Food Waste Reduction

Inventory Notifier

CSE321 Fall 2021: Project 3

Misha Nelyubov

December 3, 2021

Table of Contents

[01. Introduction 2](#_Toc89604969)

[02. Specifications and Features 3](#_Toc89604970)

[a) Features 3](#_Toc89604971)

[b) Specifications 3](#_Toc89604972)

[03. Integration of Required Features 3](#_Toc89604973)

[a) Watchdog Integration 3](#_Toc89604974)

[b) Synchronization Integration 4](#_Toc89604975)

[c) Bitwise Driver Control Integration 4](#_Toc89604976)

[d) Critical Section Protection 4](#_Toc89604977)

[e) Multithread Implementation 4](#_Toc89604978)

[f) Interrupt Implementation 4](#_Toc89604979)

[04. Design Process 4](#_Toc89604980)

[a) Ask 4](#_Toc89604981)

[Purpose 4](#_Toc89604982)

[Inputs 4](#_Toc89604983)

[Outputs 4](#_Toc89604984)

[Constraints 5](#_Toc89604985)

[b) Research/Imagine 5](#_Toc89604986)

[05. Block Diagram 5](#_Toc89604987)

[06. State Diagram 5](#_Toc89604988)

[07. Bill of Materials 6](#_Toc89604989)

[08. Instructions 7](#_Toc89604990)

[a) Schematic 7](#_Toc89604991)

[b) Construction Instructions 7](#_Toc89604992)

[c) Usage Instructions 8](#_Toc89604993)

[09. Test Plan Instructions 8](#_Toc89604994)

[10. Revision History Timeline 9](#_Toc89604995)

[Appendix A: References 10](#_Toc89604996)

# 01. Introduction

This system aims to assist in reducing waste of short lifespan store foods by tracking how much of a container’s inventory is occupied over the course of a work day. If there is food remaining in the container at the end of the work day, the system will generate a sound to alert workers at the end of a business day that the remaining food in the container can be taken home and consumed before it goes bad.

# 02. Specifications and Features

## a) Features

* Once powered on and configured, the system will remember the current real-world time for the duration of the power cycle, or until modified.
* A custom 24-hour “closing time” can be configured down to the second to meet the exact needs of its specific implementation.
* Any values not input during time configuration will default to 0 to save input time.
* The distance sensor regularly polls to check how much of the container is occupied and provides a real-time percentage estimate once calibration is completed.
* The points at which the container is considered empty and full can be calibrated to support a wide range of containers.
* Once configured, the system alarm can be toggled on and off by pressing the “#” key on the matrix keypad.
* The buzzer will play a series of tones to alert surrounding workers when the container is NOT empty after “closing time” has been passed.
* At any point, the key “D” can be pressed to reconfigure and recalibrate the system.

## b) Specifications

* The distance sensor has a maximum effective range of 400 cm (SparkFun Electronics).
* The distance sensor has a minimum effective range of 2 cm (SparkFun Electronics).
* The distance sensor is polled for a new distance value once every 100ms (10 Hz).
* The displayed value for the distance sensor is always the average of the last four distance sensor samples that were taken within the effective sensor range.
* The text displayed on the LCD is refreshed for updates every 100ms (10 Hz).
* Current Time and Closing Time are both 24-hour time values with valid values ranging from 00:00:00 up to 23:59:59.
* The matrix keypad is polled at 10ms per column. Polling is stopped while any matrix key is pressed down.
* If a matrix key is perceived by the system to be pressed down for 30 consecutive seconds, the system watchdog will assume that the key has gotten stuck and force a system reset.

# 03. Integration of Required Features

## a) Watchdog Integration

The system watchdog was implemented to protect users from a keypad input lockout that would risk the system being unresponsive to further user keypad input. This protection point is considered crucial because loss of access to the keypad prevents any user from being able to modify the state of the system, including loss of the ability to directly disable an ongoing alarm. Because user key inputs are tracked based on rising and falling edges of button presses with an internal variable storing the currently pressed key, the watchdog timer is allowed to count down the 30 seconds whenever any key is pressed. If no key press is detected for any 1ms during that countdown, this indicates that a user is able to press an input key again and the watchdog reset will be called off. The watchdog is periodically reset back to counting down from 30 seconds while no key is pressed.

## b) Synchronization Integration

Synchronization is necessary to ensure that data flow between input and output peripheral control is not accessed while it is being updated and that system resources are used efficiently. To ensure efficient and cooperative operation, a mutex is used to control synchronization for modifying output peripherals. Anytime that a change is made in the system that would affect the state of the output peripherals, a mutex-protected flag is raised indicating that output changes have been made. The next time that the function that updates the state of the output peripherals is called, it will check if that flag is raised to be informed if any action needs to be taken. If that flag is raised, the output peripherals will be modified. Otherwise, the function exits immediately to save system resources.

## c) Bitwise Driver Control Integration

A bitwise output driver was implemented for the distance sensor trigger. Modification of the output bit from a value of 0 to a value of 1 for a duration of 10 microseconds before returning to 0 is responsible for sending a trigger signal to the distance sensor in order to cause the sensor to generate a distance measurement (SparkFun Electronics). Configuration of the bitwise driver occur in the main function. The main function is responsible for enabling port C and configuring pin C9 as an output to be used by the distance sensor. Modifications to the output value of the pin are made by the function pollDistanceSensor.

## d) Critical Section Protection

Critical Section Protection is necessary when more than one thread needs to access to a resource that is accessed by more than one thread. Mutual exclusion is implemented to prevent race conditions where more than one thread attempts to access the same resource, resulting in an unintended system state. A Mutex lock encapsulates all instances of functions reading from and writing into the LCD output text table. If the mutex is not used, a thread writing to the output table might result in a partially changed value being set into the string before a context switch causes that partially changed value to be printed onto the LCD before the completed new data is written. Usage of the mutex ensures that any thread attempting to access the LCD output table must wait until no other thread is accessing the table before being granted access to proceed with its operation.

## e) Multithread Implementation

Individual threads are implemented in the system to control the multiple system peripherals. All operations pertaining to the distance sensor are executed by the Distance Sensor Thread. This ensures that all internal variables associated with the distance sensor are only modified by whatever function is being executed by that thread. All non-ISR functions that handle the behavior of the distance sensor are called by being put into the distance sensor event queue or by being called by a function being executed from that event queue. Use of primarily thread-internal variables that do not need to be locked by mutexes minimizes the lockout period that other threads may experience while the Distance Sensor Thread is modifying shared system variables. All threads and tickers to enqueue periodic events for the threads are started by the main function after interrupts and peripheral drivers have been configured. Function execution and data access by all implemented threads is visualized in Section [05. Block Diagram](#_05._Block_Diagram).

## f) Interrupt Implementation

Interrupts are used to immediately react to changes in the state of the echo signal sent from the distance sensor to the NUCLEO. The alternative method to using interrupts to acquire the echo signal data would be polling the input pins. Due to an accurate count of milliseconds between the rising and falling edge sensor echo signals being essential for accurately calculating the current distance, the low cost and high precision solution of interrupts is optimal. The interrupts are declared in a global scope and initialized within the main function to call two ISR functions, corresponding to a rising and falling edge trigger event. These functions record the time in milliseconds between the distance sensor being polled and the edge event being detected. The falling edge ISR enqueues a non-ISR function to be handled by the distance sensor thread to calculate the data rapidly recorded during the short interrupt handler periods.

# 04. Design Process

## a) Ask

### Purpose

In an effort to minimize food waste, foods that aren’t sold at the end of a work day can be taken home by the workers and consumed while the food is still in good condition. Design a system that will alert staff members at the end of a work day if there is still food in a container that can be taken home for consumption.

### Inputs

TODO

### Outputs

TODO

### Constraints

TODO

## b) Research/Imagine

TODO

In order to recognize if there is food remaining that can be taken home, a distance sensor can be used to approximate how much food is currently in a container. For this approximation to be accurate, the distance sensor’s data will also need reference distances for the maximum length away from the sensor when the container is empty and the minimum length away from the sensor as a reference for when the container is full. One button will be needed to allow a user to confirm these distances.

In order to recognize when workers should be alerted that there is remaining food, the system will need to be aware of the current time and the time after which it should notify workers of any remaining contents. Ten additional buttons will be needed to allow an easily accessible range of input numbers.

For both of the required user inputs, an output indicator of what the system’s user has configured and is about to confirm would improve the ease of use of the system. An LCD can be used to display the current time, closing time, and distance between the sensor and the nearest object to it in the container.

The system should notify workers in such a way that they do not need to be actively monitoring the system in order to be alerted if it is past closing time and the container is not empty. A buzzer module would be able to accomplish this by creating a sound that will alert everyone in the immediate area. This buzzer-produced sound should be designed to not be excessively unpleasant nor cause any hearing problems. An additional input button should be dedicated to explicitly shutting down/restarting the alarm.

# 05. Block Diagram

The following color code is used in the block diagrams for each thread of the program listed below:

|  |  |
| --- | --- |
| Block Color | Meaning |
| Red | This block represents a ticker. |
| Orange | This block represents a thread. |
| Yellow | This block represents an ISR function. |
| Green | This block represents a non-ISR function. |
| Half - Green | This block represents an Event Queue of non-ISR functions. |
| Blue | This block represents a direct pin or API access to hardware peripherals. |
| Purple | This block represents data access to internal variables.  Parenthesized numbers that precede variable names indicate that the variable is protected by an ordered Mutex lock. |

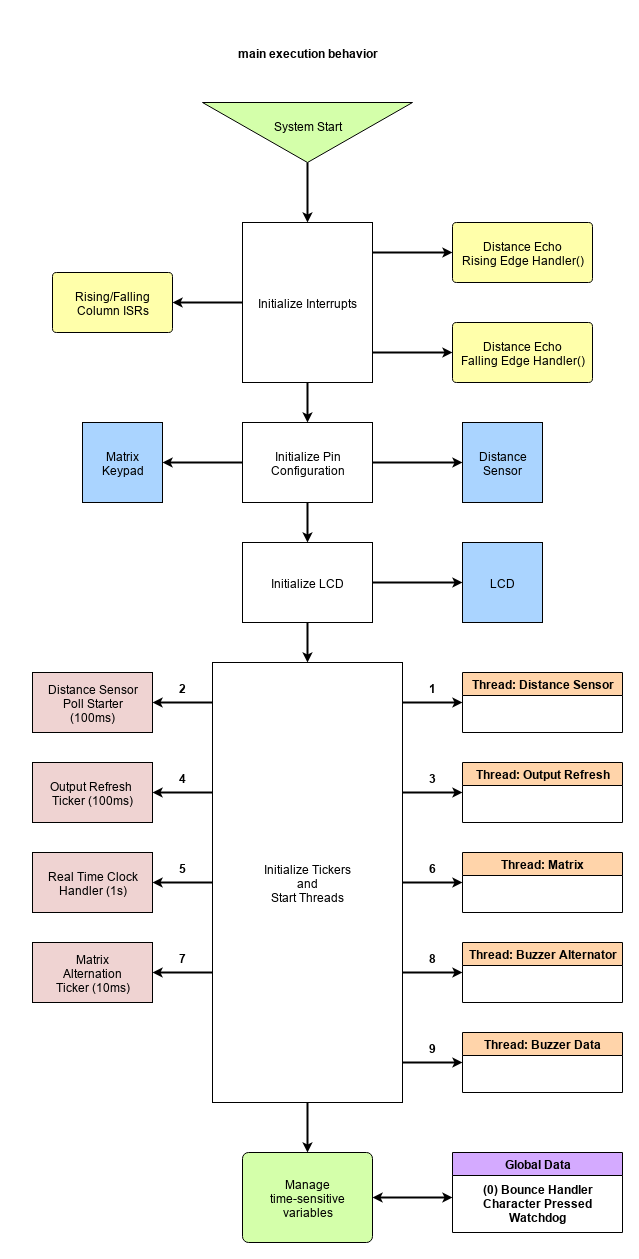
Figure 1: Main program execution behavior

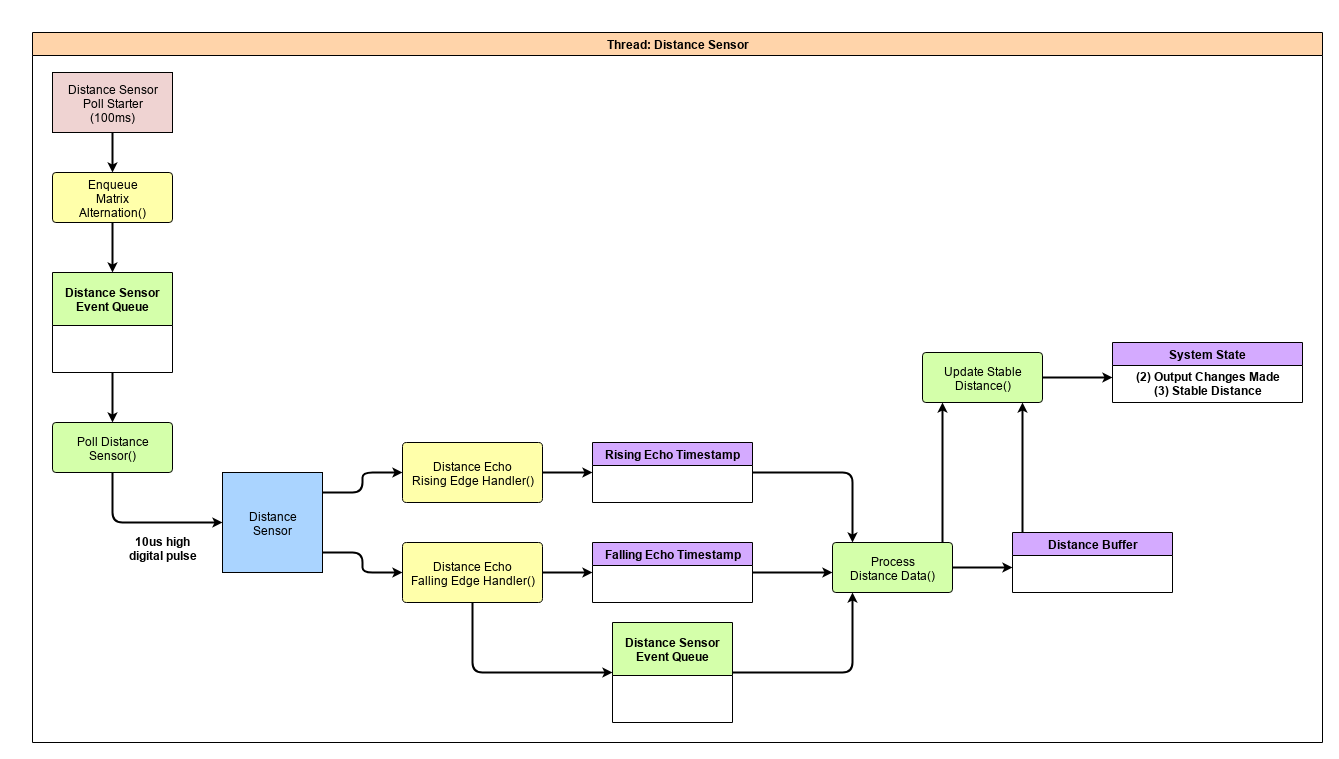
Figure 2: Distance Sensor Thread

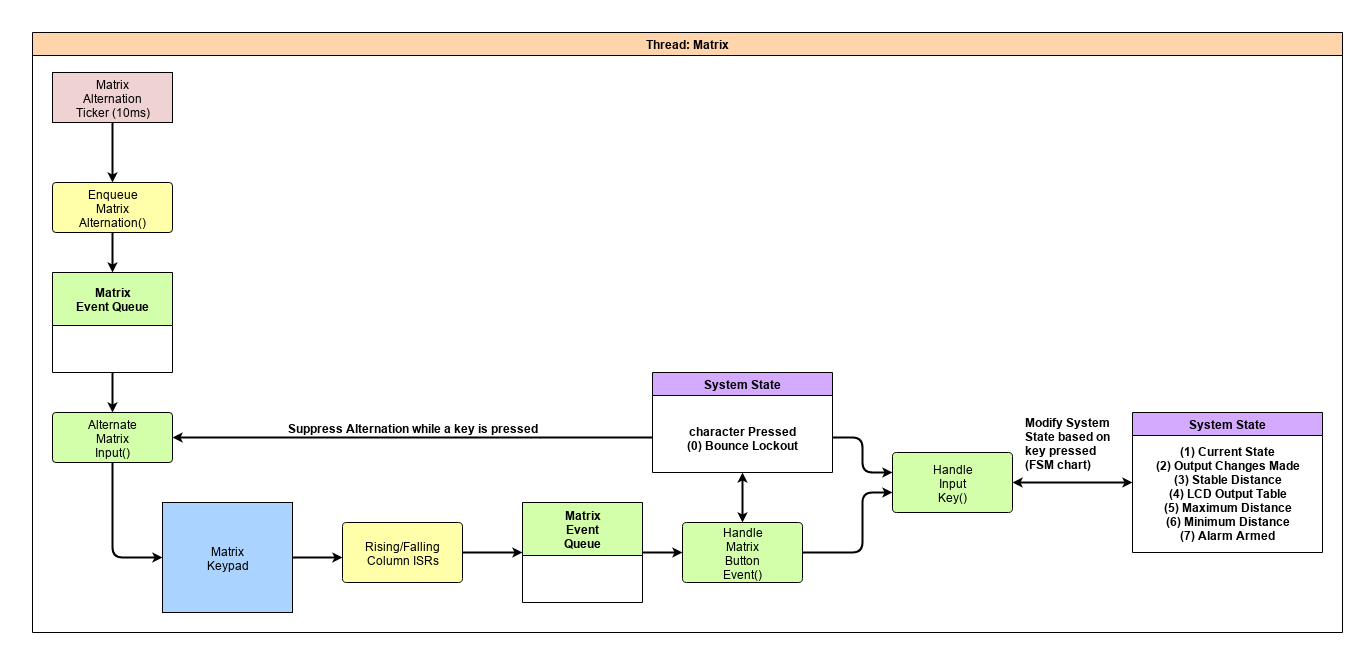
Figure 3: Matrix Input Thread

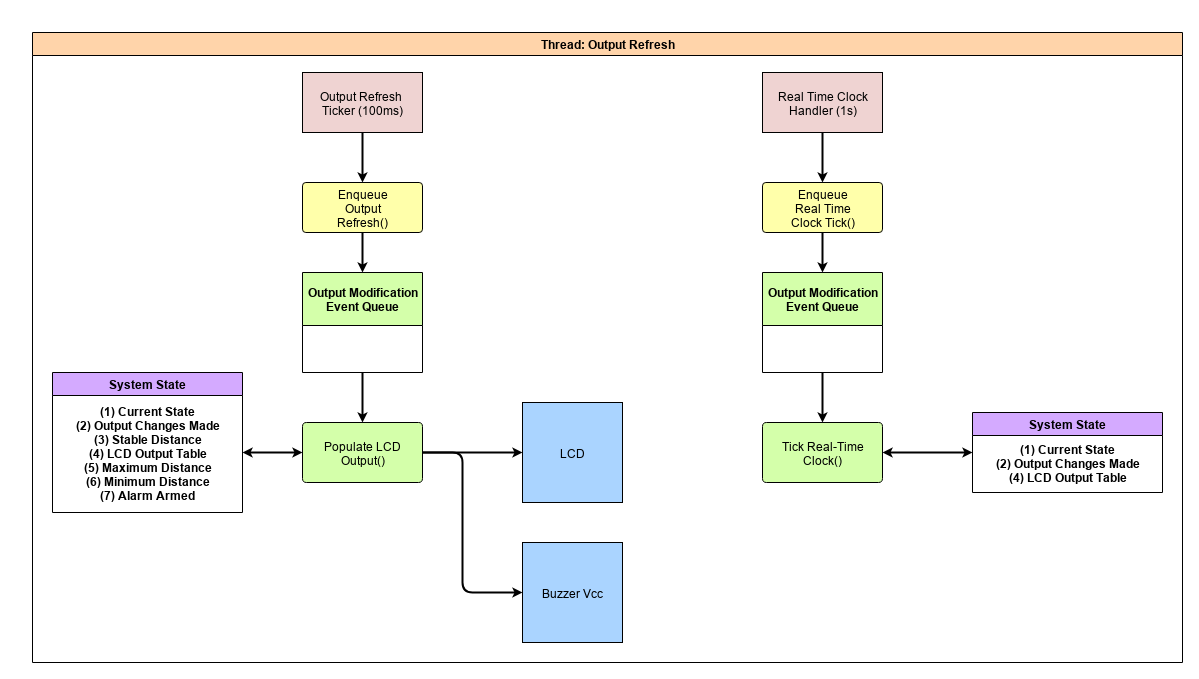
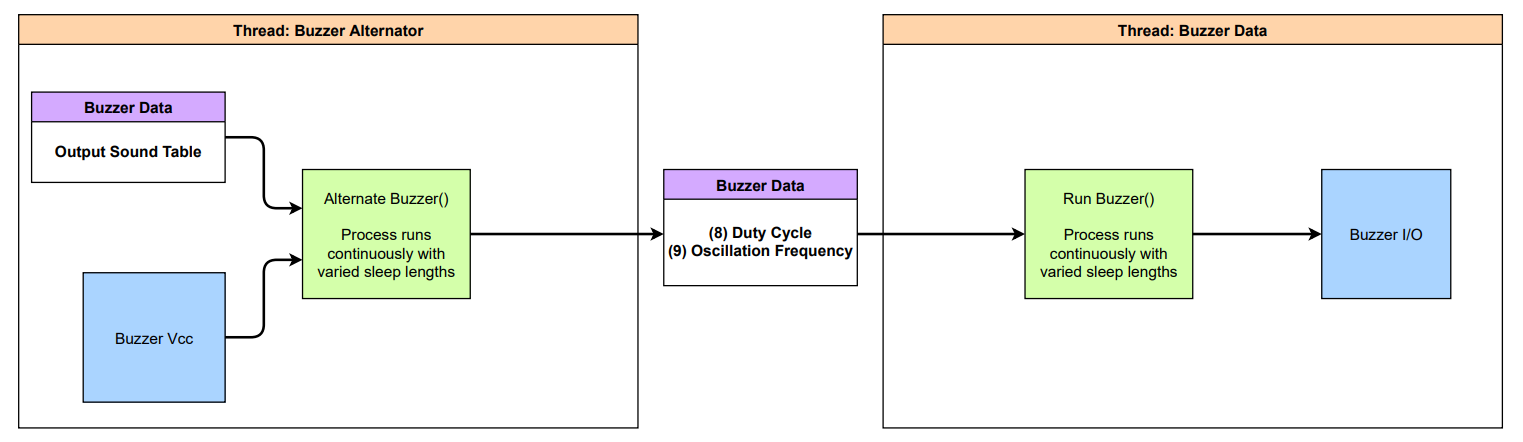
Figure 4: Output Refresh Thread

Figure 5: Buzzer Threads

# 06. State Diagram

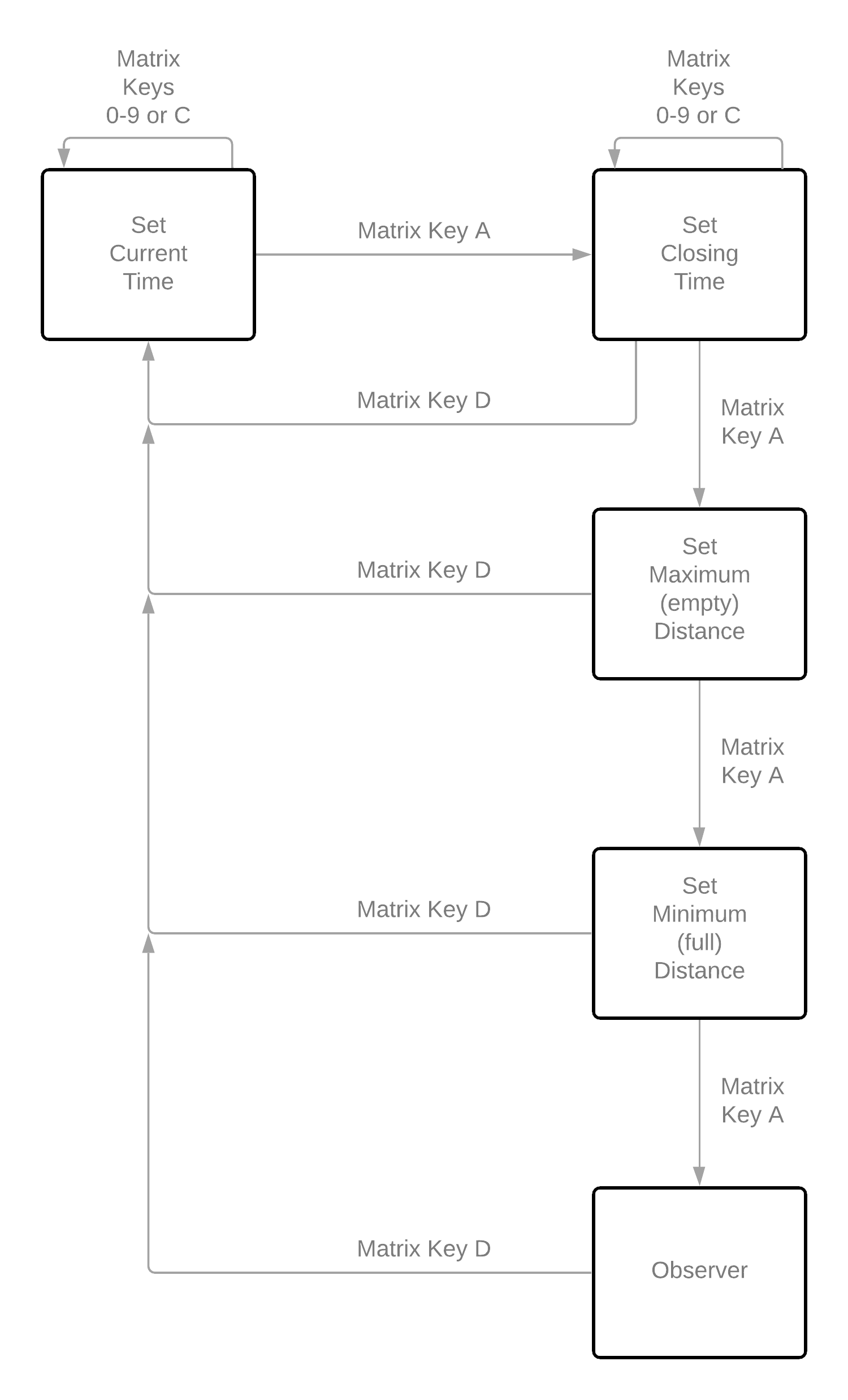
The following diagram covers the conditions necessary for the transitions between states.

The matrix key “A” is used as an option during all configuration stages to proceed to the next state.

The matrix key “C” is used in the time input stages to reset the input time to “hh:mm:ss” and reset the user data input cursor to the tens of hours position.

The matrix key “D” is used as a universal reset to the “Set Current Time” state in order to modify the system configuration.

The system buzzer peripheral will only produce sound when the system is in the Observer state.



# 07. Bill of Materials

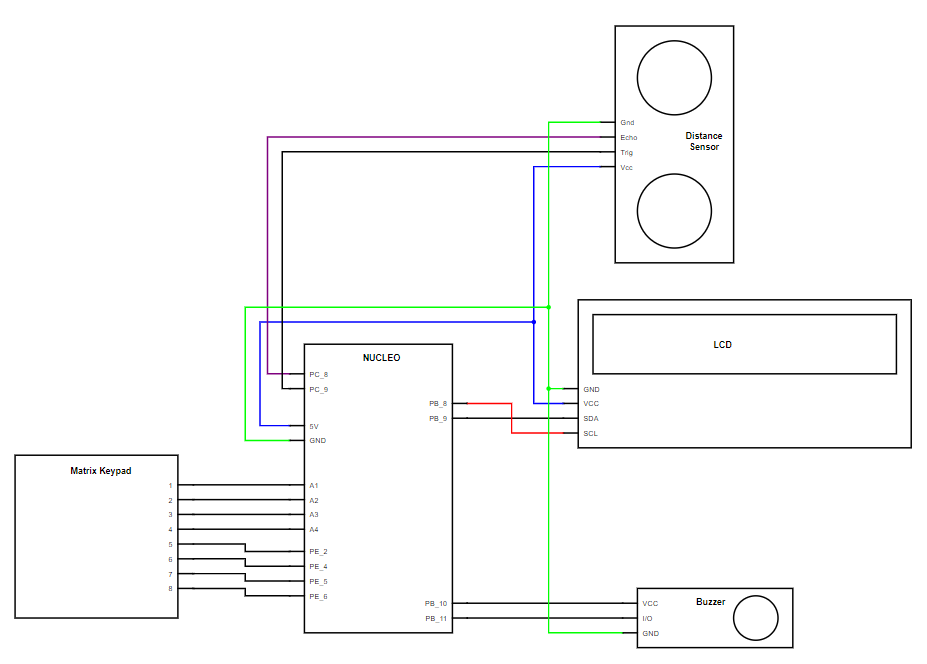
The following hardware will be required to create the system:

* NUCLEO L4R5ZI microcontroller
* 4x4 matrix keypad (8-pin)
* JHD1804 LCD
* 4-pin ribbon cable for connecting LCD to breadboard
* HC-SR04 Distance Sensor
* MH-FMD Buzzer Module (Low Level Trigger)
* Solderless breadboard
* USB 2.0 A to USB 2.0 Micro B cable
* Jumper wires (no less than 21)

# 08. Instructions

## a) Schematic

The following set of connections between the NUCLEO and peripherals need to be established (USB connection to a computer not pictured):



## b) Construction Instructions

In order to build the system, the following connections need to be established between the NUCLEO and peripherals using jumper wires. Use of a solderless breadboard and side rails for 5V (VCC) and GND is recommended.

Matrix Keypad:

* Matrix pin 1 must be connected to the NUCLEO Pin A1 (PC\_0)
* Matrix pin 2 must be connected to the NUCLEO Pin A2 (PC\_3)
* Matrix pin 3 must be connected to the NUCLEO Pin A3 (PC\_1)
* Matrix pin 4 must be connected to the NUCLEO Pin A4 (PC\_4)
* Matrix pin 5 must be connected to the NUCLEO Pin PE\_2
* Matrix pin 6 must be connected to the NUCLEO Pin PE\_4
* Matrix pin 7 must be connected to the NUCLEO Pin PE\_5
* Matrix pin 8 must be connected to the NUCLEO Pin PE\_6

LCD:

* The LCD pin labeled GND must be connected to the NUCLEO pin GND
* The LCD pin labeled VCC must be connected to the NUCLEO pin 5V
* The LCD pin labeled SDA must be connected to the NUCLEO pin PB\_9
* The LCD pin labeled SCL must be connected to the NUCLEO pin PB\_8

Distance Sensor

* The Distance Sensor pin labeled Gnd must be connected to the NUCLEO pin GND
* The Distance Sensor pin labeled Echo must be connected to the NUCLEO pin PC\_8
* The Distance Sensor pin labeled Trig must be connected to the NUCLEO pin PC\_9
* The Distance Sensor pin labeled Vcc must be connected to the NUCLEO pin 5V

Buzzer

* The Buzzer pin labeled VCC must be connected to the NUCLEO pin PB\_10
* The Buzzer pin labeled I/O must be connected to the NUCLEO pin PB\_11
* The Buzzer pin labeled GND must be connected to the NUCLEO pin GND

Once all connections between the NUCLEO and peripherals have been established, connect the NUCLEO USB-B port labeled “USB PWR” to the computer that will load the program onto the NUCLEO. This computer will need to have MBED Studio installed to compile the code and load it onto the NUCLEO.

After MBED Studio is installed, perform the following steps to load the necessary program onto the NUCLEO.

1. Download the GitHub repository containing the necessary program. If Git is installed on the system, this can be done with the following command:
   1. git clone https://github.com/CSE321-Fall2021/cse321-portfolio-MSNelyubov.git
2. In MBED Studio, click on File > Open Workspace.
3. Navigate to the path where the repository was downloaded.
4. Click on Project 3 and then click *Select Folder*.
5. Set the *Active program* to “Project 3”.
6. Set the *Target* to “NUCLEO-L4R5ZI …”.
7. Click on the play button to run the program.

## c) Usage Instructions

TODO

# 09. Test Plan Instructions

TODO

# 10. Revision History Timeline

2021 November 19: Project Statement and preliminary Bill of Materials generated.

2021 November 20: Implemented the range testing program to control the new distance sensor peripheral in an isolated environment.

2021 November 22: Implemented a multithreaded testing program to verify the effective utilization of Mutexes to control access to shared resources by multiple threads.

2021 November 24: The distance sensor and LCD were implemented as two independent threads of the main program with Mutex-protected access to shared resources.

2021 November 25: The Matrix keypad code from Project 2 was merged into the main program of Project 3 as a third thread to allow for User Input to control the state machine elements of the system.

2021 November 26: The state pattern and signal to the buzzer output were implemented into the main program.

2021 November 29 – December 01: Comments were added to previously developed code.

2021 November 30: The first implementation of the watchdog timer was integrated into the main program.

2021 December 01: The watchdog timer was moved to the main method and modified to use the kick method due to the stop method not operating as intended.

2021 December 01: A test program to send a square wave signal to the Buzzer I/O pin was created.

2021 December 02: A duty cycle parameter was added to the buzzer test program and a sound effect was developed to play on the buzzer instead of a single constant note.

2021 December 02: The sound effect buzzer developed in the buzzer test program was merged into the main program utilizing two additional threads.

2021 December 03: Comments were added to the newly developed buzzer code.

# Appendix A: References

SparkFun Electronics, “Ultrasonic Ranging Module HC - SR04,” *HC-SR04 Datasheet by SparkFun Electronics*. [Online]. Available: https://www.digikey.com/htmldatasheets/production/1979760/0/0/1/hc-sr04.html. [Accessed: 05-Dec-2021].

(STMicroelectronics NV, 2021)

STMicroelectronics NV, “RM0432 Reference manual Revision 9,” *STM32L4+ Series advanced Arm®-based 32-bit MCUs - Reference manual*, 22-Jun-2021. [Online]. Available: https://www.st.com/resource/en/reference\_manual/dm00310109-stm32l4-series-advanced-armbased-32bit-mcus-stmicroelectronics.pdf. [Accessed: 05-Dec-2021].

TODO

\* Buzzer datasheet: https://www.mouser.com/datasheet/2/400/ef532\_ps-13444.pdf

\* MBED OS API: timer https://os.mbed.com/docs/mbed-os/v6.15/apis/timer.html

\* MBED OS API: Watchdog https://os.mbed.com/docs/mbed-os/v6.15/apis/watchdog.html