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

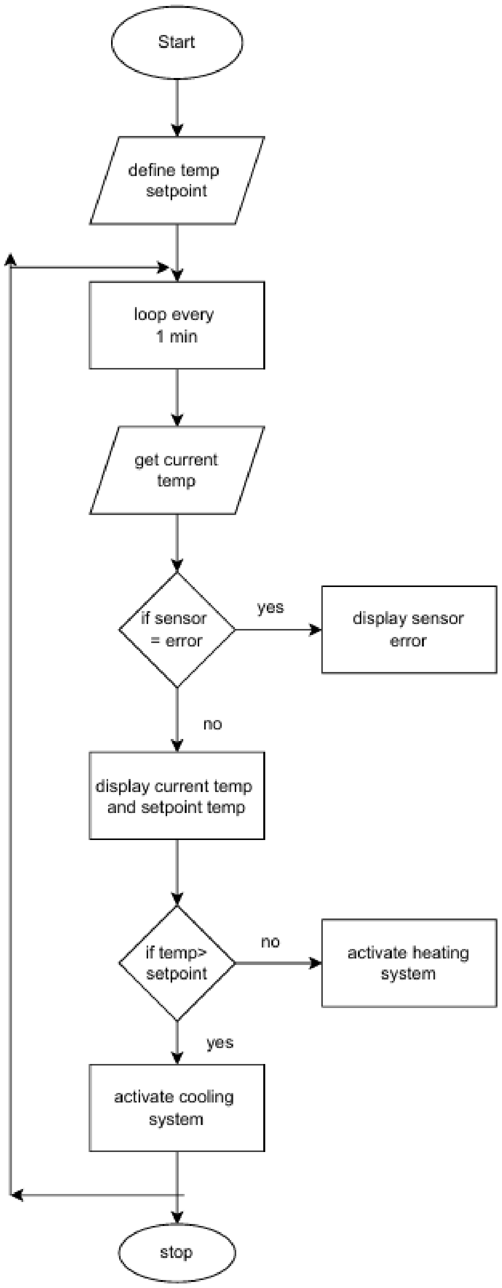
```
1-setpoint=user defined temperature

2-Loop every 1 minute:

        current= current temperature
        If (sensor=error)
             print "Sensor error"
        Else:
          print(current,setpoint)
          If (current> setpoint)
            Activate cooling system
          Else
            Activate heating system

3-end loop
```

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.

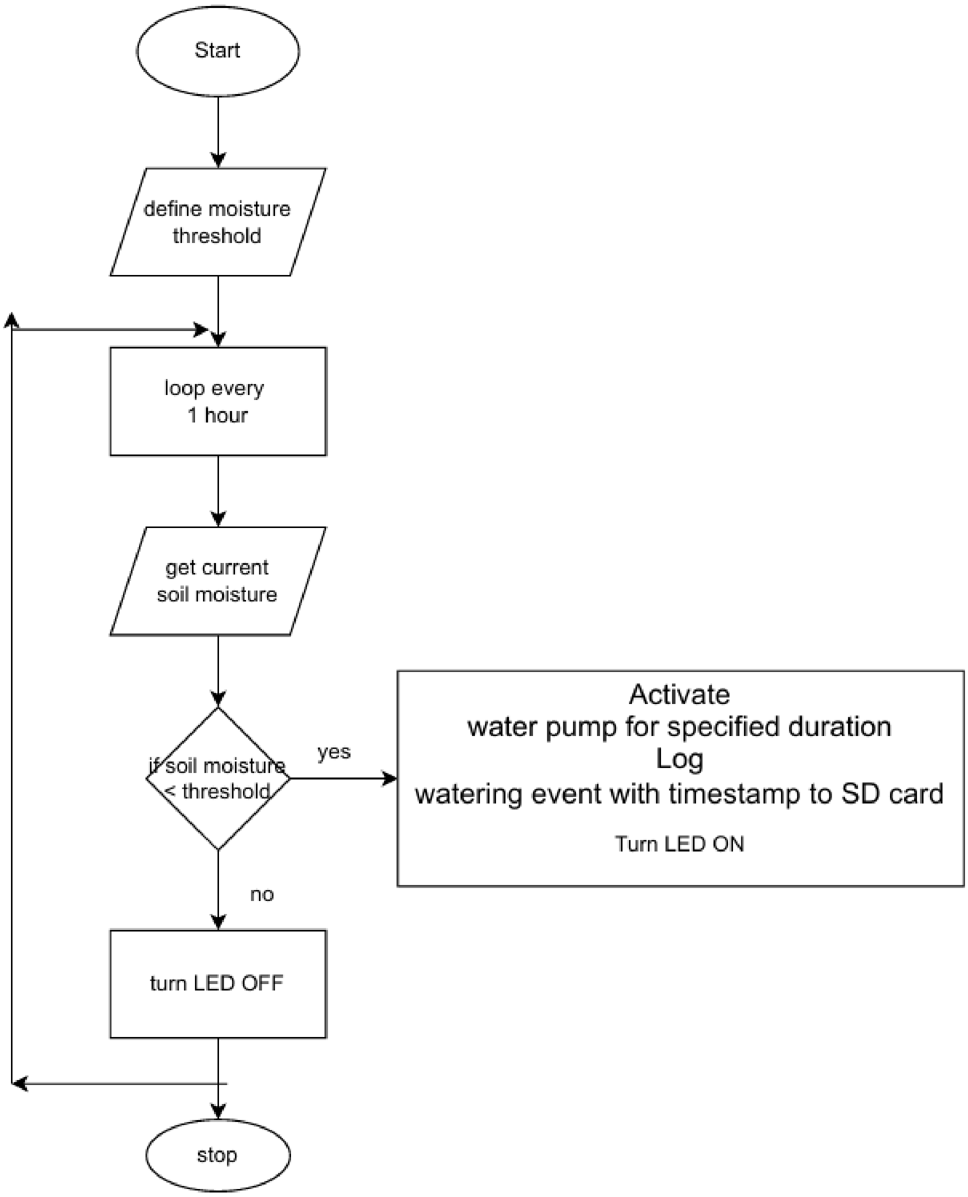
- Log the watering events with timestamps to an SD card.

- Provide feedback through an LED indicator (e.g., LED ON when watering).

Pseudocode

```
1-threshold=moisture threshold
2-Loop every 1 hour:
        soil moisture =level of soil moisture from sensor
        If (soil moisture level < threshold)
                Activate water pump for specified duration
                Log watering event with timestamp to SD card
                Turn LED ON
        Else:
                Turn LED OFF
3-End Loop
```

Flowchart


Start
define moisture threshold
loop every 1 hour
get current soil moisture
if soil moisture < threshold
yes
Activate
water pump for specified duration
Log
watering event with timestamp to SD card
Turn LED ON
no
turn LED OFF
stop

3). Motion Detection Alarm System

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

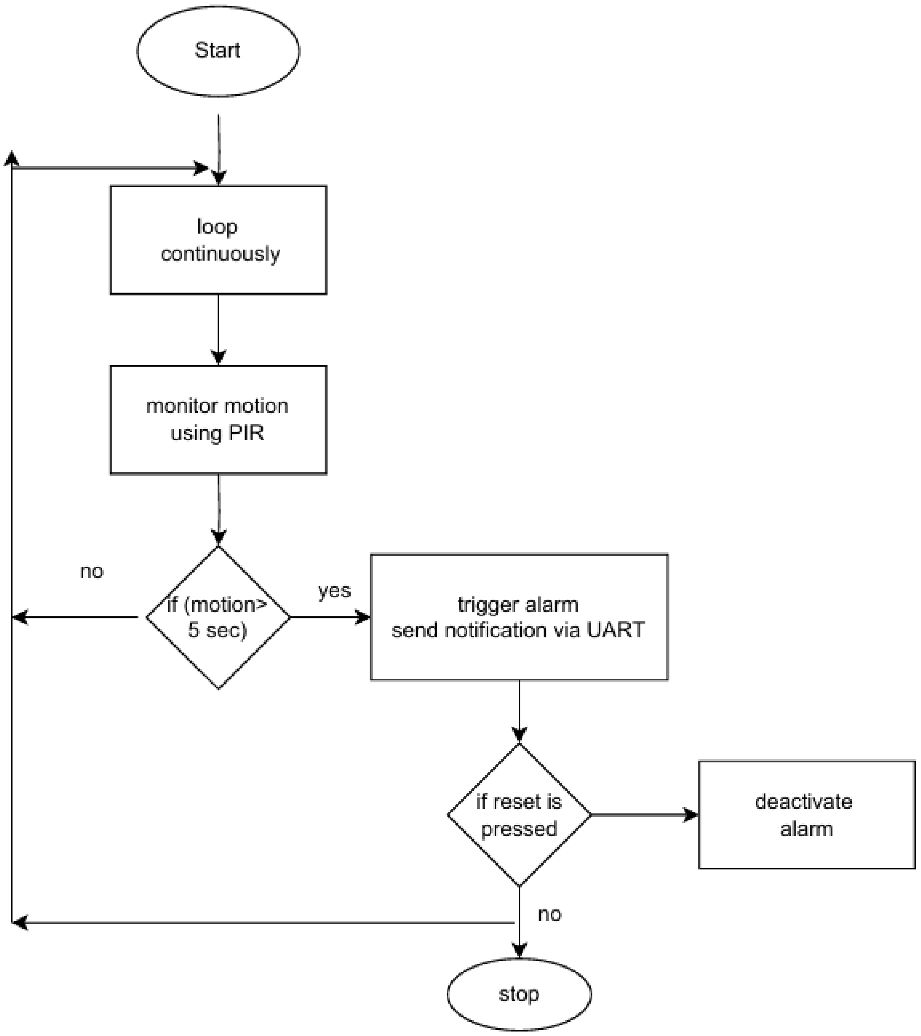
Requirements: • Continuously monitor motion detection status. • If motion is detected for more than 5 seconds, trigger an alarm (buzzer). • Send a notification to a mobile device via UART communication. • Include a reset mechanism to deactivate the alarm.

Pseudocode

1-Loop continuously:

```
2-Monitor motion detection using PIR sensor
  If (motion > 5 seconds):
       Trigger alarm
        Send notification via UART
        If reset button pressed:
              Deactivate alarm
3-End Loop
```

Flowchart



4). Heart Rate Monitor

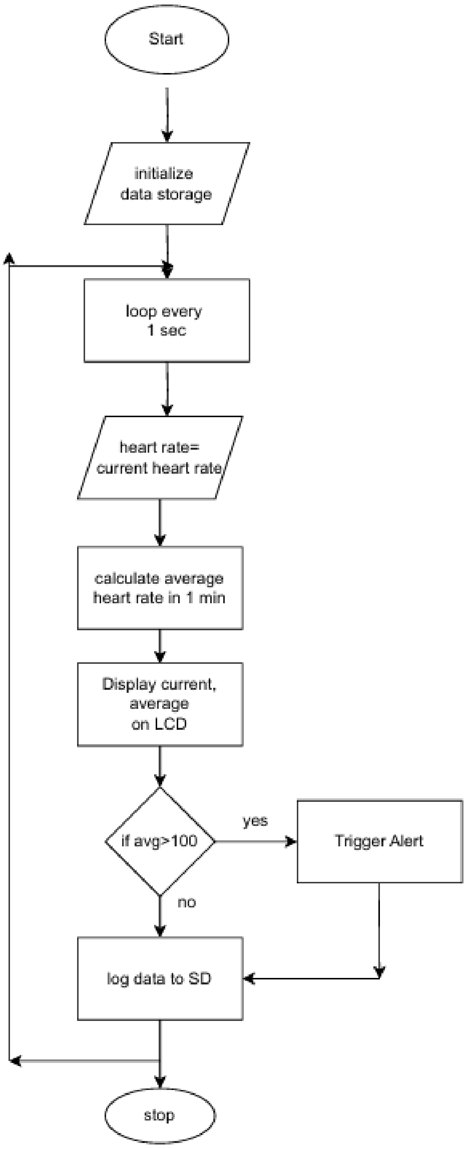
Problem Statement: Implement a heart rate monitoring application that reads data from a heart rate sensor.

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

```
1-Initialize heart rate data storage
2-Loop every 1 second:
    heart rate=current heart rate
    Calculate average heart rate over last 1 minute
    Display current and average heart rate on LCD
    If (heart rate > 100 bpm):
        Trigger alert (activate buzzer)
    Log heart rate data to SD card
3-End Loop
```

Flowchart


Start
initialize data storage
loop every 1 sec
heart rate= current heart rate
calculate average heart rate in 1 min
Display current, average on LCD
if avg>100
yes
Trigger Alert
no
log data to SD
stop

5.) LED Control Based on Light Sensor

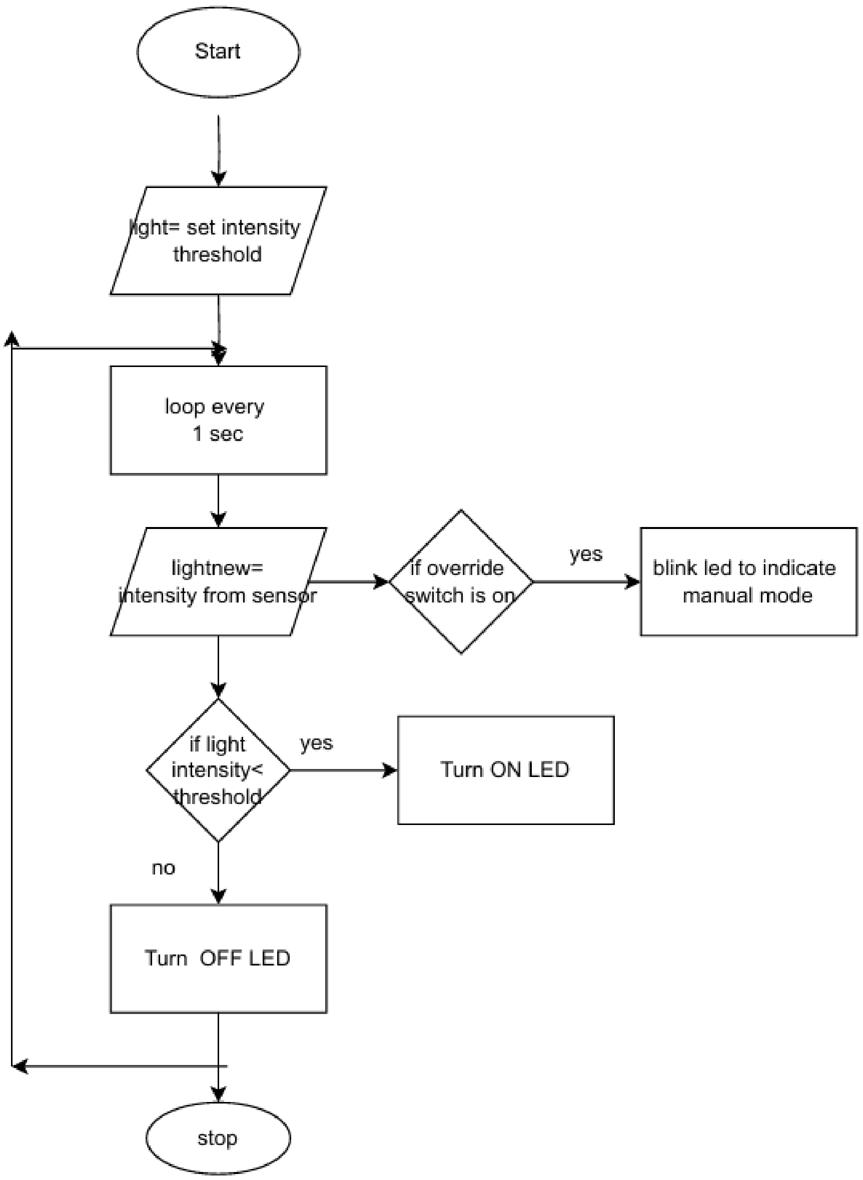
Problem Statement: Create an embedded application that controls an LED based on ambient light levels detected by a light sensor.

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

```
1-light =intensity threshold
2-Loop every 1 minute:
   Lightnew= intensity from sensor
   If manual override switch = ON:
         Blink status LED to indicate manual mode
   Else If light intensity < threshold:
      Turn ON LED
   Else:
      Turn OFF LED
3-End Loop
```

Flowchart


Start
light= set intensity threshold
loop every 1 sec
lightnew= intensity from sensor
if override switch is on
yes
blink led to indicate manual mode
if light intensity< threshold
yes
Turn ON LED
no
Turn OFF LED
stop

6.) Digital Stopwatch

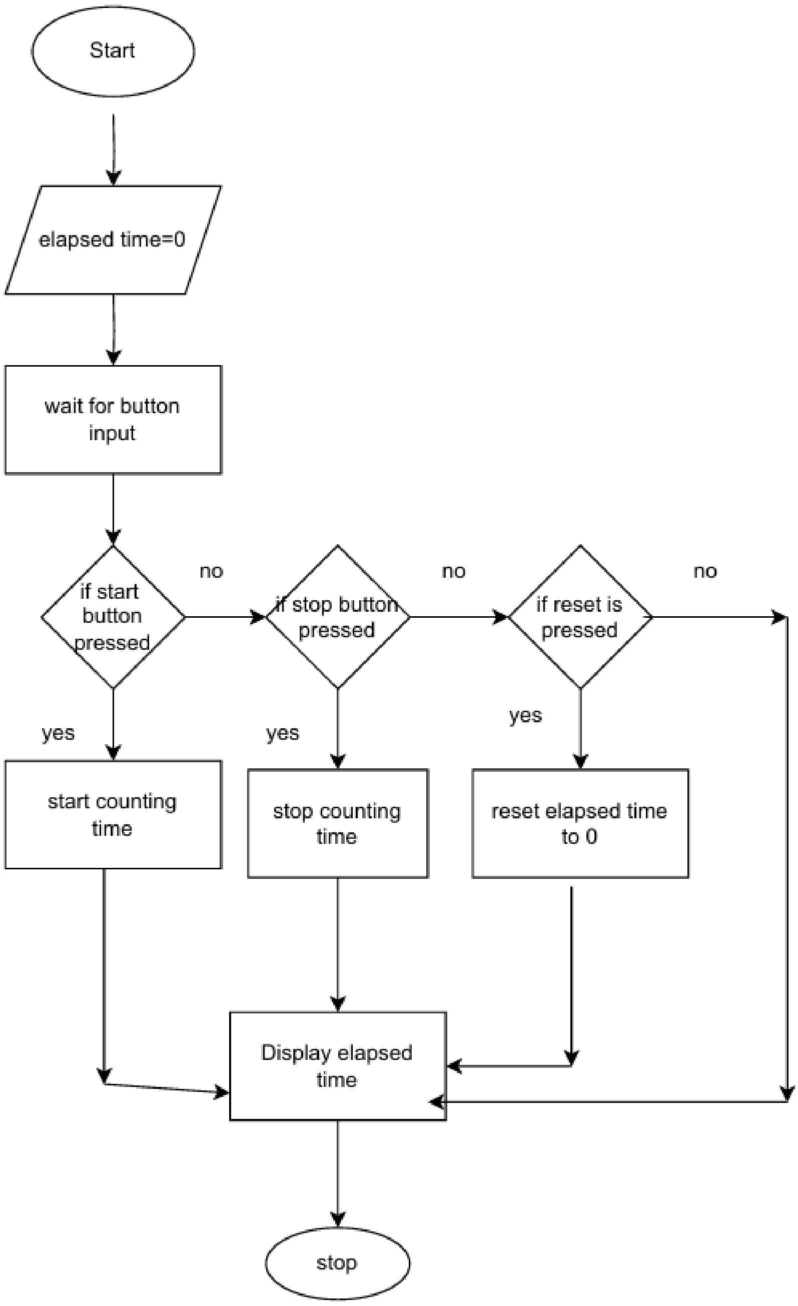
Problem Statement: Design a digital stopwatch application that can start, stop, and reset using button inputs.

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

```
1-Start
2- elapsed time = 0
3-Wait for button input:
  If Start button pressed:
    Start counting time
  If Stop button pressed:
    Stop counting time
    Log start and stop times to SD card
  If Reset button pressed:
    Reset elapsed time to 0
  Display elapsed time on LCD in hours, minutes, seconds
4-End
```

Flowchart



7. Temperature Logging System

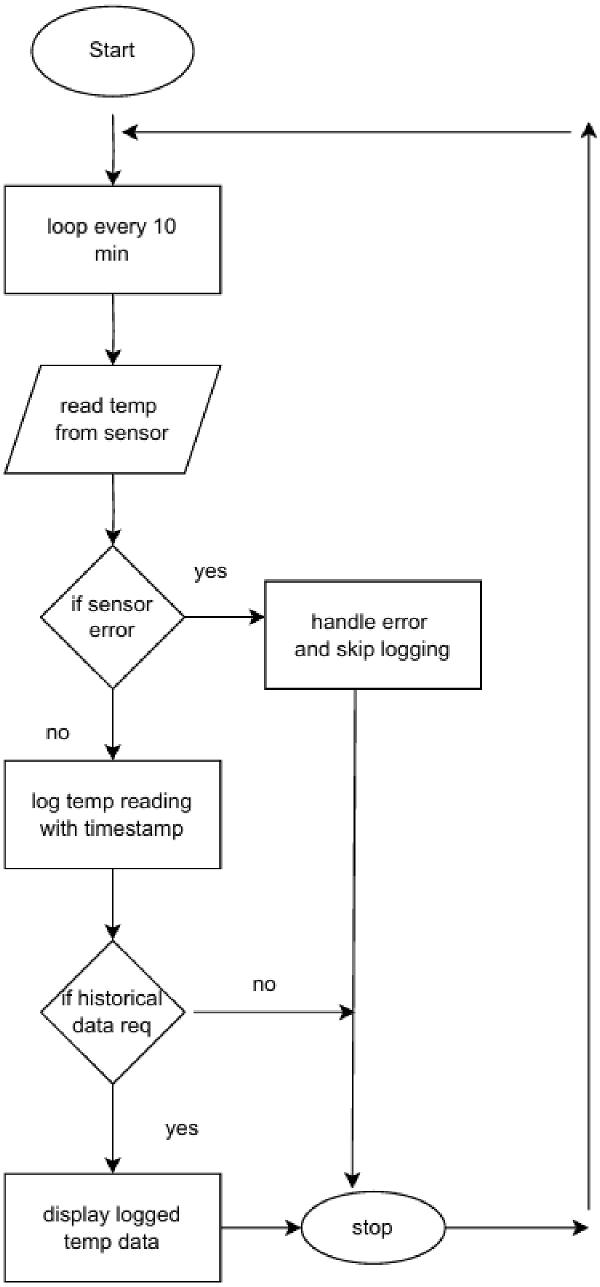
Problem Statement: Implement a temperature logging system that records temperature data at regular intervals.

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

```
1-Start
2-Loop every 10 minutes:
    Read temperature from sensor
    If sensor error, handle error and skip logging
    Else:
        Log temperature reading with timestamp
    If historical data requested:
        Display logged temperature data
3-End Loop
4-End
```

Flowchart



8. Bluetooth Controlled Robot

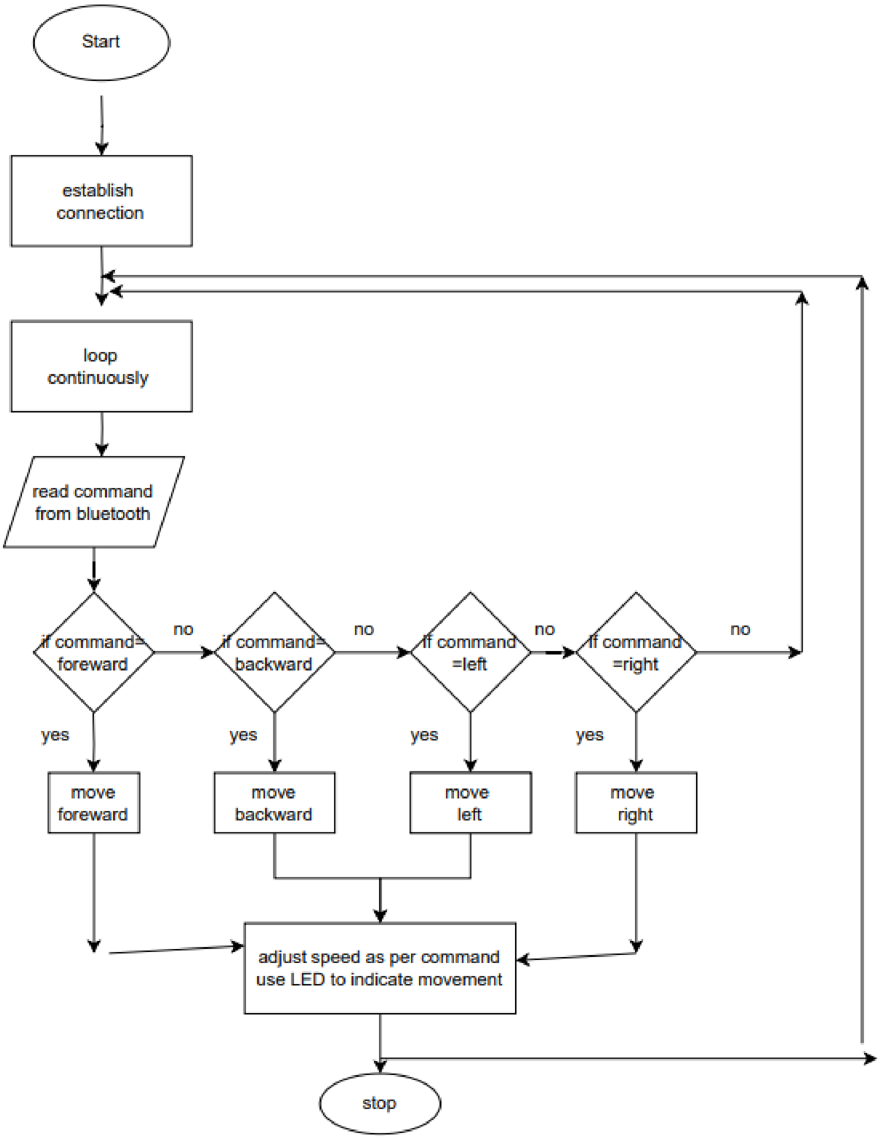
Problem Statement: Create an embedded application for controlling a robot via Bluetooth commands.

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

```
1-Start
2-Establish Bluetooth connection
3-Loop continuously:
    Read command from Bluetooth
   If command is 'forward':
      Move = forward
   Else If command is 'backward':
      Move =backward
   Else If command is 'left':
      move =left
   Else If command is 'right':
      Move=right
   Adjust speed as per command
   Use LEDs to indicate movement state
4-End Loop
5-End
```

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:

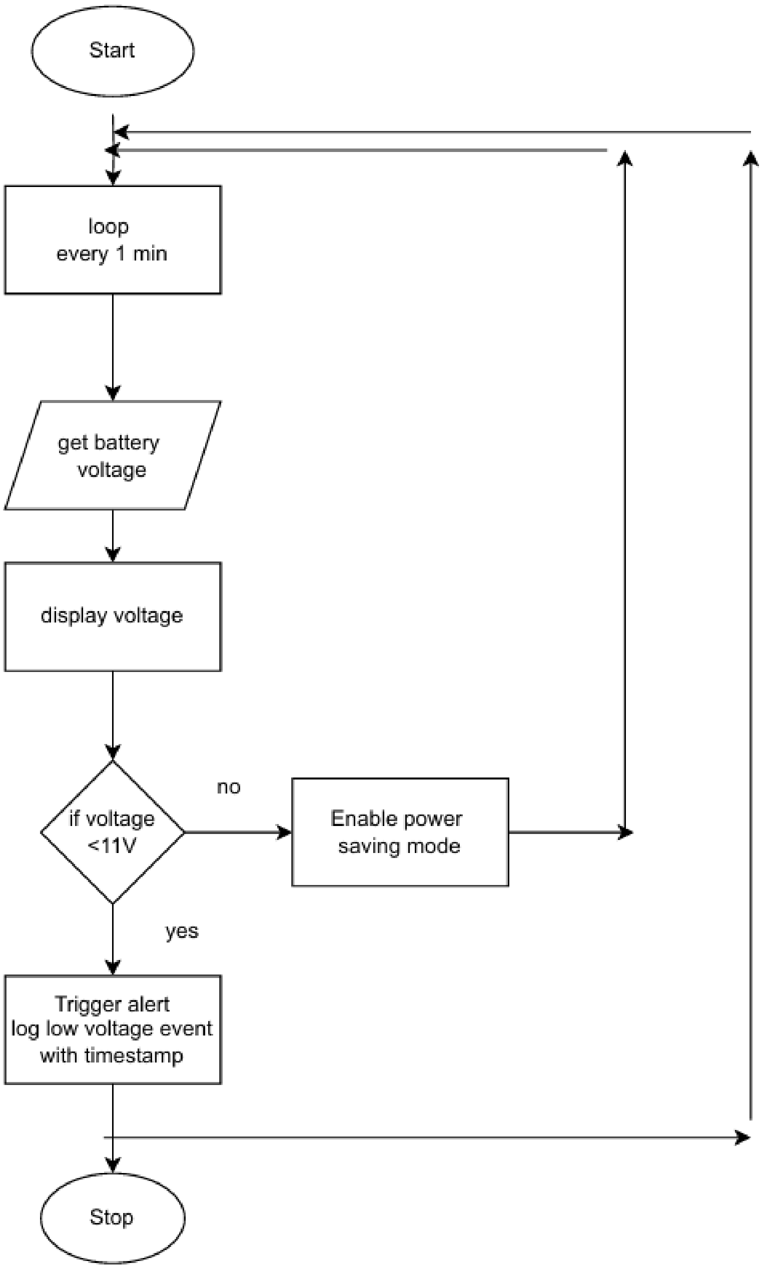
- Measure battery voltage every minute using an ADC (Analog-to-Digital Converter).
- If voltage falls below 11V, trigger an alert (buzzer) and log the event to memory.

- Display current voltage on an LCD screen continuously.
- Implement power-saving features to reduce energy consumption during idle periods.

Pseudocode

```
1-Start
2-Loop every 1 minute:
    Measure battery voltage using ADC
    Display current voltage on LCD
    If voltage < 11V:
        Trigger alert (activate buzzer)
        Log low voltage event with timestamp
    Enable power-saving during idle periods
3-End Loop
4-End
```

Flowchart



10. RFID-Based Access Control System

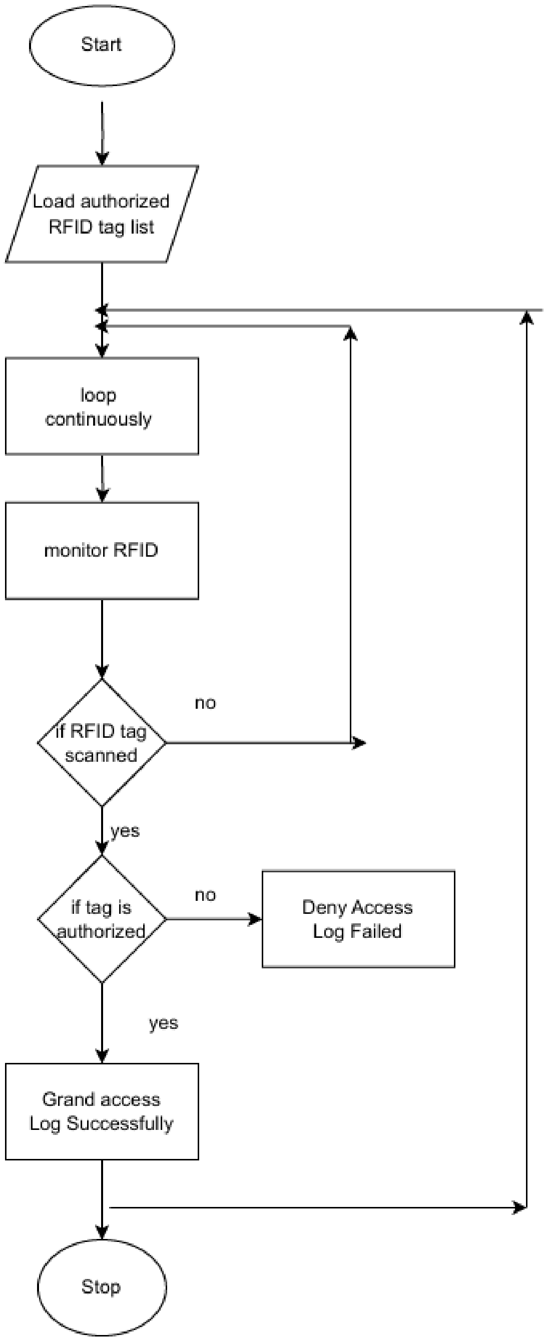
Problem Statement: Design an access control system using RFID technology to grant or deny access based on scanned RFID tags.

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-Start
2-Load authorized RFID tag list
3-Loop continuously:
  Monitor RFID reader for scans
  If RFID tag scanned:
    If tag is authorized:
      Grant access (activate relay)
      Log successful access attempt with timestamp
    Else:
      Deny access (activate buzzer)
      Log failed access attempt with timestamp
4-End Loop
5-End
```

Flowchart


Start
Load authorized RFID tag list
loop continuously
monitor RFID
if RFID tag scanned
no
yes
if tag is authorized
no
Deny Access Log Failed
yes
Grand access Log Successfully
Stop