Group Members: Justin Davis
CPE 301
May 12 2024

Final Project ReadMe: Cooler

Description of Cooler With Components:

- **3 Buttons**: I used 3 Buttons during this project. The leftmost button was responsible for starting the cooler. It would start in a disabled state when run was pressed, but for it to proceed to idle, a start button must be pressed so it can start it over functionality. The middle button is responsible for restarting. This button is used when we reach an error and need to return to idle. Finally the rightmost button is used to stop the machine altogether and return to the disabled state.
- **4 LEDS**: I used 4 LEDs to represent each state the cooler is currently in. I used red to represent the error state, blue to represent the running state, green to represent the idle state, and yellow to represent the disabled state.
- **Water Level Detection:** I used a water level detection sensor to basically keep track of the water in the cooler. If the water level is too low, it would cause an error state to occur.
- **DHT Sensor:** The DHT sensor is used to gather two types of information. First, it is used to gather the temperature of the environment it is in, and it is used to gather the humidity in the environment it was in. This information was used to determine which stage of the cooler it should transfer to.
- **Stepper Motor and Driver Module:** The stepper motor is responsible for controlling the vents. I used this in all stages besides error to represent vents opening and closing to allow airflow. The driver was used to assist the motor in moving.
- **Liquid Crystal:** The liquid crystal was used to display text to the user of the cooler. Firstly it would display the temperature and humidity of the current environment, and also if the cooler reaches an error state, it would display that “Water Level is too low”
- **L293D and Fan:** in this project, the L293D IC and the Fan worked hand in hand to control when the fan turns on and what it does. The L293D IC was basically responsible for sending power to the fan and deciding how fast it would turn and its direction of spin. In the code, the fan would be on during the running phase of the cooler and would be off in every other phase.
- **GPIO with Pins:** For this assignment, I only used 3 ports, which are Port A, Port B, and Port H. I used Port A for the three buttons I have on the breadboard. I treated it as input to read the button presses. Port B was used for the LED lights, I used it as an output. And finally port H was used for the L293D IC and the fan.
- **Millis:** the millis function is used within my code to essentially slow it down and give everything time to execute without having every be constantly executing all at once. It worked together with these 3 variables, currentMillis, previousMillis, and interval. These three variables essentially slowed down the code.
- **AttachInterrupt:** I used the attach interrupt as my starting button, basically when the start button was pressed, the AttachInterrupt would be called and begin the cooler.
- **RTC:** The RTC, also called the Real-Time Clock was used to keep track of the real time that certain events occur. In this project, it was used to keep track of the time in which the stepper motor was turned off. When the motor was turned off, I would call the

printDateAndTime() function to essentially print all the needed information into the terminal.

- **ARDUINO MEGA 2560:** This piece of equipment is the brain of the cooler. It is responsible for reading the code I type in, and executing it by receiving and transmitting information from the pins.

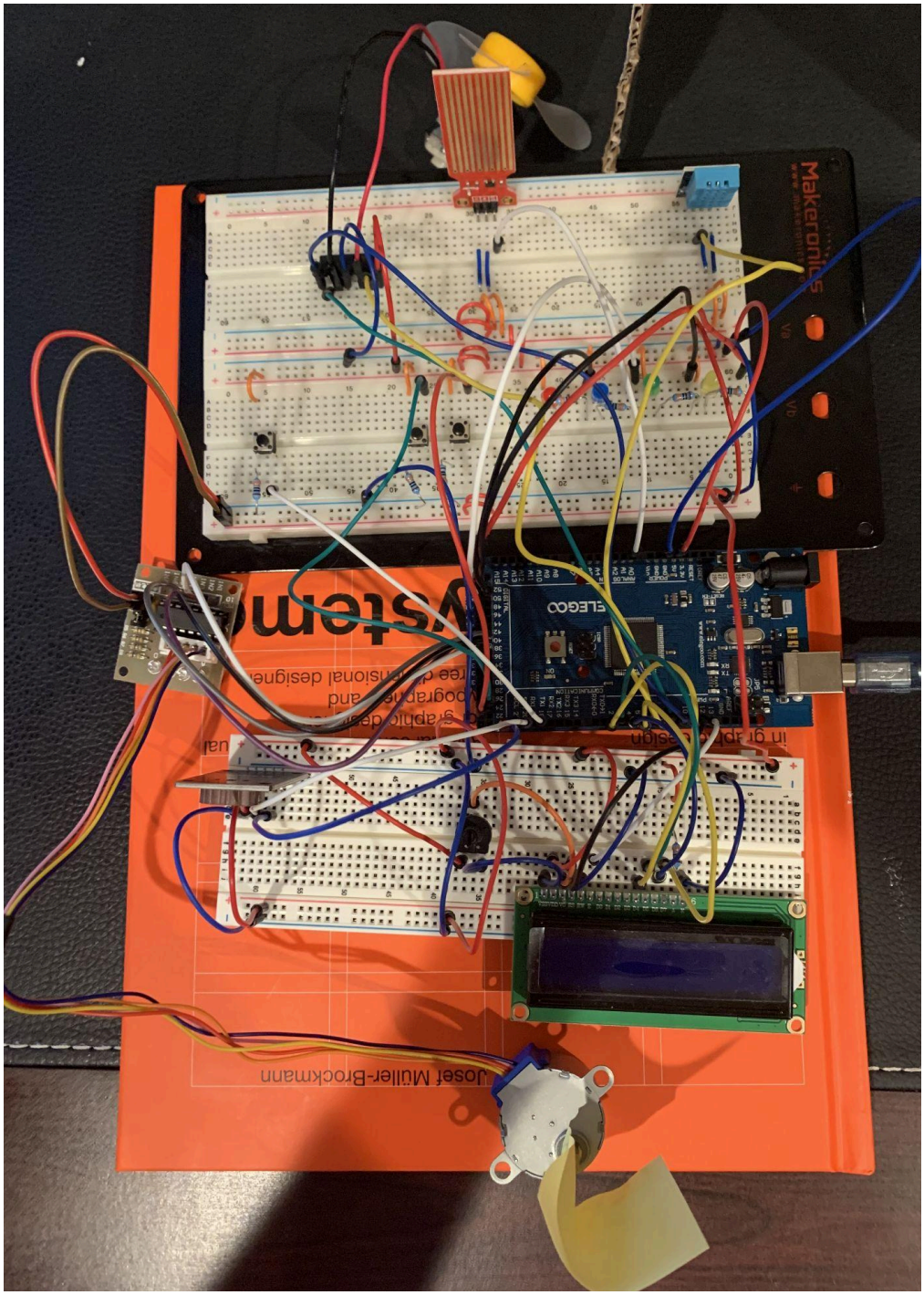
Description of Cooler Functions::

- **Void setup():** this function is responsible for doing that before the main code runs. It sets up the UART, it sets up the LCD, the ADC and the RTC. It also sets up the DDR ports by making them either inputs or outputs. I also used it to display the temperature on the LCD at the beginning so it is already set up and I called the attachInterrupt in here so when the start button is pressed the cooler starts.
- **Void loop():** this loop is quite straightforward, because it is just an if-statement. It is responsible for calling a function updateCurrentState() (which is explained below) and then after that function executes, it checks whether the currentState, which was modified in updateCurrentState(), was changed to be “DisabledState, IdleState, RunningState, or ErrorState”.
- **Void printStatuses():** This is a debugging function, I am leaving it in here if I ever want to modify this code and need something to debug. Basically it is responsible for printing most of the current values of variables in the code.
- **void updateCurrentState():** This code basically is responsible for the change of one state to another. It is basically an if-statement that has all of the possibilities of transition scenarios. So, it will keep checking if a certain condition has been met for the currentState to be switched into another state.
- **void checkStop():** checkStop() simply checks if the stop button was pressed. If the stop button was pressed, it updates the stopButton boolean to true, else, it stays false.
- **void checkRestart():** checkRestart() simple checks if the restart button was pressed. If the restart button was pressed, it updates the restartButton boolean to true, else it stays false.
- **void setStepperSpeed(int speed):** this function is responsible for setting the speed of the stepper motor, and deciding how fast it should turn.
- **void disabledState():** This function is responsible for executing all of the actions that the disabledState should do. It turns on the Yellow LED and turns off the rest, and it makes sure that the fan is off and not moving.
- **void idleState():** This function is responsible for executing all of the unique actions that the idleState should do. It turns on the green LED and it turns off the fan.
- **void runningState():** this function is responsible for executing all the unique actions that the runningState should do. It turns on the blue LED, turns on the fan, and turns on the stepper motor and causes it to close the vent.
- **void errorState():** this function is responsible for executing all of the unique actions of code that the errorState should do. It turns on the red LED, and it turns off the motors and displays the error to the LCD that the water level is too low.

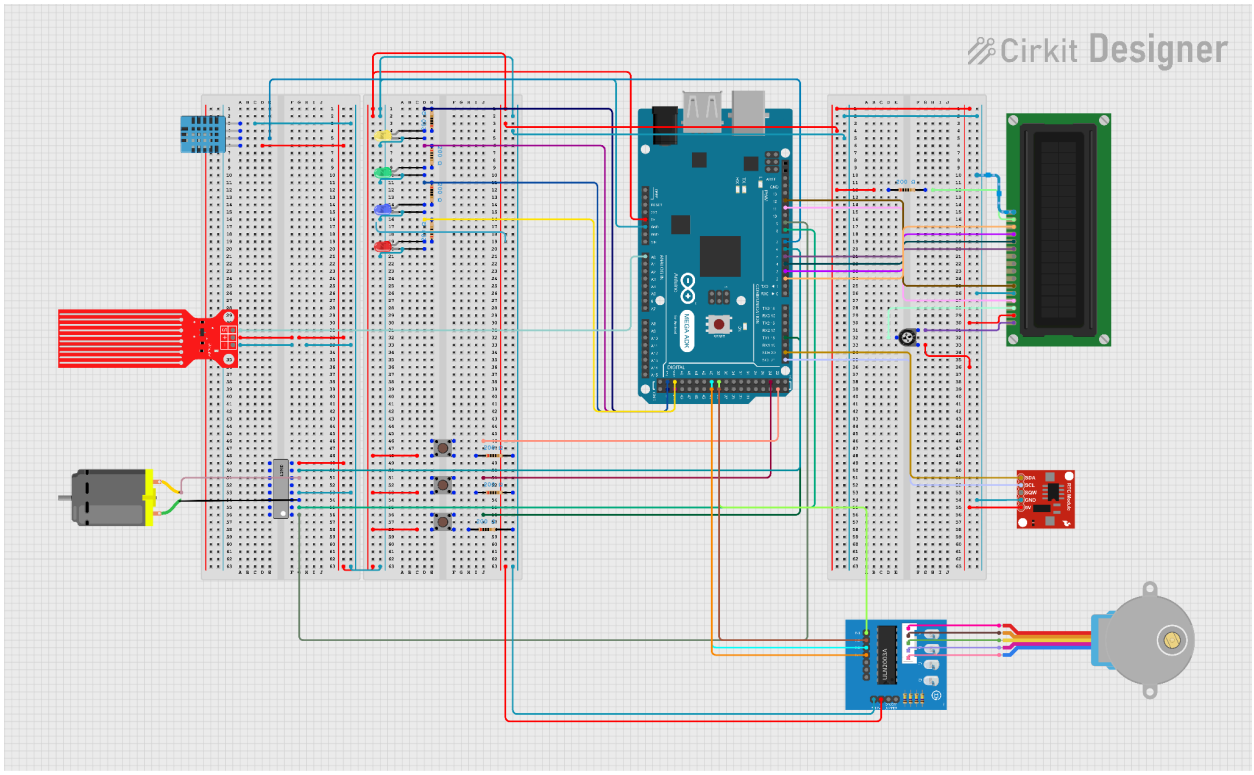
- **void allStates():** this function is responsible for doing the code that all of the states have in common besides the errorState. It updates the temperature, it displays the temperature, and it checks if the code should transition back to the disabledState.
- **void startButton():** This is the function that the ISR calls, it turns on the cooler.
- **void printDateAndTime():** This function prints the date and time into the serial monitor.
- **void directionControl(bool onOrOff):** this function is responsible for turning on or off the fan. It works if the bool sent in as a parameter is true or false. If true, it turns on the fan, if false, it turns off the fan.
- **void updateTemperature():** this function is responsible for getting up to date information on the current temperature of the DHT sensor and the water level sensor.
- **void displayTemperature():** this function is responsible for displaying the temperature information onto the LCD.
- **void displayError():** This function is responsible for displaying the error message that occurs when the water level is too low, (the errorState), onto the LCD.
- **void U0init(int U0baud):** This function is responsible for initializing the UART.
- **void U0putchar(unsigned char U0pdata):** this function is responsible for putting char into the serial monitor.
- **unsigned int adc_read(unsigned char adc_channel_num):** this function reads information from the sent in ADC pin.
- **void adc_init():** this function initializes the ADC.
- **void my_delay(unsigned int delay):** this function delays the program a specific amount of time.

GitHub Repository Link:
https://github.com/1101-Davis-Justin/CPE-301-Final-Project-Justin-Davis/blob/main/CPE301_Final_Exam_Justin_Davis.ino

Picture of Circuit:



Cooler Circuit Image:



Video Link:
https://drive.google.com/drive/folders/1aKrR9WdBsNyBe6G78gvW9aJU-YJjvjgs?usp=sharing

Sources Used:

https://lastminuteengineers.com/l293d-dc-motor-arduino-tutorial/
https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/
https://docs.arduino.cc/built-in-examples/digital/BlinkWithoutDelay/
https://www.electronicshub.org/wp-content/uploads/2021/01/Arduino-Mega-Pinout.jpg
https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf
https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/
https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/
https://howtomechatronics.com/tutorials/arduino/arduino-ds3231-real-time-clock-tutorial/#google_vignette