**1.** Smart Home Temperature Control

Problem Statement:

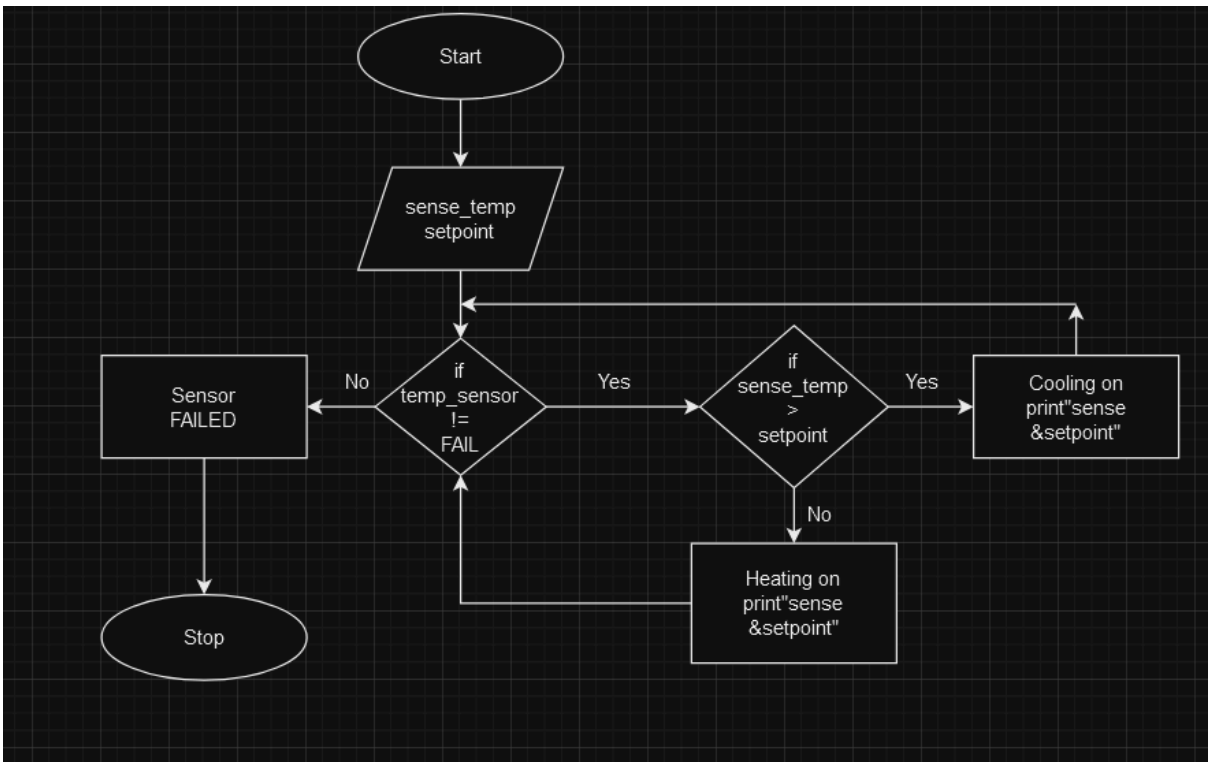
Design a temperature control system for a smart home. The system should read the current temperature from a sensor every minute and compare it to a user-defined setpoint.

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.

ANS::

1.Take Sensor Temperature as sense_temp
2.Take user defined setpoint as setpoint
3.for every time==1minute
    3.1 if (Temp_Sensor!=FAIL)
        3.1.1 if (sense_temp > setpoint)
        3.1.1.1 Cooling_system=ON;
        3.1.1.2 print "sense_temp & setpoint";
    3.1.2 else
        3.1.2.1 Heating_system=ON;
        3.1.2.2 print "sense_temp & setpoint";
    3.2 else
        3.2.1 print "SENSOR FAILED";

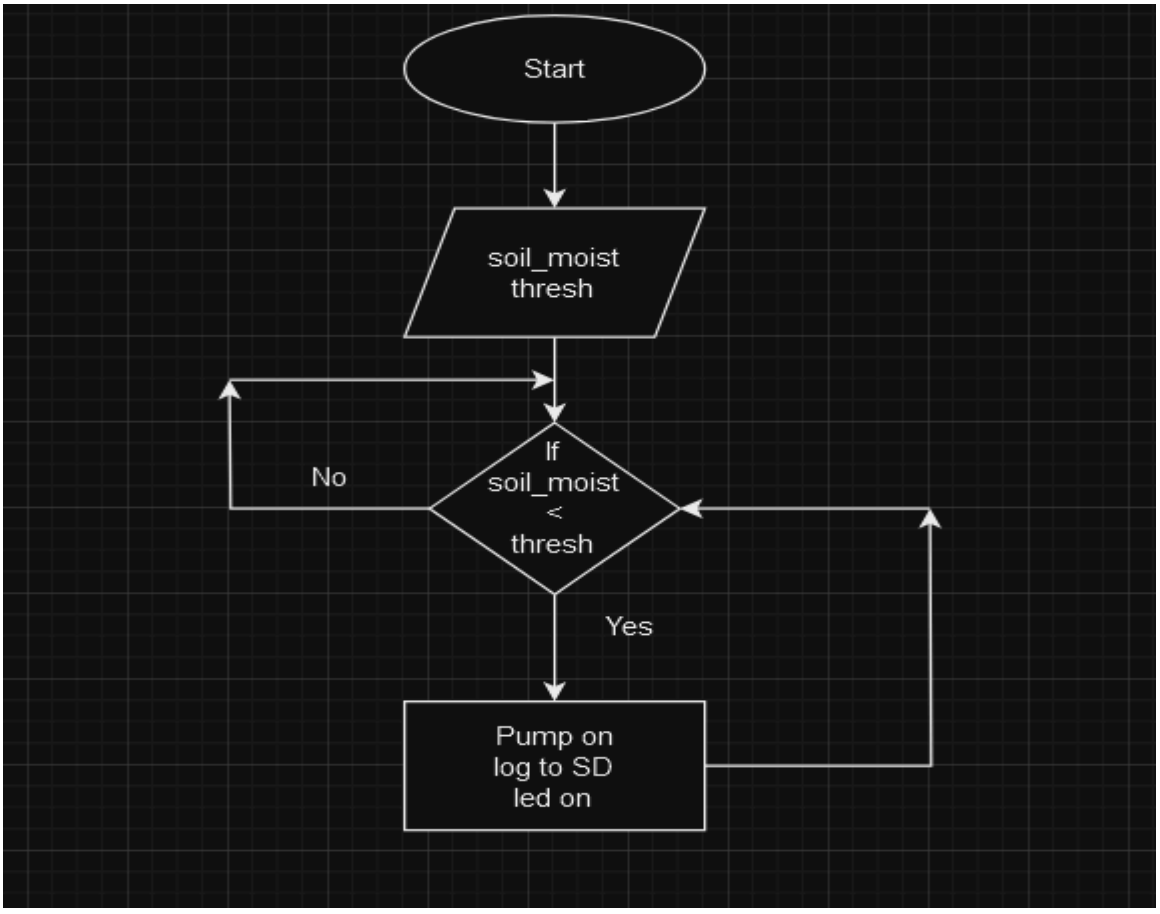



**2.** Automated Plant Watering System

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).

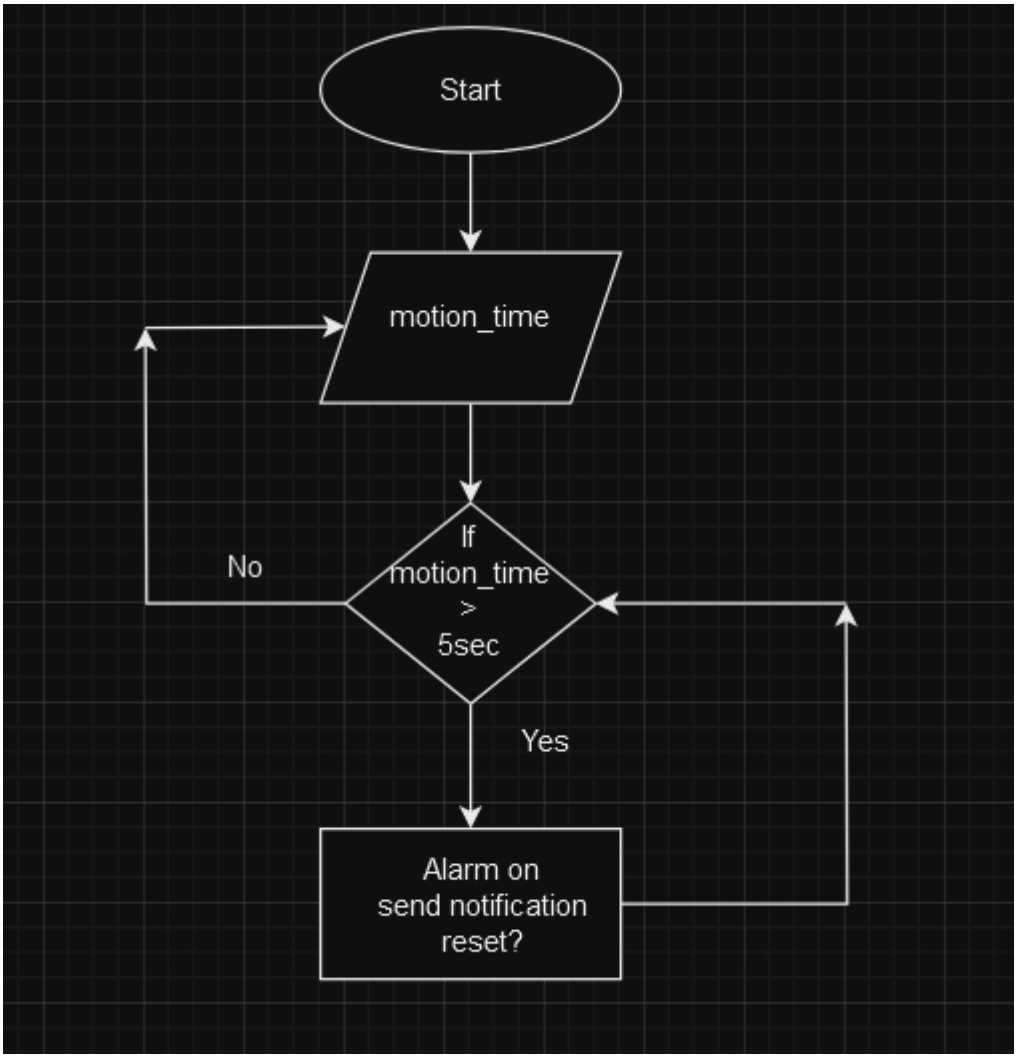
ANS::
1.Take soil moisture as soil_moist
2.Take user defined threshold as thresh
3.for every time==1hour
    3.1 if (soil_moist < thresh)
        3.1.1 for water_time=3minute
            3.1.1.1 pump=ON;
            3.1.1.2 log time to SD;
            3.1.1.3 LED=ON;
    3.2 else
        3.2.1 continue;




**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.

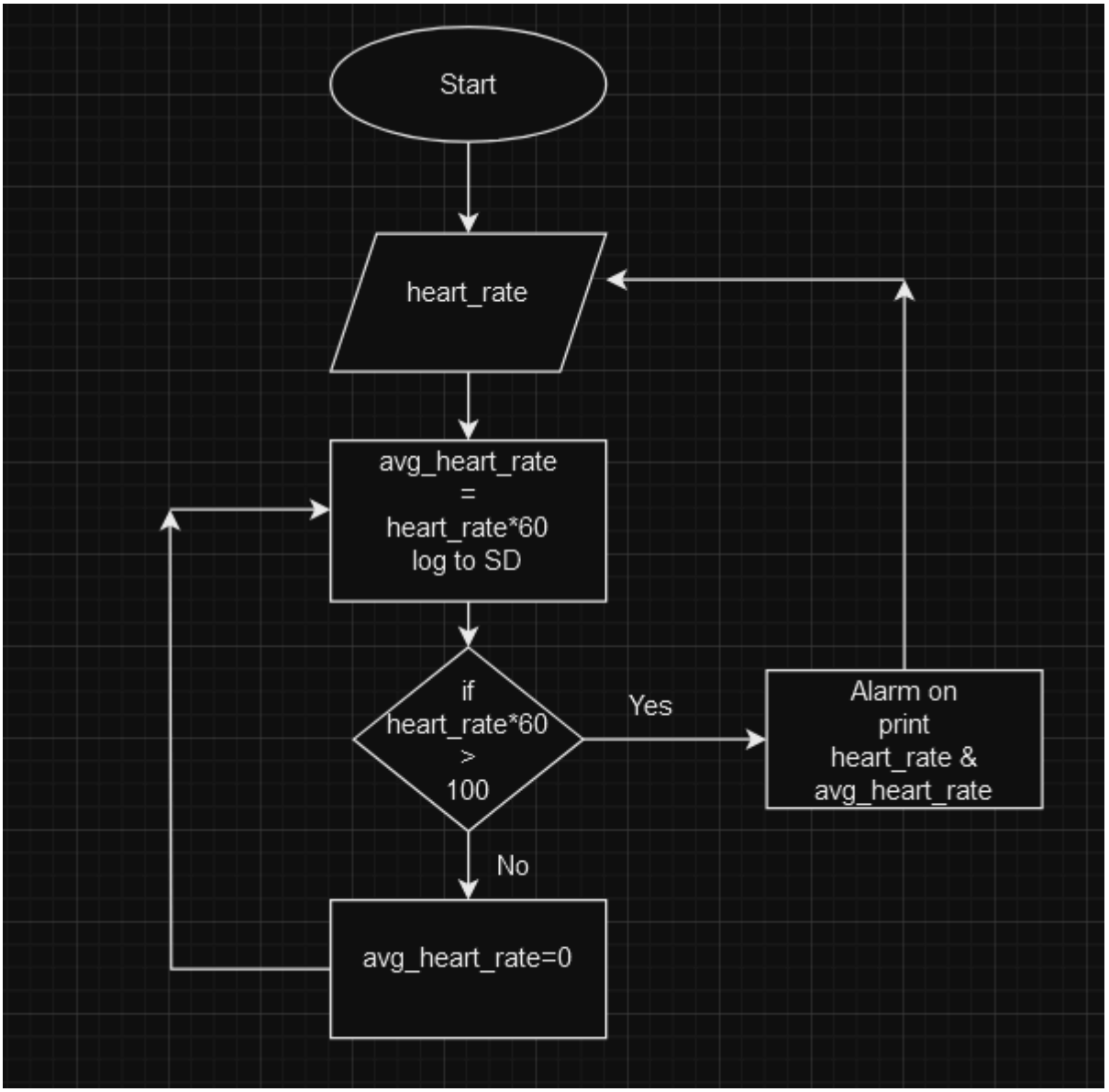
ANS::
1.Take PIR sensor value as motion_time
2.if (motion_time > 5 sec)
    2.1 Alarm=ON;
    2.2 send notification via UART;
    2.3 reset alarm yes or no:
3.else
    3.1 continue




**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.

ANS::
1.Take Heart rate from sensor as heart_rate
2.for every time=1second
    2.1 avg_heart_rate=heart_rate*60;
    2.2 log heart_rate & avg_heart_rate to SD:
    2.3 if (heart_rate*60 > 100)
        2.3.1 Alarm=ON;
        2.3.2 print "heart_rate & avg_heart_rate";
    2.4 else
        2.4.1 break;
    2.5 avg_heart_rate=0;



**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).

ANS::

1.Take intensity of light from sensor as light_intensity

2.Take user defined threshold as threshold

3.for every time=1minute

3.1 if (SWITCH == OFF)

3.1.1 if (light_intensity < threshold)

3.1.1.1 main_LED=ON;

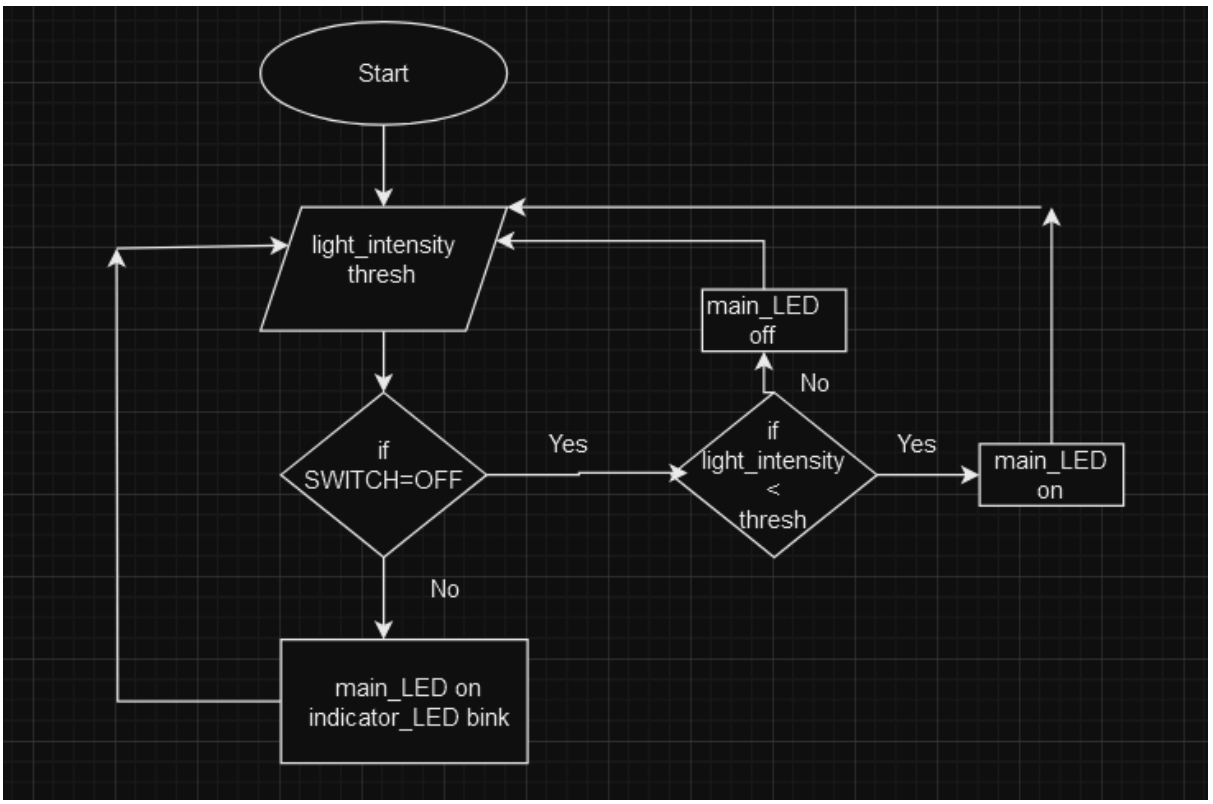
3.1.2 else

3.1.2.1 main_LED=OFF;

3.2 else

3.2.1 main_LED=ON;

3.2.2 indicator_LED=BLINK;



**6.** Digital Stopwatch

Problem Statement:

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

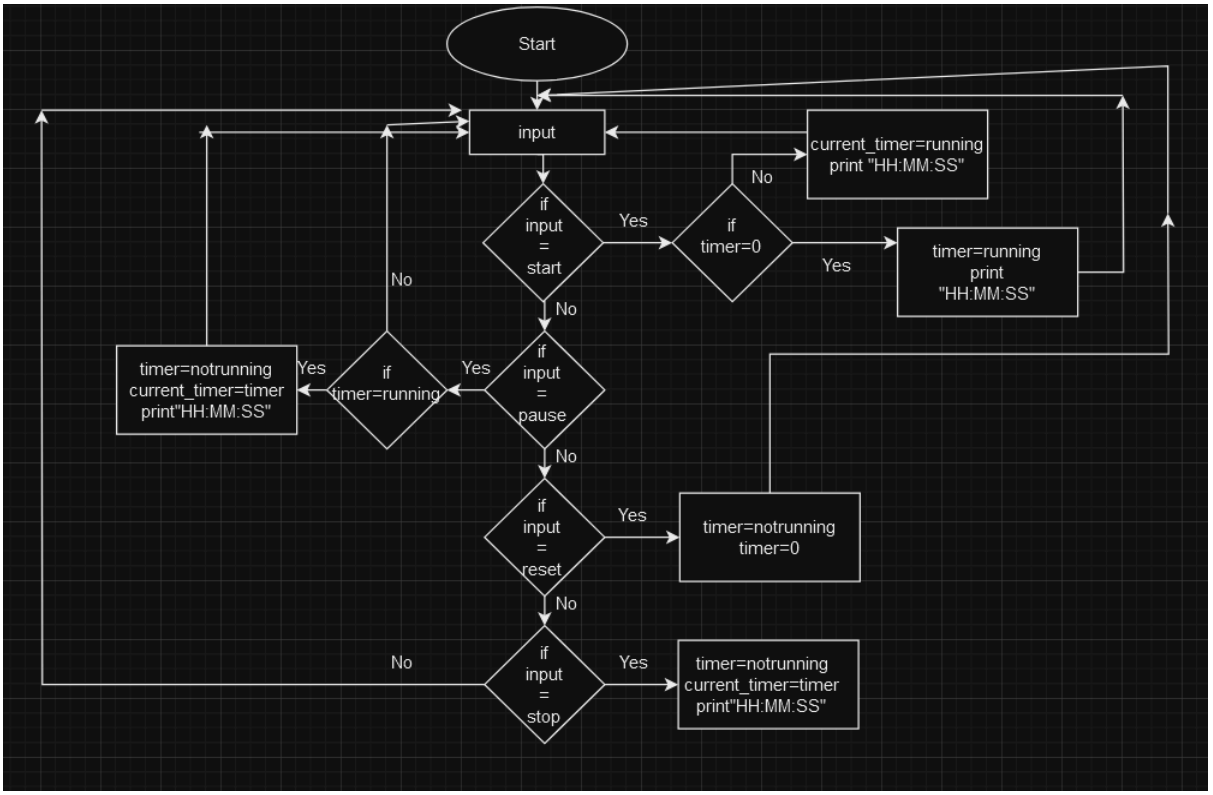
Requirements:

• 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.

ANS::

1.Take input_button
2.Switch(input_button)
    2.1 Case "Start"
        2.1.1 if ( timer=0)
            2.1.1.1 timer=running;
            2.1.1.2 print "HH:MM:SS";
        2.1.2 else
            2.1.2.1 current_timer=running
            2.1.2.2 print "HH:MM:SS";

    2.2 Case "Pause"
        2.2.1 if (timer =running)
            2.2.1.1 timer=notrunning
            2.2.1.2 current_timer=timer;
            2.2.1.3 print "HH:MM:SS";
    2.3 Case "Reset"
        2.3.1 timer=notrunning;
        2.3.2 timer=0;
    2.4 Case "Stop"
        2.4.1 timer=notrunning;
        2.4.2 current_timer=timer;
        2.4.3 print "HH:MM:SS";




**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.

ANS::

1.Take sensor temp as sense_temp

2.for every time=10min

    2.1 if (Sensor !=FAIL)

        2.1.1 for i in range (0,timer)

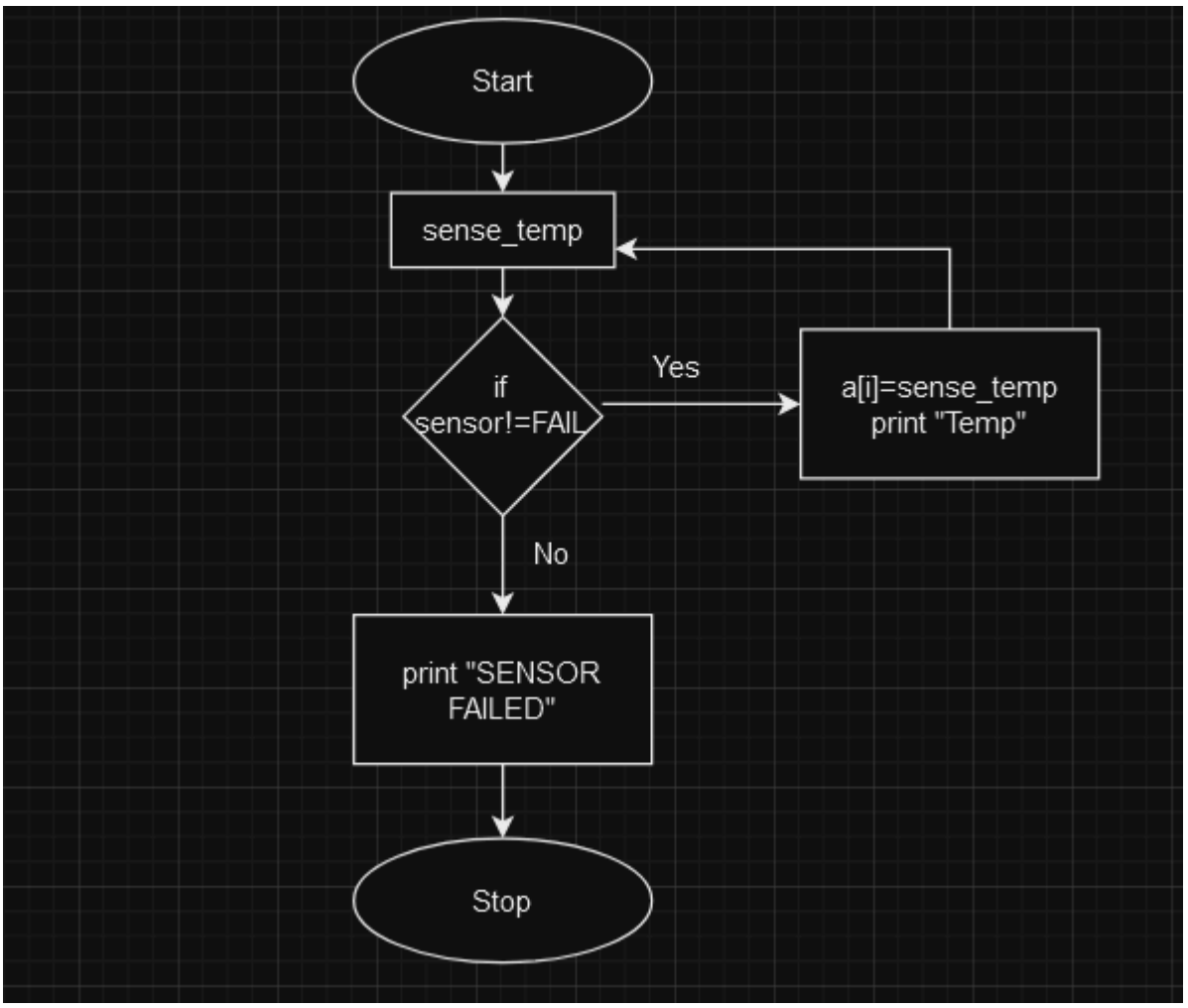
        2.1.2 a[i]=sense_temp;

    2.2 else

        2.2.1 print "SENSOR FAILED"

3.Retriving function take value of i

    3.1 print "a[i]";




**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).

ANS::

1.Connect_with_bluetooth
2.Take user input as input
3.Switch(input)
    3.1 Case "Forward"
        3.1.1 move=forward;
    3.2 Case "Backward"
        3.2.1 move=backward;
    3.3 Case "Left"
        3.3.1 move=Left;
    3.4 Case "Right"

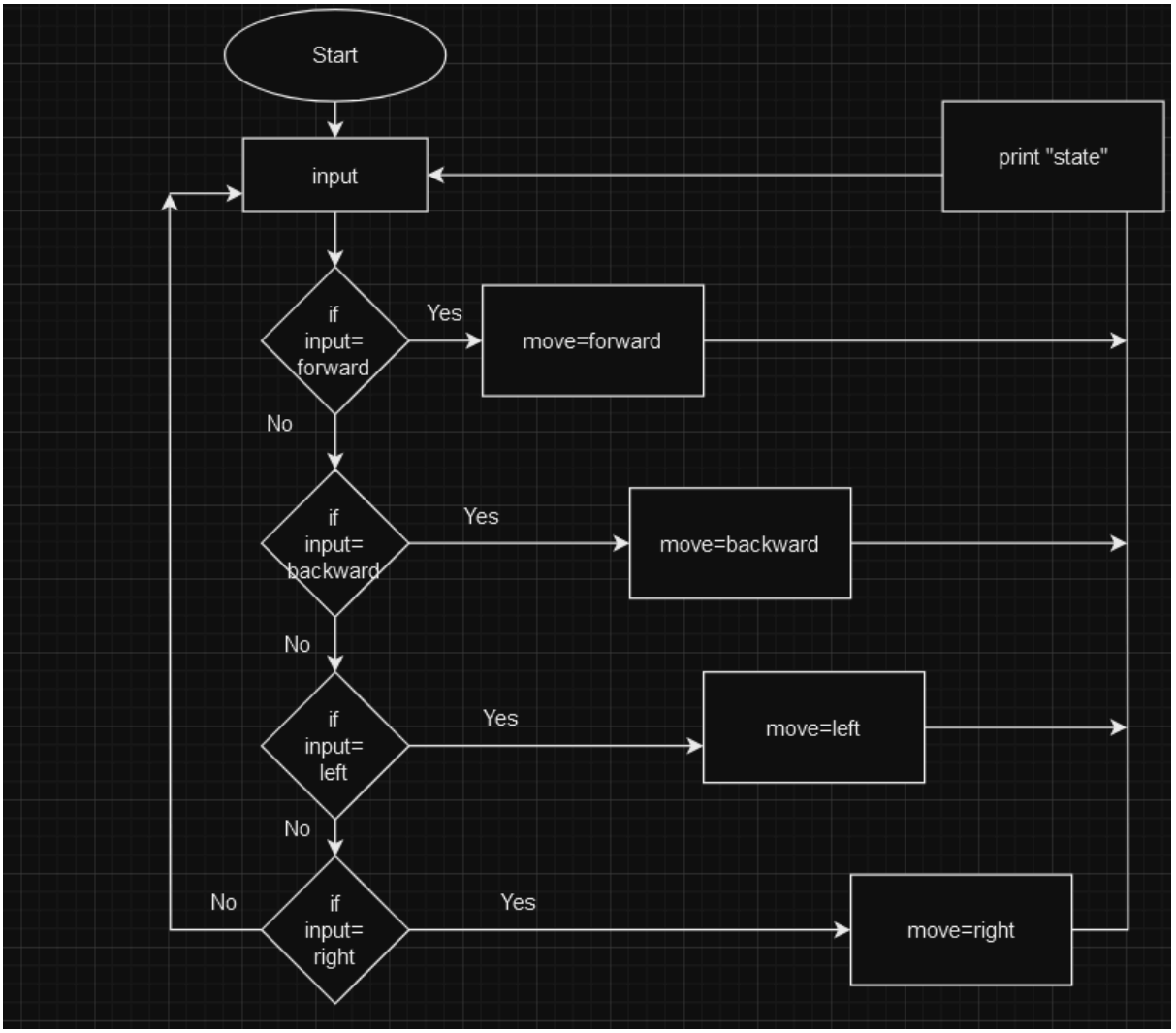
3.4.1 move=Right;

3.5 Case "++"

3.5.1 move=speed++;

3.6 Case "--"

3.6.1 move=speed--;




**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.

ANS::

1.Take Battery reading from sensor as read_battery

2.for every time=1min

2.1 if (read_battery < 11V)

2.1.1 print "read_battery";

2.1.2 Alarm=ON;

2.1.3 log Data to memory;

2.1.4 if (battery_saver!=ON)

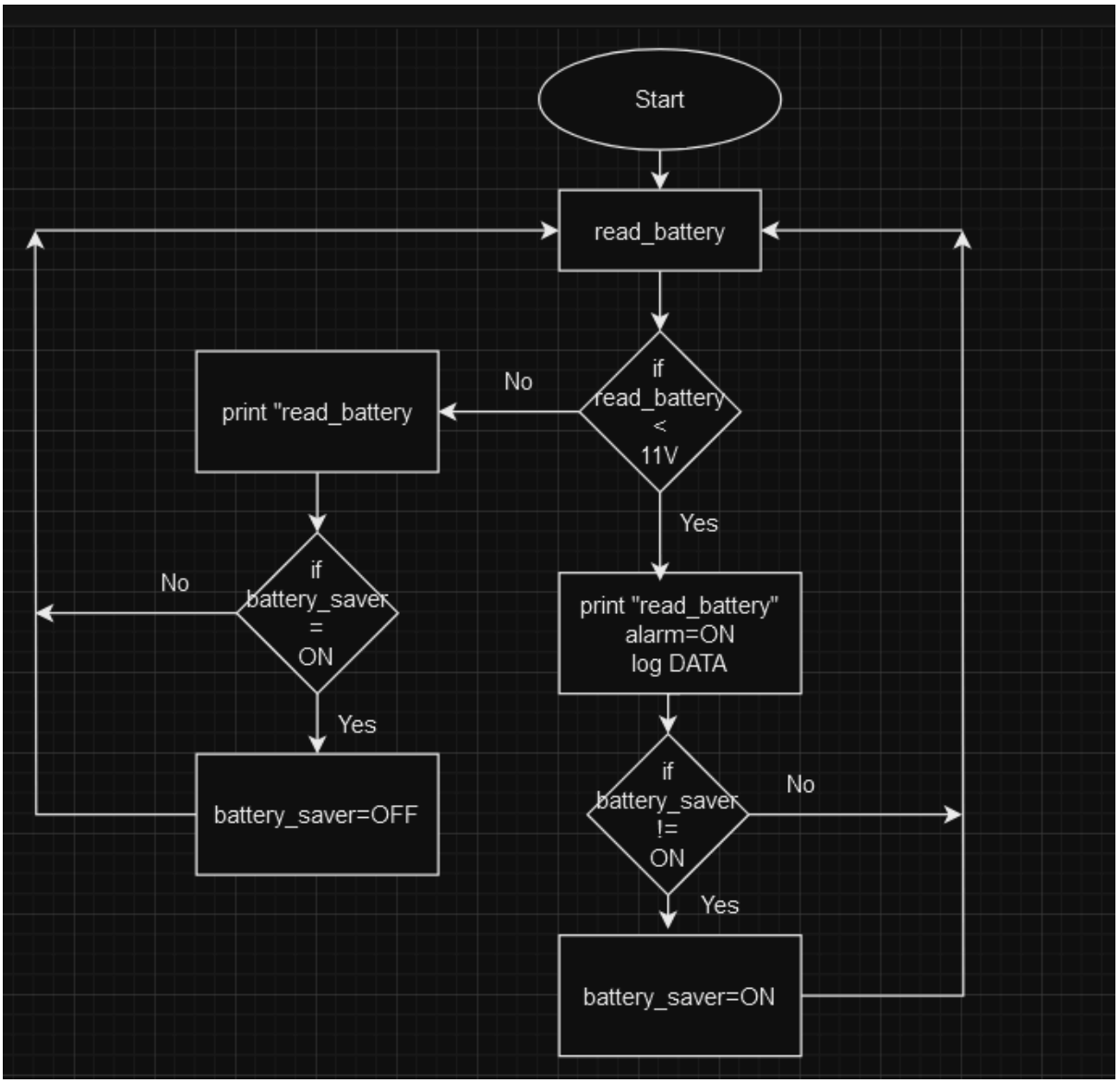
2.1.4.1 battery_saver=ON;

2.2 else

2.2.1 print "read_battery";

2.2.2 if (battery_saver=ON)

2.2.2.1 battery_saver=OFF;




**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.

ANS::

1.Take RFID input as scan_rfid
2.if (scan_rfid == stored_rfid)
    2.1 Activate relay;
    2.2 log "scan_rfid Successful(time)" to SD;
3.else
    3.1 Buzzer=ON;
    3.2 log "scan_rfid Unsuccessful(time)" to SD;