**ELEC 291 Section 20C**

**LAB #: 3**

**LAB SECTION: 20D**

**TEAM #: D\_5C**

|  |  |  |
| --- | --- | --- |
| ***Student name*** | ***Student number*** | ***Contribution percentage*** |
| **LINDA MUNISI** | **22535158** | **33.3%** |
| **SIRINE TRIGUI** | **46443157** | **33.3%** |
| **YIXIN ZHAO** | **49427140** | **33.3%** |

*Contribution summary*:

We worked together to complete the lab. Since we were always together while working on the lab, every group mate contributed to almost every part of the project. All of us did the soldering by alternating two pins for each person(one on either side). Linda and Sirine were working on connecting the circuit while Yixin was writing the skeleton code for the commands. Then we all contributed and worked together to make the LCD display the characters at first. Then Linda and Yixin with the assistance of the instructor optimized the code to make it work better and show the message. Linda wrote the displaying of the two messages on the LCD, and got started on the keypad code while Yixin was working on implementing the game. Then Sirine wrote the code for making the message scroll and also fixed the bug that was in the game and made it work. Together we fixed the code for the keypad. We all demoed our project to the TA.

**A. Introduction and motivations**

**Introduction**

In this lab, the main objective was interfacing an arduino with standalone display. A numeric keypad was also used and linked to the arduino for more functions. The first part of the lab required setting up a display on the LCD. To do this, we were required to write initialization functions following the command table and the data sheet. The instructions were both 8 bits and 4 bits,the one we were tasked to use. We therefore had two different command functions, one that uses 4-bits and one that uses 8-bits and used them depending on the number of bits required by the command(the interface initially uses the 8-bit mode). The functions are commandLCD and commandLCD8 respectively.

For the second part, we were required to add a numeric keypad to the arduino. The function of the keypad was in the last part of the game where we used it to enter the values of the numbers that we were testing.

**Motivation**

The motivation for this lab was the desire to create a working game. Also we were motivated to know how a keypad works and how to write code that would make the keypad work by taking the numbers and ignoring the pound and star as error messages.

**B. Lab Description**

In the first part, we began with soldering the header connector to the LCD. We had to be very careful in this process, and very quick when soldering each pin because the high temperature may damage the pin. After connecting it correctly, we need to code for the LCD to display two messages properly. To make sure that our connection was right, we uploaded the code from the LiquidCrystal Library and uploaded it to the arduino. The code worked properly and the message “Hello World!” was displayed on the screen of the LCD. Then since we were not planning on using the library and coded ourselves, we started on writing the code. First, we followed the command table and the data sheet and wrote initialization functions for clear display, cursor home, cursor mode, entry mode set, display control and all the other functions in the command table. After initialization and passing the values corresponding to the functions that we needed to use, we realized that our LCD was displaying characters that were not corresponding to the value that we passed, ie the character that we wanted to display. We only could display o and \_. After observation we realized that when we pass the values of the character that are all high in the lower 4 bits, the character was successfully displayed but other than that all other characters were not displaying other than o and \_.

We tried to debug this problem for three days, tried writing different functions and passing different values for the respective command but we could not figure out the problem. But one of the larger problems we were facing here was how to set the address for the CGRAM and the DDRAM. We were wondering how to shrink the 8 bits to make them 6 or 7 since from the command table it shows that the address of the CGRAM is 6 bits and that of DDRAM is 7 bits. Then we went to see our instructor to ask for help. First, he told us that we did not necessarily need the functions for setting CGRAM address, DDRAM address and the function for reading the busy flag. The instructor suggested that we did not need these functions necessarily. We could implement the letters and the other functions without these commands. Therefore, we got rid of the functions.

Then we remained with the functions that correspond to the command that clears display, cursor mode, entry mode set, display mode and the function set. We tried used our commandLCD function to pass the value representing the character that we wanted displayed. We passed our functions twice with the first call indicating the first higher bits and the second representing the lower four bits since we were using a 4-bit interface. We did not see any changes and our LCD still displayed only o and \_. We went to see our instructor for more assistance

Our instructor looked at our code and suggested that we were making mistakes in the commandLCD in the way that we were converting the bits to make them 4 bits from 8-bits. He then suggested that we should have two functions, one that will deal with the commands that need 8 bits and another that deals with the commands that are 8 bits. Then we implemented the two command functions called “commandLCD” and “commandLCD8” for each respectively. What commandLCD does, is it sets us to use the higher 4 bits of the D4, D5, D6, D7 with the most significant bit being at D7. We set RS to high and then set the bits using the following code;

*digitalWrite(D4, data&0x10);*

*digitalWrite(D5, data&0x20);*

*digitalWrite(D6, data&0x40);*

*digitalWrite(D7, data&0x80);*

After that, we set enable to high, set a delay and then set enable back to low. Then we set the lower bits too by doing the code similar with the one above except the value different. Example for D4 the line would change to “digitalWrite(D4, data&0x01);” For the commandLCD8 function, we the code is the same except we pass the bits all at once since we want all 8-bits. Then we had an initialization function that sets entry mode, display control and function set by passing the respective values in commandLCD.

We tested our code by first setting the position of the cursor using a commandLCD and the corresponding position(We referred to datasheet for respective positions), then passing a string in writeLCD followed by a delay like this;

*commandLCD(0xC5); writeLCD('D'); delay(50);* On testing this on the LCD, we were able to see the character that we put in writeLCD. Then we were required to make the message that we print on the LCD display for one second then the first row should scroll from right to left while the second row scrolls the same way too but the far left character of the second row going to the far most character on the first row. For this, we put the characters that we needed to scroll in two arrays each array for each message. Then we have two nested for loops, the inner loop is printing and clearing the message so that when the outer for loop is looping we start printing at the first position but without displaying the first(previous first character).For example, if “Hello” is printed the first time with displaying ‘H’ in the first position(outer loop starting from 0 and inner loop as well),then “ello” will be displayed having ‘e ’ printed at the first position(outer loop starting from 1(whatever char in index 0 in the array will not be considered), same for the inner for loop).Same process will be followed until reaching the last character.

Moving on, we made the keypad connection for the next part of the lab. Since the analog pins were all unused and we needed 7 pins, we connected our keypad to the six analog pins of the arduino and the last pin was connected to pin 8 of the arduino. We used the code from the arduino library to test if our keypad connection to the arduino was right. After determining that our connection was right, we proceeded to write the code for the keypad. With online help from sources included in the references, we got different ideas on how to write functions for the keypad. First, we have two arrays one for columns and another one for rows of the keypad. Then we have a that iflag that was initialized to 0 that is used to ensure that no button is pressed. This is the "no\_press\_flag” Then we had two arrays that are used to read and ensure that none of the buttons in the keypad are pressed. We have multiple loops that check the buttons if the buttons are pressed or not. If a button is pressed, we have a loop that scans through the row and checks the buttons, if they are low it makes the output low(means it has been pressed), if the button is not pressed it sets the output to high(means it has not been pressed). Then we have two more loops that does a similar scan for the columns and sets the output to low or high depending on whether it was pressed or not. In the end of the loops, we reset the flag to 0 to ensure that no key is pressed before we start the check again by passing through the loops. After the end of the loop, we return a value of 50. On searching online, we read that for a keypad, it has a default of being set to 50 when no value is pressed therefore we return 50 to make the keypad have a default of not being pressed if we don’t press anything. In the setup, we added code that sets the input pinmode as pull-up and the output of the pinmode as output as follows;

*pinMode(Input[s],INPUT\_PULLUP);*

*pinMode(Output[i],OUTPUT);*

In addition, we have case statements that enable us to choose the number on the keypad that has been pressed. The case statements are arranged in sequential order following the keypad, ie 1-9 then pound, 0 and star last. We then initialized the communication with the serial monitor so that we can use it to check if the value that we are pressing on the keypad is the right one being displayed. On testing this, it worked as expected and we were able to see all the characters we pressed.

The last part was setting up the game. First we wrote a function for a win in the game, a loose in the game and taking a guess. Inside these functions, we call use writeLCD to write the strings to the LCD at the appropriate positions. Then after that we initialize the variables chances to 3 and guess to 0. Chances is the number of attempts that you have to try until the game decided whether you have won or lost. We then added a function that makes the LCD select a random number, the function was; “*r =random(0,9);*” then we set the game to work as follows;

The computer selects random numbers from 0 to 9, then the user takes a guess. If the guess is higher than the actual number a statement displays on the LCD stating “Choose higher” and if the number is lower it states “Choose lower”. At the third attempt, if the user is still wrong then the LCD displays “You lost” and if right the LCD displays “You win”. The game was working fine but we had to add functionalities for two special cases that are the # and \*. Therefore after the case statement for # and \* we set it so that the cases are not decremented and that it displays a message that is “Invalid Number!!” and the chances that the user has are not decremented.

For the fritzing schematic, we imported the 4x3 keypad from the online library since we could not find it in the ‘My Parts’ bin of the fritzing version in our computers. The ports of the keypad used in our fritzing schematic are slightly different from the port connections of the physical one that we used in the lab. The connection for the keypad used in fritzing was col, row, col, row, col, row, row and we used two colors blue and yellow, yellow being for the rows and blue for the columns(refer to appendix 1). In addition to the keypad, the red wires in the schematic represent connection to the 5V pin of the arduino while the grey wires represent connection to the ground pin of the arduino. For the POT, the middle wire was connected to one of the pins on the LCD while the other two wires were connected to the ground and the 5V respectively.

**C. Conclusions**

Generally, we learnt a lot about how to debugging and to improve our code. We found that using Serial.print to verify the output and instructions we sent in every step is very useful to detect the problem. After this lab, we realized although we understood how to initialize LCD and how the instructions was transferred using 4-bit interface, we coded our program in a complexed way. Our first attempt of commandLCD() sent RS and R/W to the registers, which was unnecessary because the RS was always 0 for command and R/W was connected to the ground not a pin. Besides, always review the code and consider if there is a more simple way to implement the functions. We used shifting and adding to get specific bit at first; however, it turned out we can use 0x01, 0x02, 0x04… to get those specific bit, which was more readable and easier. Through our debugging, we mainly focused on finding the bug yet forgot to improve or simplify our program. After talking with instructor, we found that optimizing the implementation not only helps others to understand, but also helps debugging it.

Also, we learnt how to connect keypad and how to import elements to fritzing in the second part. In third part, we understood that in order to display characters properly on DDRAM, we need to set specific position for each one respectively. In order to clear the display in the way we were asked to, the concept was implemented using a nested for loop and by placing the messages in arrays of characters.First time we loop(over the inner and the outer for’s) ,we display the whole message starting from array[0] till the end of the array(for the welcome example array size was 29) starting from the first position of the LCD then we clear it by calling the cleaR display command(commandLCD(0x01)). Keep looping with keeping the position of the cursor to start from the first position, we start displaying from where the position of the for loop’s index is(if index =1, we start displaying from array[1] .This means that the first character is not displayed).The process will be followed until we reach the stage of printing the last character in the first position.This is how the scrolling was encoded.

**D. References**

- The following are the references that we used in our report;

* To get the part to use in fritzing, we downloaded the parts from github from the following reference:

<https://github.com/brucetsao/Fritzing/blob/master/KEYPAD%204x3.fzpz>

* For how to scan the keypad and how to implement the rows and columns of the keypad:

<http://www.allaboutcircuits.com/projects/use-a-keypad-with-your-arduino/>

[http://www.instructables.com/id/Keypad-With-Arduino-Without-USing-Keypad-library-F/?](http://www.instructables.com/id/Keypad-With-Arduino-Without-USing-Keypad-library-F/?ALLSTEPS)

[ALLSTEPS](http://www.instructables.com/id/Keypad-With-Arduino-Without-USing-Keypad-library-F/?ALLSTEPS)

* Arduino Reference for the functions:

<http://arduino.cc/en/Reference/HomePage>

* LCD datasheet: [TC1602A-01T.pdf](https://connect.ubc.ca/bbcswebdav/pid-3136665-dt-content-rid-14227006_1/courses/SIS.UBC.ELEC.291.20C.2015W2.59753/ELEC291_15W2/L3/291_L3_LabInfo_images/TC1602A-01T.pdf)
* LCD’s extra POT

<http://www.digikey.com/product-detail/en/3386F-1-103TLF/3386F-103TLF-ND/1232544>

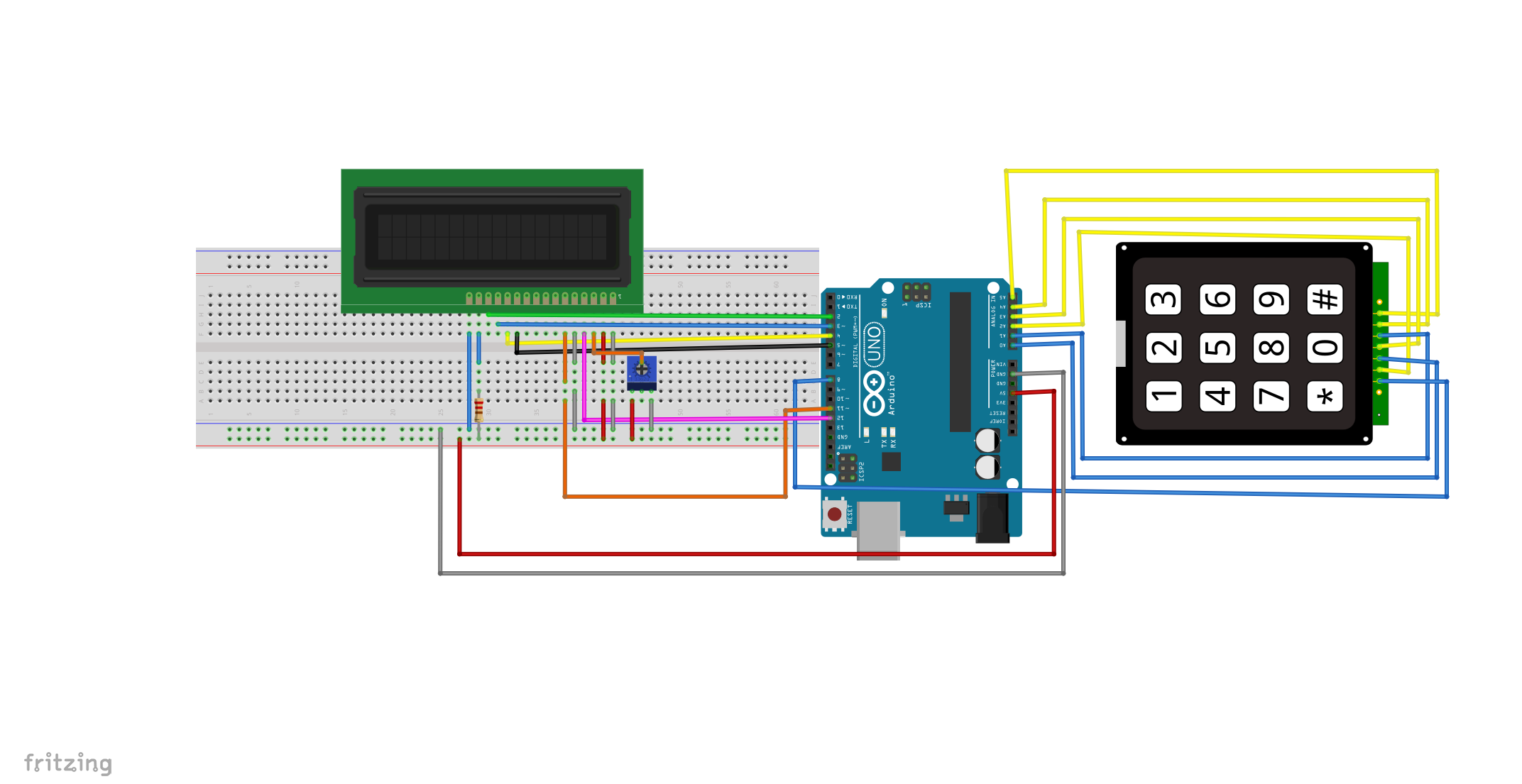
* LCD commands descriptions:

<http://mil.ufl.edu/4744/docs/lcdmanual/commands.html>

* Keypad library:<http://playground.arduino.cc/Code/Keypad#Download>
* Keypad info:<http://www.digikey.ca/product-detail/en/419/1528-1136-ND/5353596>

**Appendix I**

Below is the snapshot of the Fritzing breadboard schematic;



**Appendix II**

#define RS 12 //HIGH: DATA LOW: INSTRUCTIONS

#define E 11 //ENABLE READ OR WRITE DATA

#define D4 5 //LOW: USE 4-BIT charERFACE

#define D5 4 // R/W HIGH: READ LOW: WRITE

#define D6 3

#define D7 2

byte h=0;

byte v=0; //variables used in for loops

const unsigned long period=50; //little period used to prevent error

unsigned long kdelay=0; // variable used in non-blocking delay

const byte rows=4; //number of rows of keypad

const byte columns=3; //number of columns of keypad

const byte Output[rows]={A5,A4,A3,A2}; //array of pins used as output for rows of keypad

const byte Input[columns]={A1,A0,8}; //array of pins used as input for columns of keypad

byte keypad() // function used to detect which button is used

{

static bool no\_press\_flag=0; //static flag used to ensure no button is pressed

for(byte x=0;x<columns;x++) // for loop used to read all inputs of keypad to ensure no button is pressed

{

if (digitalRead(Input[x])==HIGH); //read every input if high continue else break;

else

break;

if(x==(columns-1)) //if no button is pressed

{

no\_press\_flag=1;

h=0;

v=0;

}

}

if(no\_press\_flag==1) //if no button is pressed

{

for(byte r=0;r<rows;r++) //for loop used to make all output as low

digitalWrite(Output[r],LOW);

for(h=0;h<columns;h++) // for loop to check if one of inputs is low

{

if(digitalRead(Input[h])==HIGH) //if specific input is remain high (no press on it) continue

continue;

else //if one of inputs is low

{

for (v=0;v<rows;v++) //for loop used to specify the number of row

{

digitalWrite(Output[v],HIGH); //make specified output as HIGH

if(digitalRead(Input[h])==HIGH) //if the input that selected from first sor loop is change to high

{

no\_press\_flag=0; //reset the no press flag;

for(byte w=0;w<rows;w++) // make all outputs as low

digitalWrite(Output[w],LOW);

return v\*4+h; //return number of button

}

}

}

}

}

return 50; //default number when no key is pressed}

void DispWelcomeMessage();

//Write values to data pins

void writeLCD(unsigned char data){

digitalWrite(RS, HIGH); // setting RS to high to select passing data not instruction

digitalWrite(D4, data&0x10); // selecting the MSB and check if it is 1 or 0

digitalWrite(D5, data&0x20);

digitalWrite(D6, data&0x40);

digitalWrite(D7, data&0x80);

//enable is set to one to start the writing cycle

digitalWrite(E,HIGH);

delay(1);

digitalWrite(E,LOW);

delay(1);

digitalWrite(RS, HIGH);

digitalWrite(D4, data&0x01);

digitalWrite(D5, data&0x02);

digitalWrite(D6, data&0x04);

digitalWrite(D7, data&0x08);//selecting the LSB

digitalWrite(E,HIGH);

delay(1);

digitalWrite(E,LOW);

delay(1);

}

// command initialization for 4 bit interface

void commandLCD(unsigned char command) {

digitalWrite(RS, LOW); //setting RS to 0 to pass an instruction

//getting bit by bit from the upper 4bits

digitalWrite(D4,command&0x10);

digitalWrite(D5, command&0x20);

digitalWrite(D6, command&0x40);

digitalWrite(D7, command&0x80);

//enable writing

digitalWrite(E,HIGH);

delay(1);

//disable writing

digitalWrite(E,LOW);

delay(1);

digitalWrite(RS, LOW);

//getting bit by bit from the lower 4bits

digitalWrite(D7, command&0x08);

digitalWrite(D6, command&0x04);

digitalWrite(D5, command&0x02);

digitalWrite(D4, command&0x01);

//enable writing

digitalWrite(E,HIGH);

delay(1);

//disable writing

digitalWrite(E,LOW);

delay(1);

}

// command initialization for 8 bit interface

void commandLCD8(unsigned char command) {

digitalWrite(RS, LOW); //setting RS to 0 to pass an instruction

digitalWrite(D4,command&0x10);

digitalWrite(D5, command&0x20);

digitalWrite(D6, command&0x40);

digitalWrite(D7, command&0x80);

//enable writing

digitalWrite(E,HIGH);

delay(1);

//disable writing

digitalWrite(E,LOW);

delay(1);

}

void setup() {

for(byte i=0;i<rows;i++) //for loop used to make pinmode of outputs as output

{

pinMode(Output[i],OUTPUT);

}

for(byte s=0;s<columns;s++) //for loop used to make pinmode of inputs as input pullup

{

pinMode(Input[s],INPUT\_PULLUP);

}

Serial.begin(9600); //to use serial monitor

// put your setup code here, to run once:

pinMode(RS,OUTPUT); //HIGH: DATA LOW: INSTRUCTIONS

pinMode(E, OUTPUT); //TELL LCD WHETHER SHOULD READ OR WRITE

pinMode(D4,OUTPUT);

pinMode(D5,OUTPUT);//HIGH:READ MODE LOW: WRITE MODE

pinMode(D6,OUTPUT);

pinMode(D7,OUTPUT);

delay(100);

initial();

// call the display function to print the message on LCD

DispWelcomeMessage();

}

void initial(){

//setting the enable to low to disable the write cycle

digitalWrite(E,LOW);

delay(16);

//the interface is in an 8bit mode initially, first function after powering the LCD on

commandLCD8(0x30);

delay(5);

// second call , still in the 8bit interface

commandLCD8(0x30);

delay(1);

// 3rd and last call in the 8bit interface

commandLCD8(0x30);

delay(1);

commandLCD8(0x30);

commandLCD8(0x20);

commandLCD(0x28);

//display control

commandLCD(0x10);

commandLCD(0x0C);

//set entry mode

commandLCD(0x06);

delay(1);

}

// to display take a guess on the LCD

void takeAGuess(){

// display clear

commandLCD(0x01);

// printing characters in different locations on the LCD

commandLCD(0x80);

writeLCD('T');

commandLCD(0x81);

writeLCD('a');

commandLCD(0x82);

writeLCD('k');

commandLCD(0x83);

writeLCD('e');

commandLCD(0x84);

writeLCD(' ');

commandLCD(0x85);

writeLCD('a');

commandLCD(0x86);

writeLCD(' ');

commandLCD(0x87);

writeLCD('g');

commandLCD(0x88);

writeLCD('u');

commandLCD(0x89);

writeLCD('e');

commandLCD(0x8A);

writeLCD('s');

commandLCD(0x8B);

writeLCD('s');

commandLCD(0x8C);

writeLCD(':');

delay(1000);

}

// To display guess lower on the LCD

void lower(){

//display clear

commandLCD(0x01);

// printing characters in different locations on the LCD

commandLCD(0x80);

writeLCD('G');

commandLCD(0x81);

writeLCD('u');

commandLCD(0x82);

writeLCD('e');

commandLCD(0x83);

writeLCD('s');

commandLCD(0x84);

writeLCD('s');

commandLCD(0x85);

writeLCD(' ');

commandLCD(0x86);

writeLCD('l');

commandLCD(0x87);

writeLCD('o');

commandLCD(0x88);

writeLCD('w');

commandLCD(0x89);

writeLCD('e');

commandLCD(0x8A);

writeLCD('r');

delay(1000);

}

// To display guess higher on the LCD

void higher(){

//display clear

commandLCD(0x01);

// printing characters in different locations on the LCD

commandLCD(0x80);

writeLCD('G');

commandLCD(0x81);

writeLCD('u');

commandLCD(0x82);

writeLCD('e');

commandLCD(0x83);

writeLCD('s');

commandLCD(0x84);

writeLCD('s');

commandLCD(0x85);

writeLCD(' ');

commandLCD(0x86);

writeLCD('h');

commandLCD(0x87);

writeLCD('i');

commandLCD(0x88);

writeLCD('g');

commandLCD(0x89);

writeLCD('h');

commandLCD(0x8A);

writeLCD('e');

commandLCD(0x8B);

writeLCD('r');

delay(1000);

}

// To display you won on the LCD

void youWon(){

//display clear

commandLCD(0x01);

// printing characters in different locations on the LCD

commandLCD(0x80);

writeLCD('Y');

commandLCD(0x81);

writeLCD('o');

commandLCD(0x82);

writeLCD('u');

commandLCD(0x83);

writeLCD(' ');

commandLCD(0x84);

writeLCD('w');

commandLCD(0x85);

writeLCD('o');

commandLCD(0x86);

writeLCD('n');

commandLCD(0x87);

writeLCD('!');

delay(1000);

}

// To display you lost on the LCD

void youLost(){

//display clear

commandLCD(0x01);

// printing characters in different locations on the LCD

commandLCD(0x80);

writeLCD('Y');

commandLCD(0x81);

writeLCD('o');

commandLCD(0x82);

writeLCD('u');

commandLCD(0x83);

writeLCD(' ');

commandLCD(0x84);

writeLCD('l');

commandLCD(0x85);

writeLCD('o');

commandLCD(0x86);

writeLCD('s');

commandLCD(0x87);

writeLCD('t');

commandLCD(0x88);

writeLCD('!');

delay(2000);

}

// To display Invalid on the LCD

void InvalidNumber(){

//display clear

commandLCD(0x01);

// printing characters in different locations on the LCD

commandLCD(0x80);

writeLCD('I');

commandLCD(0x81);

writeLCD('n');

commandLCD(0x82);

writeLCD('v');

commandLCD(0x83);

writeLCD('a');

commandLCD(0x84);

writeLCD('l');

commandLCD(0x85);

writeLCD('i');

commandLCD(0x86);

writeLCD('d');

commandLCD(0x87);

writeLCD('!');

commandLCD(0x88);

writeLCD('!');

delay(2000);

}

int chance = 3; // allowing the player to guess three times

byte r; // the random number

int won =0; // variable to be used to check if the user won

// the game logic

void compare(){

int g = 0; //number guessed

delay(200);

if(millis()-kdelay>period) //used to make non-blocking delay

{

byte key = keypad(); // key is set to be which button was pressed

kdelay=millis(); //capture time from millis function

// waiting until one of the buttons are pressed

while(key == 50){

key = keypad();

}

switch (key) //switch used to specify which button

{

// in case user pressed 1

case 0:

{g=1;

Serial.println(g);

}

break;

// in case user pressed 2

case 1:

{g=2;

Serial.println(g);

}

break;

// in case user pressed 3

case 2:

{g=3;

Serial.println(g);

}

break;

// in case user pressed 4

case 4:

{g=4;

Serial.println(g);

}

break;

// in case user pressed 5

case 5:

{g=5;

Serial.println(g);

};

break;

// in case user pressed 6

case 6:

{g=6;

Serial.println(g);

}

break;

// in case user pressed 7

case 8:

{g=7;

Serial.println(g);

}

break;

// in case user pressed 8

case 9:

{g=8;

Serial.println(g);

};

break;

// in case user pressed 9

case 10:

{g=9;

Serial.println(g);

};

break;

// in case user pressed \*

case 12:

{Serial.println("Star");

chance = chance +1;

InvalidNumber();

}

break;

// in case user pressed 0

case 13:

{g=0;

Serial.println(g);

}

break;

// in case user pressed #

case 14:

{Serial.println("numberSign");

chance = chance +1;

InvalidNumber();

}

break;

default:

;

}

}

// if the user consumed all the attempts and the guess was not equal to random, display you lost!

if (chance == 0 && g!=r){

youLost();

}

//if the guess is lower than the random number , display guess higher

else if(g < r ){

higher();

}

//if the guess is higher than the random number , display guess lower

else if(g > r ){

lower();

}

// else the guess is equal to random

else{

youWon();

won = 1;

}

}

int rndomNumber =0;

// welcome message is allocated in an array1

char welcome []= {'H', 'e', 'l', 'l', 'o', ' ', '&', ' ', 'w', 'e', 'l', 'c', 'o', 'm', 'e', ' ', 't', 'o', ' ', 'o', 'u', 'r', ' ', 'd', 'e', 'm', 'o', '!',' '};

// elec message is allocated in an array2

char elec [] = {'E', 'L', 'E', 'C', ' ', '2', '9', '1', '-', '2', '0', 'C', ' ', 'T', 'e', 'a', 'm', ' ', 'L', '2', 'D', '-', '5', 'C'};

// to display the messages on the LCD with scrolling it until the display is clear (from right to left)

void DispWelcomeMessage(){

// to be able to write to LCD

commandLCD(0x28);

// setting the # of line to be 2, smallest font

commandLCD(0x80);

// clear display

commandLCD(0x01);

int i=0,j=0; //variables to be used in for loops

byte curs = 0x81; //set the cursor at the first position

// the for loops are used to clear the display from right to left as requested

for (j=0;j<29;++j){

//displaying msg starting depending on the position of the cursor(displaying each char at the designed position)

for (i=j;i<29;++i){

writeLCD(welcome[i]);

//setting the cursor position

commandLCD(curs+i-j);

// to set the cursor to C0(first position in the 2nd line)

if ((curs+i-j)==0x8F){

curs = 0xBF-i+j;

}

}

//set the cursor to start properly

curs = 0x81;

delay(500);

// display clear

commandLCD(0x01);

}

// same thing for displaying the 2nd msg

i=0; j=0;

curs = 0x81;

for (j=0;j<24;++j){

for (i=j;i<24;++i){

writeLCD(elec[i]);

commandLCD(curs+i-j);

if ((curs+i-j)==0x8F){

curs = 0xBF-i+j;

}

}

curs = 0x81;

delay(500);

commandLCD(0x01);

}

delay(5000);

}

void loop() {

//set r to be 0<= r <=9

r =random(0,9);

//print on serial monitor

Serial.print("The chosen random number is: ");

// print the value of r on serial monitor

Serial.println(r);

//print take a guess on LCD

takeAGuess();

// decrease the # of attempts and compare the guess

while((chance--) && (won ==0))

{

// check what is the user status: won,lost,guess higher/lower

compare();

//print how many chance are left to guess

Serial.print("Chances left: ");

Serial.println(chance);

}

// restart the game by resetting the attempts

chance = 3;

won = 0;

}