Fall 21

Project 3

CSE 321 – REALTIME AND EMBEDDED SYSTEMS – FALL 21

NICK MINOR

PURPOSE:. THIS IS A SAFETY SYSTEM THAT SOUNDS AN ALARM WHEN AN OBJECT PASSES THROUGH A USER-SPECIFIED AREA.

Table of Contents

- Introduction
- Feature, Specifications and Constraints
- Integration of Required Internal Features
- Block Diagram
- Functionality Diagram
- Bill of Materials
- User Instructions
- Test Plan
- Development Timeline / Revision History

Introduction

This system determines if something has passed through an area of interest using an ultrasonic sensor and notifies the user. The system sounds a buzzer if anything passes within a certain range and the distance to the nearest object is displayed on the LCD screen. The range that the alarm will sound within can be modified by the user using the matrix keypad. This system can be used for **safety** applications such as keeping people away from dangerous areas, used as a security alarm or being placed on a vehicle to help a driver avoid collisions.

Features, Specifications and Constraints

Features

- A user can interact with the system by doing the following using the Matrix Keypad:
    - Pressing A:
        - Change alarm range
    - Pressing B:
        - Apply the inputted alarm range
    - Pressing 0-9:
        - Specifying distance for the alarm range
- A buzzer will ring when an object enters within the alarm range.
- The LCD screen will display the distance to the nearest object in its path and the Range at which the alarm will ring at. It is displayed as: XXX / YYY cm.

Specifications:

- The system will poll for objects every 500ms. This is done by a trigger pulse being sent to the ultrasonic sensor, eight 40kHz ultrasonic signals being output and echoing back.
- To get the distance to an object, the width of the returning pulse is divided by 58 to get the distance in centimeters.
- Valid distances are between 2cm and 400cm.

Constraints:

- The ultrasonic sensor will be most reliable when trying to detect objects within a 30-degree angle.
- The time for the HC-SR04 to detect if an object is present takes at minimum 50ms. This is the time for an electric signal to be sent to the sensor, the ultrasound burst to be emitted and consequently echoed back.
- Measurements may be less accurate when the area being surveyed is not a smooth plane. Small objects are difficult for the sensor to detect at longer ranges.

Integration of Required Internal Features

- **Watchdog**
  - The watchdog is set to activate after 2 seconds and is refreshed within the main loop every time a signal is sent and received by the ultrasonic transducer. Since a pulse is sent using a Ticker every 500ms, 2 seconds is plenty of time for this process to complete. If the process takes longer then something went wrong, and the watchdog will restore the program to a safe state.

- **Synchronization**
  - Inside the main function, the thread `t2` associated with the matrix keypad is given priority over thread `t1`. This is since receiving an input from a user is much more important than sending a trigger to the HC-SR04. If a trigger is late, then the value on the LCD will simply update later, whereas if the rows aren't cycling in time a user's input could be read incorrectly or missed. Furthermore, the period of the cycling power through the rows using `rowCycler()` is 80ms whereas the period for sending the trigger to the ultrasonic transducer using `isr_start` is 500ms. This means that the trigger can better afford to wait than the row cycle, as it has more time before its next period begins.

- **Bitwise Driver Control**
  - The buzzer and the matrix keypad are both controlled using bitwise operations on GPIO ports. In the main function, the clock is enabled on ports A, B and C .
  - In the main function, the MODER for PB2 (buzzer) is set to general purpose output mode (01). Bitwise control on the buzzer is also used within the `ringBuzzer()` and `silenceBuzzer()` methods of the `IO_Interface` class. These methods set the ODR bit for PB2 to 0 and 1 respectively.
  - In the `MatrixKeypad` class, the MODER for pins PB4, PA5, PB3 and PB5 are set to general purpose output mode (01) within the default constructor. Bitwise control is used on the ODR registers for each pin in the `rowCycler()` function inside the main file, which is attached to the `Ticker KeypadCycler` to

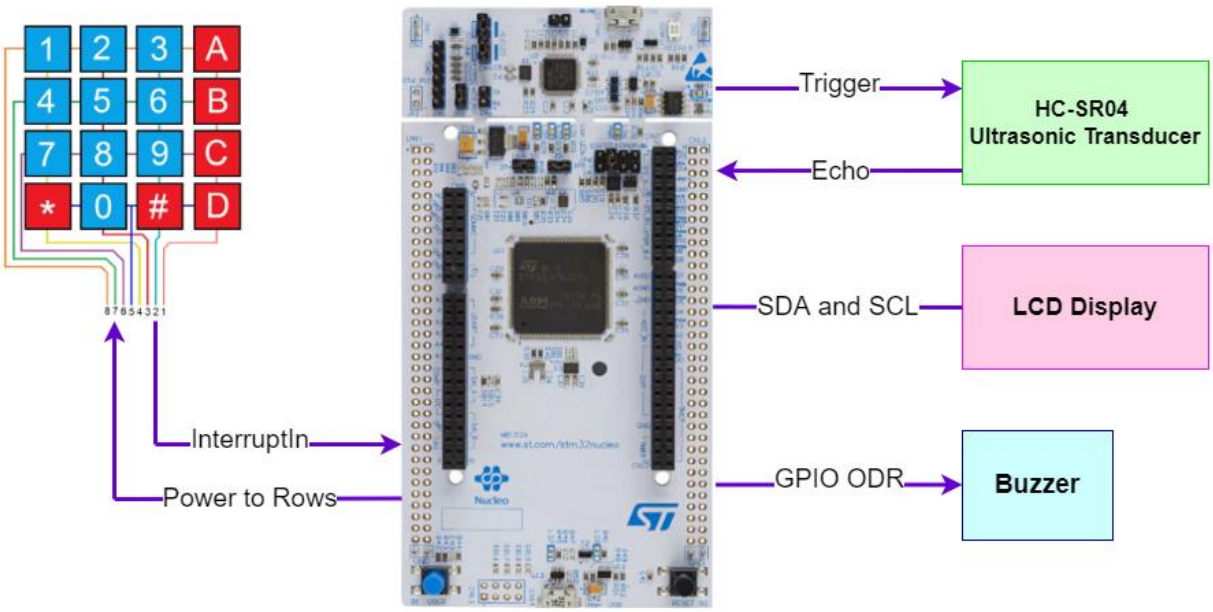
execute every 80ms. This function cycles through powering each row so when an interrupt is triggered from the keypad, both the column and row of the key that was pressed is known.

- **Critical section protection**
  - The Event Queue q attached to thread t2 is used to prevent critical section issues inside the ISRs isr_col1, isr_col2, isr_col3 and isr_col4.
    - Allows for the safe and proper execution of printf statements inside of the ISRs.
    - Protects the global `IO_Interface` object `io` by placing calls to `insertAlarmDist`, `A_ChangeDist` and `B_CompInput` into the queue so the alarm range value doesn't alter.

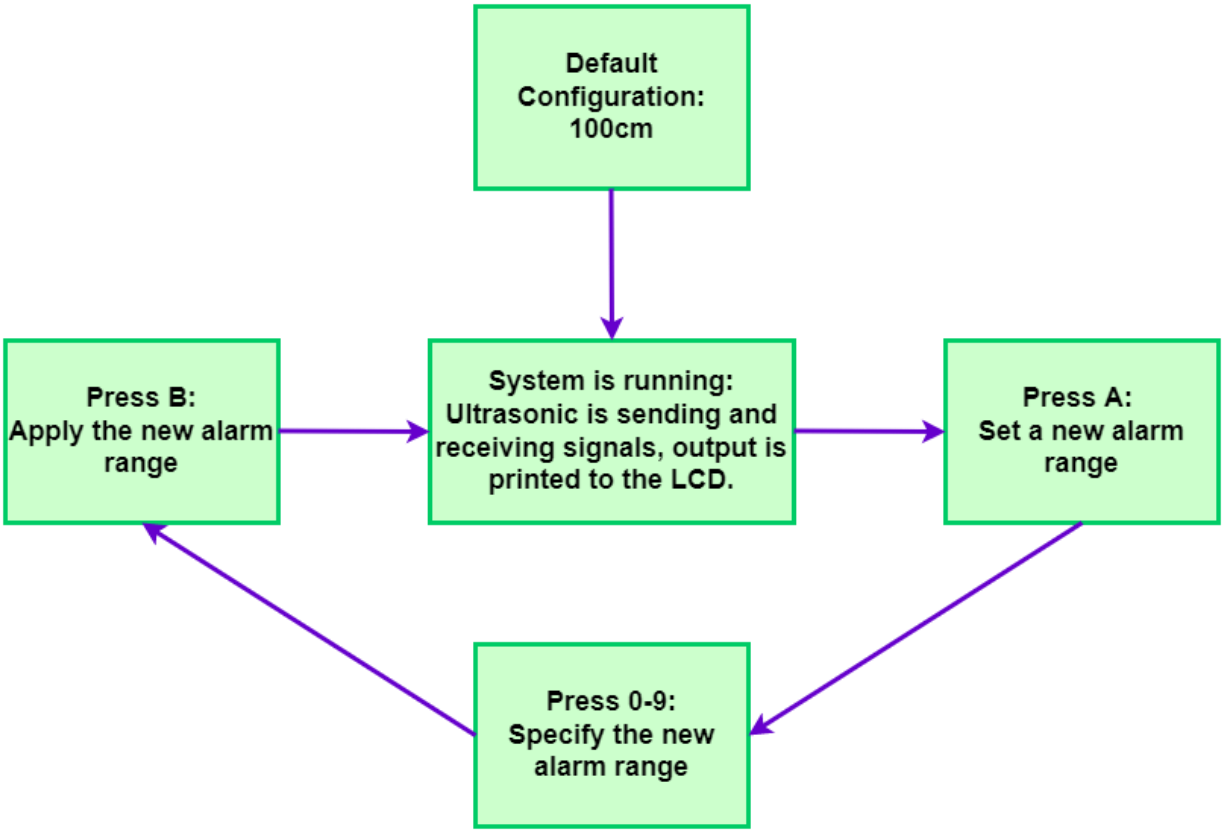
- **Threads/Tasks**
  - There are two threads used in the system: `t1` for the ultrasonic sensor and `t2` for the matrix keypad. Event Queues `q1` and `q` are attached to each thread.
    - `q1` is attached to the `Ticker tick`, which calls the ISR `isr_start` every 500ms, which sends a trigger to the ultrasonic transducer,
    - q is used for the events related to the user pressing a button on the matrix keypad:
      - A `A_ChangeDist` is placed in q
      - B `B_CompInput` is placed in q
      - 0-9 `insertAlarmDist` is placed in q

- **Interrupt**
  - Interrupts are used for receiving an echo from the ultrasonic transducer and for determining the column of the key pressed from the matrix keypad.
  - The interruptIn `echo` from the HC-SR04 is set up to PA5 on the Nucleo. When a falling edge occurs on PA5 the ISR `isr_stop` is called. This allows printing to the LCD screen within the main loop and stops the timer `sentinel` that records the time between a sending a trigger and receiving an echo.
  - The interruptIns `col1`, `col2`, `col3` and `col4` are attached to PC11, PC10, PC9 and PC8 respectively. When any of these receive rising edges, this indicates that a circuit was completed in the matrix keypad due to a user pressing a button. The interruptIn that received the rising edge indicates the column of the key and the value of row within the `MatrixKeypad` object `mk` indicates the column.

Block Diagram



Functionality Diagram



Bill of Materials

Bill of Materials:

- Nucleo L4R5ZI
- LCD
- Matrix Keypad
- HC-SR04 (Ultrasonic sensor)
- Buzzer
- Solderless Breadboard
- Jumper wires
- USB-A to micro-USB-B cable
- Computer to run Mbed Studio (Windows, MacOS or Linux versions available at https://os.mbed.com/studio/)

Data sheets and User Manuals:

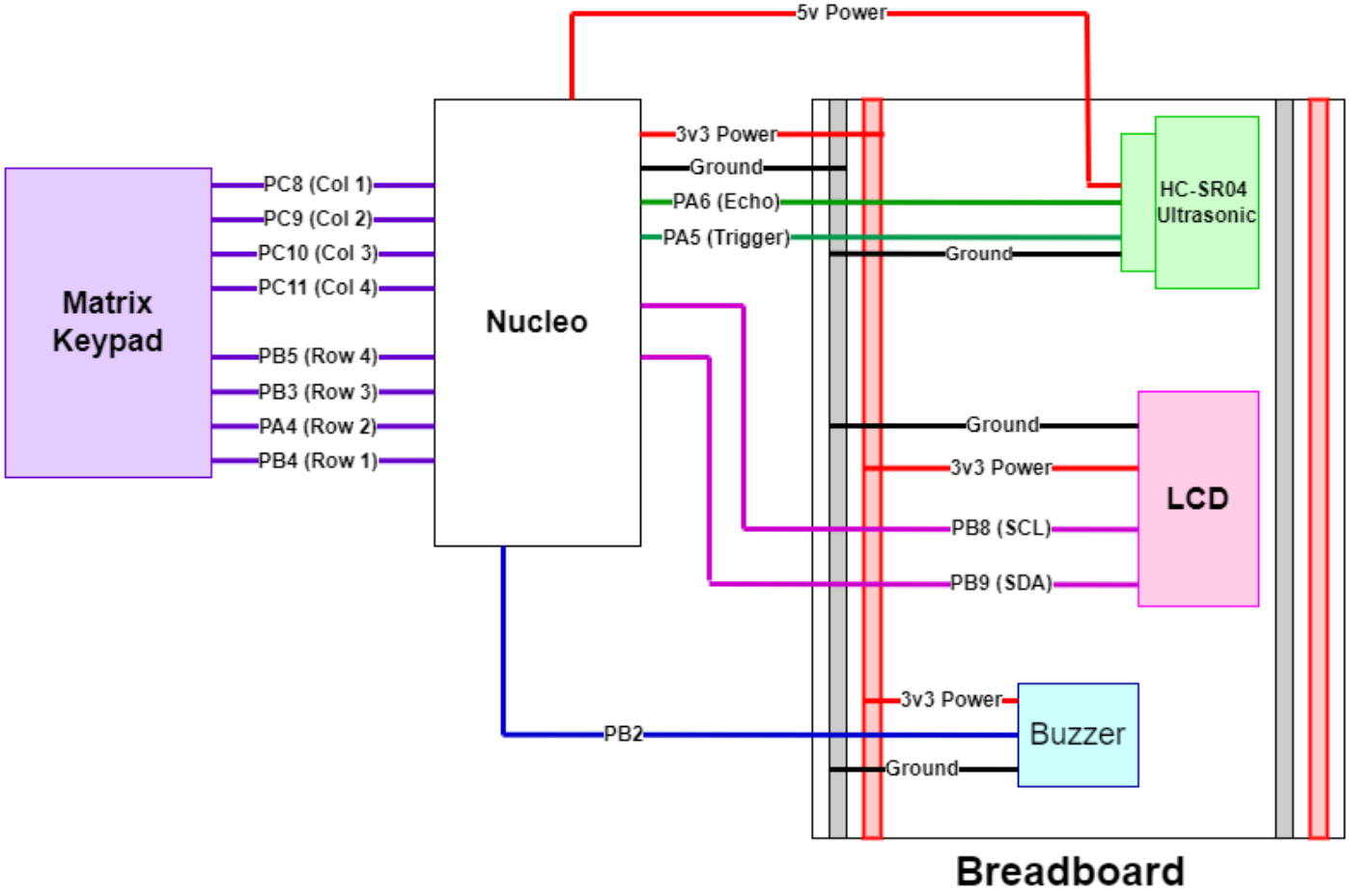
- Nucleo L4R5ZI:
  - UM2179
    - https://www.st.com/resource/en/user_manual/dm00368330-stm32-nucleo144-boards-mb1312-stmicroelectronics.pdf
  - RM0432
    - https://www.st.com/resource/en/reference_manual/rm0432-stm32l4-series-advanced-armbased-32bit-mcus-stmicroelectronics.pdf
- Ultrasonic Sensor:
  - https://www.electroschematics.com/wp-content/uploads/2013/07/HC-SR04-datasheet-version-2.pdf
  - https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
- Buzzer:
  - http://tinkbox.ph/sites/tinkbox.ph/files/downloads/5V_BUZZER_MODULE.pdf

Additional References:

- Referred during implementation for ultrasonic transducer functionality:
  - https://os.mbed.com/components/HC-SR04/
- Much of the code for the matrix keypad was adapted from Project 2 in CSE321_project2_naminor_main.cpp:
  - https://github.com/CSE321-Fall2021/cse321-portfolio-naminor/blob/main/Project%202/CSE321_project2_naminor_main.cpp

User Instructions

Schematic



How to Build the System

Connect **3v3 power** and **ground** pins to the breadboard **rails**.

Connect the following pins to the **Matrix Keypad**:

Rows

- Row 1: **PB4**
- Row 2: **PA4**
- Row 3: **PB3**
- Row 4: **PB5**

Columns

- Column 1: **PC11**
- Column 2: **PC10**
- Column 3: **PC9**
- Column 4: **PC8**

Connect the following pins to the **LCD**:

- SDA: **PB9**
- SCL: **PB8**

- VCC: **3v3 Power** from breadboard rail
- GND**:** **Ground** from breadboard rail

Plug the buzzer into the breadboard and connect the following pins to the **buzzer** in the associated group of five:

- I/O: **PB2**
- VCC: **3v3 Power** from breadboard rail
- GND**:** **Ground** from breadboard rail

Plug the buzzer into the breadboard and connect the following pins to the **HC-SR04** (ultrasonic transducer) in the associated group of five:

- VCC: **5v Power** from Nucleo to group of five
- GND: **Ground** from breadboard rail
- ECHO: **PA6**
- TRIG: **PA5**

How to Use the System

Upon startup, the system will begin operation assuming the default configurations and the LCD will light up. On the LCD, the current distance to the closest object in the path of the ultrasonic sensor will be displayed next to the default alarm range of 100cm. If that distance is less than 100cm, the buzzer will also be ringing. For example:

```
If:       145 / 100 cm              Buzzer is silent
If:        35 / 100 cm              Buzzer is ringing
```

To change the alarm range, press the A key on the matrix keypad. The 0-9 keys can be pressed to specify the new range and the value displayed on the LCD screen will update accordingly. Press the B key to apply the change.

In the following example, the nearest object is 63 cm away and the range was changed from the default of 100 cm to 50 cm.

```
Before:    63 / 100 cm              Buzzer is silent
After:     63 / 50 cm               Buzzer is ringing
```

Test plan

Configure Matrix Keypad

- Create Interrupt Service Routines for each column of the keypad.
- Assign each column to a pin and create InterruptIn objects for each column.
- Have ISR be triggered on a rising edge for any keypress.

Configure LCD Screen

- Place 1802.h and 1802.c in project folder. Include 1802.h in main file.
- Create CSE321_LCD object with the number of columns and rows on the display, dots, SDA pin and SCL pin.
- Create a class that is used to hold and update data that will be printed to the LCD screen.

Configure Buzzer

- Assign a pin to send an output signal to the buzzer.
- Create functions to toggle between outputting a signal and not.

Configure HC-SR04

- Assign pins to the trigger and echo.
- Set up the trigger to be sent periodically.
- Set up an InterruptIn object to detect when an echo is received.
- Create a Timer that will track the time between a trigger and an echo.

Development Timeline and Revision History

- Created `IO_Interface` class that facilitates interaction between the user, the inputs, and the outputs. It holds data and contains methods used by peripherals including the ultrasonic transducer, buzzer, LCD screen and matrix keypad.
- Implemented functionality for the ultrasonic transducer using interrupts for the echo and a Ticker for the trigger. Set up interaction between the ultrasonic sensor and the LCD screen using methods written in the `IO_Interface` class.
- Created `MatrixKeypad` class in a separate file to make the code more organized. This introduced large delays and errors in determining which row was pressed. Putting ISR's in main instead eliminated this issue by reducing the latency. The `MatrixKeypad` class was still used to hold the current row and to set up the MODER configurations for each row.
- Originally planned to allow for an inches mode on the display in addition to the existing centimeters mode. Abandoned this as it would add unnecessary complexity to the system in return for little utility. This elimination freed up time to make the core parts of the system more feature rich and reliable.
- A future improvement to the system is the inclusion of an IR sensor. This would help compensate for the ultrasonic sensor's poor detection of non-flat, small, and distant objects.