Araam Zaremehrjardi

Thomas Simcox

CPE 301 – Team Project

**Group 35**

Project Overview Document

*Design Overview*

The main goal for this project was to produce a system that adheres to the criteria as outlined in the assignment’s description. However, beyond that, another goal was to create a simple, readable, and maintainable implementation of the C code component as well as the hardware component. We aimed to achieve this in the following ways:

**C Macros -** We used a series of macros that all follow a naming convention to keep everything readable. In our system, each macro name is prefixed with a “M\_”, followed by a description of what it defines. This allows us to clearly keep track of state throughout the rest of the program without having to remember exactly what value each state represents.

*C Macro Examples in our code:*

#define M\_SYSTEM\_STATE\_IDLE 0

#define M\_SYSTEM\_STATE\_DISABLED 1

#define M\_SYSTEM\_STATE\_ERROR 2

#define M\_SYSTEM\_STATE\_RUNNING 3

**Naming conventions for output variables –** With the same goal in mind, keeping code simple and readable, we prefixed each variable that references anything having to do with output in the system. this includes LEDs, the display screen, etc.

*Example:*

unsigned char o\_running\_state\_LED

unsigned char o\_error\_state\_LED

unsigned char o\_idle\_state\_LED

unsigned char o\_disabled\_state\_LED

**Naming conventions for global variables –** Finally, we adhered to a similar naming convention for global variables:

*Example:*

unsigned char g\_system\_state = M\_SYSTEM\_STATE\_DISABLED

*Programming Approach*

The main loop of our program mostly consists of a single switch statement, that switches on the global variable g\_system\_state, which represents the system state. Each case represents a different state option to determine which state we’re in, and what action we need to perform next (i.e. switch to idle, switch to error state).

case M\_SYSTEM\_STATE\_IDLE:

• if the current water level is less than the threshold, set the system state to idle.

case M\_SYSTEM\_STATE\_DISABLED:

• Illuminate LEDs.

case M\_SYSTEM\_STATE\_ERROR:

• if the current water level is greater than or equal to the threshold, set the system state to idle.

case M\_SYSTEM\_STATE\_RUNNING:

• Illuminate corresponding LEDs.

*Wiring Schematic:* **[Insert Wiring Schematic here]**

*GitHub Repository Link:*

<https://github.com/TommySimcox/CPE301-Araam-Thomas>

*Test Plan:*

General Acceptance Criteria:

* The system produces an alert when the water level is below a certain threshold.
  + The system does not produce an alert when the water is above that threshold.
* The system monitors the current air temperature.
  + The system enables or disables a fan when the temperature goes out of bounds of a given range.
* The system displays the current air temperature on the LCD screen.
* The system monitors the current humidity.
* The system displays the current humidity on the LCD screen.
* The system has a user-controlled output vent.
* The system can be toggled on/off by a user.
* The system logs the time and date every time it is turned on or off.

Test Cases:

State-indicating LEDs:

* Disable the system. A yellow LED light should be illuminated, and no monitoring of any sort should occur.
* Induce some sort of error to the system (i.e. water level too low). A red LED should illuminate and an error message should be displayed.
* Enable the system and ensure that a green LED illuminates.
* Set up the system so that it is in the running state (motor should be on) and assert that a blue LED is illuminated.

State:

* Press the power on/off button. The system should immediately enter the idle state.
* Move to a warmer environment, or start the system in a warmer environment. Assert that state changes to running.
* Allow the system to run, ensure that state returns to idle once the temperature drops sufficiently.
* Remove water from the reservoir. Assert that state changes to error state.
* Add water to reservoir. Assert that state changes back to idle.