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 not 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.

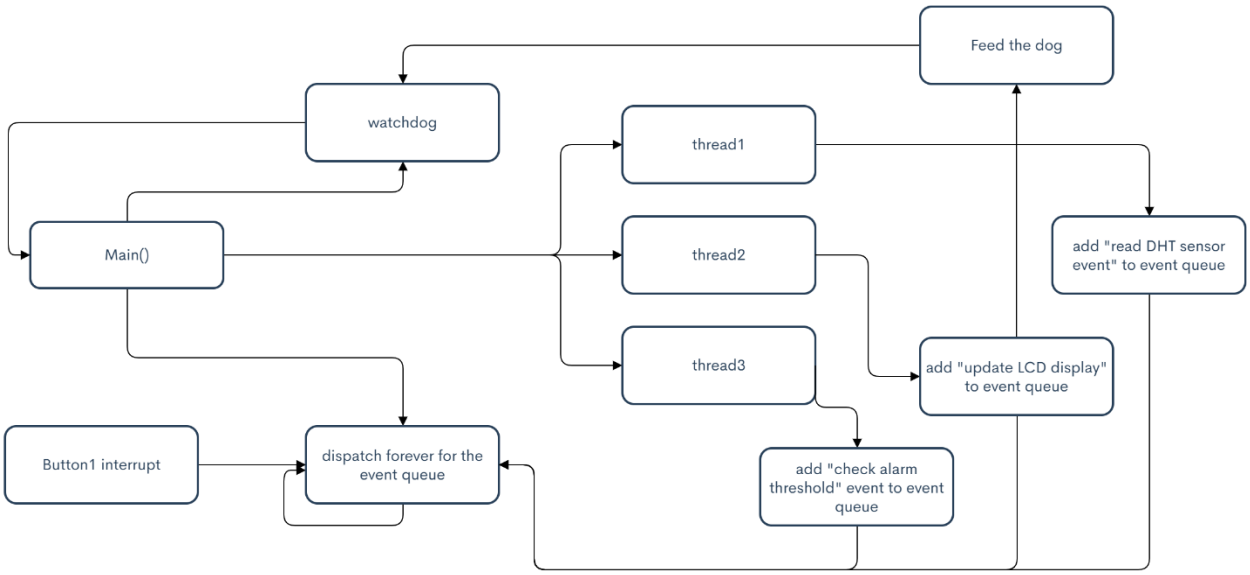
Explanation of Interrupt

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

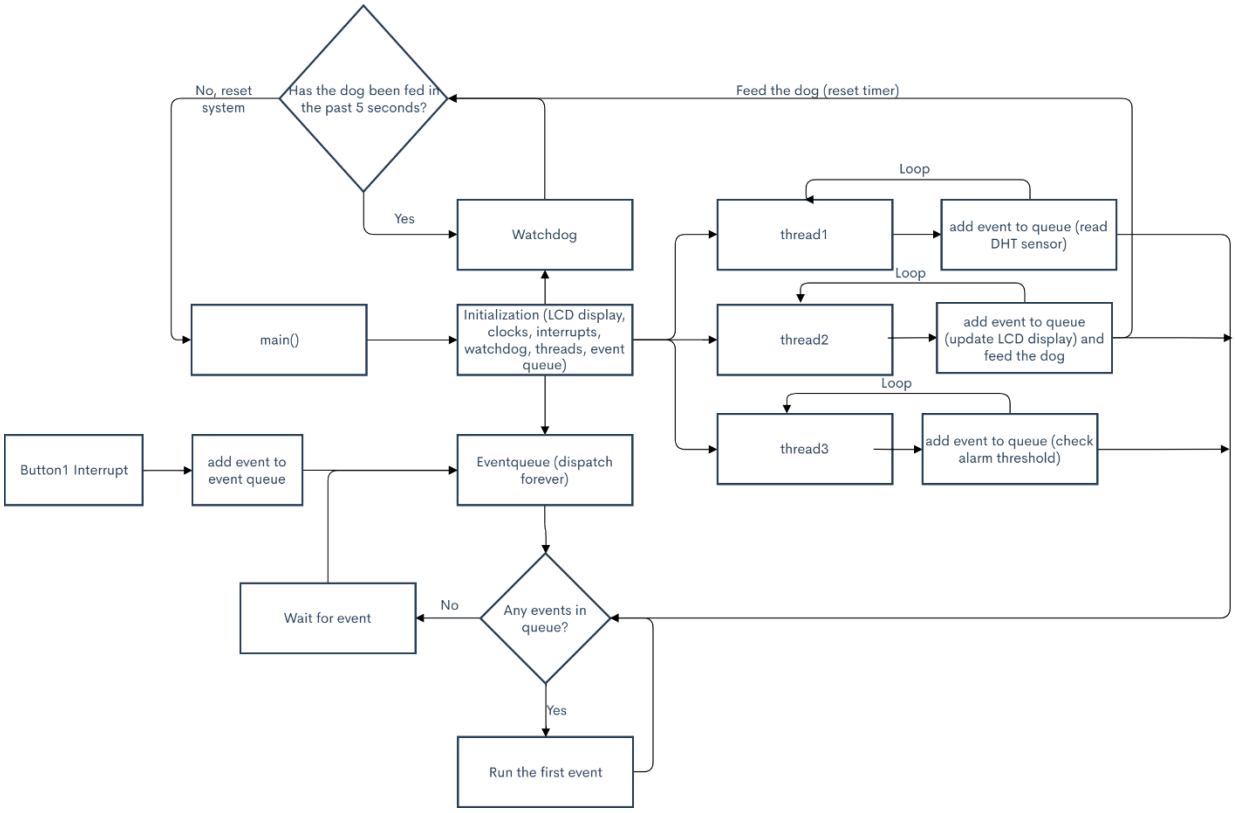
The solution was developed through a step-by-step process starting with testing the peripherals themselves then moving on to connecting them in a system and then finally adding threads to manage each peripheral. This step-by-step process allows the ability to quickly identify the source of any errors 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)

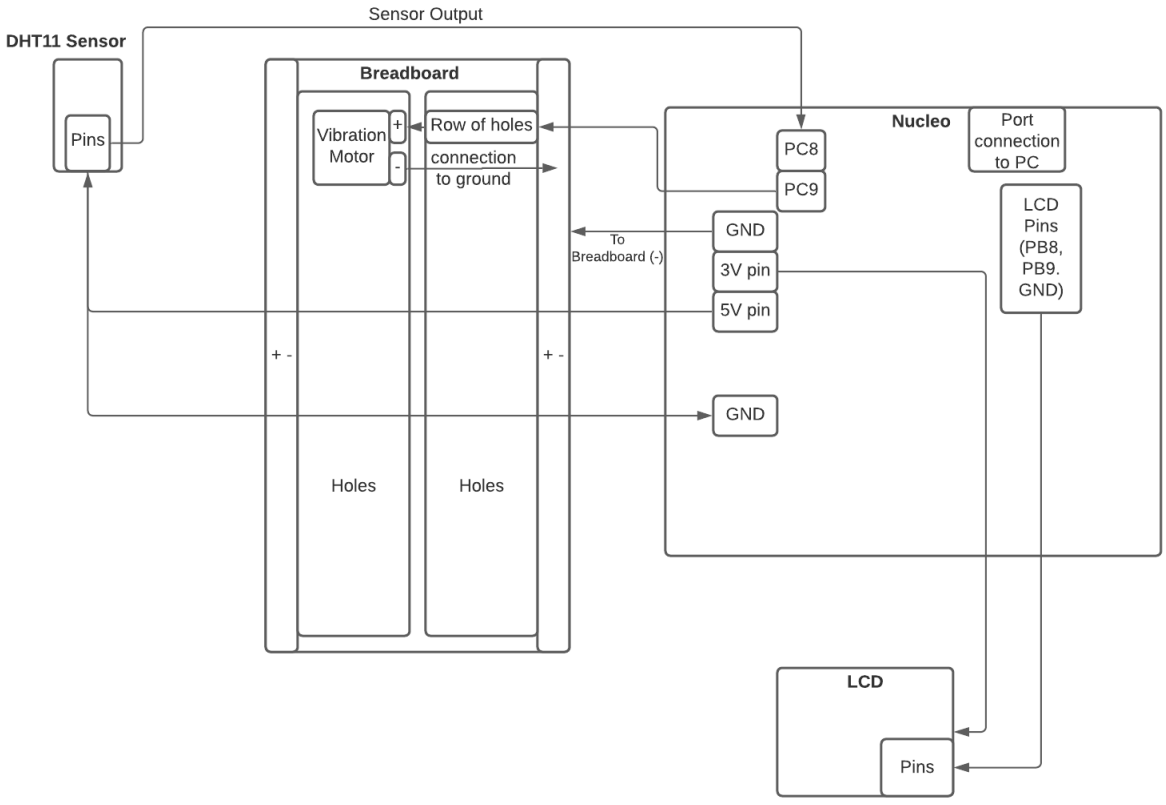
- 16x2 LCDs with I2C for Arduinos and Raspberry PI, 1602 or 1802 model with the I2C element soldered on

- Electricity (to run the computer and the Nucleo)

User Instructions

1) Place the Nucleo and its connected components either in a designated room or PC.
2) Connect the Nucleo to a PC using the USB a to Micro USB B cable.
3) In the program code on Mbed Studio, edit the MAX_TEMP and MAX_HUMIDITY constants on line 51 and 52 to a desired threshold value. If these values are exceeded, the vibration motor will turn on.
4) Now press the run button on the left side of the screen (play button) and wait for the code to finish loading.
5) Once the Nucleo is ready, the surrounding temperature and humidity level will display on the LCD.
6) The temperature unit displayed on the LCD can be swapped to Celsius or Fahrenheit by pressing on BUTTON1 on the Nucleo.
7) Unplug the Nucleo from the PC to turn off the system (if desired).

Schematic



Instructions to Build

1) Connect the LCD ports to PB_8 (SCL), PB_9 (SDA), 3V (VCC), and GND (GND) on the Nucleo.
2) Connect the DHT11 sensor ports to PC_8 (out), 5V (+), and GND (-) on the Nucleo.
3) Connect the breadboard's negative column to a GND pin on the Nucleo.
4) Connect PC_9 on the Nucleo to a row on the breadboard.
5) Connect the positive pin of the vibration motor to a hole in the same row as PC_9 on the breadboard.
6) Connect the negative pin of the vibration motor to the negative column of the breadboard which is connected to a GND pin on the Nucleo.
7) Remember to secure the vibration motor using tape or stick the adhesive side of the motor to the breadboard so that the wires don't pull out when it vibrates.

Instructions to Use

After downloading the program code to the Nucleo using a USB a to Micro USB B cable (refer to the User Instructions section for more information), the system is ready to be used.

The LCD display will show the current temperature and humidity of the surrounding area. Press BUTTON1 on the Nucleo to alternate between Fahrenheit and Celsius.

If the temperature or humidity level exceeds the threshold set in the program code on lines 51 and 52, the vibration motor will turn on, otherwise it will stay off.

Test Plan

The following test plan is to be completed as the system is developed.

1) Test the DHT11 sensor
    a. In a while loop, repeatedly print the results of the sensor to the console.
    b. Try to increase the temperature of the sensor by touching it and verify that the printed values have increased.
    c. Wait a moment and verify the printed values have decreased.
    d. Repeat b and c for the humidity level.
2) Test the LCD display
    a. Print "LCD" to line 1 of the LCD and "Initialized" to line 2 of the LCD, and verify these statements appear on the LCD display
    b. Try to print the results of the DHT11 sensor to the LCD and verify that the LCD displays them.
3) Test the vibration motor

   a. Configure the device driver bitwise to output power to the motor and verify that the motor is running.
   b. Configure the device driver bitwise to stop output to the motor and verify that the motor has stopped running.
   c. Create a clock that turns on the motor on and off every second for one second each time and verify that this works.
4) Test running all three peripherals in the main thread
   a. Try to run a prototype version of the alarm system in the main loop and verify that all three peripherals run correctly.
   b. Verify that the button interrupt works and that it correctly changes the temperature unit displayed on the LCD. Also verify that the values converted is correct using a calculator.
5) Test the system where each peripheral is managed by a separate thread.
   a. Verify that each peripheral is working correctly. The values displayed on the LCD should increase when the DHT11 sensor is being touched and decrease slowly afterwards when the DHT11 sensor is let go. The alarm (vibration motor) should also trigger when the temperature/humidity exceeds the chosen threshold.
   b. Once again verify that the button1 interrupt works correctly.
   c. Let the system run for a few minutes to verify that the program's watchdog is working properly and that the code runs forever.

Outcome of Implementation

Considering the scale of this project, it was successful. There did not seem to be any major bugs that appeared in the end. The temperature/humidity alarm system worked as it intended to be. The values displayed on the LCD also seemed to be correct when compared to a thermometer. In fact, the values displayed by the DHT11 sensor may even be more accurate than a physical thermometer. The system also updates the displayed values on the LCD faster than a physical thermometer.

Future Considerations

Identification of shortfalls

The system is very fragile. If the vibration motor did not have an adhesive side that could be glued on to the breadboard, it would remove itself and all its pins from the breadboard by simply vibrating. The wires connecting each peripheral to the Nucleo and breadboard are just too easy to pull out. All the components are also not waterproof since they have exposed wiring and can be easily damaged if they were to be dropped onto the floor.

The DHT11 sensor takes a relatively long time to return to room temperature if it was heated up a little. This maybe due to the small size of the peripheral and its narrow openings that limits airflow.

Since the components are wired, it is difficult to place the system somewhere far from a computer. Without a power source connection, the system cannot run. This limits the range of the system to the length of the USB cable.

General Improvement

The system can be improved by soldering the components together and creating a custom container for all the parts. This will strengthen the connections between each component and limit the damage that can occur from liquids and other materials by preventing them from touching any of the fragile parts. A battery can also be installed along with some kind of wireless transmitter so that the placement of the system will not be limited by the USB cable length.

A keypad can be added to the system so that the thresholds can be manually adjusted without having to access the program's code. Currently, the thresholds are defined in the code itself which makes it difficult to separate the system completely from the computer.