0. General Information

Student Name: Casey Merola, Keegan Smith, Jeffrey Nguyen
Student ID number:
Team Name/Number: The Capybaras
Team member names: Casey Merola, Keegan Smith, Jeffrey Nguyen
Date of completion: 4/21/2023
Demonstration method: zoom

1. Design:

1. Hardware Design

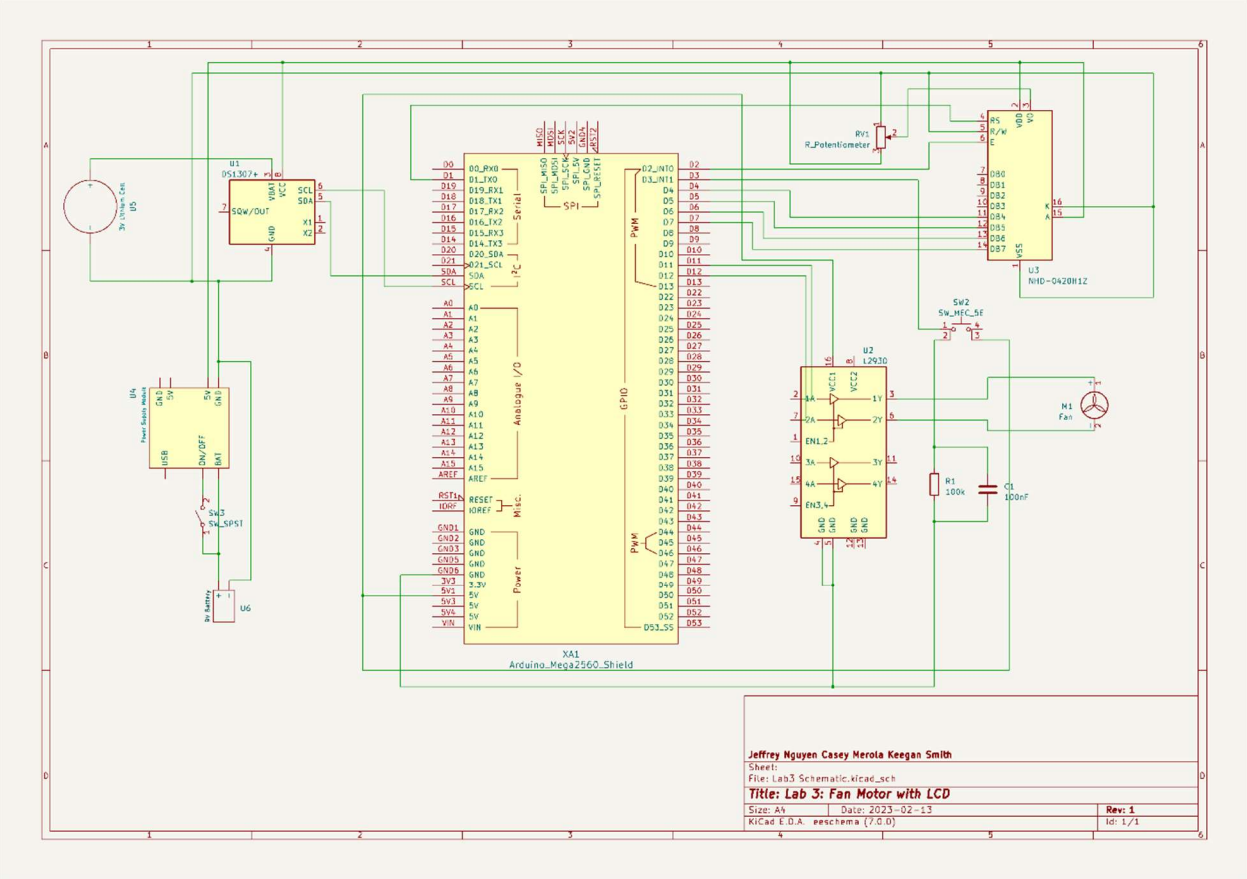

Figure 1. Schematic

Figure 1 shows the schematic that was used to produce the functioning circuit. The schematic shows the LCD display, that connects in a combination of parallel and serial data to interface properly with the Arduino. There is a potentiometer connected to Vo on the LCD, which controls the contrast of the display, and must be set properly to see the display. The RTC connects to the I2C port of the Arduino and communicates using the 2-wire configuration. The module also has a coin battery connected to it, which is there so the module can keep the current time loaded even when power is disconnected. The breadboard power supply module is also shown in the

diagram, which is used to supply the high current that the motor needs to run from. The motor circuitry is shown as well, which includes a motor driver that interfaces with the low current logic and the high current motor. When one side of the motor is high and the other is low, the motor spins in one direction, and when the sides switch polarity, the motor switches direction. This chip is necessary to get the direction change to occur when the button is pressed. The button that changes the motor direction is present and includes a pull-down resistor and a noise-filtering capacitor to help increase the fidelity of the signal reaching the Arduino.

2. Software Design

https://github.com/CaseyMerola/lab_3

The program works by utilizing the “LiquidCrystal.h” and “RTClib.h” header files to interface with the liquid crystal display (LCD) and the real time clock (RTC), as well as other built-in functions to interface with the motor controller IC chip. The added header files provide commands to begin communication and control the peripherals. The code is initialized by setting the pin modes as outputs for pins 11, 12, and to input for 18. Pins 11 and 12 are used to adjust the direction of the motor and pin 18 as the input for the button. Then, the LCD, RTC, and the software interrupt for the button linking it to the interrupt function, are initialized. After that, the “rtc.adjust()” command is used to set the real time clock’s time to that of the compiler, adjusting for any inaccuracies. However, this command could only be included once in the compilation if desired.

The main loop continually updates the LCD’s displayed time as well as motor speed and direction. When the button is pressed, the interrupt is triggered, executing a debounce delay, inverting the motor direction variable to 1 or 0 and setting the output pins to LOW. Once returned to the main loop body, the if/else statements are hit, changing the desired output pin to the PWM value, turning the motor that direction.

The motor speed function contains a switch and if/else statement. The motor speed is defined as part of the initializations, and is passed through the function call during the main loop. In the if/else statements, the direction printout is updated to “C” for clockwise if the direction variable is 0, or “CC” if the value is 1. In the switch statement, the speed value passed into the

function is used to enter one of four cases. Each case prints the specified speed of “FULL”, “3/4”, “1/2” or “OFF” as well as changing the PWM value to match printed speed.

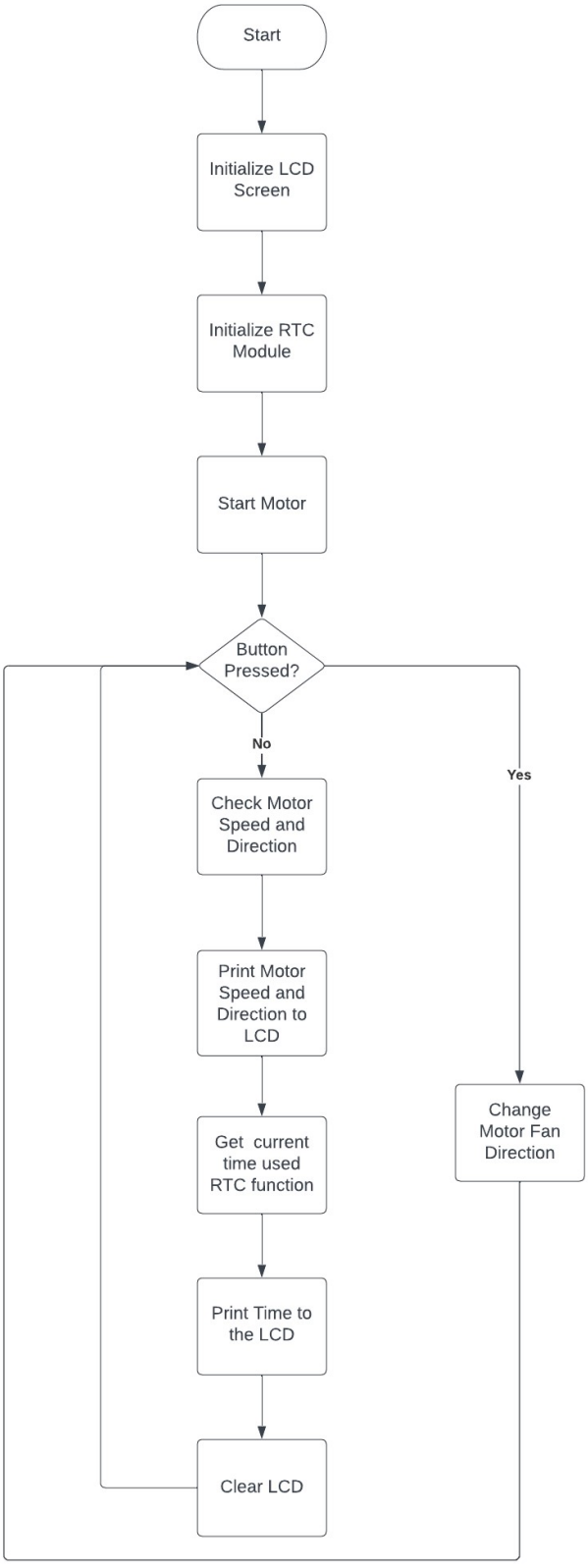

Figure 2. Software Flow Chart

3. Results

This lab performed as expected, with the LCD screen showing the time and the status of the DC motor. The RTC was initialized on compilation with the current time of the compiler and counted upwards every second. The transition from seconds to minutes and minutes to hours was observed and was found to be an accurate portrayal of the time. The motor spun at the correct speed as expected, and the 1/2 and 3/4 speeds are observed to be about half and three-quarters of the motor at full speed. The button presses accurately switched the direction of the motor, and the direction of the motor was updated properly on the LCD.

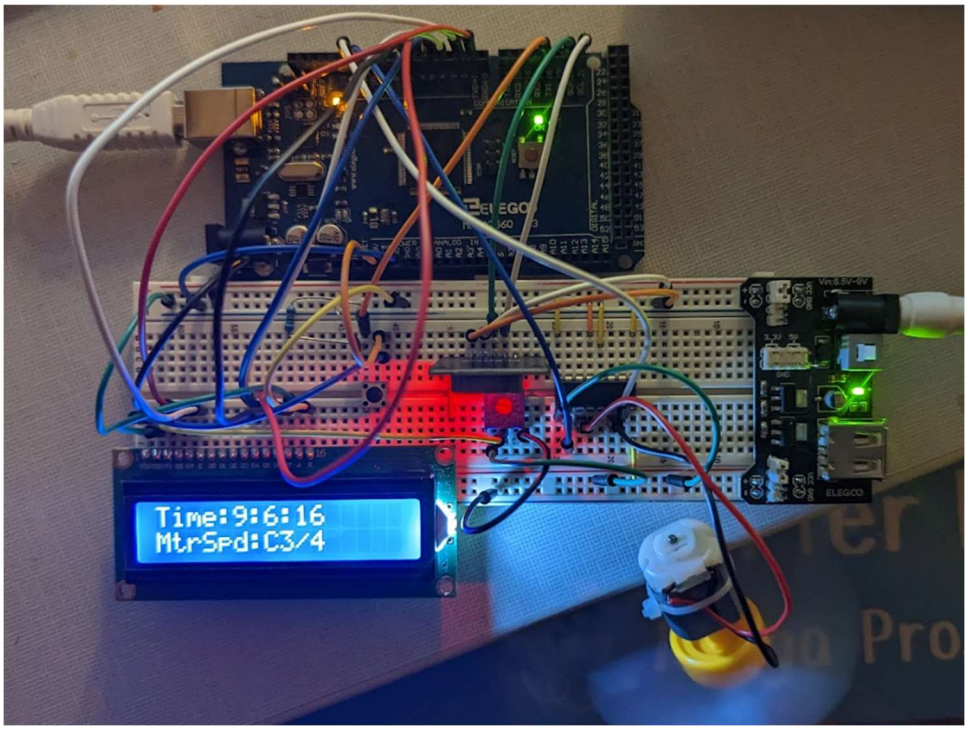

Figure 3. Circuit with clockwise motor

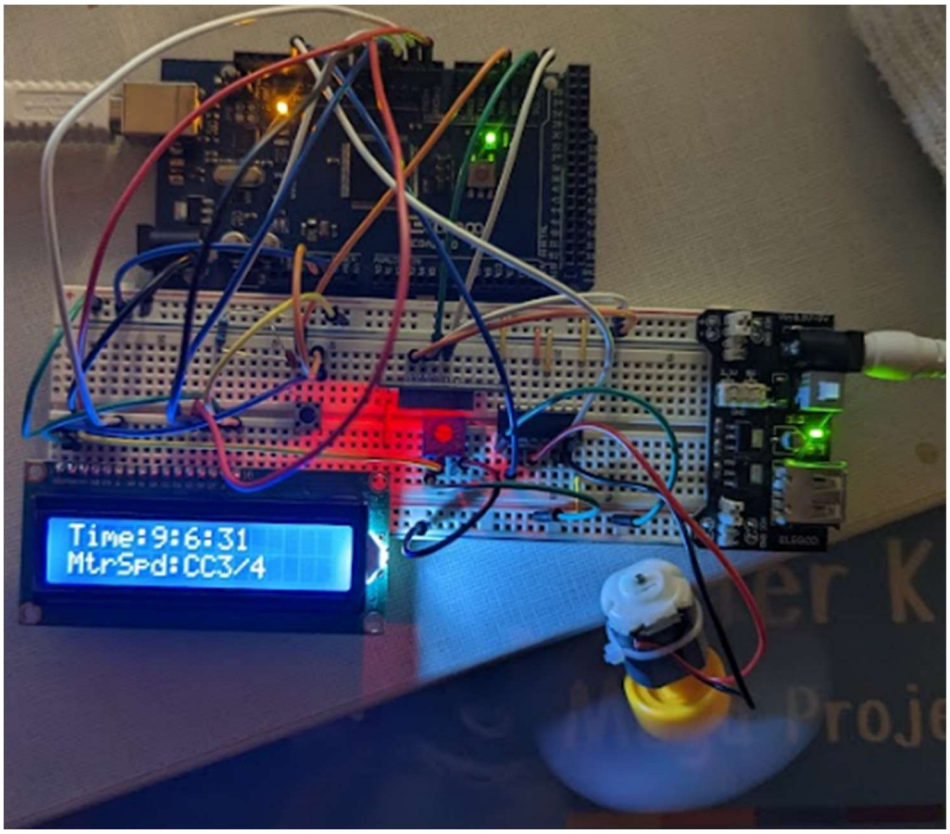

Figure 4. Circuit with counterclockwise motor

4. Problems Encountered and Solved:

One issue faced with the LCD was to have it not print garbage. During the interface with the RTC, a serial dialog was opened so that the time could be monitored. It turned out that the serial communication was interfering with the LCD and was causing the garbage to be printed. This was fixed by removing the serial dialog. Another instance of the LCD receiving interference was due to the motor. The motor introduced lots of noise into the circuit so when the interrupt button was pressed, the LCD would display garbage. This issue was fixed by placing a 100nF capacitor in parallel with the pull-down resistor.

The button introduced another problem into the circuit, which was that the noise from the motor was sometimes high enough voltage to trigger the interrupt, which caused a number of false triggers. This required a better button debouncing function to be created to ensure false triggers cannot happen as much as they were. The better debounce function worked in tandem with the capacitor that filters noise from the input of the Arduino.

Since the lab required the use of more current than can be provided by the Arduino, the breadboard power supply had to be used, which has an increased current supply. This is the first-time multiple power sources were used in this class, and previous circuit concepts had to be

reintroduced when combining different power supplying devices. For example, the grounds must be connected to each other so the logical inputs are at the same level, because otherwise a logic high can be any voltage when the reference voltage changes. Also, the 5V should not be connected with each other in case one voltage is slightly different. This could cause current to flow into the Arduino or the breadboard power supply, which can cause damage. Finally, the motor power rail was kept separate from the lcd and logic power to try to keep the back EMF from the motor from influencing the logic signals, which was seen initially with the circuit.

This lab included the most components out of the others for this class, which introduced a number of challenges into the programming. Since the RTC, LCD, motor, motor driver, and the button needed to interact with each other, it gets difficult to determine what components may not be working properly, and debugging became harder. For example, When the motor noise was causing issues with the LCD display, on initial inspection it seemed like the button interrupt was the cause of the issues, and too much energy was spent debugging a portion of the program that was working properly. It was only when the motor noise was inspected with an oscilloscope that the actual cause was found.

5. Personal Contribution to the Lab (Technical Details):

Keegan:

1) Interface with RTC
2) Interface with LCD
3) Debugged serial interference with LCD
4) Wrote functions to print the motor speed and direction to the LCD
5) Contributed to report

Jeff:

1) Created Schematic
2) Created Flow Chart
3) Designed and programmed Motor and motor Driver Circuit
4) Debugged/Tested the full circuit/program
5) Contributed to report

Casey:

1) Designed and programmed direction change of motor
2) Programmed and tested button debounce
3) Debugged program and tested circuitry
4) Contributed to report

6. Lessons Learnt:

In this lab, several things were learned throughout the process of programming and designing stages. We learned how to utilize datasheets to build our circuit, which helped us

understand how to the use the motor driver IC (L293D) to drive the motor. We also learned how to use an external power supply module to power the LCD and the RTC. In turn it helped us learn the importance of power distribution as there were issues running all the parts from the same power source. We learned how to debounce a button, especially during cases where the motor was producing noise so false triggers occurred more often. We learned how to set up the LCD and how it works to be able to set the screen to show whatever data we want and update the screen if data needed to be changed. We also learned how to utilize the RTC module alongside the LCD to display the current time to the LCD. We also encountered some issues regarding the LCD screen getting messed up during a button press, which helped us learn how to debug step by step to understand where the issue was occurring and how to prevent it from happening.