

ECE 4220—Real Time Embedded Computing Final Project Report

Abstract

The goal of my project was to implement a remote thermostat system with a dual temperature/humidity sensor and the TS-7250 board. The intended idea using the concepts that was taught throughout the semester in an attempt to achieve this system. The general framework is present for. However, the complete desired system was not accomplished.

Introduction

As previously mentioned, the intent of this project was to create a remote thermostat system. Using a client, someone could access a server which keep track of the temperature and humidity level in a room. Additionally, using the client, someone could set the temperatures in which the air conditioner or heater turns on, as well the relative humidity level in which a humidifier turns on. Lastly, a data log of the temperature, humidity, and time stamp is written. When appliance LEDs are turned on or off, it is also documented in the data log. Essentially, the system created has rudimentary smart home applications, giving a realistic, practical example of how to apply the topics covered in this semester.

Background

Although the general idea of this project has been implemented many times by various individuals, I did not investigate the methods others utilized. I used the knowledge gained the labs. As an example, the rate for new data to be generated by the sensor that I chose is two seconds. Resultantly, a real time task with a period of two seconds was created to consistently gather.

Proposed method / System description / Implementation

Figure 1 is roughly how I implemented my project. The center blocks starting with the sensor read is a kernel module and the remaining blocks on the left and right are a user space server program.

Figure Implementation Diagram

The sensor chosen for this project was the RHT03. The sensor required a minimum voltage of 3.3V, with a nominal voltage at 5V, and its output was digital. With this in mind, the LCD port of the TS-7250 was used as digital I/O lines primarily because there is also a 5V source on the LCD port that could be used to power the sensor. The line from the LCD port are accessed using Port A of the EP9301 Processor. The sensor was connected to LCD_0 and three LEDs to LCD_1, LCD_2, and LCD_4. The LEDs are representative of actual appliances that would be used for such a system. They signify an air conditioner, a furnace/heater, and a humidifier.

Figure 2 is the actual connections to the TS-7250 and figure 3 are the components themselves.

Figure Hardware Connections to TS-7250

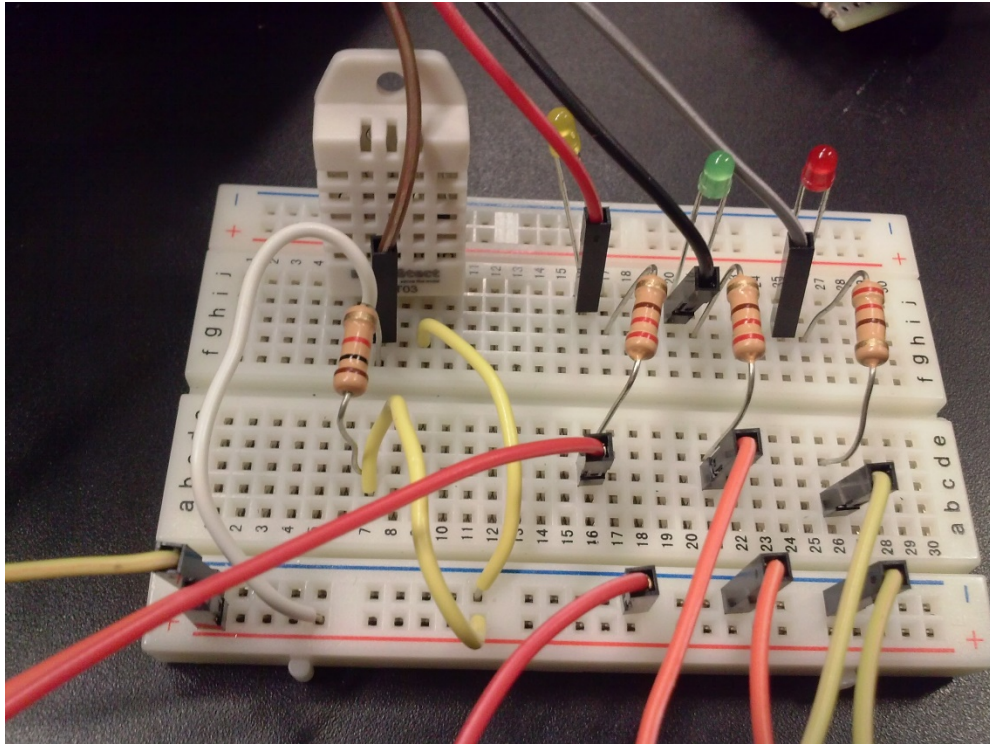


Figure RHT03 Sensor and LEDs

The data from the RHT03 sensor is transmitted in 40 bits. Although the memory mapping was correct, the LEDs connected on this port could be controlled, I was unable to properly retrieve the sensor data. As a result, simulated data was generated and sent through a FIFO to a user space program. Additionally, on the kernel module are interrupt handlers. The interrupt on Port B is set and the handler toggles one of the three LEDs. Additionally, there is a FIFO in that handler to tell the user space program that one of the three LED “appliances” has been turned on or off. The other interrupt is a software interrupt that will trigger when the user space program wants to toggle the LED.

The user space program is a server and has several threads in order to achieve multiple tasks simultaneously. The main thread receives the commands from the client. The commands gets the current temperature, the current humidity, sets the AC turn on temperature, sets the heat turn on temperature, sets the humidifier turn on value, and displays the currents settings. The server using UDP and only broadcasts the reply to the commands to the client that sent them. A original client was not used, but the provided client from Lab 6 was used.

One of the threads reads from FIFO the data from sensor and stores the value in a circular buffer. Furthermore, a time stamp from when the data is received in stored along with the sensor data. The thread then signals a parse data thread with a semaphore. The semaphore prevents the

parsing thread from parsing data that has not been read yet. The circular buffer allows the read FIFO thread to continuously read without waiting on the parsing thread or without losing data.

The parsing thread calculates the temperature and humidity values. The current data, which is a structure that has the newest temperature or humidity, is set in this thread. Another thread that writes the data log is signaled here. Once again a circular buffer and semaphore is used for the same reasons as the previous thread. A thread controls with the software interrupt is triggered. When the current values are either higher or lower than the set turn on values.

Experiments and Results

All of the aspects of the system does not operation as desired. The hardware and software interrupts generally work. Occasionally, the logic behind when to turn on and off the LEDs seems flawed as the lights something do not turn off during conditions in which they should.

Discussion and Conclusions

The architecture of the envisioned system is present. If I utilized the time better, the project would work better, including actually using the RHT03 sensor instead of using simulated data. Additionally, the server should use TCP and have an authorization system to ensure that only certain individuals would be about to access and modify components of the system. The project was a good example of using an embedded system for monitoring data and activating parts under certain conditions.