CSE 321 Project 3 Report:

Temperature and humidity alarm system using the Nucleo embedded platform

Prepared By:           Chaktim Wong, Senior

                                                   Department of Computer Science and Engineering

                                                   School of Engineering and Applied Sciences

                                                   University at Buffalo

**Table of Contents**

**Introduction**

This project involves implementing an alarm system that alerts the user with a vibration motor when the nearby temperature or humidity exceeds the chosen thresholds. The project is implemented using the Nucleo embedded platform, and the focus of this project is on the creation of a real-time embedded system that can be used to help solve a problem. This system will utilize important embedded operating systems concepts including the principles of RTOS.

The alarm system implemented in this project can help users in a variety of ways. One way that this system can be used is monitoring the temperature and humidity levels inside a PC. The user can install the system inside their PC and have it alert the user when the chosen thresholds have been exceeded. The thresholds can be adjusted within the code. Another use for this system is monitoring the temperature and humidity of a room inside a house. Both methods can be used to ensure the user’s safety by alerting the user when the temperature of the area surrounding the system exceeds the chosen thresholds.

**Project Requirements**

Internal Elements

* Watchdog timer
* Synchronization technique using an event queue
* Bitwise driver configuration (vibration motor)
* Critical Section protection is required for entire implementation
* Uses multiple threads
* Button interrupt

Constraints

* Temperature can be in degrees Fahrenheit or degrees Celsius.
* The user can press BUTTON1 on the Nucleo to change the unit of measurement for temperature (Fahrenheit or Celsius).
* Humidity can go from zero to one hundred percent.
* The threshold at which the vibrating motor turns must be set in degrees Fahrenheit for temperature and a percentage for humidity.

**Overview of Specifications and Features**

Specifications

* The LCD will display “Temp(F): “ or “Temp(C): “ followed by the current temperature on the first line.
* The LCD will display “Humidity: ” followed by the current humidity on the second line.
* Whenever the surrounding temperature or humidity changes, the LCD is updated with the new information.
* The vibrating motor turns on when the temperature or humidity exceeds the chosen threshold, and turns off otherwise.
* Must run “forever”.

Features

* Alarm system using a vibration motor
* LCD displays the current temperature and humidity of the surrounding area
* Temperature can be displayed in either Fahrenheit or Celsius by the press of a button.
* Vibrating motor is quiet but very noticeable.
* All-in-one system
* Uses the Nucleo Embedded Platform
* Affordable
* Safe
* Does require a lot of electricity to run

**Explanation of Watchdog Element**

The watchdog element protects the system from unexpected errors which can stop the system from working entirely. It works through a timer that is repeatedly reset after a set amount of instructions is complete. If the timer is not reset in time and reaches zero – because of a system error for example – the entire system is reinitialized and continues running.

In this project, the watchdog element was incorporated in the main loop where a watchdog was initialized with a timer of 5 seconds. Every time the LCD display update function runs, the watchdog is “fed” or reset. If the system somehow goes haywire, the watchdog will not be fed and after 5 seconds have passed will reset the system.

**Explanation of Synchronization Technique**

The synchronization technique used in this project is the event queue. Each time a thread is ran, it adds an event to the queue. The queue then processes these events in a first in first out approach, thus ensuring that events are ran in the order they are called.

This system incorporates an event queue through the main loop. The main loop initializes all the necessary parts of the system including all the threads before dispatching forever to handle events in the event queue. Each time a thread is ran, it adds an event to the queue and the main loop will run them. This way, functions and data variables are accessed in an orderly fashion.

**Explanation of Bitwise Driver Control**

The vibration motor in this system is manipulated using bitwise driver control. The wire that provides power to the motor is controlled by a pin set to output in the GPIO register. The clock associated with the port this pin is in, is also turned on. The motor can then be controlled by setting the output to this pin to 1 to turn the motor on and 0 to turn it off.

**Explanation of Critical Section Protection**

The critical data variables in this system are protected through an event queue. Each event is essentially one function. Since threads can only add events to the event queue and the event queue can only run one event at a time, all the data variables will be protected from being inappropriately accessed by multiple threads at the same time.

**Explanation of Threads/Tasks**

In this system, there are four threads: the main thread, the thread that runs the sensor, the thread that updates the LCD display, and the thread which checks if the threshold has been exceeded or not. The main thread runs first, and then the other three threads start when the main thread calls them. These threads then work in tandem to run the system through an event queue.

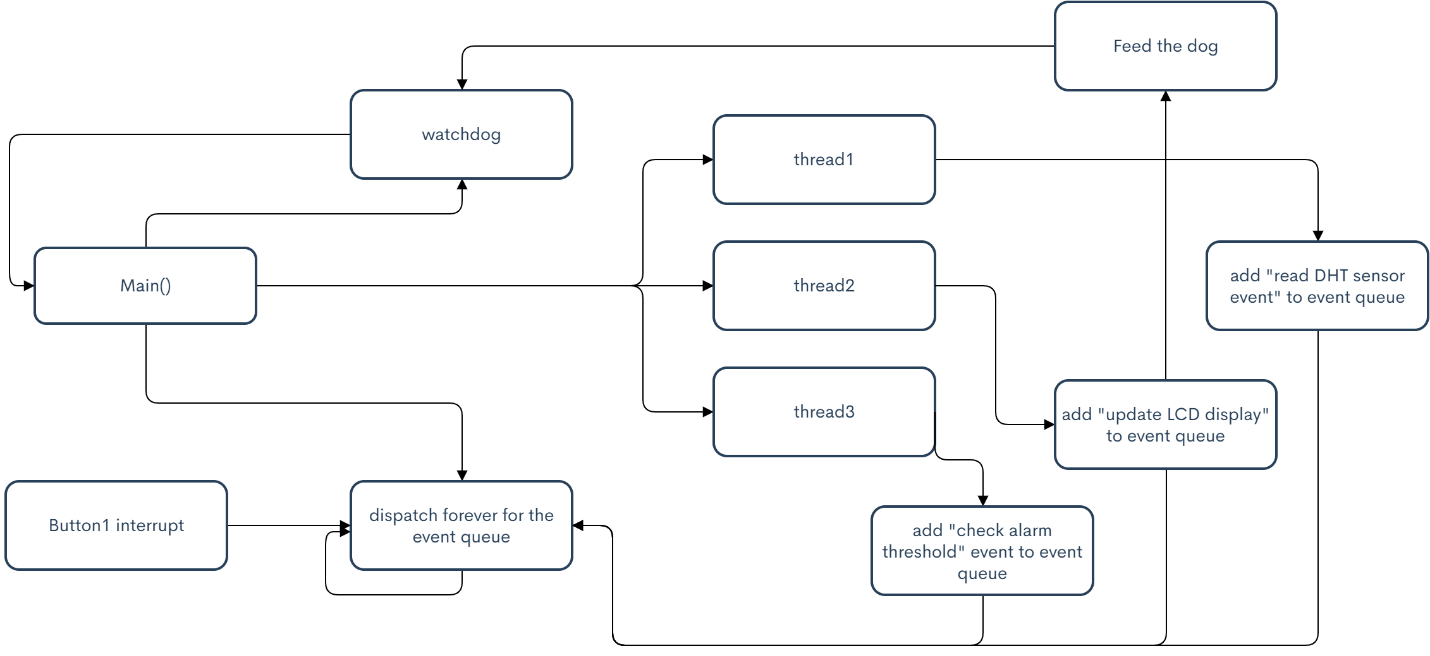
**Explanation of Interrupt**

The button (1) on the Nucleo can interrupt the system and change the temperature unit. It works through the Interrupt Service Routine. The callback function adds an event to the event queue instead of directly changing the data variable so that there would be no conflict if a function were to be accessing the data variable at the same time the interrupt was called.

**Solution Development**

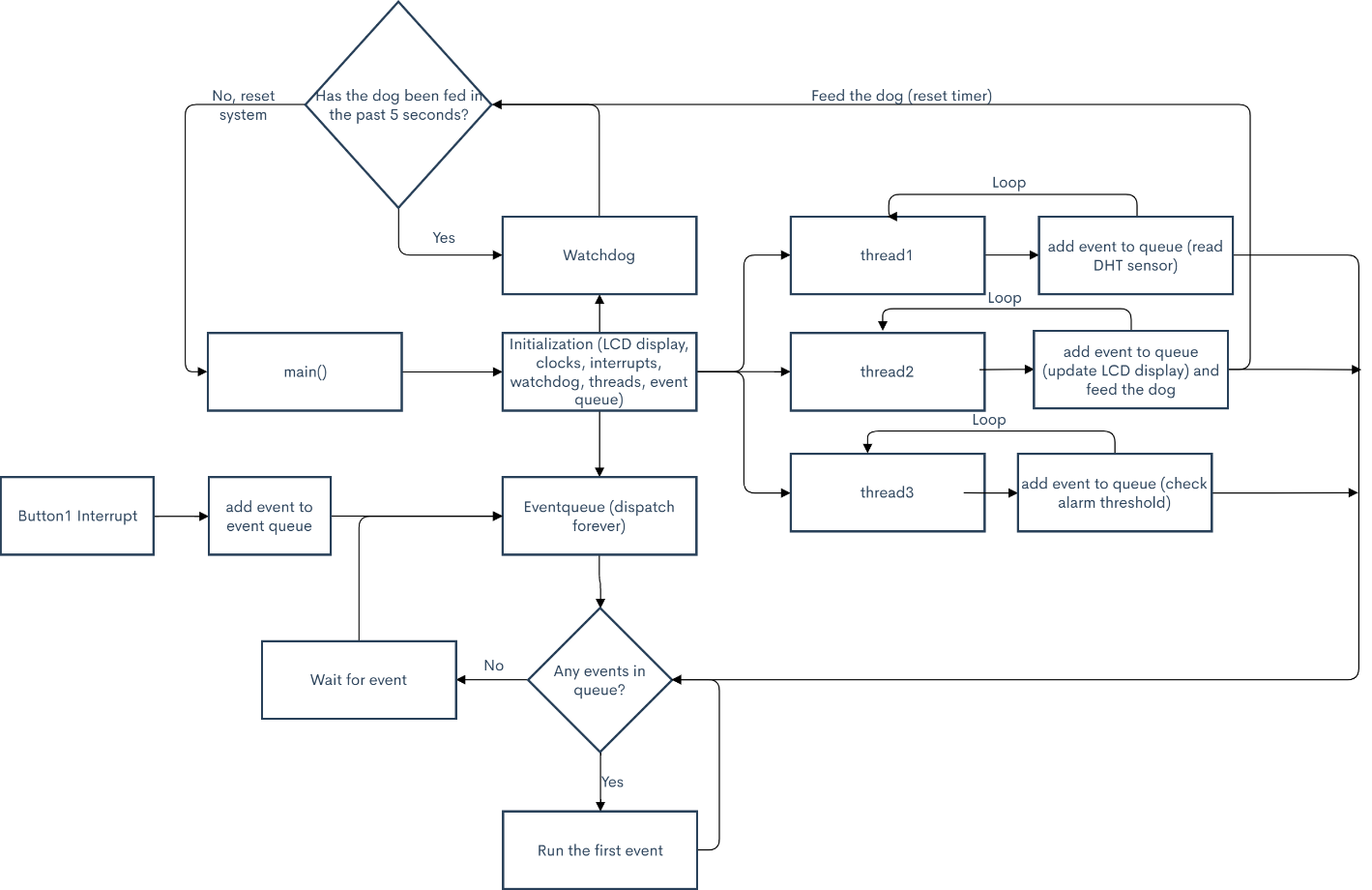
The solution was developed through a step-by-step process starting with testing the peripherals themselves then moving on to connecting them in a system and then finally adding threads to manage each peripheral. This step-by-step process allows the ability to quickly identify the source of any errors and quickly fix them, creating a smooth development process. At least this is the goal of such a process, and of course, this does not happen all the time. Overall, however, the development process for this system was relatively smooth since each function was developed incrementally.

**Block Diagram**

****

**Diagram**

* Flow Chart



**BOM**

Materials needed for this project include the following:

- DHT11 (Temperature and humidity sensor)

- Vibration motor (for the alarm)

- Solderless Breadboard (to form connections)

- Jumper Wires (at least 7 for convenience)

- A computer to program the Nucleo

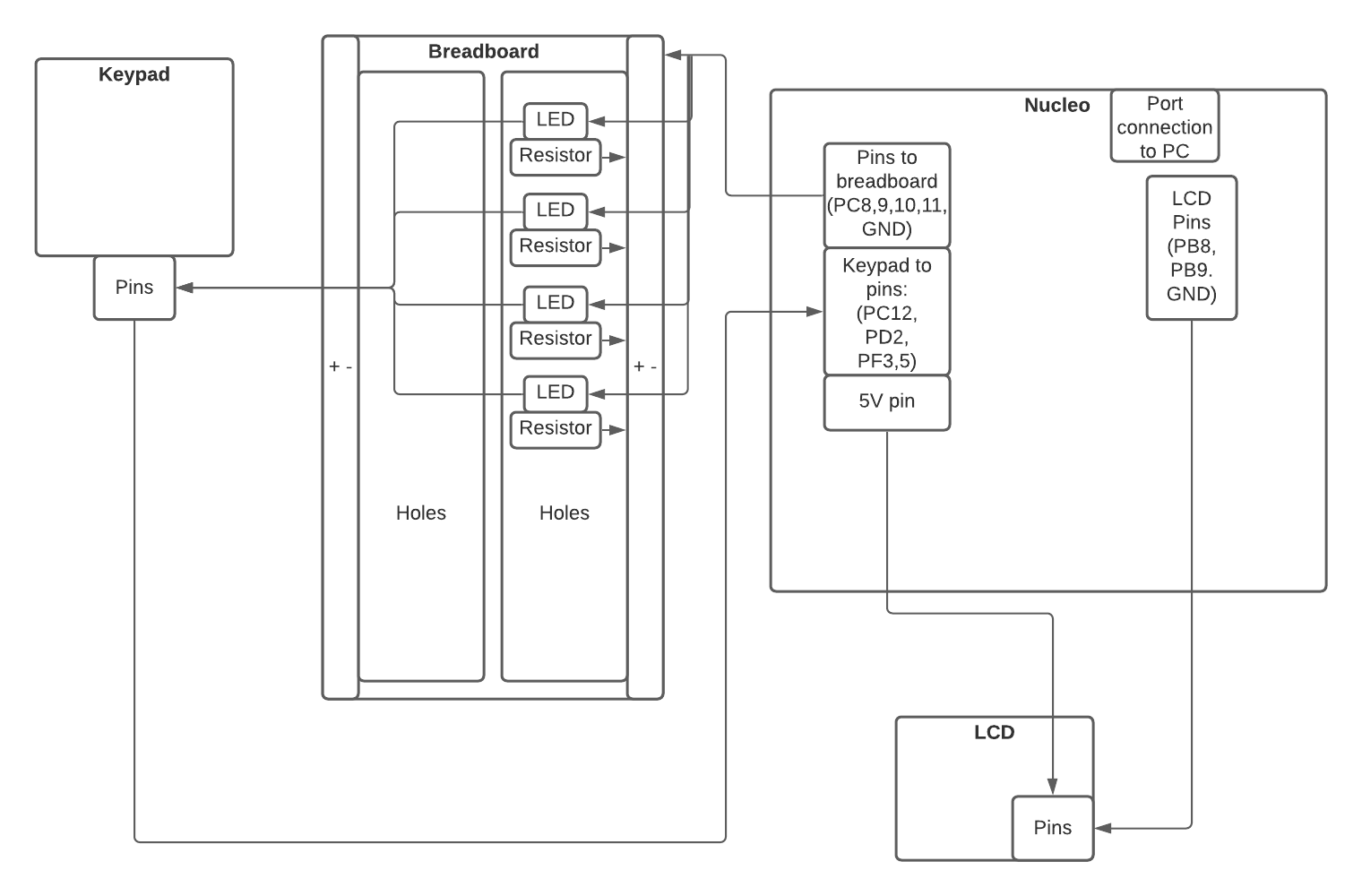
- USB a to Micro USB B cable (connects the Nucleo to the computer)

- 16x2 LCDs with I2C for Arduinos and Raspberry PI, 1602 or 1802 model with the I2C element soldered on

- Electricity (to run the computer and the nucleo)

**User Instructions**

**Schematic**



**Instructions to Build**

**Instructions to Use**

**Test Plan**

* Check Display
  + Verify that the display shows “Timer initialized” on startup.
* Timer Setup
  + Verify that LCD displays “Set Timer: m:ss” in timer setup mode.
  + Verify that values can only be inputted into the timer after pressing button “D”.
  + Verify that every button press that inputs a value lights up an LED.
  + Verify that the inputted values are displayed on the LCD.
    - Make sure that the value cannot exceed 9 min and 59 secs by pressing “1”, “0”, “0”, “0”.
  + Verify that pressing D again resets the values displayed on LCD.
* Timer Start
  + Verify that pressing button “A” starts the timer (numbers on LCD begin decreasing)
  + Verify that for every second that passes, the timer also decreases by 1 second.
    - Make sure that minutes also decreases by 1 after seconds hit 0.
* Timer Pause/Stop
  + Check that pressing button “B” stops the timer.
  + After a pause has been initiated (from pressing “B”), verify that pressing “A” starts the timer again from where it left off when it paused.
  + During the pause state, verify that pressing button “D” resets the timer, and allows for new values to for the timer.
* End of timer
  + Verify that when the timer hits 0 mins and 0 secs, all the LEDs are lit and the LCD displays “Times Up”.
  + Verify that timer can be set to a new value from this state.
* Forever Loop
  + Run the timer multiple times by setting the timer to a low value such as 5 seconds, and verify that the timer works each time.

**Outcome of Implementation**

The project was completed after many hours of work. It was a bittersweet experience overall. There were many stressful moments, but also a few dopamine filled moments when a bug has been removed.

Some roadblocks that appeared included setting up the keypad and getting rid of the bounce that occurs from each keypress. Unfortunately, bounce has not been completely resolved (it may not even be a bounce issue); currently, there is a chance that a keypress may be accepted twice even with interrupt delay and polling delay.

The rest of the project was completed relatively quickly, including the LCD configuration.

**Future Considerations**

Identification of shortfalls

General Improvement

**Recommendations for Improvement**

A major point of potential improvement is dealing with the occasional double input from one keypress. This issue may or may not be related to bounce. Currently, there are no leads to why it occurs.

Another point where improvement can be made is adding additional functionality, starting with the keypad button “C”. Currently, no functionality exists for this button.

A small improvement can be made in wire length and wire placement. The LCD and keypad have a very limited range of movement and cannot be moved very from the breadboard and Nucleo. There are limitations to this, but a better setup may improve user experience.

The LEDS can be configured to a pattern when the timer ends to improve the visibility in the alarm feature of the timer.

Finally, the materials themselves can also be improved. A better keypad may stop the double input, but this is untested. A more integrated or modular setup may be possible with different materials. The LCD can be changed to display more data or to give the setup a better overall look. The durability of the setup may be improved with better wires, resistors, and LEDs.

Although the core features of this project has been completed, there are still many points of possible improvements. However, for this class, most of them are not that important, except for the double input issue that has yet to be resolved.