Group: Andrew Debolt, Jenavieve Layosa, Melody Dorsey
CPE 301.1001
Shawn Ray
December 13, 2022

Final Project Overview

The project consists of designing and constructing an evaporating cooler system using an Arduino Mega 2560. The build consists of the DHT11 temperature and humidity sensor, water level sensor, four push buttons, liquid crystal display (LCD), fan motor using an additional power supply and L293D IC, and four LEDs consisting of green, red, yellow, and blue. The cooler works by monitoring the water levels in a cup, followed by printing an alert if the water level becomes too low. It also monitors the current air temp and humidity, displaying the results on the LCD screen. The system will start and stop the fan motor, and enable or disable the system using an on/off button. Lastly, it records the time and date when the motor is turned on or off. The system constraints included limits of the switch buttons for the vent.

The temperature and humidity sensor (DHT11) employs a capacitive humidity sensor and a thermistor to measure temperature. It then records the results and sends a digital signal to the data pin. The sensor is defined using **DHT_PIN 7** and **DHT_TYPE DHT11.** As the system reads the temperature and humidity from the DHT11 sensor, it records the values using the **DHT.h** library and the **readTemperature()** and **readHumidity()** functions. The temperature reading changes states from idle to running and vice versa.

The water level sensor acts as a monitor for the amount of water it detects. To change states from idle/running to error and error to idle/running, the sensor must identify a surrounding liquid. By defining **WATER_LEVEL** to its designated pin, the system checks the water level using the **adc_read()** function. If the value falls below the **WATER_THRESHOLD**, it returns the current state to error.

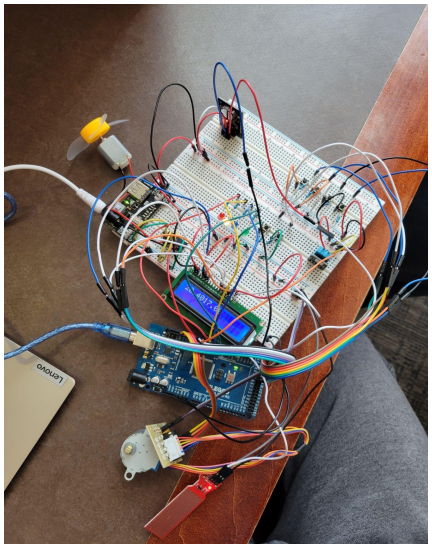
We implemented four push buttons into our design, each tasked with its own designated intention. The button defined as **BUTTON_ON_OFF** allows the system to switch between the disabled and enabled state. Since the system is idle by default, we added a second button defined as **BUTTON_RESET** to reset the build back to its idle state. For the step motor, we included 2 more push buttons to control the direction of the vent. These directions, **BUTTON_STEPPER_UP** and **BUTTON_STEPPER_DOWN** ultimately required limitations for the switch button setting.

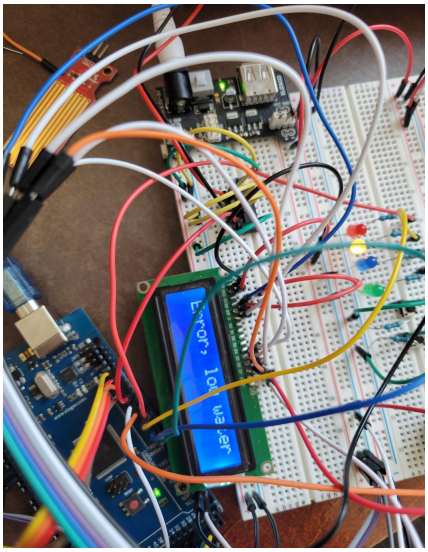
The liquid crystal display shows the current air temp and humidity recorded by the temperature and humidity sensor. In conjunction with the DS1307 module to ensure accuracy,

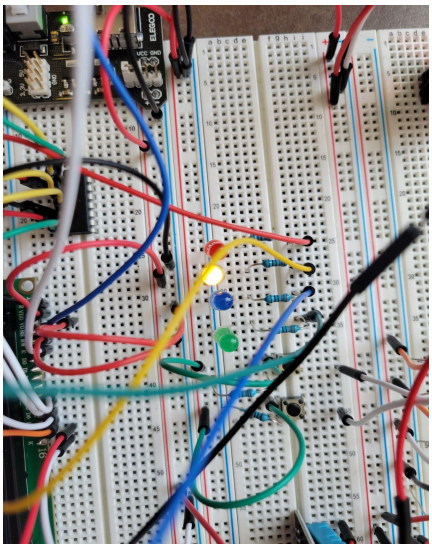
our results exhibited a temperature of 20℃. The LCD also works with the water level sensor to project an error message **“Error: low water”** when put into an error state by low water levels. The display and its pins are defined as **lcd** within the **LiquidCrystal.h** library. The **RTClib.h** is used to set the **DateTime()** function for the projected results.

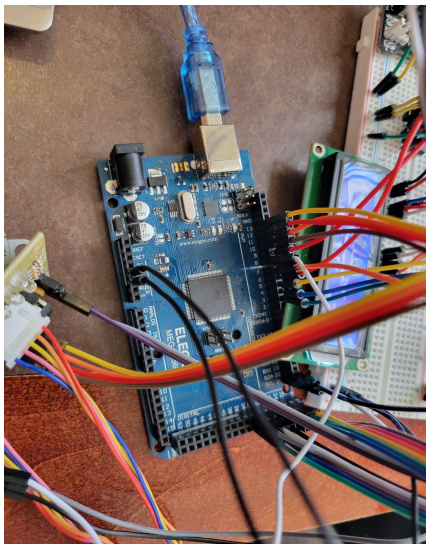
The fan motor works with the L293D IC chip to run only when the system is enabled through appropriate water levels and temperature values. The motor itself is defined by **MOTOR_PIN**, setting the motor between its disabled, idle, and running states depending on the conditions. The fan’s rate of speed is controlled using the **setFanMotor()** function when in its enabled state. By following the L293D pinout sheet and using the direction control pins we could control whether the motor rotates forward or backward. Furthermore, the motor is supported using the 545043 power supply to provide an additional direct power source.

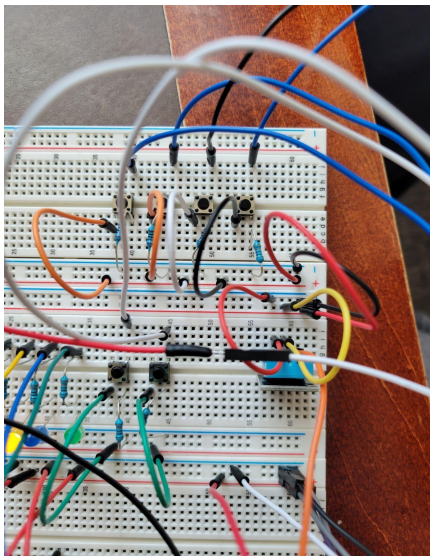
The build contains four different colored LEDs, each representing a different state of the project’s components. The green LED is lit when the circuit is in idle. The yellow LED indicates the system is disabled. The blue LED demonstrates the necessary conditions have been met and the system is running. The red LED designates the opposite, meaning the idle state has been interrupted and an error has occurred. Each light is defined by its individual color and delegated to its separate pin. The LED is given power through the function **turnLEDOn()**.

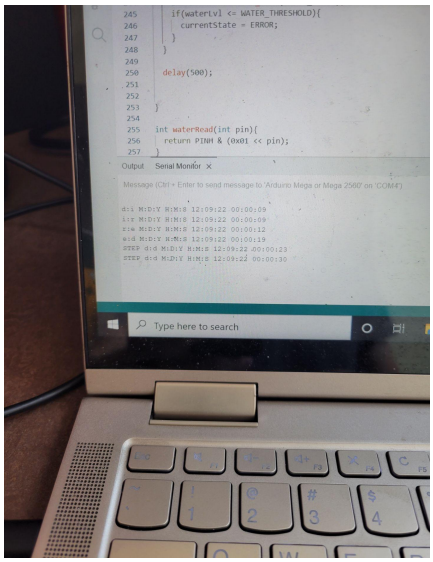



Error, low water

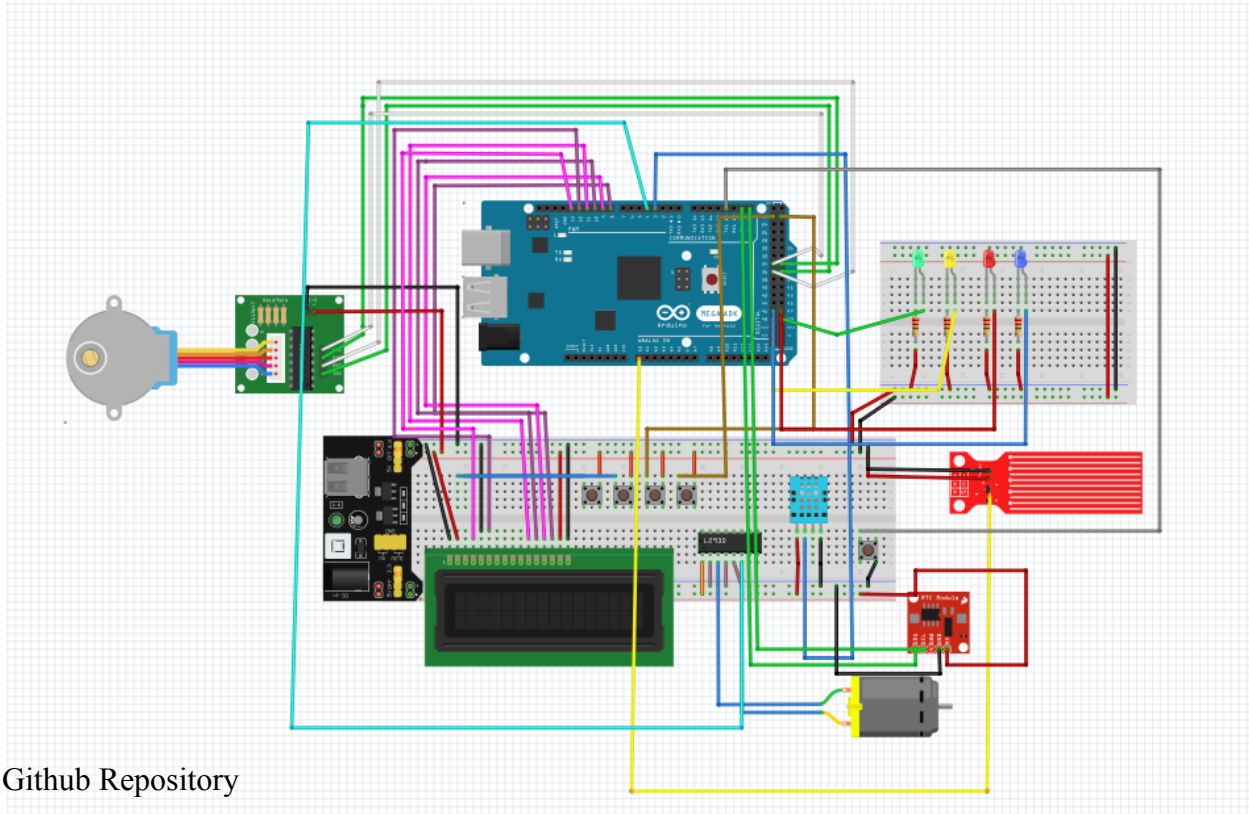







if(waterLvl <= WATER_THRESHOLD){
currentState = ERROR;
}
}
delay(500);
}
int waterRead(int pin){
return PINH & (0x01 << pin);
}
Output
Serial Monitor
Message (Ctrl + Enter to send message to 'Arduino Mega or Mega 2560' on 'COM4')
Type here to search

Schematic



Github Repository

https://github.com/UNR-Teaching/Final-Project-CPE-301-A-J-M

Videos

https://drive.google.com/drive/folders/11mdKAyN36zugl75LQbT4EKbC6chD9-Ia?usp=sharing