Queen Mary
University of London

EMBEDDED SYSTEMS

Project Lab C

"Number Input using LCD"

Lecturer: William Marsh

Surname	Cruz Felix	**First name**	Frank
Student number	150684871		
Project Letter	C	**Project title**	Number Input using LCD
GIT Hub Link	https://github.research.its.qmul.ac.uk/ECS642-714-EmbeddedSystems/bt15585ProjectC.git		

December 5th, 2018

1 Information about your System Requirements

The aim of this project is to use the LCD and the buttons on the LCD shield to input or select a three digital number. The system has achieved the following requirements:

Given Requirement	Status	Detailed Behaviour
Enabling the use of the four buttons on the LCD shield (Up, Down, Right, Left)	Fully Implemented	The software developed allows users to interact with the keys (Left, Right, Up, Down) of the LCD shield. This requirement was achieved by reading the voltage from the keys and then to identify the key press, intervals were created to match the reading measured, as this is an analog one.
Compose a 3-digit number.		The system allows users to input numbers between 000 and 999.
The left and right keys move from one number to another	Fully Implemented	The left and right key move from one number to another and cannot go over a number before the left-most digit and for the right-most digit creates an empty space in order to play the audio tone. This allows the user to get back to the right most digit in order to set a new frequency.
The up and down keys change the values of the current digit.	Fully Implemented	The up and down keys change the values of the current digit from the range 0 to 9 if the up key was pressed. If the down key was pressed, the value will loop from 9 to 0. These values are stored in a char array.
Text is displayed to prompt the user to enter a number	Fully Implemented	The systems displays “Enter frequency” (Enter number) to prompt the user to input a number.
Pressing the “Right” key when the cursor is positioned over the right-most digit, causes the digits currently displayed to be “entered” as a number.	Fully Implemented	The digits input are constantly stored and updated after any changes in an array according to their position (less most digit and right-most digit). This array will be the output of the number. Example, position[shift] = digit[*updown]; //array storing the current digit according to the position.
Software to enter a number allows to be included in another system	Fully Implemented	The system has been developed developed in a form that it can be implemented in another system.
The functionality to enter a number should be presented through a simple API	Fully Implemented	This requirement has been achieved by splitting the different tasks into small sub-functions.
Demonstration by using the functionality to set the frequency in Hz of an audio tone.	Fully Implemented	The audio tone was calculated by using the number input (frequency) and the following formula: T/2 = 10485760/(frequency)*2

		After getting the PIT value, the tone played will be accordingly to it.
Error displayed and the frequency does not change	Fully Implemented	Pressing the right key when the cursor is positioned over the right-most digit, the task3finalDisplay will display either of these messages: “Audiotone Played” or “Error, try again” depending on the input.

2 Design System

2.1 Overview

The system is a cyclic system, which does not use the ISR (Interrupt Service Routine) or Interrupt handler, meaning that it polls for state changes and responds to them. This system implements the SysTick counter to generate a periodic interrupt every 10 ms.

The system allows users to input digits, which are then converted into a 3-digit number if the user press the RIGHT key in the right-most digit. The values of each digit can be incremented or decremented. The position of the cursor can only barrel between the three different positions assigned ("000").

2.1 Peripherals and Pins

From the ADC:

- ADC_CHANNEL (8): Freedom board KL25Z ADC channel on Port B pin 0 (PTB0), the number into brackets means the channel using.

  ADC0_SE8 → PTB0/J10, 2

  This pin used to read analog signals from the board.

  From the LDC:

- Pins for the LCD are as follows:
  LCD_EN (5)   // on port D
  LCD_RS (13)   // on port A
  LCD_DB7 (9)   // on port C
  LCD_DB6 (8)   // on port C
  LCD_DB5 (5)   // on port A
  LCD_DB4 (4)   // on port A

- The speaker uses 5VCC and a ground pin and the PWM has been connected to the PORTD pin 6. Pulse Width Modulation (PWM) was not used in this software.

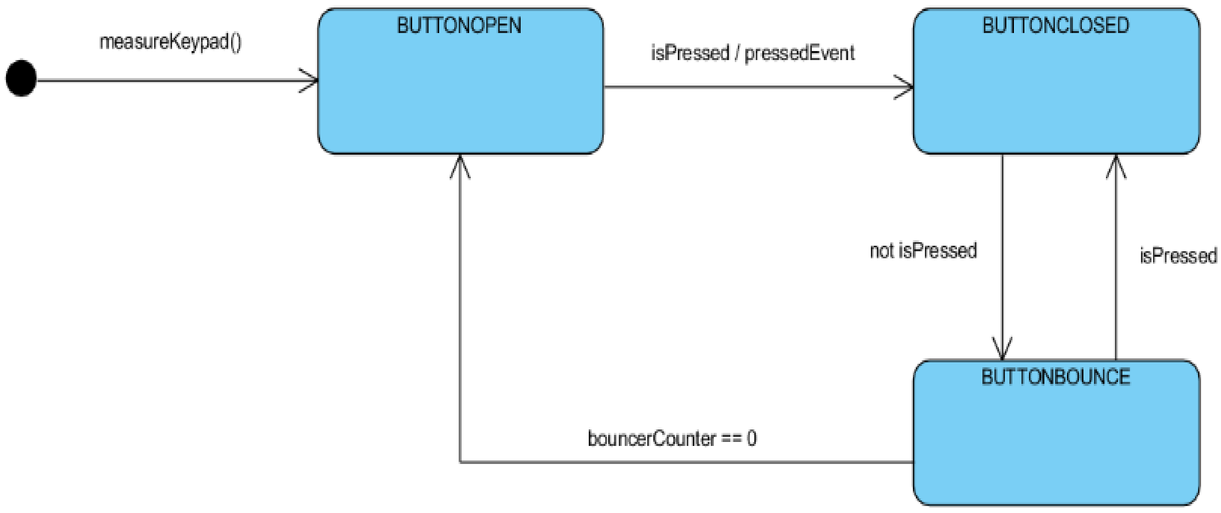
  AUDIO_POS (6) // PORTD pin 6

2.2 *Task 1: task1ButtonPressed*

The first task detects button press events. This task polls the button to see if one of them has been pressed. The button is initially up (BUTTONUP). When the button is pressed generates a pressed event (BUTTONCLOSED). The system will stay in the same state "down" as long as it the key is pressed. Once the button is released, the isPressed function will update the value of the variable button pressed and return true if the voltage read belongs to any of those intervals. Then this goes to the bouncing state (BUTTONBOUNCE). This function updates the values of the array created for this task, which are the state, pressed and bouncerCounter. The array state stores the current state of the button (BUTTONOPEN, BUTTONCLOSED, and BUTTONBOUNCE). The array pressed stores the press event, which will be either 0 or 1 and the array bouncerCounter stores the counter value.

MeasureKeypad() function is called in order to detect which key was pressed. This function returns an integer variable containing the number (zero, one, two or three) of the specific button.

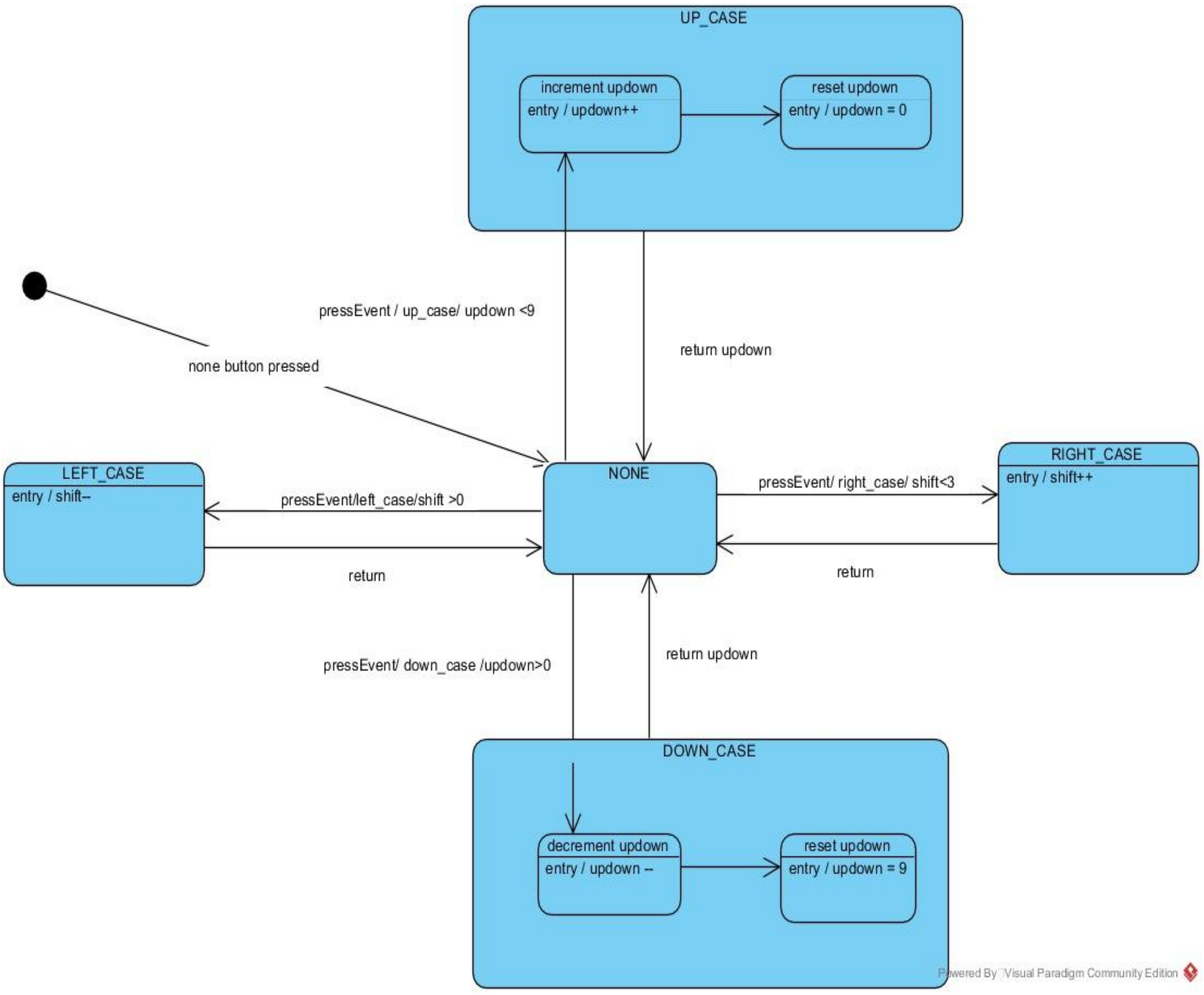
The state diagram for this task is as follows:



The state diagram above is the representation for a single key press performance. However, the four keys (UP, DOWN, RIGHT, LEFT) perform same as the other ones.

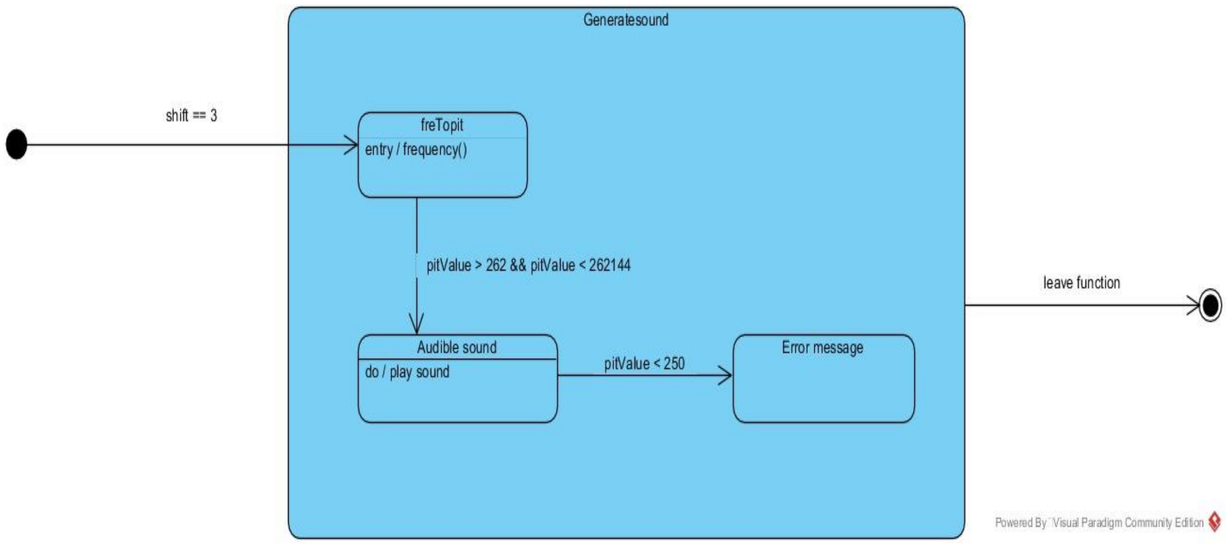
2.3 Task 2: task2StateLCD

This function performs depending on which key has been pressed. The leftFunction and rightFunction will move the cursor from one position to another for the corresponding range. The upFunction and downFunction will change the values of the current digit (between 0 and 9). Three integer variables (updown1, updown2, updown3) have been created to store the values of the digits on each position.



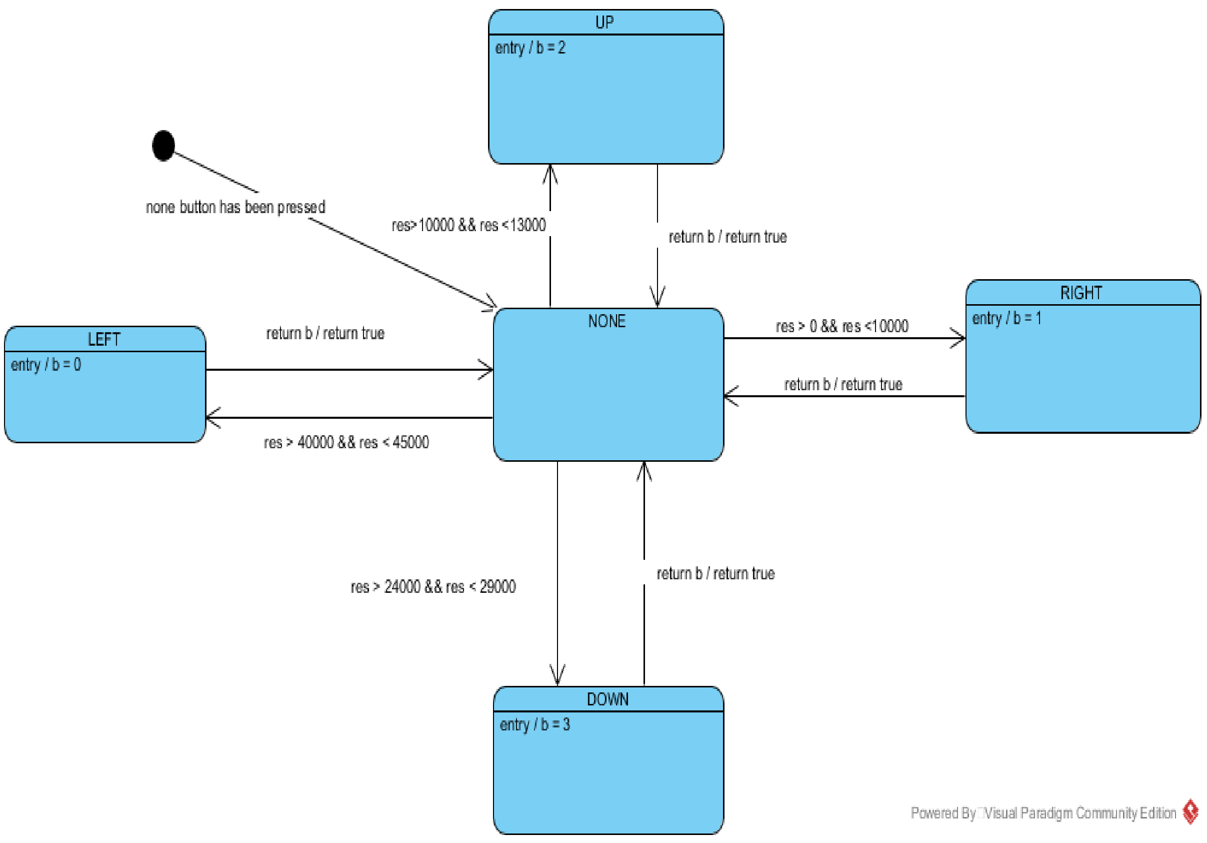
2.4 *Task 3: task3makeInteger*

This task converts the digits entered into a 3-digit number. The conversion is performed by calling the function frequency, which makes the takes the values updown1, updown2 and updown3 and makes a mathematics operation. If this new 3-digit number is less than 2, which is a frequency below the audible range (20hz to 20KHz), otherwise it will return the value converted as a PIT (Programmable interval Timer) value.



2.5 *Task 4: isPressed*

The isPressed Boolean function matches the right button according to the voltage readings obtained from the measureKeypad function. The form this function "isPressed" works is as follows:


UP
entry / b = 2
none button has been pressed
res>10000 && res <13000
return b / return true
RIGHT
entry / b = 1
NONE
res > 0 && res <10000
LEFT
entry / b = 0
return b / return true
return b / return true
res > 40000 && res < 45000
res > 24000 && res < 29000
return b / return true
DOWN
entry / b = 3
Powered By Visual Paradigm Community Edition