Swamp Cooler

Leslie Becerra, Kj Moreno, Khoa Minh Do

University of Nevada Reno

CPE 301.1104

1 Design Overview

Our project involves building a functional swamp cooler using the Arduino Mega 2560 and a range of connected components. The system uses a DHT11 sensor to keep track of temperature and humidity, while a water level sensor ensures there's enough water for proper cooling. A stepper motor adjusts the airflow direction based on user input. Users can interact with the system through an LCD screen and several control buttons. To keep everything running smoothly, we included indicator LEDs for system status and a separate power supply to safely drive the motors.

1.1 Components Used

- **Arduino Mega2560** – The main microcontroller running the entire system, coordinating all sensors, motors, and inputs.
- **4 LEDs (Green, Blue, Red, Yellow)** – These indicate the current state of the swamp cooler (IDLE, RUNNING, ERROR, etc.) based on the system's internal state variable.
- **6mm Push Button** – Acts as the power button. It toggles the system on or off using an interrupt service routine (ISR) for immediate response.
- **LDC1602 LCD Display** – Shows real-time temperature and humidity data every 60 seconds, or displays an error if the water level gets too low.
- **DS1307 RTC Module** – Keeps track of the time to accurately log when state changes or key events occur in the system.
- **DHT11 Temp & Humidity Sensor** – Monitors the environment and provides up-to-date temperature and humidity readings.
- **Water Level Sensor** – Checks the water level to ensure there's enough for the cooling process. Triggers an error if it's too low.
- **3–6V DC Motor** – Represents the swamp cooler fan. It kicks on when the temperature is too high and shuts off once things cool down.
- **L293D Motor Driver IC** – Controls the DC motor. We only use the enable pin to turn the fan on and off—no speed control needed.
- **10kΩ Potentiometer** – Used by the user to control the vent angle. Its value is mapped to a full 360° range to set the vent position.
- **28BYJ-48 Stepper Motor** – Simulates the vent. It moves to a position based on the reading from the potentiometer.
- **ULN2003A Stepper Motor Driver** – Powers and controls the stepper motor for accurate vent adjustments.
- **Power Supply Module** – Keeps the motors running safely by supplying external power, so we don't overload the Arduino board.

1.2 Constraints

This project also presented several constraints that we had to keep in mind.

- Usage of Arduino library functions like pinMode, analogRead, etc. were not allowed; only register-level code was permitted.

- All hardware components had to come from the provided Arduino kit—no third-party modules or extra parts could be used.
- The DHT11 sensor can only operate accurately between 0 to 50°C, limiting testing to indoor room temperature ranges.
- Fan motor required an external power supply since the Arduino couldn't safely power it on its own.
- LCD display had to be positioned vertically due to breadboard space, which slightly affected readability.
- Only one LED could be active at a time without overcrowding the breadboard or causing wire interference.
- The water level sensor was tested in a small cup rather than a full tank, limiting full-capacity testing.
- Stepper motor was taped to a paper flap to simulate a vent since we didn't have parts for a physical vent.
- Real-time clock (RTC) had to be left undisturbed, as jumper wires could disconnect easily with any movement.
- Temperature, humidity, and water level readings could be slightly off if components were too close to heat sources like the motor or LEDs.
- All timing and state transitions had to be clearly reported, including vent position changes and cooler activation.
- Our power button did not always respond when pressed. We suspect this was due to either a loose connection on the breadboard or a debounce issue that caused the ISR to not always trigger reliably. This sometimes made turning the system on or off inconsistent during testing.

2 Schematic Diagram

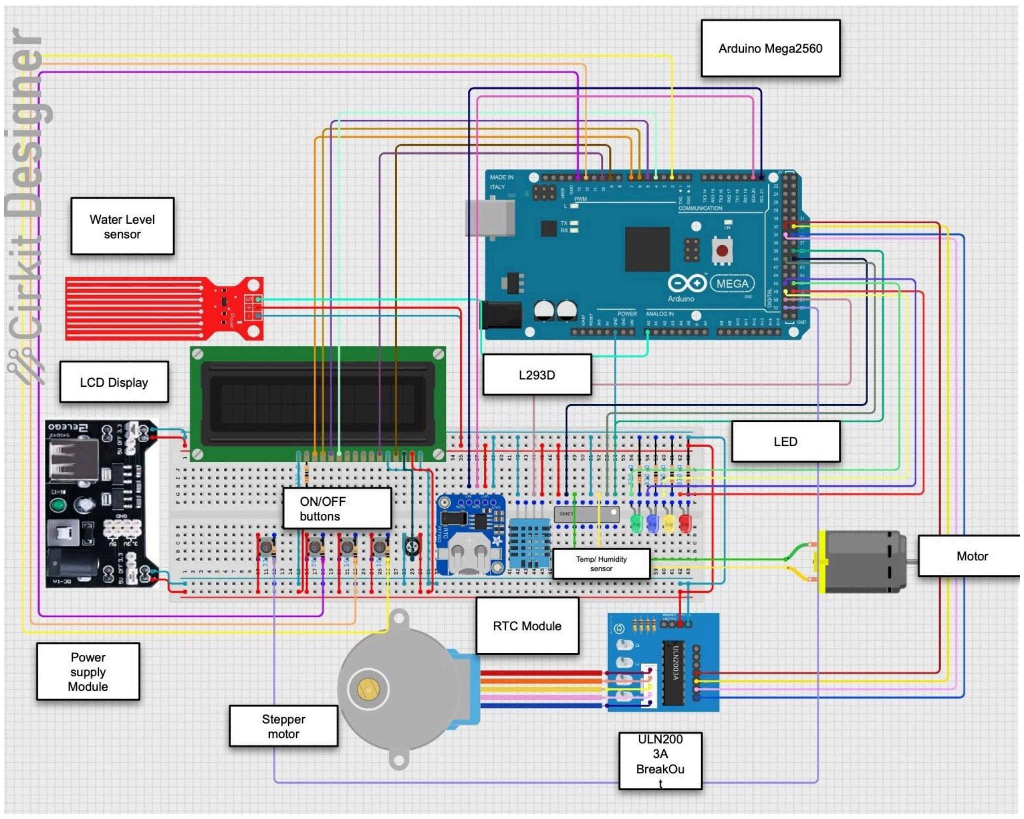

Image 1 : Diagram of Cooler Circuit

3 Physical Circuit

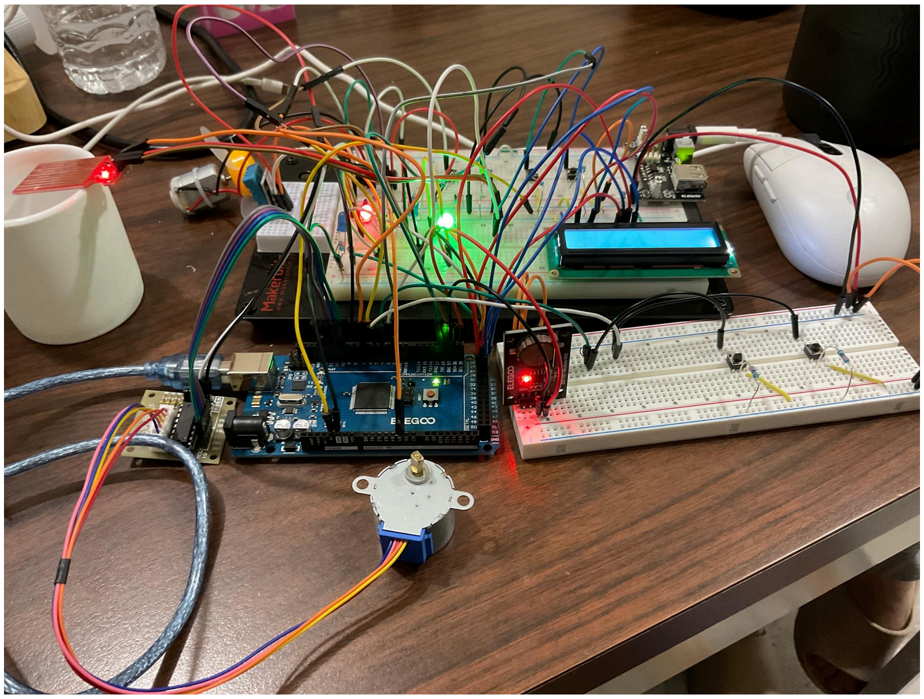

Image 2 : Circuit design

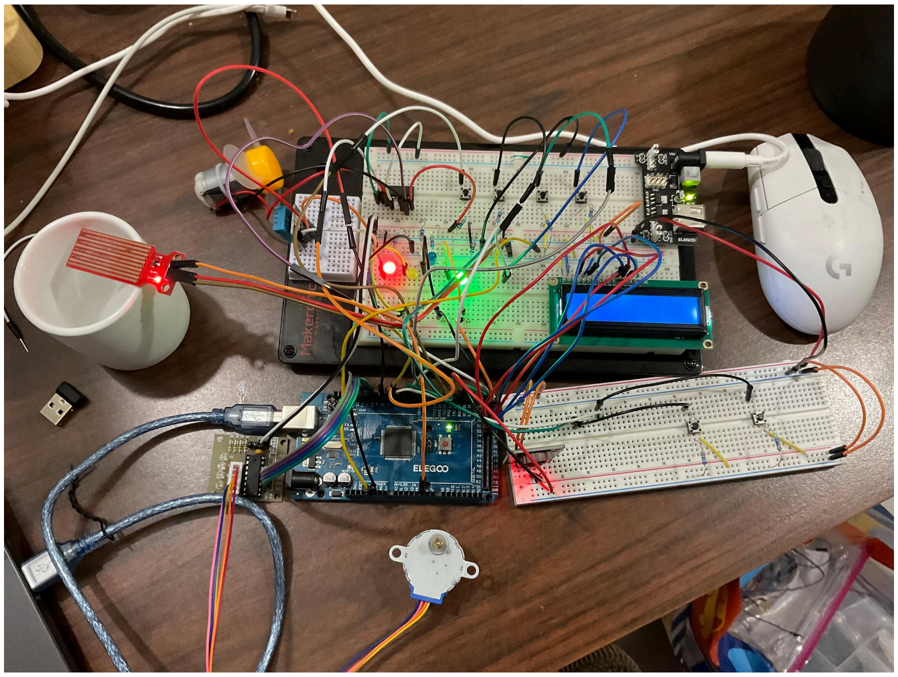

Image 3: Circuit design pt.2

4 Description

During testing, we ran into a few problems with our system. First, the power button didn't always work when we pressed it, it was supposed to trigger an interrupt, but sometimes it didn't respond at all, possibly because of a loose connection or a debounce issue. We also couldn't get the DHT11 sensor to show any readings on the LCD or Serial Monitor, so we weren't able to track temperature or humidity as we had planned. The stepper motor worked fine when tested by itself, but once we tried to run it with the full program, it stopped responding correctly, maybe due to power limits or pin conflicts. On the bright side, the IDLE state worked like it was supposed to, and the interrupt to trigger that state seemed to function reliably.