ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E
MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	5
EXPERIMENT DATE	:	6.12.2020
LAB SESSION	:	MONDAY - 13.30
GROUP NO	:	G9

GROUP MEMBERS:

150190702	:	Berktuğ Kaan Özkan
150180001	:	Mehmet Arif Demirtaş
150180043	:	Yusuf Alptigin Gün

SPRING 2020

Contents

1 INTRODUCTION

In this experiment, we used the interrupt pins on Arduino to display some values on seven segment displays depending on button presses. In the first part, we implemented a counter in real time that can reset and change directions based on button press interrupts. In the second part, we implemented a number game where a player is trying to guess a target number by pressing a button while a counter is counting. In the third part, we made it a two-player number game where players tried to catch each others number. The multiplexing method from the last experiment was used in parts 2 & 3 for showing numbers on serial connected displays.

2 MATERIALS AND METHODS

- Tinkercad design tool with Arduino coding was used for all parts of the experiment.
- There were 3 different circuits used for completing the experiment. Each circuits is shown before talking about how the experiments was progressed.

2.1 Part 1

The circuit used for part 1 of the experiment can be seen below.

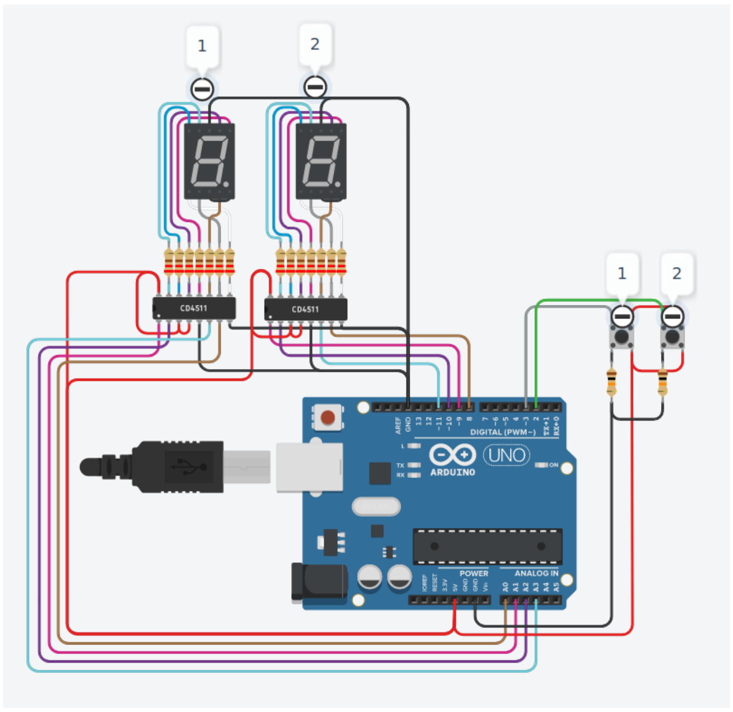

Figure 1: Board used in Part 1

In part 1 of the experiment, we implemented a 2-digit counter that counts up in range $[0, 99]$ that resets to 0 when the first button goes from 0 to 1 and starts counting in reverse direction when the second button goes from 0 to 1. We used interrupts for this task.

We declared 4 variables, for holding the counter value, for holding the time returned by millis function, for the counting direction and a constant time that determines the delay between counting. The variables that are not constant are declared as volatile, since they are changed in interrupt service routines.

We wrote 3 functions. Display function takes a counter value and shows the $counter(mod 10)$ value on the right display and tens digit on left. Reset function sets the counter to zero, resets timer and starts again. Change mode function changes the counting direction.

For the setup, we assigned our input and output ports. We used port manipulation registers. DDRB and DDRC registers are used to set right and left displays as output and DDRD is used to set buttons as input.

We used attachInterrupt function for interrupts. This function takes in 3 parameters, an interrupt number, a function name that is going to be the interrupt service routine (ISR), and a mode constant that determines when the ISR is called. We used RISING to call ISR's as soon as the button is pressed (goes from 0 to 1). Interrupt numbers are generated by another function, digitalPinToInterrupt, as advised by Arduino docs.

We also set counter and mode to 0 and true for initial values, displayed the counter for initialization. We used millis function for delay instead of delay function. For this, we set time variable to millis() value in setup. In loop, we checked if the difference between time and current millis() is greater than our delay time constant, and if not, we let the loop run. If it is equal, we reset the time variable, do the operations and let the loop continue.

For the loop, we incremented or decremented the counter based on the mode variable. We also added checks for overflow (counter going over 99) and underflow (counter going below 0). After that, we called the display function that showed the digits.

2.2 Part 2

The circuit used for part 2 of the experiment can be seen below.

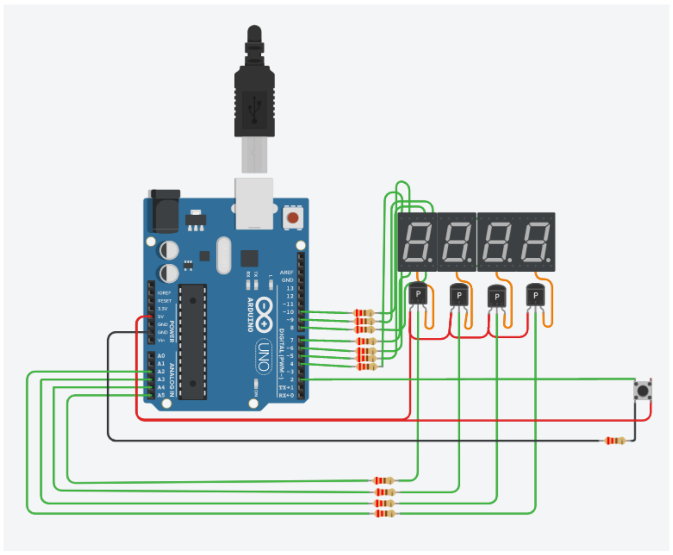

Figure 2: Board used in Part 2

In this part, a counter that counts from 0 to 15 by one per second is displayed at the leftmost 7-segment. When the button clicked, the displayed value shifts right. The aim is to find the target value by clicking the button. This target value is three digit long. Therefore, the player must enter the digits in a correct order to win. After finding the value, the program stops.

In our initialization, we have an array to transform from variable value to bit representation for 7-segments. There are also variables that hold the counter and hold the delay related times and constants.

In order to accomplish this task, we used an interrupt function to catch the button clicks. An array which has 3 elements in size was used to store pushed values. Whenever a click occurs, this function copies the values one slot right and inserts the displayed value, which is variable counter, to the array. Since there are four 7-segments connected to same ports, in order to display the counter and the elements of the array, we used the multiplexing approach from the previous experiment. The twist was the restriction of not using a delay. We introduced new variables and switch cases to simulate a delay.

In our display function, we use a variable to keep track of which 7-segment display was shown last. Based on this, we have a switch statement and the corresponding case runs. At the end of each case, we reset a disp_time variable. In our loop, we compare the disp_time and current time and if the time elapsed is greater than our propagation delay constant, we call the display function again. This helped us achieve the same effect with the previous experiment where it is seen like all four displays are on at the same time.

2.3 Part 3

The circuit used for part 3 of the experiment can be seen below.

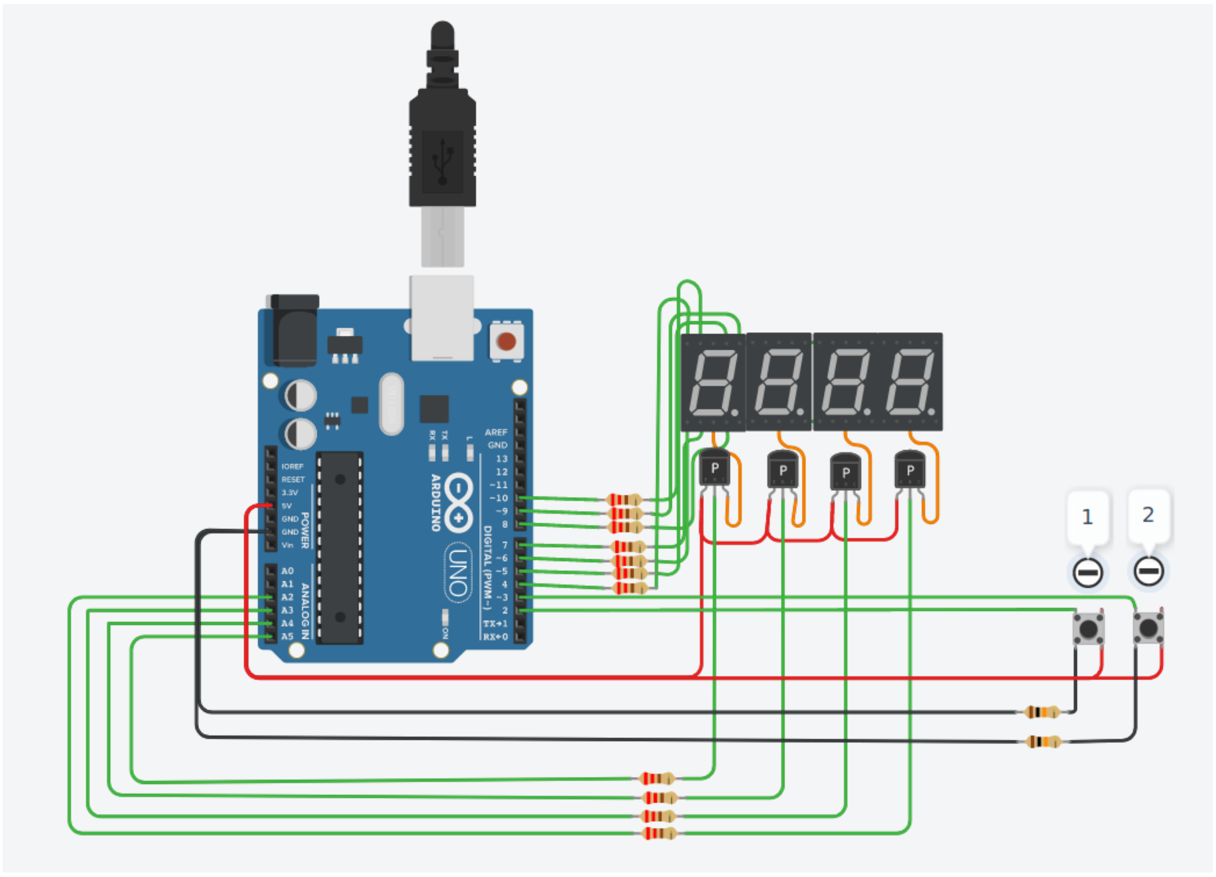

Figure 3: Board used in Part 3

In part 3 of the experiment, we used our knowledge from the first two parts to implement a two player number game. We use a similar display function to part 2, num_to_bit array for transforming numbers to 7-segment bit representations and similar interrupts.

In addition to constants and variables for delays and display, we declared 4 volatile variables that is used in ISRs for this game. These hold the counter for player 1 and player 2, the player to play now and the winner if the game has finished.

We have three new functions. Two of them are ISRs for button 1 and button 2, which are almost identical with only the player numbers exchanged. If a player presses a button, we first check if its their turn. If not, nothing happens. If it is, we check if they matched the count of the other player and in that case, winner is set to 1 and player to play is set to 0. If not, it becomes the other players turn. In only the first player's button press, we also set DDRC to output in this case because we do not want to show numbers on player 2 counter before the first player presses the button for the first time. Third new function is a simple win function that sets all displays on and shows the given number on all displays.

For setup, we set DDRB, upper half of DDRD and and two displays of DDRC to output. The other two displays will be opened with the first button press. We set some of DDRD as input ports for buttons. Then we use PORTC to set all displays to off. We attach RISING interrupts to buttons and we set counters to initial values. To avoid false winnings, we initialize player 1 to zero but player 2 to -1. We set first player to play and initialize our delay routine.

For the loop, we check if there is a player to play. If so, we check the delay. If 200ms has passed, we increment the counter of player playing just like in the other parts. After that, we check the propagation delay for displays and call display function if the time is correct. If there is no player to play, we understand that someone has won and call the win function to show the winner.

3 RESULTS

3.1 Part 1

In part 1, our circuit counted with 1 second interval up or down based on the interrupting button presses. The results of it can be seen below.

Time (s)	**Counter**	**Button 1**	**Button 2**
0	97	0	0
1	98	0	0
2	99	0	0
3	00	0	0
4	01	0	0
5	02	0	0
6	00	1	0
7	01	1	0
8	02	0	0
9	01	0	1
10	00	0	0
11	99	0	0

3.2 Part 2

In part 2, a counter that counts from 0 to 15 by one per second is displayed at the leftmost 7-segment. When the button clicked, the displayed value shifts right. The aim is to find the target value by clicking the button. This target value is three digits long. Therefore a player must enter the digits in correct order to win. After finding the value, the program stops. For target value of 247, an example run of the program can be seen in the below table. Note: Displays are numbered increasingly from left to right.

time (seconds)	1. display	2. display	3. display	4.display	button clicked
0	0	0	0	0	no
1	1	0	0	0	yes
2	2	1	0	0	yes
3	3	2	1	0	no
4	4	2	1	0	yes
5	5	4	2	1	no
6	6	4	2	1	no
7	7	4	2	1	yes
8	7	7	4	2	don't care
9	7	7	4	2	don't care

3.3 Part 3

In part 3, a two player number game is created. It starts with player 1 counting and player 2 display off; when player 1 presses the button, player 2 display starts counting. This continues until one of the players catch the others number and in that case, the winning players number is shown on all displays.

We will show three images to showcase the results, where the first player just started the game, where the game has moved forward and where the second player has won.

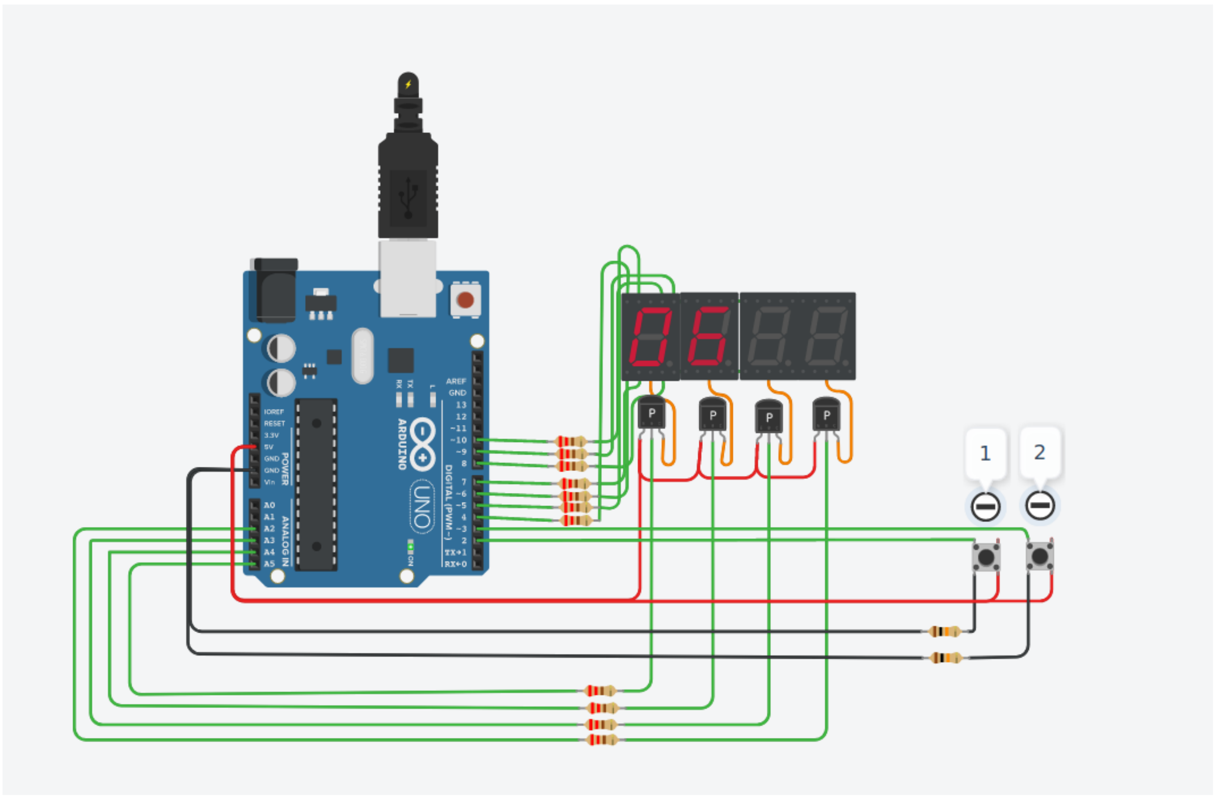

Figure 4: First round of the game

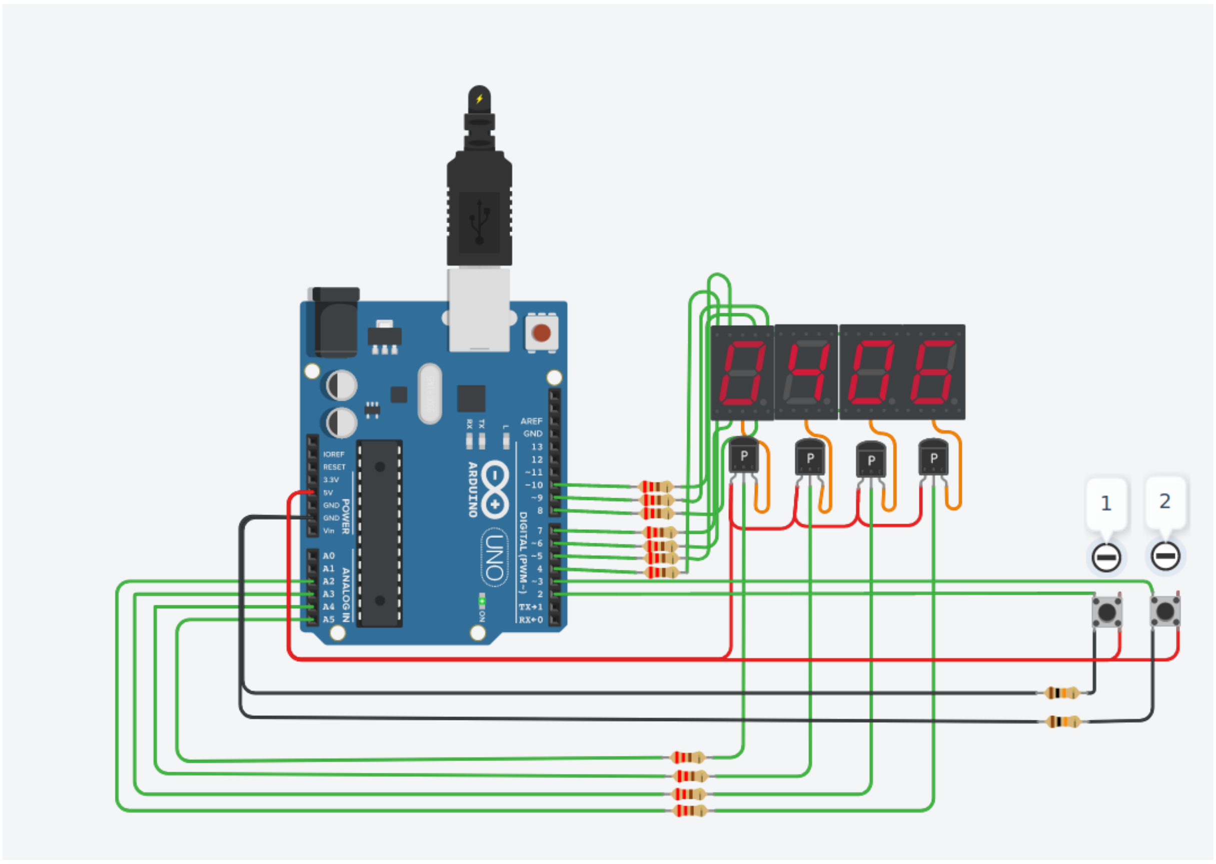

Figure 5: During game

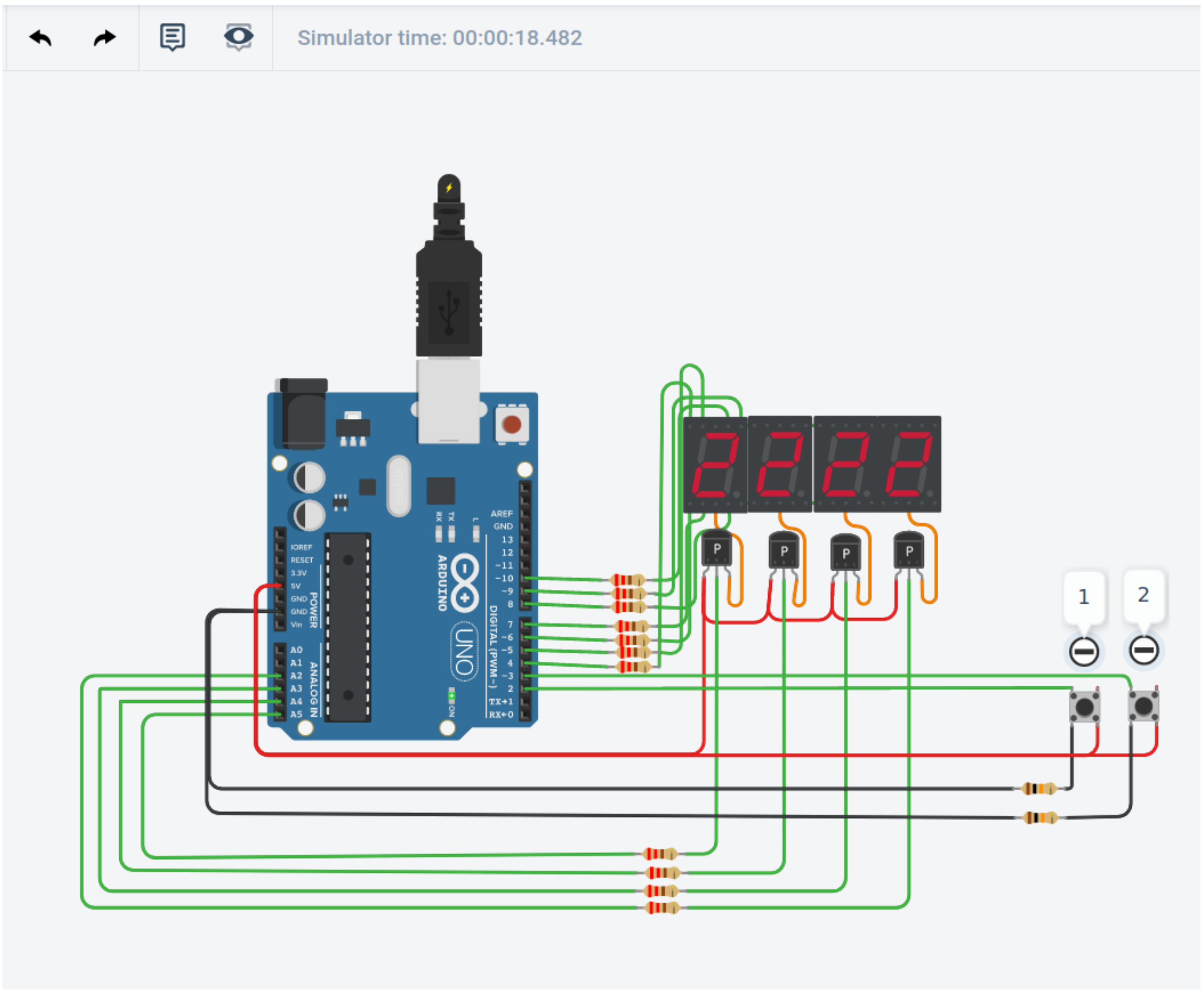

Figure 6: After player 2 has won

4 DISCUSSION

In the first part of the experiment, we used two buttons connected to interrupt pins to make an enhanced counter. We started development by implementing a counter in real time. We experimented a bit with the millis() function to make it work for 1 second delays. While implementing the delay logic, we first used a integer variable to hold the last time and compared it to current millis() output. However, our circuit broke around 33 seconds in. We realized that we needed to use a longer data type than integer so we changed to unsigned long. This way, our program can run for approximately 50 days without error. After we got our counter to work correctly, we did some research on interrupt techniques on Arduino. We found out that there is a function called attachInterrupt that binds a function to a button press. We wrote the functions for inverting the counter direction and resetting the counter. We weren't troubled by anything else so we finished our design and tested it to get results.

In part 2 of the experiment, we changed to another seven segment display model. Previous part had decoders for them, but in this part they were connected serially. So we created an array to hold bit representations to show on these displays. We also used a trick called multiplexing from the previous experiment where we rapidly show different digits by closing and opening the displays one by one so it is seen as they are shown at the same instant. Since we did not have access to the delay functions, we again used millis function to create a pseudo-delay for this propagation. We used a similar counter setup as above, used arrays to move digits to other displays and hold the moved digits. We envisioned this part as a safe where we entered the combination to open. So target number is a password and we enter the digits of the password from right to left. Counter stops when the correct combination is entered.

In part 3 of the experiment, we used an LCD display for the first time. We were able to display our names and surnames on this LCD display using the needed functions we created. This was the first time we were using an LCD so we had to take a bit of time to understand what was needed from us starts the experiment. We started by creating the functions needed for the circuit to work. We had to create 5 functions. Building these functions, we just followed the instructions that were given to us in the experiment and had a very smooth time building them. initLCD() and waitMillis()/waitMicros() functions were given to us so that made 2 functions we already had before even starting this part of the experiment. We then built the other three functions that were needed; triggerEnable(), sendChar(byte data), sendCMD(byte data) function. Building the triggerEnable() funtions was very easy since all we had to do assign EN to high and low consecutively using the waitMillis()/waitMicros() delay functions we already had. Then we built the sendChar(byte data), sendCMD(byte data) and functions. Both these functions actually worked in the same way were the only difference was the assigning of the RS input. Reason for building these functions was to process the data. Since these functions worked on 4-bits and the data was 8-bits, we had to implement a unique method with the usage of shifts to the data as 4-bits each time. This method would take the data and get the upper/lower 4-bits consecutively using shift operations. The only problem we had was actually in this part were without the shifting operations method we used, out CMD would display random characters which corresponded to different data then we had sent. Also we had to use the triggerEnable() functions which sent these data to the LCD. We were able to solve these with the shifting method. This briefly describes our journey building these functions. After these, we updated the setup part so we would have our names and surnames displayed on the LCD. We then tested the circuit and finished out design.

In part 4 of the experiment, we again used a LCD like in the previous part but this time we had some different things to do. This part had us use the timer interrupts like part 1 and 2, on the LCD. Initially speaking we thought of this part to be a lot harder than the other parts but just adding a timer interrupt on the LCD didn't prove to be so difficult

5 CONCLUSION

In this experiment, we learned the interrupt capabilities and usage of interrupt service routines on Arduino. We used the interrupt capabilities to build a real-time counter without using a delay. We implemented similar circuits using Assembly but we did not know how to use interrupts then so this was another interesting experiment where we played around with real time interactions with the program. The experiment itself was not very hard but it was partly because we were more used to concepts like seven-segment displaying techniques and the logic of the interrupts from the previous weeks.