ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E
MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	6
EXPERIMENT DATE	:	25.12.2020
LAB SESSION	:	FRIDAY - 15.30
GROUP NO	:	G11

GROUP MEMBERS:

150170063	:	BURAK ŞEN
150180080	:	BAŞAR DEMİR
150190719	:	OBEN ÖZGÜR

FALL 2020

Contents

1 INTRODUCTION

In this week's experiment, we have designed our Arduino circuitries given in the experiment using timer interrupt feature. In the experiment, we have calculated our clock pulse according to our desired period and find prescalar values and compare numbers. With this timer interrupt we designed some patterns. We also learned how to use LCD display.

2 MATERIALS AND METHODS

- Tinkercad
- Arduino Uno R3
- 4 Transistors
- 4 Sevent Segment Displays
- 8 LEDs
- 4 Switches
- 1 Buzzer
- 1 LCD
- 1 Potentiometer

2.1 PART 1

In this part of the experiment we implemented two patterns with time interrupt. Firstly we determine our pins state and after that we block all the interrupts with noInterrupt function to set our intterrupt operations. After this we had to learn how the time interrupt work. We found out our microcontroller's clock is 16Mhz (which is $16 * 10^6$ hz) with this we could find our desired clock times. If we want our clock with 0.5 second period we could achive this with this $0.5 * 16 * 10^6$ which is equal to 8000000 clock pulse. This number is really big for us to implement our patterns. After some research we encounter with prescalar values. With these values we can decrease the clock cycle numbers to reasonable values. To implement this firstly we had to set TCCR1A register to 0 for enabling value compare. For setting prescalar value we have to user TCCR1B register's prescalar select bits. We learned that there are three clock select bits (CS10, CS11, CS12). With these bits we can easily set our prescaler values to 0, 1, 8, 64, 256 and 1024 in this part we used prescalar value 256. If we use prescalar value 256 we have to calculate our compare value. We can calculate this by dividing our original clock pulse number by 256, 8000000/256 = 31250 as we can see out compare number is 31250, to set our prescalar we have to make CS12 1, CS11 0 and CS10 0 after this we just have to make or operation with TCCR1B. After this we set our timer (which is counting up for each 256 clock pulse now) to 0. We can set our timer with TCNT1 = 0;

To compare our timer with our compare number we had to learn some other things. We found out there is an output compare register that compares the timer and whatever number you provided. We set our compare value to output compare register (OCR1A). For compare operation to work we had to enable timer interrupt mask register's (TIMSK1's) output compare match interrupt enable (OCIEA) bit to 1 like this TIMSK1 |= (1 ¡¡ OCIE1A). After that we just enabled all of the interrupt and attached an interrupt to our button.

When timer interrupt happens out timer function executed. Because we used TIMER 1 and OCR1A we have to implement our timer interrupt function ISR(TIMER1_COMPA_vect) like this, with TIMER1_COMPA_vect we can execute our timer interrupt function. In this function firstly we look which pattern are we in if our pattern number is 0 we execute the first pattern, if it is 1 we execute the second pattern.

In the first pattern there are two shift operations. One of the lights shift right and other one shift left all the time. We just created two variables for these two lights and after we shift them accordingly we just make an or operation and assign them to another variable. With these we can make our pattern as desired behaviour. If these variables are 0 which means they shift out from the registers, we just set them according to where they came from, and we or them like previous step. Finally if we want to show this pattern on the LEDs we just shift 6 times to left and assign it to PORTD (because first two LEDs on PORTD), and we shift 2 times to right and assign it to PORTB (becasue last six LEDs on PORTB).

In the second pattern there is a LED that lights up when the counter value is even, and there is another LED that walks between the sides. Firstly we implemented the LED that walks between the sides, if we divide the counter by 7 and take it's modula by 2 we can find which way our LED is going, and according to this we shifted our LED, after this we just take the counter's modula by 2 we and find if counter is even or not, if it is even we just make an or operation with the first LED and assign it to another variable, if it is not even we just assign the first LED to the other variable. And finally we just assign the variable's value like in the previous paragraph, and lastly we increment our counter by 1 and, reset our timer.

In the interrupt button part we attached an interrupt function to pin 3. When we press the button our interrupt function (changePattern) immediately called. In the changePattern function we change the pattern, after changing the pattern we look which pattern are we changed and according to that we set starting positions of the leds. And Finally we reset our timer and counter.

Our implementation is given below.

```
int comp = 31250;
int timer = 0;

byte pattern = 0;

const byte button = 3;

void changePattern();

int counter = 0;
int temp3;
byte temp0 = 0b00010000;
byte temp1 = 0b00001000;
byte temp2 = 0b00000000;

byte twos = 0b10000000;

void setup()
{
  DDRD = 0b11110111;
  DDRB = 0b111111;

  PORTD = 0b00000000;
  PORTB = 0b000110;

   // initialize timer1
   noInterrupts();
  // disable all interrupts
  TCCR1A = 0;

  TCCR1B |= (1 << CS12);
  // 256 prescaler
  TCCR1B &= ~(1 << CS11);
  TCCR1B &= ~(1 << CS10);

  TCNT1  = timer;
  // interrupt number 31250 lerde xd
  OCR1A = comp;
  // compare match register

  TIMSK1 |= (1 << OCIE1A);
  // enable timer compare interrupt
  interrupts();
  // enable all interrupts
  attachInterrupt(digitalPinToInterrupt(button),
                  changePattern, RISING);
}

ISR(TIMER1_COMPA_vect)
{// timer compare interrupt service routine

  if(pattern == 0) //first patter
  {
    temp0 = temp0 << 1; //shift the value by 1 to left
    temp1 = temp1 >> 1; //shift the value by 1 to right

    temp2 = temp0 | temp1;// or the two values

    if(temp2 == 0)//if our value is 0
    {
      temp1 = 0b10000000;//jump to start
      temp0 = 0b00000001;

      temp2 = temp0 | temp1;//or the value
    }


    PORTD = temp2 << 6;//shift the value by 6
    PORTB = temp2 >> 2;//shift the value by 2
  }
  else
  {

    if(counter%2 ==0){
      //calculate if counter is even or not
      temp2 = temp0|twos;
      //or the value with 0b10000000
    }else{
      temp2 = temp0;
    }
    if((counter/7)%2 == 0){
      //calculate if our led is going left or right
        temp0 = temp0 << 1; // shift left
    }else{
        temp0 = temp0 >> 1;//shift right
    }



    PORTD = temp2 << 6;
    PORTB = temp2 >> 2;
  }
  counter++;//increment counter
  TCNT1 = timer;//reset timer
}

void loop()
{
}

void changePattern()
```

```
{
  pattern = !pattern;//change the pattern
  if(pattern == 0)//pattern 1
  {
    temp0 = 0b00010000;//set the starting values
    temp1 = 0b00001000;

        PORTD = 0b00000000;
        PORTB = 0b000110;
  }
  else
  {
        temp0 = 0b00000001;//set the starting values
    temp1 = 0b10000000;

    PORTD = 0b01000000;
        PORTB = 0b100000;
  }
  TCNT1 = timer;//reset timer
  counter = 0;//reset counter
}
```

2.2 PART 2

In this part of the experiment we implemented a stopwatch that counts in centiseconds with 3 buttons. First button starts the stopwatch, second button displays the lap info and third buttons resets the stopwatch.

At first, we have defined our variables for interrupt buttons and temporary variables to manage our counting effect. Then we have defined our variables for lap function. Finally, we have defined our byte array in order to display numbers on seven segment displays.

For counting the time, we have used the timer interrupt functionality of Arduino as in the previous part of the experiment accordingly.
For starting button of the counter, we have designed an external interrupt subroutine called “start”. Start interrupt sets the start flag of the circuit which will later be used

inside the main loop.
For lap button of the counter, we have designed another external interrupt subroutine called “lap”. Lap subroutine calculates the time past in that specific lap and prints it with the total lap count and total time past.

In the loop part, we have updated each seven segment display inside each if statement using our flag variable. We have set the increment for centiseconds accordingly.

Our implementation is given below.

```
int comp = 20000;
int timer = 0;
int prev1 =0;
const byte button_first = 2;
const byte button_second = 3;

int lap_counter = 0;
int previous_lap_time = 0;
int current_time = 0;

int start_flag =0;
int display_number=0;
int flag =1;

byte sevensegment0[16] = {
  0b000,
  0b100,
  0b001,
  0b000,
  0b100,
  0b010,
  0b010,
  0b000,
  0b000,
  0b000,
  0b000,
  0b110,
  0b011,
  0b100,
  0b011,
  0b011
};

byte sevensegment1[16] = {
  0b00010000,
  0b11110000,
  0b00100000,
  0b01100000,
  0b11000000,
  0b01000000,
  0b00000000,
  0b11110000,
  0b00000000,
  0b01000000,
  0b10000000,
  0b00000000,
  0b00010000,
  0b00100000,
  0b00000000,
  0b10000000
};

void start(){
  start_flag = 1;
}

void lap(){ //lap interrupt
  lap_counter++;
  int current_lap = current_time - previous_lap_time;
  previous_lap_time = current_lap;

  float printed_lap = current_lap/100.0;
  float printed_time = current_time/100.0;

  Serial.print("Lap Number:");
```

```
    Serial.println(lap_counter);
    Serial.print("Lap Time:");
    Serial.println(printed_lap);
    Serial.print("Total Time:");
    Serial.println(printed_time);
}

void setup()
{
    //output and input sets
    DDRD = 0b11110011;
    DDRB = 0b011111;
    DDRC = 0b111111;
    Serial.begin(9600);

    PORTD = 0b00000000;
    PORTB = 0b000110;

    // initialize timer1
    noInterrupts();             // disable all interrupts
    TCCR1A = 0;

    TCCR1B &= ~(1 << CS12);     // 8 prescaler
    TCCR1B |=  (1 << CS11);
    TCCR1B &= ~(1 << CS10);

    TCNT1  = timer;

    OCR1A = comp;               // compare match register

    TIMSK1 |= (1 << OCIE1A);    // enable timer compare interrupt
    interrupts();               // enable all interrupts
    attachInterrupt(digitalPinToInterrupt(button_first), start, RISING);
    attachInterrupt(digitalPinToInterrupt(button_second), lap, RISING);
}

ISR(TIMER1_COMPA_vect)          // timer compare interrupt service routine
```

```
{
  if(start_flag){
    current_time++;
    if(current_time == 10000){
      current_time=0;
    }
  }
  TCNT1=0;
}



void loop()
{
  if(PINB>>5 == 0b000001){
        lap_counter = 0;
    previous_lap_time = 0;
    current_time = 0;
    start_flag =0;
  }

 int current = millis(); //take current time

 if (current - prev1 >= 2 && flag == 1) { //if current
 time - previous time is 5ms and flag is 1 (for the
 first delay)
    prev1 = current;     // set current time to previous time
        display_number = current_time%10;
        PORTC = 0b111011;
        PORTD = sevensegment1[display_number];
        PORTB = sevensegment0[display_number];
    flag =2; // set second if statement's flag
  }
  if (current - prev1 >= 2 && flag == 2) { //if current time - previous
  time is 5ms and flag
  is 1 (for the first delay)
    prev1 = current;     // set current time to previous time
```

```
        display_number = (current_time/10)%10;
        PORTC = 0b110111;
        PORTD = sevensegment1[display_number];
        PORTB = sevensegment0[display_number];
    flag =3; // set second if statement's flag
  }

   if (current - prev1 >= 2 && flag == 3) { //if
   current time - previous time is 5ms and flag is 1
   (for the first delay)
    prev1 = current;     // set current time to previous time
        display_number = (current_time/100)%10;
        PORTC = 0b101111;
        PORTD = sevensegment1[display_number];
        PORTB = sevensegment0[display_number];
    flag =4; // set second if statement's flag
  }

    if (current - prev1 >= 2 && flag == 4) { //if
    current time - previous time is 5ms and flag is 1
    (for the first delay)
    prev1 = current;     // set current time to previous time
        display_number = (current_time/1000)%10;
        PORTC = 0b011111;
        PORTD = sevensegment1[display_number];
        PORTB = sevensegment0[display_number];
    flag =1; // set second if statement's flag
  }
}
```

2.3 PART 3

In this part of the experiment we were expected to write our names and surnames on each column of 16x2 LCD in our circuitry. We were restricted to use libraries' default functions for our LCD manipulation. Therefore, we have checked the documents given to us by the assistants to perform our experiment.

At first we had to set configure our LCD settings because LCD normally works in 8 bit mode but we had only 4 bits connected to our circuitry. The function named "initLCD()" given to us by the assistants configures the circuit accordingly. Now that we have configured the circuit inside setup mode, we can send data to our LCD in two cycles since it is in 4 bit mode.

After initialization, we were required to implement sendCMD, sendChar and triggerEnable functions. "triggerEnable" just changes the value of EN to high then, makes it low so, we have easily implemented it PORTB.
For "sendCMD" we were supposed to send the required command in two cycles as leftmost 4 bits first then, rightmost 4 bits. Therefore we have set leftmost bits to PORTD then, we have triggered EN and sent the rightmost bits to PORTD using rightshift and logical and operation.
For "sendChar" operation we have taken two parameters called letter and index. "letter" is the character to be written and "index" is the location for our characters to be written. First we have set the command accordingly. The given command makes our circuit point to the memory location determined by the index because characters will be to written to corresponding memory locations to be displayed. After cursor is set, now it's time to write to memory. Given character is also sent in two cycles using triggerEnable. For writing into memory Write signal is also sent to pin 2 by logical or operation.

After function implementations are done, loop can be explained. We have defined our names and surnames inside char arrays to be printed on LCD. Inside our loop function we have traversed our name array first and the index. After our names are written to upper row we have traversed our surnames array with 64 offset for cursor to point on lower row.

Our implementation is given below.

```
//char arrays to be displayed
char names[16] = {'b','u','r','a','k','
','b','a','s','a','r',
' ','o','b','e','n'};
char surnames[16] = {'s','e','n','
','d','e','m','i','r',
' ','o','z','g','u','r', ' '};
```

```
//function to write on LCD
void sendChar(byte letter, int index){
  sendCMD((index | 0b10000000)); //set the index at corresponding location
  PORTD = ((letter & 0b11110000)) | 0b00000100; //upper part of the letter
  triggerEnable();
  PORTD = ((letter & 0b00001111)<< 4) | 0b00000100;
  //lower part of the letter and write command
  triggerEnable();
  waitMicros(55); //wait
}

//function to send command
void sendCMD(byte cmd){
  PORTD = ((cmd & 0b11110000)); //send upper part of the command
  triggerEnable();
  PORTD = ((cmd & 0b00001111)<< 4); //send lower part of the command
  triggerEnable();
  waitMicros(55); // wait
}

void triggerEnable()
{
        PORTB = 0b000001;
        PORTB = 0b000000;
}

void waitMicros(int time)
{
  long start_time = micros();
  while(micros() - start_time >= time);
}

void waitMillis(int time)
{
  long start_time = millis();
  while(millis() - start_time >= time);
}
```

```
void initLCD(){

  PORTD = PORTD & B11110011;
   waitMillis(100);

  PORTD = (PORTD & B00001111) | B00110000;
   triggerEnable();
   waitMillis(5);

  PORTD = (PORTD & B00001111) | B00110000;
   triggerEnable();
   waitMicros(150);

  PORTD = (PORTD & B00001111) | B00110000;
   triggerEnable();
   waitMicros(150);

  PORTD = (PORTD & B00001111) | B00100000;
   triggerEnable();
   waitMicros(150);

  PORTD = (PORTD & B00001111)|B00100000;
   triggerEnable();
  PORTD = (PORTD & B00001111)|B10000000;

   triggerEnable();
   waitMicros(55);

  PORTD = (PORTD & B00001111);
   triggerEnable();
  PORTD = (PORTD & B00001111)|B10000000;
   triggerEnable();
   waitMicros(55);

  PORTD = (PORTD & B00001111);
   triggerEnable();
  PORTD = (PORTD & B00001111)|B00100000;
  triggerEnable();
  waitMillis(5);


  PORTD = (PORTD & B00001111);
  triggerEnable();
  PORTD = (PORTD & B00001111)|B01100000;
  triggerEnable();
  waitMicros(55);

  PORTD = (PORTD & B00001111);
  triggerEnable();
  PORTD = (PORTD & B00001111)|B11100000;

  triggerEnable();
  waitMicros(55);
}


void setup() {
  DDRD = 0b11111111; //output
  DDRB = 0b11111111; //output
  initLCD(); //start lcd
}

int counter =0;
int flag =0;
int isFinished =0;
void loop() {
  if(!isFinished){
    if(flag ==0 ){ //go until names are written
      sendChar(names[counter], counter); //send names to write
      counter++;
      if(counter ==16){ //when name writing is done
        flag =1; //set flag to 1
        counter =0; //decrease counter
      }
        }
    if(flag ==1 ){
      sendChar(surnames[counter], counter+64); //send surnames
      //64 is going to line below
      counter++;
      if(counter ==16){ //when surname writing is done
        flag =0;
        counter =0;
        isFinished=1; //set finished flag to one
      }
        }
  }
}
```

2.4 PART 4

In this part of the experiment we implemented a countdown timer. Firstly we used the same sendChar, sendCmd, initLcd, waitMicros, waitMillis functions from the part 3. We created a character array for lable to first row of the 16x2 lcd display. In the setup function we used the same timer interrupt implementation in the other parts but we change some input values. For 1s timer interrupt when our prescalar value 8 our interrupt compare value must be 62500 from this formula $\frac{(period*16*10^6)}{prescalar}$ = compare value.

In the timer interrupt function we basicly created a countdown mechanism with hour, minute and second variables. Before the timer interrupt we set timer values by hand and in the timer interrupt function each second our values count down by one. If second variable is not 0 we basically decrease the value of second variable and we print in the 16x2 display in here we just change the second variable becaus of that we just print the value of the second variable on top of the previous value. With this mechanism we don't have to refresh the display. If second variable is 0 then we look if minute variable is 0 or not. If minute variable is not 0 we decrease the value of the minute variable and set the second variable to 59, we display both second variable and minute variable like in the previous step, if minute variable is 0 then we have to look if hour variable is 0 or not. If hour variable is not 0 we can decrease the hour variable and set both minute and second variables to 59 after this we have to display all of the changed values on the 16x2 display. If it is 0 we just display all of the values. WHen we want to display integer value we had to convert them to ascii representations, to convert them to ascii we had to add all of the values 48.

In the loop function, we firstly control if our hour, minute and second variables is 0 or not. If they all are 0 we basically just vibrate the piezo. After that we are looking if our label (first line in the 16x2 display) is printed or not. If it is not printed we print all of the characters from the title array we created. In the title array there is 16 character and for printing the title we have to create a counter that counts up to 16. We use this counter for determining the cursor for each of the characters. If counter is 16 we basically make counter 0 and set the fits line finished flag to 1. If the first line finished flag is set our program will not be refreshing the first line again. After this we created another flag for the counting display system. In here we basically display hour, minute and second values. In the loop function we just display the starting value, after this we set the flag to 1. When flag value is 1 we would not be able to enter this part of the code.

Our implementation is given below.

```
//array that contains the label
char title[16] = {
  'C','O','U','N','T',
  'D','O','W','N',' ',
  'T','I','M','E','R',':'};

//timer comparison number
int comp = 62500;
//timer variable
int timer = 0;

//set cursor function sets cursor at an index
void setCursor(int index){//change the ddram address
   sendCMD((index | 0b10000000));
   waitMicros(55);
}

void triggerEnable()//Enable the lcd
{
        PORTB = 0b000001;//enable
        PORTB = 0b000000;//disable
}

//send char function prints characters on lcd
void sendChar(int cursor, byte cmd){
  setCursor(cursor); // first sets the cursor
  PORTD = ((cmd & 0b11110000)) | 0b00000100;
  // send upper 4 bit of the character
  triggerEnable(); // enable lcd
  // send lower 4 bit of the character
  PORTD = ((cmd & 0b00001111)<< 4) | 0b00000100;
  triggerEnable(); // enable lcd
}

//send command function sends commands
```

```
// we use this to cursor set
void sendCMD(byte cmd){
  //send upper 4 bits of the command
  PORTD = ((cmd & 0b11110000));
  triggerEnable();// execute the command
  //send lower 4 bits of the commands
  PORTD = ((cmd & 0b00001111)<< 4);
  triggerEnable();// execute the command
}



void waitMicros(int time){
  long start_time=micros();
  while(micros()-start_time < time);
}

void waitMillis(int time){
  long start_time=millis();
  while(millis()-start_time < time);
}


void initLCD(){
  PORTD= PORTD&B11110011; //Clear RS and R/W
  waitMillis(100);  //Wait 100 ms

  PORTD= (PORTD&B00001111)|B00110000; //Special case of 'Function Set'
  triggerEnable();               //Send Enable Signal
  waitMillis(5);   //Wait 5ms

  PORTD= (PORTD&B00001111)|B00110000; //Special case of 'Function Set'
  triggerEnable();               //Send Enable Signal
  waitMicros(150);

  PORTD= (PORTD&B00001111)|B00110000;
  //Function set, Interface is 8 bit longs
```

```
triggerEnable();                //Send Enable Signal
waitMicros(150);

PORTD= (PORTD&B00001111)|B00100000;
//Initial 'Function Set' to change interface
triggerEnable();                //Send Enable Signal
waitMicros(150);

PORTD= (PORTD&B00001111)|B00100000;
//'Function Set' DL=0 // Dataline 8bits
triggerEnable();                //Send Enable Signal
PORTD= (PORTD&B00001111)|B10000000;
//'Function Set' N=1 //2 lines
// F =0 5x8 dots
triggerEnable();                //Send Enable Signal
waitMicros(55);

PORTD= (PORTD&B00001111); //D splay On Of control
triggerEnable();                //Send Enable Signal
PORTD= (PORTD&B00001111)|B10000000;
triggerEnable();                //Send Enable Signal
waitMicros(55);

PORTD= (PORTD&B00001111); //Clear display
triggerEnable();                //Send Enable Signal
PORTD= (PORTD&B00001111)|B00010000;
triggerEnable();                //Send Enable Signal
waitMillis(5);

PORTD= (PORTD&B00001111); //Entry mode set
triggerEnable();                //Send Enable Signal
PORTD= (PORTD&B00001111)|B01100000; //ID=1 Increment, S=0
triggerEnable();                //Send Enable Signal
waitMicros(55);

PORTD= (PORTD&B00001111); //D splay On Of control
triggerEnable();                //Send Enable Signal
```

```
  PORTD= (PORTD&B00001111)|B11100000;
  //Display =1
  //Cursor =1
  //Blink cursor=0
  triggerEnable();                //Send Enable Signal
  waitMicros(55);
}

void setup() {
  DDRD = 0b11111111; // set portd registers as output
  DDRB = 0b11111111; //set portb registers as input
  initLCD(); //initialize lcd panel

  // initialize timer1
  noInterrupts();             // disable all interrupts
  TCCR1A = 0;

  TCCR1B |= (1 << CS12);      // 256 prescaler
  TCCR1B &= ~(1 << CS11);
  TCCR1B &= ~(1 << CS10);

  TCNT1  = timer;
  // interrupt number 31250 lerde xd
  OCR1A = comp;               // compare match register

  TIMSK1 |= (1 << OCIE1A);   // enable timer compare interrupt
  interrupts();               // enable all interrupts
}

int hour = 0;
int minute = 1;
int second = 1;

ISR(TIMER1_COMPA_vect)
{
  // timer compare interrupt service routine
  TCNT1 = timer; // set timer to 0
```

```
if(second == 0){ // if seconds are 0
  if(minute == 0){ // if minutes are 0
    if(hour ==0){ // if hours are 0
      sendChar(64,hour/10+48);
      // send first digit of the hour in ascii
      sendChar(65,hour%10+48);
      // send second digit of the hour in ascii
      sendChar(67,minute/10+48);
      // send first digit of the minute in ascii
      sendChar(68,minute%10+48);
      // send second digit of the minute in ascii
      sendChar(70,second/10+48);
      // send first digit of the second in ascii
      sendChar(71,second%10+48);
      // send second digit of the second in ascii
    }else{
      hour--; // if hour is not 0 decrease by 1
      minute = 59; //set minute to 59
      second = 59; //set second to 59
      sendChar(64,hour/10+48);
      sendChar(65,hour%10+48);
      sendChar(67,minute/10+48);
      sendChar(68,minute%10+48);
      sendChar(70,second/10+48);
      sendChar(71,second%10+48);
    }
  }else{
    minute--; //if minute is not 0 decrease by 1
    second = 59;      // set seconds to 59
    sendChar(67,minute/10+48);
    sendChar(68,minute%10+48);
    sendChar(70,second/10+48);
    sendChar(71,second%10+48);

  }
}else{
    second--; // if second is not 0 decrese by 1
    sendChar(70,second/10+48);
    sendChar(71,second%10+48);
  }
}


int counter =0;
int flag = 0;
int isFirstLineFinished =0;
int prev = 0;
void loop() {
  int current = millis();
  if(current - prev >= 5) // 5ms delay
  {
    if(hour==0&&minute==0&&second ==0){
        PORTB = 0b001000;
        // if time is up vibrate the piezo
        }
        if(!isFirstLineFinished){
     // if first line is not written
     sendChar(counter, title[counter]);
     //send first lines chars
     counter++; // increase counter by 1
     if(counter ==16){  // if counter is 16
       counter =0;      //set counter to 0
       isFirstLineFinished = 1;
       // and set finish first line
     }
        }
        if(!flag){ // if flag is not set
        sendChar(66,':');
        // set chars to second line
        sendChar(69,':');
                sendChar(64,hour/10+48);
                sendChar(65,hour%10+48);
                sendChar(67,minute/10+48);
                sendChar(68,minute%10+48);
                sendChar(70,second/10+48);
                sendChar(71,second%10+48);
                sendChar(72,' ');
                sendChar(73,' ');
                sendChar(74,' ');
        sendChar(75,' ');
                sendChar(76,' ');
                sendChar(77,' ');
                sendChar(78,' ');
                sendChar(79,' ');

                flag = 1; // set flag to 1
        }
    prev = current;
  }
}
```

3 RESULTS

3.1 PART 1

After our code implementation, we observed our patterns accordingly

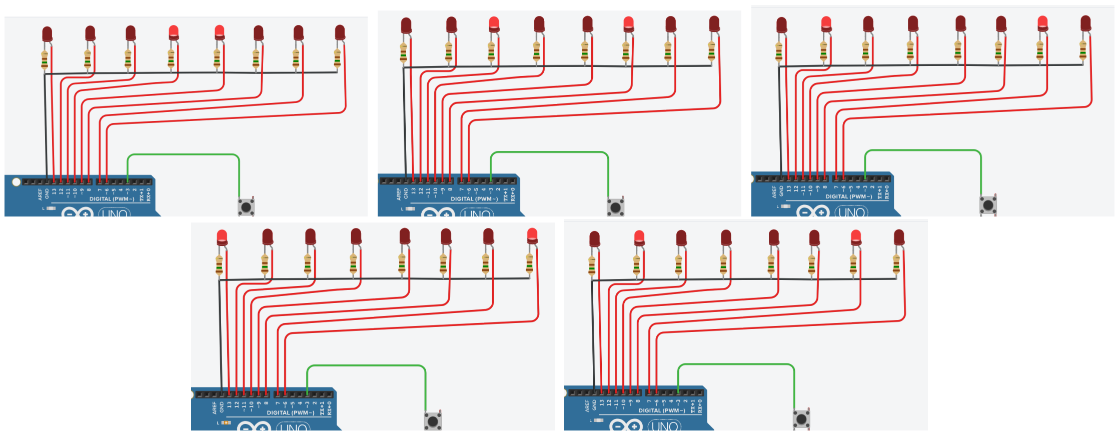


Figure 1: First Pattern

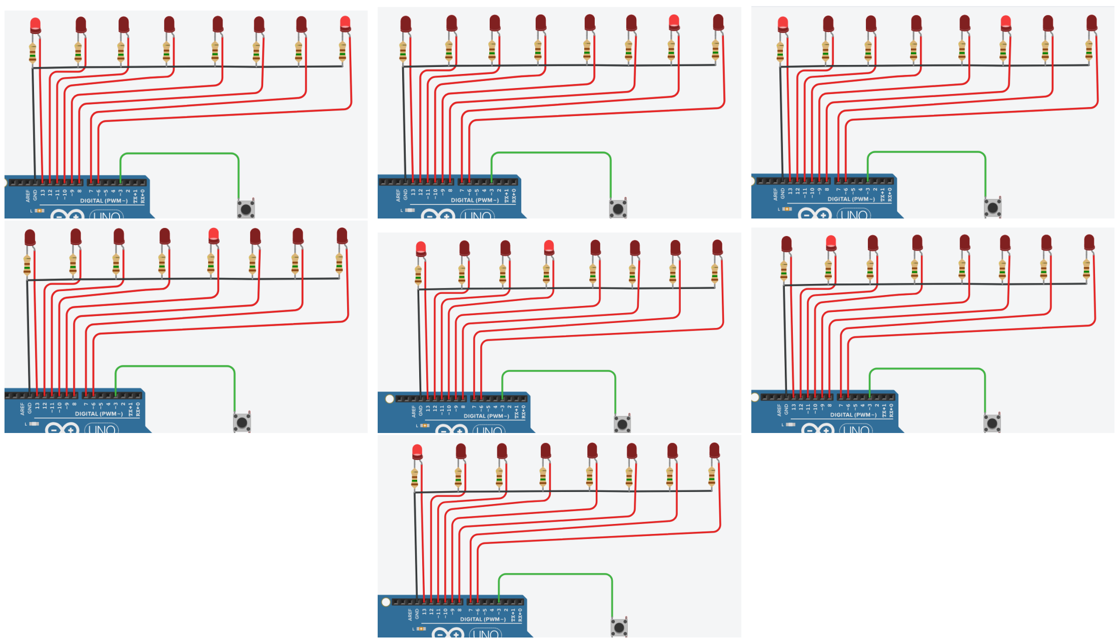


Figure 2: Second Pattern

3.2 PART 2

After our code implementation, We can use our stopwatch easily as shown below.

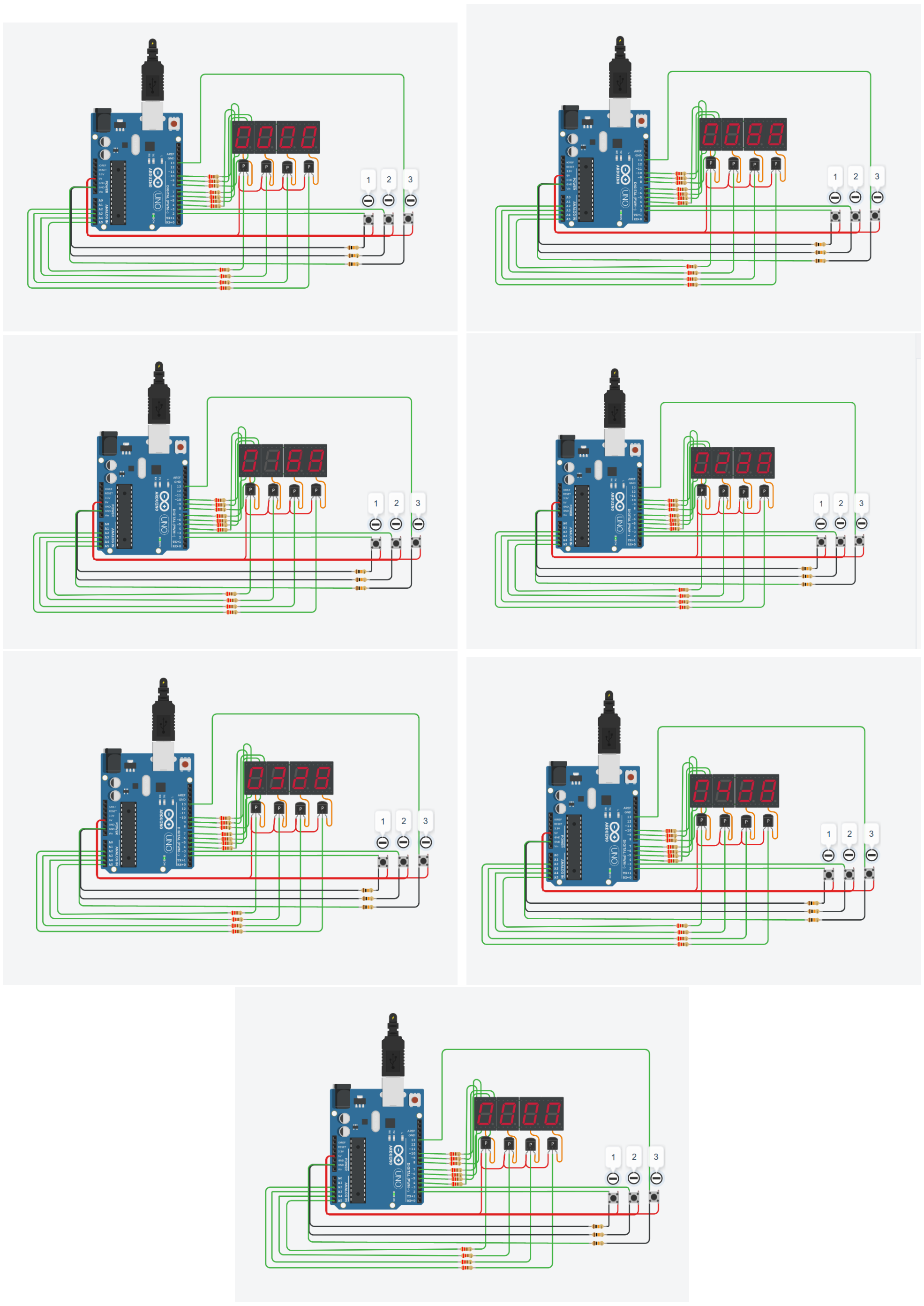
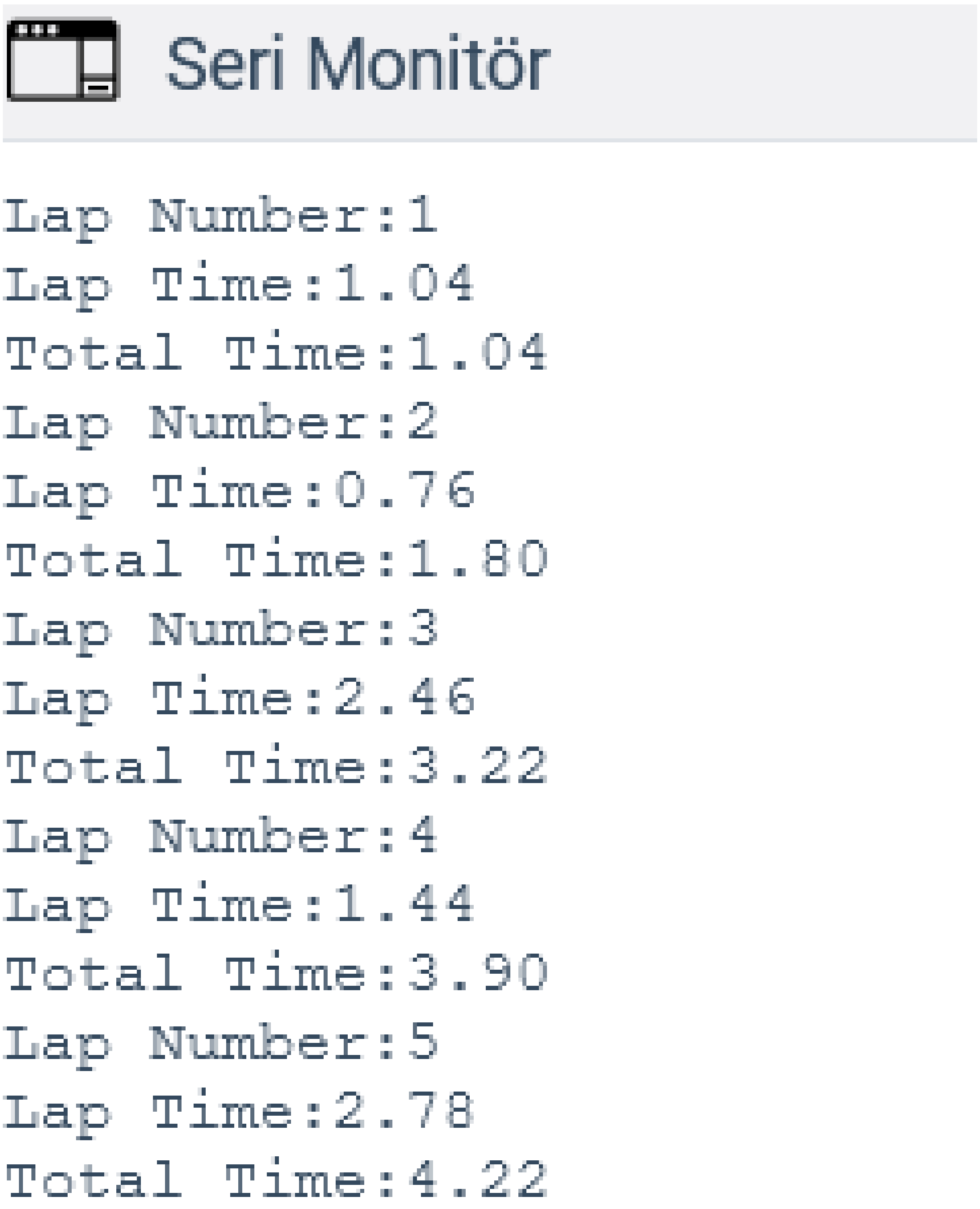


Figure 3: Stopwatch with starting and counting and finally reset



```
Lap Number:1
Lap Time:1.04
Total Time:1.04
Lap Number:2
Lap Time:0.76
Total Time:1.80
Lap Number:3
Lap Time:2.46
Total Time:3.22
Lap Number:4
Lap Time:1.44
Total Time:3.90
Lap Number:5
Lap Time:2.78
Total Time:4.22
```

Figure 4: Stopwatch lap information on serial monitor

3.3 PART 3

We have observed that our circuit works as we have expected. Names are displayed clearly.

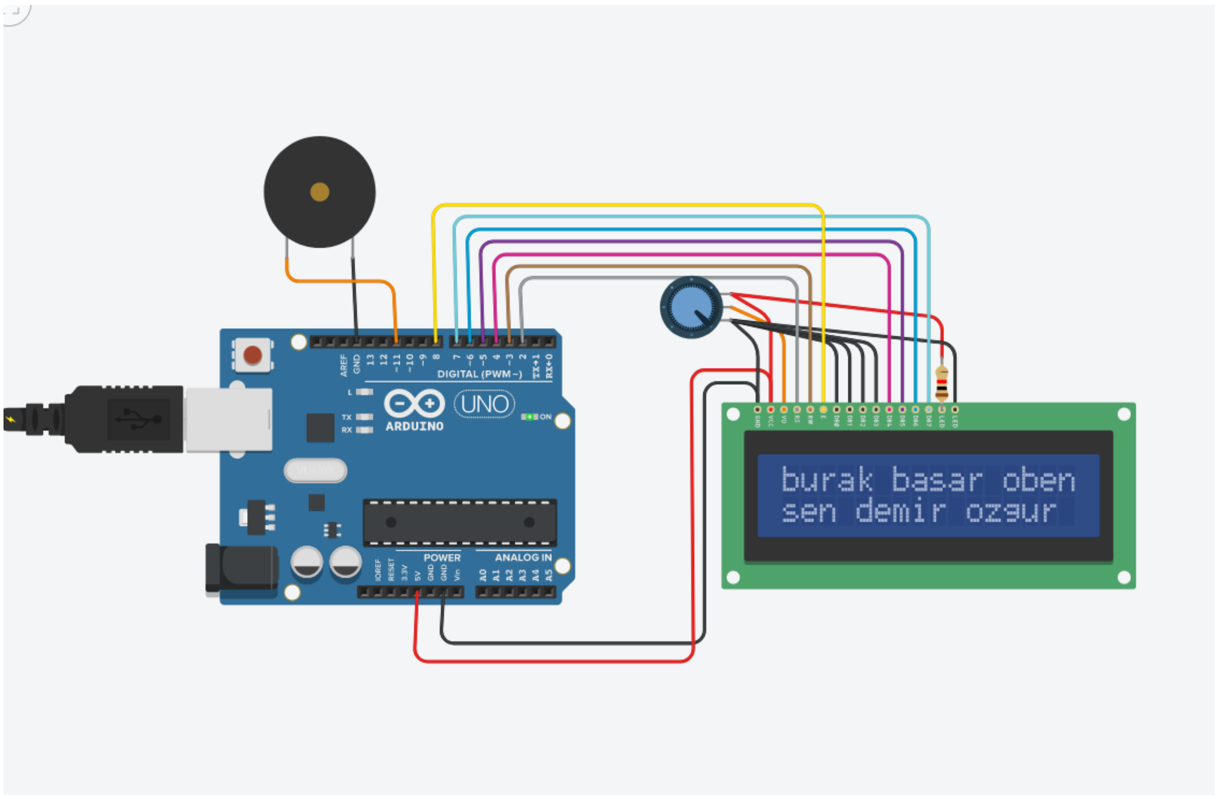


Figure 5: Our names and surname

3.4 PART 4

We have observed that our countdown works perfectly and when it is zero our piezo is vibrating.

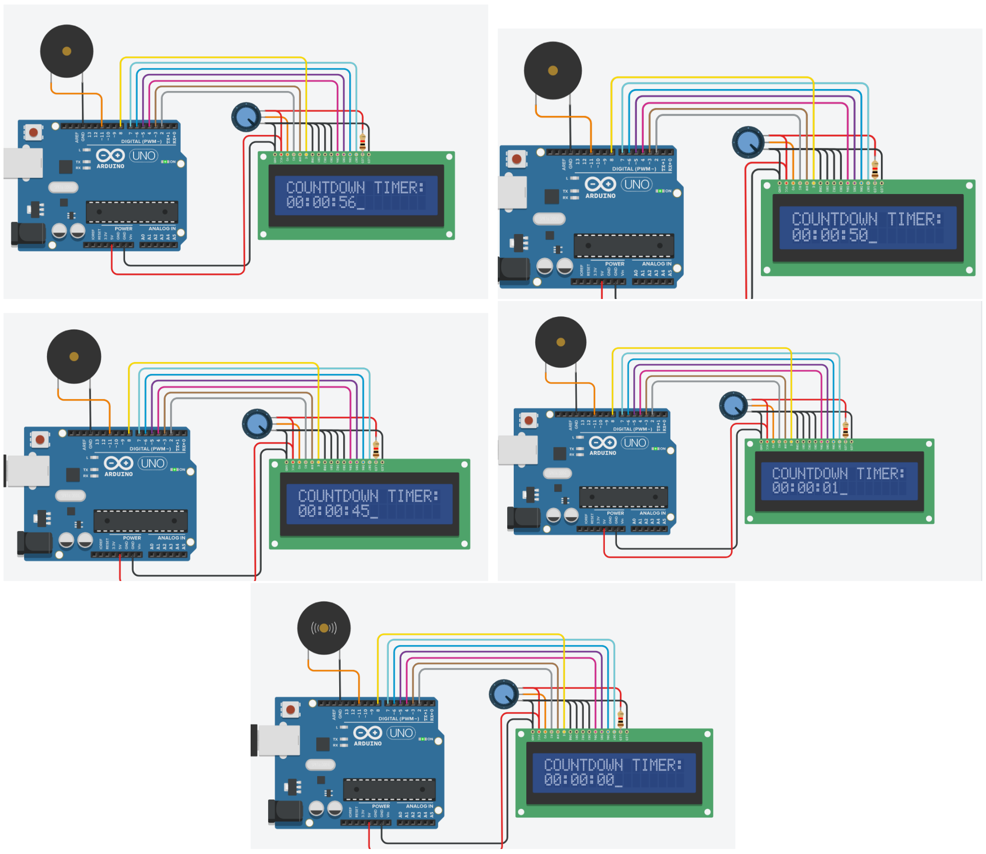


Figure 6: Coundown display

4 DISCUSSION

4.1 PART 1

In this part of the experiment, we have implemented some patterns with LEDS similar to previous weeks. We had regular a change mode button and we were supposed to implement a interrupt subroutine for changing mode. Thanks to previous week's experiment we have easily implemented the regular interrupt. The hard part of this experiment for us was to implement timer interrupt.

We have searched for implementation of timer interrupts. After realizing that Arduino's clock works in 16Mhz we tried to adapt pulse number according to the clock of Arduino so that we would make our circuit have a reasonable value for pattern designs. We manipulated T-timer/counter control registers and output compare register after our acknowledgements.

After learning timer interrupt feature, we have widened our interrupt knowledge a lot and easily performed the experiment. Overall we were satisfied with our acknowledgments about timer interrupts.

4.2 PART 2

In this part of the experiment, we were expected to implement a stopwatch counts in centiseconds using the seven digit digit displays and serial monitor of Arduino. We had a button for starting, resetting and displaying lap information for our stopwatch. We have implemented the buttons using external interrupts and basic if-else statements easily thanks to our experience from previous experiments.

For the counting effect of our Arduino circuitry, we were required to use timer interrupt. After learning about timer interrupts in the first part of the experiment and our experience with sevent segment displays from previous weeks, we have easily conducted the experiment. We have also used the serial monitor first time other than debugging as opposed to previous weeks. We were very satisfied with our results.

4.3 PART 3

In this part of the experiment, we are expected to manipulate simple LCD display by writing our names and surnames without using any built-in function. It was looking simple because of its static structure but it was not. Firstly, we have determined subjects that are not familiar to us such as writing character to display, moving cursor, necessary commands. It was hard to understand documentation and 4-bit working principle of the display. In documentation, functions and commands are given but while learning new structure, most important part is understanding the working mechanism and if exist, its electronic background. Therefore, our start point of our learning process was learning circuitry of LCD. We have learnt its registers, character representation standards from reading documentation. Cursor movement using DD RAM was key knowledge for us about this part. We were able to access cells of the display using hard-coded addresses. After that, we have started to implementation. Coding logic was actually simple, we have used some arrays to keep our names and required functions that are given in homework file. After implementation phase, we have double checked our code to get rid from delays and increase readability.

In essence, it was a hard experiment for inexperienced people because of the lack of internet resources and long documentations but after learning the structure of the LED, it is actually easy. We thought that this part of experiment was good for us to learn a new concept about the Arduino. We are happy with our result and working labor.

4.4 PART 3

In this part of the experiment, we are expected to an timer system that sends a sound signal when it finishes using a LCD and piezo. Actually, it was an experiment that allowed us to apply all the Arduino programming structures that we have learned so far and tested our knowledge. Implementing an counter system is an easy process for us. We have defined separate variables for each time unit and we have decremented each of them according to time rules. For this part, we have directly used the timer interrupt that is written for first part. LCD manipulating phase was have quite complicating design that is occurred because of the our purpose that is higher system performance. We have realized that LCD display keeps all unit of display in registers. Therefore, if we do not change any unit display, we are not required to assign same character to the same address. To reduce refresh operations and power consumption, we determined necessary time unit in every time update operation and we performed update operation to only these units. It was the key point of this experiment. Piezo usage was simple and it requires only one if statement.

In brief, this part of experiment was significant in terms of making us use what we have learnt. It was a pleasure to realize that we understood the structure and feel that we were improving.

5 CONCLUSION

Achievement of this week's experiment is to make us learn about timer interrupts and make us manipulate the LCD manually using our Arduino circuitry. After learning about the timer interrupt subroutines we have designed some patterns with LEDS and a stopwatch. For the LCD part, after reading some documentations we have learned LCD manipulation and designed a name displayer and a timer.Thanks to this week's experiment we have widened our understanding of timer interrupt cycles in LCD displays.

References

[1] Kadir Ozlem. *BLG 351E – Microcomputer Laboratory Slides*. 2020.

[2] Arduino. Arduino documentation. `https://www.arduino.cc/reference/en/`, 2018.

[3] Autodesk. Tinkercad. `https://www.tinkercad.com/`, 2011.