLD_MOISTURE_LEVEL = 20%
WATERING_DURATION=10 minutes
//Initialise variables
Curr_SoilMoisture_Level = 0%
Setpoint= THREASHOLD_MOISTURE_LEVEL
//main loop
Curr_SoilMoisture_Level= read_soilmoisture_sensor()
If (Curr_SoilMoisture_Level< Setpoint)
   Activate_watering_system() for WATERING_DURATION time
   Display(“Watering”)
Else
    Deactivate_watering_system()
    Display(“not watering”)
Wait for SENSOR_READ_INTERVAL of time
```

FLOWCHART


Start
Define constants
SENSOR_READ_INTERVAL=1hour,THREASHOLD_MOISTURE_LEVEL = 20%, WATERING_DURATION=10 minutes
Initialize variables
Curr_SoilMoisture_Level = 0%, Setpoint= THREASHOLD_MOISTURE_LEVEL
If (Curr_SoilMoisture_Level< Setpoint)
NO
YES
Deactivate_watering_system()
Display("not watering")
Activate_watering_system() for WATERING_DURATION time
Display("Watering")
Wait for SENSOR_READ_INTERVAL of time
Stop

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 PIR sensor, buzzer, and UART communication

Initialize a variable motion_start_time to track motion detection time

Read the PIR sensor to check for motion

If motion is detected:

If current time - motion_start_time > 5 seconds:

Trigger the alarm

Send notification via UART to mobile device

Press reset command

Else:

Deactivate the alarm

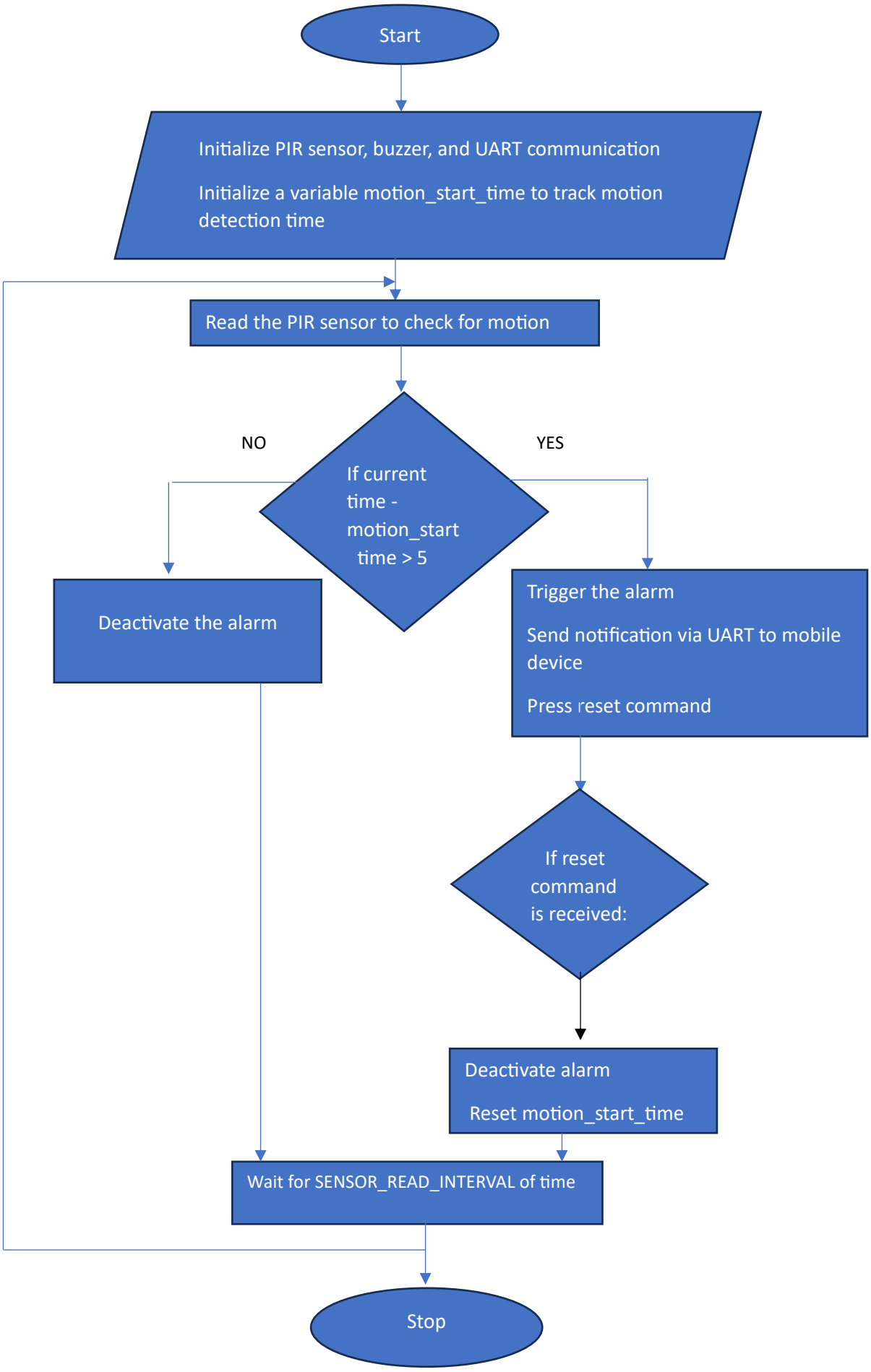
If reset command is received:

Deactivate alarm

Reset motion_start_time

Wait for the next loop iteration

FLOWCHART


Start
Initialize PIR sensor, buzzer, and UART communication
Initialize a variable motion_start_time to track motion detection time
Read the PIR sensor to check for motion
NO
YES
If current time - motion_start time > 5
Deactivate the alarm
Trigger the alarm
Send notification via UART to mobile device
Press reset command
If reset command is received:
Deactivate alarm
Reset motion_start_time
Wait for SENSOR_READ_INTERVAL of time
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, LCD display, buzzer, and SD card

Initialize an array or list to store heart rate data for one minute

Set heart_rate_sum to 0

Set count to 0

Read current heart rate from the sensor

Display the current heart rate on the LCD screen

Add current heart rate to heart_rate_sum

Add current heart rate to the array of heart rate data

Increment count

If count equals 60 :

Calculate average_heart_rate = heart_rate_sum / 60

Display average_heart_rate on the LCD screen

If average_heart_rate > 100:

Trigger alert (activate buzzer)

Log current heart rate and average heart rate to the SD card

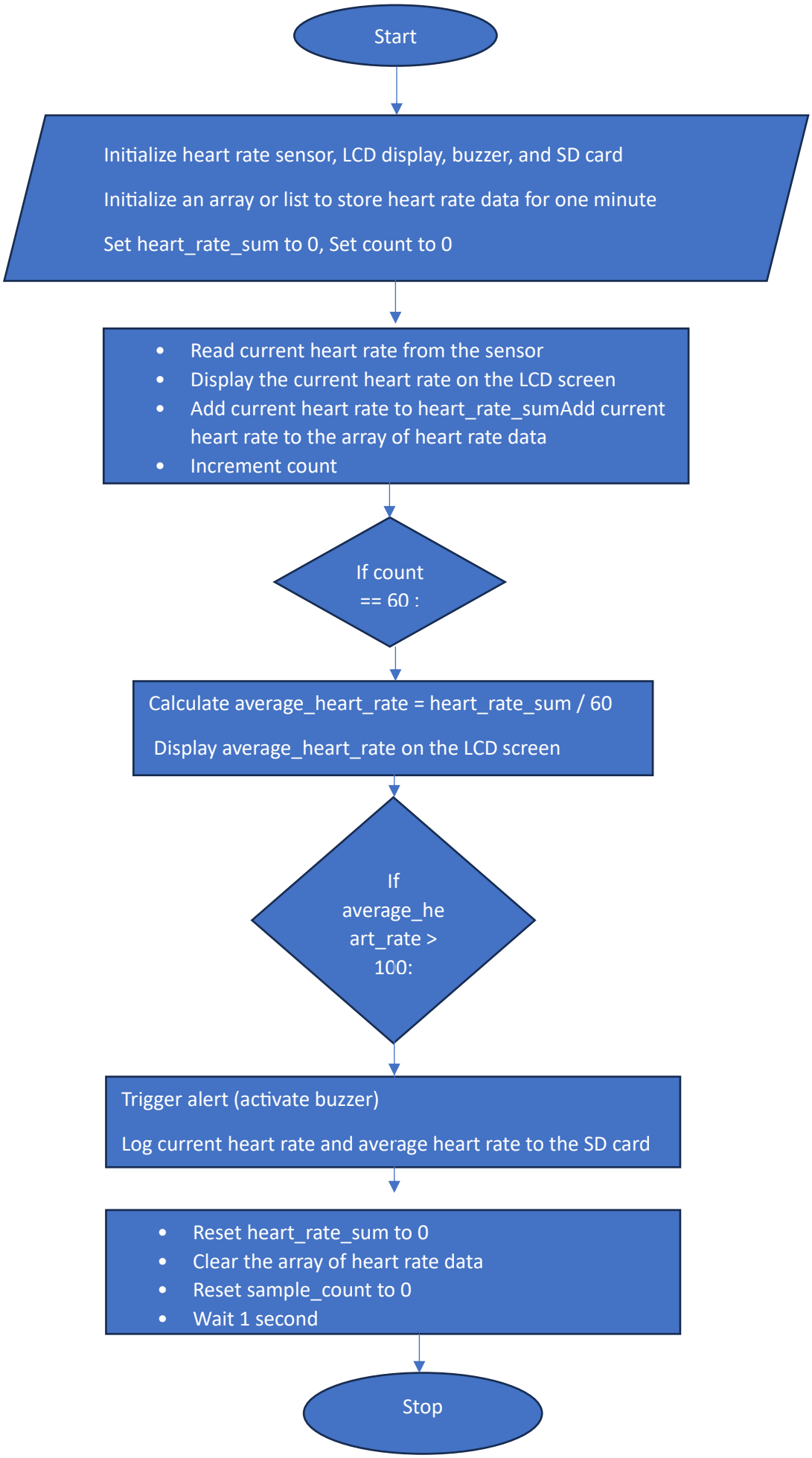
Reset heart_rate_sum to 0

Clear the array of heart rate data

Reset sample_count to 0

Wait 1 second

FLOWCHART


Start
Initialize heart rate sensor, LCD display, buzzer, and SD card
Initialize an array or list to store heart rate data for one minute
Set heart_rate_sum to 0, Set count to 0
Read current heart rate from the sensor
Display the current heart rate on the LCD screen
Add current heart rate to heart_rate_sumAdd current heart rate to the array of heart rate data
Increment count
If count == 60 :
Calculate average_heart_rate = heart_rate_sum / 60
Display average_heart_rate on the LCD screen
If average_heart_rate > 100:
Trigger alert (activate buzzer)
Log current heart rate and average heart rate to the SD card
Reset heart_rate_sum to 0
Clear the array of heart rate data
Reset sample_count to 0
Wait 1 second
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 light sensor, main LED, status LED, and manual override switch

Set threshold_light_intensity to a predefined value

Set manual_mode to False

Loop every minute:

Read light intensity from the sensor

If manual override switch is activated:

Set manual_mode to True

Blink status LED to indicate manual mode

If switch is ON, turn ON main LED

If switch is OFF, turn OFF main LED

Else:

Set manual_mode to False

Turn OFF status LED

If light intensity < threshold_light_intensity:

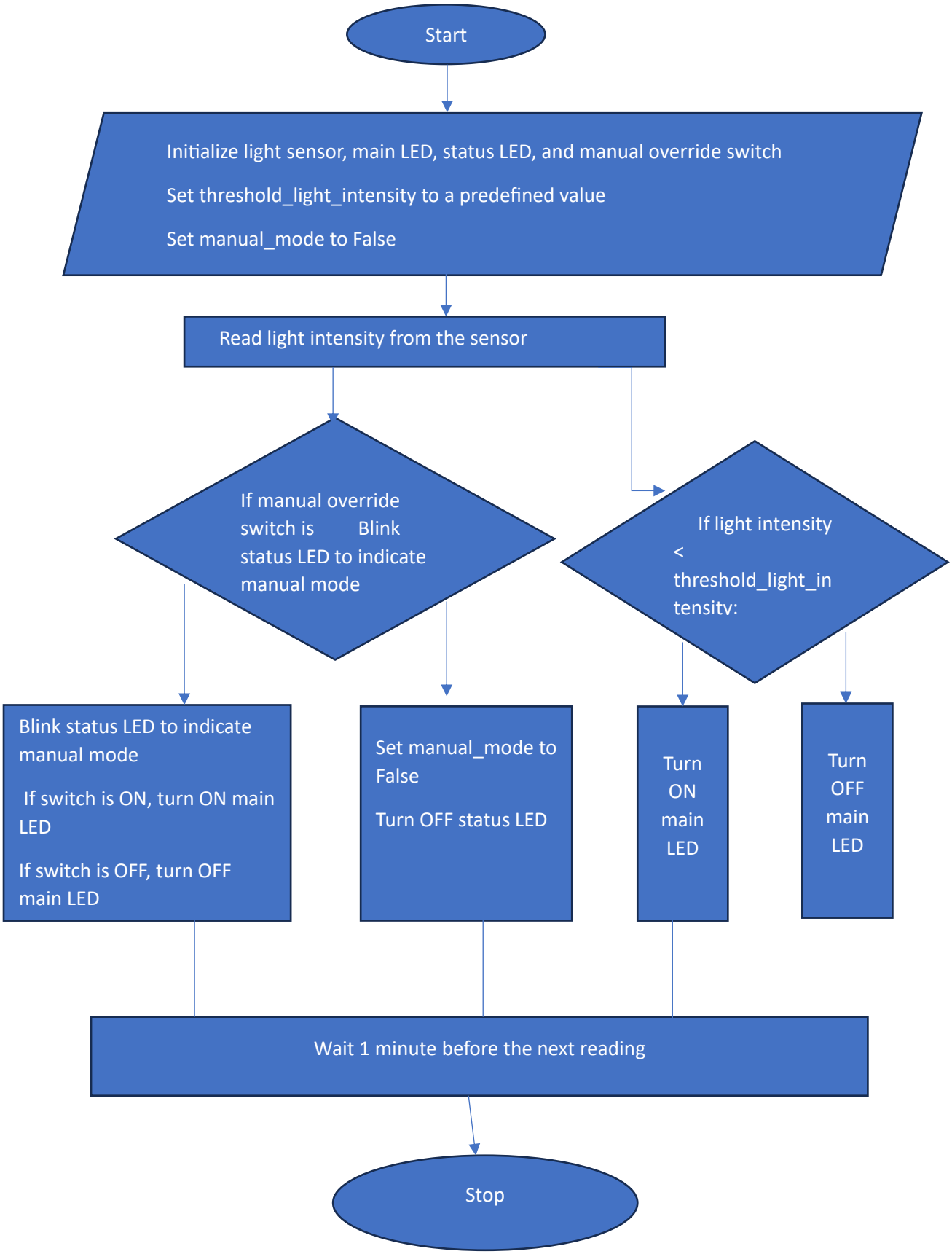
Turn ON main LED

Else:

Turn OFF main LED

Wait 1 minute before the next reading

FLOWCHART


Start
Initialize light sensor, main LED, status LED, and manual override switch
Set threshold_light_intensity to a predefined value
Set manual_mode to False
Read light intensity from the sensor
If manual override switch is Blink status LED to indicate manual mode
If light intensity < threshold_light_intensitv:
Blink status LED to indicate manual mode
If switch is ON, turn ON main LED
If switch is OFF, turn OFF main LED
Set manual_mode to False
Turn OFF status LED
Turn ON main LED
Turn OFF main LED
Wait 1 minute before the next reading
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 display, Start button, Stop button, Reset button, and SD card

Set elapsed_time to 0 seconds

Set stopwatch_running to False

Set 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

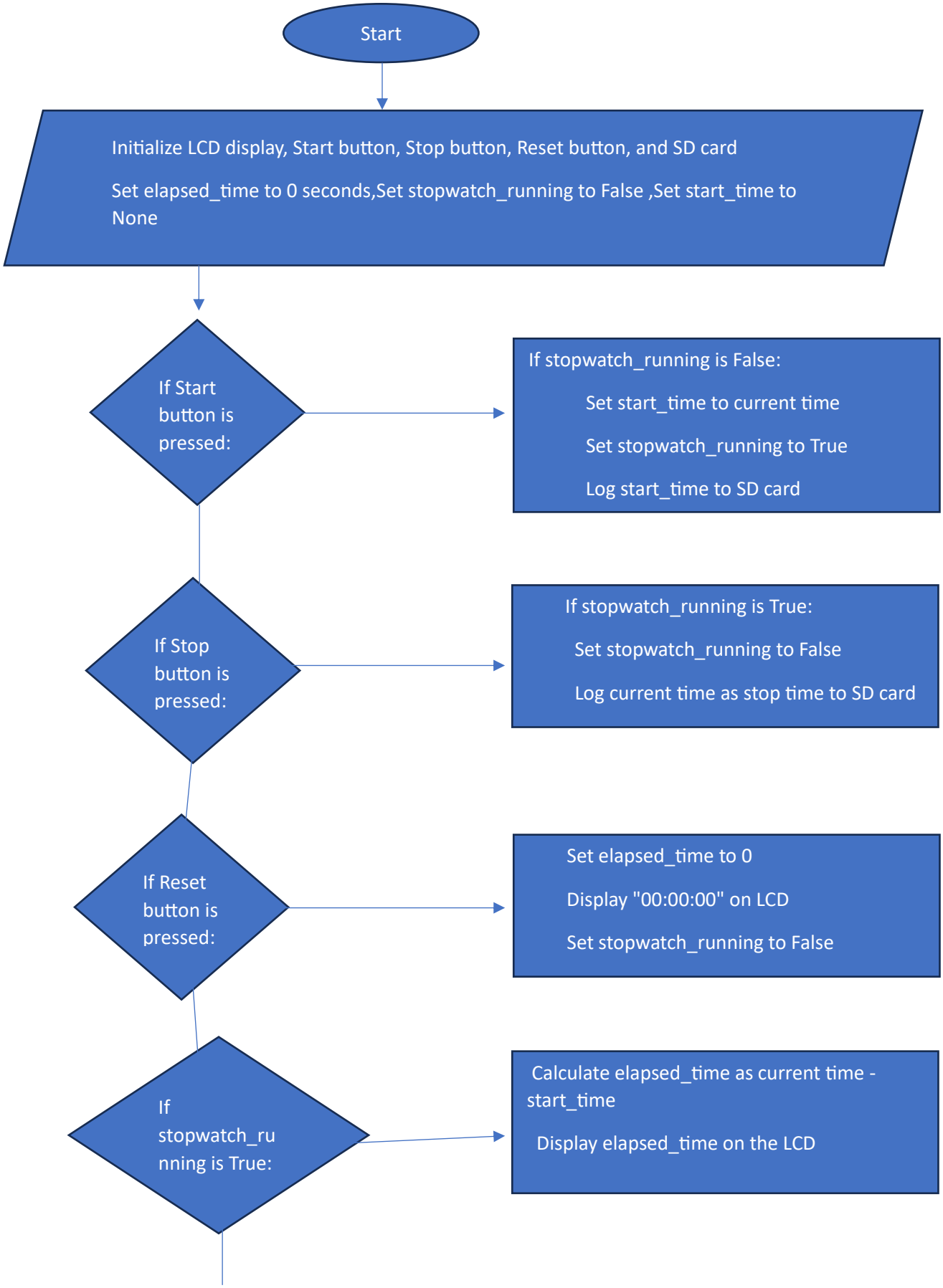
If stopwatch_running is True:

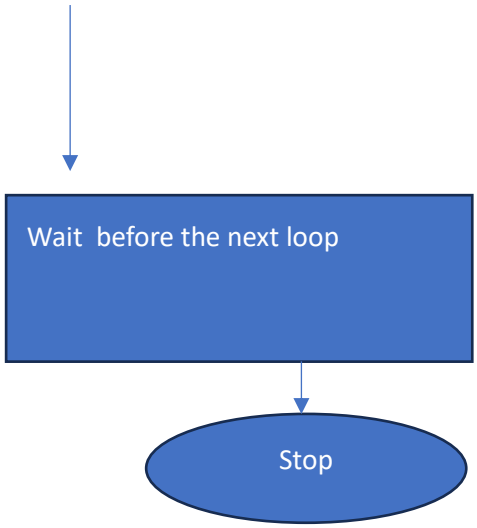
    Calculate elapsed_time as current time - start_time

    Display elapsed_time on the LCD

Wait before the next loop

FLOWCHART


Start
Initialize LCD display, Start button, Stop button, Reset button, and SD card
Set elapsed_time to 0 seconds,Set stopwatch_running to False ,Set 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
If stopwatch_running is True:
Calculate elapsed_time as current time - start_time
Display elapsed_time on the LCD


Wait before the next loop
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 temperature sensor, storage (array or log file), and display

Set logging_interval to 10 minutes

Initialize an empty array or open a log file for temperature data storage

Try to read temperature from the sensor

If sensor read is successful:

Get the current timestamp

Store the temperature reading and timestamp in the array

Else:

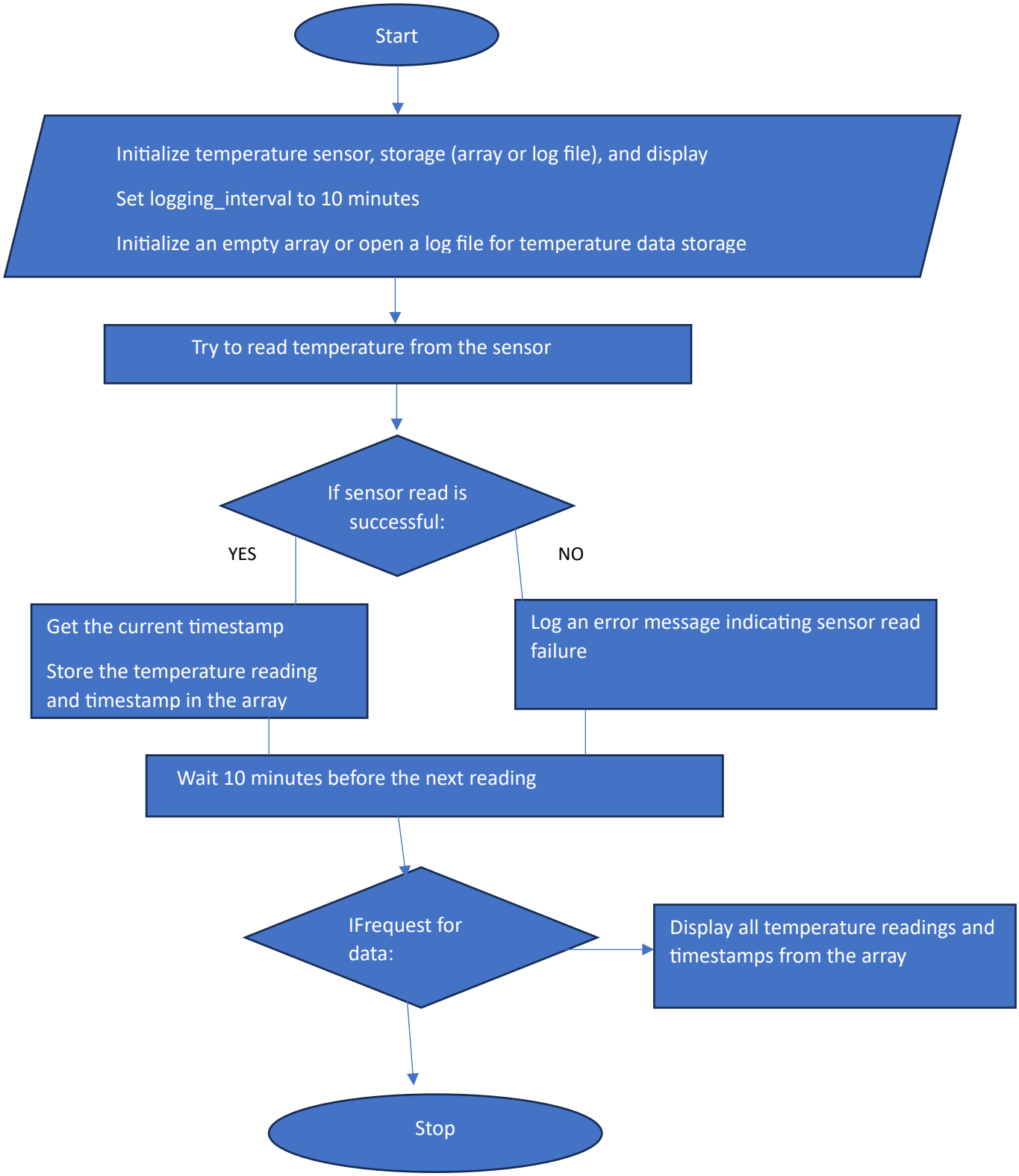
Log an error message indicating sensor read failure

Wait 10 minutes before the next reading

IF Function request for data:

Display all temperature readings and timestamps from the array

FLOWCHART


Start
Initialize temperature sensor, storage (array or log file), and display
Set logging_interval to 10 minutes
Initialize an empty array or open a log file for temperature data storage
Try to read temperature from the sensor
If sensor read is successful:
YES
NO
Get the current timestamp
Store the temperature reading and timestamp in the array
Log an error message indicating sensor read failure
Wait 10 minutes before the next reading
IFrequest for data:
Display all temperature readings and timestamps from the array
Stop

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, motors, speed control, and status LEDs

Define commands: "FORWARD", "BACKWARD", "LEFT", "RIGHT", "STOP", and "SPEED:<value>"

Set default speed to a moderate level

If command is "FORWARD":

Set robot_state to "MOVING FORWARD"

Turn on LED to indicate moving

Move robot forward at the current speed

Else if command is "BACKWARD":

Set robot_state to "MOVING BACKWARD"

Turn on LED to indicate moving

Move robot backward at the current speed

Else if command is "LEFT":

Set robot_state to "TURNING LEFT"

Turn on LED to indicate moving

Turn robot left at the current speed

Else if command is "RIGHT":

Set robot_state to "TURNING RIGHT"

Turn on LED to indicate moving

Turn robot right at the current speed

Else if command is "STOP":

Set robot_state to "STOPPED"

Turn off LED to indicate stopped

Stop all robot movement

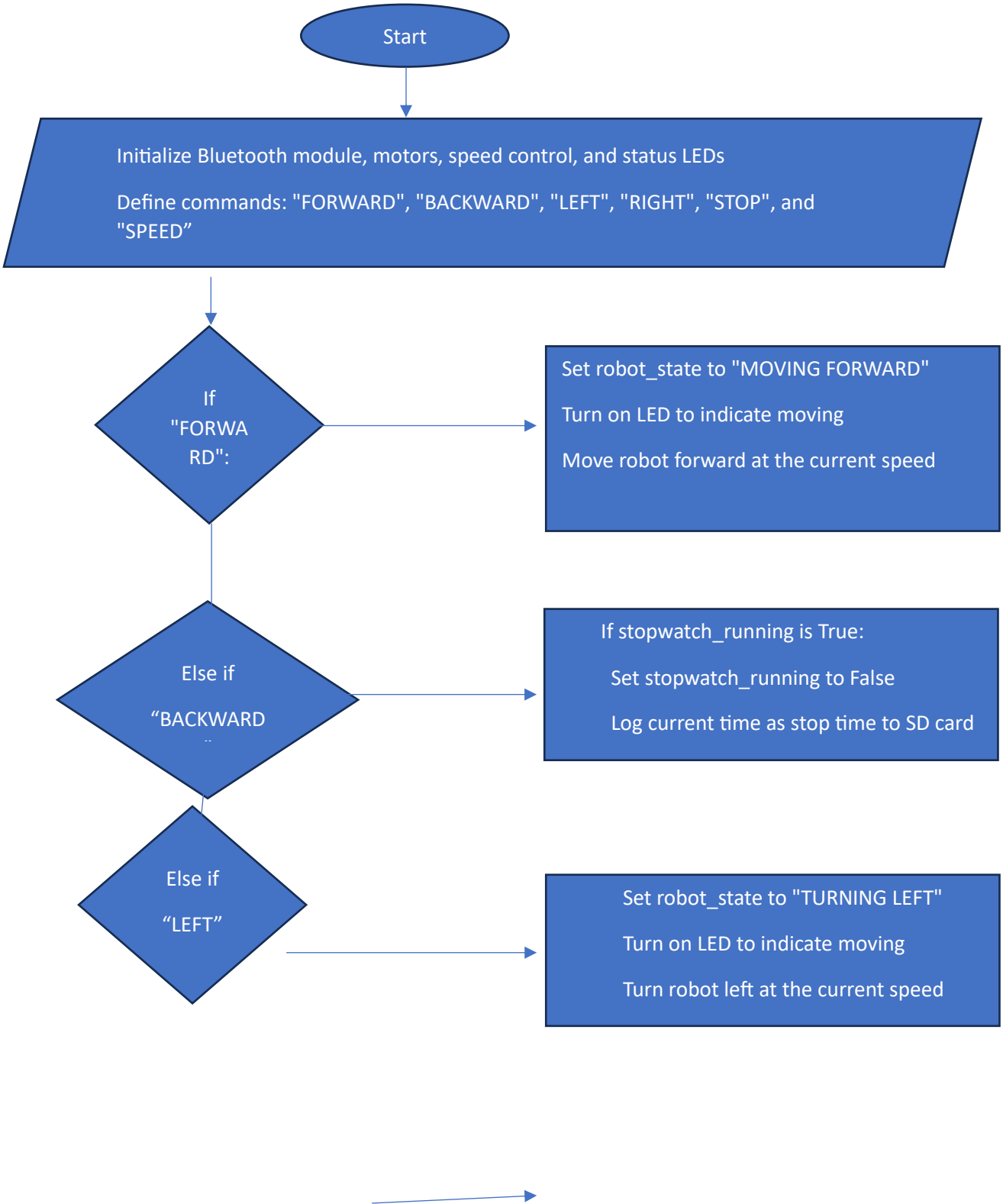
Else if command starts with "SPEED:":

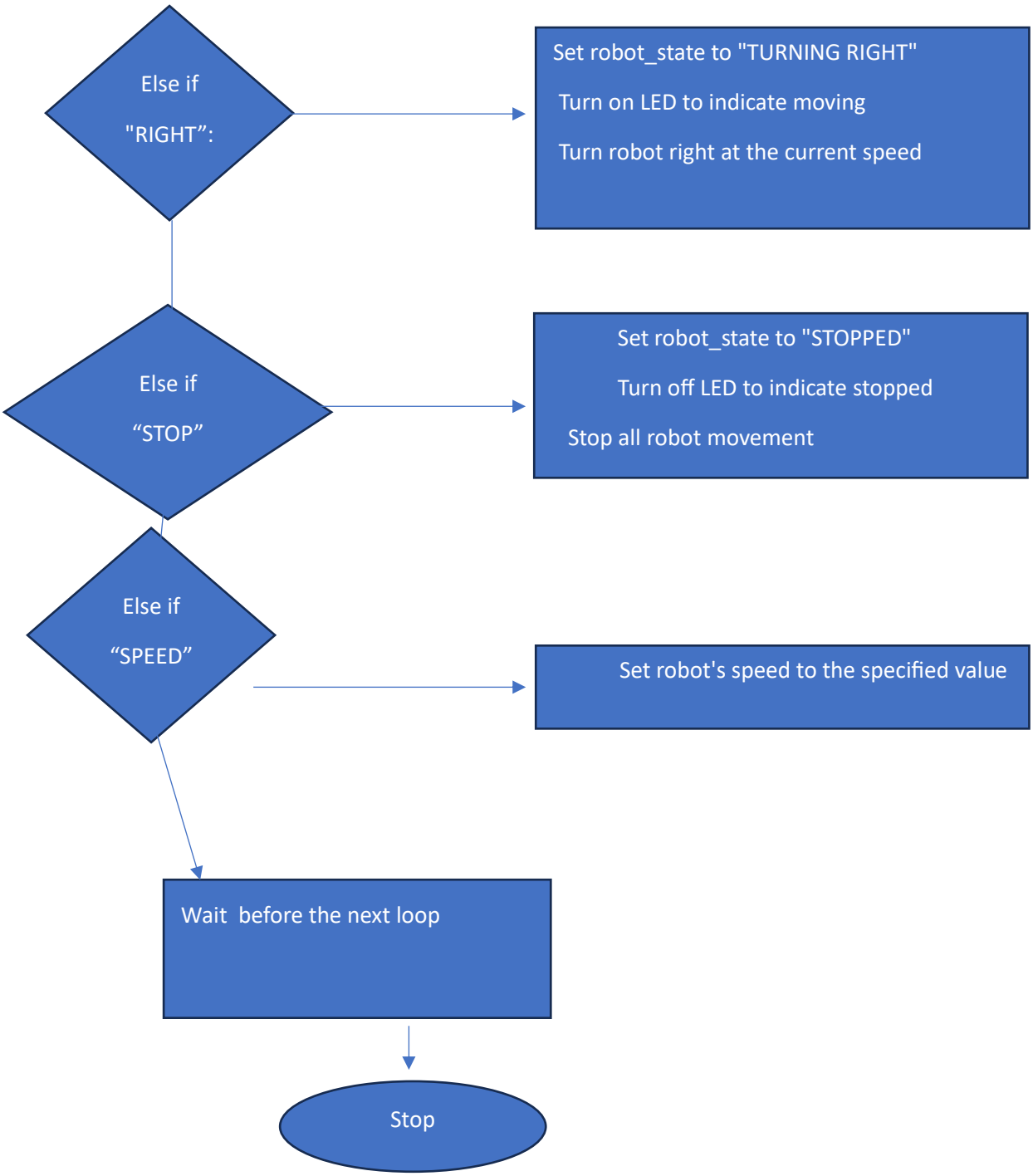
Set robot's speed to the specified value

Display current robot_state on LED

Wait before checking for the next command

FLOWCHART




Else if
"RIGHT":
Set robot_state to "TURNING RIGHT"
Turn on LED to indicate moving
Turn robot right at the current speed
Else if
“STOP”
Set robot_state to "STOPPED"
Turn off LED to indicate stopped
Stop all robot movement
Else if
“SPEED”
Set robot's speed to the specified value
Wait before the next loop
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 for battery voltage measurement, buzzer, LCD display, and memory for logging

Set low_voltage_threshold to 11V

Set power_saving_mode to False

  Measure battery voltage using ADC

  Display the current battery voltage on the LCD screen

  If battery voltage < low_voltage_threshold:

    Trigger alert by activating buzzer

    Log low voltage event with timestamp to memory

  If power_saving_mode is enabled:

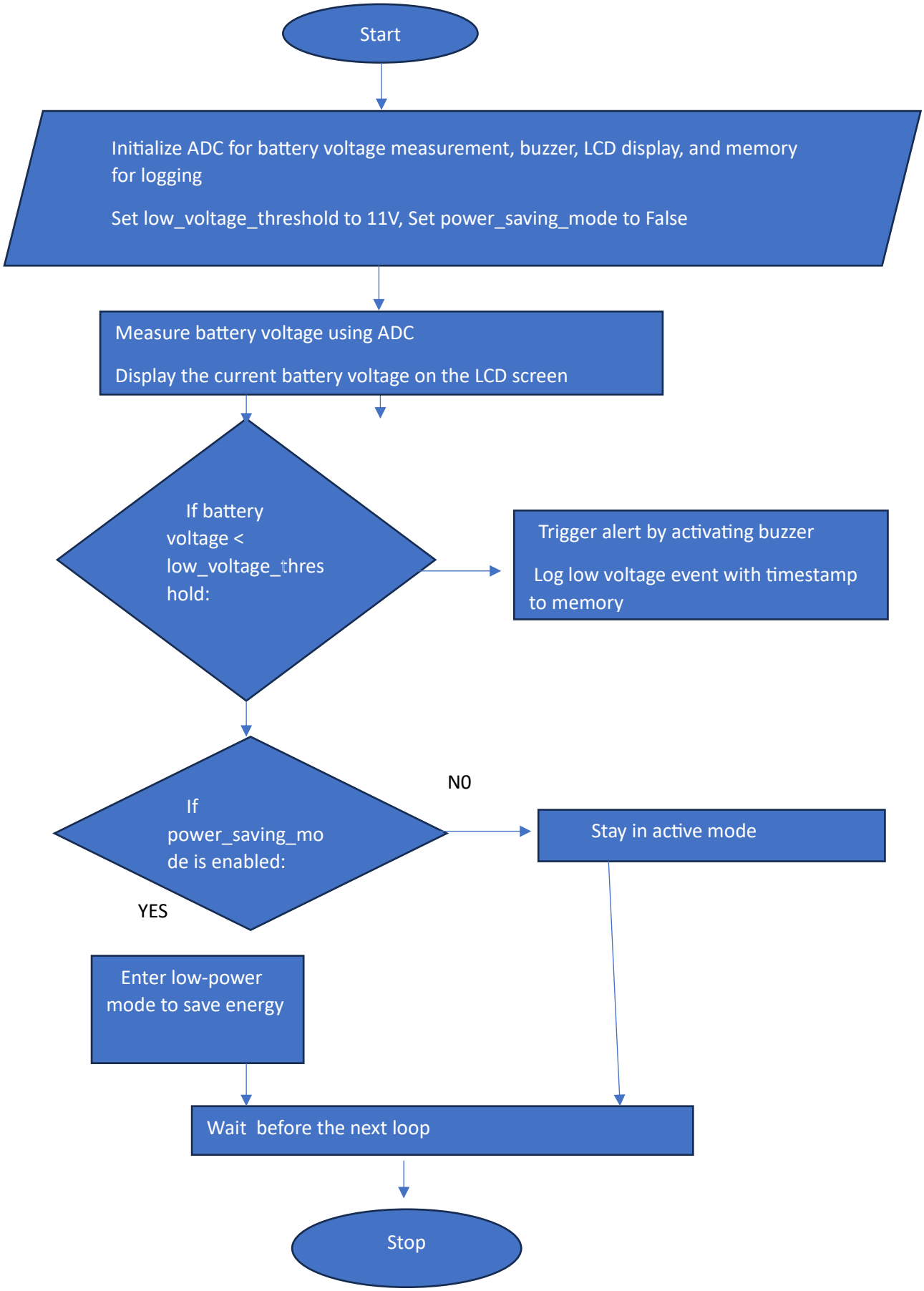
    Enter low-power mode to save energy

  Else:

    Stay in active mode

  Wait 1 minute before the next loop iteration

FLOWCHART


Start
Initialize ADC for battery voltage measurement, buzzer, LCD display, and memory for logging
Set low_voltage_threshold to 11V, Set power_saving_mode to False
Measure battery voltage using ADC
Display the current battery voltage on the LCD screen
If battery voltage < low_voltage_threshold:
Trigger alert by activating buzzer
Log low voltage event with timestamp to memory
If power_saving_mode is enabled:
N0
Stay in active mode
YES
Enter low-power mode to save energy
Wait before the next loop
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.

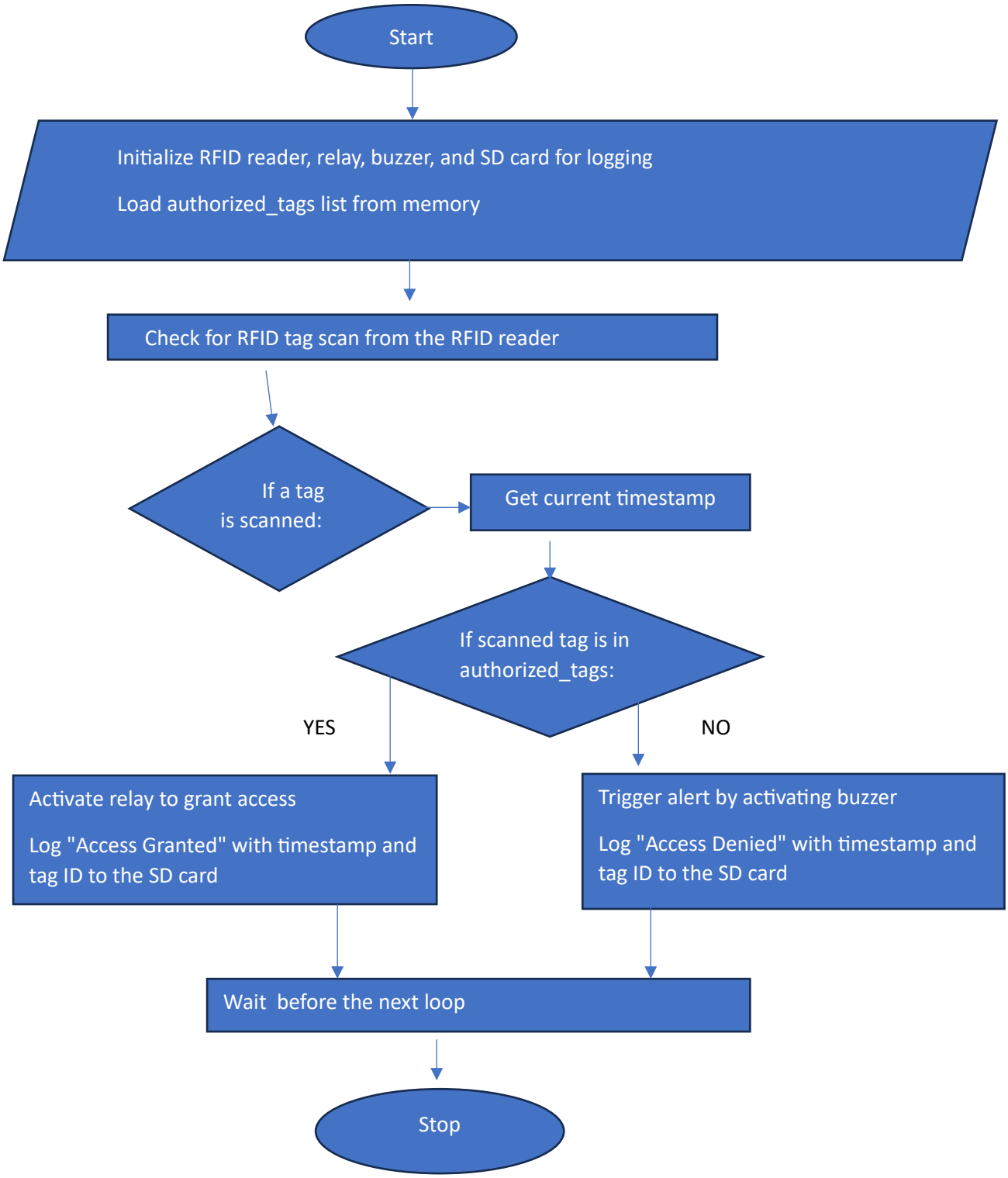
Requirements:

- 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, and SD card for logging
Load authorized_tags list from memory
  Check for RFID tag scan from the RFID reader
  If a tag is scanned:
    Get current timestamp
    If scanned tag is in authorized_tags:
      Activate relay to grant access
      Log "Access Granted" with timestamp and tag ID to the SD card
    Else:
      Trigger alert by activating buzzer
      Log "Access Denied" with timestamp and tag ID to the SD card
  Wait before checking for the next tag scan
```

FLOWCHART


Start
Initialize RFID reader, relay, buzzer, and SD card for logging
Load authorized_tags list from memory
Check for RFID tag scan from the RFID reader
If a tag is scanned:
Get current timestamp
If scanned tag is in authorized_tags:
YES
NO
Activate relay to grant access
Log "Access Granted" with timestamp and tag ID to the SD card
Trigger alert by activating buzzer
Log "Access Denied" with timestamp and tag ID to the SD card
Wait before the next loop
Stop