CPE 301 Final Project

Brandon Lafferty

December 12, 2025

1 Project Description

This project implements a fully functional evaporative cooling (swamp cooler) system using an Arduino AtMega 2560. The system monitors ambient temperature and humidity using a DHT11 sensor and ensures safe operation by continuously checking the water level in the reservoir through a water level sensor. Based on these sensor inputs, the system controls a fan motor to provide cooling when environmental conditions exceed the defined temperature threshold.

The system is structured as a finite state machine with four operating states: DISABLED, IDLE, RUNNING, and ERROR. User interaction is provided through dedicated Start, Stop, Reset, and vent control buttons. A stepper motor adjusts the vent direction in real time, and a real-time clock (RTC) records timestamps for all major events, including state transitions, sensor readings, fan activation, and vent position changes. All logged events are transmitted to a host computer via UART over USB.

2 System Overview

The swamp cooler system operates autonomously once enabled by the user via the Start button, which is handled using an external interrupt. Temperature and humidity data are sampled from the DHT11 sensor once per minute using a non-blocking timing approach based on the millis() function. These values are displayed on a LCD1602 and logged with timestamps provided by the RTC module.

The system uses a temperature threshold of 24.3°C to determine when cooling is required. When the temperature meets or exceeds this threshold and sufficient water is present, the system transitions to the RUNNING state and activates the fan motor. A water level threshold of ADC value 300 is used to detect low water conditions. If the water level drops below this threshold at any time, the system immediately transitions to the ERROR state and disables the fan.

All timing constraints are implemented without blocking delays to ensure responsive operation. The fan motor is powered using a separate power supply board to prevent damage to the Arduino's output circuitry.

3 State Machine Design

3.1 State Diagram

The system logic is implemented as a finite state machine to ensure predictable behavior and safe operation. State transitions occur based on sensor readings, button inputs, and system events. Each transition is logged with a real-time timestamp using the RTC module.

3.2 State Descriptions

3.2.1 Disabled State

- Default state on power-up
- Fan is forced into the off mode
- No sensor sampling is performed

- LCD displays a ”disabled” message
- Yellow LED is turned ON
- Start Button is monitored using an external interrupt (ISR)

3.2.2 Idle State

- Temperature and humidity are sampled once per minute
- Water level is continuously monitored via ADC
- Green LED is turned ON
- transitions to RUNNING when temperature is above 24.5°C
- Transitions to ERROR if water level falls below the threshold

3.2.3 Running State

- Fan motor is engaged via PORTH bit PH5
- Blue LED is turned ON
- Temperature and humidity continue to be monitored
- Transitions to IDLE when temperature drops below the threshold
- Transitions to ERROR if water level falls below the threshhold

3.2.4 Error State

- Fan motor is disabled
- Red LED is turned ON
- LCD displays a low water warning message
- Reset button returns system to IDLE if the sensor detects adequate water level

4 Component Details

4.1 Arduino AtMega 2560

The Arduino Mega 2560 serves as the central controller for the system. It manages all sensor inputs, state transitions, motor control, LCD updates, interrupt handling, and UART communication. Its expanded I/O capability allows simultaneous connection of multiple peripherals.

4.2 DHT11 Temperature and Humidity Sensor

The DHT11 sensor is connected to digital pin 22 and provides ambient temperature and humidity measurements. These readings are used for fan control logic and user feedback via the LCD display.

4.3 Water Level Sensor

The water level sensor is connected to ADC channel 0 (PF0). The sensor is sampled using direct ADC register access rather than Arduino ADC library functions. If the ADC reading falls below 300, the system transitions into the ERROR state.

4.4 Fan Motor and Power Supply

A DC fan motor from the kit is connected to digital pin D8 (PH5). The motor is powered through a dedicated external power supply board. Fan control is achieved by directly manipulating PORTH registers.

4.5 2N2222 Transistor

A power transistor is used on the motor power supply board to act as an electronic switch between the fan motor and its external power source. The Arduino controls the base/gate of the transistor using a low-current digital signal, allowing the higher-current motor circuit to be safely switched on and off. This isolation protects the Arduino from high current loads and voltage spikes generated by the motor.

4.6 Diode Rectifier

A flyback diode is placed across the fan motor terminals to protect the circuit from inductive voltage spikes. When the motor is switched off, the collapsing magnetic field can generate a high-voltage transient that may damage the transistor or microcontroller. The diode provides a safe path for this current, clamping the voltage and ensuring reliable long-term operation of the power supply circuitry.

4.7 Stepper Motor

A stepper motor connected to pins 50–53 controls the vent direction. Two buttons allow the user to rotate the vent left or right in increments of 10 steps. Vent movement is rate-limited to once every 150 ms and is disabled only in the DISABLED state. All vent movements are logged with timestamps.

4.8 LCD1602 Display

A 16x2 LCD connected to pins 30–35 displays system state, temperature, humidity, and error messages. The display is refreshed during each sensor update cycle.

4.9 DS1307 Real-Time Clock Module

A DS3231 RTC module provides accurate timekeeping. It is used to timestamp state transitions, sensor readings, fan events, and vent adjustments. All timestamps are transmitted via UART to a host computer.

4.10 Buttons and LEDs

Three system buttons (Start, Stop, Reset) and two vent control buttons are used for user interaction. LEDs indicate system state:

- Yellow: Disabled
- Green: Idle
- Red: Error
- Blue: Running

4.11 Resistors

Resistors are used throughout the project to ensure safe and reliable operation of the electronic components. Current-limiting resistors are placed in series with the state indicator LEDs to prevent excessive current draw and protect both the LEDs and Arduino I/O pins. Pull-up or pull-down resistors are used on button inputs to maintain stable logic levels and prevent floating inputs when buttons are not pressed. Additional resistors on the motor power circuitry provide proper transistor biasing, ensuring consistent and reliable switching of the fan motor.

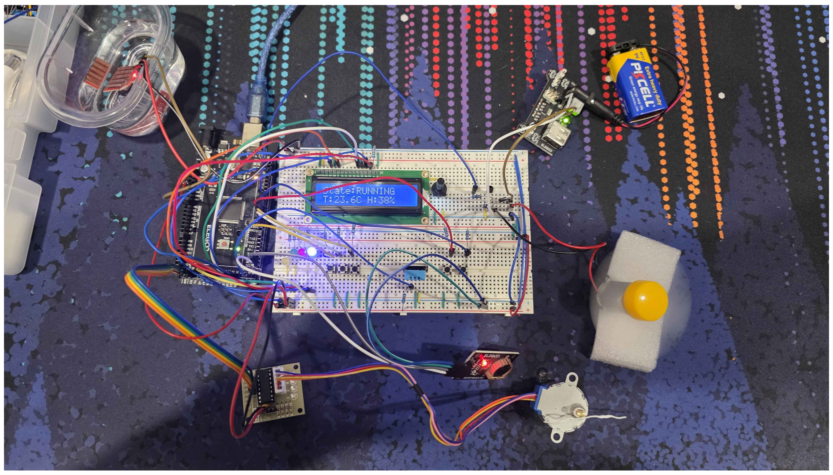


Figure 1: Completed breadboard build.

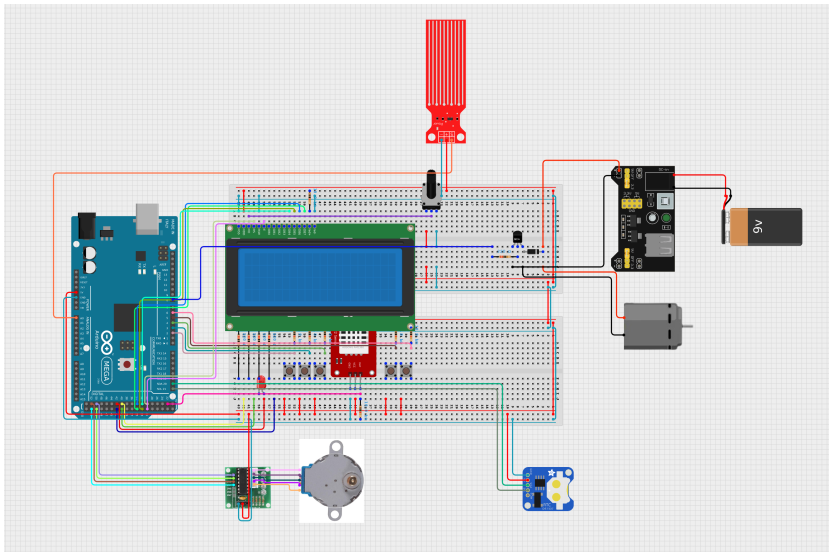


Figure 2: Schematic diagram utilizing Cirkit software.

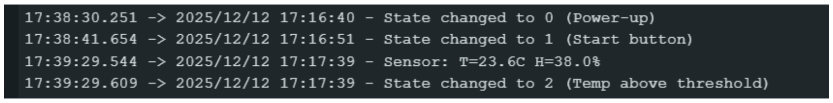
5 Circuit Design

5.1 Physical Image

Figure 1. shows a full image of the completed swamp cooler circuit, with all components and wiring connections.

5.2 Schematic

Figure 2 shows a diagram that illustrates the pin connections between the Arduino Mega and all peripherals, including sensors, motors, buttons, LEDs, and the LCD.



```
17:38:30.251 -> 2025/12/12 17:16:40 - State changed to 0 (Power-up)
17:38:41.654 -> 2025/12/12 17:16:51 - State changed to 1 (Start button)
17:39:29.544 -> 2025/12/12 17:17:39 - Sensor: T=23.6C H=38.0%
17:39:29.609 -> 2025/12/12 17:17:39 - State changed to 2 (Temp above threshold)
```

Figure 3: Serial Monitor Log showcasing the state changes with real-time clock.

6 Software Design

6.1 Reigister-Level Programming

Direct register manipulation is used throughout the project. GPIO registers control LEDs, buttons, and the fan motor. The ADC is configured using ADMUX and ADCSRA registers to sample the water level sensor. UART communication is implemented using UART0 registers for event logging.

6.2 Interrupt Service Routine

An external interrupt is configured on digital pin 2 for the Start button. The ISR sets a volatile flag that is handled in the main loop, ensuring immediate response while maintaining system stability.

6.3 Library Usage

Default Arduino libraries are used for the LCD, DHT11 sensor, RTC module, and stepper motor. Restricted Arduino functions such as pinMode(), digitalWrite(), digitalRead(), delay(), and analogRead() are not used in the program whatsoever.

7 Conclusion

This project demonstrates a complete embedded system integrating sensors, actuators, interrupts, and real-time monitoring. By implementing a structured state machine and adhering to low-level programming constraints, the system fulfills all project requirements while reinforcing key embedded systems concepts.