Group: Nick Campbell, Jacob Campau, Nathaniel Dulcero
Team #: 99
Date: 05/09/2025

CPE 301 Final Project Report

1 Project Description

For our final project we are tasked with creating the systems and components needed to build our own swamp cooler. Swamp coolers are meant to be used in dry and hot climates where evaporation coolers will provide more energy efficient forms of air conditioning over typical air conditioning units. They operate by having air pulled in from the outside onto a pad that is soaked in water. The evaporation caused by this cools and humidifies the air, creating an air conditioned environment contained within the cooler.

1.1 Project Components

In order to accomplish this task we used a combination of a water level sensor, a stepper motor, a LCD screen, our real-time clock module, the DHT11 temp sensor, a fan blade connected to a motor, five buttons, 4 LEDs, one potentiometer and a carefully designed program to control transitions between states. The water level sensor is utilized in order to read the level of water at the base of our cooler. The stepper motor is used as our vent direction and can be controlled by a potentiometer in order to move the direction of this vent. Our LCD display is connected in order to pass important information to the user depending on the state the cooler is in. The real-time clock module allows us to read the current time and send updates to the serial monitor as updates and important events happen within the cooler. The DHT11 sensor allows us to read in the temperature and humidity of the cooler. The information gained by this component is required if we want to make our humidity based cooler run efficiently. The LEDs work to give the user a visual understanding for the four states the cooler can be in at any moment. Three of the buttons allow us to program a start, stop, and reset functionality into the cooler. The other two buttons will give the user control over how the vent moves. The potentiometer will work to control the brightness of the LCD screen. Finally, the fan blade connected to a motor works to help cool down the cooler utilizing the water soaked pad when needed.

1.2 Project Program

Along with these components we programmed a set of 4 states the cooler can be in. These states include ‘DISABLED’, ‘IDLE’, ‘ERROR’, and ‘RUNNING’. In all of these states, we display the time of state transitions and changes to the stepper motor as they happen to our serial port. When in any state besides the ‘DISABLED’ state, we continuously report the humidity and temperature to an LCD screen. These reports are sent to the LCD screen once per minute. In addition to these reports, we have a stop button that is monitored and, if pressed, will turn off the motor and change the current state to ‘DISABLED’.

In our ‘DISABLED’ state, we set our yellow LED to be on. When in this state, there is no monitoring of our cooler’s temperature or water. This is determined by a loop checking that this

state is not active before monitoring those values. To exit this state, there will be a button to start the cooler.

In our ‘IDLE’ state, set our green LED to be on. We also ensure that time stamps are printed to our serial monitor for the times in which transitions occur. Unlike in the ‘DISABLED’ state, the water level is continuously monitored in this state. If the water level is detected as being too low, we display a “WARNING: Water Level Too Low” message to the LCD screen. We also record the temperature every minute. Using this temperature measurement we will also compare it to our temperature threshold to determine when or if to transition to the ‘RUNNING’ state. If the stop button is pressed while in ‘IDLE’, the state will be set to ‘DISABLED’.

In our ‘ERROR’ state, we first make sure the motor is turned off. No matter what temperature is read from this point, no detections in temperature or water will turn the motor back on. The message “ERROR” is sent to the LCD screen to ensure the user knows of the error and a red LED is turned on. To get the cooler to exit our ‘ERROR’ state, there is a reset button. Once pressed, the water level is read, and if high enough, will send the cooler to the ‘IDLE’ state to continue operating the cooler. If the stop button is pressed while in this state, the cooler will transition to the ‘DISABLED’ state.

In our ‘RUNNING’ state, our fan motor is turned on and a blue LED is activated for the entire time the cooler is in this state. If the temperature is read as going too low in the cooler, then the program will change from ‘RUNNING’ to ‘IDLE’. If the program instead reads that the water level is too low, then it will change to the ‘ERROR’ state. Finally, if the stop button is pressed, the cooler will transition to the ‘DISABLED’ state.

Combining these programmed states with the components mentioned previously, we are able to create the functioning parts of a swamp cooler as we were tasked to in this project. The following sections will give more details to the component operations and workings of our final product.

2 Component Details

2.1 Water Level Sensor

The water level sensor is what allows our cooler to read in the amount of water in the soaked pad within the cooler. It is what controls the transition from both the running and idle states to error when the water level goes below our determined water threshold. The sensor itself has 3 pins to connect from the sensor to our arduino. Two of them include the ground and vcc pins that are used to get the sensor working. The third pin is what sends the water level detected to our arduino.

2.2 Stepper Motor and Motor Driver

The stepper motor and motor driver work in tandem to bring together our vent system. The stepper motor is plugged directly to the motor driver via the 5-pin port on the driver. These connections are continued from the motor to the driver so they can be connected from the driver to the arduino. In addition to the 5 pins going from the motor to the driver, the driver introduces two more pins. These pins are the ground and vcc pins for the driver, both of which have a direct connection to our breadboard. Four of the five remaining pins are wired directly to the

arduino and one of the five is connected to the breadboard where it passes through a potentiometer. This allows us to control the direction of the vent with the potentiometer.

2.3 DHT11 Humidity and temperature sensor

The DHT11 is what allows our cooler to read in both the humidity and temperature of the room. These are crucial pieces of information as they are what allows the program to know when to switch between the idle and running states. The DHT11 has 3 pins in total, one contains the output of the sensor and the other two contain the vcc and ground pins. The pin for relaying the information read by the sensor is connected directly to the arduino while the other pins are connected to the breadboard for ground and power.

2.4 RTC Module

The real time clock module is what allows us to read in the time for our cooler. The cooler needs this information for when it reports on the state transitions within the serial monitor. The RTC module is connected via 5 pins to our circuit. However, only 4 of the pins are in use by our cooler. The four pins used are for ground, power, the serial data line and the square wave driver. Our system has no need for the Serial clock line, so it is left not plugged in. The ground and power pins are once again connected to the breadboard and the other pins are plugged directly into the arduino.

2.5 LCD Screen

Our LCD allows us to pass important information from the system to the user. It is connected via 12 pins from the 16 total available pins. For our use of the LCD we connected the vss, r_w, and k pins all to ground on the breadboard. The VDD and A pins are connected to the power line on the breadboard. Our V0 pin is plugged to the breadboard through a potentiometer so we can control the brightness on the LCD screen. The pins for DB4, DB5, DB6, and DB7 are all passed directly to the arduino so we can send text information to the LCD screen. The final two pins for enable and register select are also passed to the arduino for the same reasons.

2.6 Arduino Mega

The Arduino Mega is used to house all the pins and code needed to control the cooler. The arduino is the brains of the cooler and contains multiple pin inputs for the various other components to connect to. It also houses the code needed to receive and send signals to the other components. Since the code has to send information to the serial monitor on occasion, it is connected to a laptop at all times. This is how the arduino, and the rest of the circuit, gets the power needed to run.

2.7 Kit Motor and Fan Blade

Our kit motor and fan blade are used to cool down the air based on the state the cooler is in. If our cooler is in its running state, then the fan blade is turned on. It is activated by connecting the power of the fan to one of the digital pins on the arduino. This pin controls when and how much power is passed into the fan motor. The other pin from the motor goes directly to ground.

2.8 Breadboard and Controls

Controls for our design have five dip push buttons and one potentiometer to control the functionality of our design. The controls designated for the buttons are start, stop, reset, left vent direction, and right vent direction respectively. The start button begins the process of our swamp cooler design, setting the state to ‘IDLE’. The stop button stops the process of our circuit by setting the state to ‘DISABLED’. The reset button resets the cooler and preps it to start from zero in the ‘IDLE’ state after being in the ‘ERROR’ state. The potentiometer controls the brightness on the LCD screen. The final two buttons control left and right turning direction on the stepper motor vent. Our breadboard contains all of the mentioned components and distributes current to them all.

3 System Overview

Our system uses a variety of values to help make these state transitions possible. These values are our thresholds that determine how much water there should be in the system and what the surrounding air temperature should be. Through manual testing, we found that our water level threshold for our sensor would be 10. Then for the temperature readings, we found a threshold of 79 degrees fahrenheit would be a good threshold for our project. Combining these thresholds with programming logic and buttons on the breadboard allows the cooler to transition between 4 individual states smoothly and logically.

During the design process for our breadboard and arduino circuit we had to abandon our original attempt at making the whole circuit fit onto one breadboard. The original design we had was too compact and made it difficult to not only debug issues on the board, but to even use the buttons and the potentiometer on the board. Additionally, we were able to use better wires in our final breadboard that helped our board gain additional readability and clearness. The following figure is to show the original design our breadboard had.

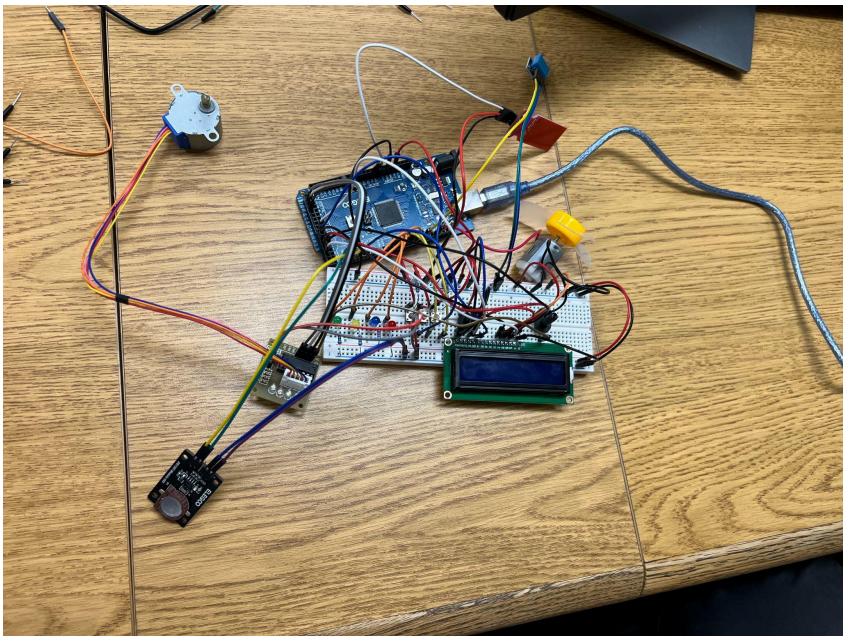

Figure 1: Original breadboard design

4 Circuit

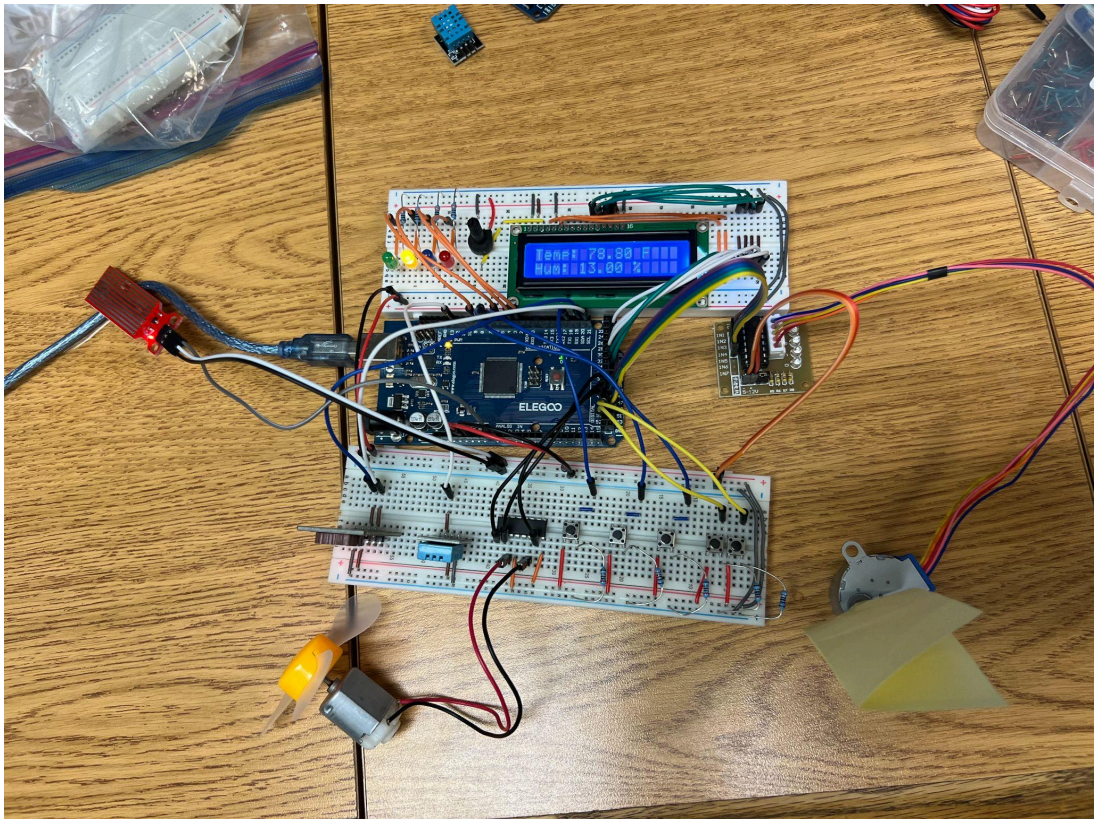

Figure 2: Full image of completed swamp cooler components

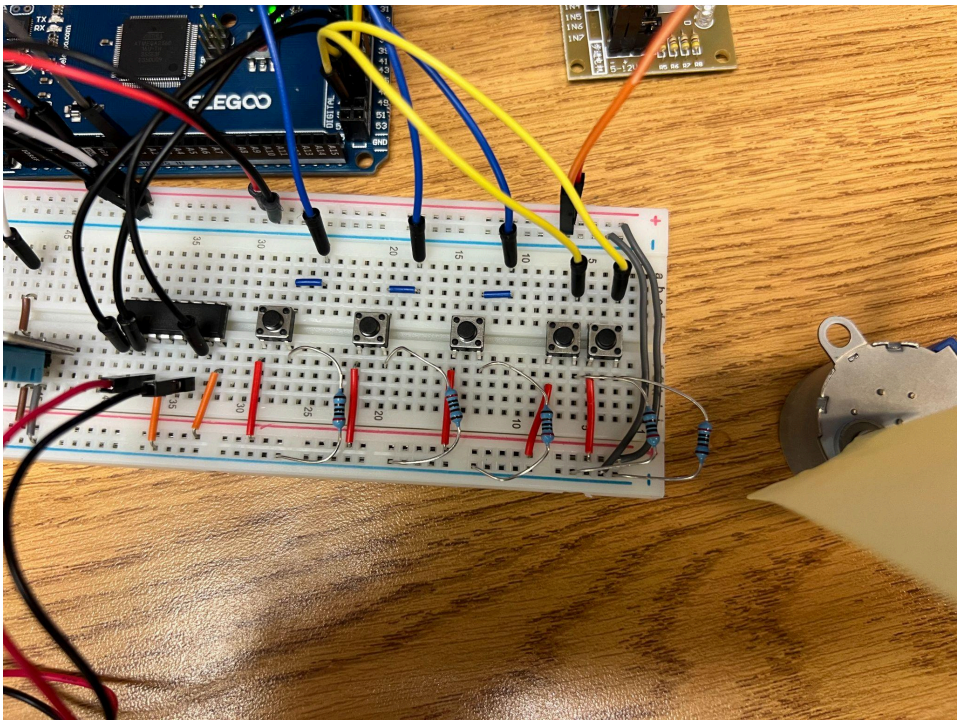

Figure 3: Image of user control interface

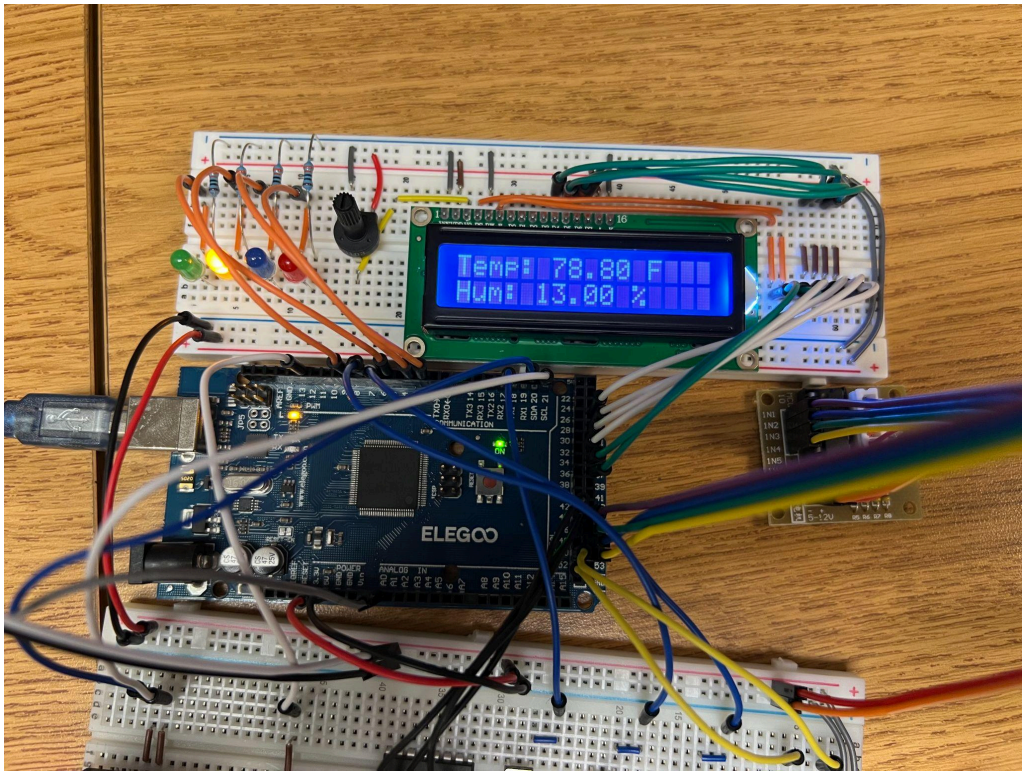

Figure 4: Image of the displayed information breadboard

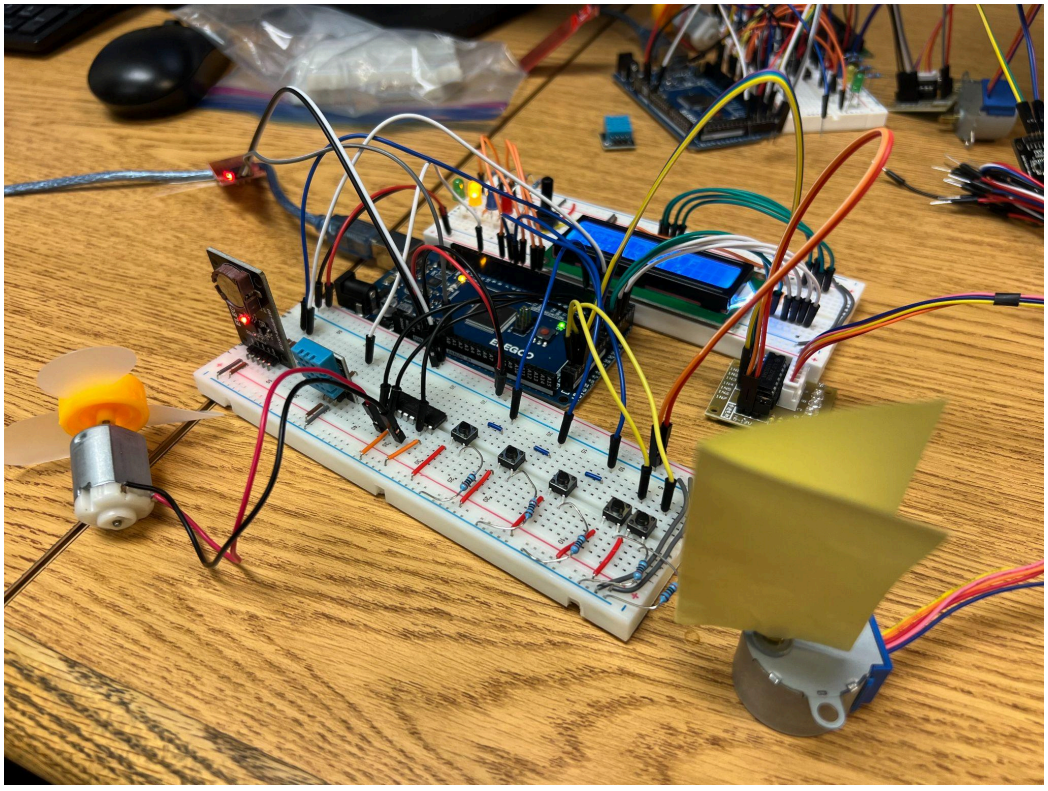

Figure 5: Image of all connected components to our second breadboard

5 Schematic Diagram

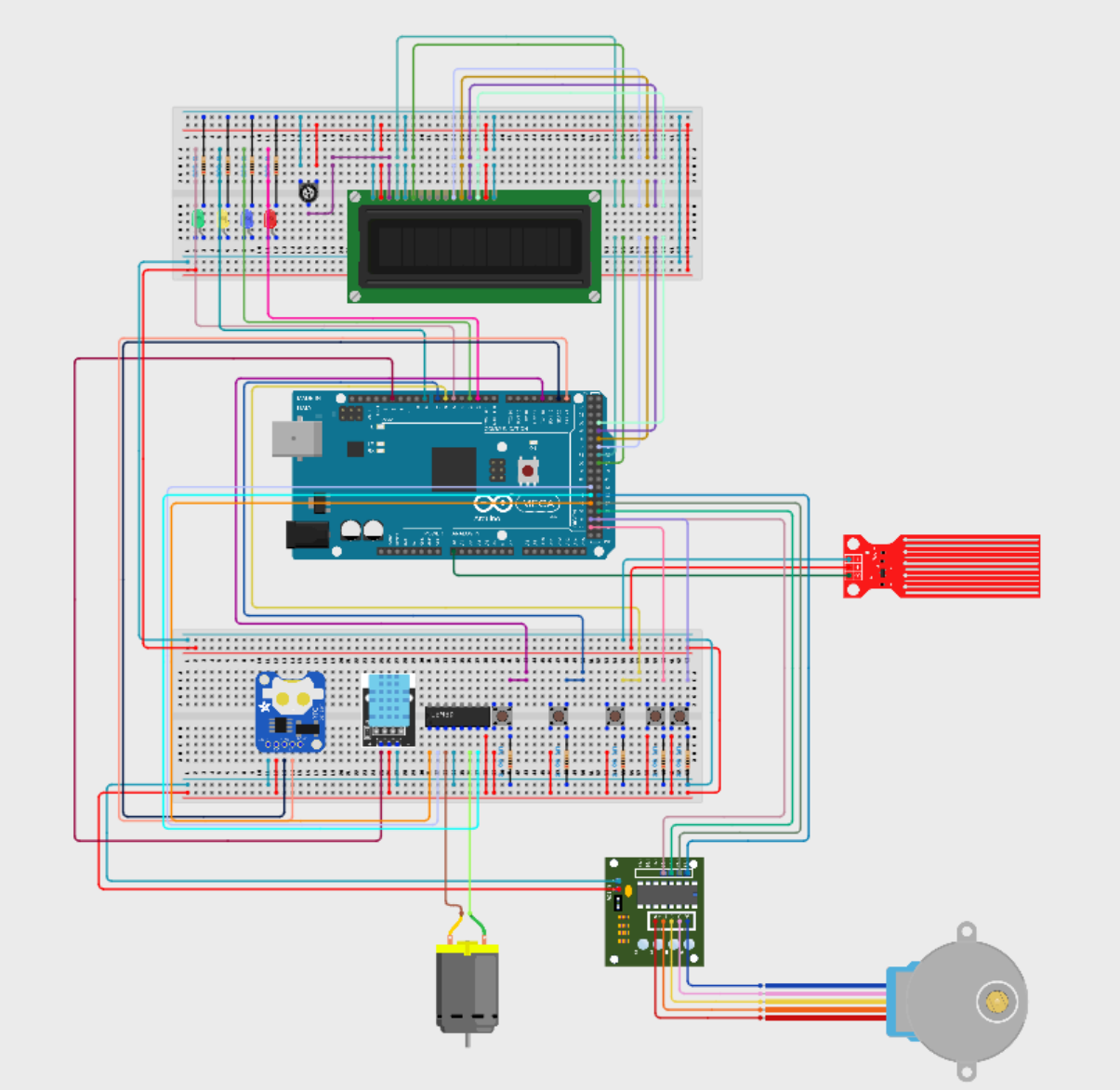

Figure 6: Full Schematic designed on cirkit

6 System Demonstration

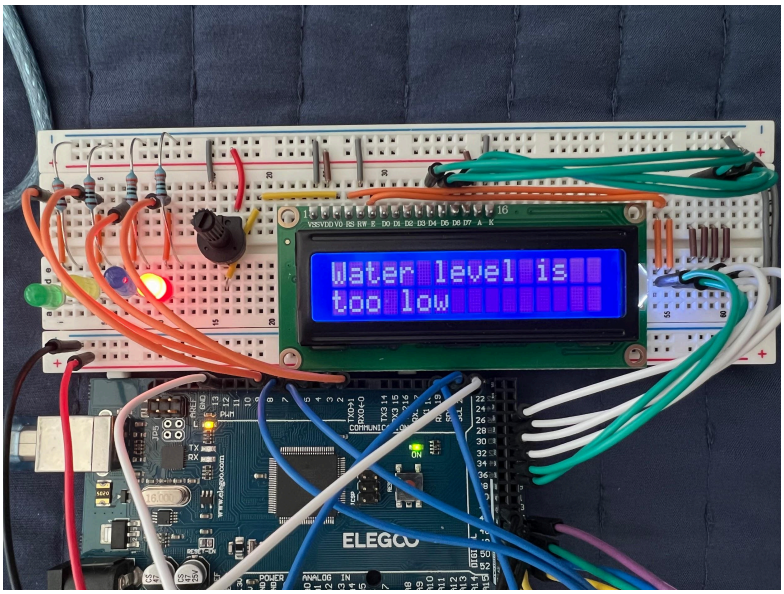

Figure 7: Image of error screen

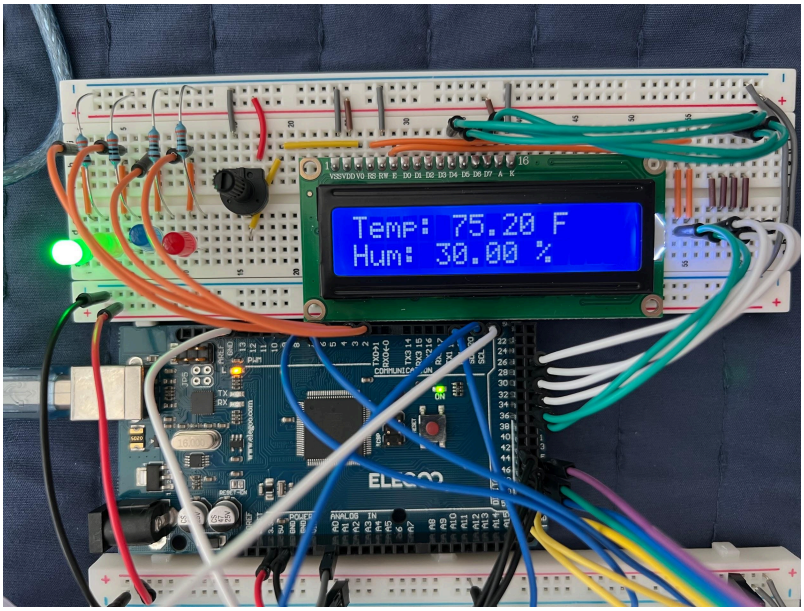

Figure 8: Image of running / idle screen

6.1 Links for the project's files and demonstration video:

Public Github repository: https://github.com/JacobCampau/SwampCooler
Video demonstration: https://youtu.be/ADKcF54Cq5g