Problem Statement 1: Temperature Monitoring System

Objective: Design a temperature monitoring system that reads temperature data from a 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 (e.g., LED or buzzer).
- Include functionality to reset the alarm.

```
1. start
2. define input pin for temperature sensor
3. define output pin for buzzer
4. define a threshold value
5. read the analog input from temp sensor
6. compare with the  threshold value
7. if greater than the value sound the buzzer
8. end
```

****

Problem Statement 2: Motor Control System

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

Requirements:

- Use a potentiometer to read user input for desired motor speed.
- Control the motor speed using PWM (Pulse Width Modulation).
- Display the current speed on an LCD.

```
1. start
2. define input pin for the potentiometer
3. define output pin with pwm for motor
4. define output pins for LCD
5. read the  input value from the potentiometer.
6. process the input.
7. give the processed input value to pwm pin.
8. display the processed input value as speed using an LCD
9. end
```

****

Problem Statement 3: LED Blinking Pattern

Objective: Create an embedded system that controls an array of LEDs to blink in a specific pattern based on user-defined settings.

Requirements:

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

-- ----

1. start
2. define required led pins as ouput
3. set the required registers for timer
4. set the required delay using timer
5. set the required registers for interrupt
6. write the interrupt program to blink led fast
7. slow led blink if isr has not occurred.
8. end

****

Problem Statement 5: 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 (e.g., temperature, humidity) at specified intervals.
- Store collected data in EEPROM or flash memory.
- Implement functionality to retrieve and display logged data

-- --

1. start
2. set input and output pins for desired sensors
3. read the sensor value
4. temporarily store the value in register
5. move the register value to the rom
6. access the value in rom when required
7. move the copy into ram
8. perform necessary action
9. display the value
10. end

****

Factorial Calculation
Problem Statement: Write a program to calculate the factorial of a given non-negative integer.
Requirements:
1. Prompt the user to enter a non-negative integer.
2. Calculate the factorial using a loop.
3. Display the factorial of the number.

-- ---

```
enter number;
if number<0;
print error
else if number = 0 or number = 1
print "factorial=1"
else if number>1
for(count=1 to number)
factorial=factorial*count
print "factorial"
```

****

Simple Calculator
Problem Statement: Write a program that functions as a simple calculator. It should be able to perform addition, subtraction, multiplication, and division based on user input.
Requirements:
1. Prompt the user to enter two numbers.
2. Ask the user to select an operation (addition, subtraction, multiplication, division).
3. Perform the selected operation and display the result.
4. Handle division by zero appropriately.

-- ----

```
enter number1 and number2
operation=(+,-,*,/)
for operations[any]
if +
return number1 + number2
if -
return number1 - number2
if /
return number1 / number2
if *
return number1 * number2
```

****

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.

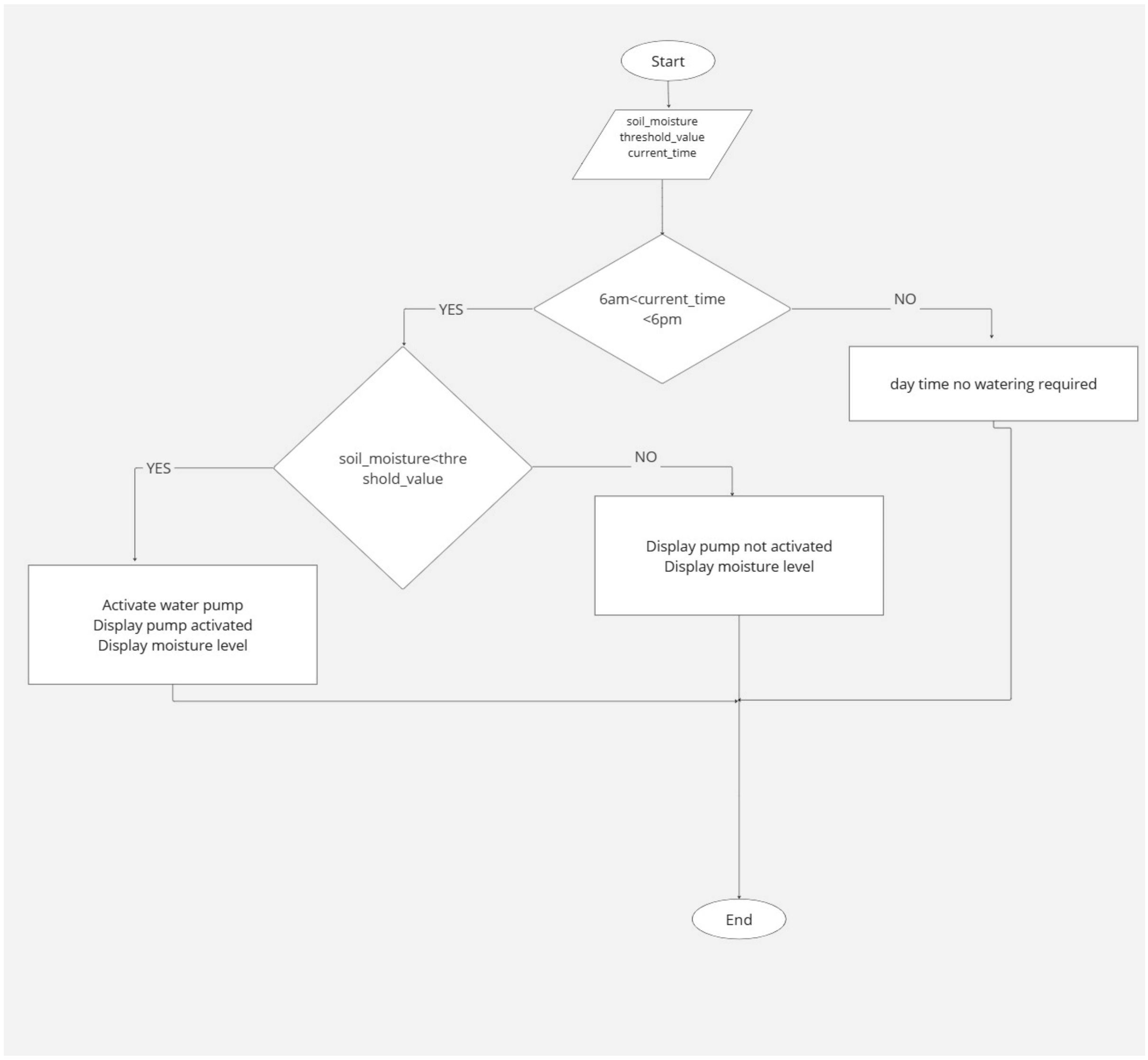
Requirements:

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)**

Pseudo code

```
get_soil_moisture(){
return soil_moisture value;
}
get_threshold(){
return threshold value;
}
get_time(){
return current time;
}
soil_moisture = get_soil_moisture()
threshold = get_threshold()
time = get_time()
 Check if it's daytime
if 6am <= time < 6pm
  if soil_moisture < threshold:
    activate_pump()
    display("Pump activated. Soil moisture: "  soil_moisture)
  else:
    display("Watering not needed. Soil moisture: " soil_moisture)
else:
  display("It's not daytime. No watering needed.")
```

Flowchart


Start
soil_moisture
threshold_value
current_time
6am<current_time <6pm
YES
NO
day time no watering required
soil_moisture<threshold_value
YES
NO
Display pump not activated
Display moisture level
Activate water pump
Display pump activated
Display moisture level
End