Swamp Cooler Project: Technical Document

- **Title**: *Swamp Cooler*
- **Author**: Cole Gilliam, Jordan Cadelina
- **Date**: December 10th, 2024
- **GitHub:** https://github.com/1203-Gilliam-Cole/CPEFInalProject

Table of Contents

1. Introduction:

The swamp cooler automation project focuses on creating an efficient and practical cooling system by integrating a start/stop push button, a water level detection sensor module, and a motor for air circulation. This project is designed to enhance the functionality and usability of traditional swamp coolers, which are known for their energy efficiency and environmental benefits compared to conventional air conditioning systems.

The project incorporates core concepts of embedded systems and electronics, including the use of a microcontroller (Arduino) to control various components. The system uses push buttons for toggling the operation, making it user-friendly and intuitive.

This project aims to develop a reliable, low-cost cooling solution suitable for residential or small-scale industrial applications, while also providing a hands-on learning experience in circuit design, sensor integration, and motor control. The final system will showcase how simple yet effective automation can be implemented to improve everyday devices.

2. Objectives:

The primary goal of this project is to develop a functional evaporative cooling system, commonly known as a swamp cooler, using the Arduino 2560 microcontroller and associated sensors and components. This system will provide a more energy-efficient alternative to air conditioning in dry, hot climates by utilizing the cooling effect of water evaporation. The specific objectives include:

1. Monitoring water levels in a reservoir and displaying alerts when levels are too low to ensure safe operation.
2. Continuously monitoring and displaying air temperature and humidity data on an LCD screen.
3. Automatically starting and stopping the fan motor based on temperature thresholds to maintain a comfortable environment.
4. Allowing the user to adjust the angle of the system’s air vent using a stepper motor, controlled via a potentiometer or buttons.
5. Enabling and disabling the system using an intuitive on/off push button.
6. Recording and transmitting motor operation events, including timestamps of start and stop actions, to a host computer over USB.
7. Ensuring safe electrical operation by using separate power supplies for high-current components like the motor.

3. System Overview:

The swamp cooler system integrates various hardware and software components to automate and enhance the functionality of a traditional evaporative cooling device. At its core, the system is controlled by an Arduino 2560 microcontroller, which manages inputs from sensors and user controls while driving the outputs for the fan motor, LCD, and stepper motor.

The system operates through several key states, including Disabled, Idle, Error, and Running, with transitions triggered by user inputs or sensor data. It monitors environmental parameters such as temperature and humidity, displays real-time data on an LCD screen, and adjusts the system's operations accordingly. The system also includes the following functional features:

1. **Water Level Monitoring**: A water level sensor ensures that the reservoir maintains adequate water for cooling. If the water level falls below a set threshold, the system transitions to an error state and disables the fan motor to prevent damage.
2. **Temperature and Humidity Monitoring**: A DHT11 sensor continuously measures air temperature and humidity. This data is displayed on an LCD screen and used to control the fan motor based on pre-set temperature thresholds.
3. **Vent Direction Control**: A stepper motor adjusts the angle of the air vent, allowing users to direct airflow as needed. The angle is controlled via a potentiometer or buttons, depending on the user's preference.
4. **User Interface and Control**: A push button toggles the system between enabled and disabled states. In the disabled state, all monitoring ceases, and a yellow LED indicates the inactive status. A reset button re-enables the system from an error state if conditions allow.
5. **Event Logging and Reporting**: A real-time clock (RTC) module records timestamps for critical events, such as motor start and stop, and transmits this data to a host computer via USB for analysis and record-keeping.

The system is designed to be energy-efficient and user-friendly while ensuring safe and reliable operation through careful integration of sensors and hardware components. By leveraging modular and reusable code, the project serves as a practical demonstration of embedded system design principles.

4. Materials and Components:

- Arduino 2560 microcontroller
- Water level sensor for monitoring the reservoir
- DHT11 temperature and humidity sensor
- DC fan motor with a separate power supply board
- Stepper motor for vent direction control
- 16x2 LCD display for data visualization
- Start/stop push button for enabling and disabling the system
- Reset button for recovering from the error state
- Potentiometer for adjusting vent direction (optional)
- Real-time clock (RTC) module for event logging and reporting
- Yellow LED for indicating the disabled state
- Green LED for indicating the idle state
- Red LED for indicating the error state
- Blue LED for indicating the running state
- Assorted jumper wires and connectors for circuit connections
- USB cable for programming and data transmission
- Breadboard or PCB for circuit assembly
- Resistors for pull-up or pull-down configurations

5. Circuit Diagram and Wiring

Our notes, wiring, and circuit diagrams were mostly done on GOODNOTES. We considered using tinkercad but there is no water level sensor similar to the one needed in the project so we decided not to continue with it. Wiring was not deeply thought out enough at first so constant changes had to be made through the code and the bread board.

6. Software Design, Challenges, etc

The software design for the swamp cooler project focused on managing the system's states and transitions effectively. To achieve this, we implemented a state machine using a switch-case structure within the main loop of the program. This approach was chosen because it allows for clear organization of each state, detailing the specific functions and conditions that govern state transitions. The states include Disabled, Idle, Running, and Error, each with distinct behaviors such as monitoring sensors, controlling outputs, and responding to user inputs. Using switch-case made it straightforward to visualize the system's workflow and ensured that each state operated independently with minimal overlap in functionality.

One major challenge we encountered involved the Interrupt Service Routines (ISRs) for handling the push buttons. Initially, the system worked well with a single Start/Stop button. However, when the Reset button was added, the ISRs began to exhibit unpredictable behavior. Inputs from one button were sometimes mistaken for the other, or both buttons triggered the same action despite being programmed separately. These conflicts resulted in unintended state transitions, requiring additional debugging and mitigation efforts.

Another issue was the overall performance of the code, particularly in terms of register manipulations. Although we avoided using Arduino libraries for certain operations to gain finer control, the code often exhibited unexpected behaviors, such as delayed responses or skipped actions. These problems suggested that our implementation was not fully optimized for direct register control, possibly due to inefficiencies in handling bit-level operations or managing interrupts.

7. Contributions

Cole Gilliam: I was mostly on the software development side but we did work on the code together in person. I also helped in debugging / designing the circuitry. The workload I would say is evenly split.

Jordan Cadelina: I worked mostly on the breadboard and wiring. We did most of the work in person but I helped debug any code when the time was needed. The workload was fair split where one person specialized more on a specific area.

8. Video Description

In the video it explains the process of each state and demonstrates how each state works and what should and shouldn't be happening. The only part of the video that doesn't follow the rubric is the Start/Stop button from IDLE-RUNNING. This is because our DHT11 was not operable so we couldn't implement that into our build. Because of this in order to get from IDLE-RUNNING we used the Start/Stop button.

9. Conclusion

The swamp cooler project was a success, achieving its primary goals of automating and enhancing the functionality of an evaporative cooling system. By including different uses of sensors, wiring, and interactions of the interface we were able to create a successful swamp cooler that anyone would be able to use.. While challenges with button ISRs and optimization highlighted areas for improvement, the project provided valuable hands-on experience in embedded system design and problem-solving.