Ryan Ransom
05/12/2024

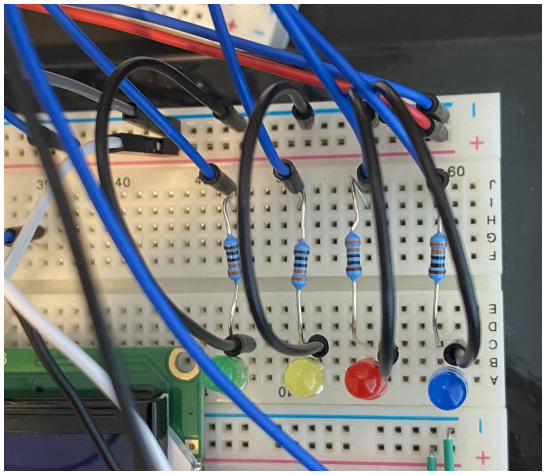
# Final Project

Link to circuit video:
https://youtu.be/yExeMYU8CE4

Link to github repo:
https://github.com/Ryan-Ransom/cpe301_finalprojcet

Description: For this project I wired a circuit and coded the software for a functioning swamp cooler. Swamp coolers work by using a fan to blow water over a water reservoir creating cool air. To achieve this I used the Arduino Mega 2560, LCD screen, 10k Potentiometer, 330Ω resistors, 1000Ω resistors, LEDs, buttons, Water Level Detection Sensor, DHT11 Temperature and Humidity Module, DS1307 RTC Module, Power Supply Module with 9V connection, L293D, 3-6V Motor, ULN2003 Stepper Motor Driver, and Stepper Motor. Using these components I coded the program for the cooler using four states Disabled, Idle, Running, and Error.

Components:
- Four LEDs connected to digital pins 2, 3, 4, and 5. Each a different color to signify what state the cooler is in yellow for disabled, green for idle, blue for running, and red for error. Each led is connected to power using a 330Ω resistor to step the voltage down from 5V

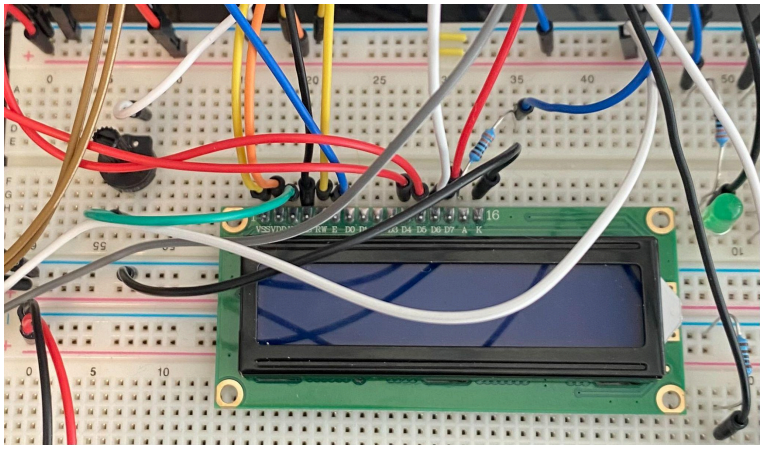


```
// logic for controlling led power
void setLed() {
  // disable all leds
  PORTE &= ~(0x01 << BLUE_LED);
  PORTE &= ~(0x01 << RED_LED);
  PORTG &= ~(0x01 << YELLOW_LED);
  PORTE &= ~(0x01 << GREEN_LED);

  switch (ledSelect) {  // enable led based on value of ledSelect
    case 2:
      PORTE |= (0x01 << BLUE_LED);
      break;
    case 3:
      PORTE |= (0x01 << RED_LED);
      break;
    case 4:
      PORTG |= (0x01 << YELLOW_LED);
      break;
    case 5:
      PORTE |= (0x01 << GREEN_LED);
      break;
  }
}
```

```
// led
#define BLUE_LED 4  // pin 2
#define RED_LED 5 // pin 3
#define YELLOW_LED 5  // pin 4
#define GREEN_LED 3 // pin 5
```

-LCD and 10k Potentiometer: LCD connected to digital pins 52, 50, 48, 46, 44, and 42 using a 10k potentiometer to control the brightness displays and a 330Ω resistor to step down the voltage from 5V. Displays the current temperature and humidity when in the Idle and Running states, and displays a “Water level low” message when in the Error state.



```
// lcd
LiquidCrystal lcd(52, 50, 48, 46, 44, 42);  // pins 52, 50, 48, 46, 44, 42
```

```
// displays error message on lcd screen
void errorMsg() {
  lcd.setCursor(0, 0);
  lcd.print("Water Level is");
  lcd.setCursor(0, 1);
  lcd.print("too low");
}
```

```
// displays temperature and humidity on lcd screen
void displayTemp() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(DHT.temperature);
  lcd.print((char)223);
  lcd.print("C");
  lcd.setCursor(0, 1);
  lcd.print("Humidity: ");
  lcd.print(DHT.temperature);
  lcd.print("%");
}
```

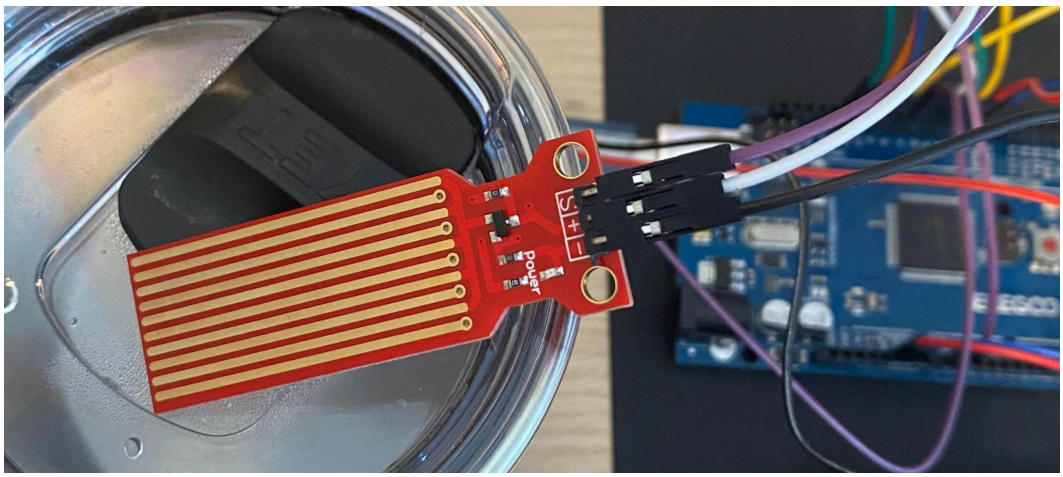
-Water Level Detection Sensor Module is connected to digital pin 6, and analog pin A0. This component provides an analog input providing us with the current water level in the water reservoir. This value is used to determine when the state of the cooler should switch from Idle or running to error based on the variable minimumWaterLevel.

```
// water level
#define WATER_POWER 3  // pin 6
#define WATER_SIGNAL 0  // pin A0
unsigned int waterLevel;  // value holds current water level
```

```
  waterMonitorOn(); // turn on water level monitoring
  if (minimumWaterLevel > waterLevel) { // if water level is too low change to error state and display state change
    previousState = currentState;
    currentState = 3;
    displayChange();
    error = true;
    lcd.clear();
    running = false;
    controlFan(false);
  }
```

```c
// turns on water monitoring
void waterMonitorOn() {
  PORTH |= (0x01 << WATER_POWER);
  my_delay(10);
  waterLevel = adc_read(WATER_SIGNAL);
  waterMonitorOff();  // disable water monitoring
}

// turns off the water level monitor
void waterMonitorOff() {
  PORTH &= ~(0x01 << WATER_POWER);
}
```

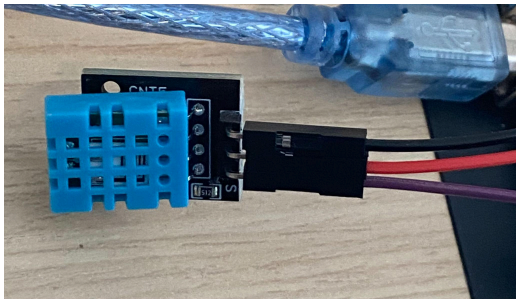



-DHT11 Temperature and Humidity Module is connected to digital pin 7. This component provides the temperature and humidity readings which determines if the state should be Idle or Running based on the minimumTemp variable, and is displayed on the LCD screen.

```
// temp and humidity
dht DHT;
#define DHT11_PIN 7 // pin 7
```

```
if (minimumTemp >= currentTemp) { // if temperature is under threshold change to idle state and display state change
  previousState = currentState;
  currentState = 1;
  displayChange();
  idle = true;
  running = false;
  controlFan(false);
}
```

```
// monitor for temperature and humiditys
void monitorTempHumidity() {
  int chk = DHT.read11(DHT11_PIN);
  currentTemp = DHT.temperature;
  currentHumidity = DHT.humidity;
}
```



-DS1307 RTC Module is connected to digital pins DSA 20 and SCL 21. This component is used as a clock to provide us with accurate readings of date and time used to display state changed in the serial monitor.

```
// displays state changes on serial monitor
void displayChange() {
  rtc.refresh();  // refresh rtc values

  // save rtc values
  int year = rtc.year();
  int month = rtc.month();
  int day = rtc.day();
  int hour = rtc.hour();
  int min = rtc.minute();
  int sec = rtc.second();

  // arrays for putChar calls
  char num[10] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };
  char disabledState[8] = { 'D', 'I', 'S', 'A', 'B', 'L', 'E', 'D' };
  char idleState[4] = { 'I', 'D', 'L', 'E' };
  char runningState[7]{ 'R', 'U', 'N', 'N', 'I', 'N', 'G' };
  char errorState[5] = { 'E', 'R', 'R', 'O', 'R' };

  // display from
  putChar('F');
  putChar('r');
  putChar('o');
  putChar('m');
  putChar(' ');
```

```
// display previous state
switch (previousState) {
  case 0:
    for (int i = 0; i < 8; i++) {
      putChar(disabledState[i]);
    }
    break;
  case 1:
    for (int i = 0; i < 4; i++) {
      putChar(idleState[i]);
    }
    break;
  case 2:
    for (int i = 0; i < 7; i++) {
      putChar(runningState[i]);
    }
    break;
  case 3:
    for (int i = 0; i < 5; i++) {
      putChar(errorState[i]);
    }
    break;
}

// display to
putChar(' ');
putChar('t');
putChar('o');
putChar(' ');
```

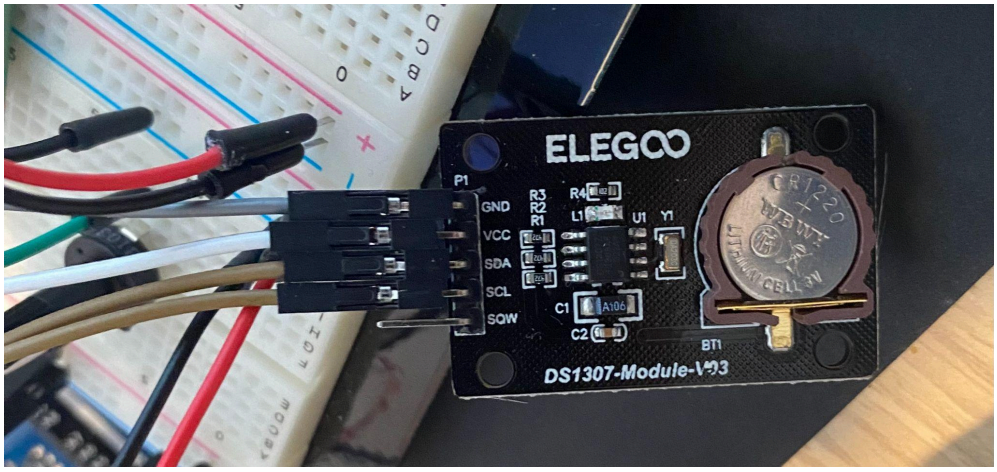
```
//  display current state
switch (currentState) {
  case 0:
    for (int i = 0; i < 8; i++) {
      putChar(disabledState[i]);
    }
    break;
  case 1:
    for (int i = 0; i < 4; i++) {
      putChar(idleState[i]);
    }
    break;
  case 2:
    for (int i = 0; i < 7; i++) {
      putChar(runningState[i]);
    }
    break;
  case 3:
    for (int i = 0; i < 5; i++) {
      putChar(errorState[i]);
    }
    break;
}

// display at
putChar(' ');
putChar('a');
putChar('t');
putChar(' ');
```

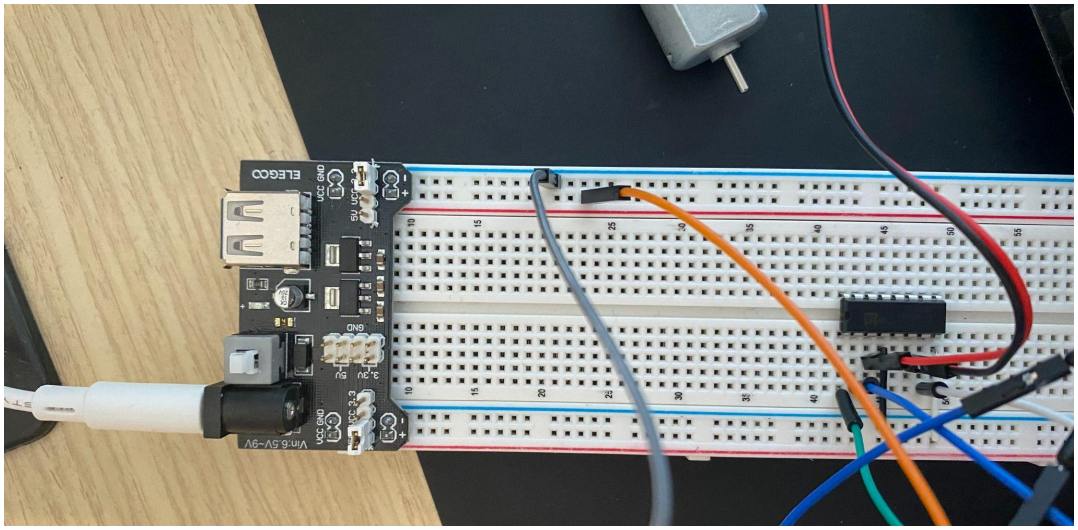
```
  // display date mm/dd/yy
  putChar(num[(month / 10) % 10]);
  putChar(num[month % 10]);
  putChar('/');
  putChar(num[(day / 10) % 10]);
  putChar(num[day % 10]);
  putChar('/');
  putChar(num[(year / 10) % 10]);
  putChar(num[year % 10]);
  putChar(' ');

  // display time hh:mm:ss
  putChar(num[(hour / 10) % 10]);
  putChar(num[hour % 10]);
  putChar(':');
  putChar(num[(min / 10) % 10]);
  putChar(num[min % 10]);
  putChar(':');
  putChar(num[(sec / 10) % 10]);
  putChar(num[sec % 10]);
  putChar('\n');

  previousState = -1;
}
```



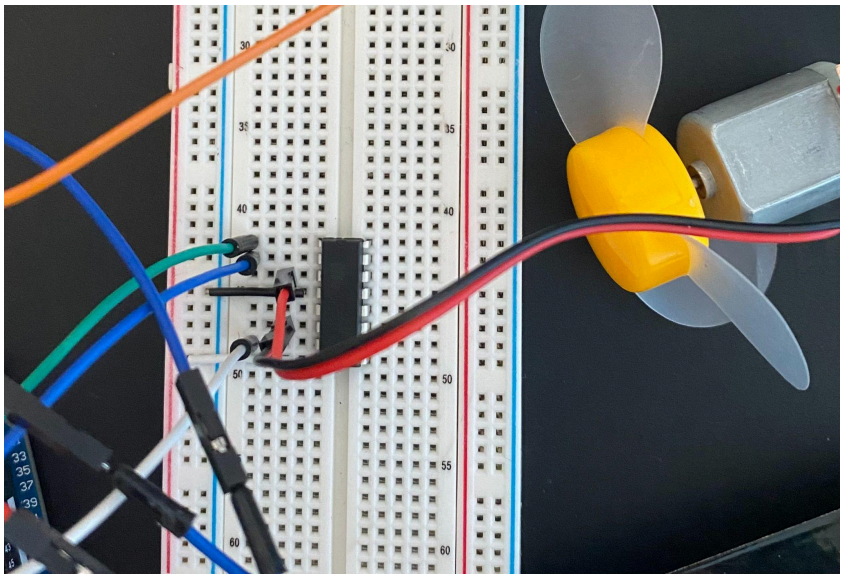

-9V Power Supply Module and Bread Board is a power supply connected to a 9V power source used to provide power for the 3-6V fan motor and stepper motor.



-L293D Chip and 3-6V Fan Motor: The L293D Chip is a motor control driver which is connected to the 9V power supply and digital pins 31, 33, and 35. This chip allows us to control the 3-6V fan motor which blows the cool air lowering the overall temperature. The motor is powered when the program is in the Running state meaning currentTemp is greater than the minimumTemp.

```
// fan motor
#define speedPin 6  // pin 31
#define dir1 4  // pin 33
#define dir2 2  // pin 35
```

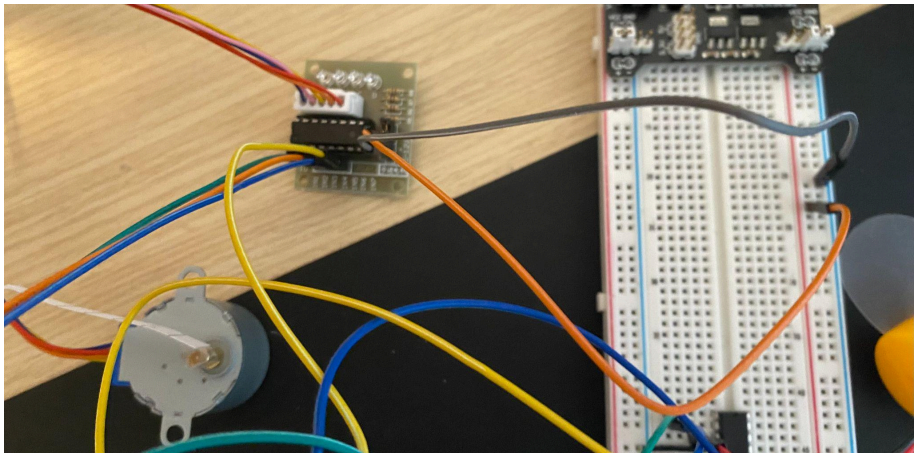
```
// controls if fan is enabled or disabled
void controlFan(bool power) {
  if (power) {
    PORTC |= (0x01 << dir1);
    PORTC &= ~(0x01 << dir2);
    analogWrite(31, 255);
  } else {
    PORTC &= ~(0x01 << dir1);
    PORTC &= ~(0x01 << dir2);
    analogWrite(31, 0);
  }
}
```

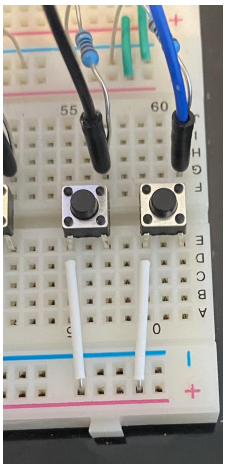


-ULN2003 Stepper Motor Driver Module and Stepper Motor with control buttons: The stepper motor is connected to the ULN2003 Stepper Motor Driver Module which is connected to the 9V power supply and digital pins 12, 11, 10, and 9. This motor controls the vent position while in the Idle, and Running state using two buttons connected to digital pins 45, and 49 and grounded using 1000Ω resistors, to spin the stepper motor either clockwise or counterclockwise.

```
// stepper motor
const int stepsPerRevolution = 200;
// Creates an instance of stepper class
// Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
Stepper myStepper = Stepper(stepsPerRevolution, 13, 11, 12, 10);  // pins 13, 11, 12, 10
#define ventCW 0  // pin 49
#define ventCCW 4 // pin 47
```

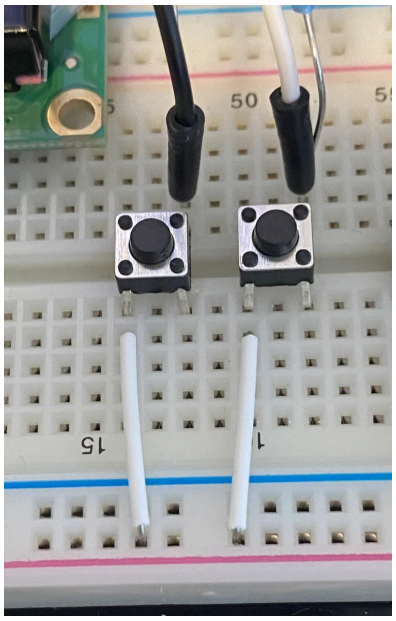
```
if (vent) { // if vent is true
  while (PINL & (0x01 << ventCW)) { // if clockwise button is pressed
    myStepper.step(1);  // move vent clockwise
    my_delay(10);
  }
  while (PINL & (0x01 << ventCCW)) {  // if counter clockwise button is pressed
    myStepper.step(-1); // move vent counter clockwise
    my_delay(10);
  }
}
```





-Start/Stop and Reset Buttons: The Start/Stop button is connected to digital pin 18 and is used alongside an interrupt to change the state of the program from Disabled to Idle or any state to Disabled. The reset button connected to digital pin 19 is used to change the state of the program from Error to Idle when the waterLevel is greater than the minimumWaterLevel.

```
// buttons
#define START_STOP 3   // pin 18
#define resetButton 2  // pin 19
```



```
// logic for when start/stop button is pressed
void startStop() {
  if (disabled) { // if disabled swap to idle state
    previousState = currentState;
    currentState = 1;
    idle = true;
    disabled = false;
    displayTemp();
  } else {  // switch to disabled if not already disabled
    previousState = currentState;
    currentState = 0;
    idle = false;
    disabled = true;
    running = false;
    error = false;
  }
}
```

```
// interupt
attachInterrupt(digitalPinToInterrupt(18), startStop, FALLING);
```

States:

-Disabled:

- -yellow led is powered
- -lcd display is off
- -vent controls are off
- -water level monitoring is off
- -fan is off
- -swap to idle if start button is pressed

-Idle:

- -green led is powered
- -temperature and humidity are displayed on lcd
- -vent controls are on
- -water level monitoring is on
- -fan is off
- -swap to error if water level is too low
- -swap to running if temperature is too high
- -swap to disabled if stop button is pressed

-Running:

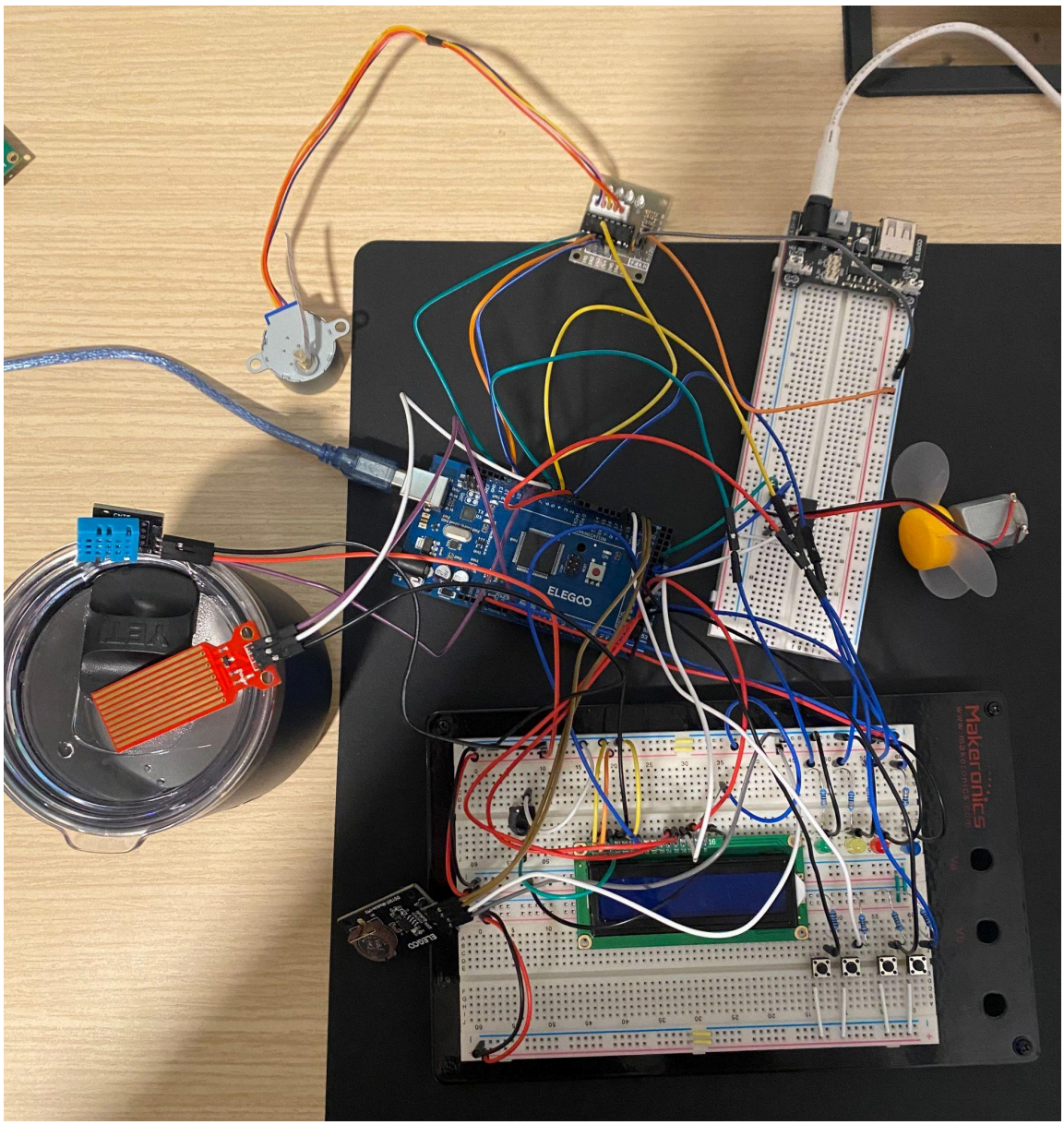
- -blue led is powered
- -temperature and humidity are displayed on lcd
- -vent controls are on
- -water level monitoring is on
- -fan is on

-swap to error if water level is too low
-swap to idle if temperature is below threshold
-swap to disabled if stop button is pressed

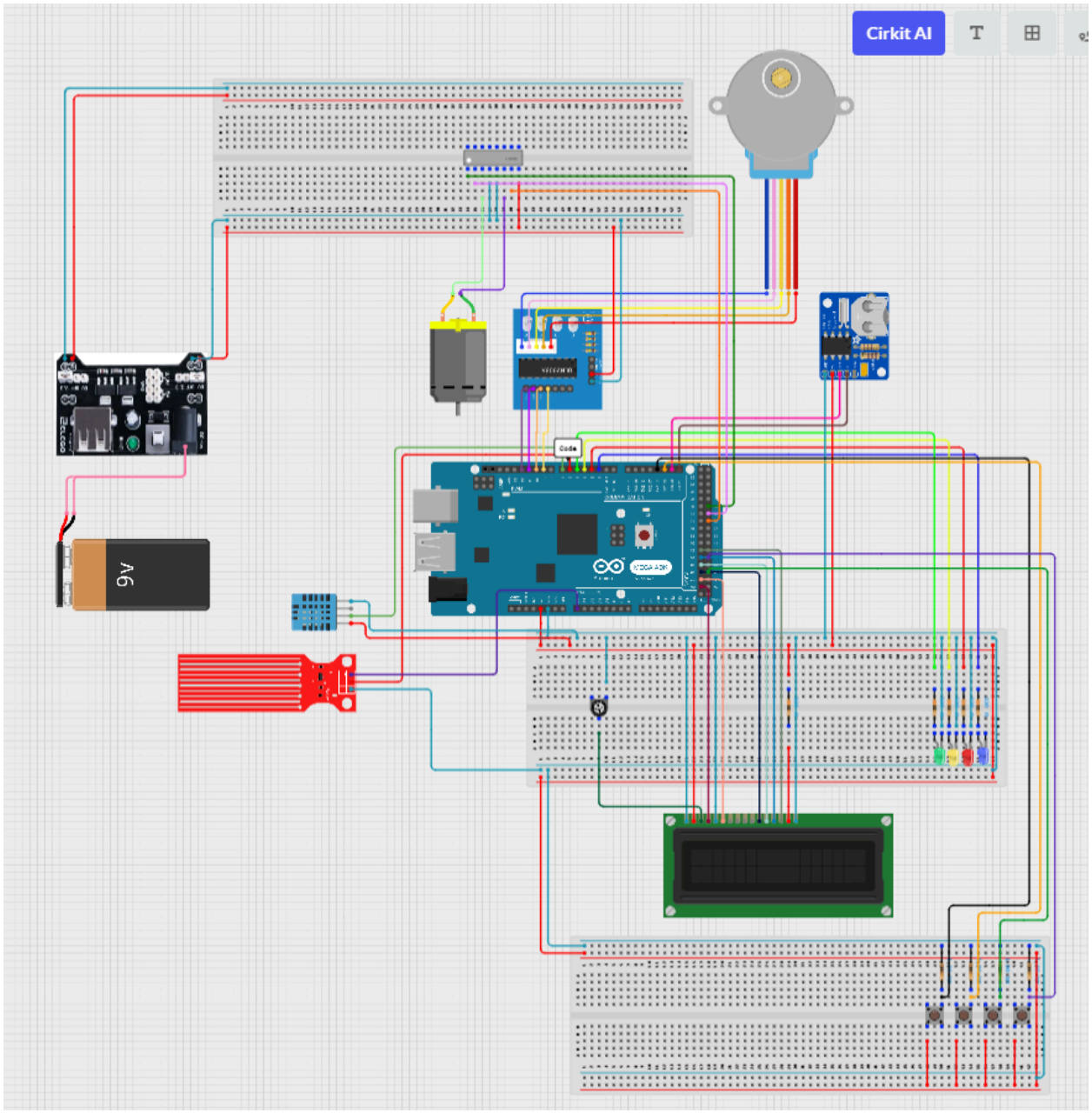
-Error:

-red led is powered
-error message is displayed on lcd
-vent controls are off
-water level monitoring is on
-fan is off
-swap to disable if stop button is pressed
-swap to idle if reset button is pressed and waterLevel is above minimumWaterLevel

Full Circuit Picture:



Schematic:



Result:

```
From DISABLED to IDLE at 05/12/24 21:56:05
From IDLE to ERROR at 05/12/24 21:56:05
From ERROR to DISABLED at 05/12/24 21:56:06
From DISABLED to IDLE at 05/12/24 21:56:16
From IDLE to ERROR at 05/12/24 21:56:21
From ERROR to IDLE at 05/12/24 21:56:28
From IDLE to RUNNING at 05/12/24 21:56:44
From RUNNING to IDLE at 05/12/24 21:57:21
From IDLE to RUNNING at 05/12/24 21:57:25
From RUNNING to ERROR at 05/12/24 21:57:44
From ERROR to IDLE at 05/12/24 21:57:52
From IDLE to RUNNING at 05/12/24 21:57:53
```

Spec Sheets:


https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf
https://www.electronicshub.org/wp-content/uploads/2021/01/Arduino-Mega-Pinout.jpg
https://components101.com/displays/16x2-lcd-pinout-datasheet
https://wiki-content.arduino.cc/documents/datasheets/LEDY-L-7113YT.pdf
https://wiki-content.arduino.cc/documents/datasheets/Button.pdf
https://arduinogetstarted.com/tutorials/arduino-water-sensor
https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/
https://www.electronicoscaldas.com/datasheet/ULN2003A-PCB.pdf
https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/
https://www.sparkfun.com/datasheets/Components/l293.pdf
https://wiki-content.arduino.cc/documents/datasheets/DCmotor6_9V.pdf