**ELEC 291 Section 20C**

Lab 3

L2C

Team 2A

*Student name Student number Contribution percentage*

Andy Ruan 36863141 33.3%

Kevin Wong 32105132 33.3%

Clarence Su 36387132 33.3%

Contribution Summary:

Andy Ruan designed and wired the main circuit along with creating the schematic in fritzing. He also wrote the skeleton of and code for our custom LCD library except for the printing functions.

Kevin Wong wrote the code to print to the LCD and the code for the game logic. He also did the soldering for the LCD pins and organized the lab report.

Clarence Su wrote the code to read the number keypad along with the code to bring all the parts of the lab together, utilizing our own LCD library.

**Introduction and motivations**

This report outlines the process in which Team 2A used to complete lab 3 of ELEC 291-20C. It is organized in the following manner: First, it will describe the lab and its sub-components. The next part will describe the conclusion. Appendix I will include an image of the fritzing breadboard schematic. Appendix II will discuss the Arduino code used in the lab.

The objective of this lab is to interface the LCD and use a numeric keypad to play a game called “Guess the Number”. Through this lab, students will improve their coding skills by writing their own LCD functions and Keypad functions.

**Lab Description**

Lab 3 was divided into four parts:

1. Display on the LCD
2. Adding the keypad
3. Creating the Guess the Number Game
4. Bonus features

After implementing all four parts, the result will be a completed circuit that uses a numeric keypad to input values and a LCD to display information to user such as number inputted and game instructions to play a game of Guess the Number.

For part one, our team first soldered the head connector to the LCD. Then the team wrote code for the functionality of the LCD such as increment cursor, print character, and clear screen. For the LCD circuit, the team decided to use 4 digital pins instead of 8 in order to have the circuit look clean and presentable. This design caused the team to write software that handles readings of bits of 4. The team was able to solve the problem by sending the first four bits then writing to enable to change it to high then writing to enable to set it to low and finally repeating the same process for the remaining four bits.

The team used the LiquidCrystal library functions to test if the soldering job and wiring of the circuit was functional. Originally our team was planning to complete the entirety of the lab with the library and then proceed to replace the library functions with our own, maintaining the same specifications. However, we quickly realized that we did not need as much versatility in our functions and ended up coding our basic the LCD commands first and utilizing them to complete the lab.

A problem encountered while working with the LCD was matching the data pins to the proper command values. The reason for this was due to the the fact that the command read from d7 to d0 while the pins on the lcd size read from d0 to d7. Luckily our code allowed for an easy fix as we only needed to reverse the pin constants at the top. An obstacle encountered while debugging the LCD at first was that due to the nature of needing to initialize the screen on power up. It was difficult to pinpoint whether a bug was within the commands or the initialization until we got an expected result for a command which would imply that the initialization worked. Significant time was also used to understand the behaviour of the LCD when the cursor reached the end of a row due to printing, moving the screen or moving the cursor itself.

For part two, the team implemented the numeric keypad by using 7 digital pins of the Arduino. The team implemented a 2-D array that stores the key numbers and wrote a loop to scan the array to determine which key on the keypad is clicked by determining if the element is LOW. The loop is looking for LOW because the pins are using pull-up resistors where LOW indicates that a key is pressed (a closed circuit) and HIGH indicates a key is not pressed (an open circuit).

A problem encountered with the keypad was not implementing the debounce time causing the user to not be able to input consecutive keys. After debugging and implementing the debounce time, the problem was solved.

A design consideration that the team decided for the numeric keypad was to accept the top and leftmost number pressed on the keypad when user clicks on multiple keys exactly at the same time. For instance if the user clicked on 1, 4 and 7 at the same time the input would be 1 and if the user clicked on 1, 2 and 3 at the same time the input would be 1. Other implementations were possible such as ignoring every other key except the first key and accepting all values. However, our team felt that implementing our solution was much more clear and simpler than other solutions.

For part three, the team implemented a simple function that would that would be called in the loop of the Arduino code to indicate that the player wants to play Guess the Number game. It used a random number generator and the modulo of that number with 10 generates a number of 1 to 9. The user input from the keypad would be recorded and compared to the number generated by the random number generator. If the number generated is greater than the guess number, the user will lose an attempt and will need to guess again. If the number generated is less than the guess, the user will lose an attempt and will need to guess again. The player can play the game until the user loses all of their attempts, which is three in this game.

A problem was having the random number generator inside the game logic function. Since the function is inside a loop, the number would change each time the player plays the game. As a result, the team decided to implement the random number generator outside the function in order to stop this problem. The team tested the game by using the numeric keypad and LCD to play the game. Using Serial.println() to determine the input the and the number generated then comparing the expected output with the actual output.

For the bonus part, the team implemented the challenge mode, which allows users to guess a number between 0 and 99 for the “Guess the Number” game. The team implemented a choice for the user to select which difficulty they wanted to play as well as adding another random number generator, which generates numbers between 0 and 99 for the user to guess. In addition, the team added a Piezo buzzer and transformed the keypad into musical keys by determining the frequency for musical notes and calling the tone() function when user clicks a key on the keypad. Clicking one key will produce a tone and clicking another key will produce a second tone.

**Conclusions**

Lab 3 provided students the opportunity to write their own library functions for the Keypad and LCD. In addition to writing library functions, students also learned to solder by soldering the header connector to the LCD.

Our team learned important lessons from this lab. Firstly, for future labs, the team should construct the circuit earlier instead of starting the circuit in the lab session. This provides the team with extra time to write and debug code together instead of working in parallel and merging the code only to see it fail due to small issues.

Secondly, good documentation and version control of the code was extremely important due to the scale of the lab. The team had a lot of code to write and was writing them separately, so when merging the code together there were problems with pin assignments. Having good documentation and version control helps with debugging and solving problems quicker.

**References**

[1] http://www.rapidtables.com/code/text/ascii-table.htm

**Appendix I**

Below is a schematic of the circuit used to demo lab 3, including the bonus, produced in Fritzing.

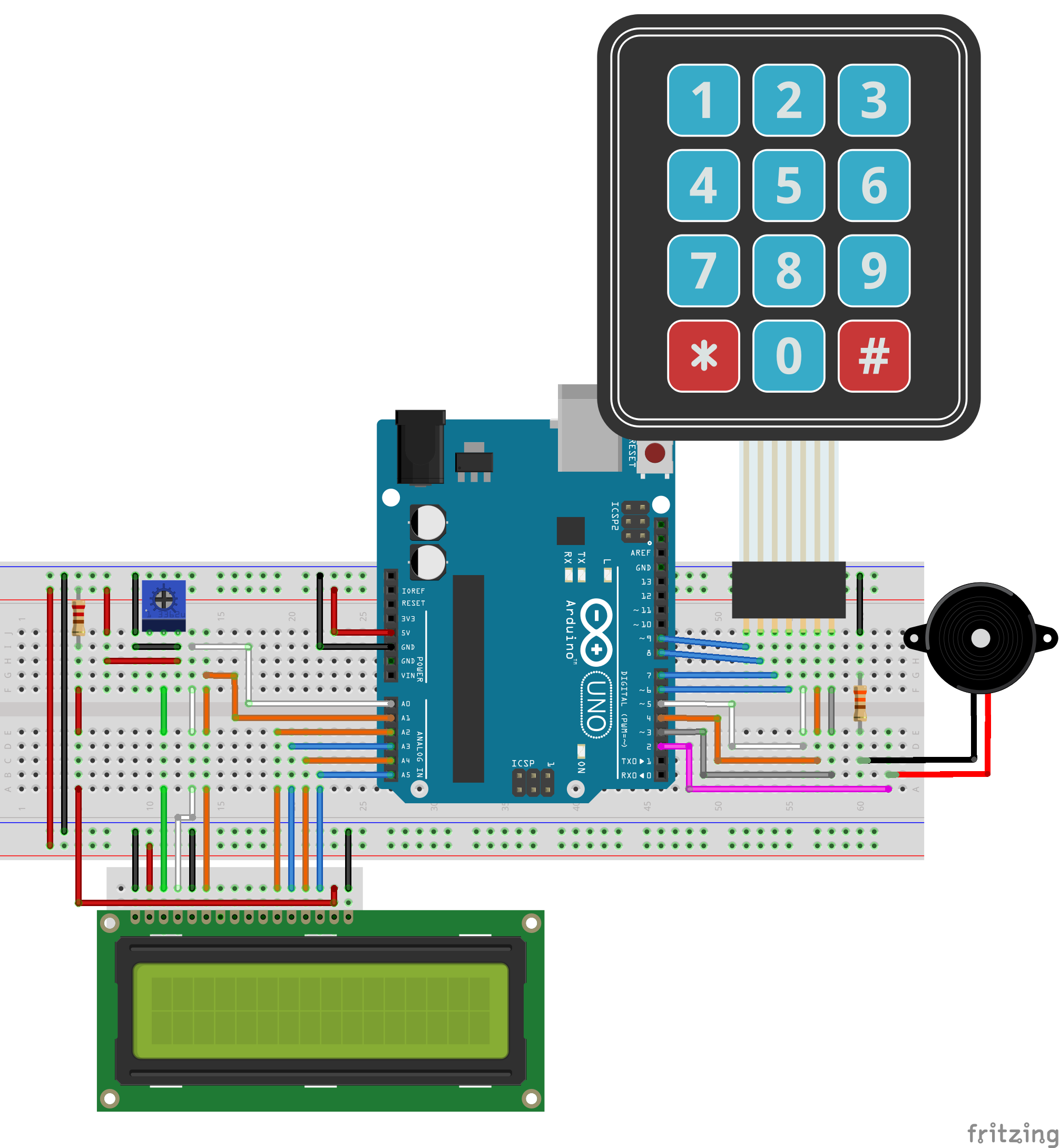


Figure 1: Fritzing schematic of the circuit used in lab 3

**Appendix II**

Below is the code written for Lab 3.

//pins assignment

// LCD analog pins

const int LCDrs = A0;

const int LCDenable= A1;

const int LCDdata[4]={A5,A4,A3,A2}; // d7, d6, d5, d4

// R/W to ground

// VSS to ground

// VCC to ground

// 10K pot, ends to +5V and ground

const int KeyRowP[]={3,4,5,6};

const int KeyColP[]={7,8,9};

const int Buzzer = 2;

//local variable

char keyPad[4][3] = {

{'1','2','3'},

{'4','5','6'},

{'7','8','9'},

{'\*','0','#'}

};

char Binary[][9] =

{

"01000001", //A 0

"01000010", //B 1

"01000011", //C 2

"01000100", //D 3

"01000101", //E 4

"01000110", //F 5

"01000111", //G 6

"01001000", //H 7

"01001001", //I 8

"01001010", //J 9

"01001011", //K 10

"01001100", //L 11

"01001101", //M 12

"01001110", //N 13

"01001111", //O 14

"01010000", //P 15

"01010001", //Q 16

"01010010", //R 17

"01010011", //S 18

"01010100", //T 19

"01010101", //U 20

"01010110", //V 21

"01010111", //W 22

"01011000", //X 23

"01011001", //Y 24

"01011010", //Z 25

"00110000", //0 26

"00110001", //1 27

"00110010", //2 28

"00110011", //3 29

"00110100", //4 30

"00110101", //5 31

"00110110", //6 32

"00110111", //7 33

"00111000", //8 34

"00111001", //9 35

"00100110", //& 36

"00101010", //\* 37

"00100001", //! 38

"00101101", //Hyphen 39

"00100011", //# 40

"00101110", //Period (.) 41

"00100000", //Space 42

"01100001", //a 43

"01100010", //b 44

"01100011", //c 45

"01100100", //d 46

"01100101", //e 47

"01100110", //f 48

"01100111", //g 49

"01101000", //h 50

"01101001", //i 51

"01101010", //j 52

"01101011", //k 53

"01101100", //l 54

"01101101", //m 55

"01101110", //n 56

"01101111", //o 57

"01110000", //p 58

"01110001", //q 59

"01110010", //r 60

"01110011", //s 61

"01110100", //t 62

"01110101", //u 63

"01110110", //v 64

"01110111", //w 65

"01111000", //x 66

"01111001", //y 67

"01111010", //z 68

"00111111" //? 69

};

int debounceTime = 1000;

int numArray[] ={ LOW,HIGH };

int LCDDisplay = HIGH;

int LCDCursor = LOW;

int LCDCursorBlink = LOW;

String lcdOffSet=" ";

//global varialbe

int attempts=3;

char lastInput='a';

int randomNum1;

int randomNum2;

boolean gameOver=false;

char gameChoice;

char difficulty;

void setup() {

//KeyPad assignment

Serial.begin(9600);

for(int index=0; index<4; index++){

pinMode(KeyRowP[index], INPUT\_PULLUP);

}

for(int index=0; index<3; index++){

pinMode(KeyColP[index], OUTPUT);

}

//LCD assignment

pinMode(LCDrs, OUTPUT); // initialize lcd pins

pinMode(LCDenable, OUTPUT);

pinMode(Buzzer, OUTPUT);

for (int i = 0; i < 4; i++) {

pinMode(LCDdata[i], OUTPUT);

}

lcdInit(); // initialize lcd to 4-bit mode

lcdClear();

//welcome information display

newGame();

lcdBlink();

welcomeMessage();

}

void loop() {

lcdCursorHome();

lcdNoAutoScroll();

//mainMenu

mainMenu();

//entering game function according user's choice

if(gameChoice=='2'){

lcdClear();

delay(1000);

lcdPrint("playing ");

//reset the debounceTime to improve the piano game

debounceTime=100;

while(!gameOver){

playPiano();

}

debounceTime=1000;

}

//keep running the game function until it is game over

while(!gameOver){

//display game rules to aid the user

if(difficulty=='1'){

lcdPrint("Guess a number ");

lcdPrint(lcdOffSet);

lcdPrint("From 0-9 ");

while(!gameOver){

playNumGame(convertToInt(readKey()),randomNum1); //take the user's input and the random number and see if they matched

}

}

else if(difficulty=='2'){

lcdPrint("Guess a number ");

lcdPrint(lcdOffSet);

lcdPrint("From 0-99 ");

while(!gameOver){

playNumGame(readDoubleDigt(),randomNum2); //take the user's input and the random number and see if they matched

}

}

}

lcdClear();

//ask user if want to go back to the main meno

lcdPrint("input \* to go ");

lcdPrint(lcdOffSet);

lcdPrint("back menu ");

while(gameOver){

if(readKey()=='\*'){

gameOver=false;

newGame(); //update and reset all the game parameter

}

}

}

/\*

\* Section for KeyPad

\*/

//parameter: none

//return the key pressed by the user

char readKey(){

int startTime = millis();

int col, row;

//read the pins of Keypad

//implemented a super loop to keep keypad listening to the input

while(true){

boolean noKey=true;

//when there is no input keep the arduino scanning for the keys

while(noKey){

for(int index=0; index<3; index++){

digitalWrite(KeyColP[index], HIGH);

}

for(int index=0; index<3; index++){

digitalWrite(KeyColP[index], LOW);

for(int rowIndex=0; rowIndex<4; rowIndex++){

if(digitalRead(KeyRowP[rowIndex])==LOW){

row=rowIndex;

col=index;

noKey=false;

break;

}

}

if(!noKey) break;

}

delay(25);

}

if(keyPad[row][col]!=lastInput||(millis()-startTime>=debounceTime))break;

//prevent the arduino from reading too many inputs. It only read new input when the new input does not equal to last input or exceeds the debounce time

}

//return the result

lastInput=keyPad[row][col];

return keyPad[row][col];

}

//read 0~99 from user

int readDoubleDigt(){

char input[4];

int returnNum;

boolean finished = false;

//keep reading until the user finish input

while(!finished){

for(int index = 0; index < 4; index++){

input[index]=readKey();

if(input[index]=='#'){

finished = true;

break;

}

}

}

if(input[1]=='#'){

returnNum = convertToInt(input[0]);

}

else{

//convert the char array to integer

returnNum =convertToInt(input[0])\*10 + convertToInt(input[1]);

}

return returnNum;

}

/\*

\* LCD Section

\*/

// LCD FUNCTIONS - LOW/MID LEVEL

// Flashes the LCDenable pin to read instructions/data

void flashEnable() {

digitalWrite(LCDenable, LOW);

delay(1);

digitalWrite(LCDenable, HIGH);

delay(1);

digitalWrite(LCDenable, LOW);

delay(1);

}

// Sets the LCDdata pins

void setLCDPins(int rs, int d3, int d2, int d1, int d0) {

digitalWrite(LCDrs, rs);

digitalWrite(LCDdata[0], d3);

digitalWrite(LCDdata[1], d2);

digitalWrite(LCDdata[2], d1);

digitalWrite(LCDdata[3], d0);

}

// Sends the data/instruction to the lcd and flashes enable

void sendCode(int rs, int d7, int d6, int d5, int d4, int d3, int d2, int d1, int d0) {

setLCDPins(rs, d7, d6, d5, d4);

flashEnable();

setLCDPins(rs, d3, d2, d1, d0);

flashEnable();

}

// Increments the cursor position to the right (moves to next line on the left if end of line)

void incrementCursor() {

sendCode(LOW, LOW, LOW, LOW, HIGH, LOW, HIGH, HIGH, LOW);

}

// Decrements the cursor position to the left (moves to previous line on the right if end of line)

void decrementCursor() {

sendCode(LOW, LOW, LOW, LOW, HIGH, LOW, HIGH, LOW, LOW);

}

// Shifts entire display to the right, cursor follows

void displayRight() {

sendCode(LOW, LOW, LOW, LOW, HIGH, HIGH, HIGH, LOW, LOW);

}

// Shifts entire screen to the left, cursor follows

void displayLeft() {

sendCode(LOW, LOW, LOW, LOW, HIGH, HIGH, LOW, LOW, LOW);

}

// Prints a single char onto the lcd

void printChar(char c) {

//Convert character to ASCII number

int value = (int) c;

//ASCII for 0-9

if(value >= 48 && value <= 57){

value = value - 22;

getCharacter(value);

}

//ASCII for A-Z

else if(value >= 65 && value <= 90){

value = value - 65;

getCharacter(value);

}

else if(value >= 97 && value <= 122){

value = value - 54;

getCharacter(value);

}

//ASCII for period

else if(value == 46){

getCharacter(41);

}

// ASCII For exclamation

else if(value == 33){

getCharacter(38);

}

// ASCII For &

else if(value == 38){

getCharacter(36);

}

//ASCII For \*

else if(value == 42){

getCharacter(37);

}

//ASCII for - (Hyphen)

else if(value == 45){

getCharacter(39);

}

//ASCII for #

else if(value == 35){

getCharacter(40);

}

//ASCII for Space

else if(value == 32){

getCharacter(42);

}

//ASCII for ?

else if(value == 63){

getCharacter(69);

}

}

int convertToInt(char character){

int aNumber = character - '0';

return aNumber;

}

void getCharacter(int value){

String inputNumber = Binary[value];

sendCode(HIGH, numArray[convertToInt(inputNumber.charAt(0))],

numArray[convertToInt(inputNumber.charAt(1))],

numArray[convertToInt(inputNumber.charAt(2))],

numArray[convertToInt(inputNumber.charAt(3))],

numArray[convertToInt(inputNumber.charAt(4))],

numArray[convertToInt(inputNumber.charAt(5))],

numArray[convertToInt(inputNumber.charAt(6))],

numArray[convertToInt(inputNumber.charAt(7))]

);

}

// LCD FUNCTIONS - HIGH LEVEL

// Initializes the lcd to 4-bit mode at power on

void lcdInit() {

delay(50); // wait for VDD

for (int i = 0; i < 3; i++) { // repeat x3

setLCDPins(LOW, LOW, LOW, HIGH, HIGH);

flashEnable();

delay(40);

}

setLCDPins(LOW, LOW, LOW, HIGH, LOW); // 4-bits mode

flashEnable();

setLCDPins(LOW, HIGH, LOW, LOW, LOW); // display lines and font

flashEnable();

setLCDPins(LOW, HIGH, LOW, LOW, LOW); // display off

flashEnable();

setLCDPins(LOW, LOW, LOW, LOW, HIGH); // display clear

flashEnable();

setLCDPins(LOW, LOW, HIGH, HIGH, LOW); // increment cursor, no display shift

flashEnable();

}

// Clears the LCD screen and returns the cursor home

void lcdClear() {

sendCode(LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW, HIGH);

delay(5);

}

// Returns the cursor home

void lcdCursorHome() {

sendCode(LOW, LOW, LOW, LOW, LOW, LOW, LOW, HIGH, LOW);

delay(5);

}

// Turns ON the display

void lcdDisplay() {

LCDDisplay = HIGH;

sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);

}

// Turns OFF the display

void lcdNoDisplay() {

LCDDisplay = LOW;

sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);

}

// Displays the cursor

void lcdCursor() {

LCDCursor = HIGH;

sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);

}

// Hides the cursor

void lcdNoCursor() {

LCDCursor = LOW;

sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);

}

// Blinks the cursor

void lcdBlink() {

LCDCursorBlink = HIGH;

sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);

}

// Stops blinking the cursor

void lcdNoBlink() {

LCDCursorBlink = LOW;

sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);

}

// Turn ON scrolling when writing

void lcdAutoScroll() {

sendCode(LOW, LOW, LOW, LOW, LOW, LOW, HIGH, HIGH, HIGH);

}

// Turn OFF scrolling when writing

void lcdNoAutoScroll() {

sendCode(LOW, LOW, LOW, LOW, LOW, LOW, HIGH, HIGH, LOW);

}

// Increments cursor position k times

void lcdMoveCursorRight(int k) {

for(int i = 0; i < k; i++) {

incrementCursor();

}

}

// Decrements cursor position k times

void lcdMoveCursorLeft(int k) {

for(int i = 0; i < k; i++) {

decrementCursor();

}

}

// Scrolls the display to the right k times

void lcdScrollRight(int k) {

for (int i = 0; i < k; i++) {

displayRight();

}

}

// Scrolls the display to the left k times

void lcdScrollLeft(int k) {

for (int i = 0; i < k; i++) {

displayLeft();

}

}

// Prints a string onto the lcd

void lcdPrint(String s) {

for(int i = 0; i < s.length(); i++){

printChar(s.charAt(i));

}

}

/\*

\* Game logic section

\*/

void welcomeMessage(){

//welcome Message

String welcomeStringp1="Hello & welcome ";

String welcomeStringp2="to our demo!";

String welcomeStringp3="ELEC 291-20C Team L2C-2A ";

String welcomeStringp4="am L2C-2A ";

//welcome message

lcdClear();

lcdNoAutoScroll();

lcdPrint(welcomeStringp1);

lcdPrint(lcdOffSet);

lcdPrint(welcomeStringp2);

delay(2000);

//team information

lcdClear();

lcdNoAutoScroll();

lcdPrint(welcomeStringp3);

lcdPrint(welcomeStringp4);

lcdMoveCursorRight(60);

delay(2000);

lcdAutoScroll();

for(int index=0; index<welcomeStringp3.length()-16;index++){

lcdPrint(" ");

delay(500);

}

delay(1000);

}

void mainMenu(){

boolean selected=false;

String menuString= "Choose a game by";

String menuString1= "push the number";

String menuString2= "1.Guess Number ";

String menuString3= "2.Piano ";

String menuString4= "Select a ";

String menuString5= "difficulty ";

String menuString6= "1 for 0-9 ";

String menuString7= "2 for 0-99 ";

//ask user to select a game

lcdPrint(menuString);

lcdPrint(lcdOffSet);

lcdPrint(menuString1);

delay(2000);

lcdClear();

lcdPrint(menuString2);

lcdPrint(lcdOffSet);

lcdPrint(menuString3);

while(!selected){

gameChoice=readKey();//set the global varialbe to the choice user made

if(gameChoice=='1'||gameChoice=='2'){

selected=true;

}

}

//ask user to input select a difficulty for the guessing game

if(gameChoice=='1'){

lcdClear();

lcdPrint(menuString4);

lcdPrint(lcdOffSet);

lcdPrint(menuString5);

delay(2000);

lcdClear();

lcdPrint(menuString6);

lcdPrint(lcdOffSet);

lcdPrint(menuString7);

boolean selectDiff=false;

while(selectDiff==false){

difficulty=readKey();

if(difficulty=='1'||difficulty=='2'){

selectDiff=true;

lcdClear();

}

}

}

}

//number game;

void playNumGame(int userInput, int randNum){

// Random number between 0 and 9

lcdClear();

if(attempts != 0){

if(userInput < randNum){

lcdPrint("Guess higher");

attempts--;

delay(1000);

}

else if(userInput > randNum){

lcdPrint("Guess lower");

attempts--;

delay(1000);

}

else{

lcdPrint("You won!");

delay(2000);

gameOver=true;

}

}

if(attempts == 0){

lcdClear();

lcdPrint("You lost!");

delay(2000);

gameOver=true;

}

}

//piano game

int note\_c4 = 262; //1

int note\_d4 = 294; //2

int note\_e4 = 330; //3

int note\_f4 = 349; //4

int note\_g4 = 392; //5

int note\_a4 = 440; //6

int note\_b4 = 494; //7

int note\_c5 = 523; //8

int note\_d5 = 587; //9

int note\_g5 = 784; //0

void playPiano(){

int pace = 255;

char input = readKey();

//buzzer the note matched the user input

switch(input){

case '1': tone(Buzzer, note\_c4, pace); break;

case '2': tone(Buzzer, note\_d4, pace); break;

case '3': tone(Buzzer, note\_e4, pace); break;

case '4': tone(Buzzer, note\_f4, pace); break;

case '5': tone(Buzzer, note\_g4, pace); break;

case '6': tone(Buzzer, note\_a4, pace); break;

case '7': tone(Buzzer, note\_b4, pace); break;

case '8': tone(Buzzer, note\_c5, pace); break;

case '9': tone(Buzzer, note\_d5, pace); break;

case '0': tone(Buzzer, note\_g5, pace); break;

case '\*': gameOver=true; break;

}

}

//start a new game and reset the game parameters

void newGame(){

//reset and update the game parameter

randomNum1 = rand() % 10;

randomNum2 = rand() % 100;

Serial.print(randomNum1);

attempts=3;

boolean gameOver=false;

lastInput='a';

}