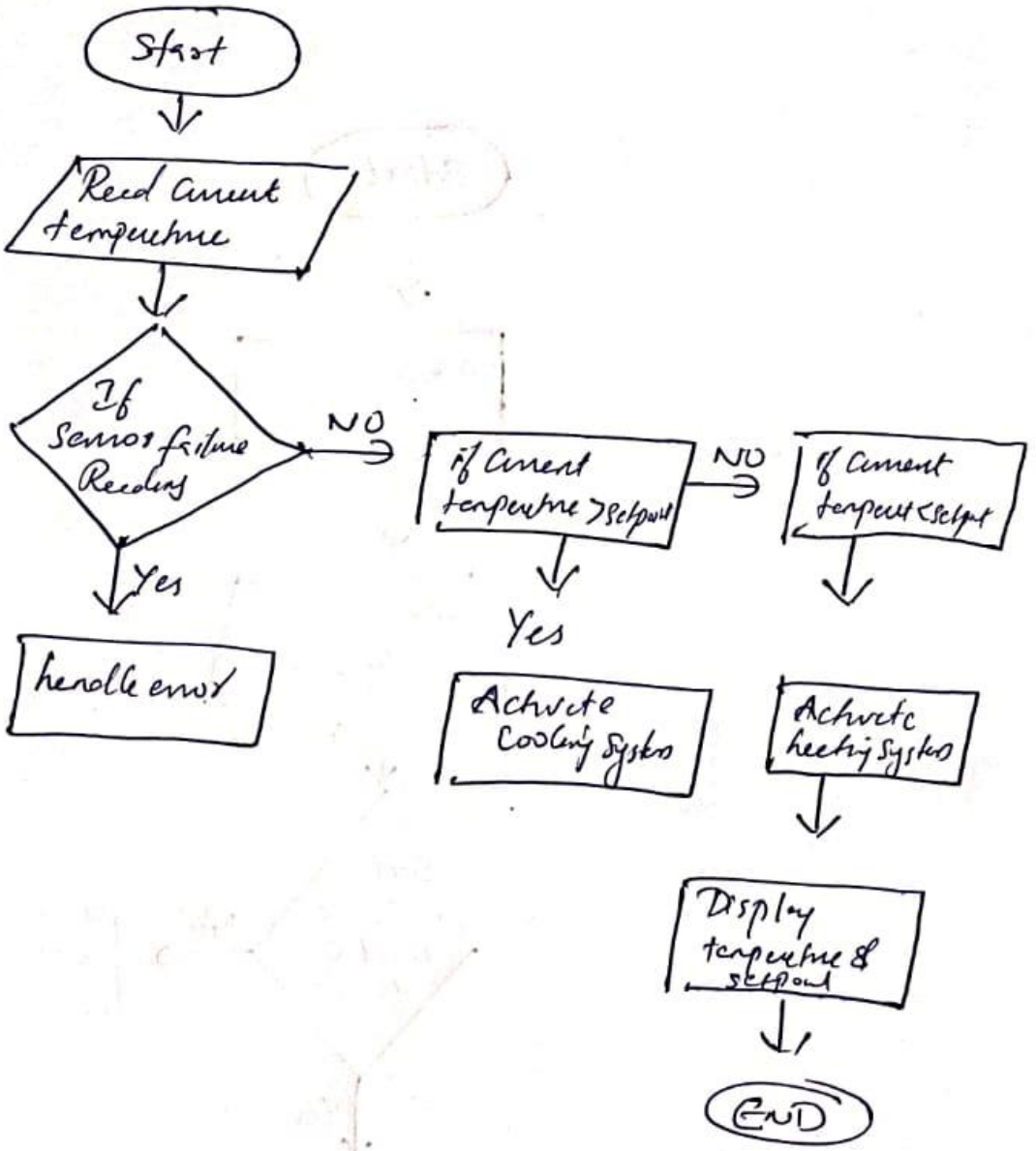
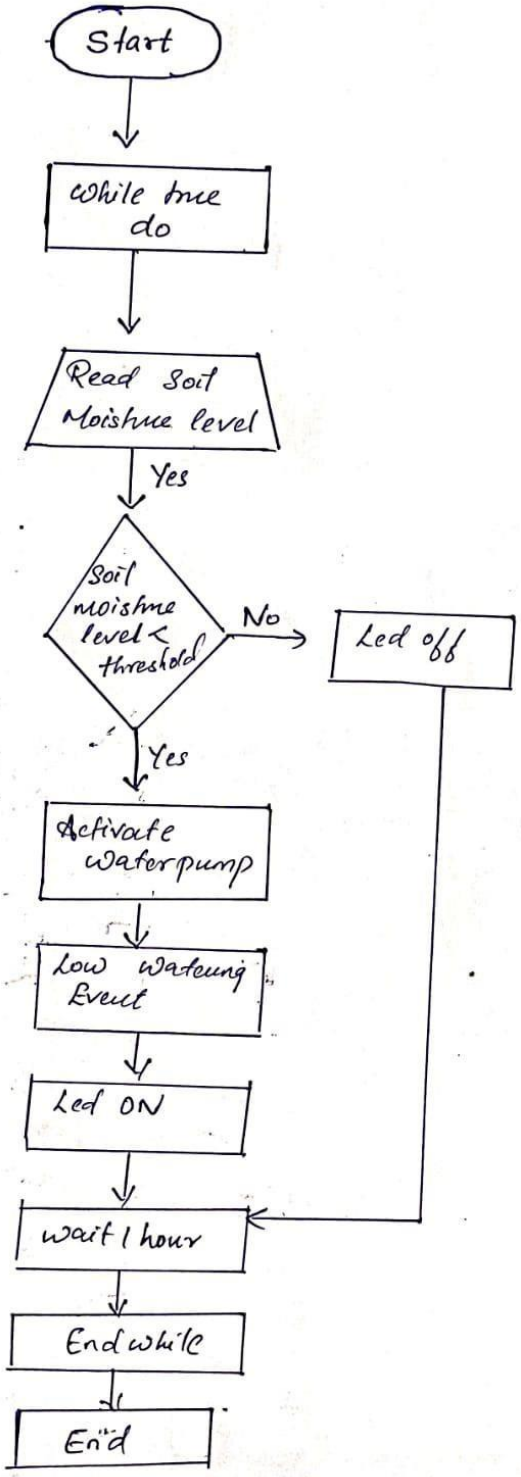
1. **Smart Home Temperature Control Problem Statement: Design a temperature control system for a smart home. The system should read the current temperature from a sensor every minute and compare it to a user-defined setpoint. Requirements: • If the current temperature is above the setpoint, activate 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.**

```
1. Start
2. Read the current temperature from the sensor.
3. if sensorReadFailure Then
     handleError()
4. else
     if currentTemperature > setpoint Then
       activateCoolingSystem()
5.   else if currentTemperature < setpoint Then
       activateHeatingSystem()
6.   else
       deactivateSystems()
7.   endif
     display(currentTemperature, setpoint)
8. end
```


Start
Read current temperature
If Sensor failure Reading
NO
Yes
handle error
if current temperature > setpoint
NO
Yes
Activate Cooling System
if current temperature < setpoint
Activate heating System
Display temperature & setpoint
END

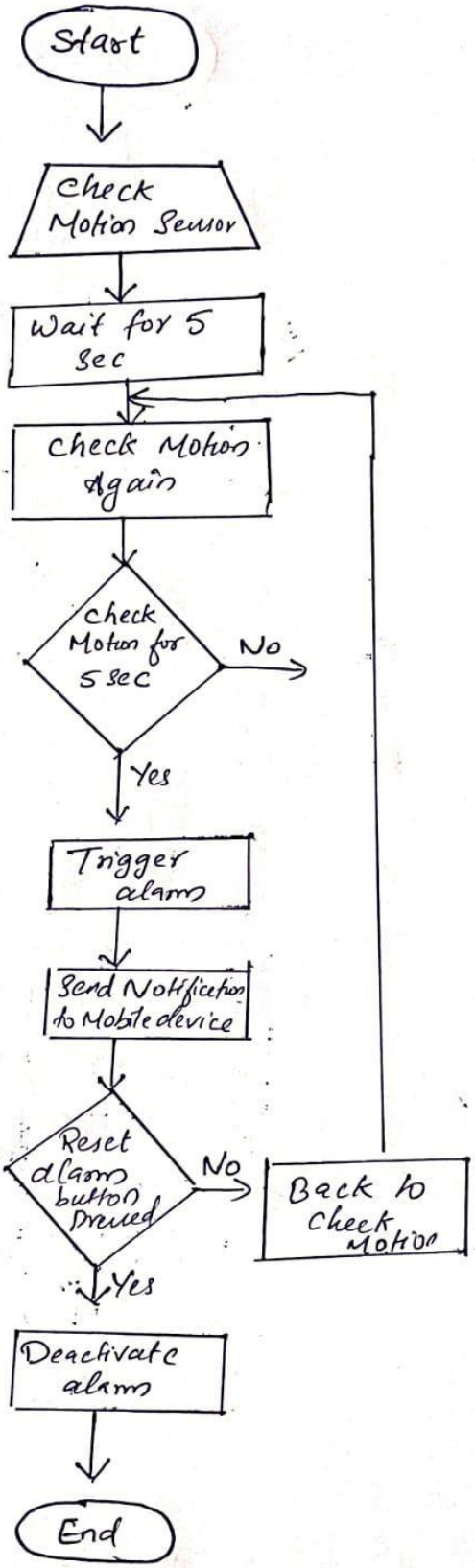
2. Automated Plant Watering System Problem Statement: Create an automated watering system for plants that checks soil moisture levels and waters the plants accordingly. Requirements: • Read soil moisture level from a sensor every hour. • If moisture level is below a defined threshold, activate the water 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).

```
START
 WHILE true DO
 soilMoistureLevel = readSoilMoistureSensor()
IF soilMoistureLevel < threshold THEN
activateWaterPump()
 logWateringEvent(timestamp)
 LED_ON()
ELSE
LED_OFF()
ENDIF
 WAIT 1 hour
ENDWHILE
END
```


Start
While true do
Read Soil Moisture level
Yes
Soil moisture level < threshold
No
Led off
Yes
Activate Waterpump
Low watering Event
Led ON
wait 1 hour
End while
End

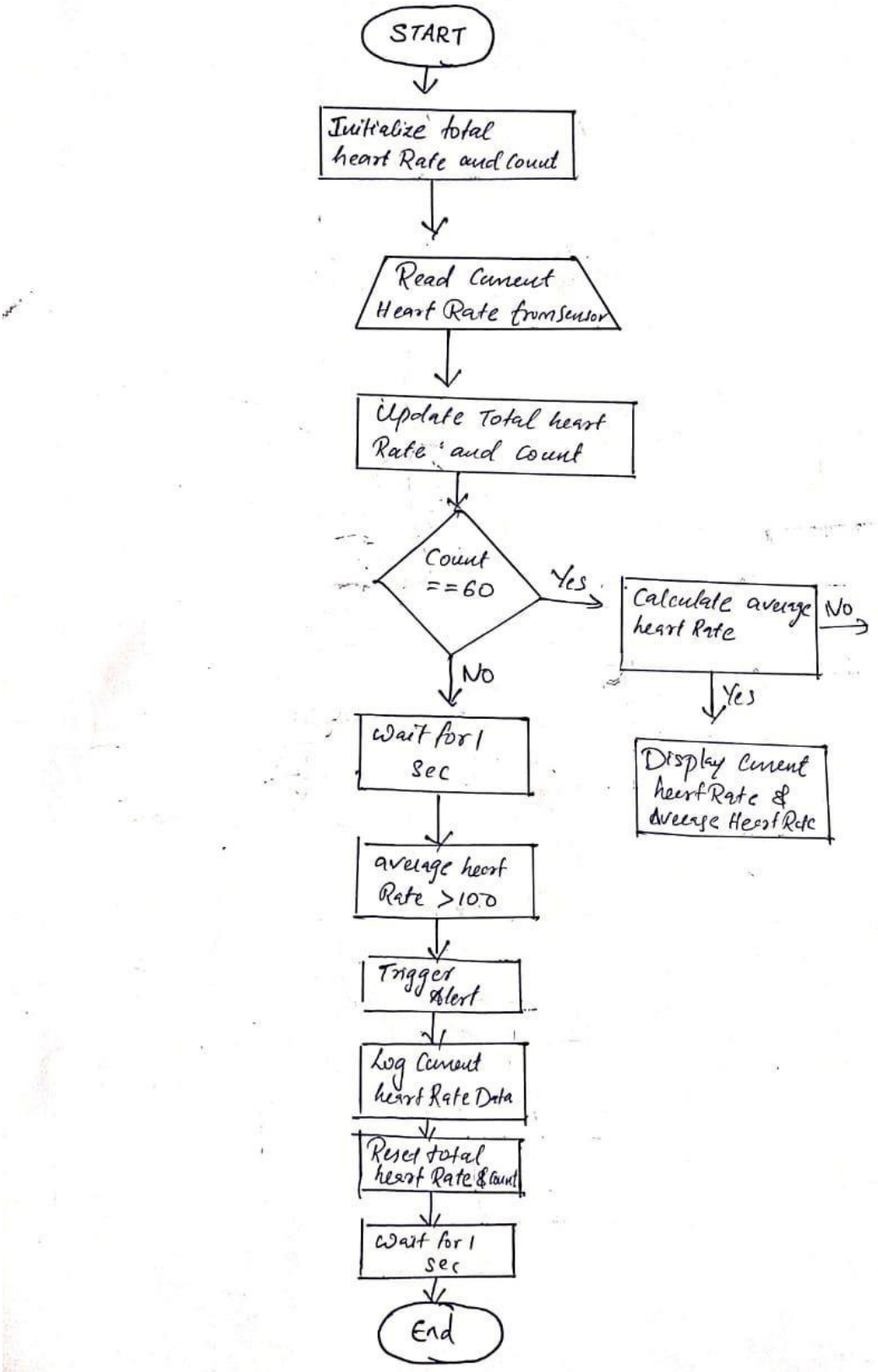
3. Motion Detection Alarm System Problem Statement: Develop a security alarm system that detects motion using a PIR sensor. Requirements: • Continuously monitor motion detection status. • If motion is detected for more than 5 seconds, trigger an alarm (buzzer). • Send a notification to a mobile device via UART communication. • Include a reset mechanism to deactivate the alarm.

```
START
WHILE true DO
 motionDetected = checkMotionSensor()
IF motionDetected THEN
WAIT 5 seconds
 IF motionDetected for 5 seconds THEN
 triggerAlarm()
 sendNotificationToMobileDevice()
ENDIF
ENDIF
IF resetAlarmButtonPressed THEN
deactivateAlarm()
ENDIF
 ENDWHILE
 END
```


Start
Check Motion Sensor
Wait for 5 sec
Check Motion Again
Check Motion for 5 sec
No
Yes
Trigger alarm
Send Notification to Mobile device
Reset alarm button Pressed
No
Back to Check Motion
Yes
Deactivate alarm
End

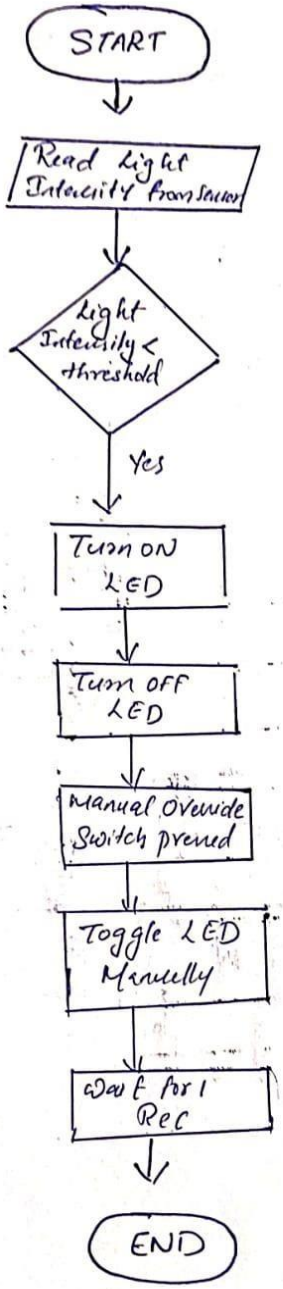
4. Heart Rate Monitor Problem Statement: Implement a heart rate monitoring application that reads data from a heart rate sensor. Requirements: • Sample heart rate data every second and calculate the average heart rate over one minute. • If the heart rate exceeds 100 beats per minute, trigger an alert (buzzer). • Display current heart rate and average heart rate on an LCD screen. • Log heart rate data to an SD card for later analysis.

```
START
totalHeartRate = 0
count = 0
WHILE true DO currentHeartRate = readHeartRateSensor()
 totalHeartRate = totalHeartRate + currentHeartRate
count = count + 1
 IF count == 60 THEN
averageHeartRate = totalHeartRate / 60
IF averageHeartRate > 100 THEN
 triggerAlert()
ENDIF
display(currentHeartRate, averageHeartRate)
logHeartRateData(currentHeartRate, timestamp)
totalHeartRate = 0
count = 0
ENDIF
 WAIT 1 second
ENDWHILE
 END
```


START
Initialize total heart Rate and count
Read Current Heart Rate from Sensor
Update Total heart Rate and count
Count == 60
Yes
Calculate average heart Rate
No
Yes
Display Current heart Rate & Average Heart Rate
No
Wait for 1 Sec
average heart Rate > 100
Trigger Alert
Log Current heart Rate Data
Reset total heart Rate & count
Wait for 1 Sec
End

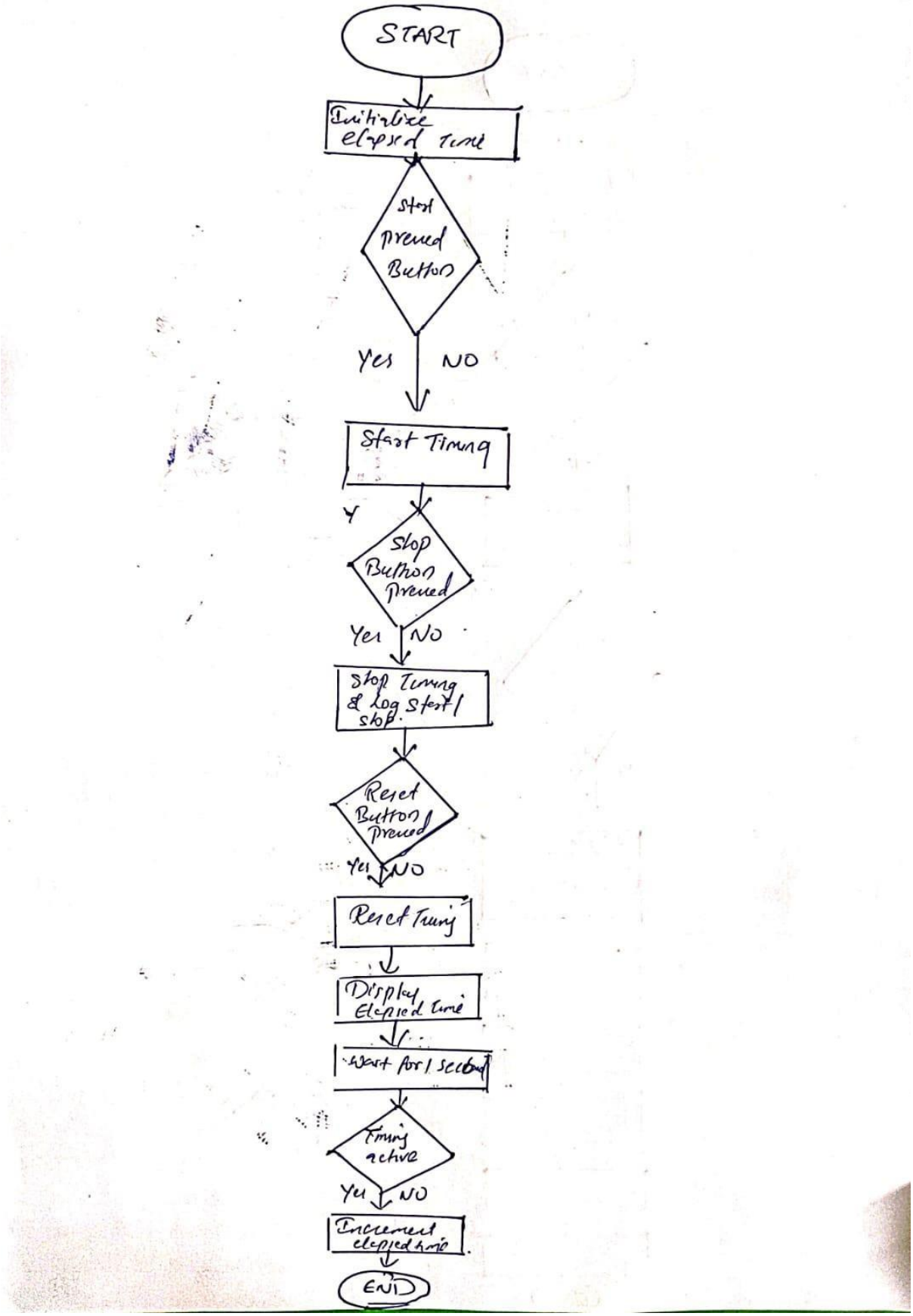
5. LED Control Based on Light Sensor Problem Statement: Create an embedded application that controls an LED based on ambient light levels detected by a light sensor. Requirements: • Read light intensity from the sensor every minute. • If light intensity is below a certain threshold, turn ON the LED; 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)

```
START
 WHILE true DO
 lightIntensity = readLightSensor()
IF lightIntensity < threshold THEN
turnON_LED()
ELSE
turnOFF_LED()
ENDIF
 IF manualOverrideSwitchPressed THEN
 toggleLEDManual()
 ENDIF
WAIT 1 minute
ENDWHILE
END
```


START
Read Light Intensity from Sensor
Light Intensity < threshold
Yes
Turn ON LED
Turn OFF LED
Manual Override Switch pressed
Toggle LED Manually
Wait for 1 Sec
END

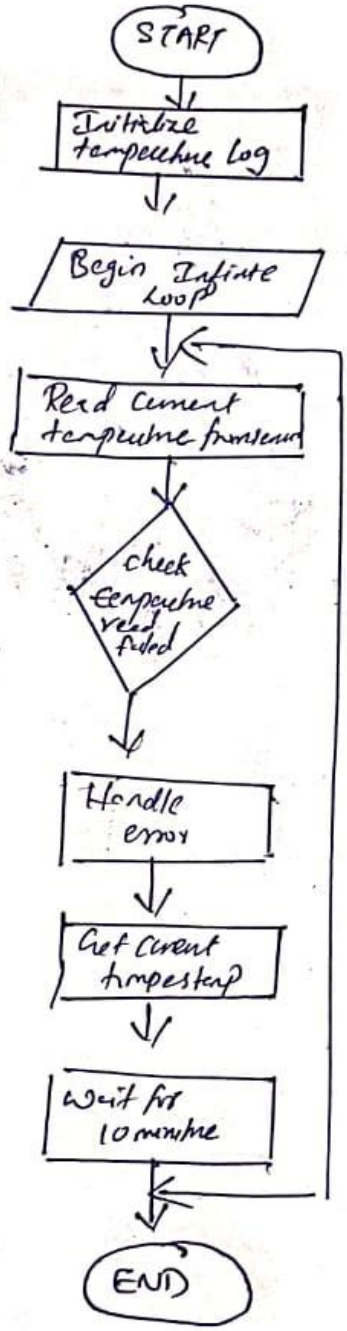
6. Digital Stopwatch Problem Statement: Design a digital stopwatch application that can start, stop, and reset using button inputs. Requirements: • Use buttons for Start, Stop, and Reset functionalities. • Display elapsed time on an LCD screen in hours, minutes, and seconds format. • Include functionality to pause and resume timing without resetting. • Log start and stop times to an SD card when stopped

```
START
elapsedTime = 0
WHILE true DO
 IF startButtonPressed THEN
 startTiming()
 ENDIF
 IF stopButtonPressed THEN
stopTiming() logStartStopTimes(startTime, stopTime)
 ENDIF
IF resetButtonPressed THEN
 resetTiming()
ENDIF
displayElapsedTime(elapsedTime)
WAIT 1 second
 IF timingActive THEN
elapsedTime = elapsedTime + 1
 ENDIF
 ENDWHILE
 END
```


START
Initialize elapsed time
Start Pressed Button
Yes
NO
Start Timing
Y
Stop Button Pressed
Yes
NO
Stop Timing & Log Start/ stop.
Reset Button Pressed
Yes
NO
Reset Timing
Display Elapsed time
wait for 1 second
Timing active
Yes
NO
Increment elapsed time
END

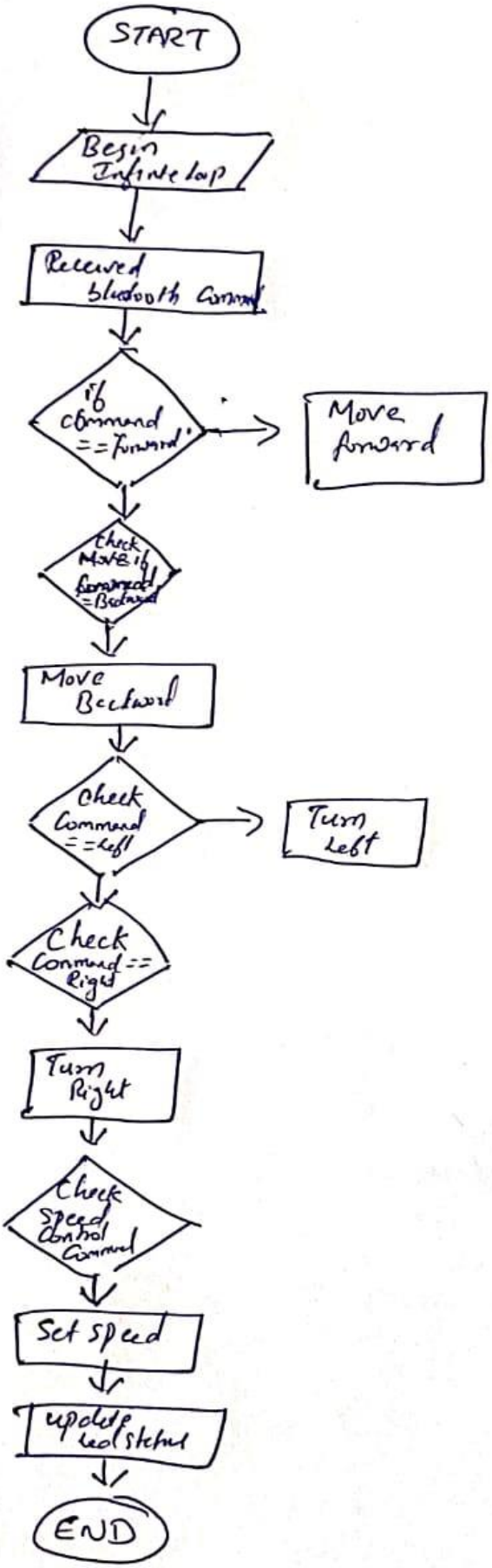
7. Temperature Logging System Problem Statement: Implement a temperature logging system that records temperature data at regular intervals. Requirements: • Read temperature from a sensor every 10 minutes. • Store each reading along with its timestamp in an array or log file. • Provide functionality to retrieve and display historical data upon request. • Include error handling for sensor read failures.

START
temperatureLog = []
WHILE true DO
currentTemperature = readTemperatureFromSensor()
IF currentTemperatureReadFailure THEN
handleError()
ELSE
timestamp = getCurrentTimestamp() logTemperature(currentTemperature, timestamp)
ENDIF
WAIT 10 minutes
ENDWHILE
END


START
Initialize temperature log
Begin Infinite Loop
Read current temperature from sensor
check temperature read failed
Handle error
Get current timestamp
Wait for 10 minute
END

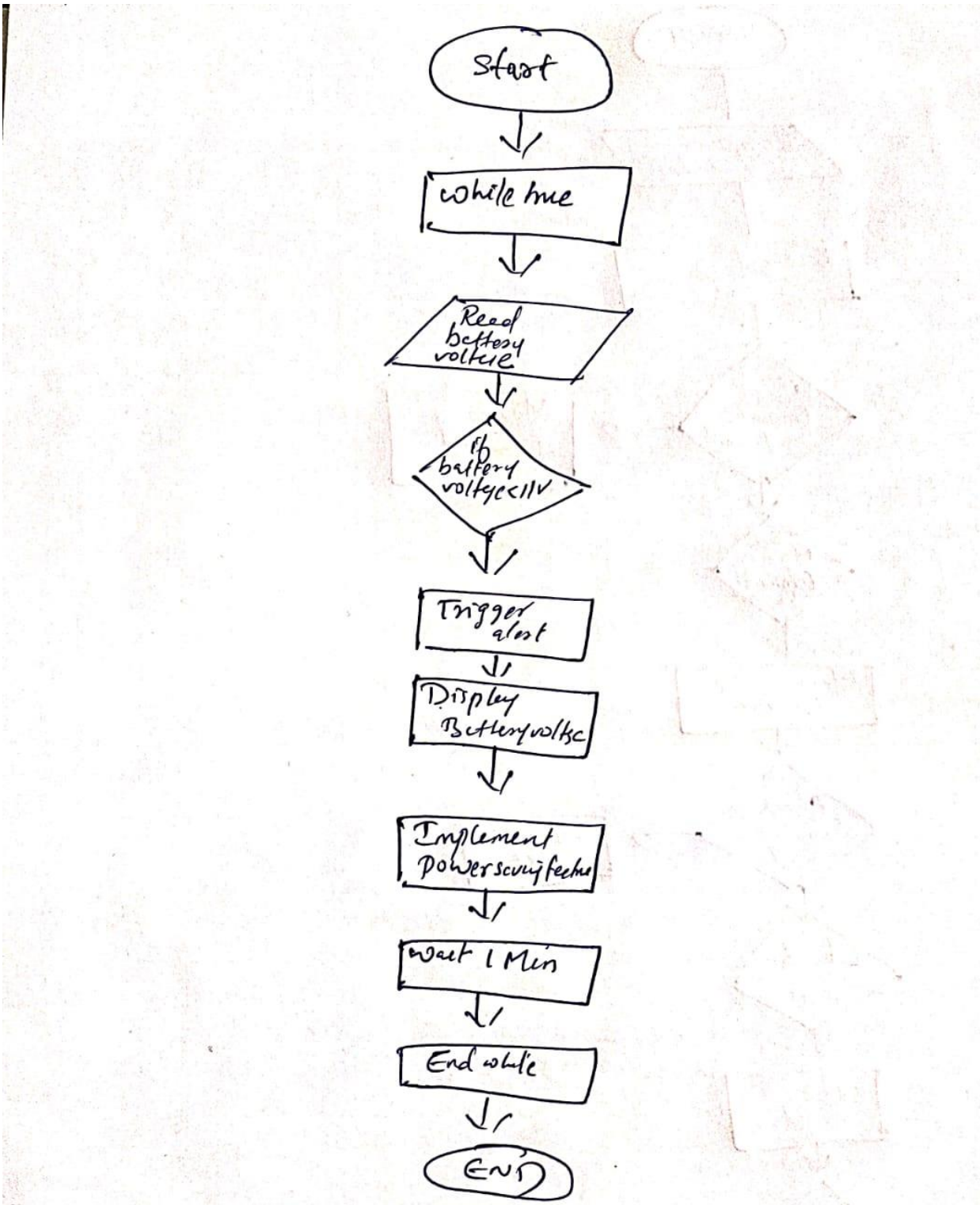
8. Bluetooth Controlled Robot Problem Statement: Create an embedded application for controlling a robot via Bluetooth commands. Requirements: • Establish Bluetooth communication with a mobile device. • Implement commands for moving forward, backward, left, and right. • Include speed control functionality based on received commands. • Provide feedback through LEDs indicating the current state (e.g., moving or stopped)

```
START
WHILE true DO
 command = receiveBluetoothCommand()
IF command == "FORWARD" THEN
 moveForward()
ELSE IF
command == "BACKWARD" THEN
moveBackward()
ELSE IF command == "LEFT" THEN
 turnLeft()
 ELSE IF command == "RIGHT" THEN
 turnRight()
ENDIF
IF speedControlCommandReceived THEN
 setSpeed(speed)
 ENDIF updateLEDStatus()
 ENDWHILE
 END
```


START
Begin Infinite loop
Received bluetooth Comm
if Command == Forward
Move forward
Check Move if Command == Backward
Move Backward
Check Command == Left
Turn Left
Check Command == Right
Turn Right
Check Speed Control Command
Set Speed
update led status
END

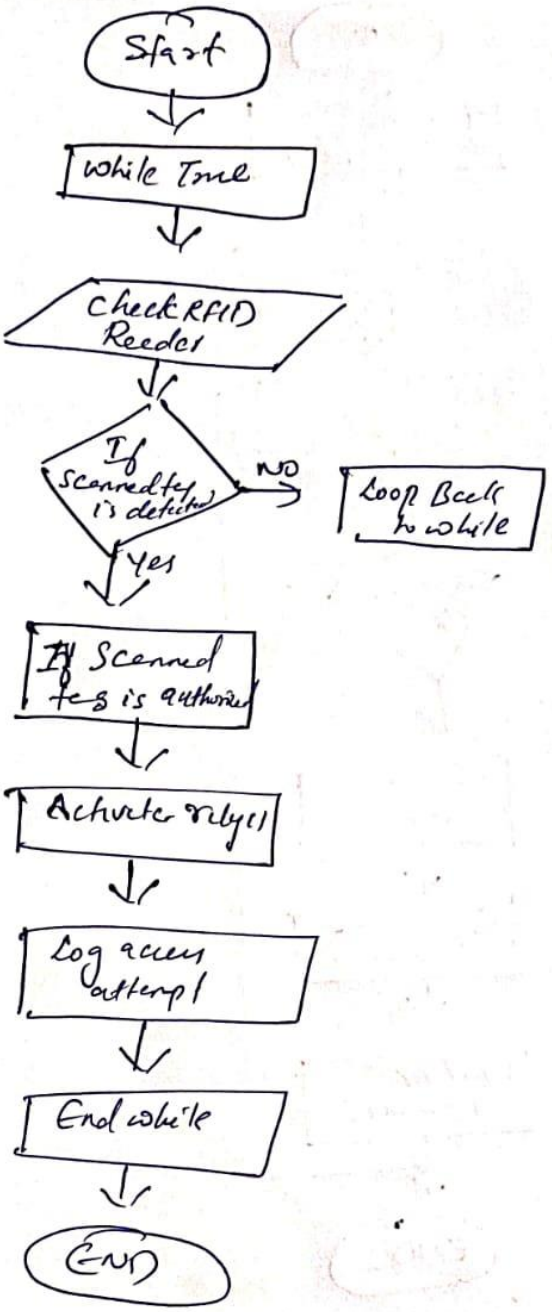
G. Battery Monitoring System Problem Statement: Develop a battery monitoring system that checks battery voltage levels periodically and alerts if voltage drops below a safe threshold. Requirements: • Measure battery voltage every minute using an ADC (Analog-to-Digital Converter). • If voltage falls below 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.

```
START
 WHILE true DO
 batteryVoltage = readBatteryVoltage()
IF batteryVoltage < 11V THEN
 triggerAlert()
 logEvent(batteryVoltage, timestamp)
ENDIF
display(batteryVoltage) implementPowerSavingFeatures()
 WAIT 1 minute
ENDWHILE END
```


Start
while true
Read battery voltage
If battery voltage<11V
Trigger alert
Display Battery voltage
Implement Power saving feature
wait 1 Min
End while
END

10. RFID-Based Access Control System

```
START
WHILE true DO
 scannedTag = checkRFIDReader()
 IF scannedTag is detected THEN
 IF scannedTag in authorizedList THEN
activateRelay()
 ELSE triggerAlert()
ENDIF
logAccessAttempt(scannedTag, timestamp)
 ENDIF
ENDWHILE
 END
```


Start
While True
Check RFID Reader
If Scanned tag is detected
NO
Loop Back to while
Yes
If Scanned tag is authorized
Activate relay 1
Log access attempt
End while
END