CSE 321 Project 3

Humidity Alarm

Ronald Chen, Senior

Department of Computer Science and Engineering

School of Engineering and Applied Sciences

University at Buffalo

Fall 2021

Contents

1 Project Overview

1.1 Introduction

This project aims to create a device that can track the humidity levels of a home, display them, and sound an alarm when they reach harmful levels. The alarm will continue to sound until manually muted by the user, or when levels return to ideal levels.

1.2 Applications

Humidity levels that are too high or too low can cause a myriad of problems for a homeowner. High levels can cause structural damage to homes, growth of mold and mildew, damage paintwork, and rot furniture. On the other hand, low levels of humidity can allow viruses and bacteria to spread more rapidly, cause dryness of the skin and eyes, and damage wood. The end goal is the creation of an alarm system that allows homeowners to prevent the damaging effects that may occur when low/high humidity levels are left unaddressed by warning them before they can occur.

1.3 Overview of features and specifications

Software Requirements:

- MBED RTOS
- Use of a synchronization technique.
- Appropriate configuration of watchdog timer.
- Direct bitwise driver configuration.
- Critical section protection for entire implementation.
- At least 1 interrupt configured.
- Use of threads/tasks.

Hardware Requirements:

- Nucleo-L4R5ZI
- DHT11 Sensor
- Buzzer Audio Module
- 16x2 LCD display
- Breadboard
- Wires
- Resistors
- USB A to Micro USB B

Features:

- Displays humidity percentage.
- Displays the state of humidity levels.
- Humidity levels are:
    - Harmful at greater than 70% or lower than 25%

    - Fair between 60 – 70% or 25-30%.
    - Ideal at 30-60%
- Audio alarm activates when state becomes harmful.
- User button B1 will mute the alarm

1.4 Explanations of how the required internal features are integrated

Explanation of thread/tasks and how it was incorporated

The humidity alarm requires 3 threads to function. Each thread is responsible for controlling the behavior of the three external peripherals, the threads that are incorporated are:

Thread 1: t_sense

- Controls the DHT11 sensor behavior.
- Calls the function “sense()”.
- Senses every 2.1 seconds.
- Puts humidity data (int) to the queue “humidityQue”.
- Puts state (string) to the queue “stateQue”.

Thread 2: t_display

- Controls the LCD display
- Calls the function “display()”
- Waits for data to be available in “humidityQue”, and “stateQue”.
- Updates the global variables “humidity” and “state”.
- Prints data to serial monitor and LCD.
- Kicks the watch dog.

Thread 3: t_audio

- Controls the buzzer audio module.
- Calls the function “audio”
- Continuously monitors the global variable “state” and “muted”.
- Writes to the global variable “muted”.

Explanation of synchronization technique and how it was incorporated

Two queues are utilized to ensure synchronization between the input peripheral and the output peripherals. Humidity is sensed with the DHT11 and put into the queue to be received by LCD waits until data is available in the queue and then gets it. This stops the two threads from accessing shared global variables at the same time.

Explanation of bitwise driver control and how it was incorporated

Bitwise driver control was incorporated for the LCD, Buzzer module, and DHT11 sensor. The following code snippets show how each driver is configured using bit masking.

- The RCC is enabled for ports B and C.

```
RCC->AHB2ENR |= 0x6;
```

- Port B holds the pin mappings for the output peripherals, the mode is set to output (01) for pins PB_8 (SCL), PB_9(SDA), and PB_15 (Buzzer module).

```
GPIOB->MODER &=~(0x800A0000);
GPIOB->MODER |= 0x40050000;
```

- Port C holds the pin mappings for the input peripherals, the mode is set to input (00) for pins PC_6 (DHT11 sensor) and PC_13 (User Button).

```
GPIOC->MODER &=~(0xC003000);
```

- The DHR11's input data register is enabled in the main function to allow for input. It is enabled with:

```
GPIOC->IDR |= 0x40;
```

- The buzzer module's output data register is set to output on an active low signal. The buzzer is disabled in the main function before any threads start executing. This is also controlled in the "audio ()" function which is responsible for buzzer behavior.
    - It is disabled with:

```
GPIOB->ODR |= 0x8000;
```

    - It is enabled with:

```
GPIOB->ODR &= ~(0x8000);
```

Explanation of critical section protection and how it was incorporated

Critical section protection was incorporated with the use of the two queues "display_stateQue", and "humidityQue". This avoids the need for the "t_sense" and t_display" threads to access the global variables "state" and "humidity" at the same time.

Explanation of interrupt and how it was incorporated

The on-board user button B1 is configured to call the function mute() on a rising edge. When called it will check if the global variable "state" == "Harmful", at which point it will set the global variable "muted" to true. This will cause the thread "t_audio" to silence the buzzer audio module instantly.

Explanation of watchdog element and how it was incorporated

Due to the DHT11 sensor only being capable of sensing at intervals greater than 2 seconds, the sensing interval is set to 2.1 seconds. The thread t_sense sleeps for 2.1 seconds and puts data to the 2 queues "display_stateQue", and "humidityQue". The thread "t_display" is configured to wait forever until it can successfully get data from the queue. This process should take only a negligible time longer than the time of the sensing interval to complete, therefore the timeout for the watchdog timer is configured to timeout at 2.2 seconds.

1.5 Recap of design process

Stage 1: Identify the problem, and constraints.

The initial step in the design process is to identify the problem that needs be addressed with the given specifications and constraints. The specifications were the incorporation of 3 external peripherals,

programmed using a RTOS, and addresses an area of application. The problem was the need for a device that can warn homeowners of unsafe humidity levels in their home

Stage 2: Research the problem

Research indicates that ideal humidity levels are range from 30-60%, fair at 60-70% and 25-30%, and harmful at levels greater than 70%, or lower than 25%. Low and high humidity levels can cause growth of mold, structural damage, increased infections, etc.… therefore this device applies around safety. As a result of the harmful effects to one's home and health, this device would be desirable to most homeowners.

This device must use synchronization techniques, threads, an interrupt, a watchdog timer, and bitwise control. Synchronization ensures two threads don't attempt to access a shared resource at the same time, threads allow for multiple control flows to occur simultaneously, interrupts signals that an event has occurred, watchdog timers can be used to detect if an error has occurred, and bitwise control uses bit masking to control drivers.

Stage 3: Imagine solutions

The peripherals that must be utilized were a 16x2 LCD, DHT11 sensor, buzzer audio module, and the B1 button. These elements can be used in conjunction to create a humidity sensor, that also functions as an alarm system.

Stage 4: Plan and design

The 2nd stage requires choosing a design and creating a plan to implement it. The design chosen was a humidity alarm that models its functionality after a smoke alarm. It would use the DHT11 sensor to read the humidity percentage of a room, the LCD to display the humidity percentage alongside it's quality level, the buzzer audio module to sound the alarm, and the B1 button to mute it. The software would utilize multiple threads to control each peripheral concurrently. Next, the hardware schematic and flowchart are created.

Stage 5: Implement

Once planning and designing are complete, the hardware schematic and code can be implemented. The initial task to complete is building the design. All peripherals are connected to the corresponding pin mappings chosen. Next would be to code the design in Mbed, incorporating all constraints specified.

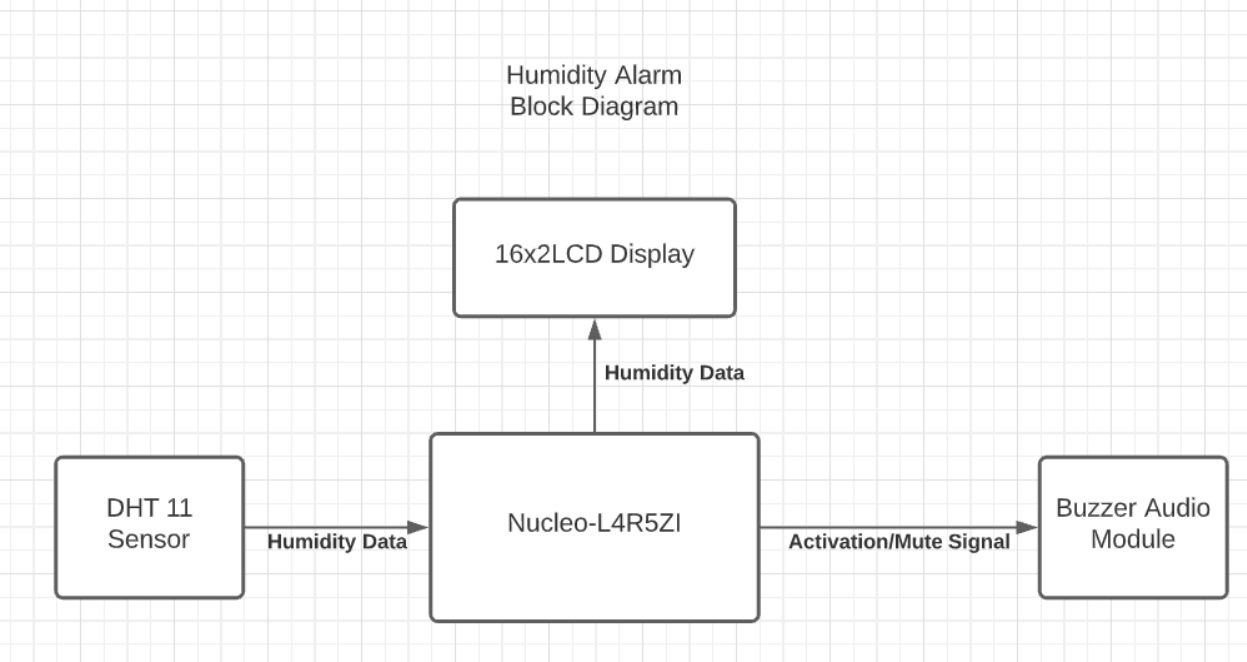
Stage 6: Test

After the implementation of the hardware and software are completed, a testing plan is developed, and the humidity alarm evaluated for correctness.

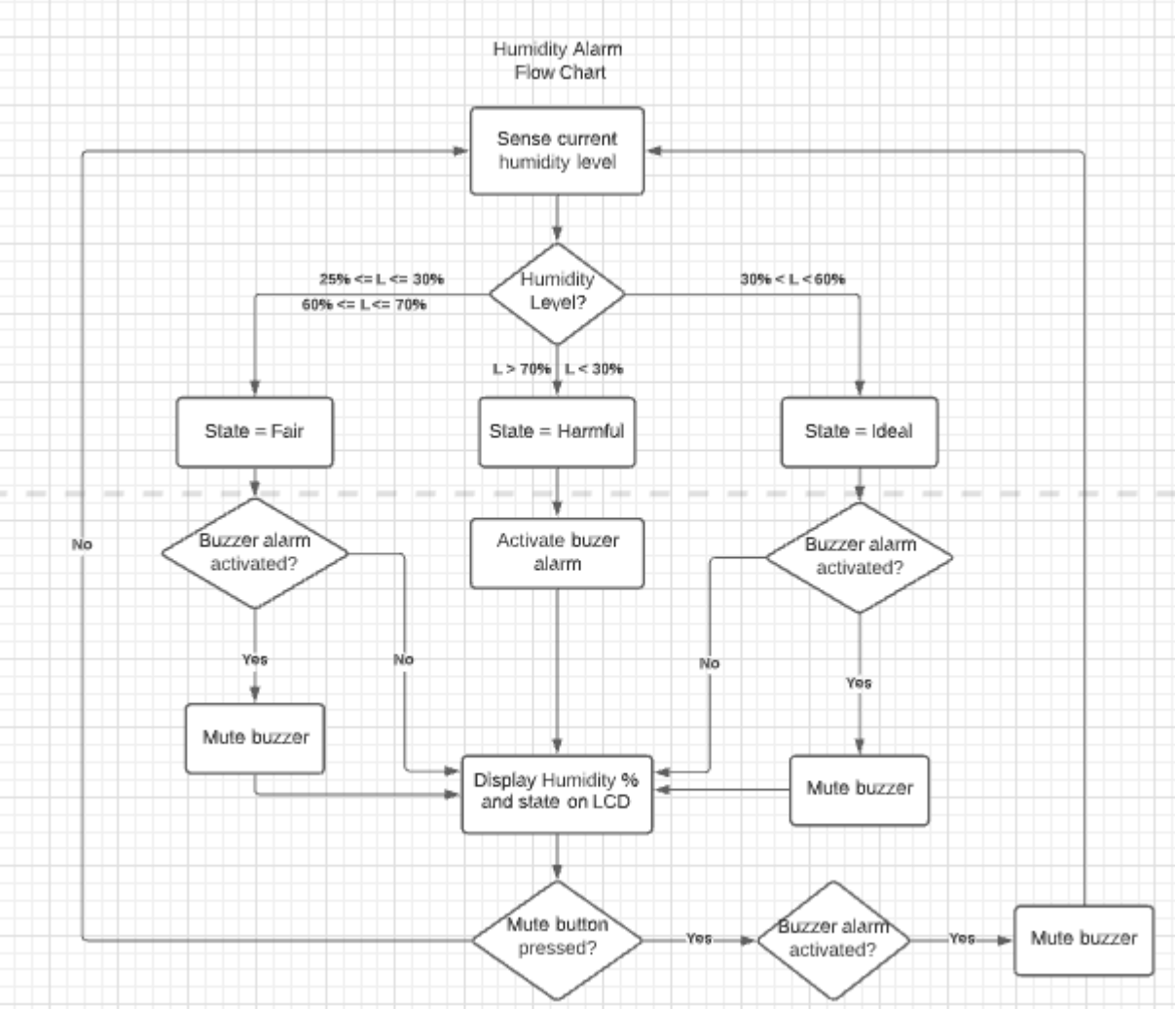
Stage 7: Improve:

The wiring of the device can be simplified, and the color coding of the wires can be improved. An event queue can be incorporated to correctly handle the mute interrupt.

1.6 Block Diagram



1.7 Flow Chart



2 Getting Started

3.1 Bill of Materials

DHT11 Sensor

Description: An external peripheral which can detect humidity levels and temperature

Price: $10.29

Reference: https://components101.com/sites/default/files/component_datasheet/DHT 11-Temperature-Sensor.pdf

Purchase Link: https://www.amazon.com/HiLetgo-Temperature-Humidity-Digital-3-3V5V/dp/B01DKC2GQ0

16x2 LCD Display

Description: An external peripheral that displays text on a 16x2 display.

Price: $8.99

Purchase Link: https://www.amazon.com/SunFounder-Serial-Module-DisplayArduino/dp/B019K5X53O/ref=sxin_13_pa_sp_search_thematic_sspa?cv_ct_cx=LCD+1602&dchild=1&keywords=LCD+1602&pd_rd_i=B019K5X53O&pd_rd_r=533f4e19-cd79- 4704-9ea7-4a9b76ea3af5&pd_rd_w=zEW74&pd_rd_wg=txPku&pf_rd_p=3b2adfc6- e3ad-467a-9f38-271e811048b0&pf_rd_r=A14V16SMA0DCWZABFJ5P&qid=1629389281&sr=1-1- a73d1c8c-2fd2-4f19-aa41-2df022bcb241-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEyNzM5N1RLMU1XWDBOJmVuY3J5cHRlZElkPUEwNzU0MTk3M0ZKRDJVQTJGOTY4SiZlbmNyeXB0ZWRBZElkPUEwNjg1NDY1M0hTTDVSTVhZQlNUQyZ3aWRnZXROYW1lPXNwX3NlYXJjaF90aGVtYXRpYyZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=&pldnSite=1

Buzzer Audio Module

Description: An external peripheral which outputs audio signals.

Price: $7.80

Reference: http://tinkbox.ph/sites/tinkbox.ph/file/downloads/5V_BUZZER_MODULE.pdf

Purchase Link: https://www.amazon.com/HiLetgo-Temperature-Humidity-Digital-3-3V5V/do/B01DKC2GQ0

Nucleo-L4R5ZI

Description: The microcontroller which will control the logic of the system.

Price: $28.14

Purchase Link: https://www.mouser.com/ProductDetail/511-NUCLEO-L4R5ZI

Wires

Description: Wires allow for the flow of electricity between components.

Price: $3.90

Purchase Link: https://www.mouser.com/ProductDetail/713-110990044

Breadboard

Description: A device upon which electric components are held and connected.

Price: $15.40

Purchase Link: https://www.mouser.com/ProductDetail/854-BB1460

Resistors

Description: Resistors regulate voltage between components.

Price: $0.95

Purchase Link: https://www.mouser.com/ProductDetail/474-PRT-14492

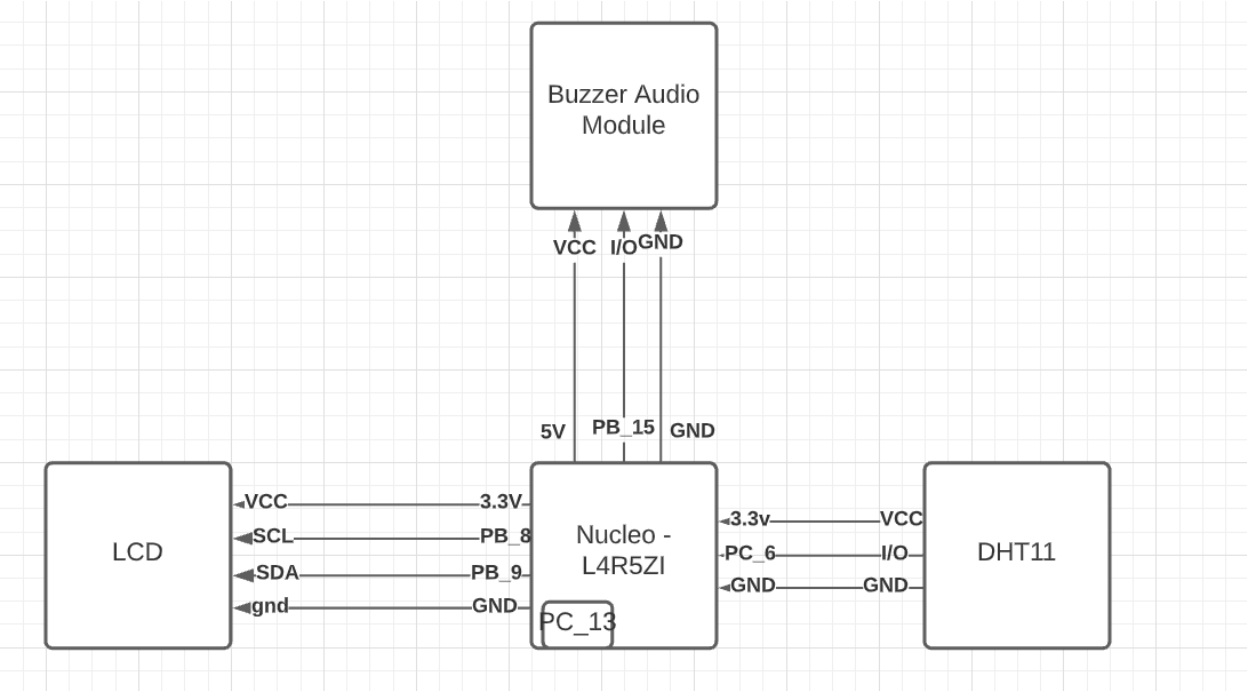
USB A to Micro USB B cable

Description: Cable which will connect the Nucleo-L4R5ZI to a computer.

Price: $7.99

Purchase Link: https://www.bestbuy.com/site/best-buy-essentials-3-micro-usb-to- usbcharge-and-sync-cable-black/6456436.p?skuId=6456436

3.2 Schematic



3.3 How to build

This section will cover the steps required to configure the hardware elements of the device. All resistors used will be 1kOhm.

1. Acquire all hardware elements as described in the BOM.
2. Connect the Nucleo-L4R5ZI to a computer with a USBA to Micro USB.
    a. Ensure a proper connection has been made, the Nucleo-L4R5ZI should appear in the device manager.
3. Set up the buzzer audio module.
    a. VCC to +5V, add a resistor.
    b. GND to GND
    c. I/O to PB_15
4. Set up the 16x2 LCD display.
    a. VCC to +3.3V, add a resistor.
    b. GND to GND.

    c. SCL to PB_8.
    d. SDA to PB_9
5. Set up the DHT11 sensor.
    a. VCC to +3.3V, add a resistor.
    b. GND to GND
    c. Data to PC_6

3.4 How to use

This section will detail how to operate the humidity alarm. Make sure the steps described in section 3.3 are completed before continuing.

1. Once the device is connected to a power source the LCD should display “Humidity Level: __%” on the 1st row and “State: ___ “on the 2nd row.
2. The device will automatically detect and update the display with the current humidity and state every 2.1 seconds.
3. The state indicates the quality of the current humidity. The alarm will sound if state becomes “Harmful” and will automatically mute once levels become “Ideal” or “Fair”.
    a. Ideal: This indicates optimal humidity %
    b. Fair: Humidity % isn’t optimal, but not harmful.
    c. Harmful: Humidity is too low or too high, consider addressing this to avoid negative effects.
4. If the alarm becomes activated, the user button B1 may be pressed to mute it.
    a. Once muted, it cannot be unmuted unless state changes to “Fair” or “Ideal” and back to “Harmful” again.

3 Analysis

4.1 Test Plan Instructions

This section details the purpose, inputs and expected outputs of each test case.

Test 1: Test behavior of low harmful humidity

Test 2: Test behavior of high harmful humidity

Test 3: Test behavior of low fair humidity

Test 4: Test behavior of high fair humidity

Test 5: Test behavior of ideal humidity

Test 6: Test behavior of buzzer when going from harmful state to fair.

Test 7: Test behavior of buzzer when going from fair to harmful.

Test 8: Test mute button functionality.

Test #	Humidity %	State	Expected LCD Display	Buzzer activation?
1	15	Harmful	Humidity Level: 15% State: Harmful	Yes
2	80	Harmful	Humidity Level: 80% State: Harmful	Yes
3	27	Fair	Humidity Level: 27% State: Fair	No
4	65	Fair	Humidity Level: 65% State: Fair	No
5	50	Ideal	Humidity Level: 50% State: Ideal	No
6	71% → 69%	Harmful → Fair	Humidity Level: 69% State: Fair	Yes → No
7	69% → 71%	Fair → Harmful	Humidity Level: 71% State: Harmful	No → Yes
8	72%	Harmful	Humidity Level: 71% State: Harmful	Yes → No

4.2 Outcome of Implementation

Test 1: Pass, buzzer activates, and the LCD updates appropriately.

Test 2: Pass, buzzer activates, and the LCD updates appropriately.

Test 3: Pass, buzzer isn't activated, and the LCD updates appropriately.

Test 4: Pass, buzzer isn't activated, and the LCD updates appropriately.

Test 5: Pass, buzzer isn't activated, and the LCD updates appropriately.

Test 6: Pass, the buzzer disables immediately after the state transitions and the LCD updates appropriately.

Test 7: Pass, the buzzer activates immediately after the state transitions and the LCD updates appropriately.

Test 8: Pass, the buzzer disables immediately after the button is pressed.

4.3 Future Considerations

The design can be expanded to also include an option to switch the device from a humidity alarm to a thermometer with the use of a new input peripheral. Additionally, critical section protection is not fully implemented as can be seen with the mute interrupt. The mute interrupt writes to a global variable "muted" when pressed, however, the audio thread also writes to this variable when switching from a

harmful state to fair/ideal. Now, no issues have been found because of this, although a potential solution may be to use an event queue

4.4 Development Timeline/Revision History

11/18/2021 - Design and planning steps completed

11/19/2021 – Hardware elements built.

11/20/2021 – Program and function headers written

11/21/2021 – Main, sense, audio, and display functionality completed.

11/24/2021 – Mute functionality completed.

11/29/2021 - Fixed interrupt bounce bug.

12/10/2021 – Code finalized.

12/16/2021 – Report Finalized