Lab 02

LCD and Seven Segment Display

Logan Barber, Ian Nail

January 28, 2022

Ian Nail

Logan Barber

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 ounce. To use the GPIO pins, the registers were written to directly. PORTB GPIO pins were used. This happen is lines 92 to 94 in the C code.

```
0    *       PORT PIN: 2
     *
2    * PIN 10:
```

Lab02.ino

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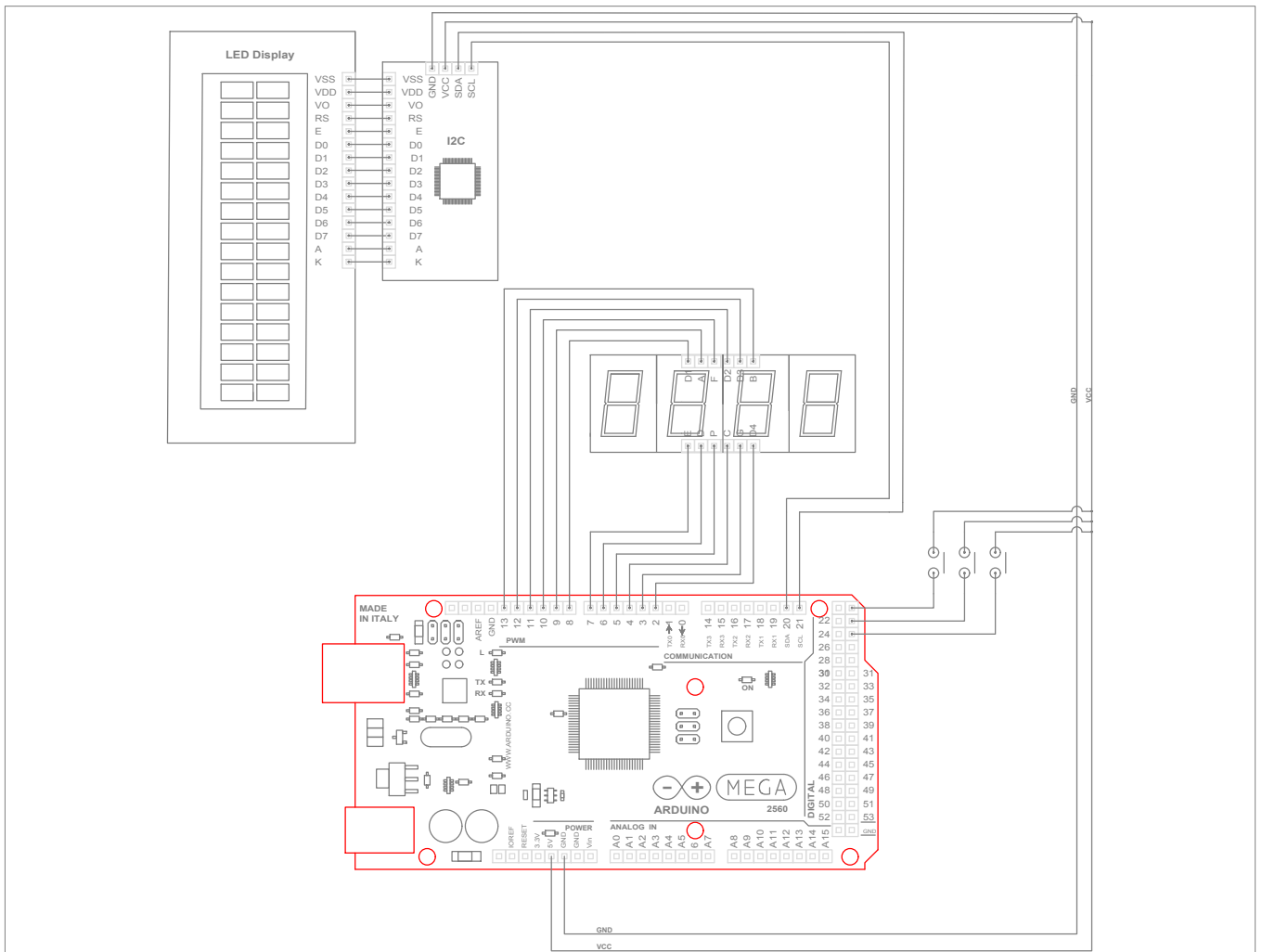

Figure 1: Circuit Diagram

2 Source Code

```
// AUTHORS: A. LOGAN BARBER; IAN NAIL
<<<<<<< HEAD
// FILE NAME: LCD_Dpad.ino
// LAST UPDATED: 19 JANUARY 2022
//
//   PURPOSE: THIS FILE IS THE MAIN FILE FOR DISPLAYING A RECIPE ON AN LCD AND
    BEING ABLE TO SCROLL USING TWO BUTTONS.
//
=======
// FILE NAME: Lab02.ino
// LAST UPDATED: 28 JANUARY 2022
/*
 *   PURPOSE: THIS FILE IS THE MAIN FILE FOR DISPLAYING A RECIPE ON AN LCD AND
    BEING ABLE TO SCROLL USING TWO BUTTONS.
 *          THIS FILE ALSO UTILIZES AND 4-7 SEGMENT DISPLAY AS A 10 MINUTE TIMER.
 */

/*
 * BUTTON0:
 *
 * DIGITAL PIN: 27
 *
 * PORT: A
 *
 * PORT PIN: 5
 */

 /*
 * BUTTON1:
 *
 * DIGITAL PIN: 28
 *
 * PORT: A
 *
 * PORT PIN: 6
 */

 /*
 * BUTTON2:
 *
 * DIGITAL PIN: 29
 *
 * PORT: A
 *
 * PORT PIN: 7
 */

 /*
  * LCD 4-7 SEGMENT DISPLAY
  *
  * PIN 1:
  *      DIGITAL PIN: 10
  *      PORT: B
  *      PORT PIN: 4
  *
  * PIN 2:
  *      DIGITAL PIN: 50
```

```c
 *        PORT: B
 *        PORT PIN: 3
 *
 * PIN 3:
 *        DIGITAL PIN: 13
 *        PORT: B
 *        PORT PIN: 7
 *
 * PIN 4:
 *        DIGITAL PIN: 51
 *        PORT: B
 *        PORT PIN: 2
 *
 * PIN 5:
 *        DIGITAL PIN: 12
 *        PORT: B
 *        PORT PIN: 6
 *
 * PIN 6:
 *        DIGITAL PIN: 22
 *        PORT: A
 *        PORT PIN: 0
 *
 * PIN 7:
 *        DIGITAL PIN: 52
 *        PORT: B
 *        PORT PIN: 1
 *
 * PIN 8:
 *        DIGITAL PIN: 23
 *        PORT: A
 *        PORT PIN: 1
 *
 * PIN 9:
 *        DIGITAL PIN: 24
 *        PORT: A
 *        PORT PIN: 2
 *
 * PIN 10:
 *        DIGITAL PIN: 11
 *        PORT: B
 *        PORT PIN: 5
 *
 * PIN 11:
 *        DIGITAL PIN: 53
 *        PORT: B
 *        PORT PIN: 0
 *
 * PIN 12:
 *        DIGITAL PIN: 25
 *        PORT: A
 *        PORT PIN: 3
 */
>>>>>>> refs/remotes/origin/master

// INCLUDE LIBRARIES
#include <LiquidCrystal_I2C.h>

// DEFINE PIN NUMBERS
#define ADDRESS 0x27
```

```cpp
#define COLS 16
#define ROWS 2

// TIMER PARAMTERS
uint8_t buttonState = 0;
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
uint8_t totalMinutes = 0;
uint8_t minutesTens = 0;
uint8_t minutesOnes = 0;
uint8_t secondsTens = 0;
uint8_t secondsOnes = 0;
uint8_t secondsTemp = 0;
float totalSeconds = minutes*60 + seconds;
float totalMilliseconds = totalSeconds*1000;
float totalMicroseconds = totalMilliseconds*1000;

// DEFINE LETTERS FOR 7 SEGMENT DISPLAY
const uint8_t ZERO = 0x3F;
const uint8_t ZERO_DEV = 0xBF;
const uint8_t ONE = 0x06;
const uint8_t ONE_DEC = 0x86;
const uint8_t TWO = 0x5B;
const uint8_t TWO_DEC = 0xDB;
const uint8_t THREE = 0x4F;
const uint8_t THREE_DEC = 0xCF;
const uint8_t FOUR = 0x66;
const uint8_t FOUR_DEC = 0xE6;
const uint8_t FIVE = 0x6D;
const uint8_t FIVE_DEC = 0xED;
const uint8_t SIX = 0x7D;
const uint8_t SIX_DEC = 0xFD;
const uint8_t SEVEN = 0x07;
const uint8_t SEVEND_DEC = 0x87;
const uint8_t EIGHT = 0x7F;
const uint8_t EIGHT_DEC = 0xFF;
const uint8_t NINE = 0x67;
const uint8_t NINE_DEC = 0xE7;

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

// MESSAGE TO PRINT
char message0[] = "Hello World!";
char message1[] = "Hallo!";
char message2[] = "Wie gehts";
const uint8_t msgArrSize = 3;
char* msgArr[msgArrSize] = {message0, message1, message2};

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

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

```
176   // RUN THIS PROGRAM
      void setup()
178   {
          // INITIALIZE THE LCD SCREEN
180       lcd.begin();

182       // PRINT MESSAGE
          lcd.print(msgArr[0]);
184       lcd.setCursor(0, 1);
          lcd.print(msgArr[1]);
186
          // SETUP BUTTON PINS AS INPUTS
188       // SETUP 7-SEGMENT SELECTOR PINS AS OUTPUT
          DDRA = 0x0F;  // 0b00001111
190
          // ENABLE INTERNAL PULL-UP RESISTOR FOR BUTTONS
192       PORTA = 0xE0;  // 0b11100000

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

198       // CALCULATE INDIVIDUAL DIGITS
          totalMinutes = totalSeconds/60;
200       minutesTens = totalMinutes/10;
          minutesOnes = totalMinutes%10;
202       secondsTemp = int(totalSeconds)%60;
          secondsTens = secondsTemp/10;
204       secondsOnes = secondsTemp%10;
      }
206
      // LOOP FOREVER
208   void loop()
      {
210       // ELSE IF BUTTON0 IS LOW SCROLL DOWN
          if((PINA & 0xE0) == 0xC0)
212       {
              // DEBOUNCE BUTTON2
214           delay(100);
              if((PINA & 0xE0) == 0xC0)
216           {
                  scroll_down();
218           }
          }
220
          // IF BUTTON1 IS LOW THEN SCROLL UP
222       else if((PINA & 0xE0) == 0xA0)
          {
224           // DEBOUNCE THE BUTTON1
              delay(100);
226           if((PINA & 0xE0) == 0xA0)
              {
228               scroll_up();
              }
230       }

232       // IF BUTTON 2 IS LOW CHANGE THE BUTTON STATE
          else if((PINA & 0xE0) == 0x60)
234       {
```

```
               delay (100);
236            if ((PINA & 0xE0) == 0x60)
               {
238                switch (buttonState)
                   {
240                    case 0:
                           buttonState = 1;
242                        break;

244                    case 1:
                           buttonState = 2;
246                        break;

248                    case 2:
                           buttonState = 0;
250
                           // RESET TIME
252                        totalSeconds = minutes*60 + seconds;
                           totalMilliseconds = totalSeconds*1000;
254                        totalMicroseconds = totalMilliseconds*1000;

256                        // CALCULATE INDIVIDUAL DIGITS
                           totalMinutes = totalSeconds/60;
258                        minutesTens = totalMinutes/10;
                           minutesOnes = totalMinutes%10;
260                        secondsTemp = int(totalSeconds)%60;
                           secondsTens = secondsTemp/10;
262                        secondsOnes = secondsTemp%10;
                           break;
264                }
               }
266        }

       // RUN TIMER IF BUTTON STATE IS IN STATE 1
268    if (buttonState == 1)
270    {
           // TIME CALCULATIONS
272        totalMicroseconds = totalMicroseconds - 2000; //totalMilliseconds++'
    for stopwatch
           totalMilliseconds = totalMicroseconds/1000;
274        totalSeconds = (totalMilliseconds/1000+1);

276        // CALCULATE INDIVIDUAL DIGITS
           totalMinutes = totalSeconds/60;
278        minutesTens = totalMinutes/10;
           minutesOnes = totalMinutes%10;
280        secondsTemp = int(totalSeconds)%60;
           secondsTens = secondsTemp/10;
282        secondsOnes = secondsTemp%10;
       }
284
       // TIMER
286    for (t = 0; t < 4; ++t)
       {
288        switch (t)
           {
290            case 0:
                   PORTA = arrD[t];
292                pickNumber(minutesTens);
                   delayMicroseconds(500);
```

```
                            break ;

                    case  1 :
                            PORTA = arrD [ t ] ;
                            pickNumber ( minutesOnes ) ;
                            PORTB |= 0x80 ;
                            delayMicroseconds ( 5 0 0 ) ;
                            break ;

                    case  2 :
                            PORTA = arrD [ t ] ;
                            pickNumber ( secondsTens ) ;
                            delayMicroseconds ( 5 0 0 ) ;
                            break ;

                    case  3 :
                            PORTA = arrD [ t ] ;
                            pickNumber ( secondsOnes ) ;
                            delayMicroseconds ( 5 0 0 ) ;
                            break ;
            }
        }
}

/*
 * TYPE: FUNCTION
 * NAME:  scroll_up
 * RETURN:  void
 * NUMBER OF PARAMETERS: 2
 * PARAMETER NAMES: char∗ messagePtr , uint8_t sizeOfArray
 * PURPOSE: THIS FUNCTION SCROLLS THROUGH THE RECEIPE DISPLAYED ON THE LCD
 */
void  scroll_up ( )
{
        // DECREMENT INDEX BY ONE
        −−index ;

        // CHECK THE BOUNDS OF INDEX (REMEMBER index IS UNSIGNED)
        if ( index > ( msgArrSize − 2 ) )
                index = 0 ;

        // CLEAR THE LCD SCREEN AND PRINT MESSAGES TO THE LCD
        lcd . clear ( ) ;
        for ( i = 0 ;  i < 2 ;  ++i )
        {
                lcd . setCursor ( 0 ,  i ) ;
                delayMicroseconds ( 1 0 0 0 ) ;
                lcd . print ( msgArr [ index + i ] ) ;
                delayMicroseconds ( 1 0 0 0 ) ;
        }
}

/*
 * TYPE: FUNCTION
 * NAME:  scroll_down
 * RETURN:  void
 * NUMBER OF PARAMETERS: 2
 * PARAMETER NAMES: char∗ messagePtr , uint8_t sizeOfArray
 * PURPOSE: THIS FUNCTION SCROLLS THROUGH THE RECEIPE DISPLAYED ON THE LCD
 */
```

```
354  void scroll_down()
     {
356      // INCREMENT INDEX
         ++index;

358
         // CHECK THE BOUNDS OF INDEX
360      if(index > (msgArrSize - 2))
             index = msgArrSize - 2;

362
         // CLEAR THE LCD SCREEN AND PRINT MESSAGES TO THE LCD
364      lcd.clear();
         for(i = 0; i < 2; ++i)
366      {
             lcd.setCursor(0, i);
368          delayMicroseconds(1000);
             lcd.print(msgArr[index + i]);
370          delayMicroseconds(1000);
         }
372  }


374  /*
      * TYPE: FUNCTION
376   * NAME: pickNumber
      * RETURN: void
378   * NUMBER OF PARAMETERS: 1
      * PARAMETER NAMES: int x
380   * PURPOSE: THIS FUNCTION PICK THE NUMBER FOR THE LCD
      */
382  void pickNumber(int x) //changes value of number
     {
384      switch(x)
         {
386          default:
                 PORTB = ZERO;
388              break;
             case 1:
390              PORTB = ONE;
                 break;
392          case 2:
                 PORTB = TWO;
394              break;
             case 3:
396              PORTB = THREE;
                 break;
398          case 4:
                 PORTB = FOUR;
400              break;
             case 5:
402              PORTB = FIVE;
                 break;
404          case 6:
                 PORTB = SIX;
406              break;
             case 7:
408              PORTB = SEVEN;
                 break;
410          case 8:
                 PORTB = EIGHT;
412              break;
             case 9:
```

```
414            PORTB = NINE;
               break;
416        }
}
```

Lab02.ino