CPE 301 Final Project Report

Group 12 Members: Dax Quick, Autumn Heckler, Tracey Msangi, Ziyue Liang
Github: https://github.com/DaxQuickUNR/CPE301_FinalProject/blob/main/MainProjectCode.ino

Project Description

The goal of this project was to create a swamp cooler system prototype using Arduino components that demonstrates a real world Embedded Design engineering application. This device monitors temperature & humidity levels, and uses this data in order to turn on a fan that cools an area. This project does not allow for the use of the regular Arduino or Serial libraries, so everything must also be coded using its raw components. The Arduino Mega 2560 and the components included in the ELEGOO starter kit are utilized for the purpose of this project. Specifics about this project can be found in the *"System Overview"* section.

Component Details

DC Motor Driver (L293D)
We used the L293D motor driver to control the DC fan in our system. The driver receives control signals from the Arduino Mega and uses them to turn the fan on or off. It is connected to three pins on the Arduino: one for enabling the motor and two for setting the direction. When the temperature gets too high, the Arduino sends a signal to the L293D, which powers the fan to help cool the system. The motor driver gets power from an external source and makes sure the fan works safely without putting too much load on the Arduino. This driver helps the system respond to temperature changes by turning the fan on when needed.

Stepper Motor Driver (ULN2003)
We used a stepper motor with a ULN2003 driver board to control the angle of the vent in our cooler system. The driver board is connected to four digital pins on the Arduino Mega (pins 22, 24, 26, and 28), which send signals to move the motor step by step. We used two push-buttons to control the direction—one button turns the motor clockwise, and the other turns it counterclockwise. These buttons are connected to pins D42 and D44 on the Arduino. In the code, we set the motor speed and used simple functions to make it move one step each time a button is pressed. The driver board uses its own power to run the motor, so the Arduino just sends small control signals. This setup helps us move the vent accurately and smoothly depending on the system's current state.

Stepper Motor (28BYJ-48)
The stepper motor adjusts the vent direction based on what the user wants. It is controlled using the Arduino Stepper library and is connected to pins 22, 24, 26, and 28. It allows the vent to rotate clockwise or counterclockwise when the respective buttons are pressed. It is designed to only work when the system is not disabled.

Real Time Clock (DS1307)

We used the DS1307 real-time clock (RTC) module to keep track of the date and time. It is connected to the Arduino Mega using the I2C interface (SCL and SDA pins). The RTC runs on its own battery, so it remembers the time even when the system is turned off. In our project, the RTC is used to record the exact time when the fan turns on due to high temperature. The time and date are sent through the serial port so we can monitor the system's activity. This module helps the system provide accurate time information and makes it easier to log events.

Temperature/Humidity Sensor (DHT11)

We used the DHT11 sensor to measure the temperature and humidity inside the system. It is connected to digital pin 7 on the Arduino Mega. The sensor sends data to the Arduino every second. In our code, we used a built-in library to read the temperature and humidity values. The temperature is shown in Fahrenheit, and the humidity is shown as a percentage. These values are displayed on the LCD screen so we can easily see the current conditions. If the temperature goes above a set limit, the system will turn on the fan to help cool down. This sensor helps the system make smart decisions based on real-time temperature and humidity readings.

Arduino Mega 2560

This is the microcontroller board used for this project. It is responsible for managing all sensors along with the inputs and outputs of the system. It is also responsible for communication between parts. In this project, we used it to connect and control the fan motor, stepper motor, RTC module, LCD, DHT11 sensor, water level sensors, and status LEDs. Its functionality was implemented through direct register access like ADC, UART, and timers. This component ensures that each part of the system is coordinated. Additionally, some communication happens between a host PC and the Arduino over Serial (not using library), in order to send time data for logging purposes.

LEDs

We used four LEDs to show the different states of the system, and to serve as a visual notifier of state changes. Yellow is used to show that the system is disabled, green for when the system is idle, blue when the system is running, and red when there is an error in the system. Only one of the LEDs is on at a time based on the system's current state. They are all connected to an output pin with a resistor to limit the current going through them.

Buttons

Our system incorporates several buttons. Two are used to adjust the angle of the "vent", and the other is an ON/OFF button. For the vent control buttons, one spins the motor clockwise, while the other spins it counterclockwise.

Power Supply Module
The power supply module receives high DC inputs and outputs lower voltages. For example, it may receive a 12V DC input and output only 5V. This is used to power the fan motor and the stepper motor without fully depending on Arduino's current and voltage supplies.

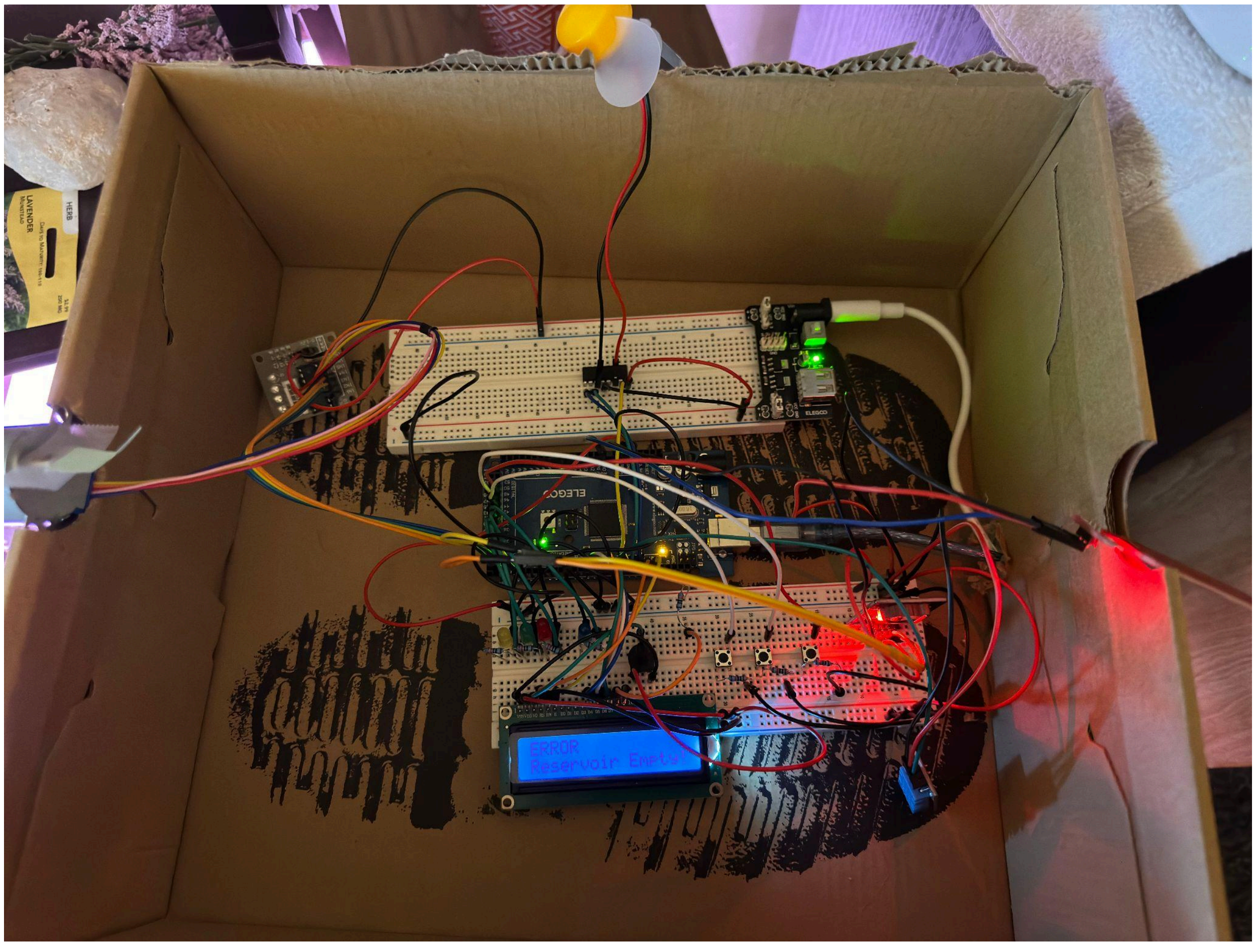
Water Level Detection Sensor
We used a water level detection sensor to check how much water is in the tank. It is connected to the Arduino Mega through analog pin A0. The sensor gives an analog value based on how much water touches its probes—the more water, the higher the value. In our code, we read this value and compare it to a set limit. If the value is too low, the system sends a warning message through the serial port. This helps make sure the system does not run without enough water, which is important for the cooler to work correctly.

System Overview

The evaporation cooling system is designed to act just like a regular cooling unit by using Arduino Mega 2560. The system has a disabled stage, an idle state, a running state, and an error state. Initially, the system is in a disabled state. In this state, the yellow LED is on and the system stays like that waiting for a start signal. When the system is activated, the system enters the idle state. When in this state, the DHT11 sensor monitors the temperature and humidity while the water level sensor records the reservoir status using ADC routines. The data is then displayed on an LCD using the liquidCrystal library. We also used millis() to schedule sensor updates after every 60 seconds for the non-blocking delays. When the temperature is higher than the threshold value and there is enough water, the system enters the running state. The fan motor also gets activated and the event is logged through an RTC. During very low water levels, the system enters the error state and the red LED lights up. This is accompanied by a warning message on the LCD.

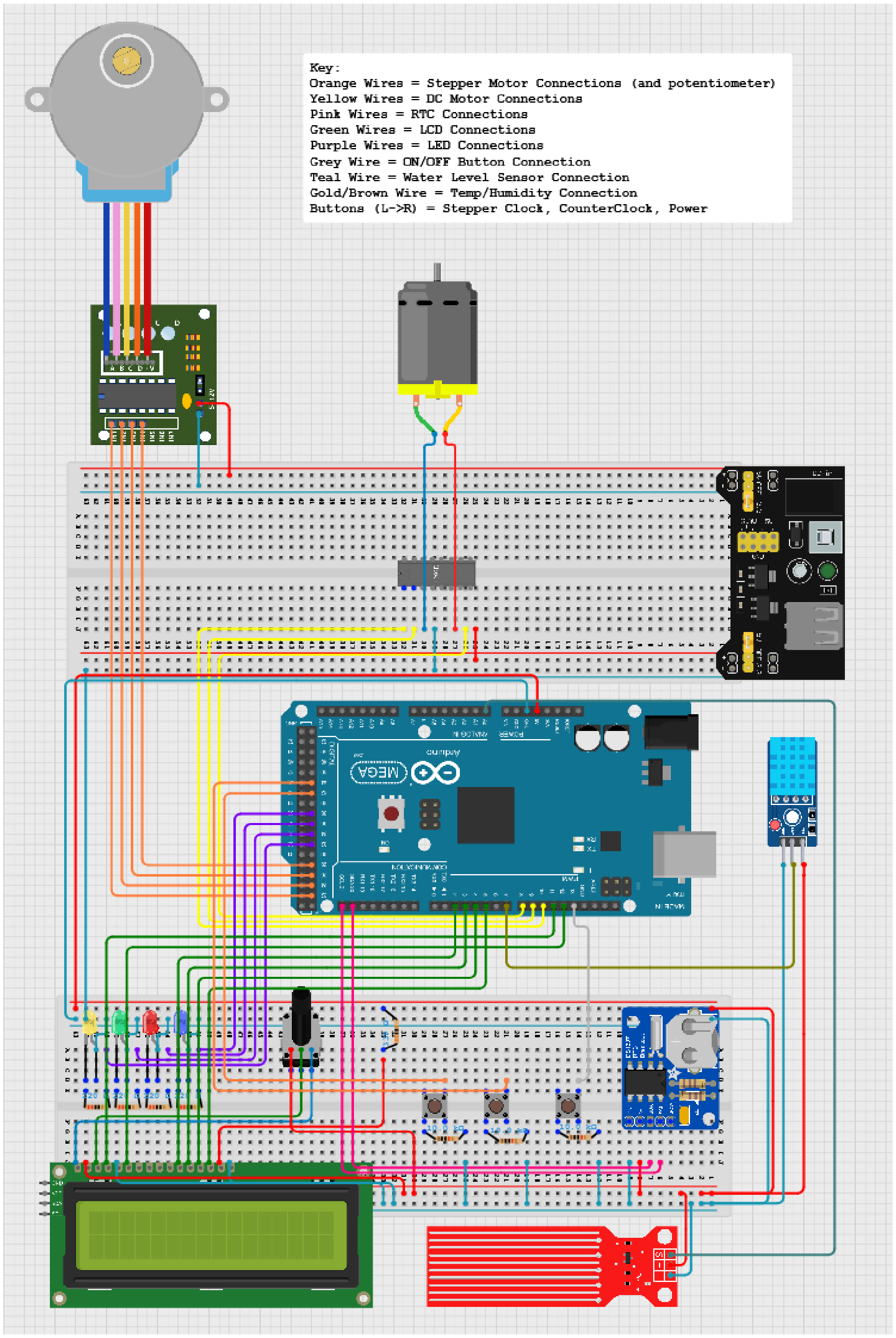
Some of the hardware used to build this system include the sensors(DHT11 and water level), modules(power supply and RTC), motors(fan and stepper), LCD, external power supply, LEDs and buttons. To avoid using restricted functions like digitalWrite, we used direct register access to handle all ADC, GPIO, UART and timer functionality. In the system, Timer1 is used for delays and UART is used to send time-stamped serial output. This logs each state change and vent adjustments. The stepper motor is controlled by the Arduino stepper library and buttons. It simulates the airflow vent direction changes and stays the same unless the system enters the disabled state. The fan motor and stepper motor are powered by an external power module to avoid overloading the Arduino regulator. A voltage of 5V is supplied to them from the external source. We assembled all this together on a breadboard and used programming to get control of them within the system.

Circuit Image



```
ERROR
Reservoir Empty
```

Schematic Diagram

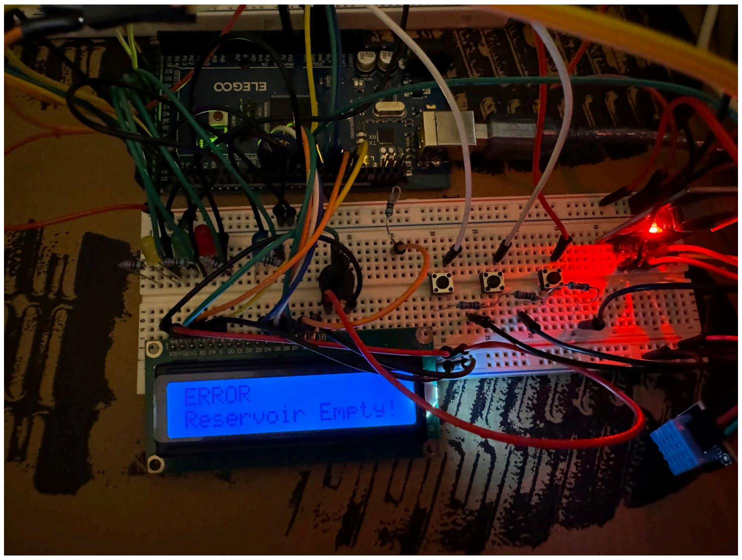

Key:
Orange Wires = Stepper Motor Connections (and potentiometer)
Yellow Wires = DC Motor Connections
Pink Wires = RTC Connections
Green Wires = LCD Connections
Purple Wires = LED Connections
Grey Wire = ON/OFF Button Connection
Teal Wire = Water Level Sensor Connection
Gold/Brown Wire = Temp/Humidity Connection
Buttons (L->R) = Stepper Clock, CounterClock, Power
MEGA

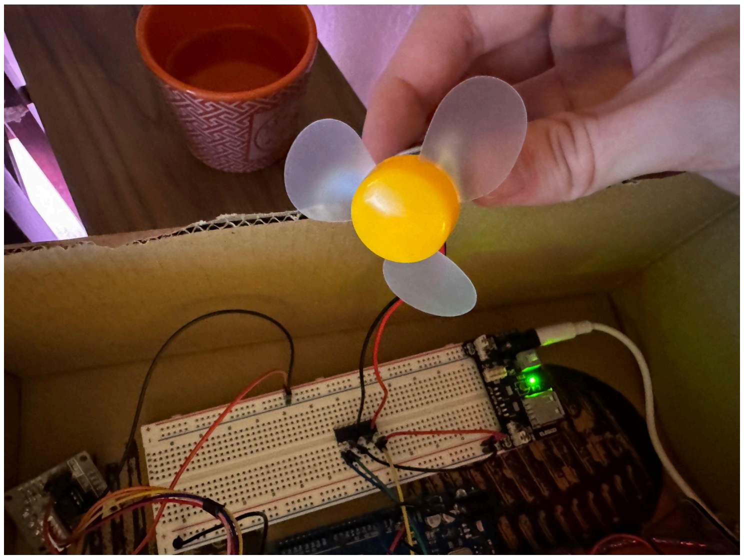
Schematic Link: https://app.cirkitdesigner.com/project/849ade3d-b360-4751-835d-927aa1de449c

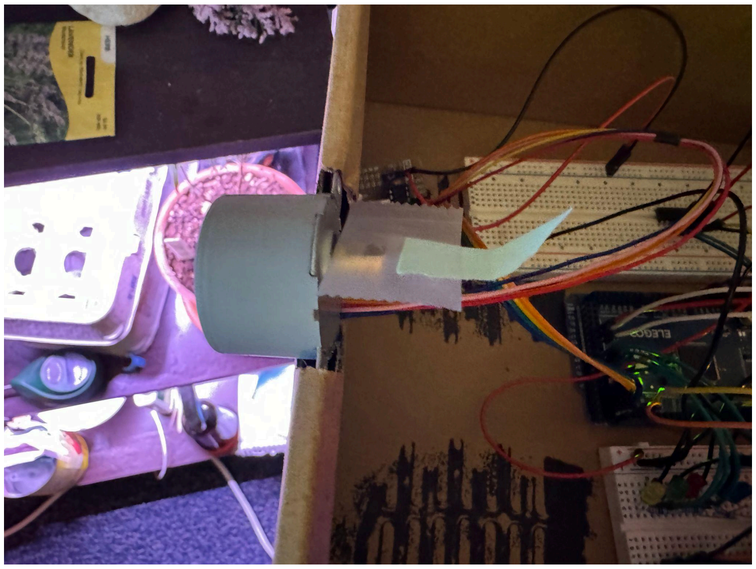
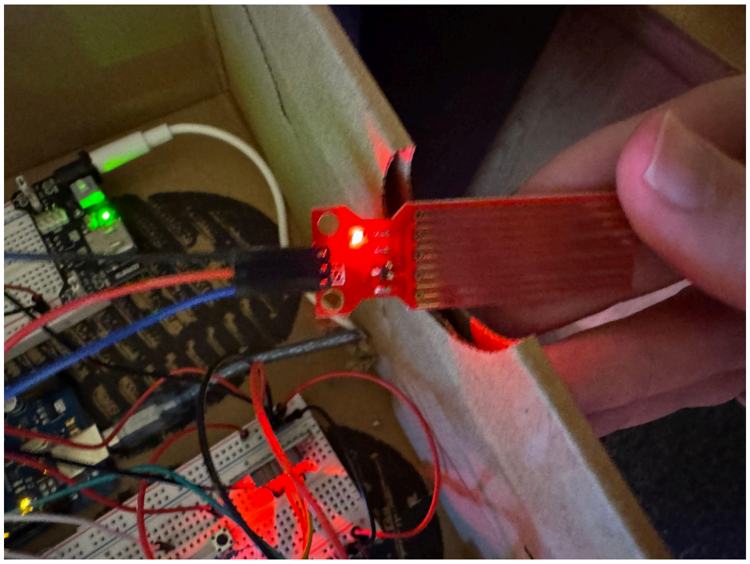
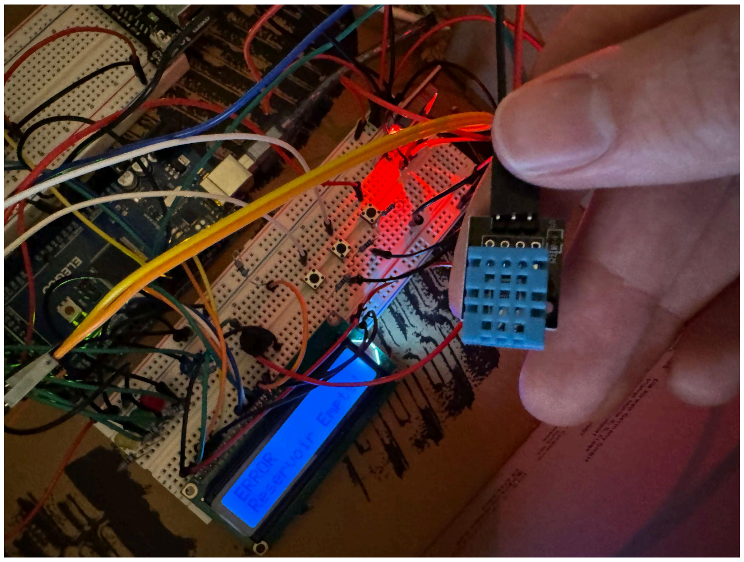
System Demonstration

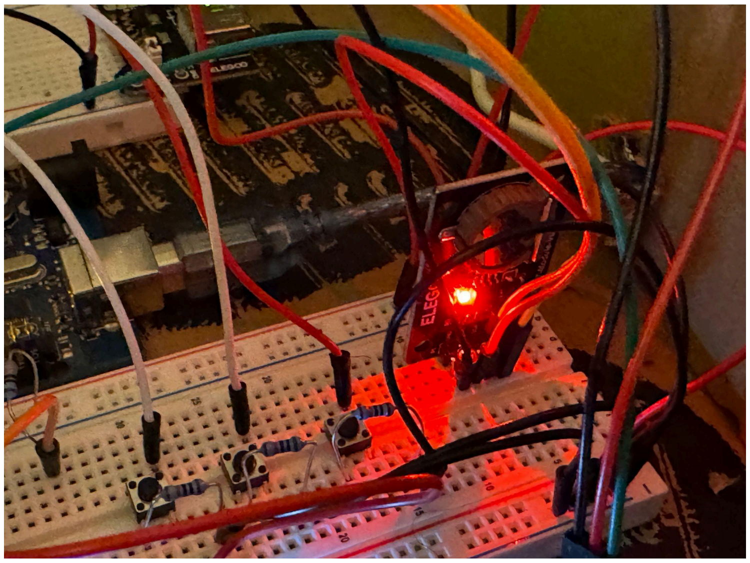
Please see video linked here for system demonstration:
https://drive.google.com/file/d/1H0WdED5H1JUxKTY1FTY1ZMQCVD70IHdq/view
087c8dc36a3effeaf320187479687a01.mp4

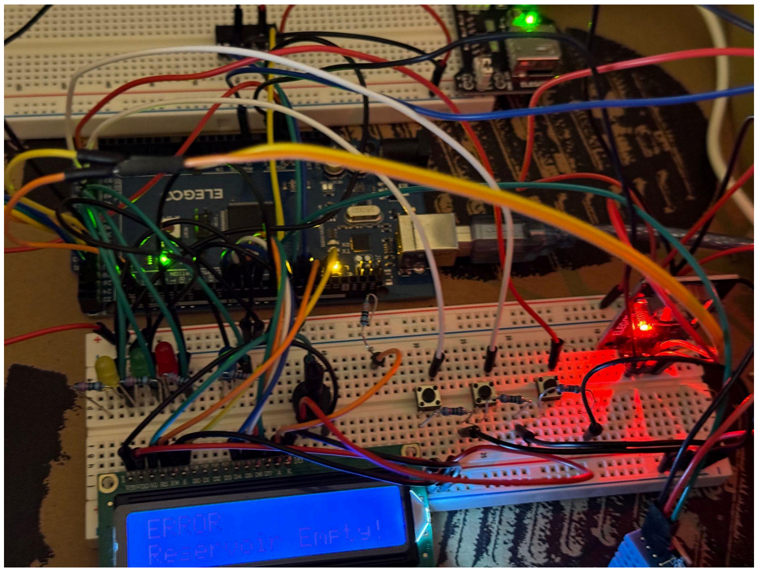
Note: Images are of Dax's prototype while video is of Zoey's prototype so they may appear to have slight positioning differences.

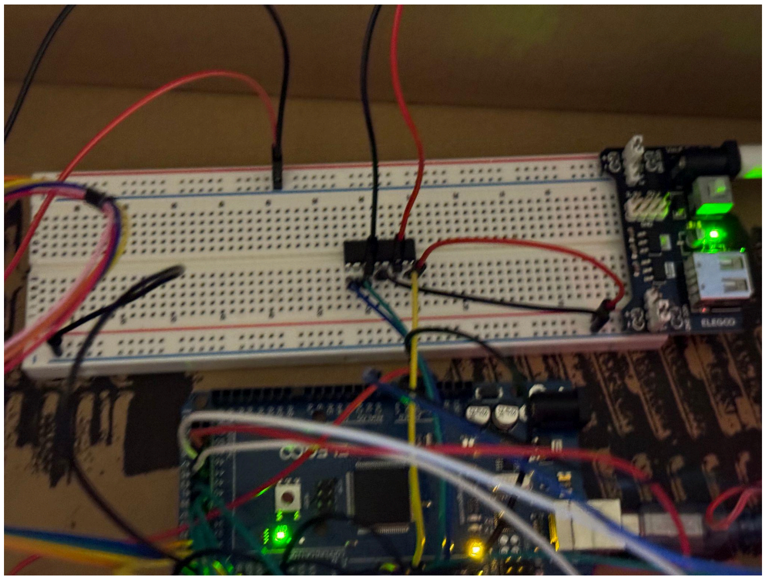







ERROR
Reservoir Empty


ELEGOO


ELEGOO
ERROR
Reservoir Empty!


ELEGOO