ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	2
EXPERIMENT DATE	:	20.11.2020
LAB SESSION	:	MONDAY - 13.30
GROUP NO	:	G10

GROUP MEMBERS:

150180103	:	EGEMEN GÜLSERLİLER
150180028	:	ABDÜLKADİR PAZAR
150180716	:	AHMET SELÇUK TUNÇER

AUTUMN 2020

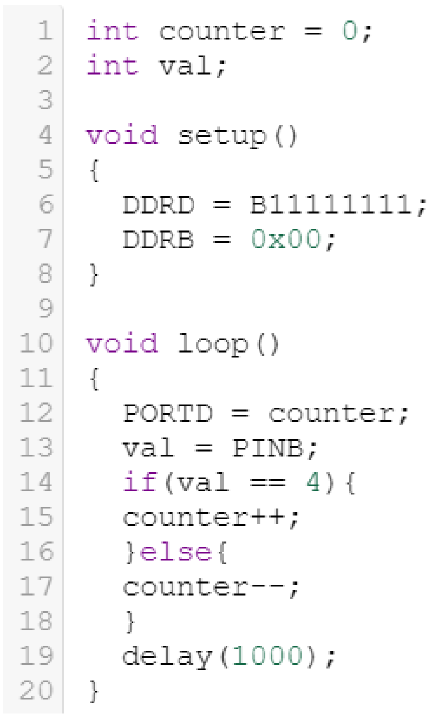
Contents

1 INTRODUCTION

In this experiment, we learned how to take inputs and interpret them in order to create various patterns with the Arduino Uno board, 2 buttons, 2 switches and 8 LED pins with Arduino programming language.

2 MATERIALS AND METHODS

2.1 2.1 Part 1: 8-bit Timer with Buttons



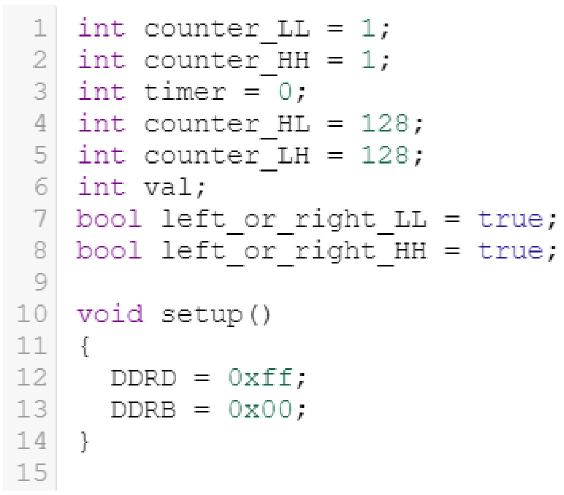
```
int counter = 0;
int val;

void setup()
{
  DDRD = B11111111;
  DDRB = 0x00;
}

void loop()
{
  PORTD = counter;
  val = PINB;
  if(val == 4){
  counter++;
  }else{
  counter--;
  }
  delay(1000);
}
```

Figure 1: Code to create 8-bit timer with a button

In this part we used the code above to create an 8-bit timer.We set DDRD register as output and DDRB register as input. It counts up if no button is pressed and counts down if any button is pressed.

2.2 2.2 Part 2: Generating Given Pattern

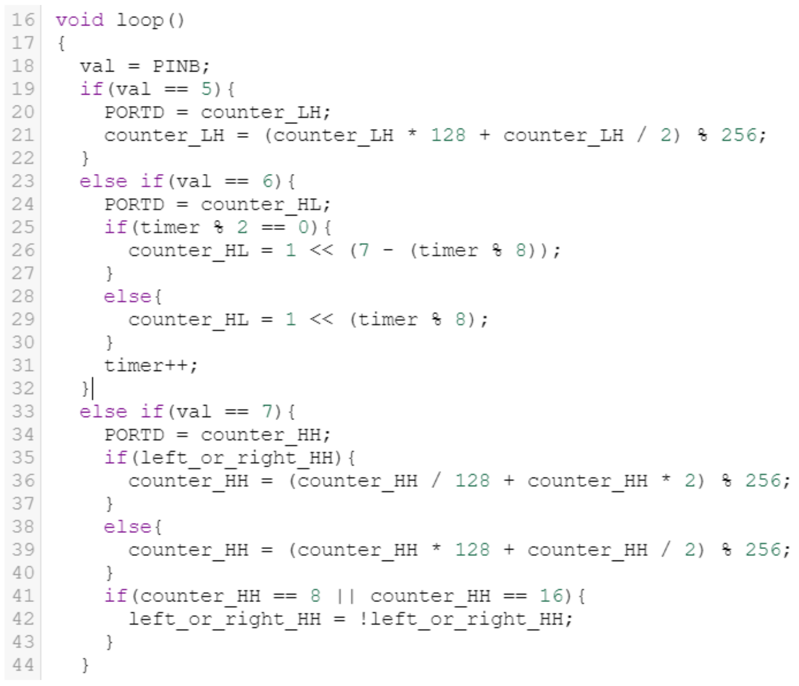
In this part we are given 2 switches and 8 led pin and we are asked to create 4 different pattern.



```
int counter_LL = 1;
int counter_HH = 1;
int timer = 0;
int counter_HL = 128;
int counter_LH = 128;
int val;
bool left_or_right_LL = true;
bool left_or_right_HH = true;

void setup()
{
  DDRD = 0xff;
  DDRB = 0x00;
}

```

Figure 2: Global variables and setup for switching patterns

At first, we defined 4 integer counter variable for each pattern, integer ”timer” for LH switch, integer ”val” for PINB as input and shifting boolean variables for periodic counts. In setup part we set DDRD register as output and DDRB register as input.



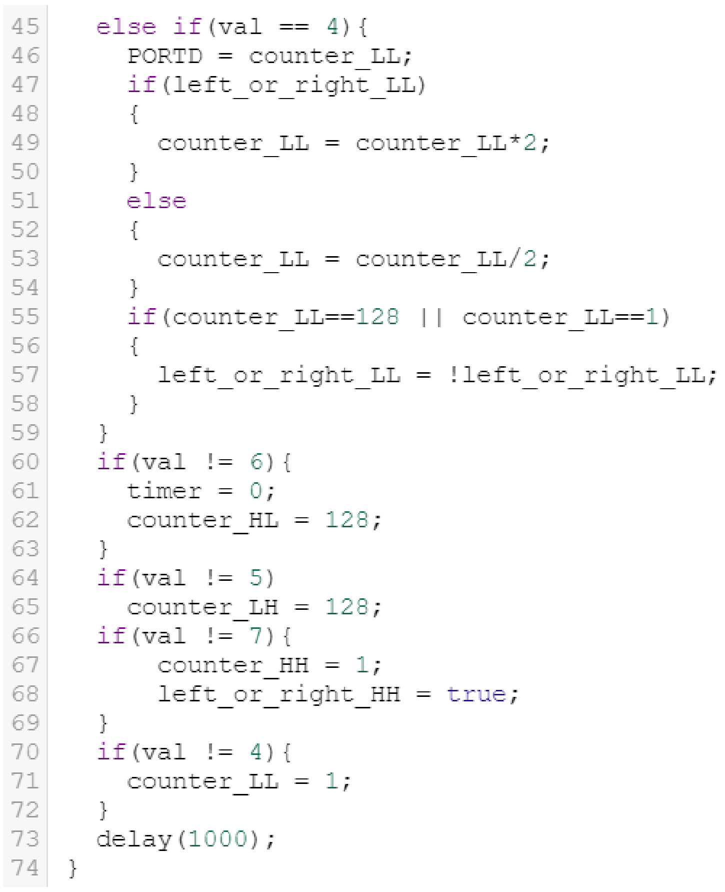
```
void loop()
{
  val = PINB;
  if(val == 5){
    PORTD = counter_LH;
    counter_LH = (counter_LH * 128 + counter_LH / 2) % 256;
  }
  else if(val == 6){
    PORTD = counter_HL;
    if(timer % 2 == 0){
      counter_HL = 1 << (7 - (timer % 8));
    }
    else{
      counter_HL = 1 << (timer % 8);
    }
    timer++;
  }
  else if(val == 7){
    PORTD = counter_HH;
    if(left_or_right_HH){
      counter_HH = (counter_HH / 128 + counter_HH * 2) % 256;
    }
    else{
      counter_HH = (counter_HH * 128 + counter_HH / 2) % 256;
    }
    if(counter_HH == 8 || counter_HH == 16){
      left_or_right_HH = !left_or_right_HH;
    }
  }
```

Figure 3: LH ,HL and HH patterns

In loop, we assigned PINB switch input to variable ”val”. After that we created our switching patterns by checking ”var”. In LH switch if ”var” equals to 5 , counter started from 128(B10000000) and divide by 2 to shift the 1 to the right by 1 bit each step. When count reach 1(B00000001) it turn back to 128(B10000000) thanks to modular 256.

In HL switch if ”var” equals to 6, HL counter started from 128(B10000000) and we used timer variable for shifting. If timer is an even number, we shifted 1(B00000001) left by 7 - (timer modular 8). If timer is an odd number we shifted 1(B00000001) left by timer modular 8. In each step we increment timer by 1.

In HH switch if ”var” equals to 7, counter started from 1(B00000001) and multiply by 2 to shift the 1 to the right by 1 bit. When count reaches 8(B00001000), it reversed due to if-else check and it starts to divide by 2 to shift by 1 to the left by 1 bit. When count reaches 1(B00000001) it becomes 128(B10000000) thanks to modular 256 and started to oscillate reverse side and so on.



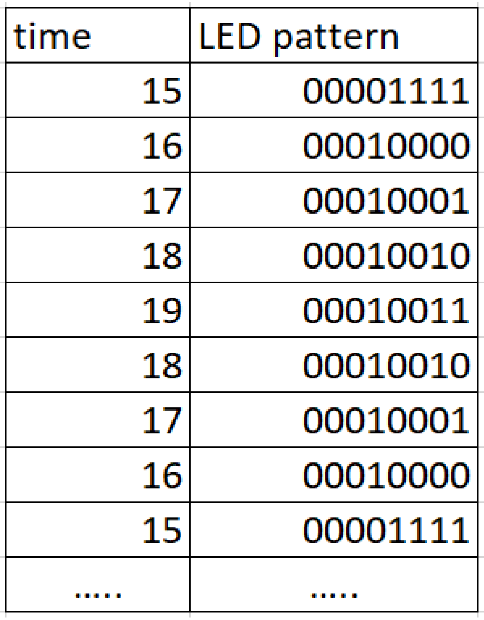
```
  else if(val == 4){
    PORTD = counter_LL;
    if(left_or_right_LL)
    {
      counter_LL = counter_LL*2;
    }
    else
    {
      counter_LL = counter_LL/2;
    }
    if(counter_LL==128 || counter_LL==1)
    {
      left_or_right_LL = !left_or_right_LL;
    }
  }
  if(val != 6){
    timer = 0;
    counter_HL = 128;
  }
  if(val != 5)
    counter_LH = 128;
  if(val != 7){
      counter_HH = 1;
      left_or_right_HH = true;
  }
  if(val != 4){
    counter_LL = 1;
  }
  delay(1000);
}
```

Figure 4: LL pattern and resetting counters

In LL switch if ”var” equals to 4, we created the requested pattern by starting from 1 and multiplying by 2 to shift the 1 to the left by 1 bit. After reaching 128 (B10000000) we divided by 2 to shift the 1 to the right by 1 bit. At the bottom of loop we reset pattern variables that are not used in order to start from beginning.

3 RESULTS

3.1 Part 1 Results



time	LED pattern
15	00001111
16	00010000
17	00010001
18	00010010
19	00010011
18	00010010
17	00010001
16	00010000
15	00001111
.....	.....

Figure 5: Time table for LED patterns produced by 8-bit counter

3.2 Part 2 Results

time	led pattern
0	00000001
1	00000010
2	00000100
3	00001000
4	00010000
5	00100000
6	01000000
7	10000000
8	01000000
9	00100000
10	00010000
11	00001000
12	00000100
13	00000010
14	00000001
15	00000010
16	00000100
...	...

Figure 6: LL switching pattern

time	led pattern
0	10000000
1	01000000
2	00100000
3	00010000
4	00001000
5	00000100
6	00000010
7	00000001
8	10000000
9	01000000
10	00100000
11	00010000
12	00001000
13	00000100
14	00000010
15	00000001
16	10000000
...	...

Figure 7: LH switching pattern

time	led pattern
0	10000000
1	00000010
2	00100000
3	00001000
4	00001000
5	00100000
6	00000010
7	10000000
8	10000000
9	00000010
10	00100000
11	00001000
12	00001000
13	00100000
14	00000010
15	10000000
16	10000000
...	...

Figure 8: HL switching pattern

time	led pattern
0	00000001
1	00000010
2	00000100
3	00001000
4	00000100
5	00000010
6	00000001
7	10000000
8	01000000
9	00100000
10	00010000
11	00100000
12	01000000
13	10000000
14	00000001
15	00000010
16	00000100
...	...

Figure 9: HH switching pattern

4 DISCUSSION

For the first part we used variable "counter" as led counter and "var" as button. We shift right or left by 1 depending on the "var" value. We count up if no button is pressed and count down if any button is pressed. We use port manipulation and learn how to take input in Arduino.
For the second part we had 4 different switching pattern as a combination of 2 switch L or H. We defined counter variables for each one. We check switching pattern thanks to variable "var" that takes input from Port B. When switch positions changed during run time, our designs also changed the pattern accordingly and start over.

5 CONCLUSION

In this experiment we learned how to takes input and interpret them. First part was relatively difficult because we spent most of our time on figuring out how buttons work. Second part is way easier because for example LL switch is same as the previous experiment's part 3 and we already figured out to handle switch operation. However, we also spent time to understand how HL switch works.
To sum up, although we had a few difficulties we overcame and have learned new concepts, improved our knowledge on port manipulation in arduino programming.

REFERENCES