Andrew Kalb and Preston Peterson
CPE 301
12/13/2022

CPE 301 Final Project

GITHUB LINK:
andkalb/cpe-final (github.com)

The above github link includes each of the code files, along with a ReadMe and the circuit schematic. The circuit schematic was created using Fritzing, and edited using Paint.net.

Overview

Design Process:

We approached this project with a plan to divide the labor as follows: Andrew was to work primarily on the code for this project, and Preston was to work primarily on the physical circuit for this project. This division was not as clear cut as we expected. We began the process of completing this project by testing individual project components first. We met at common time slots and looked up the pin diagrams for a piece, found a library that worked with the device, then wrote sample code to make sure that we could get the device to work properly, at least on its own. After we had tested all of the components we intended to use for the swamp cooler, Andrew began working on the code for the swamp cooler logic, and Preston began mapping pins and designing the circuit. Once the entire circuit was built, we uploaded the code and began working through the bugs together. Much of the code written to fix and finalize the system was worked on simultaneously by us both, as we talked through solutions and attempted to solve issues that arose from the system.

System Description:

There are four LEDs on the board to display which state the evaporative cooling system is in at a given time (Disabled, Idle, Running, Error). On the main breadboard, there are also four buttons. One button is a start/stop button, that will engage/disengage the disabled state. The start/stop button is handled using an interrupt, which is waiting specifically for a pin change on the button's input. A different button is the reset button, which will kick the system out of the error state, but only if there is sufficient water detected by the water sensor. The last two buttons are for the vent direction controls, one spins the vent clockwise, and the other counterclockwise.

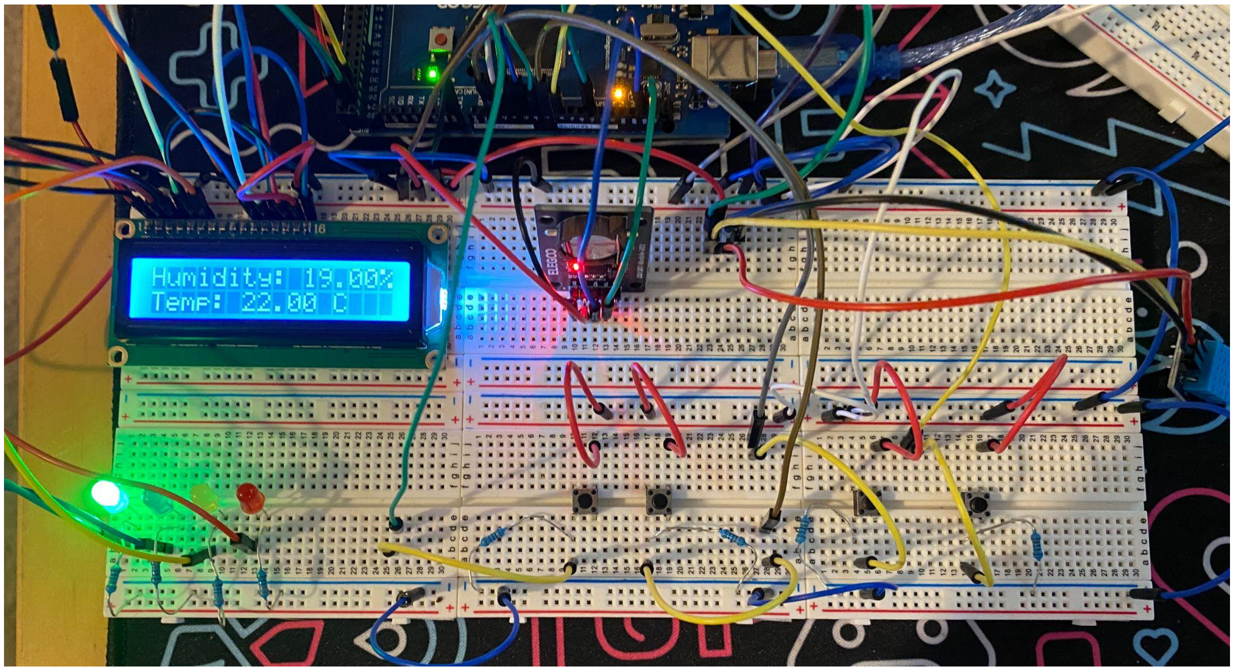

Figure 1: Primary Breadboard; the two buttons on the left are vent controls, to their right is the start/stop button, and the reset button is furthest to the right.

The water sensor performs its samping via the Arduino's ADC, and compares its measured value frequently with the determined water threshold value of 100. If the water level dips beneath 100, the system enters the error state and ceases operation until reset. The Humidity/Temperature sensor (DHT11) is monitored using the DHT11 library by Adidax. If the system is in the idle state, and the temperature threshold of 25 degrees celsius is surpassed, the system will switch into the running state. Once the temperature is lower than 25 degrees, the system will switch from the running state to the idle state. The water sensor is shown dipped in a jar of water below.

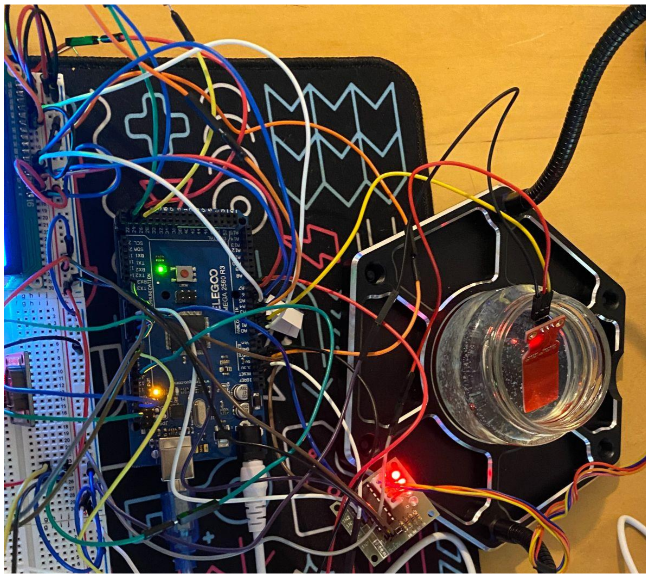

Figure 2: Water Sensor Setup

The LCD delivers temperature updates on every state change (so the user can tell why the state changed), and if no state changes are occurring, it updates once per minute. We attempted to first perform this one minute update using an interrupt on one of the arduino 16 bit timer overflows, but struggled to set up the timer to go for an entire minute, and also we experienced a high volume of weird issues from using this interrupt. The timer overflow interrupt appeared to be causing us problems in reading the temperature (perhaps because we didn't use volatiles properly, and the timer was overflowing much faster than once per minute), and prevented the system from operating correctly. Our solution was to cut out the interrupt entirely, and use the RTC module to post updates since we could easily utilize it as a timer anyway.

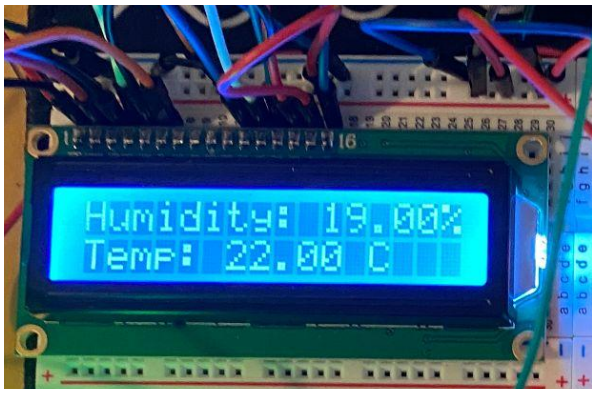

Figure 3: LCD Reading

The motor and fan setup activates when the system enters the running state. The motor is toggled on and off via severing/bridging a power connection running through the extra power supply and Arduino. The motor and fan system turns off when the running state is exited.

The real time clock module (RTC DS1307) is used in this project as a way to track certain changes in the swamp cooler system. The real time clock module is initialized in the code and synced with the computer's time, then it keeps track of time on its own and is read using the RTCLib library by Adafruit. Everytime a state change occurs, or the vent is changed, it is printed directly to the terminal using USART registers with a timestamp from the RTC.

The stepper motor is the last component in this system, and it controls the modification of the vent angle. When a vent direction button is pressed on the main breadboard, the vent slowly adjusts angle in that direction by stepping the stepper motor connected through the stepper controller. Our final issue in this project was getting the stepper motor to function, but realized it was a result of an insignificant power draw, hence fixed by the additional wall socket connection on the arduino and a move to the 5V power rail.

All States:

Every state change prints a timestamped event log entry into the terminal on the Arduino IDE using USART registers.

Disabled State:

In the disabled state of this swamp cooler, there is no code to loop through for our project. The state is completely empty and is only exited when the start button interrupt occurs. The LED is initialized to be yellow at the beginning of the program, since disabled is the default state, and then the LED is turned off when the interrupt occurs and state changes.

Idle State:

When the idle state of the swamp cooler is entered, other LEDs are turned off and the green LED is turned on. The LCD refreshes at this time. The idle process loop code first checks to see if vent buttons are being pressed, then handles the vent accordingly by stepping with the stepper motor. Then, a check is performed to see if a minute has passed since the last Liquid Crystal Display update. If a display is necessary, it is handled by a function called in the idle process loop. The temperature is checked and if it is above the threshold constant set at the top of the file, it will switch the system to the running state. The green LED will be turned off, the blue LED will be turned on, and the fan power connection will be bridged and it will begin spinning. If the water level is below the water threshold constant, the system will turn off the green LED, activate the red LED, and enter the error state.

Enabled State:

The enabled state is characterized by a lit blue LED, and a spinning fan. This state's process loop code begins by handling vent buttons, same as the idle state. It then also handles LCD updates the same as the idle state, and monitors the water level and potentially enters the error state if it is too low. It also performs a temperature check, and if the temperature is lower than the threshold, it will transition the system back to the idle state (green LED is turned on, blue LED is turned off). The fan power connection is severed when the running state is exited.

Error State:

The error state logic is very simple. In this state, a red LED is lit, and the only check that is performed in its loop is monitoring the status of the reset button. Once the reset button is pressed, it immediately checks the water level to see if it is sufficient (above the threshold) to exit the state, and then it moves to the idle state if this condition is met.

Power Requirements and Thresholds:

This system requires a power connection to a wall socket in addition to the serial port connection. The temperature threshold for this swamp cooler is set at 25 degrees celsius, and the water level threshold is set at a water level value of 100.

Final System

Here are some images of the final build:

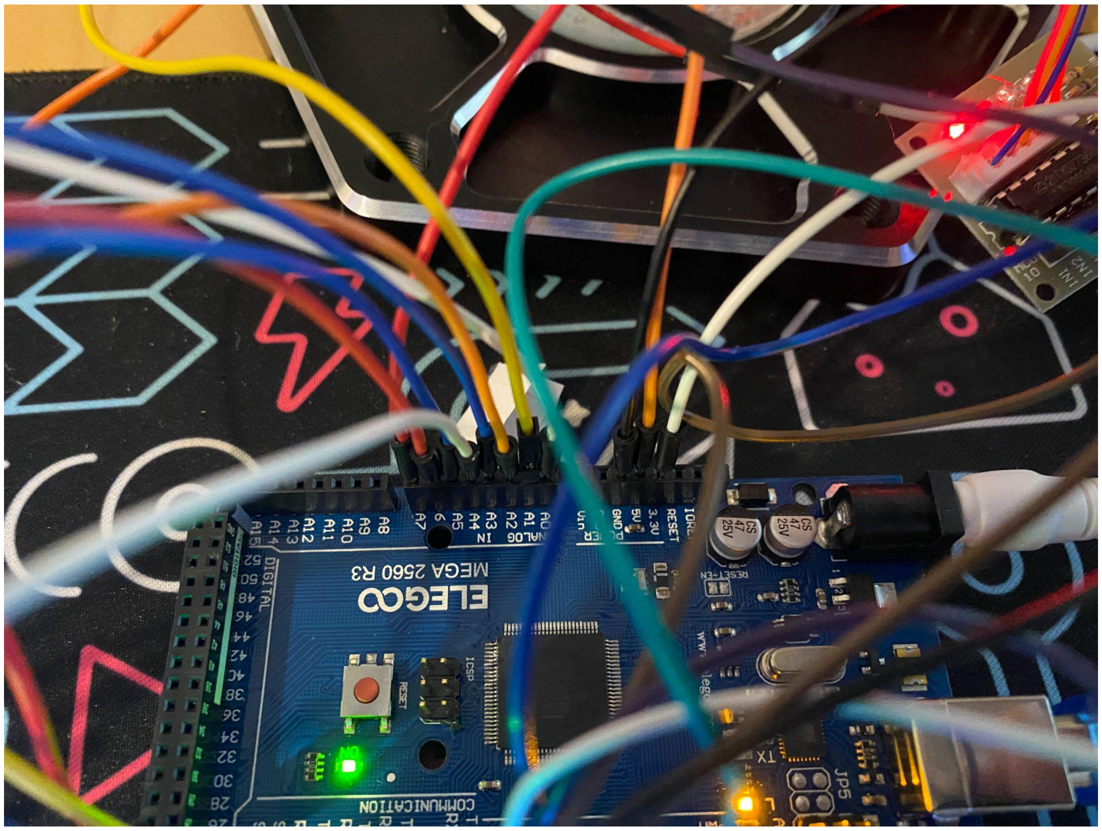

Figure 4: Arduino Analog Pin Setup

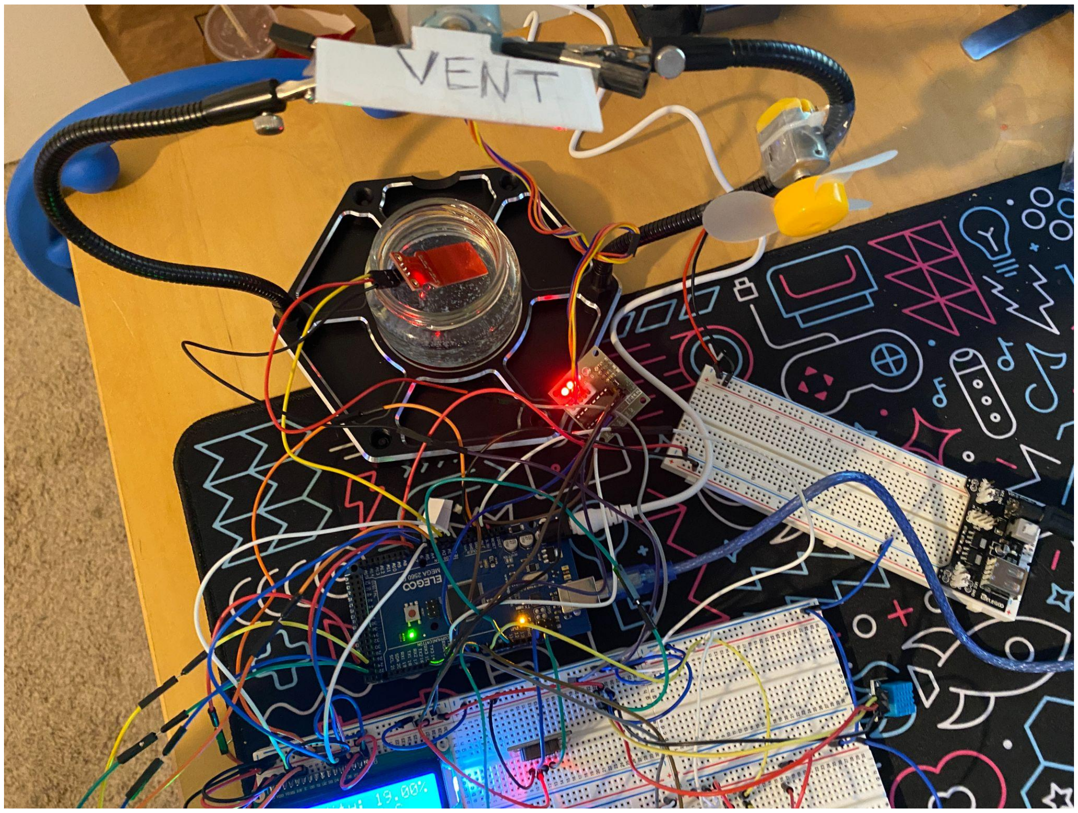

Figure 5: Fan Motor, Stepper Motor, and Water Sensor Setup

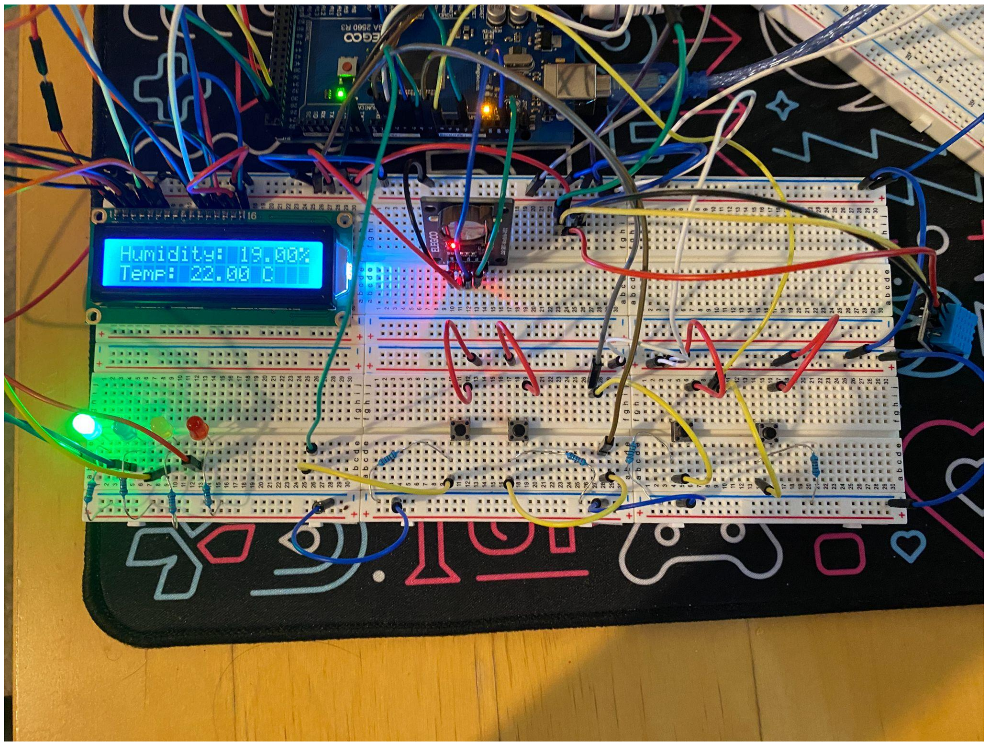

Figure 6: Primary Breadboard Setup, Including: LCD, State LEDs, Push Buttons, RTC, and Temperature and Humidity Sensor

Here is a short video demonstrating the capabilities of our project:

https://youtu.be/EuL71fJZUrA

Schematic and Resources

Arduino ATMega2560 pinout image - Arduino-Mega-Pinout.jpg (1000×951) (electronicshub.org)

Arduino ATMega2560 datasheet - ATmega640/1280/1281/2560/2561 datasheet (microchip.com)

Arduino Water Level Sensor - Arduino - Water Sensor | Arduino Tutorial (arduinogetstarted.com)

Arduino Stepper Motor - Getting Started With Stepper Motor 28BYJ-48 - Arduino Project Hub, 28BYJ-48 Stepper Motor with ULN2003 + Arduino (4 Examples) (makerguides.com)

Arduino Liquid Crystal Display - Liquid Crystal Displays (LCD) with Arduino | Arduino Documentation | Arduino Documentation

Arduino RTC DS1307 - Arduino Real Time Clock Tutorial Using DS1307 : 4 Steps - Instructables

Arduino DHT11 Humidity/Temperature Sensor - Using DHT11 - Arduino Project Hub

Toggling Fan Power Tutorial - Arduino - Controls Fan | Arduino Tutorial (arduinogetstarted.com)

Libraries:
DHT11 Sensor - adidax/dht11: DHT11 library for Arduino (github.com)

Arduino LCD - LiquidCrystal by Adafruit

Arduino Stepper Motor - Stepper by Arduino

Arduino RTC - RTCLib by Adafruit