CPE 301 Embedded System Design: Evaporation Cooling Unit

Ben Miller, Luke Shepard
University of Nevada, Reno

Overview:

The goal is to create an evaporation cooling system (aka a swamp cooler). In dry, hot climates, evaporation coolers provide a more energy-efficient 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 humidifies the air. As they rely on evaporation, they do not work in humid climates.

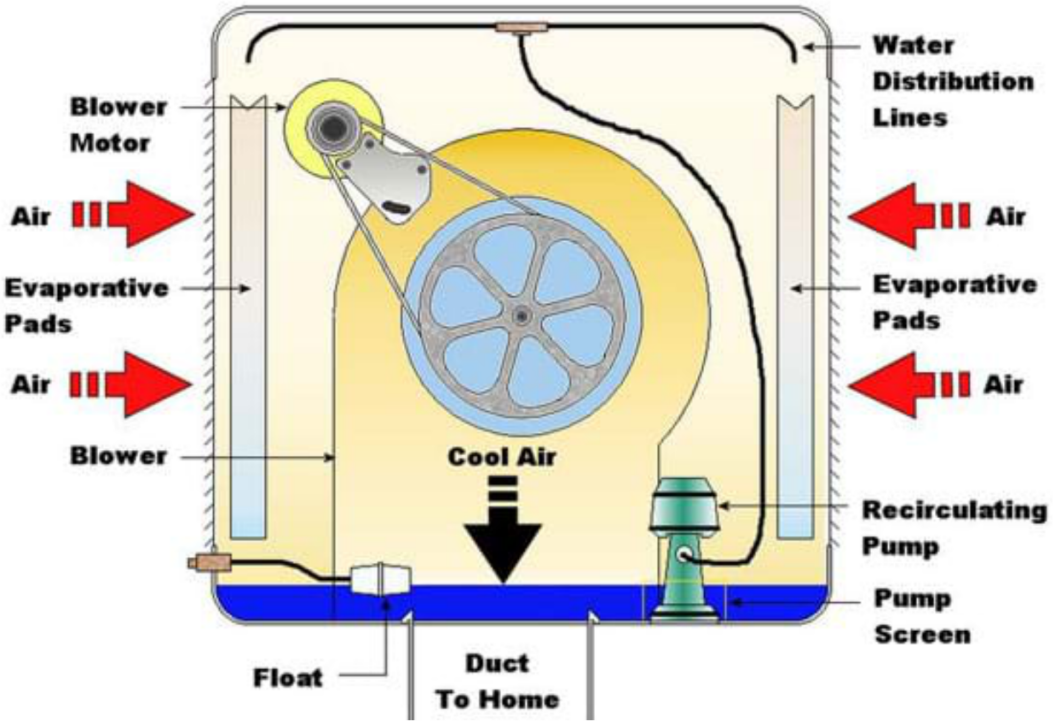


Figure 1: The general operation of the swamp cooler.

Project Requirements:

Your team is to build a working cooler using the Arduino 2560 and sensors from the Arduino kit that we used 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 specified 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/off button

- Record the time and date every time the motor is turned on or off. This information should be transmitted to a host computer (over USB)

Use of the Arduino Library

Unless it is specifically mentioned as being allowed, you can not use library functions such as pinMode, etc. You may use the predefined 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 (defined 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

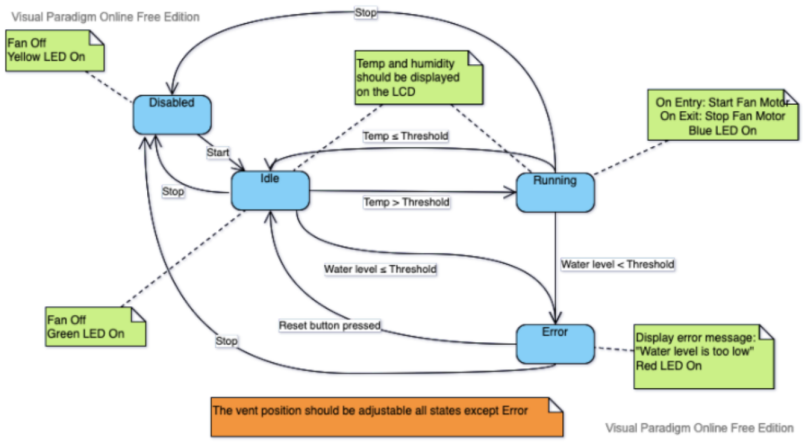


Figure 2: The state diagram for the operations of the swamp cooler

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.

State Descriptions

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

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

- All states expect DISABLED
  - Humidity and temperature should be continuously monitored and reported on the LDC screen. Updates should occur once per minute.
  - System should respond to changes in vent position control
  - Stop button should turn fan motor off (if on) and system should go to DISABLED state

- 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 the LEDs turned off)

- 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 off)

Deliverables

Project Overview

This document is a single PDF containing the following:

- An overview of the design and any constraints on the system (example: operating temperatures, power requirements, etc)
- Pictures of the final system and a link to a video of the system in operation
- A complete schematic, and links to all relevant specification sheets for the components used
- A link to the GitHub repository
- A group may consist of a maximum of 4 members

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.

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.

Design Introduction

Our final Project involves creating an evaporating cooling system, commonly referred to as a swamp cooler. In this project, an Arduino program acts as a monitoring system for temperature, humidity, and water levels. Components such as the DHT11 temperature and humidity sensor, an LCD for a text output along with some LEDs for the water level indicator sensor. The code utilizes the Liquidcrystal and DHT libraries.

In the setup function, the DHT sensor is initialized along with the pin definitions for all the other sensors and outputs. This also outputs a Booting up message when it is initially turned on.

The main loop continuously runs starting with a 1-second delay to allow for the temperature and humidity to be found and recorded, within each loop the DHT sensor readings are updated, along with the analog input from the water level sensor, which can be used for the LEDs

The water level sensor updates the LCD, and if the water level reads 100 or lower the LCD is cleared and the temperature and humidity levels are displayed, then the red LED is activated showing a loss of water

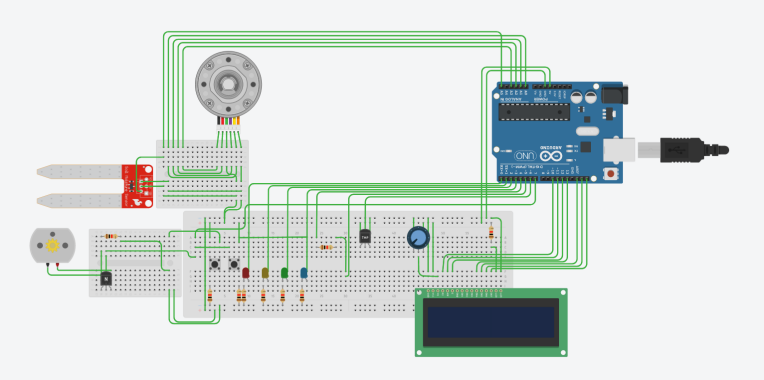
For water levels between 100 and 250, denoting a low water level, the program displays a "Water Level Low" message on the LCD's top row. The yellow LED lights up, while the red and green LEDs turn off.

If the water level surpasses 250, signaling a sufficient level, the LCD is cleared, and the green LED is illuminated. Meanwhile, the red and yellow LEDs are turned off.

GIT Repository:

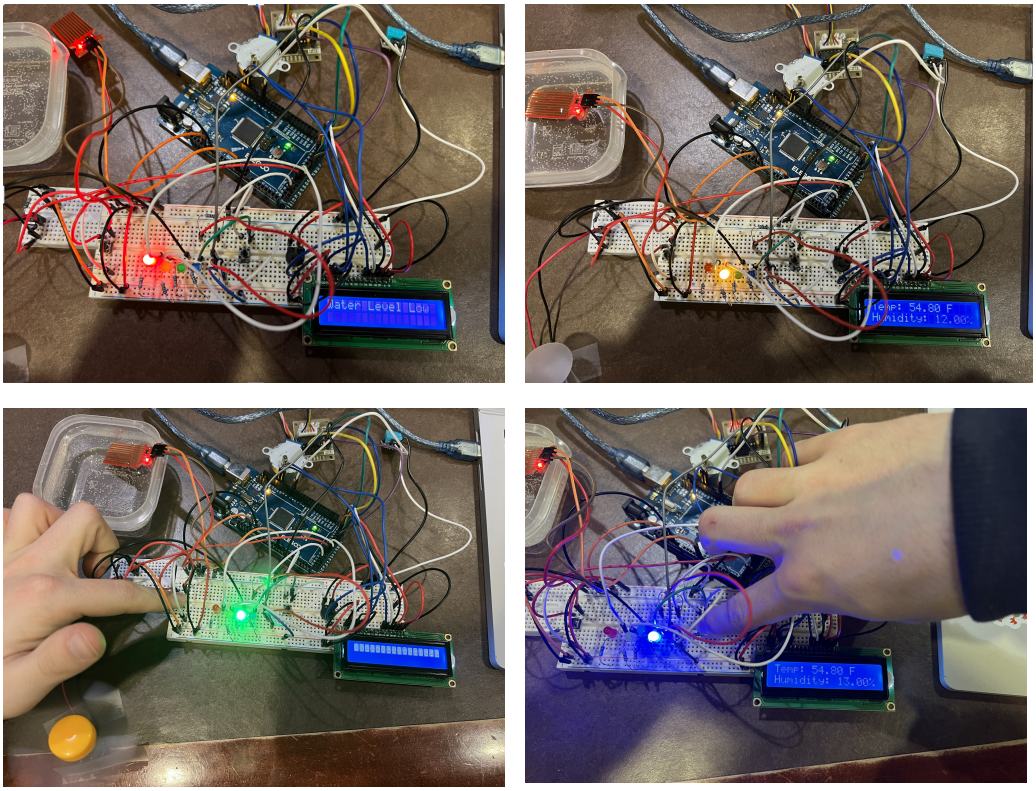
https://github.com/Luke-Shepard/CPE301-Final-Project.git

Circuit Schematic:



The circuit schematic includes an Arduino Uno in the illustration since the software lacks an available Arduino Mega representation. Nevertheless, both boards function similarly but feature distinct pin layouts.

Pictures of Circuit:



Each picture represents the different states in the circuit. See the video below for a visual explanation.

Video Link for Circuit:

https://youtu.be/sXkqvpcJqDk