Quest Global Phase 1 (7/11/24)

DAY 2

Assignment (11 am)

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.

Algorithm

Step 1: Start

Step 2: Initialize the sensor, ,LED or buzzer for the alarm and temp sensor

Step 3: set threshold value

Step 4: read the temp value at regular intervals and compare it with the threshold

Step 5: if it exceeds

Step 6: activate the led or buzzer

Step 7: reset alarm

Step 8: loop back to step 4

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.

Algorithm

Step 1: start

Step 2 : Set the potentiometer to read input from the user

Step 3: set PWM and LCD

Step 4: adjust PWM and potentiometer

Step 5:Display speed on LCD

Step 6:Stop

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.

Algorithm

Step 1: start

Step 2: Initialise LED,Timers and interrupts

Step 3: Read pattern type from user and implement blinking pattern

Step 4 : provide feedback

End

Problem Statement 4: 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

Algorithm 4

Step 1: Start

Step 2: Initialize sensors, EEPROM or Flash

```
Step 3: Read sensor values and Display
End
```

PSEUDOCODE

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 non negative integer

```
Pseudocode
Input number
if(number > 0)
Set fact=1
For i = number
fact=number*i
Return fact
```

```
2. Factorial using recursion
Input number
Function factorial(number)
if(number <=1)
Return 1
Else
Return number*factorial(number-1)
```

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.

Pseudocode

```
Input num1 num2
Input operator
If (num1==0 || num2=0)
if(operator==/)
Error statement
If (operator == +)
Return num1+num2
Else if (operator == -)
Return num1 - num2
Else if(operator == *)
Return num1 * num2
Else if(operator == /)
Return num1/num2
```

Session 2

FLOWCHARTS

Problem Statement: 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:

   Conditions:

3.
   - 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).

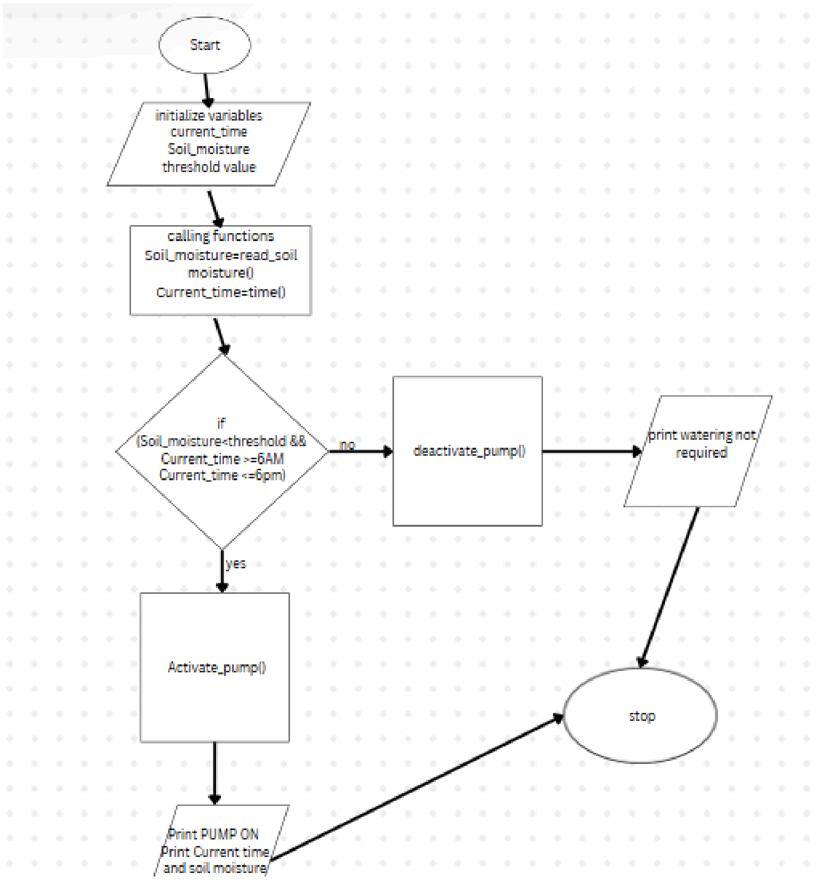
Deliverables:

- Write pseudocode that outlines the algorithm for the smart irrigation system.
- Create a flowchart that visually represents the logic of your pseudocode.

PSEUDOCODE

```
Initialize variables

Soil_moisture = read_soilmoisture()

Current_time = time()

if(Soil_moisture<threshold && Current_time >=6AM Current_time <=6pm)

Activate pump()
```

```
Print pump ON
Print Soil_moisture
Print Current_time
Else
Deactivate pump ()
Print Pump OFF
Watering not needed
```



Start
initialize variables current_time Soil_moisture threshold value
calling functions Soil_moisture=read_soil moisture() Current_time=time()
if (Soil_moisture<threshold && Current_time >=6AM Current_time <=6pm)
no
deactivate_pump()
print watering not required
yes
Activate_pump()
Print PUMP ON Print Current time and soil moisture
stop