Final Project

University of Nevada, Reno

CPE 301 Embedded System Design

Contents

1 Overview

The goal is to create an evaporation cooling system (aka a swamp cooler). In dry, hot climates, evaporation coolers provide a more energy efcient alternative to air conditioners. Air is pulled in from the outside through a pad that is soaked in water. The evaporation of the water cools and humidifes the air. As they rely on evaporation, they do not work in humid climates.

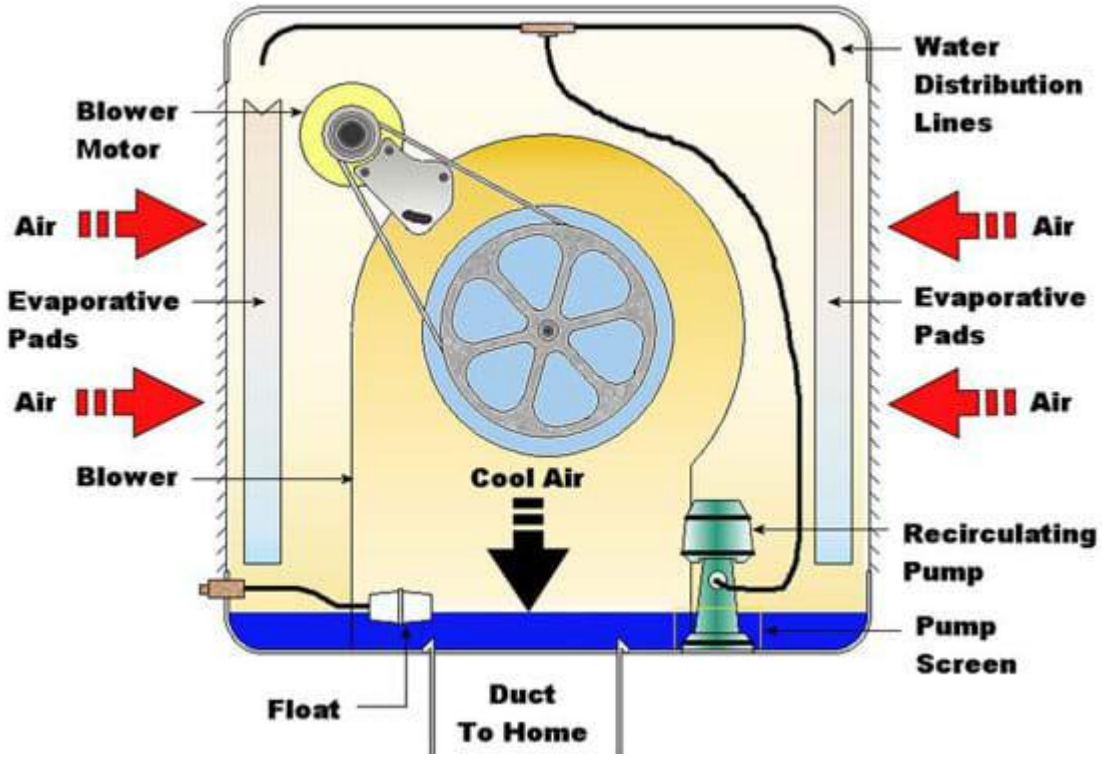

Figure 1: The general operation of the swamp cooler.

2 Project Requirements

Your team is to build a working cooler using the Arduino 2560 and sensors from the Arduino kit that we use for labs.

The completed project will

- Monitor the water levels in a reservoir and print an alert when the level is too low Monitor and display the current air temp and humidity on an LCD screen.
- Start and stop a fan motor as needed when the temperature falls out of a specifed range (high or low).
- Allow a user to use a control to adjust the angle of an output vent from the system
- Allow a user to enable or disable the system using an on/of button
- Record the time and date every time the motor is turned on or of. This information should be transmitted to a host computer (over USB)

Use of the Arduino Library

Unless it is specifcally mentioned as being allowed, you can not use library functions such as pinMode, etc. You may use the predefned macros for registers and pin positions.

Component Selection / Design Requirements

- Water level monitoring must use the water level sensor from the kit. Threshold detection can use either an interrupt from the comparator or via a sample using the ADC.

  – You may NOT use the ADC library to perform the sampling

- The vent direction control must be implemented using the stepper motor. You can use either buttons or a potentiometer to control the direction of the vent

  – You may use the Arduino libraries for the stepper motor

- The LCD display must be used for the required messages (defned below).

  – You may use the Arduino library for the LCD.

- The real-time clock module must be used for event reporting.

  – You may use the Arduino library for the clock

- The temp/humidity sensor DHT11 must be used for the temp and humidity readings

  – You may use the Arduino library for this sensor.

- The kit motor and fan blade must be used for the fan motor.

  – Be sure to use the included separate power supply board! Connecting the fan directly to the Arduino can result in damage to the Arduino output circuitry.

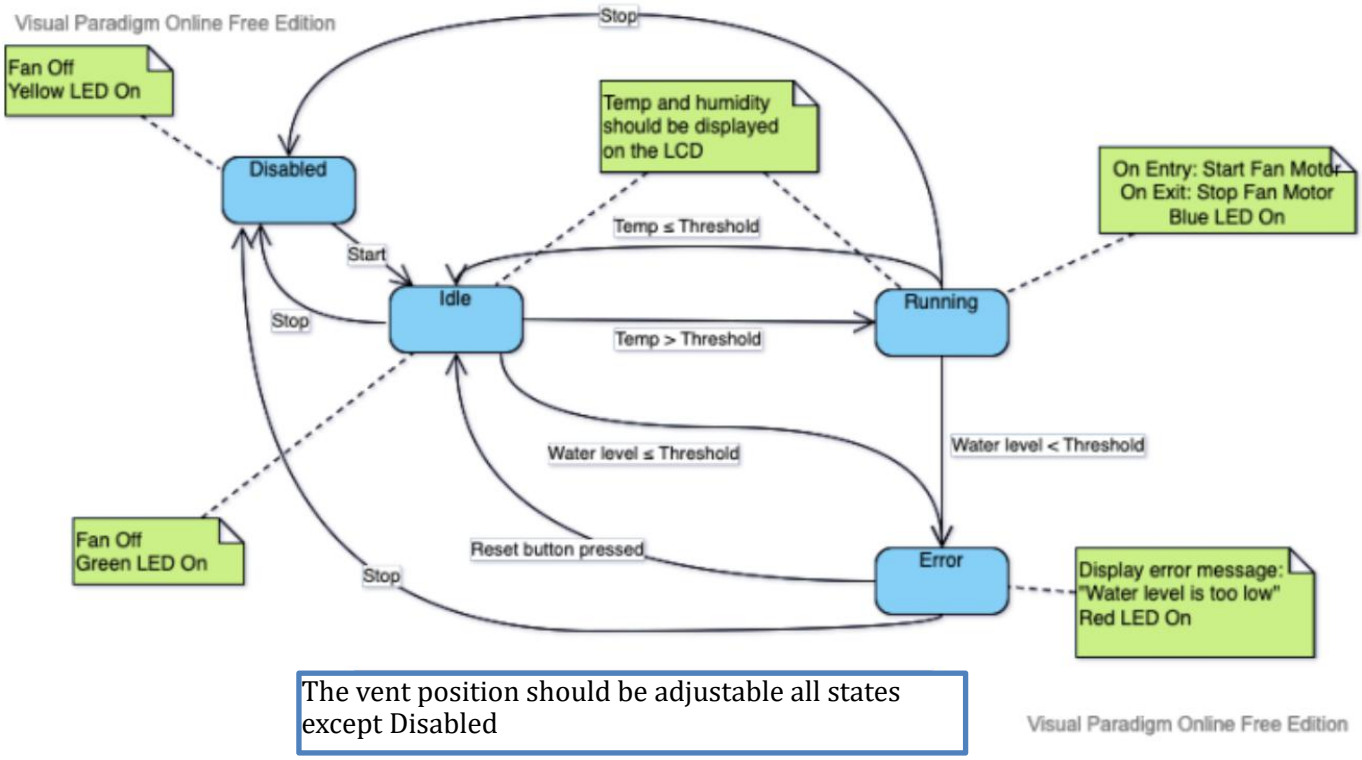

Figure 3: The state diagram for the operations of the swamp cooler.

4.1 Cooler States

The cooler continuously cycles through a number of states, with state transitions triggered by the user or by events such as temperature changes. The state diagram is shown below.

4.2 State Descriptions

Note: Some states include specifc implementation requirements, such as requiring the use of an ISR.

- All States
  - The realtime clock must be used to report (via the Serial port) the time of each state transition, and any changes to the stepper motor position for the vent.

- All states except DISABLED
    - Humidity and temperature should be continuously monitored and reported on the LDC screen. Updates should occur once per minute.
    - Stop button should turn fan motor of (if on) and system should go to DISABLED state
    - System should respond to changes in vent position control
- DISABLED
    - YELLOW LED should be ON
    - No monitoring of temperature or water should be performed
    - Start button should be monitored using an ISR
- IDLE
    - Exact time stamp (using real time clock) should record transition times
    - Water level should be continuously monitored and state changed to error if level is too low
    - GREEN LED should be ON
- ERROR
    - Motor should be off and not start regardless of temperature
    - A reset button should trigger a change to the IDLE stage if the water level is above the threshold
    - Error message should be displayed on LCD
    - RED LED should be turned on (all other LEDs turned of)
- RUNNING
    - Fan motor should be on
    - System should transition to IDLE as soon as temperature drops below threshold
    - System should transition to ERROR state if water becomes too low
    - BLUE LED should be turned on (all other LEDs turned of)

5 Deliverables

5.1 Project Overview

Final Report Guidelines

To help you prepare your final report, please follow these instructions carefully:

1. **Project Description:**
    - Provide a detailed description of your project, including the components used to build your cooler, its states, and functionalities.
2. **Component Details:**
    - Describe each component and its functionality.
    - If needed, include technical descriptions with images or screenshots of your water cooler. As mentioned in the rubric, you must describe every component used in your report.
3. **System Overview (If Necessary):**
    - Include an overview of your design, especially if there are constraints on the system such as operating temperatures, power requirements, etc.
4. **Circuit Image:**
    - Attach a clear, full-image view of your cooler circuit.
5. **Schematic Diagram (Mandatory):**
    - Include a schematic diagram of your cooler's circuit.
    - You can use a block diagram to illustrate the pin connections between components. The goal is to provide a clear visual representation of your entire circuit.
    - For drawing the schematic, you may use tools like **Google Draw**, **CirKit Designer** (https://www.cirkitstudio.com/) (recommended, free), or any platform you're comfortable with.
    - If you face difficulty creating a schematic, you can arrange the component images and show the wire connections instead.
    - Additionally, include links to relevant specification sheets for the components used.
6. **System Demonstration:**
    - Add pictures of your final system.
    - Include a link to a video of your system in operation.
7. **Submission Links:**
    - You can include your **GitHub** and **video link** directly in your report or upload everything to your GitHub and submit the link on Canvas.
    - Ensure your **GitHub repository is set to public** so it can be accessed for evaluation.

    - Additionally, include your GitHub link as a comment in your Canvas submission.
8. **Team Information:**
    - Include all team members' names in both your report and code.
    - Each group can have a maximum of **4 members**.
    - One submission from each group is alright but ensure that your report contains all of your names.
9. **Report Clarity:**
    - Ensure your report is clear, organized, and easy to read.

5.2 Github Repository

The repository link must be turned into WebCampus. The README for the project must include the group name and the names of all team members. The commits to Github will be reviewed, and will be assessed based on the comments (no "asdf" comments!) and the contributions from all team members.

Make sure to make your Github Repository public when you share your submission link.

5.3 Video Of Operation

A short video must be turned in showing the system in operation. There should be some narration as the project is demonstrated.

6 Rubrics

Out of 100 points
- 50 points deduction for no demonstration video,
- 20 Points deduction for no technical document.
- 100 points deduction for nothing in GitHub. So, make sure to check your Github time to time that if everything is committed properly with comments.
- 100 points deduction if nothing is submitted by 11:59 pm on the due date. (30 min grace will be given)
- No grade received for Plagiarism (Plagiarism is an immediate escalation to the disciplinary board).

Assuming each checkpoint is passed, the following grading scheme shall be applied to the code and demonstration:

-10 points deduction for the usage of each library that demonstrates a failure to apply each lab (GPIO, ADC, Timers, UART) Up to 40 points.
- 10 points deduction for each section of the project not attempted. Apply the ideas you have in your lab works.
- 5 points deduction for each peripheral device that does not function properly.
- 2 points deduction for each section of the technical document that is insufficient.
- 1 point deduction for each section of the technical drawing that is insufficient.

Additional Instructions for Final Project

Please follow these additional guidelines carefully:

1. **Restricted Arduino Library Functions:**
    - **Do NOT use** the following Arduino library functions:
        - pinMode(), digitalRead(), digitalWrite(), delay(), analogRead(), or any **Serial** library function.
2. **Water and Temperature Sensor Testing:**
    - To test the **water sensor**, you can sprinkle a small amount of water to trigger its response.
    - For the **temperature sensor**, simply **hold the sensor** to observe the temperature change.
3. **Vent Requirement:**
    - You are **not required** to build an actual vent.
    - Instead, attach a piece of **paper**, **wire**, **clip**, or any small visible object to the motor to demonstrate movement.
4. **Interrupt Service Routine (ISR):**
    - Implement your own **ISR** for the start button.
    - The attachInterrupt() function is allowed for this purpose.
    - Remember that attachInterrupt() only works with specific digital pins. Refer to this table for guidance:
      Arduino Reference – attachInterrupt()
5. **1-Minute Delay Requirement:**
    - For the **1-minute delay**, you are allowed to use the millis() function. Refer to this example for guidance:
      Blink Without Delay Example

6. **Fan Motor Control:**
    - You can make your **fan motor** run without using analogWrite().
    - It's acceptable to use analogWrite() for the fan motor, but **do not use analogWrite() in any other cases**.
7. **Code Usage:**
    - You are free to use any code from the **lab problems**.