ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E

MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO : 1

EXPERIMENT DATE : 20.11.2020

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 made a brief introduction to Arduino UNO using Tinkercad which is an online simulation environment to form circuitry and test code. The circuit given to us by the assistants had 8 LEDs connected to digital outputs to represent the data inside the register. We have gone through Arduino's tutorials to manipulate necessary register that is located inside Arduino in order to perform port output manipulation.

2 MATERIALS AND METHODS

- Tinkercad
- Arduino Uno R3
- 8 Resistors
- 8 LEDs

2.1 PART 1

In this part of the experiment, we are expected to implement a basic 8-bit timer that returns to 0 when it reaches 255. Initially, we thought about the concept intuitively and we determined our requirements. We have to define a variable that keeps current time and increase it by 1 at every 1 second.

While programming Arduino Micro-controllers, we have to follow some rules as other programming languages. It has setup function for initialization of Arduino components and loop function for repetitive operations.

In this experiment, we are expected to use digital 0-7 ports as an output. In Arduino, some registers identify which pin is output or input. In the setup phase, the programmer should decide characters of pins and define them. Port D Data Direction Register (DDRD) is responsible for types of digital 0-7 ports. Therefore, in setup phase, we assigned B11111111 to DDRD for setting them as output.

For the loop part, we have done simple programming. We defined a one-byte variable and set it to 0. In each iteration of the loop, we have incremented our variable with 1. To display our data in our digital ports, we have used the PORTD register that is

responsible for the state of the ports. We used delay command to observe pin transitions better and count the seconds.

```
byte data = B00000000; //Initialization of timer

void setup()
{
  DDRD = B11111111; //Set ports as output
}

void loop()
{
  data = data + B00000001; //Increment by 1
  delay(1000); //Wait 1 second
  PORTD = data; //Show timer in ports
}
```

2.2 PART 2

In this part, we are expected to implement a sequence that is similar to walking lights. If we observe the pattern, it is binary shift operation or we can define as multiplication and division with 2. Then, we have defined digital 0-7 ports as output using DDRD. In this part, we did not use additional variable and we kept our value directly in the PORTD register. We have initialized PORTD with 1 in binary format.

To create the pattern, we have to left shift our value until it reaches to end, so in decimal 127 and at this point, we should start performing right shift. In Arduino, there is a "for" condition that is same as "for" in C programming language. We have used two separate "for" to perform right and left shift operations. Also, shift operations are the same as in C. We can use them by "$\ll$" and "$\gg$" symbols. Moreover, we have added delays to provide easier observation.

```
void setup()
{
  DDRD = B11111111; //Set ports as output
  PORTD = B00000001; //Initialization of PORTD register
}


```

```
void loop()
{
  for(int i = 0; i < 7; i++) //Loop for left shift
  {
    delay(500); //Wait 0.5 second
    PORTD <<= 1; //It performs binary left shift operation
  }
  for(int i = 0; i < 7; i++) //Loop for right shift
  {
    delay(500); //Wait 0.5 second
    PORTD >>= 1; //It performs binary right shift operation
  }
}
```

2.3 PART 3

In the third part of the experiment, we are expected to implement a led pattern that shows the first letters of our group member's names. Our group consist of Burak, Oben and Başar. Therefore, we have to write "B-O-B" to the digital outputs. In Arduino, we can directly assign characters to the PORT variables. It converts characters into binary form (ASCII) and stores them. That is why, we have assigned PORTD to in order of B, O and B. Since there are two B's, we have also used 0 to prevent confusion because of the two B's.

```
void setup()
{
  DDRD = B11111111; //set ports as output
}

void loop()
{
  PORTD = 0; //Assignes 0 to PORTD register
  delay(1000); //Wait 1 second
  PORTD = 'B'; //Assignes B to PORTD register
  delay(1000); //Wait 1 second
  PORTD = 'O'; //Assignes O to PORTD register
  delay(1000); //Wait 1 second
  PORTD = 'B'; //Assignes B to PORTD register
  delay(1000); //Wait 1 second
}
```

3 RESULTS

3.1 PART 1

In this part the most important point is the cyclic behaviour of the timer. After the simulation, we have observed that we have met this edge case. Simulation was as we expected.

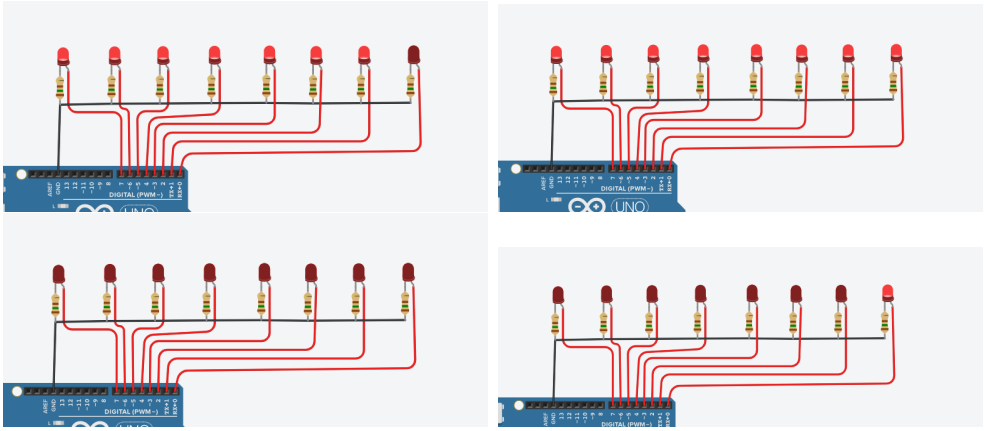

Figure 1: Simulation result and transition of LEDs from 255 to 0

3.2 PART 2

As seen in the simulation results below, the shift operation is done with the desired behaviour. It returns from the edges of the interval.

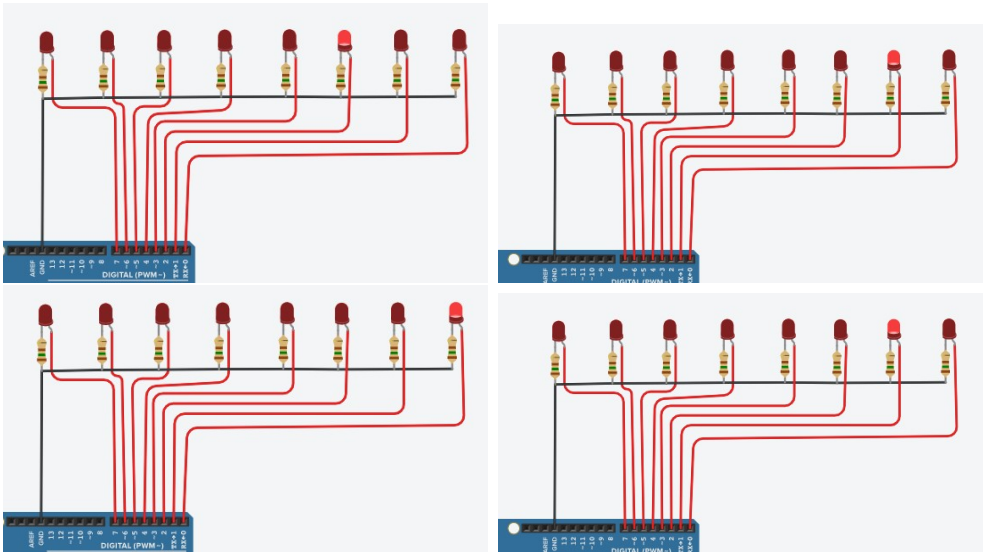

Figure 2: Representation and behaviour of the right shift

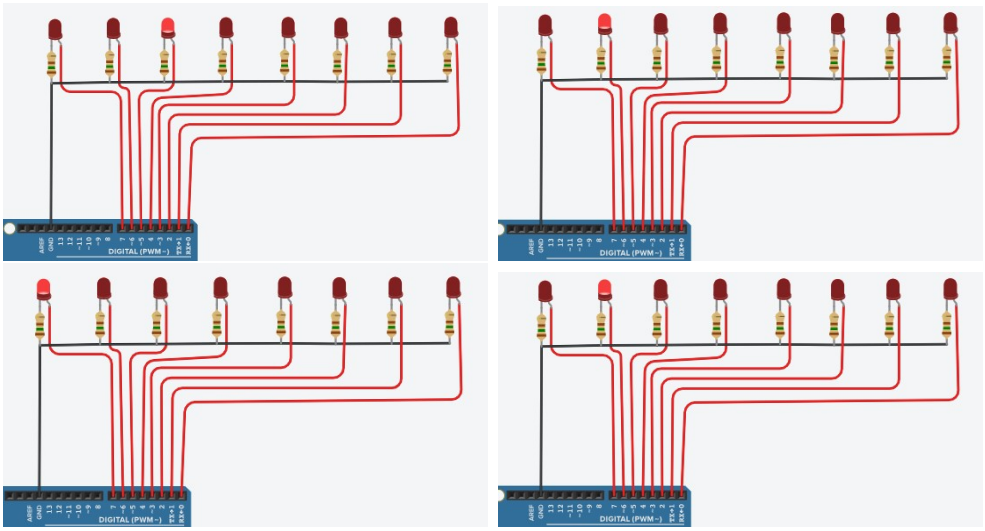

Figure 3: Representation and behaviour of the left shift

3.3 PART 3

We have simulated our code to show the first letters of our names. The result is the same as expected. LEDs show B-O-B in ASCII form as $B01000010-$B01001111-$B01000010 (66-79-66).

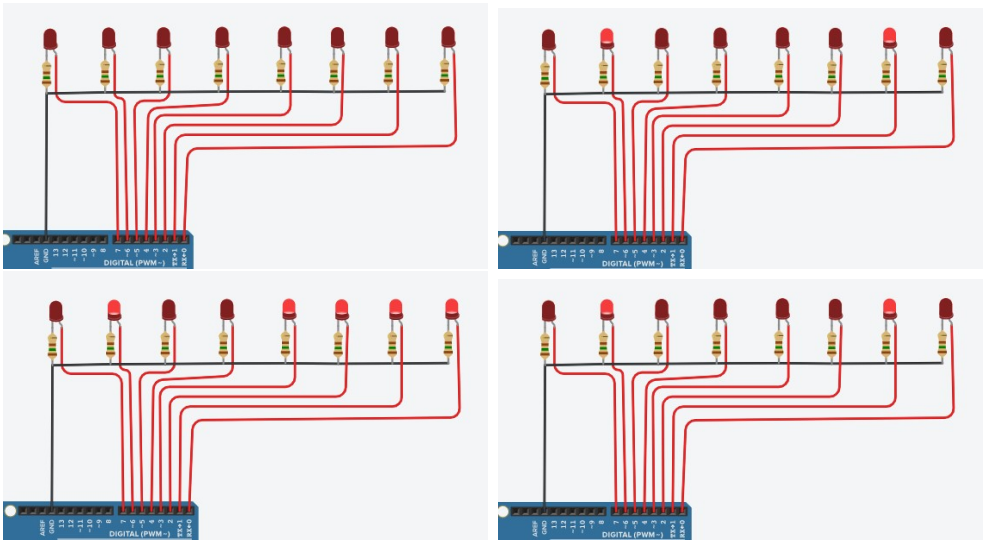

Figure 4: Representation of our first letters B-O-B

4 DISCUSSION

4.1 PART 1

In this part of the experiment, we have implemented an 8-bit timer with LEDs that counts up to 255 and reset itself to 0 when it reaches 255. We did not have any knowledge about Arduino also it's programming language. However, after reading documentation we have learned that Arduino has registers and those registers are connected to our pins. Therefore, we used these pins to control our LEDs and create our timer. At first, we implemented our timer with an if statement that manually resets our registers. After some research, we found out that our register has just 8 bits and after B11111111 it reset itself to 0.

4.2 PART 2

In this part of the experiment, we have implemented a sequence that like a led light walks across the other LEDs. At first, we thought that we could achieve this behaviour with an array of booleans that have only one 1 value. We tried to shift the values inside the array using array manipulations however we have realized that this approach was not efficient and flexible. After some research, we found out Arduino's language has shift operators $\ll$ and $\gg$ similar to other programming languages. Therefore, we could easily shift our value (1) right and left with two individual for loops.

4.3 PART 3

In this part of the experiment, we have implemented a pattern that displays the first letters of our group members' names. The letter is represented by it's ASCII value in binary form. We have discussed how to assign letters "B-O-B" and then we decided to try to assign a char value directly to the register as chars also represent integer values. Our code worked and we were very satisfied with the result. Since the C-like language of Arduino allows us to perform high-level programming language operations, we did not think about converting our first letters to binary form and language did that casting for us which made our code simpler.

5 CONCLUSION

In conclusion, we have looked at the documentation to learn how to use Tinkercad Arduino simulation and how to manipulate PORT registers with Arduino language. We found out Arduino's language is similar to C language and we easily become aware of language's features. We came to know about setup and loop functions' features step by step. We found out some different solutions for each part of the experiment therefore we learned different solutions possible in this experiment. We think this experiment is good for us to start programming Arduino and we learned a lot from it.

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.