CPE 301 Swamp Cooler

Jalen Banks
Kaiden Bell
Jeongsu Shin

Project Description

Temp and Humidity Sensor

In this project, the temp and humidity sensor give us data in the form of degrees Celcius and our percentage of humidity around the sensor. The range for these values varies, going from 20% to 90% for humidity and about 0-50 Celsius for temperature. This is perfect for our project, as temps really don't reach that high or low. Same goes for humidity. We made sure our module worked by checking its functionality first. We started the component up with library code first, then got a reading through the serial monitor. After that we implemented our code to test the functionality of the LCD Screen with our reading, and integrated both components together. This component was connected through analog pins, meaning there were some addresses in the code to convert it to digital data.

Water Level Sensor

The Water Level Sensor returns analog values which must be converted by the Arduino's ADC. From testing, it seems that the water level sensor returns values in the range of 0-320 representing completely dry and completely submerged (just below the components I wasn't sure if I should get wet or not) respectively. The system will go into an error state when the **water level is below 50**, which means that about ⅙ of the max water is available. I was observing that even when removed from the water the sensor was reporting a level of ~10, the level of 50 also gives some buffer for residual water to remain on the sensor which was the cause of this phantom level increase. This module was tested by repeatedly dipping it into a glass of water, drying it off, shaking it dry, etc.. and reading off what the Serial monitor said the level was.

Step Motor and Module

The step motor was included in the project to act as a vent control to our system, to control the flow, temp, and direction of our coming out. There is no data returned from this component, only movement that is controlled by the user with the use of a potentiometer. The user has the capability to "open" or "close" the vent. This code wasn't too hard to implement, as functionality only occurs when input is read from the potentiometer. To test this we used a similar process as the humidity sensor. We uploaded a library to get the component running, then set delays to stop it, then turn it back on after a certain amount of time.

LCD Screen

The LCD in our system is there to read our values coming from the humidity sensor. We used the LiquidCrystal Library to be able to communicate the data coming from the LCD to the arduino board. The library is then interacted with through its print functions which provide an interface very similar to the Serial library. We set a baud rate at the beginning of the code, and gave it the number of rows and columns to print to. That being 16 rows and 2 columns. In the hardware aspect, a potentiometer is used to set the contrast of the LCD. Beyond that a set of parallel communication wires connect the LCD to the arduino. It has a bit of internal memory and will store any data we send it, and has some built in utilities for operations such as clearing the display or changing where new data should be written. The information displayed on the LCD will come in two parts, the first line will specify what state the device is currently in. The second line will display information relevant to the state such as temperature, humidity, or water level.

Real Time Clock Module

The RTC(Real Time Clock) helps us understand at what time our transitions occur in the system. By transitions I mean going from Idle to Disabled, Running to Error, etc. The clock must be manually fed the timestamp after the program is compiled, or as in our test, it will automatically run itself. Our test consisted of just setting the date and time, and having the clock count up from that time in the serial monitor. The implementation wasn't too difficult on this component, as we were just reading input from the buttons to see when states were changed.

Fan

The fan module was pretty simplistic. In our code we designated it a pin, connected to our live wire, then connected it to ground. It only ran on two states, that being if the temperature of our environment was less than our threshold temp, which was defined at the start of our program. Or our second state being if our system switched to a disabled state. The logic was pretty straightforward.

Button(s)

Start-

This was the first of our three buttons. Ou system initially started at a "disabled" state

Stop(Disable)-

Reset-

LED’s

An RGB LED is used to provide the state with a visual display. All of the lights are connected through 200 Ohm Resistors. The main loop of the program will not need to be bogged down due to lighting-related pulse width modulation, but there was a previous issue of the lights taking a lot of power from the board. An RGB LED with hardware brightness control allows the Arduino to create the four needed colors using only three output ports without any software overhead.

Potentiometer