

Lab 02

LCD and Seven Segment Display

Logan Barber, Ian Nail

January 28, 2022

Ian Nail

Logan Barber

ME-4370 - Stephen Canfield

1 Executive Summary

In this lab, the objective is to output a recipe to a Liquid Crystal Display (LCD) and include a seven segment display as a timer. The minimum objectives were:

1. Tell the user what recipe is to follow.
2. Provide the recipe steps in order, timed.
3. Provide a countdown timer for any timed events in the recipe.
4. Display all instructions on the LCD screen
5. Display countdown time on one or more 7-segment displays
6. Test your product on someone outside the class, record their use of your product.

For objective 1, 2, and 4. the recipe is displayed on the LCD one item at a time. The next recipe item is displayed with the push button. For objective 3 and 5 the timed portion the timer will start and stop with a push button. The 4 digit seven segment display is used for the timer. The first two digits are the minutes, and the second two digits are the seconds.

An I2C module was used for the LCD to cut down on wires and ports needed from the Arduino Mega. The library LiquidCrystal_I2C.h was utilized. Communication happens over 4 wires. Power, ground, and 2 communication lines.

On wiring the seven segment that has 4 displays. 12 GPIO pins are required from the Arduino. 8 are required for the segments (7 segments + 1 for the decimal point). The other 4 GPIO pins select which display to turn on. Only one display should be on at a time (unless you want the same digit to be written to other displays). This means that we have to continuously write to each display faster than we can visually see. This gives the appearance that each display is being on all at once. To use the GPIO pins, the registers were written to directly. PORTB GPIO pins were used. This happens in lines 92 to 94 in the C code.

AutoCad was used to draw up the circuit diagram of all the connections, components, and devices used.

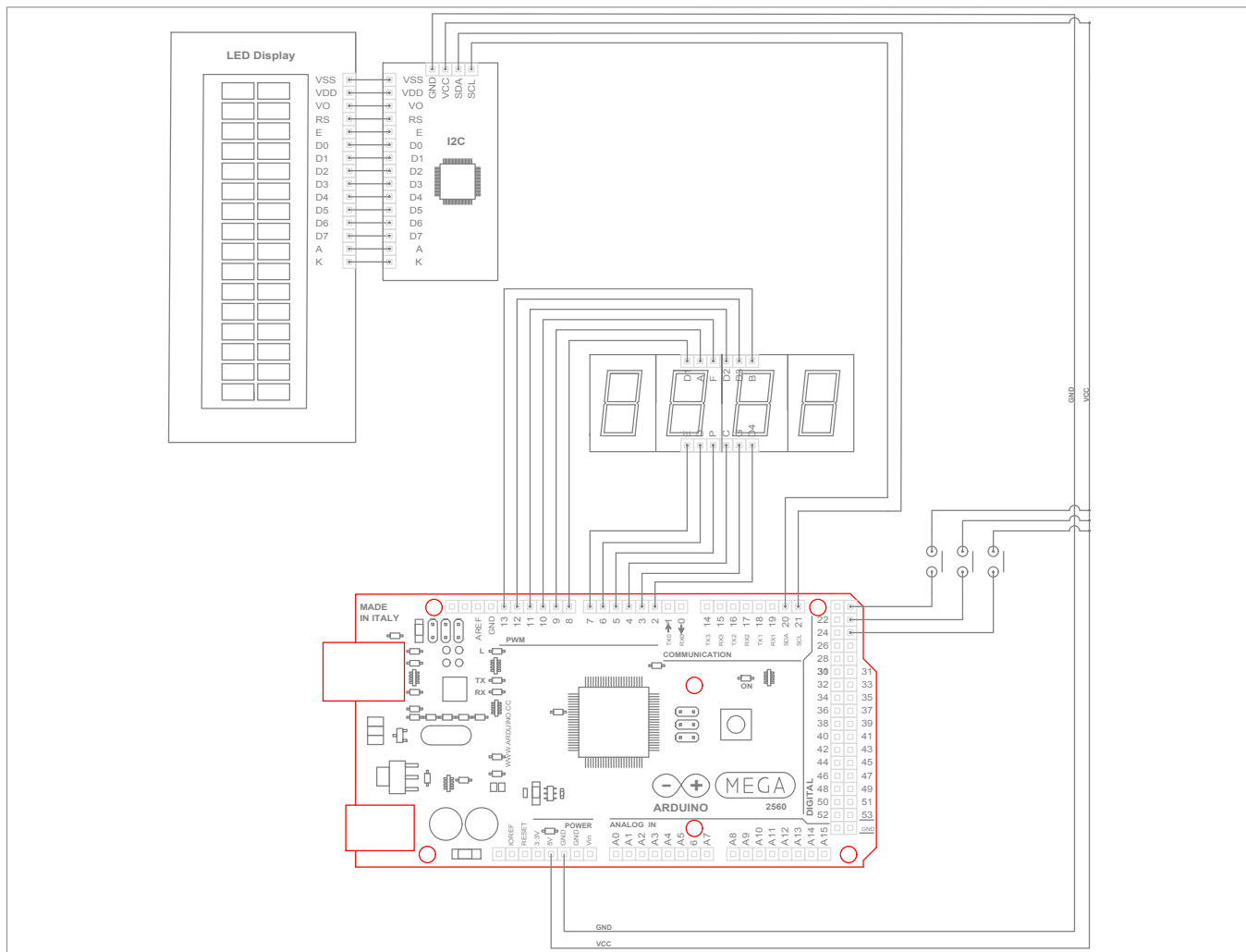


Figure 1: Circuit Diagram

2 Source Code

```
0 // AUTHORS: A. LOGAN BARBER; IAN NAIL
1 // FILE NAME: Lab02.ino
2 // LAST UPDATED: 28 JANUARY 2022
3 /*
4  * PURPOSE: THIS FILE IS THE MAIN FILE FOR DISPLAYING A RECIPE ON AN LCD AND
5  * BEING ABLE TO SCROLL USING TWO BUTTONS.
6  * THIS FILE ALSO UTILIZES AND 4-7 SEGMENT DISPLAY AS A 10 MINUTE TIMER.
7  */
8
9 /*
10  * BUTTON0:
11  *
12  * DIGITAL PIN: 27
13  *
14  * PORT: A
15  *
16  * PORT PIN: 5
17  */
18
19 /*
20  * BUTTON1:
21  *
22  * DIGITAL PIN: 28
23  *
24  * PORT: A
25  *
26  * PORT PIN: 6
27  */
28
29 /*
30  * BUTTON2:
31  *
32  * DIGITAL PIN: 29
33  *
34  * PORT: A
35  *
36  * PORT PIN: 7
37  */
38
39 /*
40  * LCD 4-7 SEGMENT DISPLAY
41  *
42  * PIN 1:
43  *     DIGITAL PIN: 10
44  *     PORT: B
45  *     PORT PIN: 4
46  *
47  * PIN 2:
48  *     DIGITAL PIN: 50
49  *     PORT: B
50  *     PORT PIN: 3
51  *
52  * PIN 3:
53  *     DIGITAL PIN: 13
54  *     PORT: B
55  *     PORT PIN: 7
56  *
```

```

56  * PIN 4:
57  *     DIGITAL PIN: 51
58  *     PORT: B
59  *     PORT PIN: 2
60  *
61  * PIN 5:
62  *     DIGITAL PIN: 12
63  *     PORT: B
64  *     PORT PIN: 6
65  *
66  * PIN 6:
67  *     DIGITAL PIN: 22
68  *     PORT: A
69  *     PORT PIN: 0
70  *
71  * PIN 7:
72  *     DIGITAL PIN: 52
73  *     PORT: B
74  *     PORT PIN: 1
75  *
76  * PIN 8:
77  *     DIGITAL PIN: 23
78  *     PORT: A
79  *     PORT PIN: 1
80  *
81  * PIN 9:
82  *     DIGITAL PIN: 24
83  *     PORT: A
84  *     PORT PIN: 2
85  *
86  * PIN 10:
87  *     DIGITAL PIN: 11
88  *     PORT: B
89  *     PORT PIN: 5
90  *
91  * PIN 11:
92  *     DIGITAL PIN: 53
93  *     PORT: B
94  *     PORT PIN: 0
95  *
96  * PIN 12:
97  *     DIGITAL PIN: 25
98  *     PORT: A
99  *     PORT PIN: 3
100 */

102 // INCLUDE LIBRARIES
#include <LiquidCrystal_I2C.h>

104 // DEFINE PIN NUMBERS
106 #define ADDRESS 0x27
107 #define COLS 16
108 #define ROWS 2

110 // TIMER PARAMTERS
uint8_t buttonState = 0;
112 uint8_t minutes = 10; //start time -> CAN CHANGE TO WHATEVER TIME YOU WANT
uint8_t seconds = 0; //start time -> CAN CHANGE TO WHATEVER TIME YOU WANT
114 uint8_t totalMinutes = 0;
uint8_t minutesTens = 0;

```

```

116 uint8_t minutesOnes = 0;
    uint8_t secondsTens = 0;
118 uint8_t secondsOnes = 0;
    uint8_t secondsTemp = 0;
120 float totalSeconds = minutes*60 + seconds;
    float totalMilliseconds = totalSeconds*1000;
122 float totalMicroseconds = totalMilliseconds*1000;

124 // DEFINE LETTERS FOR 7 SEGMENT DISPLAY
    const uint8_t ZERO = 0x3F;
126 const uint8_t ZERO_DEV = 0xBF;
    const uint8_t ONE = 0x06;
128 const uint8_t ONE_DEC = 0x86;
    const uint8_t TWO = 0x5B;
130 const uint8_t TWO_DEC = 0xDB;
    const uint8_t THREE = 0x4F;
132 const uint8_t THREE_DEC = 0xCF;
    const uint8_t FOUR = 0x66;
134 const uint8_t FOUR_DEC = 0xE6;
    const uint8_t FIVE = 0x6D;
136 const uint8_t FIVE_DEC = 0xED;
    const uint8_t SIX = 0x7D;
138 const uint8_t SIX_DEC = 0xFD;
    const uint8_t SEVEN = 0x07;
140 const uint8_t SEVEN_DEC = 0x87;
    const uint8_t EIGHT = 0x7F;
142 const uint8_t EIGHT_DEC = 0xFF;
    const uint8_t NINE = 0x67;
144 const uint8_t NINE_DEC = 0xE7;

146 // DEFINE THE DISPLAY SELECTION NUMBERS
    const uint8_t D1 = 0xE7; // 0b11100111
148 const uint8_t D2 = 0xEB; // 0b11101011
    const uint8_t D3 = 0xED; // 0b11101101
150 const uint8_t D4 = 0xEE; // 0b11101110
    const uint8_t arrD[4] = {D4, D3, D2, D1};

152 // MESSAGE TO PRINT
154 char message0[] = "Double Chocolate";
    char message1[] = "Flower Brownies";
156 char message2[] = "1/2 Cup";
    char message3[] = "Unsalted Butter";
158 char message4[] = "1 Gram";
    char message5[] = "Flower";
160 char message6[] = "1/4 Cup ";
    char message7[] = "Chocolate Chips";
162 char message8[] = "1 Tablespoon";
    char message9[] = "Molasses";
164 char message10[] = "1 Teaspoon";
    char message11[] = "Vanilla Extract";
166 char message12[] = "2 Large Eggs";
    char message13[] = " ";
168 char message14[] = "1/4 Teaspoon";
    char message15[] = "Kosher Salt";
170 char message16[] = "3/4 Cup All-";
    char message17[] = "Purpose Flour";
172 char message18[] = "Bake for 10";
    char message19[] = "minutes";
174 const uint8_t msgArrSize = 20;
    char* msgArr[msgArrSize] = {message0, message1, message2, message3, message4,

```

```

176     message5, message6, message7, message8, message9, message10, message11,
        message12,
        message13, message14, message15, message16, message17, message18, message19
    };

178 // INDEX VARIABLES
180 uint8_t index = 0; // HOLDS INDEX FOR MESSAGE
    uint8_t i = 0; // HOLDS INDEX IN for LOOPS FOR SCROLLING
182 uint8_t t = 0; // HOLDS INDEX IN for LOOP FOR THE TIMER

184 // CREATE LiquidCrystal OBJECT
    LiquidCrystal_I2C lcd (ADDRESS, COLS, ROWS);

186 // RUN THIS PROGRAM
188 void setup ()
    {
190         // INITIALIZE THE LCD SCREEN
        lcd.begin ();

192         // PRINT MESSAGE
        lcd.print (msgArr [0]);
        lcd.setCursor (0, 1);
        lcd.print (msgArr [1]);

198         // SETUP BUTTON PINS AS INPUTS
        // SETUP 7-SEGMENT SELECTOR PINS AS OUTPUT
200        DDRA = 0x0F; // 0b00001111

202        // ENABLE INTERNAL PULL-UP RESISTOR FOR BUTTONS
        PORTA = 0xE0; // 0b11100000

204        // SETUP PORT B AS OUTPUT FOR THE LCD
        DDRB = 0xFF;
        PORTB = 0x00;

208        // CALCULATE INDIVIDUAL DIGITS
        totalMinutes = totalSeconds/60;
        minutesTens = totalMinutes/10;
        minutesOnes = totalMinutes%10;
        secondsTemp = int (totalSeconds)%60;
        secondsTens = secondsTemp/10;
        secondsOnes = secondsTemp%10;
216    }

218 // LOOP FOREVER
    void loop ()
    {
220        // ELSE IF BUTTON0 IS LOW SCROLL DOWN
        if ((PINA & 0xE0) == 0xC0)
        {
224            // DEBOUNCE BUTTON2
            delay (100);
            if ((PINA & 0xE0) == 0xC0)
            {
228                scroll_down ();
            }
        }

230    }

232 // IF BUTTON1 IS LOW THEN SCROLL UP
    else if ((PINA & 0xE0) == 0xA0)

```

```

234 {
235     // DEBOUNCE THE BUTTON1
236     delay(100);
237     if((PINA & 0xE0) == 0xA0)
238     {
239         scroll_up();
240     }
241 }
242
243 // IF BUTTON 2 IS LOW CHANGE THE BUTTON STATE
244 else if((PINA & 0xE0) == 0x60)
245 {
246     delay(100);
247     if((PINA & 0xE0) == 0x60)
248     {
249         switch(buttonState)
250         {
251             case 0:
252                 buttonState = 1;
253                 break;
254
255             case 1:
256                 buttonState = 2;
257                 break;
258
259             case 2:
260                 buttonState = 0;
261
262                 // RESET TIME
263                 totalSeconds = minutes*60 + seconds;
264                 totalMilliseconds = totalSeconds*1000;
265                 totalMicroseconds = totalMilliseconds*1000;
266
267                 // CALCULATE INDIVIDUAL DIGITS
268                 totalMinutes = totalSeconds/60;
269                 minutesTens = totalMinutes/10;
270                 minutesOnes = totalMinutes%10;
271                 secondsTemp = int(totalSeconds)%60;
272                 secondsTens = secondsTemp/10;
273                 secondsOnes = secondsTemp%10;
274                 break;
275             }
276         }
277     }
278
279 // RUN TIMER IF BUTTON STATE IS IN STATE 1
280 if(buttonState == 1)
281 {
282     // TIME CALCULATIONS
283     totalMicroseconds = totalMicroseconds - 2000; //totalMilliseconds++
284 for stopwatch
285     totalMilliseconds = totalMicroseconds/1000;
286     totalSeconds = (totalMilliseconds/1000+1);
287
288     // CALCULATE INDIVIDUAL DIGITS
289     totalMinutes = totalSeconds/60;
290     minutesTens = totalMinutes/10;
291     minutesOnes = totalMinutes%10;
292     secondsTemp = int(totalSeconds)%60;
293     secondsTens = secondsTemp/10;

```



```

294         secondsOnes = secondsTemp%10;
295     }
296     // TIMER
297     for(t = 0; t < 4; ++t)
298     {
299         switch(t)
300         {
301             case 0:
302                 PORTA = arrD[t];
303                 pickNumber(minutesTens);
304                 delayMicroseconds(500);
305                 break;
306
307             case 1:
308                 PORTA = arrD[t];
309                 pickNumber(minutesOnes);
310                 PORTB |= 0x80;
311                 delayMicroseconds(500);
312                 break;
313
314             case 2:
315                 PORTA = arrD[t];
316                 pickNumber(secondsTens);
317                 delayMicroseconds(500);
318                 break;
319
320             case 3:
321                 PORTA = arrD[t];
322                 pickNumber(secondsOnes);
323                 delayMicroseconds(500);
324                 break;
325         }
326     }
327 }
328
329 /*
330 * TYPE: FUNCTION
331 * NAME: scroll_up
332 * RETURN: void
333 * NUMBER OF PARAMETERS: 2
334 * PARAMETER NAMES: char* messagePtr, uint8_t sizeOfArray
335 * PURPOSE: THIS FUNCTION SCROLLS THROUGH THE RECEIPE DISPLAYED ON THE LCD
336 */
337 void scroll_up()
338 {
339     // DECREMENT INDEX BY ONE
340     index -= 2;
341
342     // CHECK THE BOUNDS OF INDEX (REMEMBER index IS UNSIGNED)
343     if(index > (msgArrSize - 2))
344         index = 0;
345
346     // CLEAR THE LCD SCREEN AND PRINT MESSAGES TO THE LCD
347     lcd.clear();
348     for(i = 0; i < 2; ++i)
349     {
350         lcd.setCursor(0, i);
351         delayMicroseconds(1000);
352         lcd.print(msgArr[index + i]);

```

```

        delayMicroseconds(1000);
354     }
355 }
356
357 /*
358  * TYPE: FUNCTION
359  * NAME: scroll_down
360  * RETURN: void
361  * NUMBER OF PARAMETERS: 2
362  * PARAMETER NAMES: char* messagePtr, uint8_t sizeOfArray
363  * PURPOSE: THIS FUNCTION SCROLLS THROUGH THE RECEIPE DISPLAYED ON THE LCD
364  */
365 void scroll_down()
366 {
367     // INCREMENT INDEX
368     index += 2;
369
370     // CHECK THE BOUNDS OF INDEX
371     if(index > (msgArrSize - 2))
372         index = msgArrSize - 2;
373
374     // CLEAR THE LCD SCREEN AND PRINT MESSAGES TO THE LCD
375     lcd.clear();
376     for(i = 0; i < 2; ++i)
377     {
378         lcd.setCursor(0, i);
379         delayMicroseconds(1000);
380         lcd.print(msgArr[index + i]);
381         delayMicroseconds(1000);
382     }
383 }
384
385 /*
386  * TYPE: FUNCTION
387  * NAME: pickNumber
388  * RETURN: void
389  * NUMBER OF PARAMETERS: 1
390  * PARAMETER NAMES: int x
391  * PURPOSE: THIS FUNCTION PICK THE NUMBER FOR THE LCD
392  */
393 void pickNumber(int x) //changes value of number
394 {
395     switch(x)
396     {
397         default:
398             PORTB = ZERO;
399             break;
400         case 1:
401             PORTB = ONE;
402             break;
403         case 2:
404             PORTB = TWO;
405             break;
406         case 3:
407             PORTB = THREE;
408             break;
409         case 4:
410             PORTB = FOUR;
411             break;
412         case 5:

```

```
414         PORTB = FIVE;  
         break;  
416     case 6:  
         PORTB = SIX;  
         break;  
418     case 7:  
         PORTB = SEVEN;  
420         break;  
422     case 8:  
         PORTB = EIGHT;  
         break;  
424     case 9:  
         PORTB = NINE;  
426         break;  
428     }
```

Lab02.ino