Swamp Cooler Prototype

Colby Gramelspacher

April 4, 2025

1 Introduction

Swamp coolers are a more efficient replacement for a typical AC unit in dry and hot climates. Using the humidity within the unit itself, the swamp cooler is able to take air through its vents which are covered in wet pads. The wet pads help pull air into the unit. A small reservoir of water and a pump are used to consistently keep the pads moisturized. Using this method, the air will be cool, while also being more energy efficient but it is important to note that a swamp cooler is not effective in humid climates. The purpose of this project is to build a prototype of a swamp cooler using the Arduino Atmega 2560 and its respective kit.

1.1 Design

1.1.1 Hardware

The prototype was built of many components that would simulate what an actual cooler would function like. The basic components that we are already familiar with would be the LCD display, which was simple to put together and get operating, and the Arduino Micro-Controller that functioned as the main unit operating the entire prototype.

After getting the basic components ready, the sensors were installed. The water sensor allowed a small reservoir of water to be measured and would allow the system to know whether the water needed to be refilled. The water sensor used in the project worked well, but was slow to update at times. This wouldn't be a problem in commercial use as the sensor wouldn't be moved in and out of the liquid constantly. The final sensor used was the DHT11, which is a sensor used to measure the temperature and humidity in the air. While not extremely accurate, it served its purpose as a device that allowed the measurement of the temperature and humidity so that different system operations could take place, such as turning on the fan.

Following the sensors, the motors and motor drivers were used to further the simulation. In addition to the motors and their respective drivers, a PSU was used as the motors consumed a lot of power and a 9V adapter was needed. The first motor was the stepper motor and the ULN 2003 motor driver. The stepper motor was used to simulate a opening an closing vent due to the ability to accurately turn a stepper motor. The final motor was a DC motor and the L293D motor driver which functioned as the motor for the fan.

With the main components out of the way, the following devices were important but not as integral as the prior components. First up is the RTC device which allowed for an accurate recording of the current time and day which was used in the system. The remaining parts were buttons that allowed control of the stepper motor and acted as toggles, and the final parts were the 4 LEDs used to visualize the current state of the system.

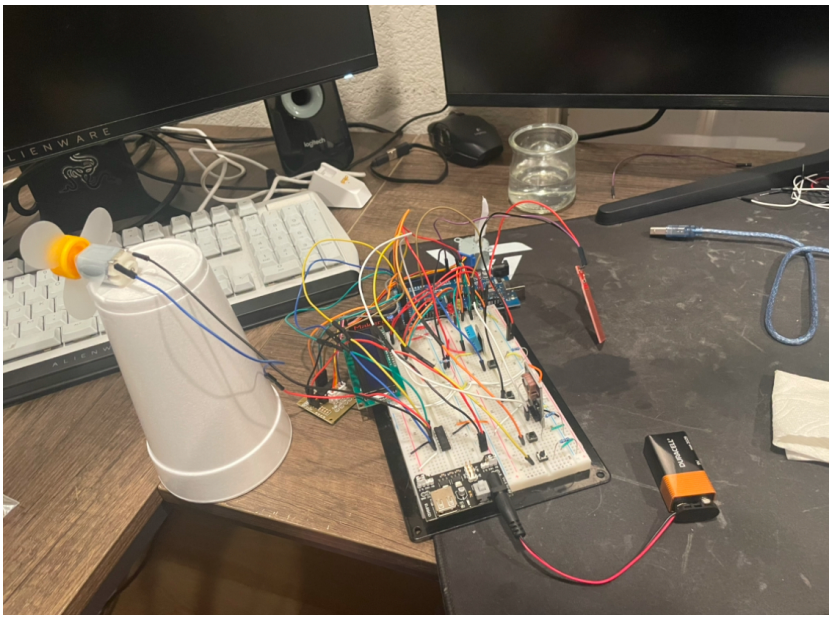
1.1.2 Software

The coding of the Arduino micro-controller was the most important part of this project as it allowed the system to function. Prior functions from the labs were used such as the UART functions, ADC functions and the attachInterrupt function. The UART functions were necessary for interacting with the serial monitor while the ADC functions were needed to read the signal from the water sensor. The attachInterrupt function was used to attach an ISR function to a specific input in the Arduino, in the case of this project, pin 19 was used.

The main system used to control the state of the system was a basic state machine built using a switch statement. The state machine controlled the flags, LED toggles and check functions when running. The check functions were simply integer functions used to determine whether a switch in state was necessary. Using the flags ,the program was able to determine what its current state was and whether it needed to switch to a different state. The flags also controlled what components were necessary for the current state of the system. While not overly complex in syntax, the program was provides many features that made debugging difficult at times.

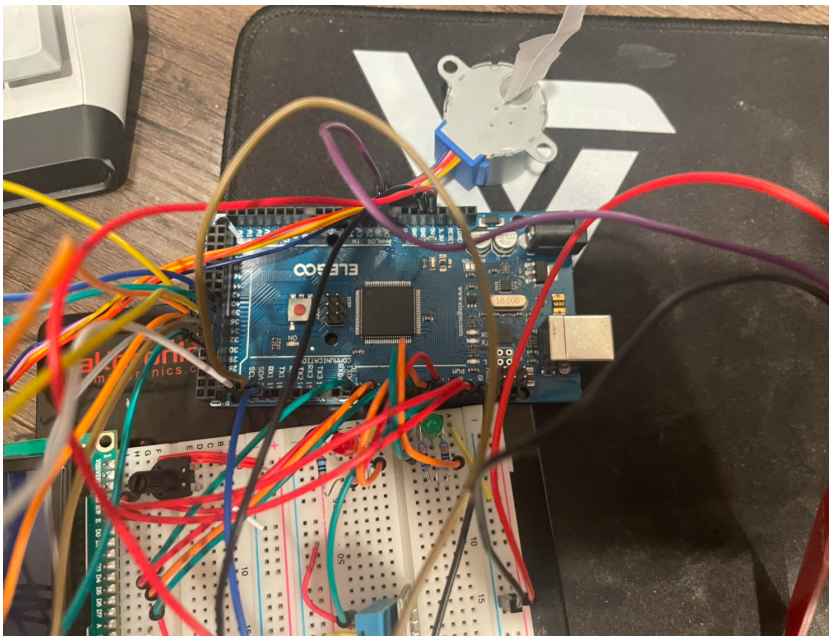
Some functions were created to reduce tedium in coding such as toggle functions for the LEDs and a varying amount of macros that allowed for a intuitive label for some hexadecimal values needed for setting and masking bits. The LED functions consisted of toggle functions but also contained the BlinkTimer function which blinked the LEDs every second. The only notable specialty of this function is the use of the millisec function opposed to the delay function. Various print functions were made for the LCD and serial monitor. Other than those main components, the main bulk of the code base is miscellaneous helper functions and variable declaration.

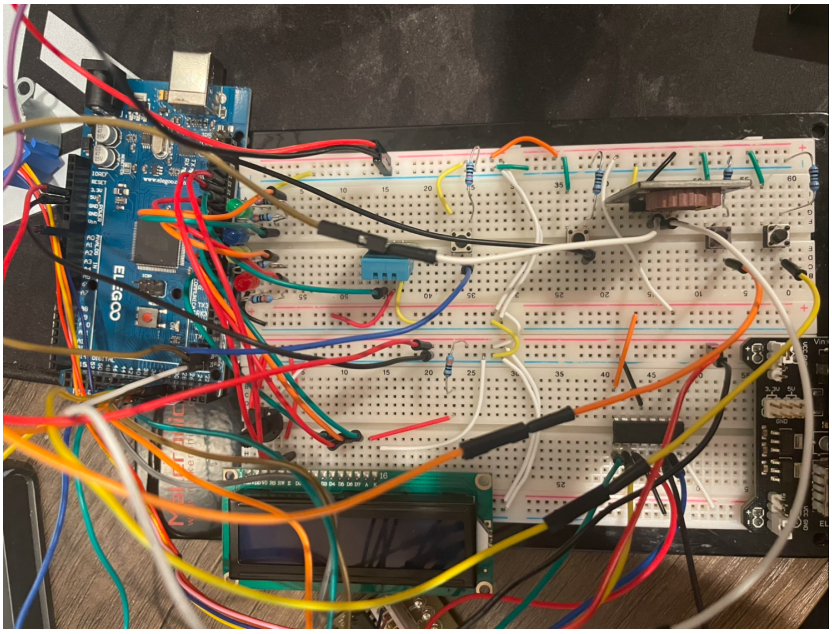
1.2 Result

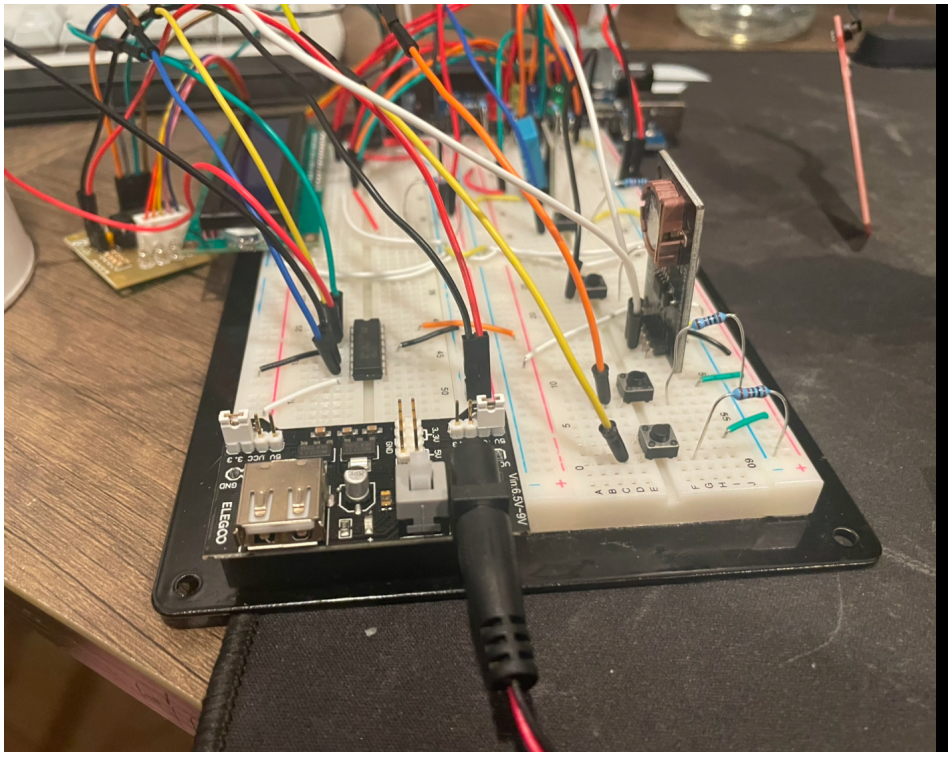


The previously mentioned design implementations allowed for the entire cooler system to come together into a functioning prototype. The starting state of the system is the disabled state. The yellow LED will be blinking and the various systems within the cooler will not be functioning, such as the motors, sensors and LCD. When transitioning from the disabled state, a on/off toggle is used which triggers an ISR that will switch the state to the idle state. The idle state represents the default state of the system and enables both sensors and the vent. The vent can be adjusted, the temperature and humidity will be read and printed to the LCD, and the water sensor will keep track of the water levels. When transitioning from idle, the system can either go to the error state, running state or disabled state. If you press the on/off toggle, the system will return to the disabled state, while the other two states are dependent on the sensors. If the temperature exceeds the threshold, the system will switch to the running state, while the state will switch to the error state if the water levels fall below the given threshold. The running state features the same actions as the idle state, except the fan motor will be turned on when entering the running state. Whenever the fan motor is turned on, the RTC will be used to record the current time and date to the serial monitor. This also happens when the motor is turned off. The running state can switch between the idle, error and disable states. In contrast to the idle state, when the system reaches below the temperature threshold, the system will switch back to the idle state. When switching between the other two states, the running state is the same as idle. In the error state, the system will display a warning to the LCD, disable all components except the vent and will require a reset. The error state can either transition to the disable state or idle state. When the water levels are restored above the threshold and the reset button is pressed, the system will switch back to the idle state. When the on/off toggle is presses, it will switch to the disabled state. Note, the vent is operational in every state except the disabled state.

1.3 Additional Pictures







1.4 Links

Schematic Link
Demonstration Video
Github Link

1.5 Resources

ULN2003 Spec Sheet
L293D Spec Sheet
DS1307 RTC Spec Sheet
LCD1602 Spec Sheet
Power Supply Spec Sheet
DHT11 Sensor Spec Sheet
Water Level Detection Sensor Spec Sheet
Arduino Atmega 2560 Spec Sheet