Jax Hendrickson
CPE 301.1001
May 12th, 2024

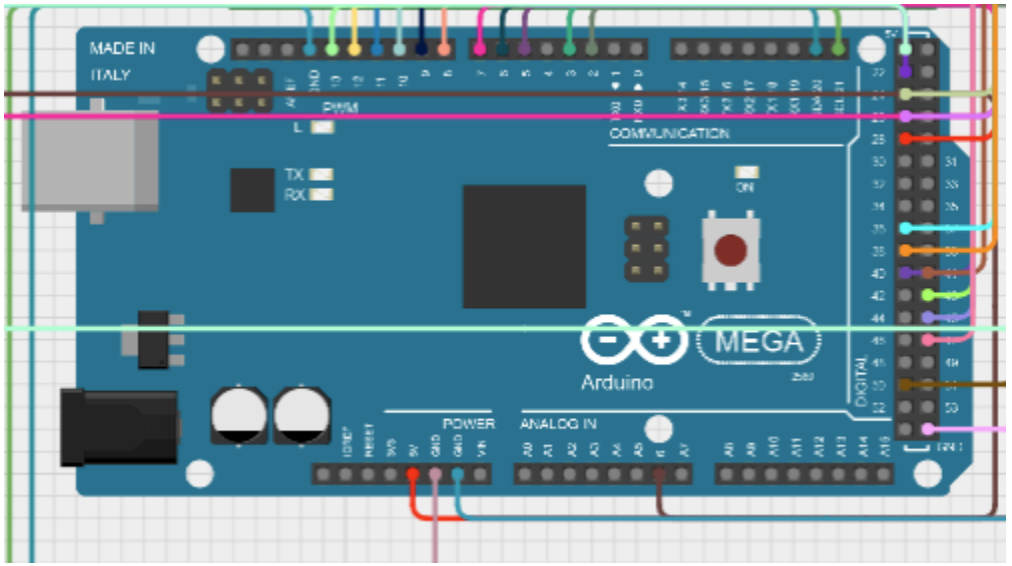
Final Project Report - Swamp Cooler

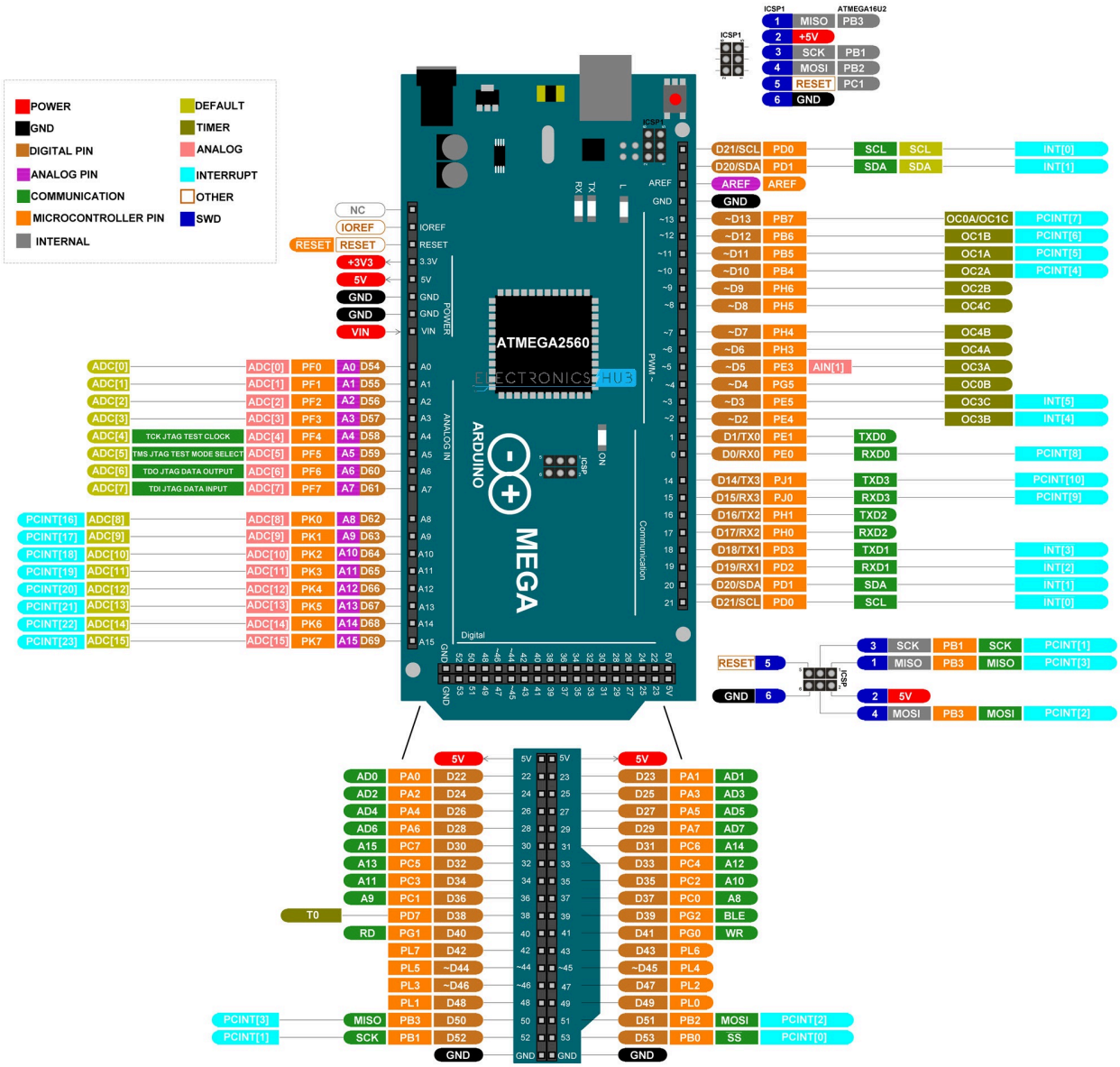
1. Components & Setup

For this project, the goal was to create a working Swamp Cooler out of parts in an arduino kit. The general idea is, the cooler would start off as disabled, until turned on. While it was on, and it had a source of water, it would monitor the air around it and, when detecting above a certain temperature or humidity, would evaporate the air in the cooler to cool down the air around it. While we didn't implement this much detail; lacking a shell and an actual means of evaporation, we have the basic premise of it down. The overall idea of it is similar to a humidifier, if you've never heard the term before. I'll now go into each of the individual components of the cooler, then into the states, and finally finish off with the full schematics of my circuit.

- **Arduino ATMEGA 2560**

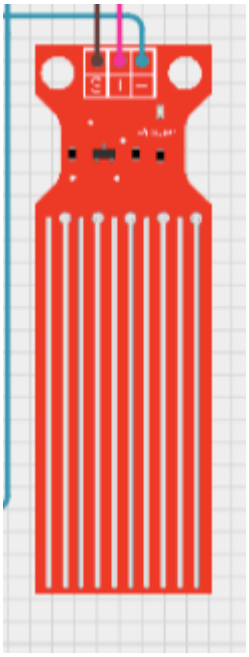
As shown below, the Arduino ATMEGA 2560 is the microcontroller we used for this project. The layout of connections I had coming from it is below; powered by a USB power cable. The pinout diagram is also available, showing the individual registers for each pin. This component had by far the most connections, and was also by far the most important component; since it's the component that actually runs the code through the Arduino IDE. Since this is the most integral piece to our circuit, I've also included my own connections in the picture below, to show which pins I've used. The arduino utilizes the following libraries as well: <LiquidCrystal.h> for the LCD, <Stepper.h> for the stepper motor, <RTClib> for communication with the RTC and to initialize the internal clock in it, and <dht.h> to provide the functionality for the DHT11 humidity sensor, to allow us to assign temperature and humidity to it, alongside reading data from it.

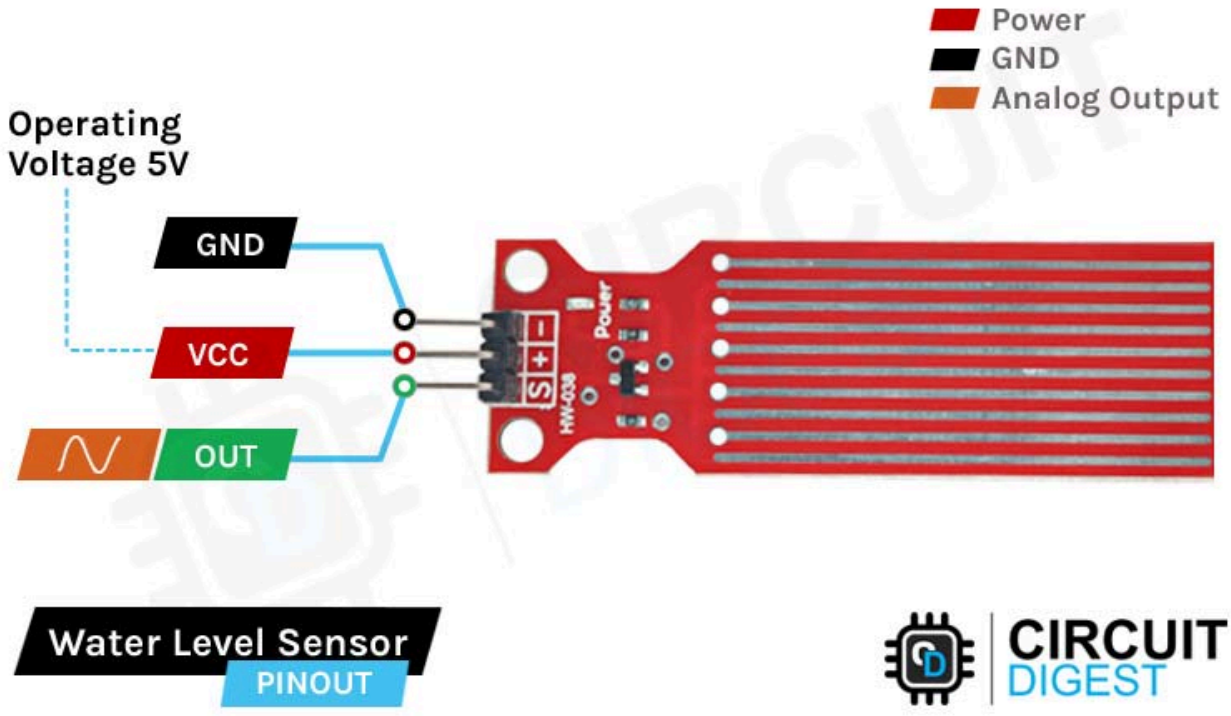

MADE IN
ITALY
PWM
L
TX
RX
COMMUNICATION
ON
MEGA
Arduino
POWER
ANALOG IN
DIGITAL


POWER
GND
DIGITAL PIN
ANALOG PIN
COMMUNICATION
MICROCONTROLLER PIN
INTERNAL
DEFAULT
TIMER
ANALOG
INTERRUPT
OTHER
SWD
ATMEGA2560
ARDUINO
MEGA
ELECTRONICS
HU3
ICSP1
ATMEGA16U2
MISO PB3
+5V
SCK PB1
MOSI PB2
RESET PC1
GND
NC
IOREF
RESET
+3V3
5V
GND
GND
VIN
POWER
ANALOG IN
Digital
PWM ~
Communication
TX
RX
L
ON
ICSP
AREF
D21/SCL PD0
D20/SDA PD1
~D13 PB7
~D12 PB6
~D11 PB5
~D10 PB4
~D9 PH6
~D8 PH5
~D7 PH4
~D6 PH3
~D5 PE3
~D4 PG5
~D3 PE5
~D2 PE4
D1/TX0 PE1
D0/RX0 PE0
D14/TX3 PJ1
D15/RX3 PJ0
D16/TX2 PH1
D17/RX2 PH0
D18/TX1 PD3
D19/RX1 PD2
D20/SDA PD1
D21/SCL PD0
SCL
SDA
TXD0
RXD0
TXD3
RXD3
TXD2
RXD2
TXD1
RXD1
SDA
SCL
AIN[1]
OC0A/OC1C
OC1B
OC1A
OC2A
OC2B
OC4C
OC4B
OC4A
OC3A
OC0B
OC3C
OC3B
INT[0]
INT[1]
PCINT[7]
PCINT[6]
PCINT[5]
PCINT[4]
INT[5]
INT[4]
PCINT[8]
PCINT[10]
PCINT[9]
INT[3]
INT[2]
INT[1]
INT[0]
ADC[0] PF0 A0 D54
ADC[1] PF1 A1 D55
ADC[2] PF2 A2 D56
ADC[3] PF3 A3 D57
ADC[4] TCK JTAG TEST CLOCK PF4 A4 D58
ADC[5] TMS JTAG TEST MODE SELECT PF5 A5 D59
ADC[6] TDO JTAG DATA OUTPUT PF6 A6 D60
ADC[7] TDI JTAG DATA INPUT PF7 A7 D61
PCINT[16] ADC[8] PK0 A8 D62
PCINT[17] ADC[9] PK1 A9 D63
PCINT[18] ADC[10] PK2 A10 D64
PCINT[19] ADC[11] PK3 A11 D65
PCINT[20] ADC[12] PK4 A12 D66
PCINT[21] ADC[13] PK5 A13 D67
PCINT[22] ADC[14] PK6 A14 D68
PCINT[23] ADC[15] PK7 A15 D69
RESET 5
GND 6
3 SCK PB1 SCK PCINT[1]
1 MISO PB3 MISO PCINT[3]
2 5V
4 MOSI PB3 MOSI PCINT[2]
ICSP2
5V
AD0 PA0 D22
AD2 PA2 D24
AD4 PA4 D26
AD6 PA6 D28
A15 PC7 D30
A13 PC5 D32
A11 PC3 D34
A9 PC1 D36
T0 PD7 D38
RD PG1 D40
PL7 D42
PL5 ~D44
PL3 ~D46
PL1 D48
PCINT[3] MISO PB3 D50
PCINT[1] SCK PB1 D52
GND
D23 PA1 AD1
D25 PA3 AD3
D27 PA5 AD5
D29 PA7 AD7
D31 PC6 A14
D33 PC4 A12
D35 PC2 A10
D37 PC0 A8
D39 PG2 BLE
D41 PG0 WR
D43 PL6
~D45 PL4
D47 PL2
D49 PL0
D51 PB2 MOSI PCINT[2]
D53 PB0 SS PCINT[0]

- **Water Level Detection Sensor Module**

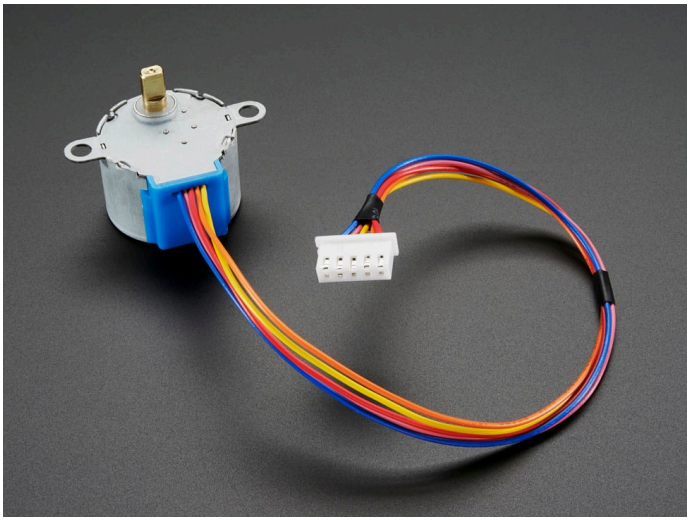
We used the Water Level Detection Sensor Module to measure the amount of water available for the swamp cooler to evaporate. We did this by taking the analog measurement of the sensor, transforming it into a digital measurement, and then judging that against a threshold that we set; if the water level was higher than the threshold, then the cooler had enough water to run. The sensor was also connected to a pin on the arduino that acted as a power pin, telling it when to turn on and off.





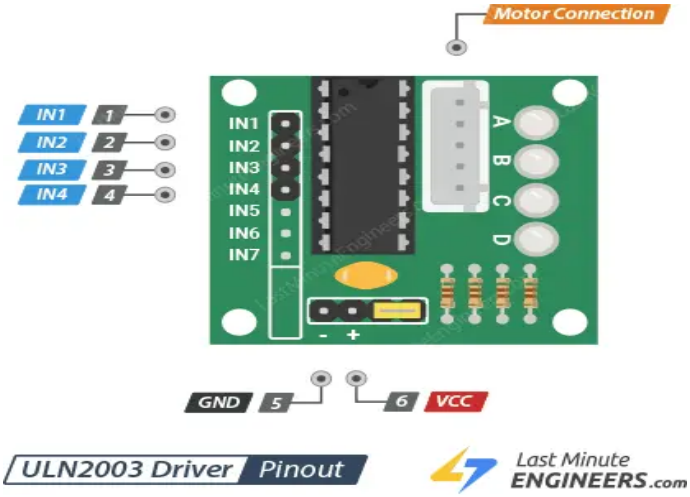
- **Stepper Motor**

The Stepper Motor was one of the simpler parts to integrate into the circuit. The stepper motor acted as a substitute for the rotating vent that would be present on a swamp cooler. It has five wires, all connected, that plug into the ULN2003 Stepper Motor Driver Module. When either of the motor buttons on the breadboard are pressed, the motor moves clockwise or counterclockwise a certain distance, which is pre-chosen by the code.



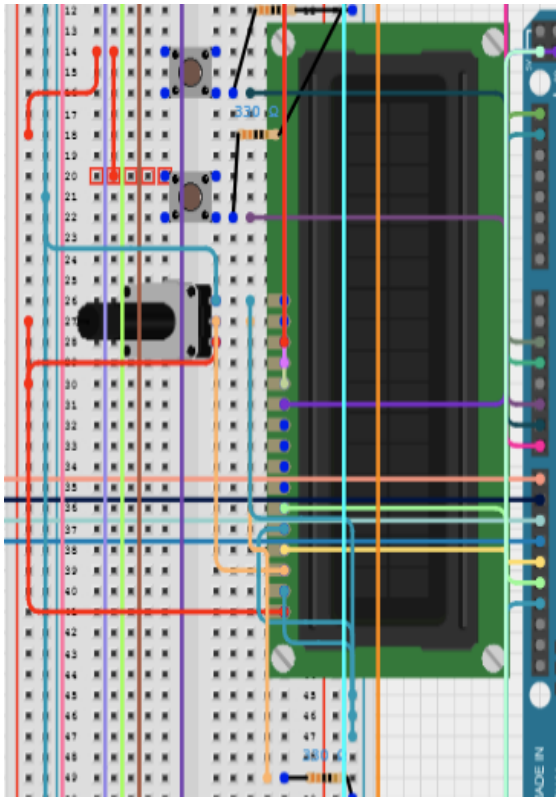
- **ULN2003 Stepper Motor Driver Module**

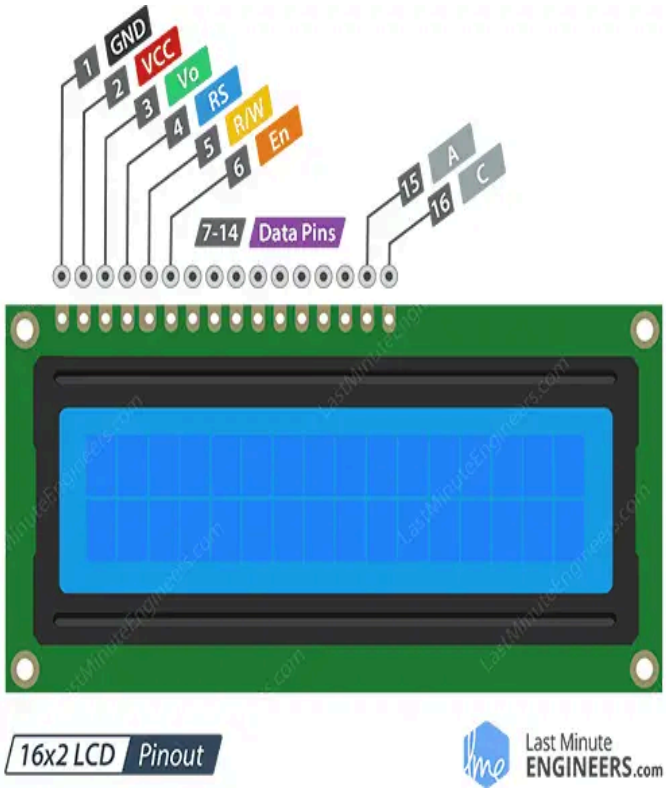
The ULN2003 Stepper Motor Driver Module is directly connected to the Stepper Motor via the wires you see above. The job of the stepper motor is to house all the logic needed to move the motor clockwise and counterclockwise. When powered, two of the lights will be active. When in use, all of the lights should be active. The motor has four input cables that are connected to the digital outputs of the arduino, and power and ground cables. The pins connected to the arduino are also connected to two buttons; the left of which, when pressed, rotates the motor counterclockwise. The right button rotates the motor clockwise. Both buttons are also connected to ground through 330 Ohm resistors. We also use the <Stepper.h> library to use the full functionality of this module.



- **LCD1602 Module (With Pin Header)**

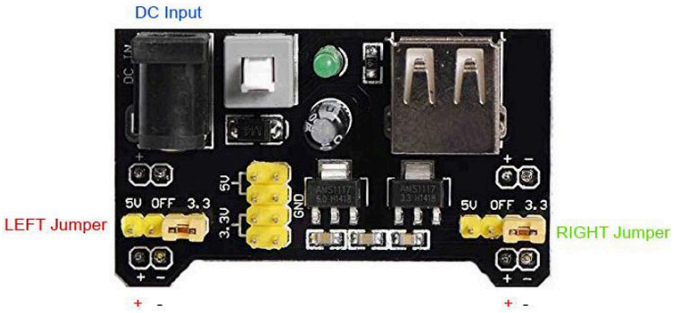
The LCD, or Liquid Crystal Display, was connected on the main breadboard, and was the output device for any errors that occurred with the swamp cooler, along with showing the user the current temperature and humidity of the area. It's a 16 by 2 display, 16 columns and 2 rows. It's connected to a potentiometer which is used to increase or lower the voltage going to the display, which in-turn makes the LCD brighter or dimmer. It's also connected to a 330 resistor to ground, along with two buttons, connected to digital outputs of the arduino, which are then connected back to the LCD. If the leftmost button is pressed, the display turns on and displays either the current temperature and humidity detected, or, if the water sensor detects no water, a "No Water" error message. As this component has a lot of connections, I'll be again showing my schematic up-close below, along with a pinout diagram. This part is meant to mimic the main display of the cooler. We use the <LiquidCrystal.h> library to display, clear, and change the display of the LCD.





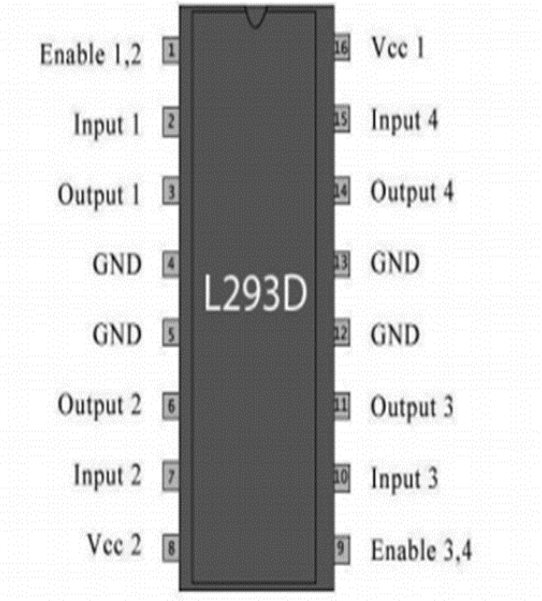
- **Power Supply Module**

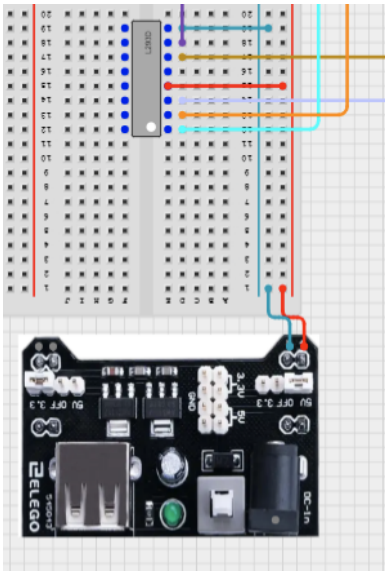
The purpose of the power supply module is to power the DC 3-6V motor. It's connected to power through a 9 Volt 1 Amp adapter, and plugged into a breadboard, supplying power to the motor and its microcontroller. We use this separately since it's dangerous to connect the arduino power to the DC motor; as it may damage either or both of them.



- **L293D Motor Driver**

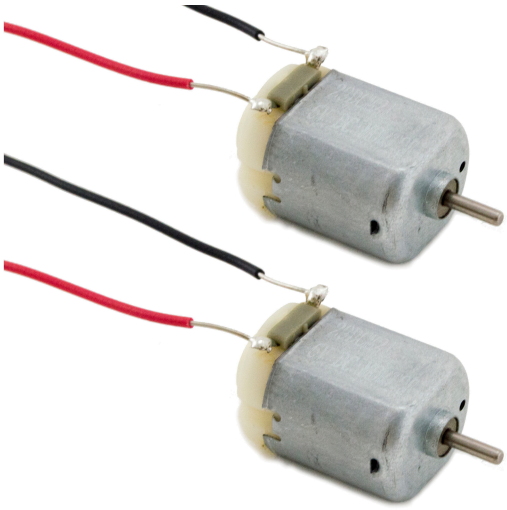
The L293D is a microcontroller that houses the logic necessary to drive the 3-6V DC motor that it's connected to. Since all of the logic is contained within the controller, no excess code is necessary. It's a 16 pin IC with 2 inputs, 2 outputs, and one enable pin per motor. By enabling the motor, which is controlled by the arduino's digital pins, the controller supplies the power given by the Power Supply Module to the Motor, making it run until the arduino cuts it off. My schematics for it, as well as the pinout diagram, are available below.





- **3-6V Motor and Fan Blade**

This component is a 3-6V output DC fan motor, connected to the previously detailed L293D Motor Driver, and the Power Supply Module. The motor has no logic of its own, just a positive and ground wire; both connected to pins of the Motor Driver. It's used to rotate the plastic fan on the end of it, which is supposed to blow the evaporated, cool air into the room through the cooler's vents.

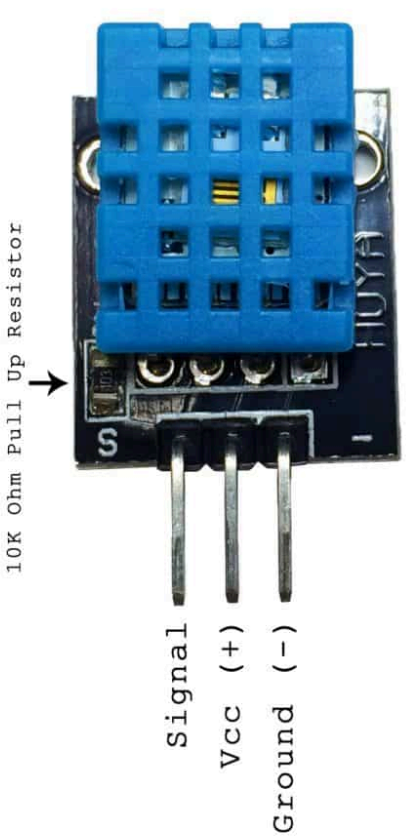


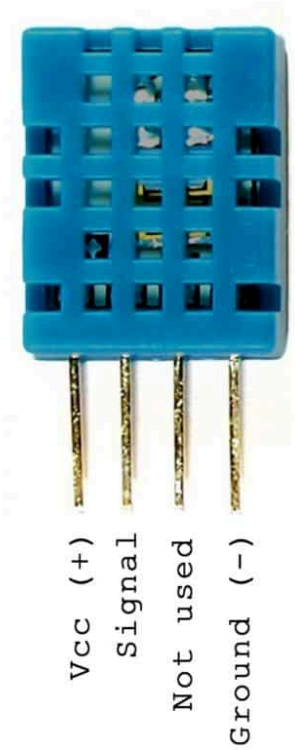
- **DS1307 RTC Module**

The RTC Module, or Real Time Clock module, is our swamp cooler's means of tracking time. The DS1307 has five pins, two being power and ground, and the other two used being SDA (Serial Data Line) and SCL (Serial Clock Line), which both have specific communication pins on the arduino. There's also a fifth pin called SQW, which produces a square wave, but we won't be using it. There's also a small battery connected internally to the RTC, which prevents it from losing time if the main power is unplugged. The clock is used to keep track of time, for two reasons. The first is to, while the cooler is idle or running, notify the user periodically once per minute the current temperature and humidity of the DHT11 sensor. The other use is to notify the user of the exact time that certain actions happen; meaning any state change, or vent rotation is logged on the serial monitor. To fully make use of the RTC module, we also used the RTClib.h library, to help initialize the clock and allow it to keep time accurately. We were also allowed to use the 74HC595 IC (It wasn't explicitly said in the instructions and offers little to no real impact on the circuit), but I didn't find any need for it.

- **DHT11 Temperature and Humidity Module**

The DHT11 Temperature and Humidity Module ties together the rest of the circuit. The DHT11 has three pins, one for ground, one for power, and a pin going into a digital pin on the arduino. With the help of the <dht.h> library, we can monitor the humidity and temperature levels of the air surrounding the sensor, and transmit that to both the serial monitor to be judged, along with the LCD to be constantly displayed. The humidity and temperature are only displayed once per minute on the serial monitor while the swamp cooler is in use, but are constantly being monitored and compared against set thresholds. If the temperature goes above 25 degrees celsius, or if the humidity goes above 75%, the fan motor is signaled to begin. This module offers most of the functionality of the circuit, as without it, the atmosphere would never be monitored, and thus none of the other components would be used.



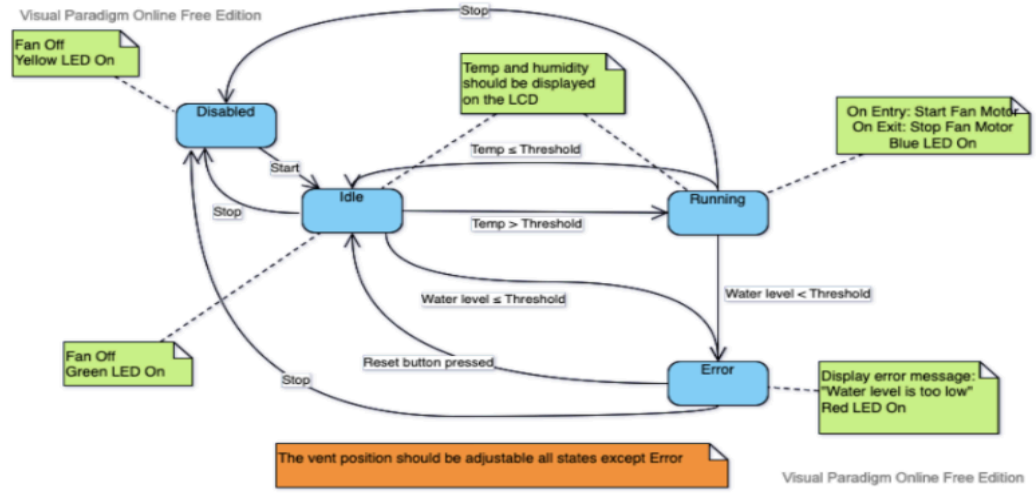


- **Miscellaneous Components**

Along with the aforementioned components, there are a handful of other pieces that were used in the creation of this circuit. These include: four push buttons (Two to control the vent motor's direction, two to control the LCD's power), five 330 Ohm resistors (One per button, and one for the LCD leading to ground), a potentiometer to alter the voltage given to the LCD, a 9 Volt 1 Amp wall adapter to power the power supply module, a USB cable to connect my PC to the arduino, two breadboards (one large and one small), and four LEDs: One Green, one blue, one red, and one orange, to distinctly show which state the cooler is in.

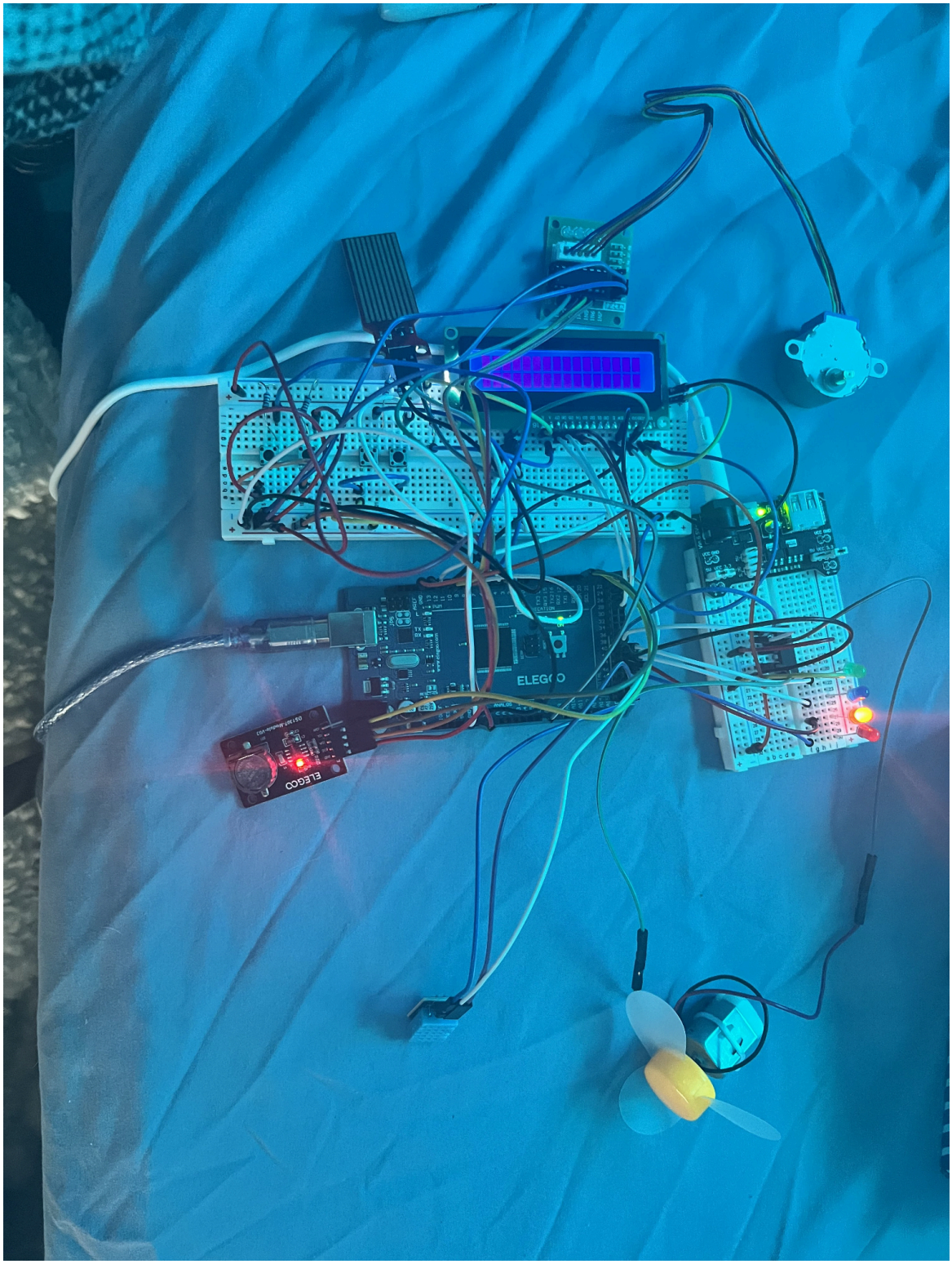
2. States

The Swamp Cooler should have four states. These states are indicated by the four LED lights just mentioned. Blue is for running, and is only active when the threshold for humidity or temperature has been met, and there is enough water detected by the water sensor to keep going. In this state, the vents can be turned, the LCD displays the current temperature and humidity, and the serial monitor displays the current values every minute. When the cooler enters this state, the serial monitor displays the message "The Swamp Cooler is Running!", alongside the time that it started. If water runs too low, it changes to the error state. And if the off button is hit, it turns off. Green is for idle, which is the state that supports running. Idle means that there is water in the tank being detected by the water level sensor, but the temperature and humidity of the surrounding area aren't high enough to break the set threshold. During this phase, like in the running phase, the LCD displays the temp and humidity values, and the serial monitor logs each value once per minute. You can also move the motor clockwise or counterclockwise like in running, and it will be displayed on the serial monitor. If water runs too low, it changes to the error state. Also, if the off button is hit, it will turn off. The one-minute timer is also shared between idle, error, and running; since the program just checks for the current time, and the previously logged time. Orange is for the Disabled state, which is the default state of the cooler, until turned on. While disabled, the cooler's vents can be moved, but will not be logged on the serial monitor. The temperature and humidity will also be off, and the LCD will be inactive. The circuit stays off until the on button is hit, and the program transfers to any other state. If the off button is hit at any time, the state will always go to disabled. And finally, red is for the Error state. The error state occurs as long as the on button has been hit, and the water level sensor is returning a value under the threshold. While the error state is on, no other actions can occur. The LCD displays "Error: Low Water Level", the motors to turn the fan won't work, humidity and temperature are measured, but are not displayed onto the serial monitor. The instructions for this section were pretty vague, if I'm being honest, so I'm sure there were some minor mistakes in how I interpreted the necessary properties of each state. For example, in the states diagram on the rubric, it says that "The vent position should be adjustable in all states except error", and below it in the "all states except DISABLED" section, it says "System should respond to changes in vent position control". Regardless, this is the diagram that shows when which state is active:



3. Schematics and Photo

The physical circuit:



Circuit Schematics: