Scott Smythe
CPE 301
Fall 2024
Final Report

Introduction: With this project our objective was to create the electrical innards of a swamp cooler device. This simple state machine reads inputs from sensors attached to a Mega 2560 AVR microcontroller and toggles a fan based on a temperature threshold. This project also combined all of our labs from the semester and required us to not use library implementations for GPIO management/Interaction, ADC usage, Serial Communication, and Timers (delay). The cooler cycles between 4 main states and 1 dummy state (NIL) which is only used internally for state management:

- DISABLED
    - When in this state the cooler has a lit yellow status LED, the display shows nothing, and the vent can still be controlled. Upon pressing the start button an interrupt triggers a state change request and the cooler switches into the next state. Pressing the start button in any other state returns to this state.
- IDLE
    - When in this state the cooler has a green status LED, the display shows the current temperature and humidity (updated every minute), the vent is also controllable in this state. Should the water threshold as detected by a water level sensor fall too low the cooler will switch into the ERROR state described below. Should the water level be high enough and the temperature also above a set threshold the cooler will transition into the RUNNING STATE.
- RUNNING
    - In this state the cooler has a blue status LED and the dc fan is switched on. Additionally the temperature and humidity are displayed just as in the IDLE state. While the software allows for the vent to be controlled in this state, the limitation of the power supply board can sometimes make using both simultaneously exhibit odd behavior (discussed more in the components section). Should the water level fall too low in this state the MCU will switch into the next state.
- ERROR
    - In this state the cooler has a red status LED and displays a message saying the water level is too low. The vents are not controllable in this state, there is however a dedicated reset button that transitions back into the IDLE state directly (as opposed to cycling through DISABLED first).

A schematic can be found here and a final image of the project can be found here.

Components Used:

- Arduino Mega 2560 MCU
    - This is the “brain” of the project that orchestrates all of the components used in this project. This microcontroller is used to read both analog and digital sensor data, as well as listen for interrupts controlled by buttons used throughout the

project. The kit datasheet can be found here, while the full MCU datasheet can be found here.

- Water Level Sensor
    - This is a simple analog sensor used to read the levels in the coolers' water reservoir. A Lower reading indicates a lower water level. Upon readings going under a set threshold the MCU transitions into an error state and writes to the LCD display that the water level is too low. The MCU interfaces with this sensor via the builtin ADC. The kit sheet can be found here, while a datasheet with technical specs can be found here.
- DHT11 Temperature Sensor
    - This is a digital sensor that sends temperature and humidity data in a serial fashion to a digital pin. When the temperature surpasses a set threshold and if the water level is high enough (i.e. not below the water level threshold) the MCU will trigger the running state and enable a 6V DC motor until such a time as the temperature goes below the threshold. The “DHT.h” library was used to interface with this component. A datasheet for this component can be found here.
- DS1307 RTC module
    - This module is used for time keeping and logging and sends time data in a serial fashion. When the MCU transitions its state or rotates the vent fans a logging message with a timestamp from the RTC module is sent over USB via a UART connection. The “RTCLib.h” library was used for interfacing with this module. A datasheet for the module can be found here.
- LDC1602 Module display
    - This is the display used to display temperature and humidity data, as well as an error message if the water level is too low. This display is hooked up to several digital pins of the MCU and is refreshed every minute in the IDLE and RUNNING states. The "LiquidCrystal.h" library was used to write to and toggle this display. A datasheet for the display can be found here.
- MB102 Power Supply Module
    - This module is used to provide power to the vent control stepper motor and the cooling fan. It has a 9V input which is not quite enough to drive both the stepper motor (rated for 5V operation) and the DC motor (rated for 6V operation) and thus we cannot run both of the motors simultaneously (without under-volting). This board is required for this project however as Motors can damage the MCU pins from pulling so much current. A datasheet for this supply can be found here.
- ULN2003 Stepper Motor Driver
    - This module is used to control the stepper motor and connects to the MCU with 4 pins. This makes interfacing with the stepper motor much more simple. A datasheet for this driver can be found here.
- 28BYJ Stepper Motor
    - This stepper motor is used to rotate the vents of the cooler, the user can press a button to rotate it either left or right. The "Stepper.h" library was used to interface with this motor. The buttons used to control this motor are connected to digital pins on the MCU that are configured as inputs with the internal pullup resistors

enabled. When the button is depressed it connects the pin to ground triggering an interrupt on the falling edge of the signal. The motor connects to the ULN2003 driver mentioned above. The datasheet for this motor can be found here.

- L293D H-Driver Motor Control IC:
    - This integrated circuit is used to control the motor used for the cooling fan. In this project PWM is not utilized to control fan speed and the motor is either running at full speed or off. The IC provides power to the motor and pulls power from the MCU for its internal logic. A datasheet for this IC can be found here.
- 6V DC Motor:
    - This motor is connected to the above IC and acts as the cooling fan for the project. When the MCU enters a RUNNING state, this is the motor that is turned on to start cooling. A datasheet for this component can be found here.

Additionally, 9 330 ohm resistors were used (datasheet), 4 push buttons were also used to control start, reset, left vent control, and right vent control (button datasheet).

Conclusion: This project was a fun and engaging practice in low level programming and a fantastic exercise in interfacing with microcontrollers and wiring simple digital circuits. While improvements can be made to the design (a larger power supply for the motors and a dedicated shroud to contain the internals) the overall function of the project served as a great demonstration for the fundamentals of embedded system design and implementation.