ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E

MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO : 5

EXPERIMENT DATE : 13.12.2020

LAB SESSION : WEDNESDAY - 13.30

GROUP NO : 3

GROUP MEMBERS:

150170013 : ALİ KEREM YILDIZ

150170082 : ARDA CÜCE

150170908 : MUHAMMAD RIZA FAIRUZZABADI

FALL 2020-2021

Contents

1 INTRODUCTION [10 points]

In this experiment, we were tasked to program given circuits' design that aims to teach implementation of counter with buttons and 7 segment display. Via using BJT, we are going to display values on proper seven segment displays. After that we are going to make operations such count up and count down or reset according to button's state. We will learn implementation of interrupt subroutines for button presses.

2 MATERIALS AND METHODS [40 points]

2.1 MATERIALS

Tools Used[1]

- Autodesk Tinkercad
- Latex (overleaf.com)
- Arduino Uno

This experiment is done via Autodesk Tinkercad Design Tool as well as the previous experiments. We used the 'Microcomputer Lab Intro' Lecture Notes as a reference regarding the overall structure of the board, alongside 'Segment Digit LED Display' User Manual and sources regarding the new concept implemented in the experiment is interrupt subroutines.

2.2 METHODS

2.3 PART 1

In this part of experiment, we will implement a simple two-digit decimal timer using two-segment display. We should display tens digit on first seven segment display, and ones digit on second seven segment. According to given design, first, via using DDR registers, pins that connected to seven segments, declared as output. Since we should implement interrupt subroutine for button press, thanks to attachInterrupt() function, according to pin number we can call reset or switchmode functions by RISING.(because it should run when button is pressed.) We use millis() function to display numbers on seven segment in real time. To achieve that, we declare a "second" variable and assign it to 1000 (1 second = 1000ms) and with using a condition, every passed second according to button's state we are going to start decrement or increment operation. We have some corner case like in 99 number displayed on while increment operation, and 0 number displayed while decrement operation. Since this mechanism must be cyclic, we consider that cases and programmed according to that. In brief, every second (via using millis()) we increment a value and display it. We display number's tens digit to first seven segment and ones digit to other seven segment. If we want to reset the counter, we click first button, since first button is RISING we read and call reset function via attachInterrupt() function. When second button is RISING we switch the mode of count operation. Some display examples given below.

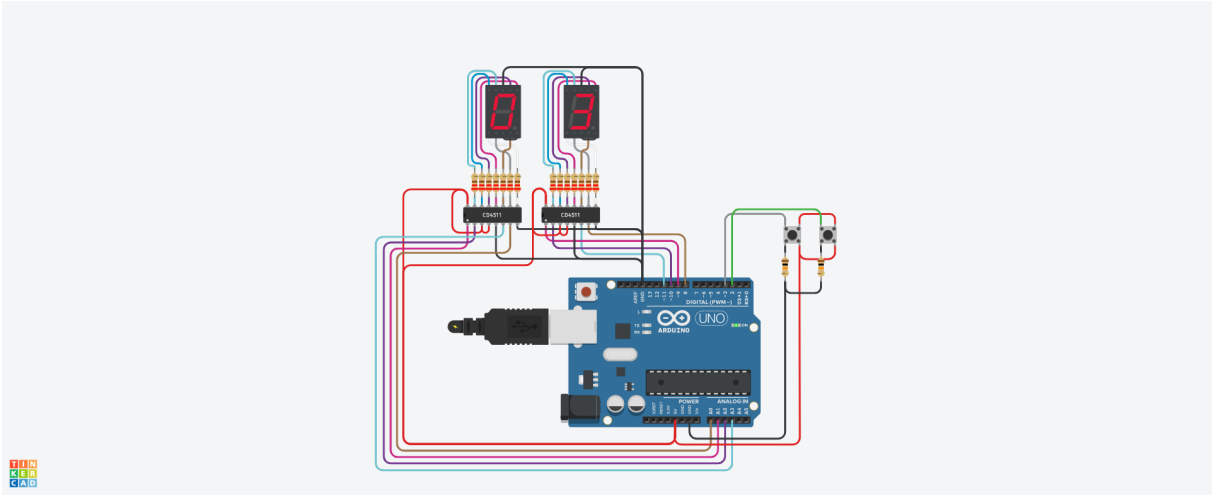

Figure 1: Default mode(count up)

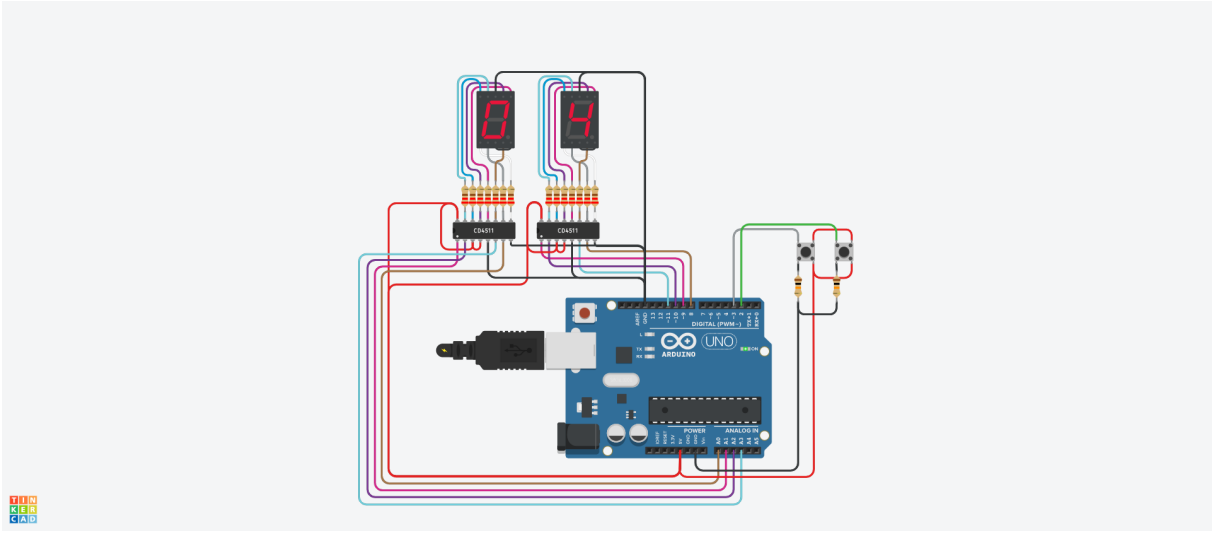

Figure 2: Default mode (count up) and previously 3, now 4

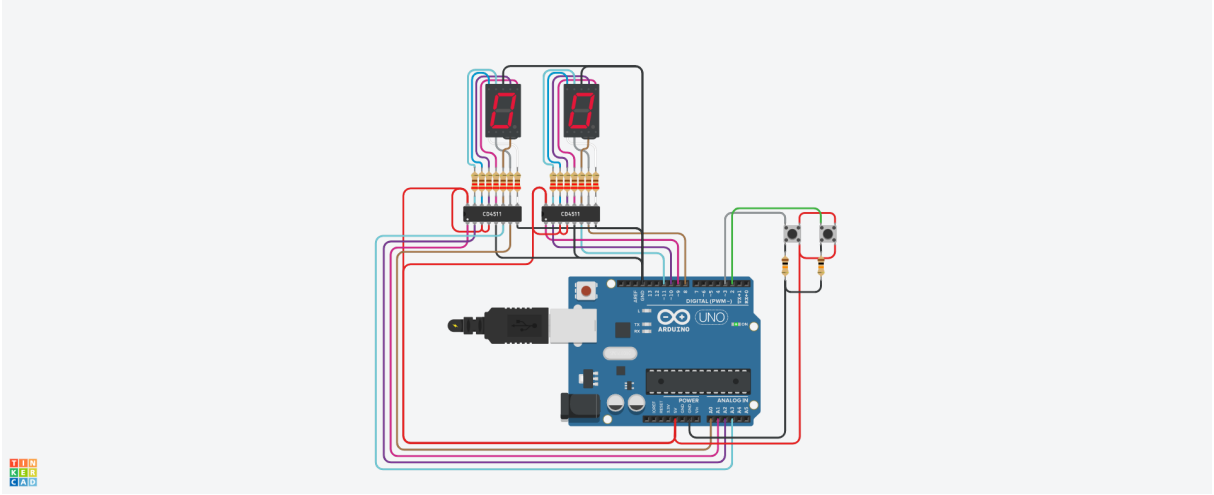

Figure 3: First button clicked. (Reset)

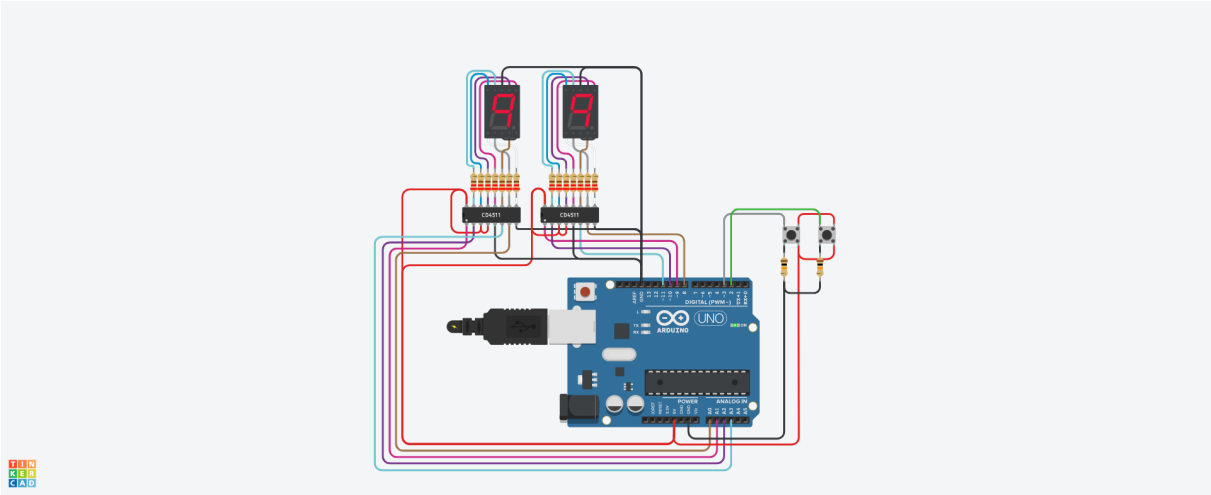

Figure 4: Second button was clicked (counts down) previously 0, now 99

2.4 PART 2

In this part of experiment, we were tasked to use 4 7-segment displays to make a number game. User is to guess a target number(which is predetermined) by inputting digits with a push button. Firstly, it is worth mentioning that the information given in the MicroLab5.pdf document weren't sufficiently clear enough at first. For instance, we predetermined the target values because it was seemingly the most logical thing to do, but we are not sure that if it is actually mentioned in the document to do so. Firstly, we declared 5 integer variables, one as an array to store the values to display, one for the target value, one to select which display to turn on, and last two is for timing. Since the 7-Segment Displays are connected to different ports, we couldn't use one array as is in the previous to store necessary bytes to show 0-F, so we divided them accordingly to 2 bytes, which unfortunately were cumbersome and we think it is absolutely unnecessary, since there were still ports to use. Other than setup and loop functions, we created two other functions. One is for interrupt subroutine, that is triggered when the pin goes from LOW to HIGH (or for this case, when button is pressed). When it is called, it selects and slides display values to the right. The next function we created was a display function, that displays any given value for any given digit. As before we set the values and the BJTs with PORT, since DigitalWrite functions were again prohibited. In the setup part we set the pins as INPUT or OUTPUT accordingly, and here is also where we use AttachInterrupt() function for the select function we mentioned above, solving the timing problems we may have had when buttons are used. Eventually inside the loop() function, we first convert the values taken in the array to integer, to later compare with the chosen value. If it is not equal, 1st display value is incrementing every 1 second (which value will be taken with a button for guessing purposes). If it is equal, the program is halted.

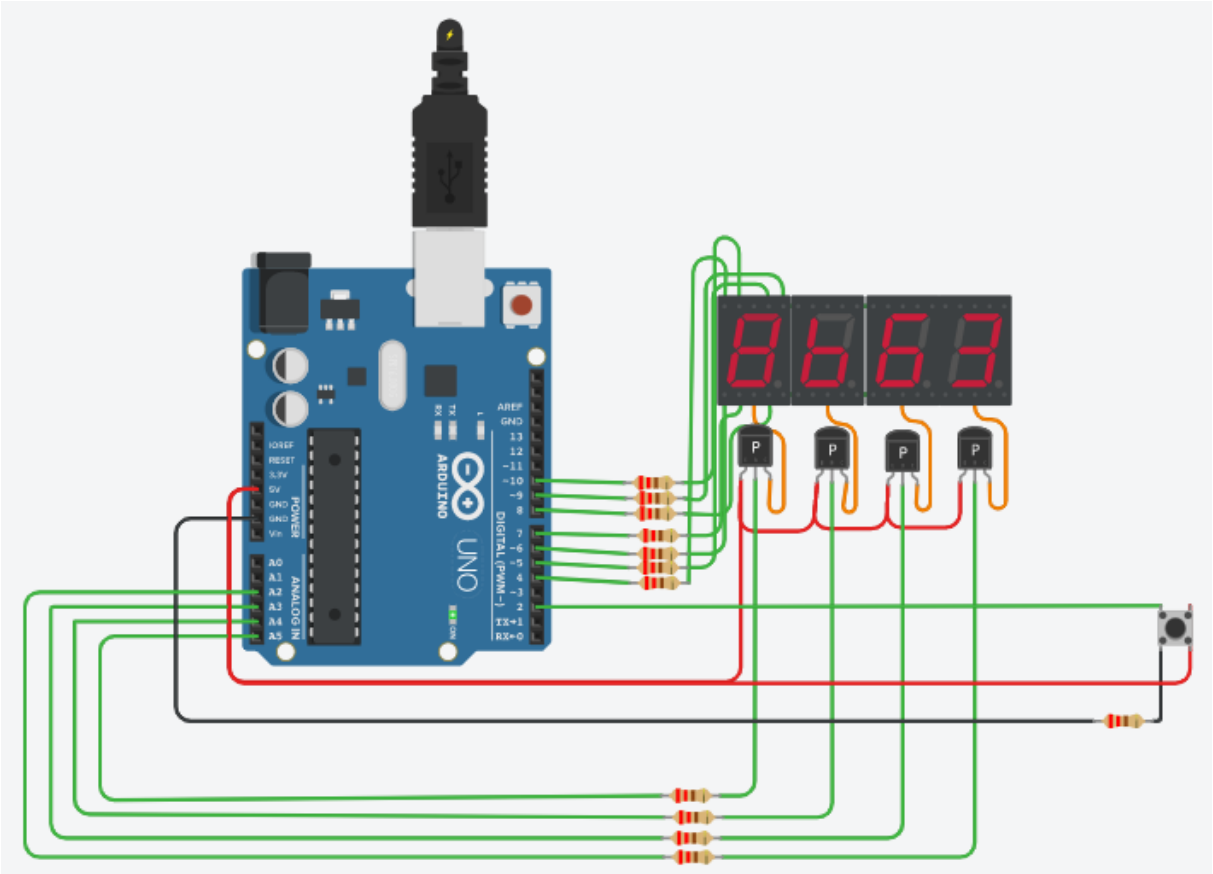

Figure 5: The condition where guessed value is false, 1st display keeps incrementing.

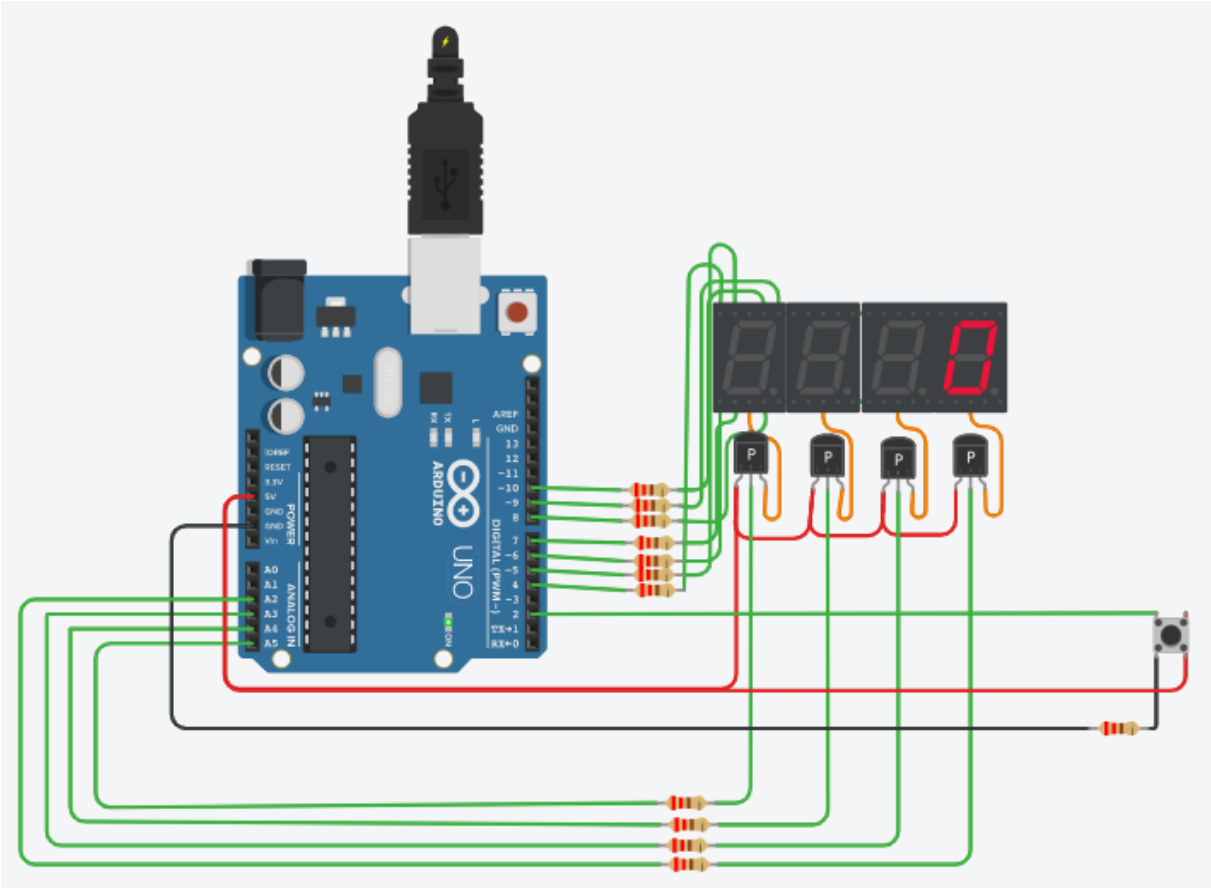

Figure 6: The condition where guessed value is true, program halts.

2.5 PART 3

In this part of the experiment, the design of the board is actually similar with Part 2. With the only difference being an addition of one other pushbutton. The program implemented is also slightly different. For instance instead of one, all displays are incrementing (separately), enables the users to choose numbers from 1 to 20. We declared same variables as in the previous part with addition of a couple of integers to distinguish players' turn, and some negative values to disable instant clicking that may lead to unfair play. Arrays of bytes for the 7-segment display are again divided into two. Having two pushbuttons this time, we have one addition of function, alongside another attachInterrupt() function inside the setup(). The two select functions are created to update player value and pass the turn. Part 2's display value is too reused here. There is not addition to the setup() function other than attachInterrupt() function which is already mentioned above, triggering the functions when the specific buttons are pressed, or the pin is set from LOW to HIGH. In the loop() function, first it is seen if the players value are equal or no, since otherwise it would mean one player's already winning. Inside this, two left or right display values are incremented, each from 0 to 9, to not exceed the limit of 20.

Same millis algorithm is used to switch the digits display rapidly to give the illusion of all of them being on in the same time. Lastly, if at first the players value are equal, player 2 wins if turn passed to 1, and player 1 wins if turn passes to 2. Then, the display values are changing to 1111 or 2222 according to the winner.

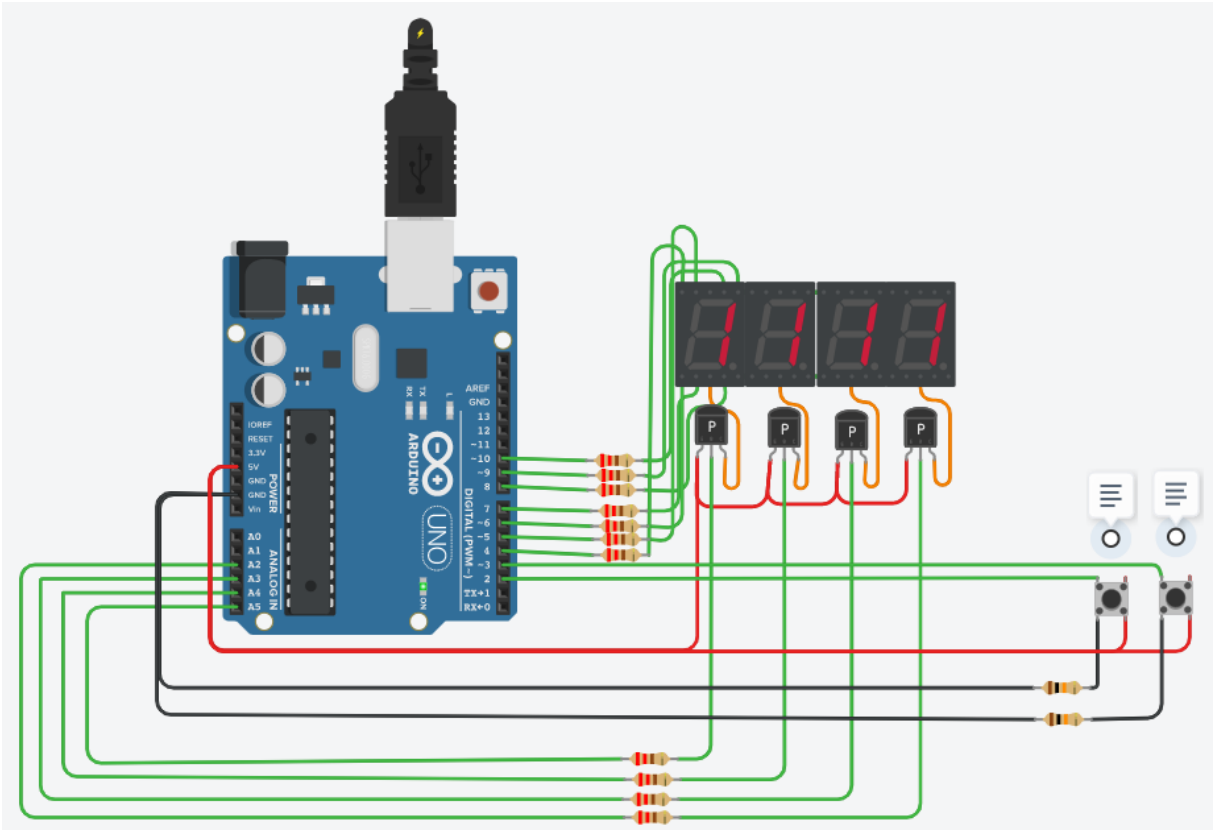

Figure 7: The display shows 1111, indicating that the player 1 has won.

3 RESULTS [15 points]

All of the three tasks given are accomplished, the functions and parts we used and implemented are not new, but in this experiment we got to tinker more complicated things with them. We managed to comprehend the concepts and implement them using the Arduino references we've already been using since the previous experiments. For the second time we got to tinker with pushbuttons. Further experimenting with 7-Segment Displays gave us better understanding on them.

4 DISCUSSION [25 points]

In this experiment, we had another opportunity to conduct more complicated experiments with 7-Segment Display like in the previous part, giving us a better understanding

on its concepts. Moreover, instead of taking input from potentiometer, we use them to display numbers which values are set with the buttons. Lastly, in accomplishing those tasks, we got to use and learn the concept of interrupt, solving problems we have with push-buttons before implementing it.

5 CONCLUSION [10 points]

Actually, we encountered problems while doing the experiment. When we want to display values on the 4 of the seven segment at the same time. Since they connected to same pins of Arduino, it is challenging for us. Via using millis() function observation of values on seven segments became more clearer. We learn interrupt subroutines and advantages of millis() function compare to delay() function.

REFERENCES

[1] Istanbul Technical University Department of Computer Engineering. Blg 351e microcomputer laboratory experiments booklet, Fall 2020-2021.