
GEBZE TECHNİCAL UNIVERTİSY

ELEC 335

# MİDTERM - 01

# REPORT

# RANDOMİZER COUNTER

BERKAY TÜRK

171024024

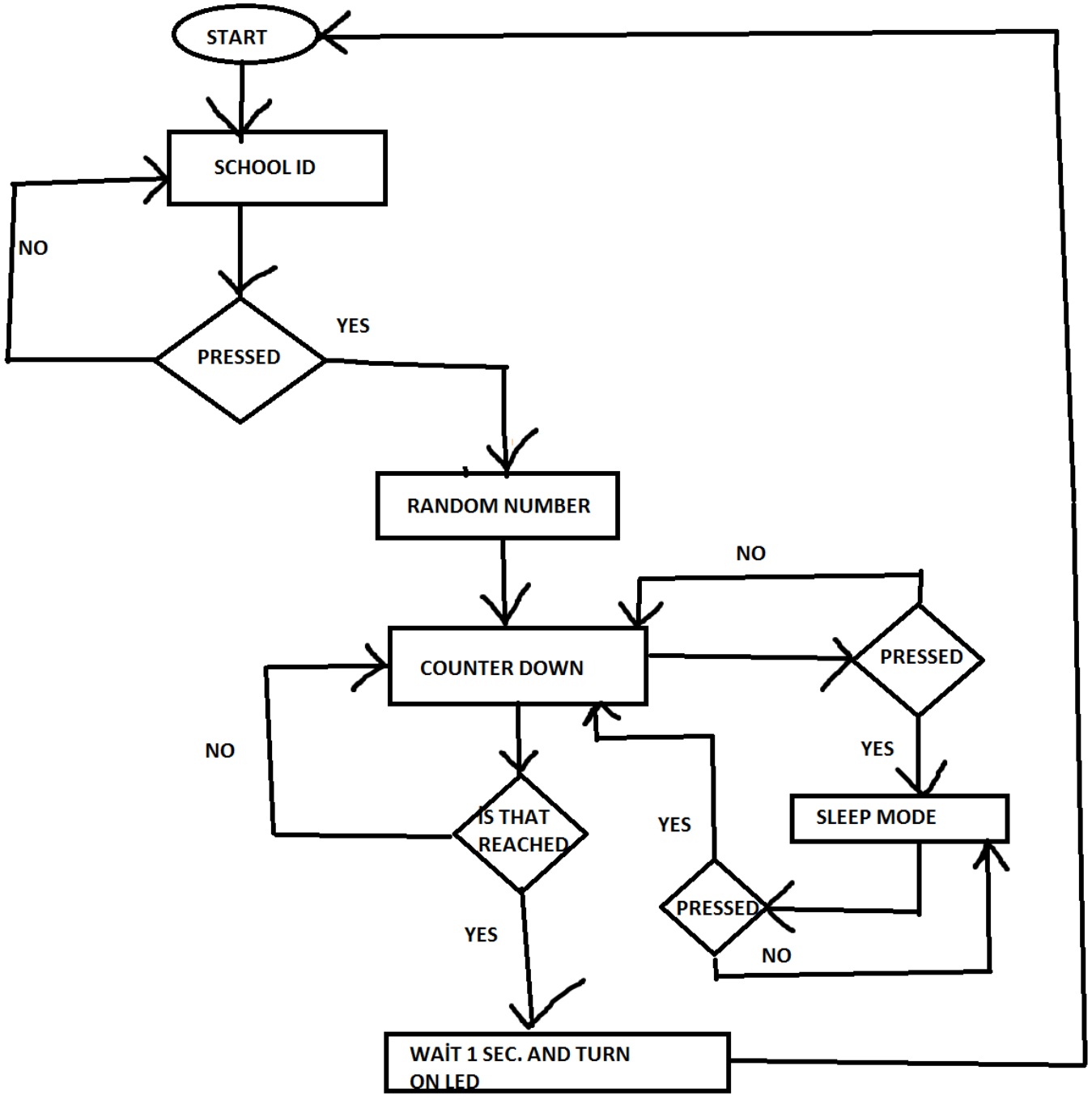
## INTRODUCTION:

Our aim in this Project is to create randomly between 1000 to 9999 and count down to zero.

## Detailed Requirements :

1) Implement a randomized counter in Assembly.
2) A 4-digit SSD should be connected that will display your ID (last 4 digits) when your code is not counting (idle state).
3) When an external button is pressed, it will generate a 4-digit random number, and start counting down to 0.
4) The generated random number should be between 1000 - 9999.
5) When the counter reaches 0, the number 0000 should be displayed for a second
6) Then the code should go back to idle state waiting for the next button press.
7) Pressing the button while counting down
8) Should pause counting and pressing again should resume counting.

I first created a flowchar in this direction and I dive into small task in flowchart. Finally I combine the tasks.




**Flowchart**

**TASK 1:**

Connect one 4xSSD to the board and turn on one part of a segment.My SSD is common katot .I make figure 1.

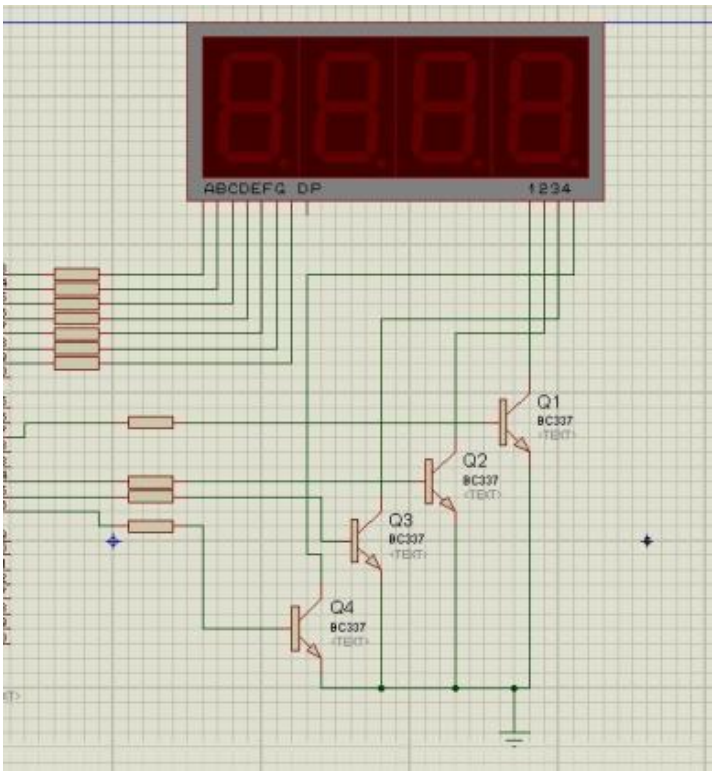



**Figure 1.**

**TASK 2:**

I know connect leds and button and I make figure 2.

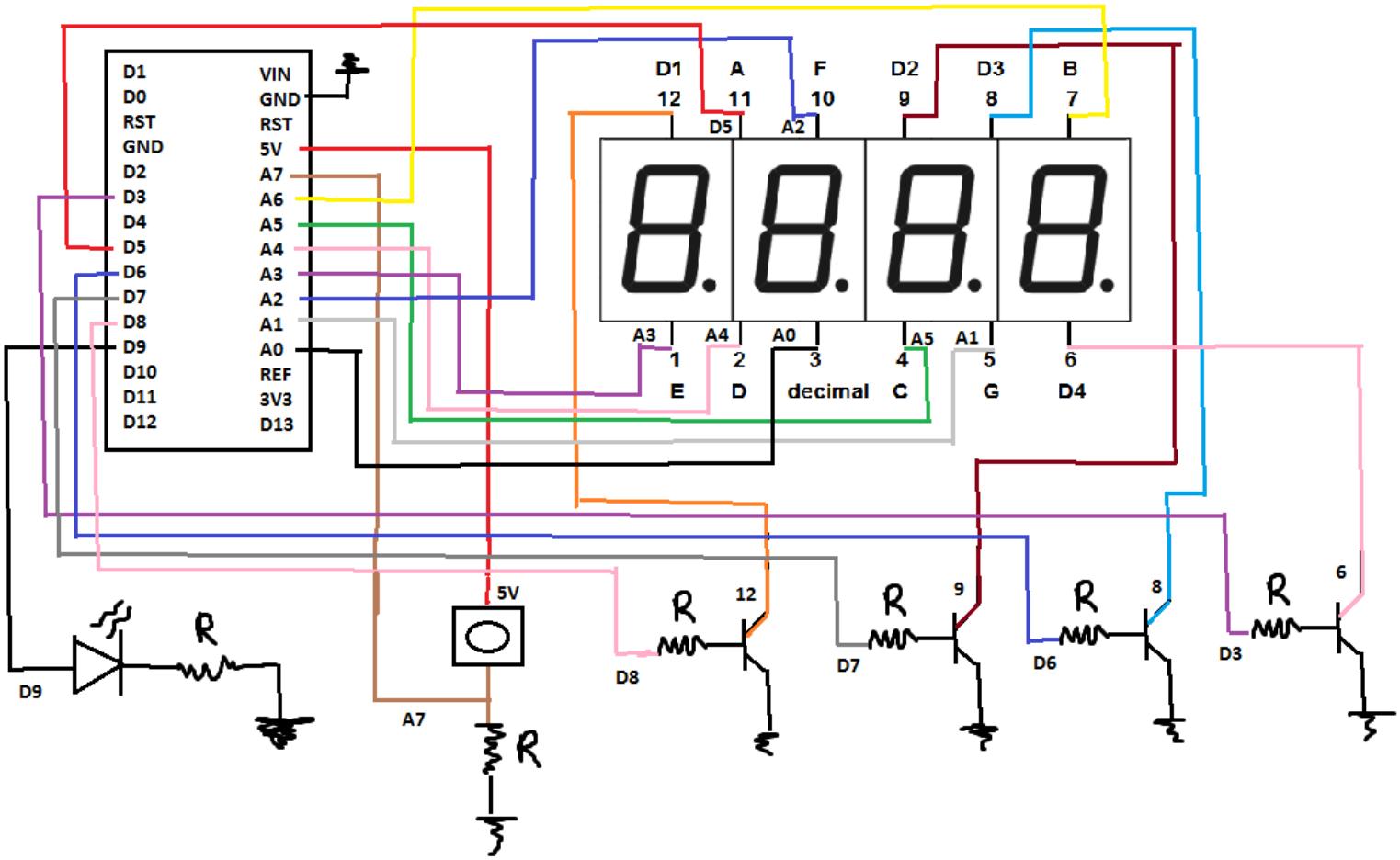



**Figure 2.Connection Diagram**

**TASK 3:**

My flowchart is too long and I divide small piece.I make figure 3. I thought how it will all burn at the same time and I fount It happens with a certain time waiting.I write simply delay function and I figured it out.

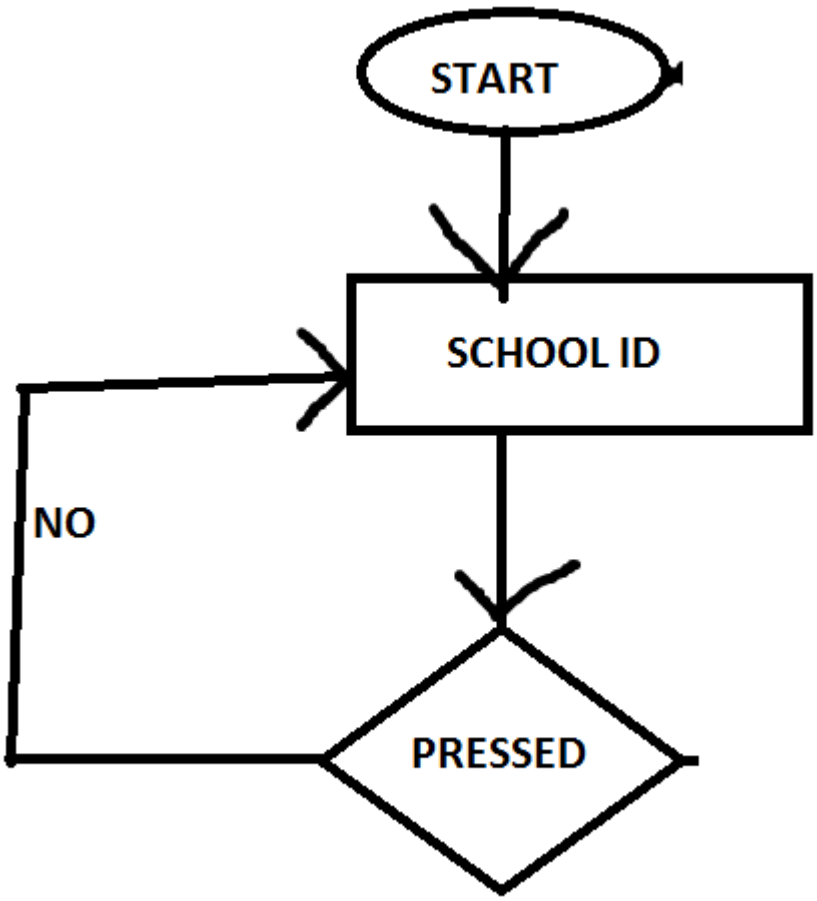



**Figure 3.**

**TASK 4:**

After I make figure 4 .I created function when pressed buton wait 1 second and back to School ID .

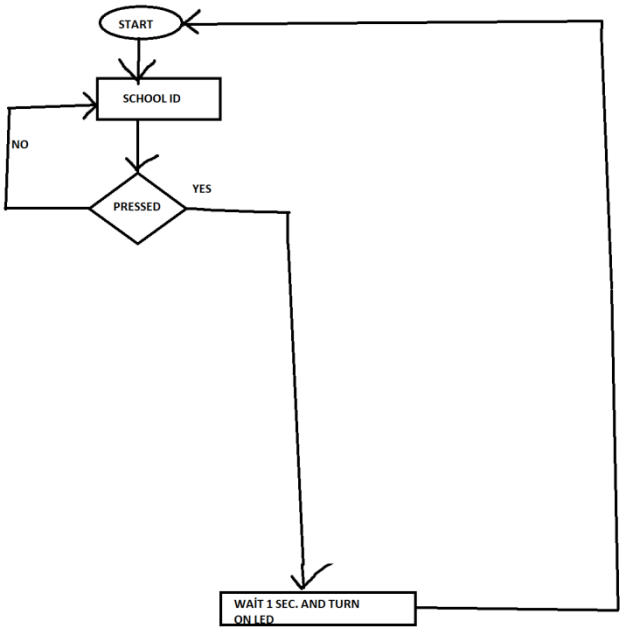



**Figure 4.**

**TASK 5:**

And  I work figure 5. But fail to come to a conclusion .There are a lot of Formula to random number genarated but I had no idea to random number and I didn't know how to show. Then I decided that I will continue as if it exist.

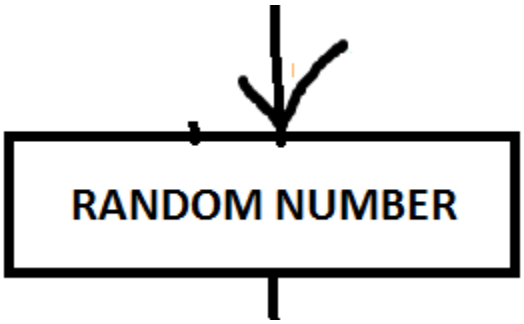



**Figure 5.**

**TASK 6:**

And same way I work figure 6. But fail to come to a conclusion .I work one digit count down and  I make but 4 digit number.I didn't know where the numbers will be kept . I created as an example so that it counts back from 4.

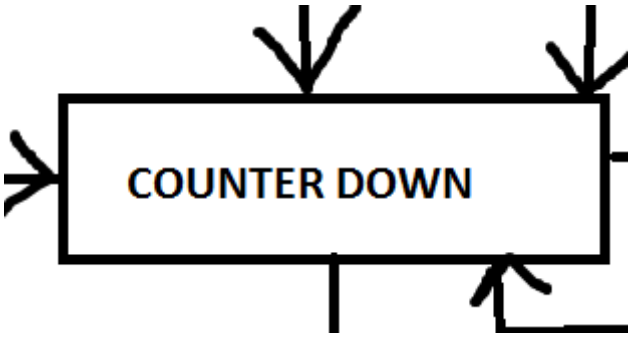



**Figure 6.**

**TASK 7:**

And same way I work figure 7. But fail to come to a conclusion. I continued this part by adding a button.

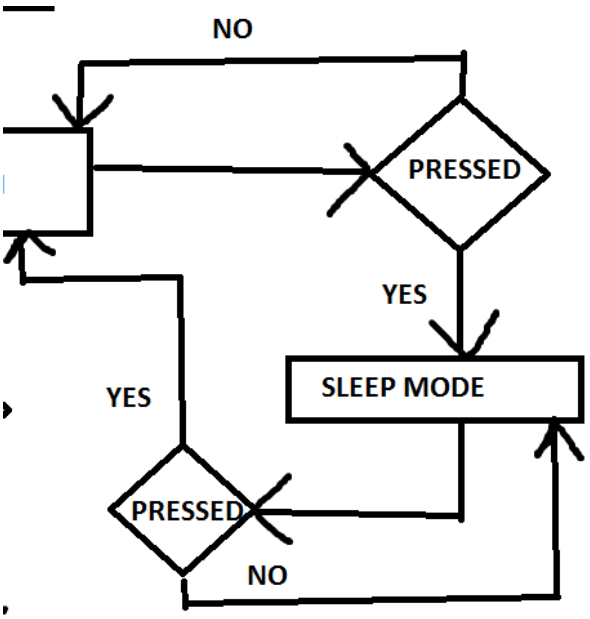



**Figure 7.**

**TASK 8:**

And same way I work figure 8. But fail to come to a conclusion . I continued this part by adding a button.

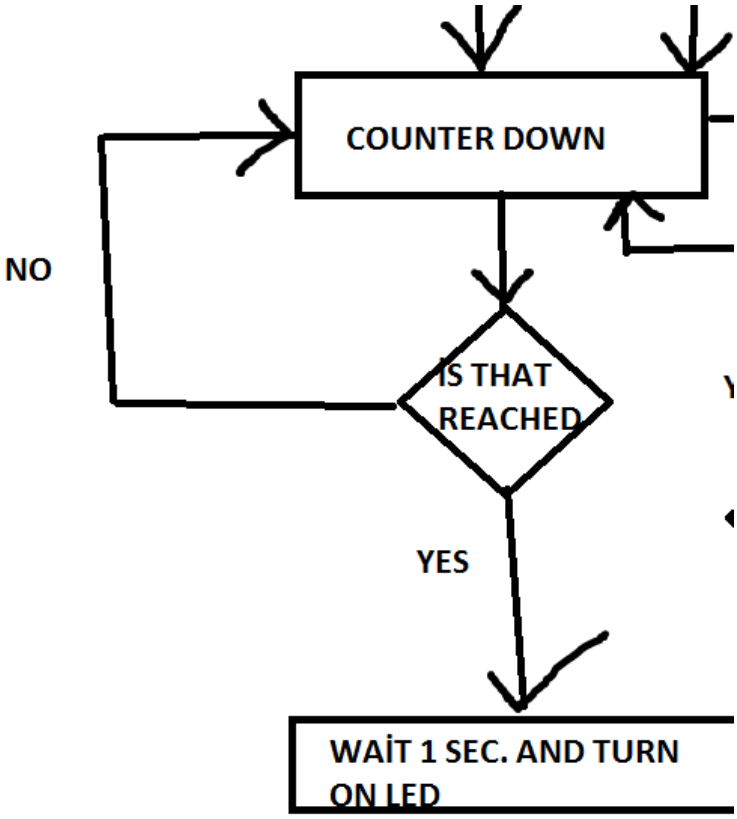



**Figure 8.**

**TASK 9:**

And same way I work figure 9. But fail to come to a conclusion and hard to collecting them all.

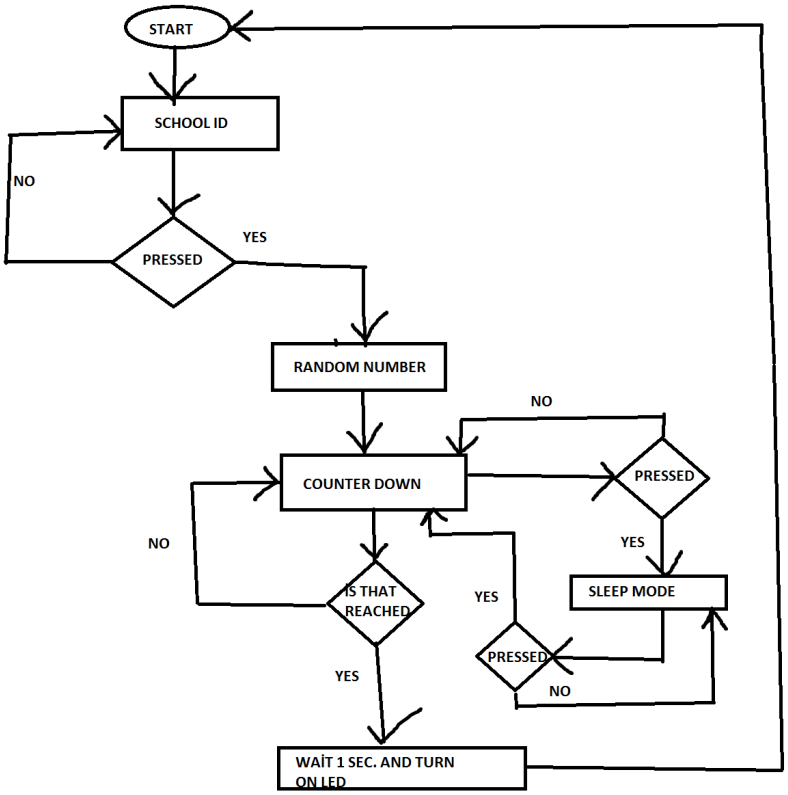
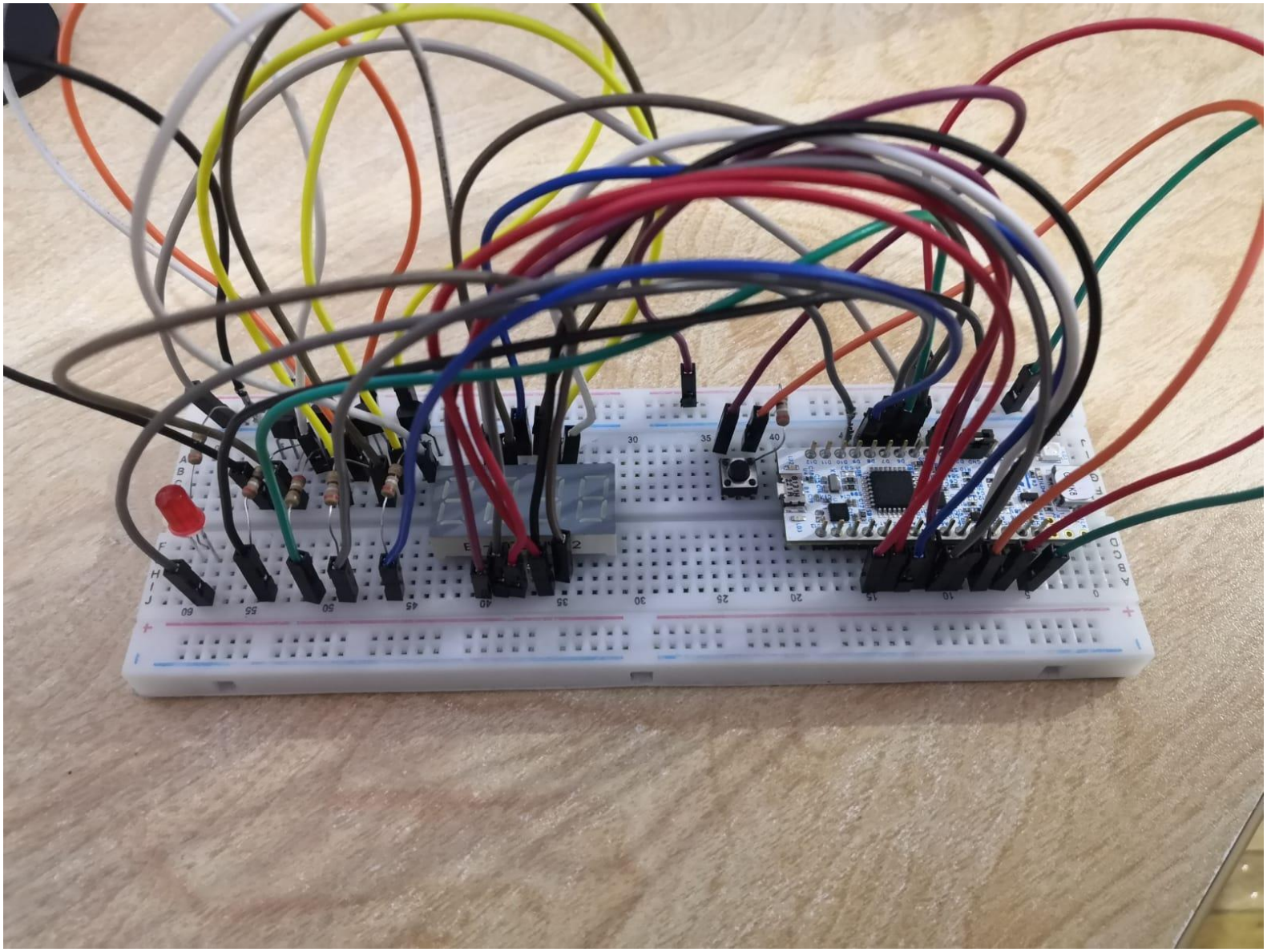
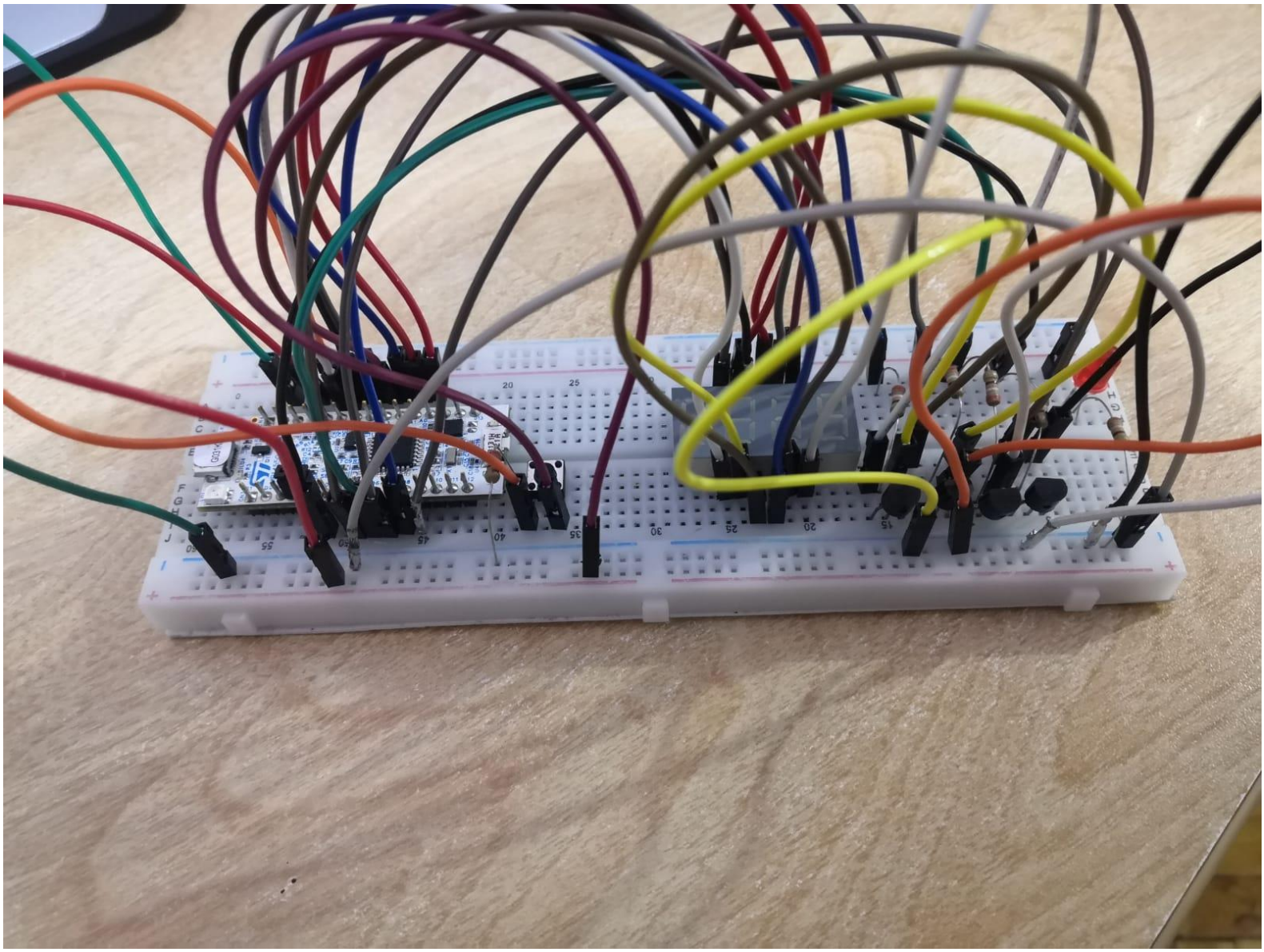



**Figure 9.**



**Front**



**Back**

## PART LİST:

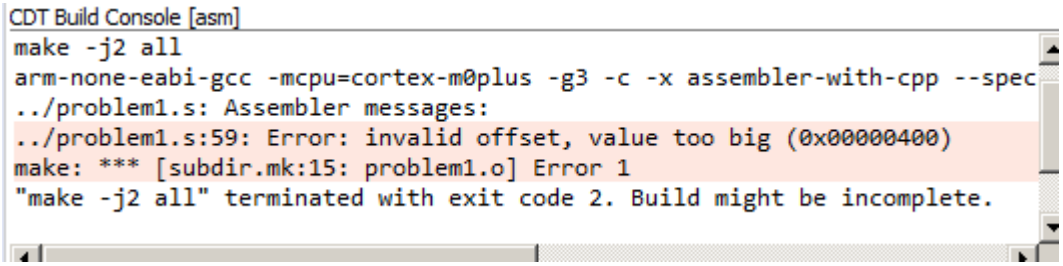
| | | |
|---|---|---|
| **NUCLEO-G031K8** | **X1** | **110TL** |
| **JUMPER CABLE** | **X25** | **5TL** |
| **RESİSTANCE 470Ω** | **X6** | **1TL** |
| **4XSEVEN SEGMENT** | **X1** | **7TL** |
| **TRANSİSTOR** | **X4** | **1TL** |
| **BUTTON** | **X1** | **0.25TL** |
| **LED** | **X1** | **0.25TL** |
| **SUM** | | **124.5TL** |

## CONCLUSİON:

As a result, I could not do some parts due to some lack of information but I learned from this project some things.For example turn on/off SSD , connected button ,connect led and some assembly language information (write function ,call function).

The project could not reach the desired place due to the difficulty in keeping value.

The biggest problem was getting the following errors. I could not write a proper code because I could not solve these errors.




## VİDEO LİNK:

https://youtu.be/YnybXfb_7Aw

## REFERANCES:

The_Definitive_Guide_to_ARM_CortexM0_M0+ *Second Edition* Joseph Yiu

RM0444 Reference manual

https://elektrokod.wordpress.com/2013/12/09/7-segment-display-sayici-uygulamasi/

**CODE:**

```
/*
 * proje1.s
 *
 * author: Berkay Türk 171024024
 *
 * description: 4 digit random number count 0
 *   G031K8 Nucleo board.
 */



.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb


/* make linker see this */
.global Reset_Handler

/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss


/* define peripheral addresses from RM0444 page 57, Tables 3-4*/
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR register offset

.equ GPIOA_BASE,       (0x50000000)          // GPIOA base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOA MODER register offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOA ODR register offset
.equ GPIOA_IDR,        (GPIOA_BASE + (0x10)) // GPIOA IDR register offset

.equ GPIOB_BASE,       (0x50000400)          // GPIOB base address
.equ GPIOB_MODER,      (GPIOB_BASE + (0x00)) // GPIOB MODER register offset
.equ GPIOB_ODR,        (GPIOB_BASE + (0x14)) // GPIOB ODR register ofset

.equ a, (0x0)
.equ b, (0x0)
.equ c, (0x0)
.equ d, (0x0)
/* vector table, +1 thumb mode */
.section .vectors
vector_table:
     .word _estack             /*     Stack pointer */
     .word Reset_Handler +1    /*     Reset handler */
     .word Default_Handler +1  /*       NMI handler */
     .word Default_Handler +1  /* HardFault handler */
     /* add rest of them here if needed */
```

```
/* reset handler */
.section .text
Reset_Handler:
      /* set stack pointer */
      ldr r0, =_estack
      mov sp, r0

      /* initialize data and bss
       * not necessary for rom only code
       * */
      bl init_data
      /* call main */
      bl main
      /* trap if returned */
      b .


/* initialize data and bss sections */
.section .text
init_data:

      /* copy rom to ram */
      ldr r0, =_sdata
      ldr r1, =_edata
      ldr r2, =_sidata
      movs r3, #0
      b LoopCopyDataInit

      CopyDataInit:
            ldr r4, [r2, r3]
            str r4, [r0, r3]
            adds r3, r3, #4

      LoopCopyDataInit:
            adds r4, r0, r3
            cmp r4, r1
            bcc CopyDataInit

      /* zero bss */
      ldr r2, =_sbss
      ldr r4, =_ebss
      movs r3, #0
      b LoopFillZerobss

      FillZerobss:
            str  r3, [r2]
            adds r2, r2, #4

      LoopFillZerobss:
            cmp r2, r4
            bcc FillZerobss

      bx lr


/* default handler */
.section .text
Default_Handler:
      b Default_Handler
```

```
/* main function */
.section .text
main:
    /* enable GPIOA, GPIOB clock, bit0 on IOPENR */
     ldr r6, =RCC_IOPENR
     ldr r5, [r6]
     /* movs expects imm8, so this should be fine */
     movs r4, 0x3
     orrs r5, r5, r4
     str r5, [r6]


    /* setup PB(0,1,2,8) for seven segment D4,D3,D2,D1 for in MODER */
     ldr r6, =GPIOB_MODER
     ldr r5, [r6]

     /* cannot do with movs, so use pc relative */
     ldr r4, =0x3003F
     mvns r4, r4
     ands r5, r5, r4
     ldr r4, =0x10015
     orrs r5, r5, r4
     str r5, [r6]


     /* setup PA(0,1,4,5,6,8,9,11,12) for seven segment A,B,C,D,E,F,G,DH for bits in
MODER */
     ldr r6, =GPIOA_MODER
     ldr r5, [r6]

     /* cannot do with movs, so use pc relative */
     ldr r4, =0x3CF3F0F
     mvns r4, r4
     ands r5, r5, r4
     ldr r4, =0x1451505
     orrs r5, r5, r4
     str r5, [r6]

    /* setup PA7 for BOTTON 00 for bits 14-15 in MODER*/
     ldr r6, =GPIOA_MODER
     ldr r5, [r6]
     /* cannot do with movs, so use pc relative*/
     ldr r4, =0x3
     lsls r4, r4 ,#14
     mvns r4, r4
     ands r5, r5, r4
     str r5, [r6]
```

```
school_ID:/*turn on school ID SSD shows '4024'*/

    bl on_SSD1
    bl off_SSD2
    bl off_SSD3
    bl off_SSD4
    bl number_4
    bl turn_on_DH

    ldr r1 ,=#30000   /*ABAULT 10 m SECOND*/
     bl delay

    bl off_SSD1
    bl on_SSD2
    bl off_SSD3
    bl off_SSD4
    bl number_0
    bl turn_off_DH

    ldr r1 ,=#30000   /*ABAULT 10 m SECOND*/
     bl delay

    bl off_SSD1
    bl off_SSD2
    bl on_SSD3
    bl off_SSD4
    bl number_2
    bl turn_off_DH

    ldr r1 ,=#30000   /*ABAULT 10 m SECOND*/
     bl delay

    bl off_SSD1
    bl off_SSD2
    bl off_SSD3
    bl on_SSD4
    bl number_4
    bl turn_off_DH

    ldr r1 ,=#30000   /*ABAULT 10 m SECOND*/
     bl delay

     bl button_ctrl1   /*go to number genarater*/

     bl school_ID  /*if not press button show ID*/
```

```
button_ctrl1:
    /*ctrl button connected to A7 in IDR*/
      ldr r6, =GPIOA_IDR
      ldr r5, [r6]
      lsrs r5,r5, #7
      movs r4, 0x1
      ands r5,r5,r4

      cmp r5,#0x1
      beq random_number
      bx lr


random_number:/*show random number */


    ldr r7, =#50 // i = 50
    loop:/* show random number abaut wait 1 sec*/

    bl on_SSD1
    bl off_SSD2
    bl off_SSD3
    bl off_SSD4
    ldr r4, =0x509
    bl rand1  //for example number_3 ,not number_0
    bl turn_on_DH

    ldr r1 ,=#30000   /*ABAULT 10 m SECOND*/
      bl delay

    bl off_SSD1
    bl on_SSD2
    bl off_SSD3
    bl off_SSD4
    ldr r4, =0x208
    bl rand2  //for example number_4
    bl turn_off_DH

    ldr r1 ,=#30000   /*ABAULT 10 m SECOND*/
      bl delay

    bl off_SSD1
    bl off_SSD2
    bl on_SSD3
    bl off_SSD4
    ldr r4, =0x107
    bl rand2  //for example number_5
    bl turn_off_DH

    ldr r1 ,=#30000   /*ABAULT 10 m SECOND*/
      bl delay

    bl off_SSD1
    bl off_SSD2
    bl off_SSD3
    bl on_SSD4
    ldr r4, =0x55
    bl rand2  //for example number_1
    bl turn_off_DH

    ldr r1 ,=#30000   /*ABAULT 10 m SECOND*/
      bl delay

    subs r7, r7, #1 // --i
    cmp r7, #0x0 // compare r7 with 0
    bne loop

    bl count_down
```

```
reach:
    bl on_SSD1
    bl on_SSD2
    bl on_SSD3
    bl on_SSD4
    bl number_0 //all leds show 0
    /*turn on led */
    ldr r6, =GPIOA_ODR
     ldr r5, [r6]
     ldr r4, =0x100
     orrs r5, r5, r4
     str r5, [r6]

    ldr r1 ,=#10000000   /*ABAULT 1 SECOND*/
     bl delay

    /*turn off led*/
     ldr r6, =GPIOA_ODR
     ldr r5, [r6]
     ldr r4, =0x100
     bics r5, r5, r4
     str r5, [r6]

     bl school_ID

rand1:
     loop1:
     subs r4,r4,#7
     cmp r4,#0x9
     bgt loop1
     beq number_9
     cmp r4,#0x8
     beq number_8
     cmp r4,#0x7
     beq number_7
     cmp r4,#0x6
     beq number_6
     cmp r4,#0x5
     beq number_5
     cmp r4,#0x4
     beq number_4
     cmp r4,#0x3
     beq number_3
     cmp r4,#0x2
     beq number_2
     cmp r4,#0x1
     beq number_1
     cmp r4,#0x0
     beq number_1
     bx lr
```

```
rand2:
    loop2:
    subs r4,r4,#3
    cmp r4,#0x9
    bgt loop2
    beq number_9
    cmp r4,#0x8
    beq number_8
    cmp r4,#0x7
    beq number_7
    cmp r4,#0x6
    beq number_6
    cmp r4,#0x5
    beq number_5
    cmp r4,#0x4
    beq number_4
    cmp r4,#0x3
    beq number_3
    cmp r4,#0x2
    beq number_2
    cmp r4,#0x1
    beq number_1
    cmp r4,#0x0
    beq number_0
    bx lr

count_down:/*start count*/
    bl count1/*count  digit*/
   /* bl count2
    bl count3
    bl count4*/
    bx lr

count1:

    ldr r4,=4
    cmp r4,#0x4
    beq four_to_zero
    bx lr
/*count2:
.equ a ,(0x4)
      ldr r4,=a
      cmp r4,#0x4
      beq four_to_zero
      bx lr
count3:
.equ a ,(0x4)
      ldr r4,=a
      cmp r4,#0x4
      beq four_to_zero
      bx lr
count4:
.equ a ,(0x4)
      ldr r4,=a
      cmp r4,#0x4
      beq four_to_zero
      bx lr*/
```

```
four_to_zero:

    bl on_SSD1
    bl off_SSD2
    bl off_SSD3
    bl off_SSD4
    bl number_4

    ldr r1 ,=#10000000   /*ABAULT 1 SECOND*/
     bl delay
     bl number_3

     bl button_ctrl2


     ldr r1 ,=#10000000   /*ABAULT 1 SECOND*/
     bl delay
     bl number_2
    bl button_ctrl2

     ldr r1 ,=#10000000   /*ABAULT 1 SECOND*/
     bl delay
     bl number_1
    bl button_ctrl2

     ldr r1 ,=#10000000   /*ABAULT 1 SECOND*/
     bl delay
     bl number_0
     bl reach

button_ctrl2:/*control the time */
    /*ctrl button connected to A7 in IDR*/
     ldr r6, =GPIOA_IDR
     ldr r5, [r6]
     lsrs r5,r5, #7
     movs r4, 0x1
     ands r5,r5,r4
     cmp r5,#0x1
     beq sleep_mod
     bx lr

sleep_mod:/*wait press button and show last number*/

    ldr r7,=#1

    bl on_SSD1

    bl number_0

    ldr r1 ,=#3000000   /*ABAULT 10 m SECOND*/
     bl delay

    bl button_ctrl3
    cmp r7, #0x1 // infinity loop
    beq sleep_mod
```

```
button_ctrl3:/*control the sleep mod */
    /*ctrl button connected to A7 in IDR*/
      ldr r6, =GPIOA_IDR
      ldr r5, [r6]
      lsrs r5,r5, #7
      movs r4, 0x1
      ands r5,r5,r4

      cmp r5,#0x1
      beq four_to_zero /*if press go to count_down*/
      bx lr          /*not press go to sleep mod*/

turn_on_DH:
    /*turn on led connected to DH in ODR*/
      ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4, =0x1
      orrs r5, r5, r4
      str r5, [r6]
      bx lr
turn_off_DH:
    /* turn off led connected to DH in ODR*/
    ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4, =0x1
      bics r5, r5, r4
      str r5, [r6]
      bx lr

number_0:/*G led close*/

    /* turn on led connected to A,B,C,D,E,F in ODR*/
      ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4, =0x1A70
      orrs r5, r5, r4
      str r5, [r6]

    /* turn off led connected to G in ODR*/
    ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4, =0x2
      bics r5, r5, r4
      str r5, [r6]

    bx lr /* to return back from function.*/

number_1:/*B and C leds open*/

    /* turn on led connected to B,C in ODR*/
      ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4, =0x840
      orrs r5, r5, r4
      str r5, [r6]

    /* turn off led connected to A,D,E,F,G in ODR*/
    ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4, =0x1232
      bics r5, r5, r4
      str r5, [r6]

      bx lr /* to return back from function.*/
```

```
number_2:/*A,B,G,E and D leds open*/

    /* turn on led connected to A,B,G,E,D in ODR*/
     ldr r6, =GPIOA_ODR
     ldr r5, [r6]
     ldr r4, =0x1262
     orrs r5, r5, r4
     str r5, [r6]

    /* turn off led connected to C in ODR*/
    ldr r6, =GPIOA_ODR
     ldr r5, [r6]
     ldr r4, =0x810
     bics r5, r5, r4
     str r5, [r6]


     bx lr /* to return back from function.*/

number_3:/*A,B,C,G and D leds open*/

    /* turn on led connected to A,B,C,G,D in ODR*/
     ldr r6, =GPIOA_ODR
     ldr r5, [r6]
     ldr r4, =0x1A42
     orrs r5, r5, r4
     str r5, [r6]

    /* turn off led connected to E in ODR*/
    ldr r6, =GPIOA_ODR
     ldr r5, [r6]
     ldr r4, =0x30
     bics r5, r5, r4
     str r5, [r6]

     bx lr /* to return back from function.*/

number_4:/*B,C,G and F leds open*/


    /* turn on led connected to B,C,G,F in ODR*/
     ldr r6, =GPIOA_ODR
     ldr r5, [r6]
     ldr r4, =0x852
     orrs r5, r5, r4
     str r5, [r6]


     /* turn off led connected to A in ODR*/
    ldr r6, =GPIOA_ODR
     ldr r5, [r6]
     ldr r4, =0x1220
     bics r5, r5, r4
     str r5, [r6]


    bx lr /* to return back from function.*/
```

```
number_6:/*B led close*/

    /* turn on led connected to A,C,D,E,F,G in ODR*/
       ldr r6, =GPIOA_ODR
       ldr r5, [r6]
       ldr r4, =0x1A32
       orrs r5, r5, r4
       str r5, [r6]


    /* turn off led connected to B in ODR*/
    ldr r6, =GPIOA_ODR
       ldr r5, [r6]
       ldr r4, =0x40
       bics r5, r5, r4
       str r5, [r6]

    bx lr /* to return back from function.*/

number_7:/*A,B and C leds open*/


    /* turn on led connected to A,B,C in ODR*/
       ldr r6, =GPIOA_ODR
       ldr r5, [r6]
       ldr r4, =0xA40
       orrs r5, r5, r4
       str r5, [r6]


       /* turn off led connected to D,E,F,G in ODR*/
    ldr r6, =GPIOA_ODR
       ldr r5, [r6]
       ldr r4, =0x1032
       bics r5, r5, r4
       str r5, [r6]

    bx lr /* to return back from function.*/

number_8:/*All leds open*/

    /* turn on led connected to all in ODR*/
       ldr r6, =GPIOA_ODR
       ldr r5, [r6]
       ldr r4, =0x1A72
       orrs r5, r5, r4
       str r5, [r6]


    bx lr /* to return back from function.*/

number_9:/*E led close*/

    /* turn on led connected to A,B,C,D,F,G in ODR*/
       ldr r6, =GPIOA_ODR
       ldr r5, [r6]
       ldr r4, =0x1A52
       orrs r5, r5, r4
       str r5, [r6]

       /* turn off led connected to E in ODR*/
    ldr r6, =GPIOA_ODR
       ldr r5, [r6]
       ldr r4, =0x20
       bics r5, r5, r4
       str r5, [r6]

    bx lr /* to return back from function.*/
```

```
on_SSD4:   /* turn on SSD 4 (RİGHT).*/
     ldr r6, =GPIOB_ODR
     ldr r5, [r6]
     ldr r4, =0x2
     orrs r5, r5, r4
     str r5, [r6]

     bx lr /* to return back from function.*/

off_SSD4:    /* turn OFF SSD 4.*/
    ldr r6, =GPIOB_ODR
     ldr r5, [r6]
     ldr r4, =0x2
     bics r5, r5, r4
     str r5, [r6]

     bx lr /* to return back from function.*/

on_SSD3:    /* turn on SSD 3.*/
     ldr r6, =GPIOB_ODR
     ldr r5, [r6]
     ldr r4, =0x1
     orrs r5, r5, r4
     str r5, [r6]

     bx lr /* to return back from function.*/

off_SSD3:    /* turn OFF SSD 3.*/
    ldr r6, =GPIOB_ODR
     ldr r5, [r6]
     ldr r4, =0x1
     bics r5, r5, r4
     str r5, [r6]

     bx lr /* to return back from function.*/

on_SSD2:   /* turn on SSD 2.*/
     ldr r6, =GPIOB_ODR
     ldr r5, [r6]
     ldr r4, =0x4
     orrs r5, r5, r4
     str r5, [r6]

     bx lr /* to return back from function.*/

off_SSD2:    /* turn OFF SSD 2.*/
    ldr r6, =GPIOB_ODR
     ldr r5, [r6]
     ldr r4, =0x4
     bics r5, r5, r4
     str r5, [r6]

     bx lr /* to return back from function.*/
```

```
on_SSD1:   /* turn on SSD 1(LEFT).*/
      ldr r6, =GPIOB_ODR
      ldr r5, [r6]
      ldr r4, =0x100
      orrs r5, r5, r4
      str r5, [r6]

      bx lr /* to return back from function.*/

off_SSD1:    /* turn OFF SSD 1.*/
    ldr r6, =GPIOB_ODR
      ldr r5, [r6]
      ldr r4, =0x100
      bics r5, r5, r4
      str r5, [r6]

      bx lr /* to return back from function.*/

delay:
    subs r1,r1,#1
    bne delay
    bx lr
```