PROBLEM STATEMENT1: TEMPARATURE MONITORING SYSTEM

OBJECTIVE:

Design a temperature monitoring system that reads temperature data from sensor and triggers an alarm if the temperature exceeds a predefined threshold.

REQUIREMENTS:

- Read temperature data from a temperature sensor at regular intervals
- Compare the read temperature with a predefined threshold
- If the temperature exceeds the threshold, activate an alarm
- Include the functionality to reset the alarm

ALGORITHM

Step1: Initialize the Temperature_Threshold value

Step1: Initialize the temperature sensor and alarm system.

Step3: Set the alarm status as inactive, alarm_status=False

Step4: Define a function for reset alarm

Step4: Read the current temperature from the temperature sensor and store it in temperature.

Step5: If temperature > Temperature_Threshold

- Set alarm_status = True
- Activate the alarm

Step6: If alarm_status= True

- Call alarm reset function foe resetting the alarm

Step7: Wait for a specific interval of time

Step8: Return to Step 4

PSEUDOCODE

```
TEMPERATURE_THRESHOLD = 50
Temperature=get the current temperature value
alarm_status=False
function alarm_reset():
    alarm_status=False
    DEACTIVATE alarm
if temperature > TEMPERATURE_THRESHOLD
```

alarm_status=True

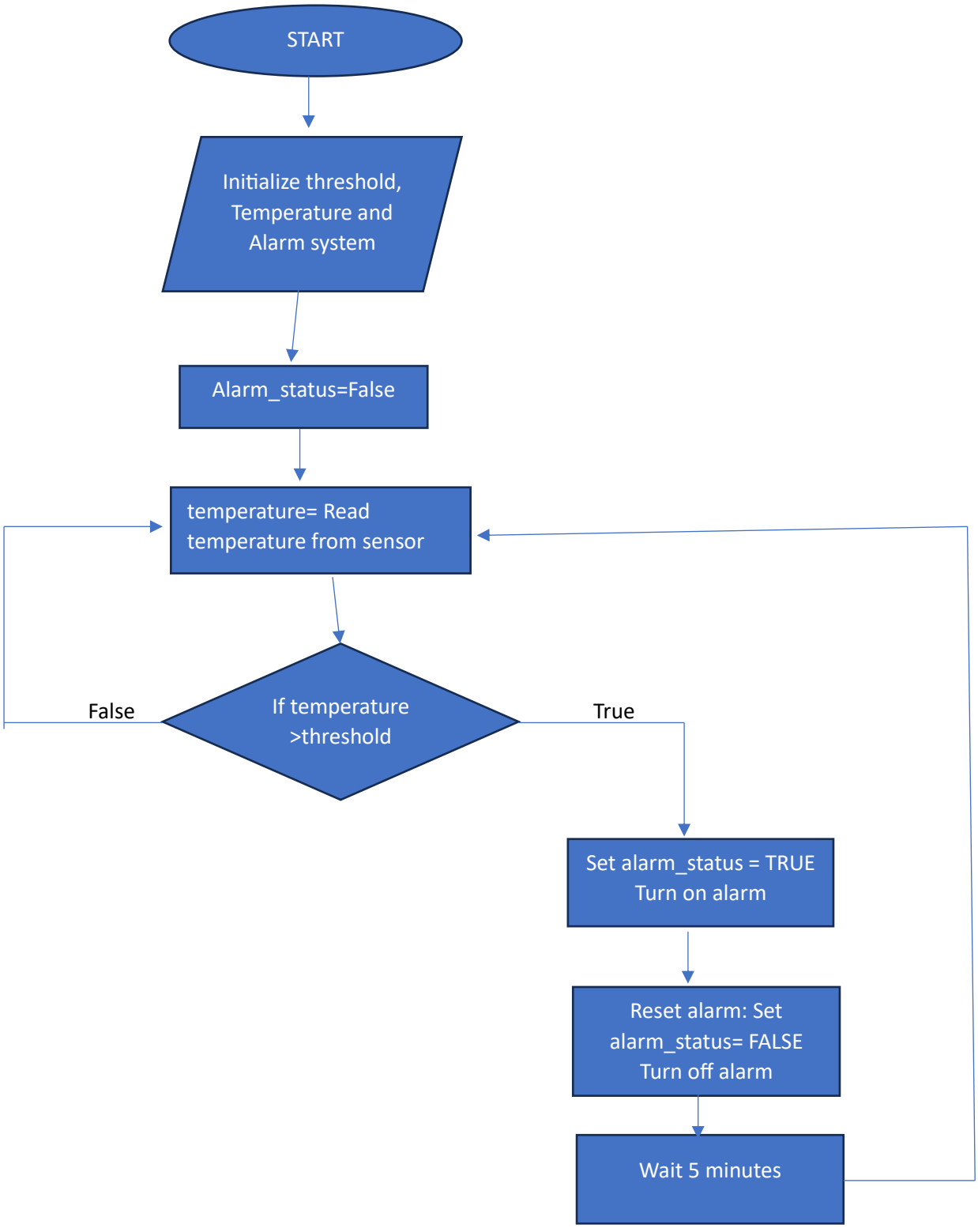
ACTIVATE alarm

If alarm_status=True:

Call alarm_reset() function

Wait (wait for 5 minutes)

FLOWCHART



PROBLEM STATEMENT2: MOTOR CONTROL SYSTEM

OBJECTIVE:

Implement a motor control system that adjusts the speed of a DC motor based on user input

REQUIREMENTS:

- Use a potetntiometer to read the user input for desired motor speed
- potentiometerValue.Display the current speed on the LCD

ALGORITHM

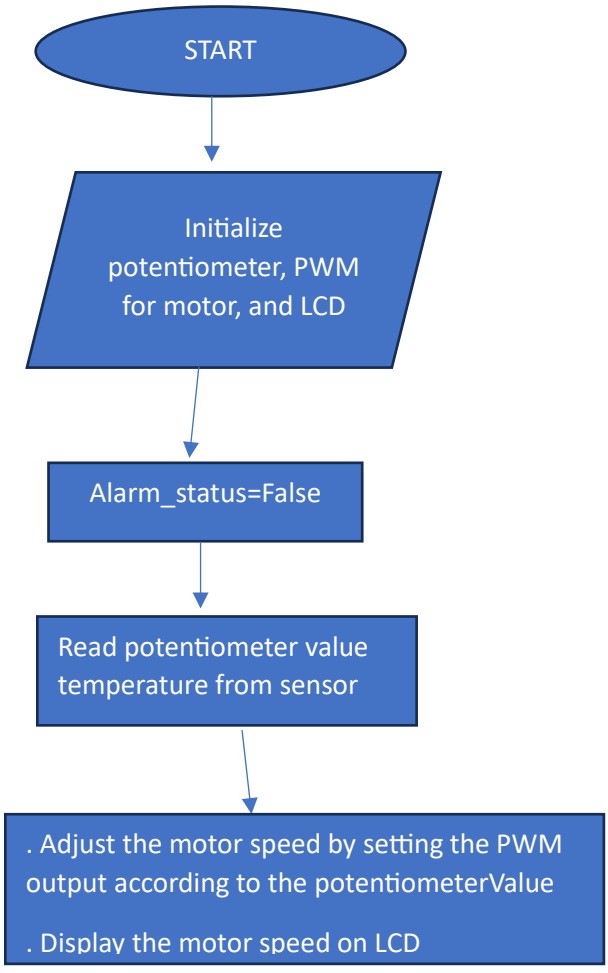
Step1: Initialize the potentiometer, PWM output for the motor, and the LCD display.

Step2: Read the input from the potentiometer and store it in potentiometerValue.

Step3: Adjust the motor speed by setting the PWM output according to the potentiometerValue.

Step4: Display the motor speed on LCD

FLOWCHART



PROBLEM STATEMENT3: LED BLINKING PATTERN

OBJECTIVE:

Create an embedded system that controls array pf LEDs to blink in a specific pattern based on user defined setting

REQUIREMENTS:

- Allow users to define blink patterns (eg. Fast,slow)
- Implement different patterns using timers and interrupts
- Provide feedback through an LCD or serial monitor

ALGORITHM

Step1: Initialize the LED array, timer, LCD , and interrupt system.

Step2: **Define** blink patterns as SLOW_BLINK and FAST_BLINK

Step3: Define interrupt handler for user input:

- If user input is "FAST", set blink pattern to FAST_BLINK.
- If user input is "SLOW", set blink pattern to SLOW_BLINK.

Step4: Display the current pattern FAST_BLINK or SLOW_BLINK on the LCD or serial monitor for user feedback.

PSEUDOCODE

Initialize LED array

Initialize Timer for delay

Initialize LCD or Serial Monitor for feedback

Define FAST_BLINK and SLOW_BLINK functions

Input=get user input as FAST or SLOW

if userInput == "FAST"

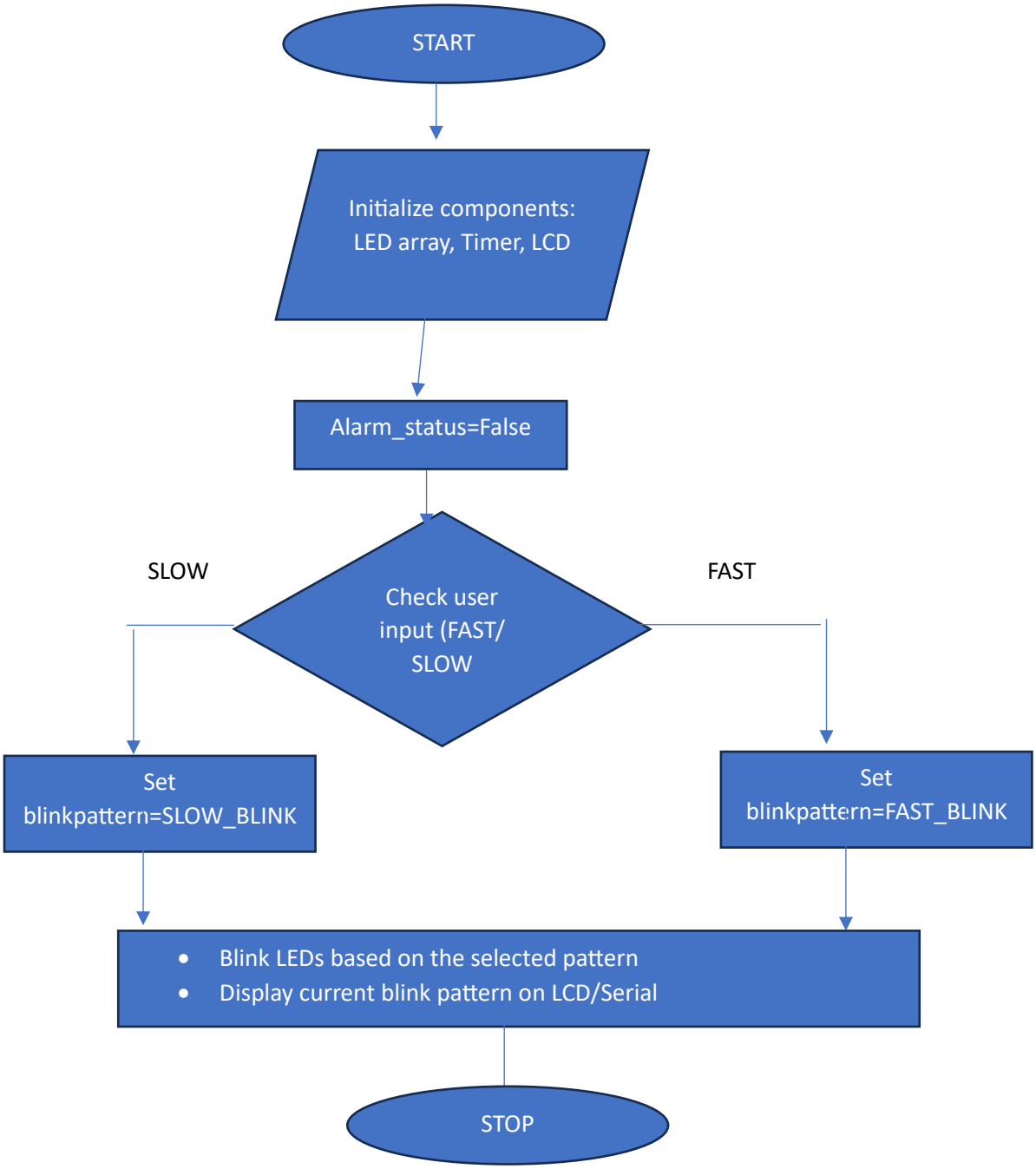
blinkPattern = "FAST_BLINK"

else if userInput == "SLOW"

blinkPattern = "SLOW_BLINK"

PRINT "Current Blink Pattern: " + blinkPattern on LCD or Serial Monitor

FLOWCHART


START
Initialize components:
LED array, Timer, LCD
Alarm_status=False
Check user
input (FAST/
SLOW
SLOW
FAST
Set
blinkpattern=SLOW_BLINK
Set
blinkpattern=FAST_BLINK
Blink LEDs based on the selected pattern
Display current blink pattern on LCD/Serial
STOP

PROBLEM STATEMENT4: DATA LOGGER

OBJECTIVE:

Develop a data logger that collects sensor data over time and stores it in non-volatile memory

REQUIREMENTS:

- Read data from sensors (eg: humidity, temperature) at specified intervals
- Store collected data in EEPROM or flash memory
- Implement functionality to retrieve and display logged data

ALGORITHM

Step1: Initialize the sensors (e.g, temperature, humidity sensors).

Step2: Initialize the non-volatile memory (EEPROM or flash memory).

Step3: Initialize a timer or interval mechanism for periodic data logging.

Step4: Set up an array to hold the collected sensor data

Step5: Wait for the specified time interval (e.g., 10 seconds).

Step6: Read sensor data (e.g., temperature, humidity).

Step7: Store the data in EEPROM or flash memory.

Step8: Repeat this process at regular intervals.

Step9: Retrieve data from EEPROM or flash memory when needed.

Step10: Display the data on an LCD or serial monitor.

PSEUDOCODE

Initialize sensors (e.g., temperature, humidity)

Initialize non-volatile memory (e.g., EEPROM or flash)

Set interval for data logging (e.g., 10 seconds)

Wait for the specified time interval

sensorData= Get temperature or humidity

Store data in EEPROM or flash memory

Function to retrieve and display data

PROBLEM STATEMENT5: SMART IRRIGATION SYSTEM

OBJECTIVE:

Design a smart irrigation system that automatically waters plants based on soil moisture levels and environmental conditions. The system should monitor soil moisture and activate the water pump when the moisture level falls below a predefined threshold.

REQUIREMENTS:

1. Inputs:

2. Outputs

3. Conditions

- The pump should only activate if the soil moisture is below the threshold and it is daytime (e.g.. between 6 AM and 6 PM).
- If the soil moisture is adequate, the system should display a message indicating that watering is not needed.
- Activate the water pump when the soil moisture is below the threshold.
- Display the current soil moisture level and whether the pump is activated or not.
- Soil moisture sensor reading (percentage).
- User-defined threshold for soil moisture (percentage).
- Time of day (to prevent watering during rain or at night).

ALGORITHM

Step1: Initialize system

- Initialize the soil moisture sensor.
- Initialize the rain sensor.
- Initialize the water pump control.
- Set the threshold value for soil moisture .
- Initialize the time of day .
- Initialize a display system to show the moisture level and pump status.

Step2: If the soil moisture is below the threshold AND it is daytime AND it is not raining:

- Activate the water pump to irrigate the plants.
- Display "Water pump activated" on the display.

Step3: Else:

- If it is raining:

  Display "It is raining, no watering needed".

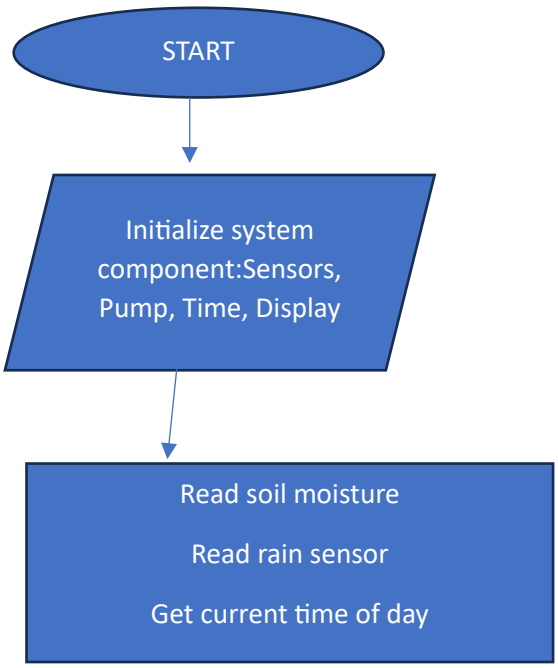
- If the soil moisture is adequate:

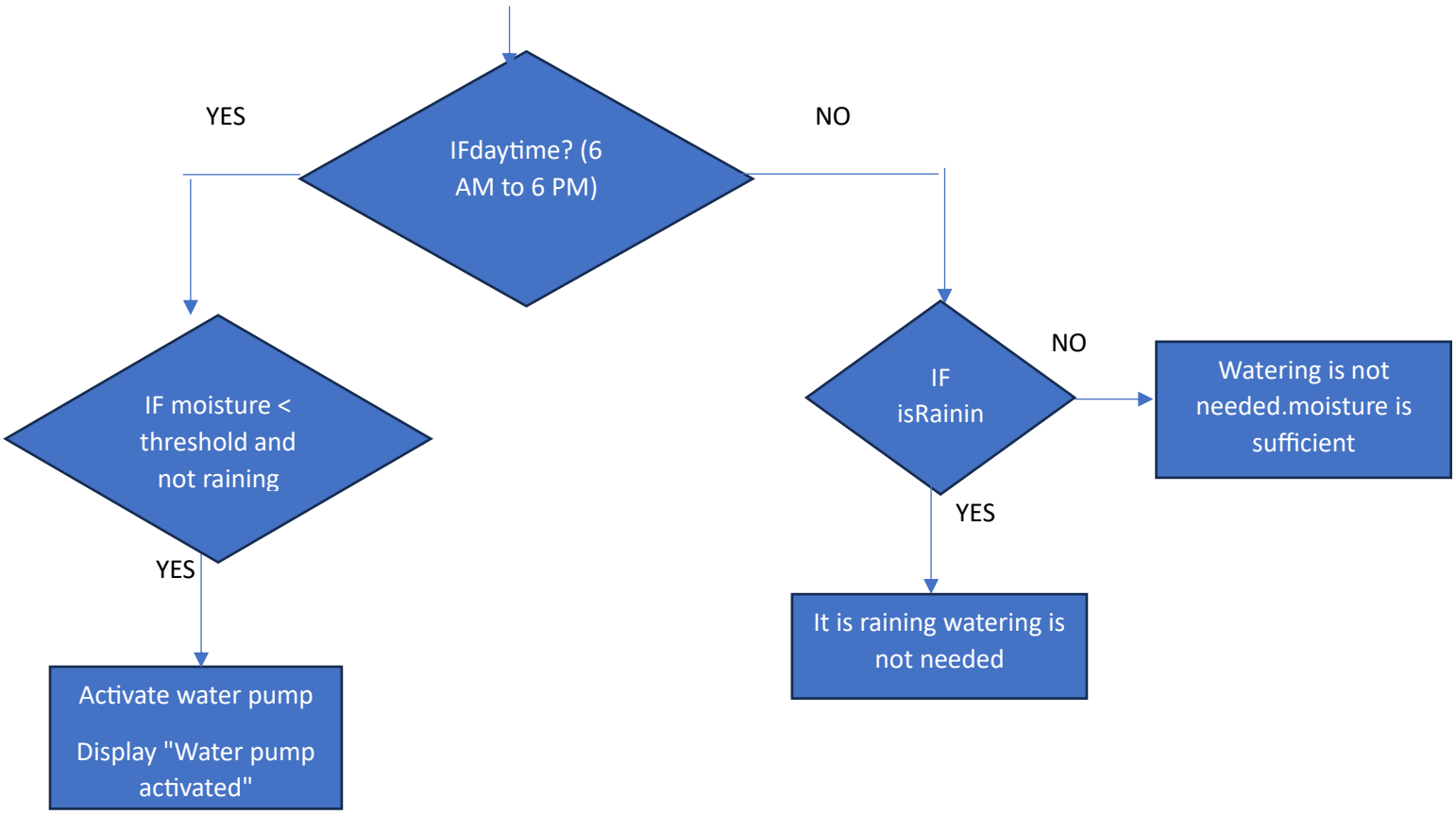
  Display "Watering not needed, soil moisture is sufficient".

PSEUDOCODE

```
Initialize Moisture
Initialize rainSensor
Initialize waterPumpControl
Set Threshold
Initialize timeChecker
Initialize display
if (currentTime >= 6:00 AM AND currentTime <= 6:00 PM)
  Daytime = TRUE
else
   Daytime = FALSE
if (Moisture <Threshold AND Daytime AND NOT isRaining)
     activateWaterPump
     print(“Water pump activated”)
else:
      if (isRaining):
         print("It is raining, no watering needed")
       elseif (Moisture >= Threshold):
          print(“Watering not needed, soil moisture is sufficient”)
```

FLOWCHART




IFdaytime? (6 AM to 6 PM)
YES
NO
IF moisture < threshold and not raining
YES
Activate water pump
Display "Water pump activated"
IF isRainin
NO
Watering is not needed.moisture is sufficient
YES
It is raining watering is not needed