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.

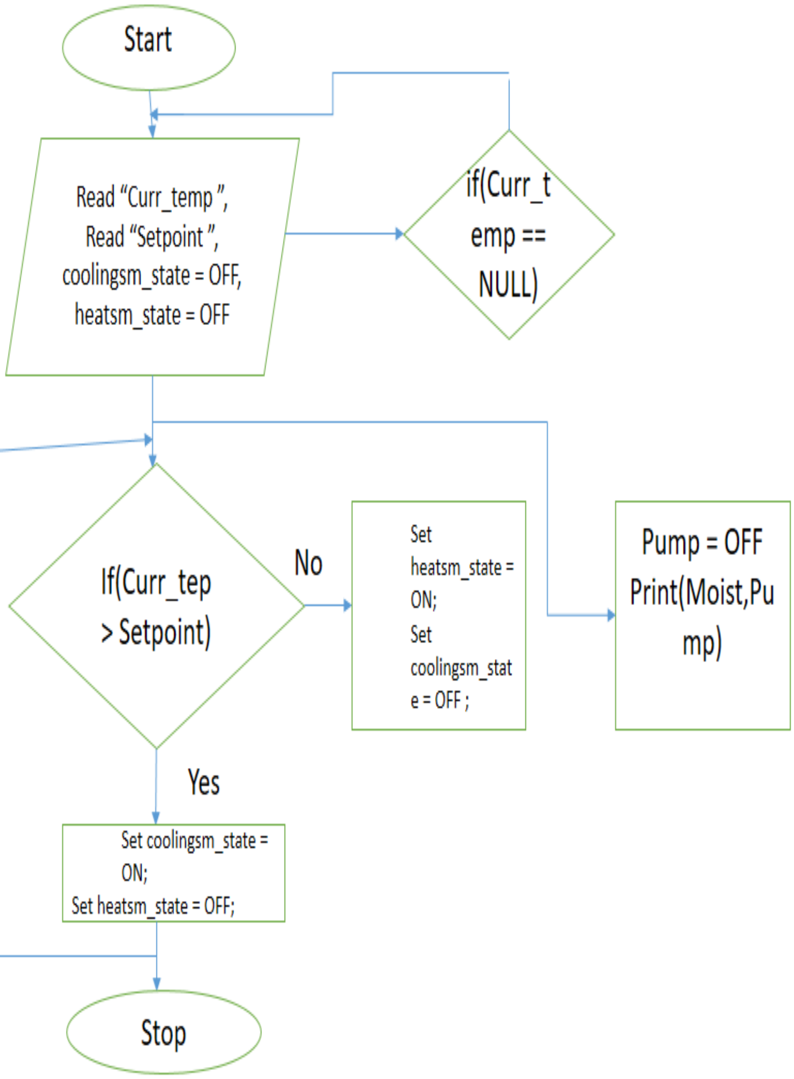
Requirements:

- 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. Curr_temp <- Read from sensor
2. Setpoint <- Read from user
3. Define coolingsm_state = OFF ;
4. Define heatsm_state = OFF;
5. If Curr_temp == NULL , do
       a. Step 1
6. if Curr_temp > Setpoint , do
       a. Set coolingsm_state = ON;
       b. Set heatsm_state = OFF;
7. else  , do
       a. Set heatsm_state = ON;
       b. Set coolingsm_state = OFF ;
8. Curr_temp , Setpoint -> Print on LCD
```

FLOW CHART :



2. Automated Plant Watering System

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

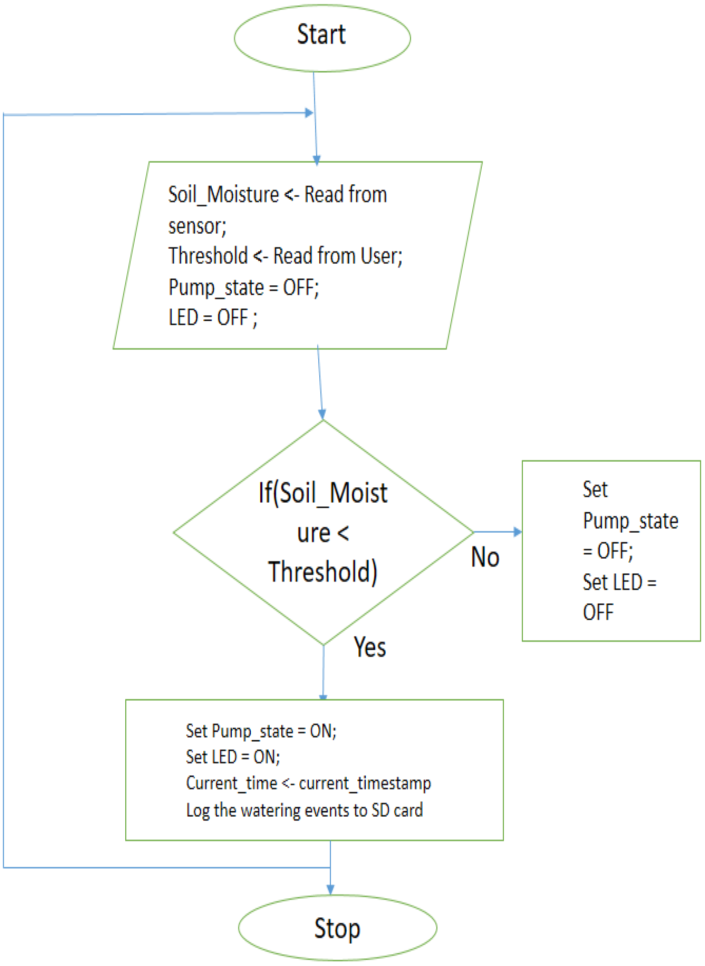
Requirements:

- 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. Soil_Moisture <- Read from sensor
2. Threshold <- Read from User
3. Pump_state = OFF;
4. LED = OFF ;
5. if Soil_Moisture < Threshold , do
    a. Set Pump_state = ON;
    b. Set LED = ON;
    c. Current_time <- current_timestamp
    d. Log the watering events to SD card
    e. After 1Hr ,Repeat Step 1
6. else , do
    a. Set Pump_state = OFF;
    b. Set LED = OFF
```

FLOW CHART :


Start
Soil_Moisture <- Read from sensor;
Threshold <- Read from User;
Pump_state = OFF;
LED = OFF ;
If(Soil_Moisture < Threshold)
No
Set Pump_state = OFF;
Set LED = OFF
Yes
Set Pump_state = ON;
Set LED = ON;
Current_time <- current_timestamp
Log the watering events to SD card
Stop

3. Motion Detection Alarm System

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

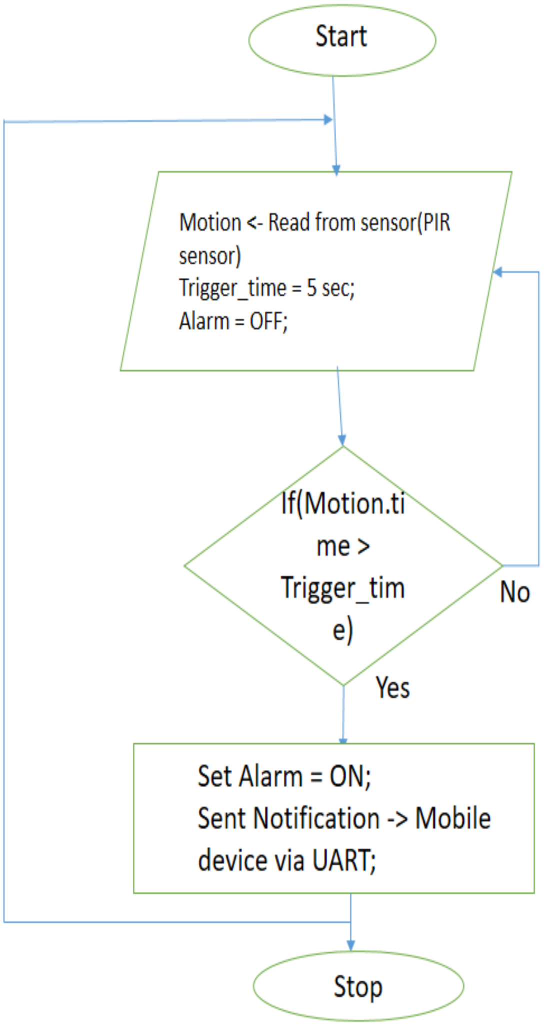
Requirements:

- 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. Motion <- Read from sensor(PIR sensor)
2. Trigger_time = 5 sec;
3. Alarm = OFF;
4. if Motion.time > Trigger_time , do
    a. Set Alarm = ON;
    b. Sent Notification -> Mobile device via UART;
    c. Set Alarm = OFF , Back to Step 1
```

FLOW CHART :


Start
Motion <- Read from sensor(PIR sensor)
Trigger_time = 5 sec;
Alarm = OFF;
If(Motion.time > Trigger_time)
No
Yes
Set Alarm = ON;
Sent Notification -> Mobile device via UART;
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.

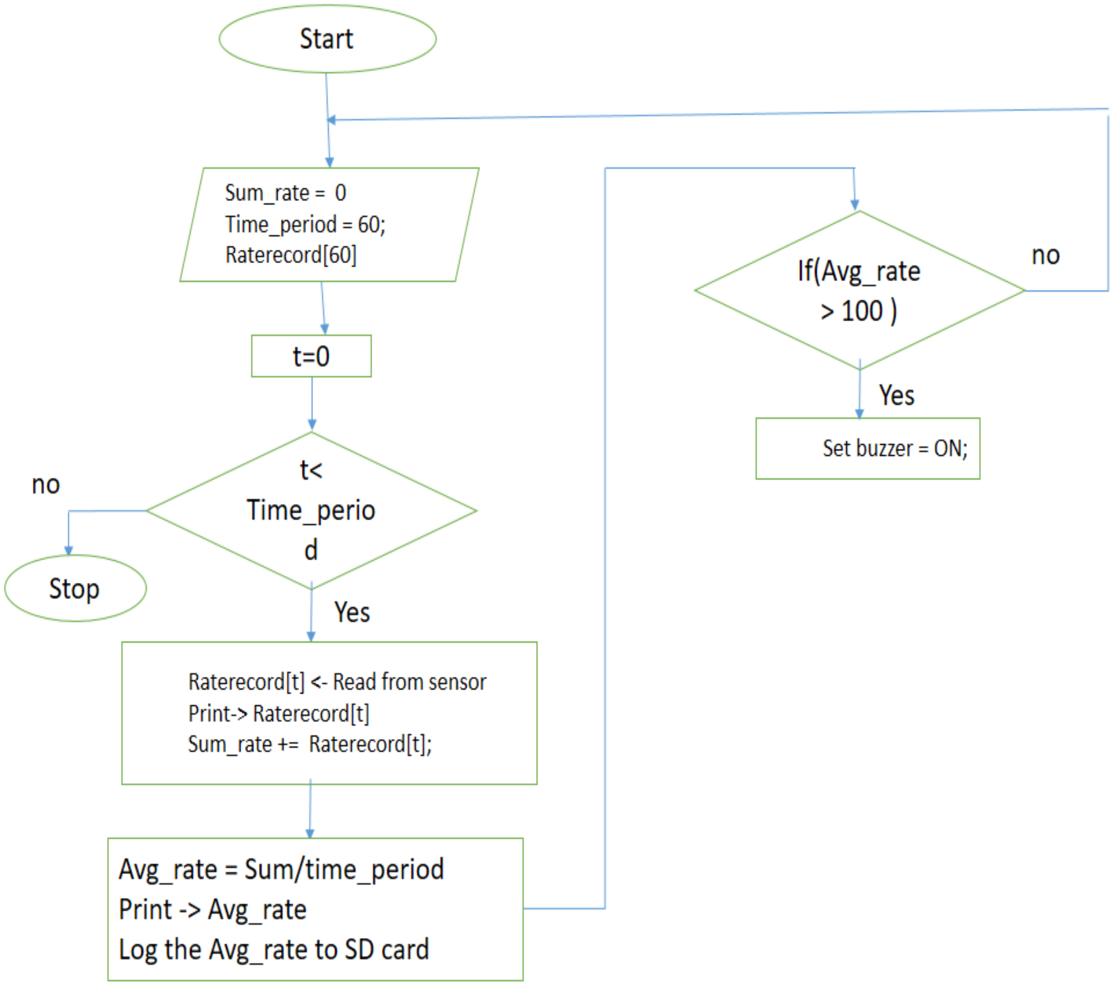
• Log heart rate data to an SD card for later analysis.

PSEUDOCODE :

```
1.  Sum_rate =  0
2.  Time_period = 60;
3.  Raterecord[60]
4.  for each t in 0 to (time_period – 1), do
        a.  Raterecord[t] <- Read from sensor
        b.  Print-> Raterecord[t]

5.  For each t in range 0 to (time_period – 1),do
        a.  Sum_rate +=  Raterecord[t];
6.  Avg_rate = Sum/time_period
7.  Print -> Avg_rate
8.  Log the Avg_rate to SD card
9.  If Avg_rate > 100 , do
        a.  Set buzzer = ON;
10. Else , do
        a.  Step 1
```

FLOW CHART :



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.

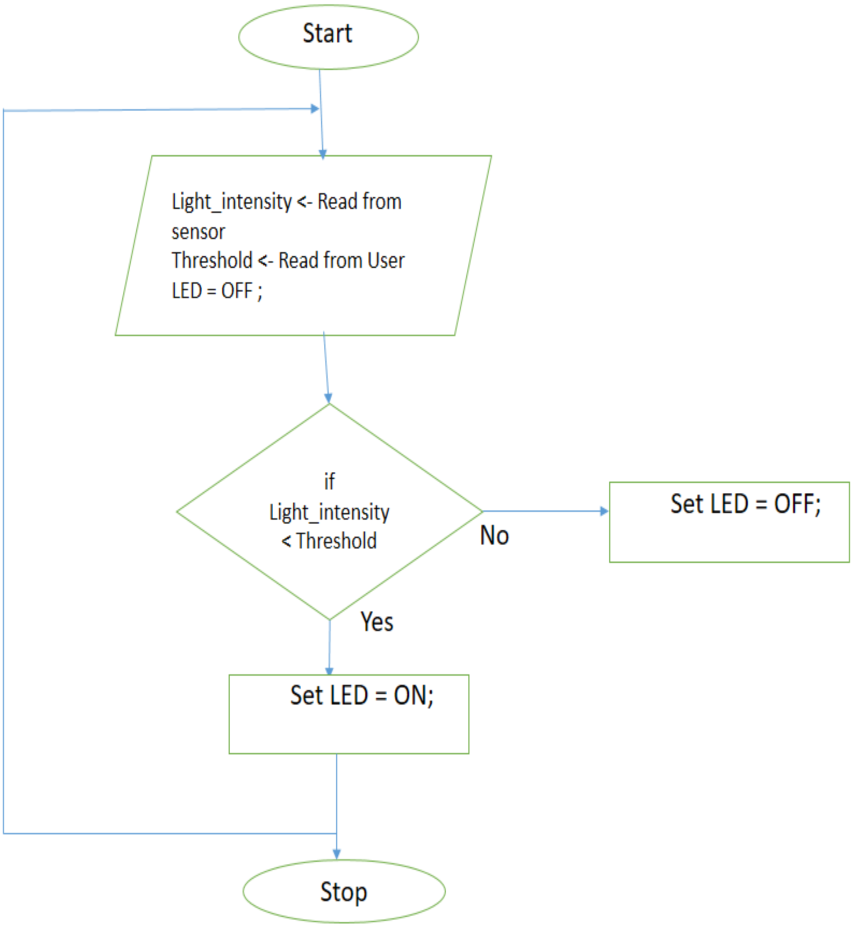
• Provide status feedback through another LED (e.g., blinking when in manual mode).

PSEUDOCODE :

```
1. Light_intensity <- Read from sensor
2. Threshold <- Read from User
3. LED = OFF ;
```

```
4.  if Light_intensity < Threshold , do
        a.  Set LED = ON;
5.  else  , do
        a.  Set LED = OFF
```

FLOW CHART :


Start
Light_intensity <- Read from sensor
Threshold <- Read from User
LED = OFF ;
if Light_intensity < Threshold
No
Set LED = OFF;
Yes
Set LED = ON;
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 :

```
Start_button=OFF
Stop_button=OFF
Reset_button=OFF
Print->time.timestap()
Pause= OFF
Resume = OFF
if start==ON , do
    a. Log time to SD card
```

7. Temperature Logging System

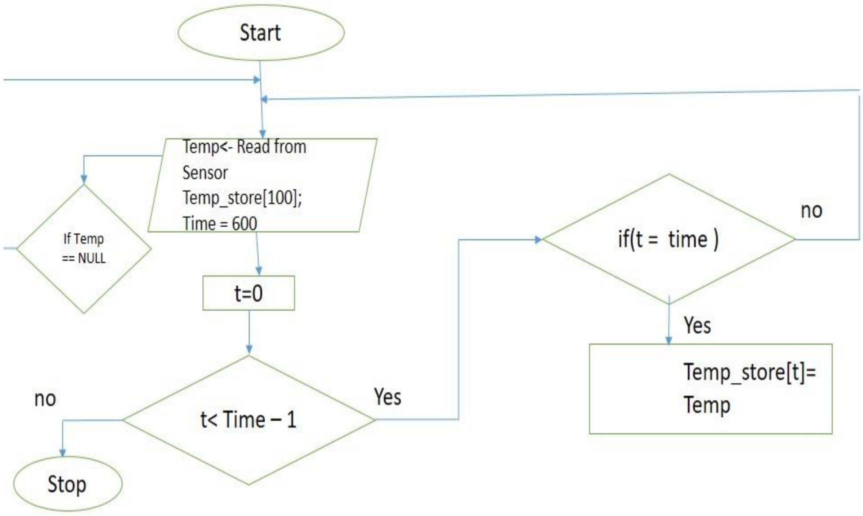
Problem Statement: Implement a temperature logging system that records temperature data at regular intervals.

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. Temp<- Read from Sensor
2. Temp_store[100];
3. Time = 600
4. For each t  in 0 to Time – 1 :
    a. If t =  time , do
        i. Temp_store[t]=Temp
5. If Temp == NULL , do
    a. Step 1
6. Print->time.timestap()
```



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).

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.

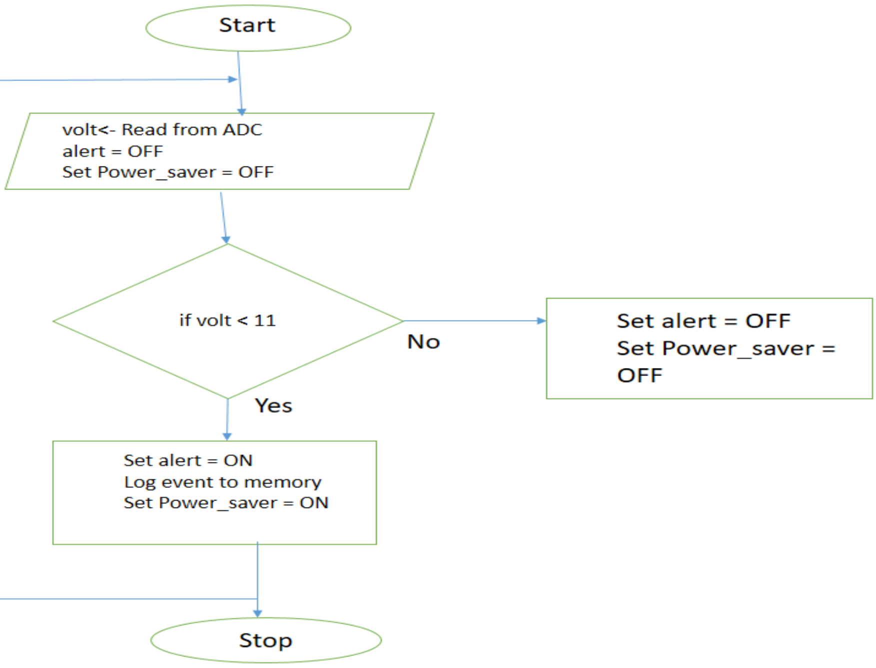
Requirements:

- 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 :

```
volt<- Read from ADC
alert = OFF
Set Power_saver = OFF
```

```
4. if volt < 11, do
      a. Set alert = ON
      b. Log event to memory
      c. Set Power_saver = ON
5. Else , do
      a. Set alert = OFF
      b. Set Power_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.

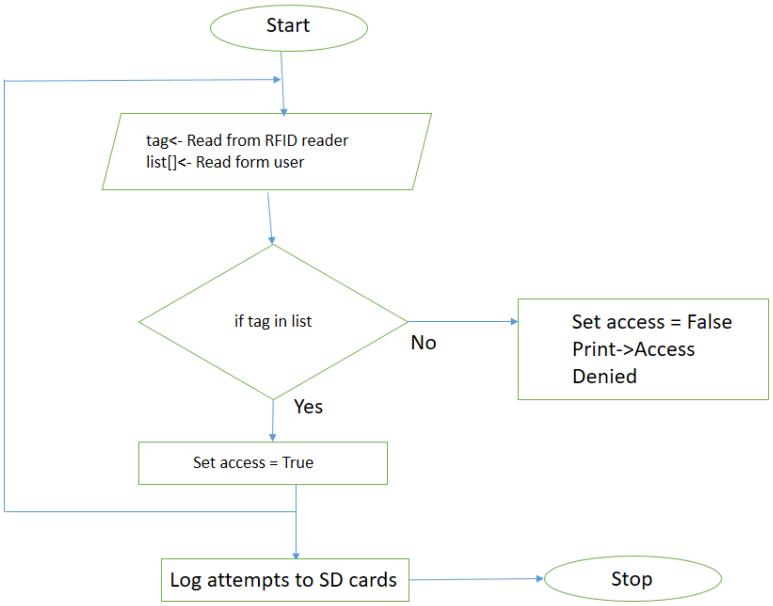
Requirements:

- 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

PSEUDOCODE :

```
1.  tag<- Read from RFID reader
2.  list[]<- Read form user
3.  if tag in list, do
        a.  Set access = True
4.  Else , do
        a.  Set access = False
        b.  Print->Access Denied
5.  Log attempts to SD cards
```


Start
tag<- Read from RFID reader
list[]<- Read form user
if tag in list
No
Set access = False
Print->Access
Denied
Yes
Set access = True
Log attempts to SD cards
Stop