Lab 4: RT-OS and Final Project

Zichao Chen, 1925752 June 10, 2022
Samuel Kaminetzky, 1779745
Assignment: Lab 4: RT-OS and Final Project

Project idea:

Our project will be a fan that is controlled by a capacitive sensor, and a temperature and humidity sensor, and can display the speed of the fan and the current temperature of the room on an LCD 2-line display. If the temperature of the room is above a certain value, the fan will be turned on automatically. If the temperature is below a certain value, the fan will be turned off automatically. The capacitive sensor can measure the capacitance changes resulting from the changing distance between a person's hand and the sensor, then change the speed of the fan accordingly. If the capacitive sensor is touched, the fan will be turned on if it was off initially, and turned off if it was on initially. The motivation behind this project was to develop an automatic fan that can turn on or off automatically based on the temperature of the room, and can allow the user to control the speed of the fan without physically touching a button or a remote control.

How we will achieve the project criteria:

1. The fan is always waiting for an input, and the temperature sensor is always taking in inputs, this ensures that the project is performing reliable time and digital I/O functions.
2. The fan is always waiting for an input, which means that it needs to be running at a very fast rate to receive any inputs that might come in at any time. And we are retaining the default FreeRTOS "tick time" of 15ms. This makes sure that our project is operating with high speed and high CPU load.
3. We will be using the fan and its motor, a temperature sensor, a capacitive sensor, and an LCD display.
4. We will be using the temperature sensor to measure the temperature of the environment to determine whether the fan stays on or off. We will be using a capacitive sensor to control the speed of the fan, and also to turn the fan on or off.
5. For the user interface, we will be using a capacitive sensor to control the speed of the fan, and an LCD display to show the current speed of the fan and the temperature of the room.
6. Using FreeRTOS features in our project means that we would be assigning different stack sizes and priorities to each task. Tasks with higher priority would override those with lower priority. For instance, the temperature sensor has a higher priority than the capacitive sensor, meaning that once it is cold in the room, the fan is turned off and cannot be turned on by the capacitive sensor, unless the temperature of the room rises above the preset value again.

Introduction:

The objective of this lab was to use a real-time preemptive operating system, RT-OS, to perform a series of tasks. This included using the RT-OS to flash an LED on for 100ms and off for 200ms, to play “Close Encounters Theme” three times with pause of 1.5 seconds in between and then stop itself, and to measure the “wall-clock time” taken to call compute an FFT 5 times. Another objective of this lab was to integrate a solution-focused software and hardware project with novel sensors, displays, or actuators that is also implemented by RT-OS. The project should utilize and explore devices or interfaces that have not been used in previous labs.

Methods and Techniques:

Lab 4:

For 4.1, we installed the FreeRT-OS library and followed the example file to make the LED flashes and to use the thumbstick to generate a varying 0-5VDC positive voltage on an analog input pin and print the values between 0-1023 to the Serial Monitor when changing the thumbstick.

For 4.2, we followed the example code file to create Task RT-1, RT-2, and RT-3. For RT-1 and RT-2, we used the similar programs from the previous labs and adjusted them to RT-OS format. For RT-3, we studied the example code from the Arduino FFT library to learn how to compute an FFT, as well as studied the FreeRT-OS documentation for how to use a queue to send and receive data, then finally implemented the RT3p0, RT3p1, and RT-4 tasks that demonstrated the desired behaviors. We used two queues, one for holding the randomly generated doubles for computing the FFT, and another one for holding a value that signals the completion of the FFT computation. After that, we calculated and printed out the “wall-clock time” for computing 5 FFTs.

As suggested in the specification, we started the FFT with 64 samples and worked our way up. Increasing the number of samples means that we had to increase the stack size of each task so the program doesn’t break. The maximum number of samples we can get to without breaking is 256, and we showed that during the demo.

We also studied the FreeRTOS development and debugging tips to read the blinking signal on the board and identified whether there was a stack overflow or there was not enough memory, then we adjusted the stack size of each task based on that. For each task, we started with a high stack size to make sure it was working, then we gradually decreased the stack size and watched how small it can be without breaking. Minimizing the stack size for a task made sure we had more stack size available for other tasks.

We utilized an Arduino Mega board, a small speaker, an LED, and a Serial Monitor to display the printed results to construct the required components for lab 4.

Final project:

For the final project, since we were using a capacitive sensor, a temperature and humidity sensor, and a LCD display, we found the corresponding libraries and documentations online and studied them carefully to extract the information we needed to implement our code. We implemented each task of the project individually first, tested out each part, and then integrated all the parts for final testing. We also used the queues in RT-OS to send data from one component to another. For example, the temperature values read by the temperature sensor is sent to the LCD screen to display the value.

To debug our program, we created a couple of debug functions that can be inserted between lines of code to show whether a particular function is running as intended. When the program didn't have enough memory to run or froze after running for a few seconds, we adjusted the stack size and the priority of each task to see how the program responded. We also divided the entire project into many individual files, each one containing only a single task or component so that it was easier to debug and maintain the code.

We utilized an Arduino Mega board, a capacitive sensor made of an aluminum foil, a temperature and humidity sensor, a LCD display, and a fan to construct the required components for our project.

Experimental Results:

Part I: RTOS FFT Runtime Calculation

For part I, we initially set up RT-1 and RT-2 fairly quickly, simply substituting our round-robin implementations for the light blinking task, but with vTaskDelay() instead of delay(), and using millis() to keep track of the runtime of the song, allowing us to forgo a counter or delay with the song. We then used a queue to send data from a setup task RT-3p0 that would start the FFT tasks RT-3p1 and RT-4 send an array of random numbers to RT-3p1, and then halt itself, to only run once. We tracked the runtime of RT-4 by using a for loop in RT-3p1 which calculated the difference in real time between the start and end of 5 consecutive runs of RT-4. While we managed to get this working for 64,128, and 256 sample FFT computations, 512 was too memory intensive to make work. The LED blinked properly, and the song played smoothly, with long runtimes for 256 sample FFT computations, reaching 6 seconds.

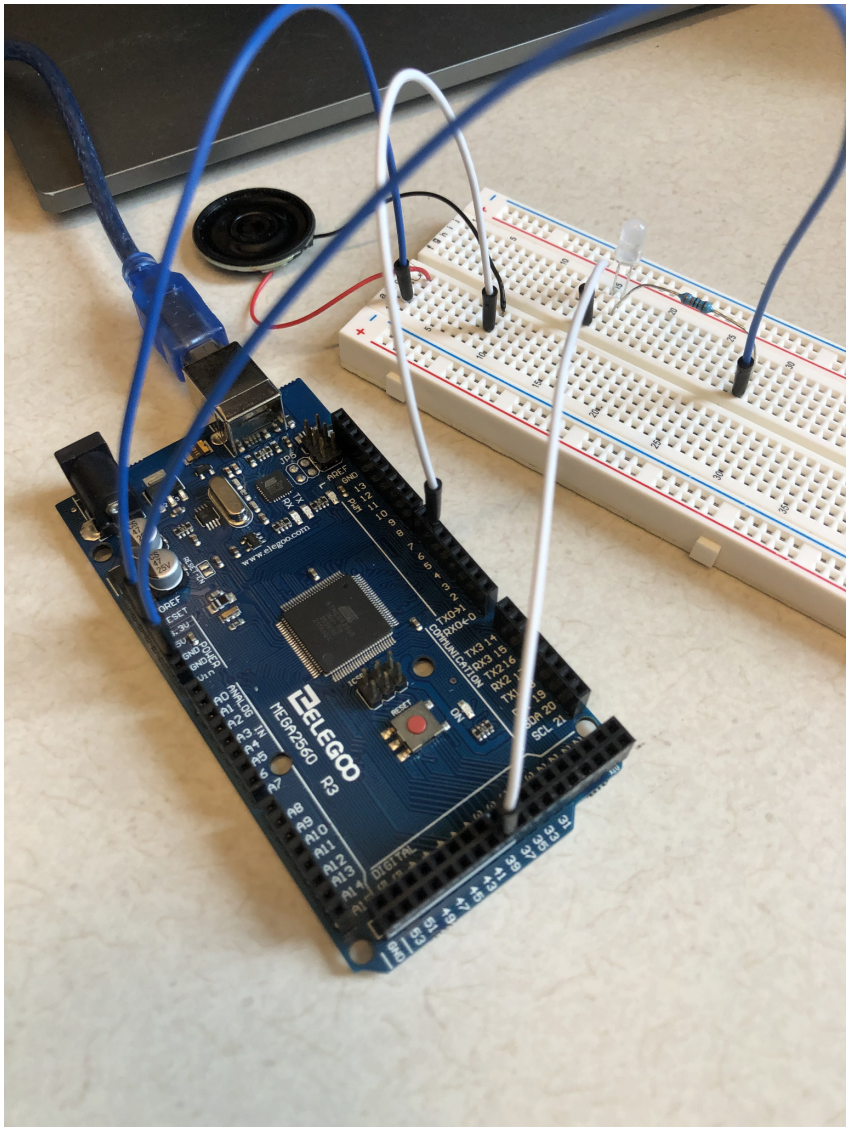

Fig 1: Arduino set up to demonstrate Part I of Lab 4, with LED, and Speaker

Part II: Final Project - Temperature and Capacitive Sensor Controlled Fan

For this project, we initially brainstormed along the lines of implementing a capacitive touch sensor controlled theremin (as one of our colleagues did, substituting a distance sensor instead), but elected to instead control a fan using capacitive touch and temperature sensors. After testing the individual components separately, we attempted to implement them all together in RTOS, using queues to shuttle data from task to task, allowing for tasks having different operating speeds, and eliminating the need for excessive global variables. While we did have some issues with timing and priority, mild trial and error made this problem less important. Instead, most problems came about from having so many wires needing access to similarly located board pins, resulting in frequent mismatches, pins not properly in contact with breadboard connections, and capacitive sensor calibration woes due to inconsistent grounding. We initially tried to run the fan off of the arduino's own power, but this proved to be too much power draw to generate consistent variable PWM performance from the fan, requiring our use of a power supply unit to ease the load on the arduino mega. This required use of an h-bridge motor driver to control the fan, resulting in even more wire complexity, and breadboard space. However, once these hardware problems were hunted down, and the capacitive sensor was tuned to have tolerance for different ground states, the system worked exactly as intended. The capacitive sensor worked like both a 360 degree distance sensor, and a button, turning the fan on and off, and modulating its speed. The temperature sensor outputted slow, but accurate and sensitive temperature readings, that turned the fan on and off consistently. The LCD screen smoothly displayed the current fan speed percentage, and temperature, validating the readings from the capacitive touch sensor and DHT sensor. The LED blinked unerringly, oblivious to the efforts of its neighboring devices.

(See next page for image of the project setup)

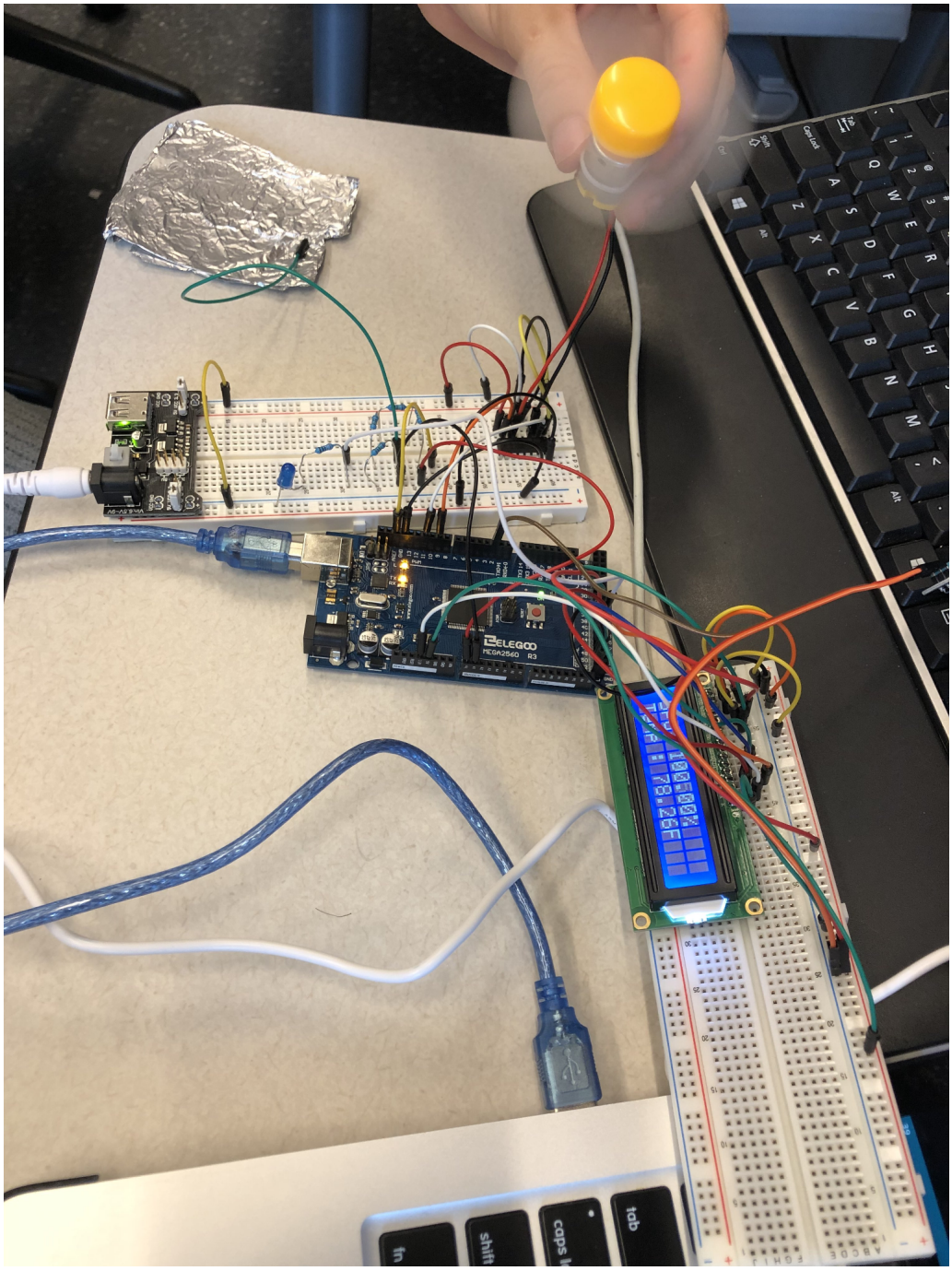

Fig 2: Arduino set up to demonstrate Part II of Lab 4, with DC fan motor, capacitive touch sensor (the metal foil), a DHT (Digital Temperature and Humidity) sensor for measuring temperature, an LED for demonstrating task RT-1, an h-bridge motor driver and power supply unit for powering the fan, and an LCD screen for displaying the temperature and fan speed.

Overall Performance Summary:

Lab 4:

We successfully demonstrated all the required tasks for lab 4.1 and 4.2. For lab 4.2 task RT-3, we were able to compute an FFT of 256 samples without breaking the program, while running task RT-1 and RT-2 at the same time.

Upon completion of lab 4, we learned how to work with an RT-OS to make a task run forever and how to delete an RT-OS when it is no longer needed. We also learned how to compute an FFT using the Arduino FFT library with various sample sizes and how to allocate the proper stack size for each task to make sure that the program doesn't break. We also gained experience working with queues in the RT-OS library, including sending and receiving data from one to the other. We had a firm understanding of how to allocate stack size and memory for each task in a large program and how to debug when the program doesn't work as expected due to memory limitations.

Project:

We successfully implemented our project with all the features working as intended, and successfully demonstrated the functionality of the project and answered all the questions from the TA.

Upon completion of the project, we gained experience working with various devices and actuators that we had not used in previous labs, such as a capacitive sensor, a temperature and humidity sensor, a LCD display, and a fan. Throughout this process, we learned how to search for documentations and Arduino libraries that allow us to operate these devices to achieve the functionality we intended. Similar to lab 4, the project was implemented using RT-OS, so we also learned how to allocate the proper stack size and priority for each task to make sure that the entire program can perform well. We also learned how to use queues to send and receive data from one device to another, so the data can be used for calculation or displaying purposes.

Our project had met all the criterias as listed in the specification. This included performing reliable time and digital I/O functions, and operating with high speed and high CPU load, since in our project all the components were actively reading, sending, and receiving data from either the sensors or other devices. In order for the fan or the LCD display to update their states quickly enough, they had to operate at a very high speed. We also used various devices to the Arduino board that we had not used in prior labs, as listed above. The defined problem that our project solved was to be able to control the spinning speed of a fan without having to physically touch a button or a remote control, and to allow the fan to turn itself on or off automatically based on the temperature of the room. When the user's hand moves closer or far away from the capacitive sensor but without touching it, the capacitance of the sensor changes, which changes the speed of the fan accordingly. Our project had used the capacitive sensor and the LCD 2-line display as the user-interfaces. We also used the FreeRTOS features in our project and eliminated any holdover code.

Teamwork Breakdown:

Both Zichao and Sam worked together on most functions in lab 4, pair programming to avoid implementation errors on either person's part. Both Zichao and Sam worked together to understand the documentations and the libraries of the FreeRTOS and the FFT, and discussed how to best implement the Task RT-1, RT-2, RT-3, and RT-4, and assign the proper task size for each task to be able to compute a larger samples of FFT.

For the project, Sam worked on the capacitive sensor and the LCD display components, as well as how they can be connected to handle the behavior of the fan motor. Zichao worked on the temperature sensor component to make the fan turn on and off at a certain temperature. Once all the components were implemented, Sam worked on integrating all the parts to work as a single unit and debugging any issues with the code.

Discussion and Conclusions:

Lab 4:

The part that was the most challenging for us was Task RT-3, in which we had to calculate the "wall-clock time" for computing FFTs and use queues to send and receive data. This was the first time we were using FreeRTOS and the Arduino FFT library so we were not familiar with the FreeRTOS Queues and FFT at all. We had to read the documentations and the example codes carefully to finally come to an understanding of what was being asked on the specification and how to implement it. Another challenging part in lab 4 was assigning the proper stack size and priority to each task to make sure there is enough memory to run all the tasks so the program doesn't break. This involved a tedious process of experimenting with different stack sizes and priorities, until we finally reached the values that can work for FFT with 256 samples.

But it was a proud moment when we finally understood how to use the queues, how to compute the FFTs, and how to calculate the running time, especially since we made heavy use of queues to implement our final project, utilizing the relaxed timing requirements of message sending to have tasks run at different speeds. Implementing the capacitive sensor was a trial of patience experimentation, but seeing the fan's speed be controlled with simple gestures was well worth it. We were very happy that we were able to assign the right amount of memory to each task to make the entire program run as intended. This also prepared us to develop a better understanding of how much stack size we should assign to each task when implementing the final project.

Project:

The part that was the most challenging for us when implementing the project was to integrate all the components and devices together to work as a single unit. We initially implemented and tested each component separately, but had trouble with them running harmoniously once they were put together. The signal light on the Arduino board indicated that it wasn't a memory issue, so we had to write some debug functions and insert them in each of the programs to print out the results and debug the issue. Another challenging part of the project was to make the fan change speed depending on the distance between the user's hand and the capacitive sensor, and also to make sure that when the user's hand touches the sensor, the fan would toggle from on to off. The fan was not functioning ideally at the beginning, which had to do with some issues relating to the calibration of the value returned from the capacitive sensor, since the fan's operation ended up interfering with the signal from the capacitive sensor, resulting in us requiring a very tolerant set of capacitance thresholds. This also required a tedious process of constantly trying and changing the formula that calibrates the sensor readings, lest the fan never run at full speed. An additional challenge we encountered was that the LCD 2-line display was not working properly. Initially we thought it was an issue with our code, but after a long time of debugging we realized that it was just a wiring issue (much to our chagrin).

The most proud moment of the project was the moment that everything worked perfectly for the first time, and every component did its job as intended. We had fixed all the issues and now the fan was responding to all the commands properly. We were happy that we finally had a finished product with all the functionalities we wanted. We were especially proud of the capacitive sensor we implemented, it was something original, cool, and has the possibility to be implemented in other practical projects.

Something that we learned in addition to the learning objectives while doing lab 4 and the project was the importance of reading library documentations and example code. It was the first time we worked with FreeRTOS and FFT libraries so we had to learn how to utilize online resources to gain an understanding of how they work. Throughout the final project, we learned so much about the devices and actuators that we had not used in prior labs, like the capacitive sensor, the temperature sensor, and the LCD display, as well as their corresponding libraries and documentations. The freedom of this project allowed us to freely explore the devices in our lab kit and add our imagination and creativity to make a project that we took pride in.

Signed Statement of Individual Contribution:

I,  hereby state that I have individually contributed to this assignment, and I recognize my partner's individual contribution to this assignment.

I, Zichao Chen, hereby state that I have individually contributed to this assignment, and I recognize my partner's individual contribution to this assignment.