ELECTRONICS ENGINEERING
ELEC335 - MICROPROCESSORS LABORATORY

LAB #2

Muhammet Cemal Eryiğit	1801022024
Şahabettin Alpcan Soydaş	1801022014
Mert Tuncay Firil	1901022285

1. PROBLEM 1

1.1. FLOW CHART

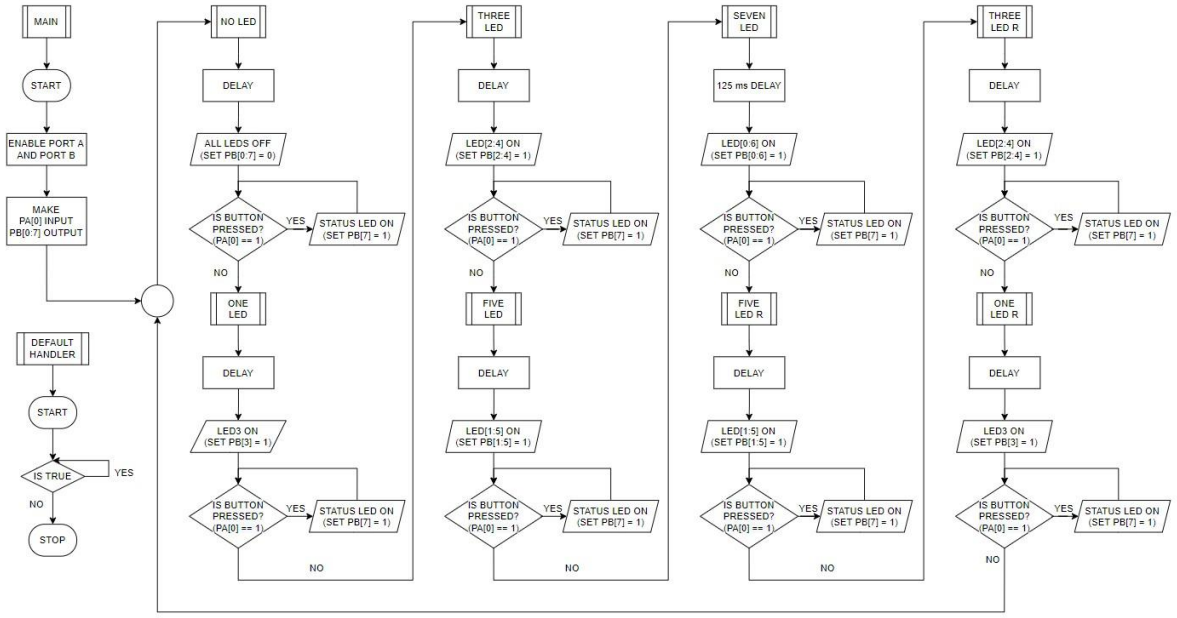

Figure 1. Flowchart

1.2. CONNECTION DIAGRAM

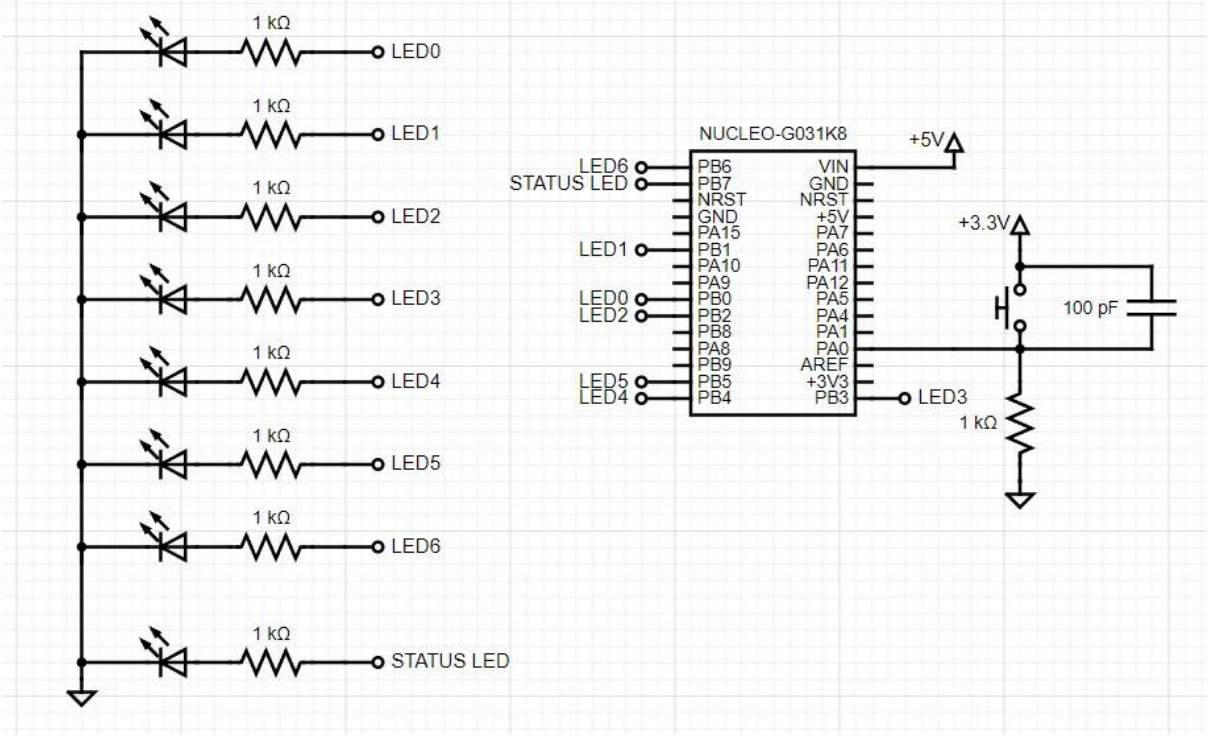

Figure 2. Connection Diagram

1.3. REQUESTED PICTURES FOR QUESTIONS

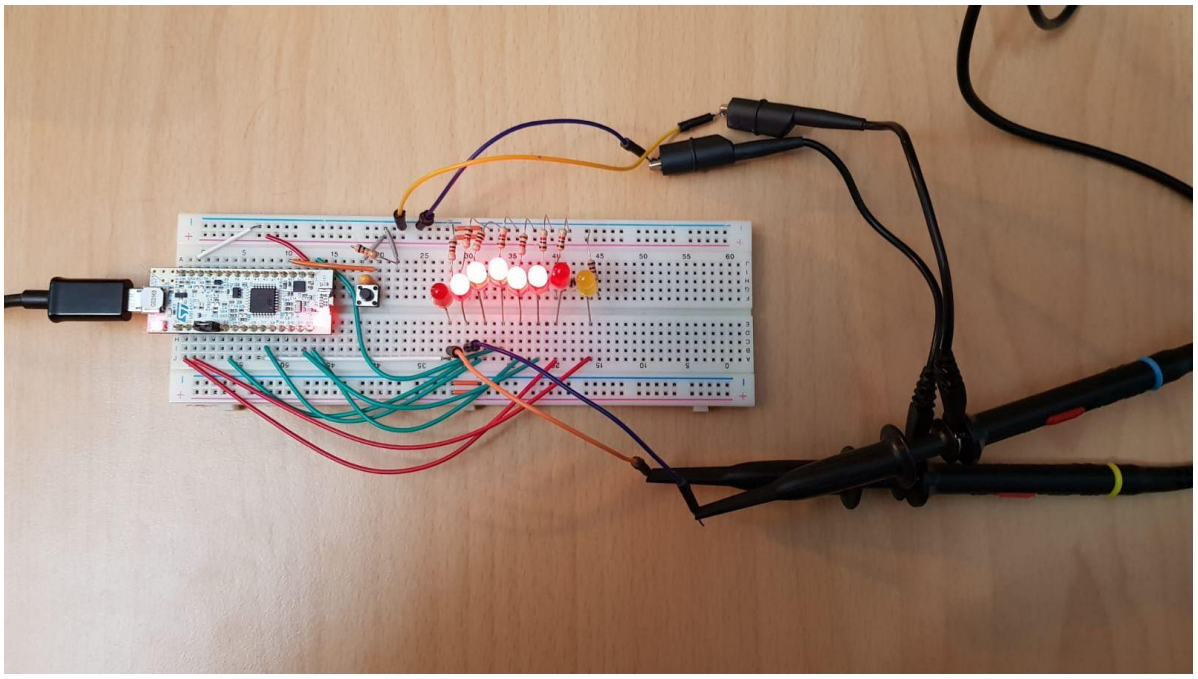

Figure 3. Circuit for Problem 1 Delay 125ms

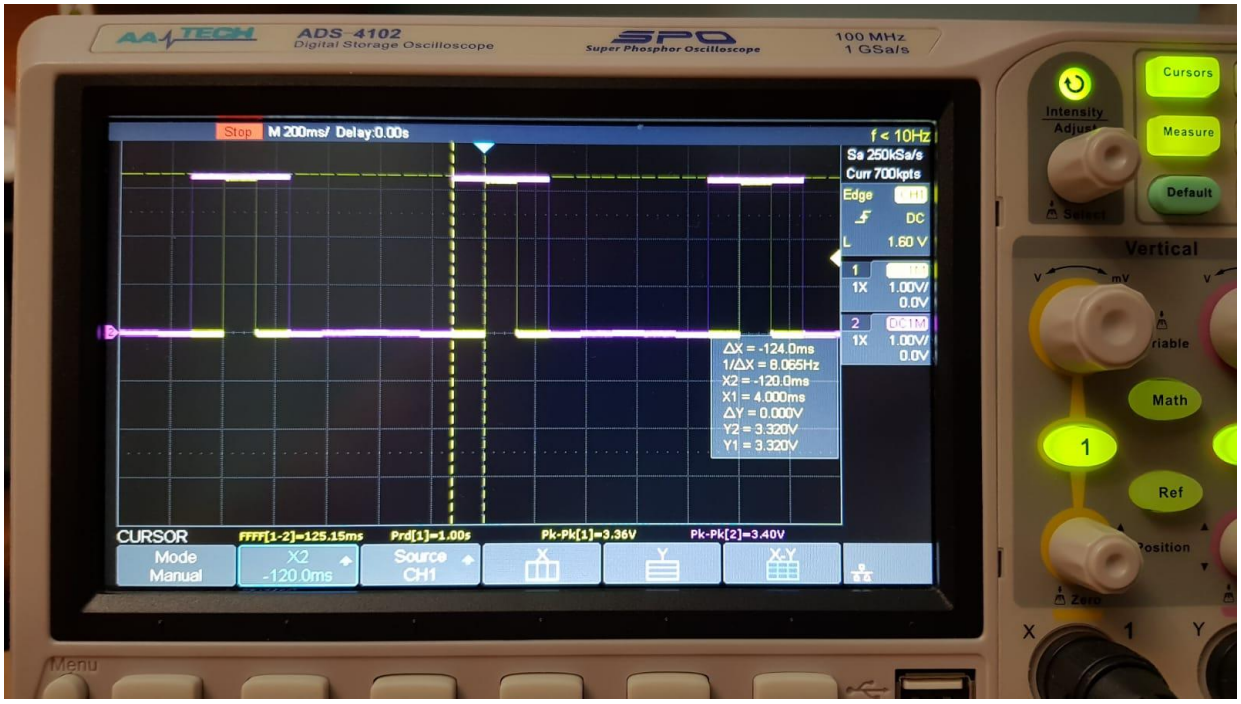

Figure 4. Oscilloscope Display for Problem 1 Delay 125ms

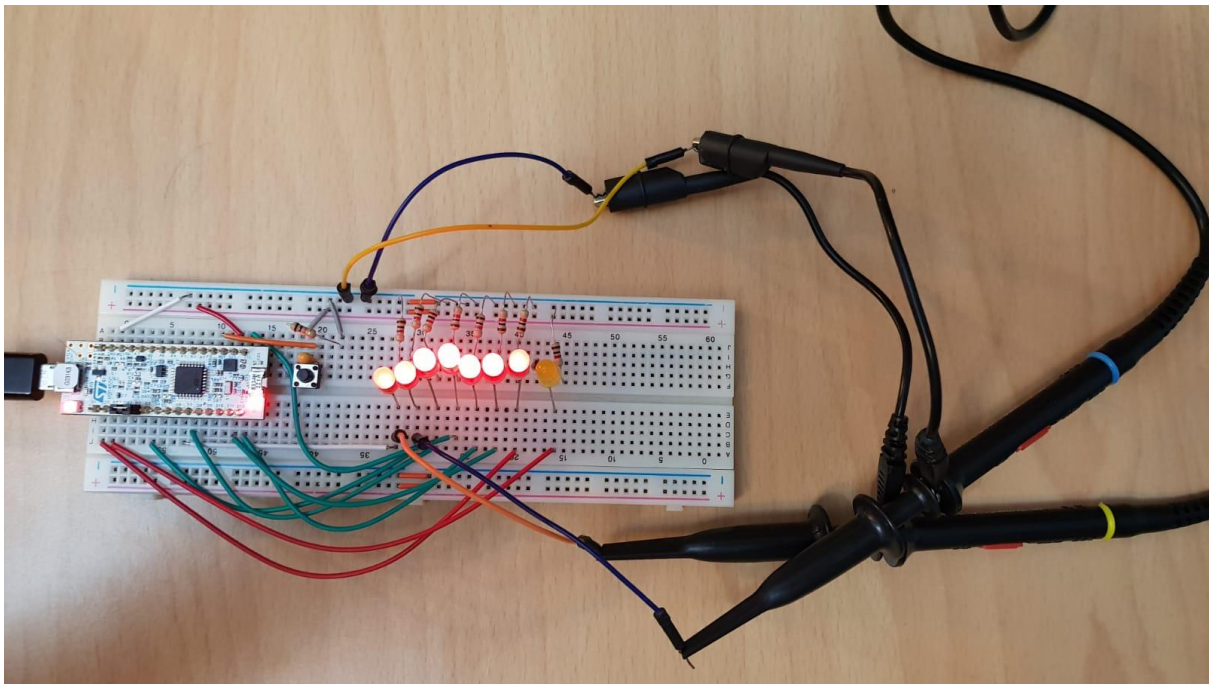

Figure 5. Circuit for Problem 1 Delay 5ms

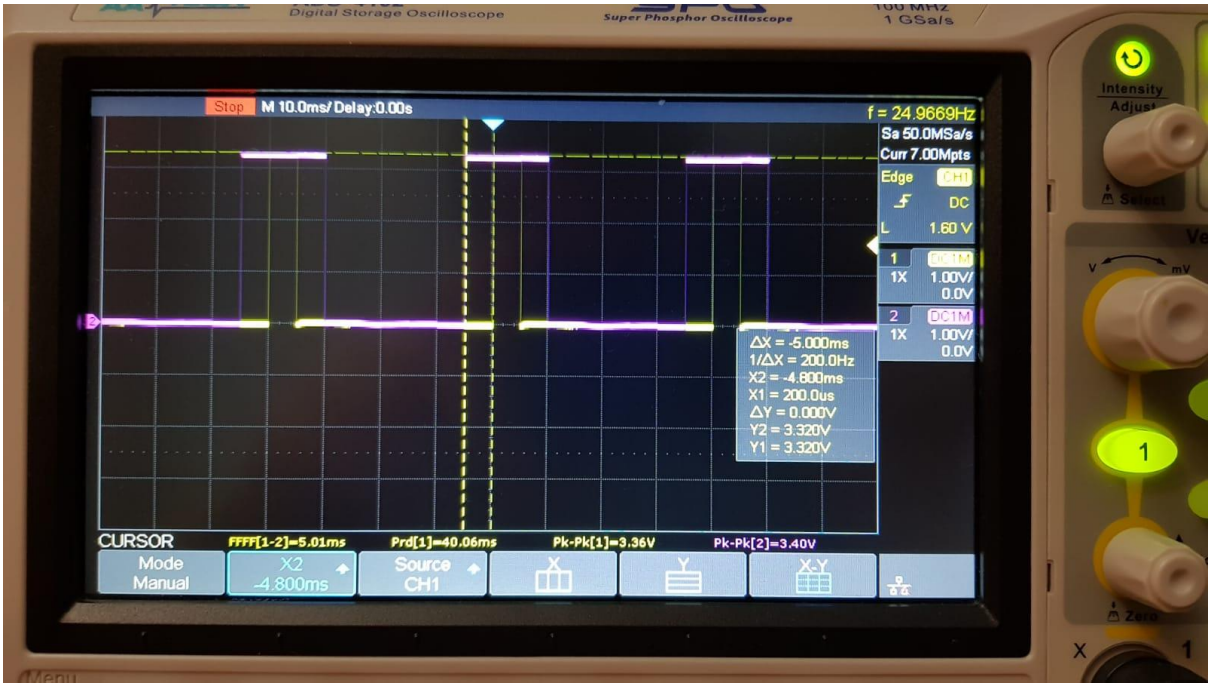

Figure 6. Oscilloscope Display for Problem 1 Delay 5ms

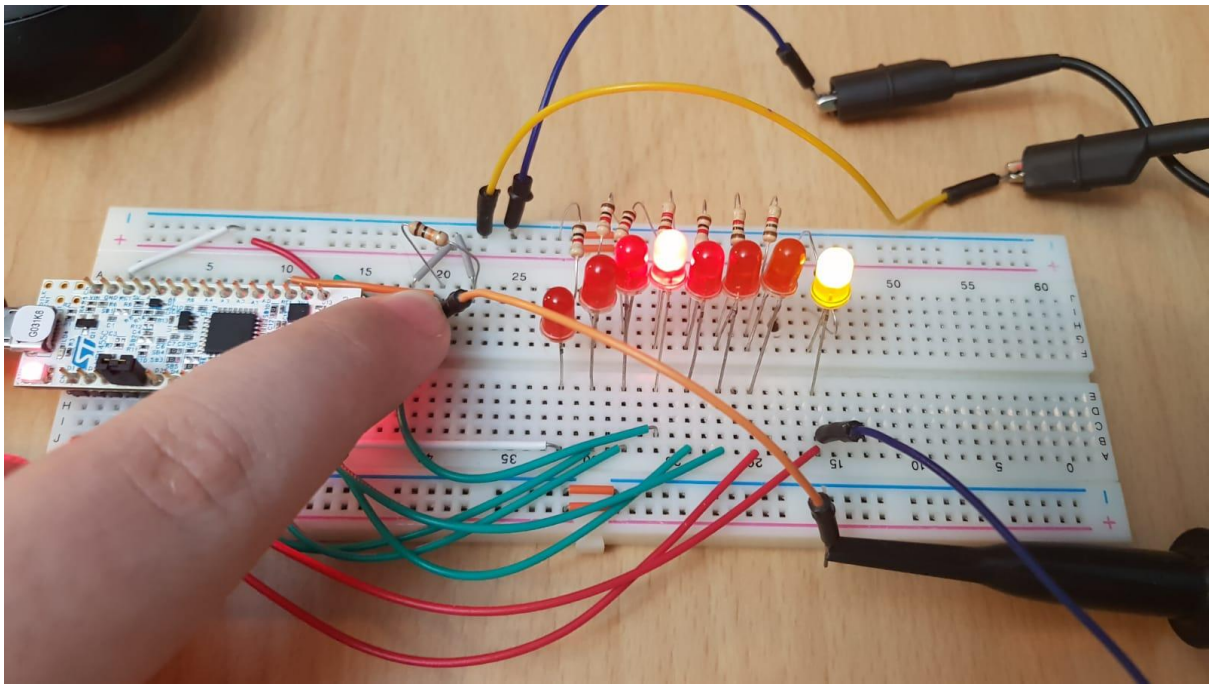

Figure 7. Circuit for Problem 1 Status LED and Button Delay

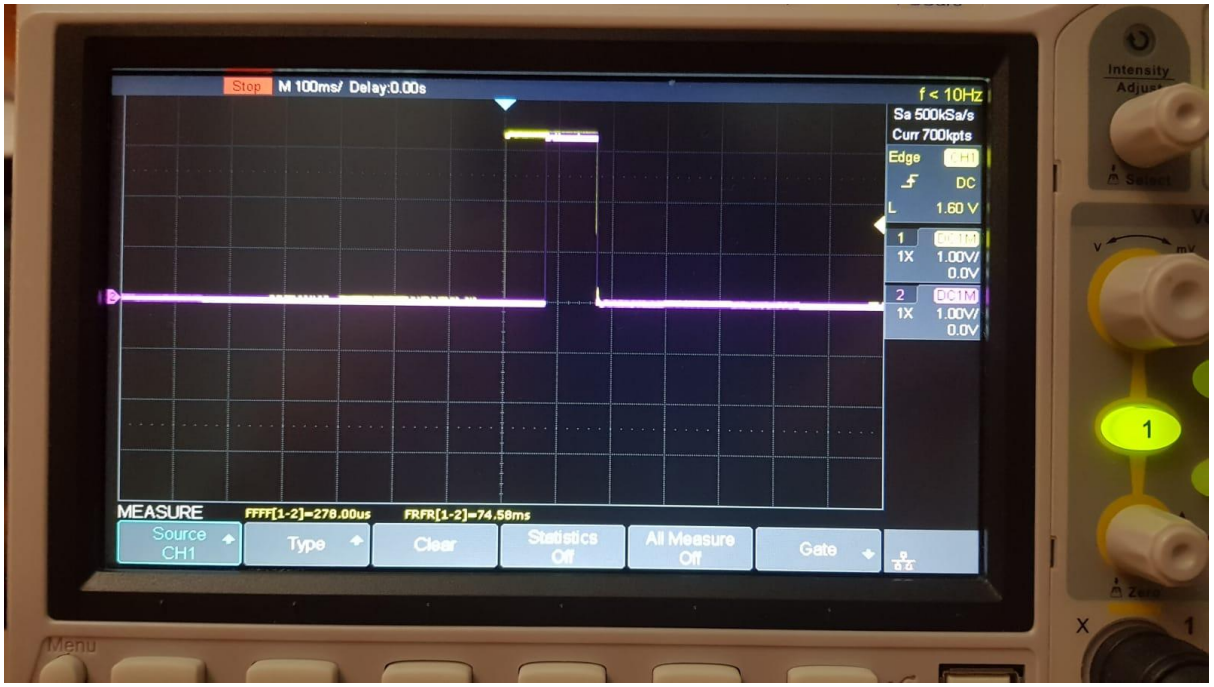

Figure 8. Oscilloscope Display for Problem 1 Status LED and Button Delay

1.4. CODE

```
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

//LEFT TO RIGHT LEDS 0-1-2-3-4-5-6-STATUS => PB0-PB1-PB2-PB3-PB4-PB5-PB6-PB7
// BUTTON-PA0
/* 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 REG OFFSET
.equ GPIOB_ODR,        (GPIOB_BASE + (0x14)) //GPIOB_ODR REG OFFSET

//.equ DELAY,          (0x022E0) //5 ms delay 0x022E0 = 8928
.equ DELAY,            (0x367EE) //125 ms delay 0x367EE = 223214
//.equ DELAY,          (0x1B3F72) //1s delay 0x1B3F72 = 1785714

/* 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 and GPIOB clock */
     ldr r6, =RCC_IOPENR
     ldr r5, [r6]
     /* movs expects imm8, so this should be fine */
     movs r4, 0x3 //PORT A AND B ACTIVE
     orrs r5, r5, r4
     str r5, [r6]

     /* setup button */
     ldr r6, =GPIOA_MODER
     ldr r5, [r6]

     /* cannot do with movs, so use pc relative */

     movs r4, 0x3
     bics r5, r5, r4
     movs r4, 0x0 // 0x00 PA0 input mode
     orrs r5, r5, r4
     str r5, [r6] // store r5 data to GPIOC_MODER //PA0-Button

     /* setup LEDs */
     ldr r6, =GPIOB_MODER
     ldr r5, [r6]

     /* */
     movs r4, 0xFF
     bics r5, r5, r4
     movs r4, 0x55 //first 4 pin 01010101=>0x55
     orrs r5, r5, r4
     lsls r5, 0x8 //shifting 8 times left to set last 4 pin as output
     orrs r5, r5, r4
     str r5, [r6] // 8 pin as output


     //button connected to PA0
     ldr r2, =DELAY
     no_led:
          ldr r6, =GPIOB_ODR
          ldr r5, [r6]
          movs r4, #0x7F
          ands r5, r5, r4
          str r5, [r6]
          subs r2, r2, #1
          bne no_led
          ldr r6, =GPIOB_ODR
```

```
        ldr r5, [r6]
        movs r4, #0xFF
        bics r5, r5, r4
        movs r4, #0x0
        orrs r5, r5, r4
        str r5, [r6]
        button_no_led:
            ldr r2, =DELAY
            ldr r6, =GPIOA_IDR
            ldr r5, [r6]
            movs r4, #0x1 //r4=0x1
            ands r5, r5, r4 // GPIOA_IDR and r4
            cmp r5, #0x1 //if GPIOA_IDR[0]==1
            bne one_led
            ldr r6, =GPIOB_ODR
            ldr r5, [r6]
            movs r4, #0x80
            orrs r5, r5, r4
            str r5, [r6]
            bl button_no_led

    one_led:
        ldr r6, =GPIOB_ODR
        ldr r5, [r6]
        movs r4, #0x7F
        ands r5, r5, r4
        str r5, [r6]
        subs r2, r2, #1
        bne one_led
        ldr r6, =GPIOB_ODR
        ldr r5, [r6]
        movs r4, #0x8
        orrs r5, r5, r4
        str r5, [r6]
        button_one_led:
            ldr r2, =DELAY
            ldr r6, =GPIOA_IDR
            ldr r5, [r6]
            movs r4, #0x1
            ands r5, r5, r4
            cmp r5, #0x1
            bne three_led
            ldr r6, =GPIOB_ODR
            ldr r5, [r6]
            movs r4, #0x80
            orrs r5, r5, r4
            str r5, [r6]
            bl button_one_led

    three_led:
        ldr r6, =GPIOB_ODR
        ldr r5, [r6]
```

```
        movs r4, #0x7F
        ands r5, r5, r4
        str r5, [r6]
        subs r2, r2, #1
        bne three_led
        ldr r6, =GPIOB_ODR
        ldr r5, [r6]
        movs r4, #0x1C
        orrs r5, r5, r4
        str r5, [r6]
        button_three_led:
             ldr r2, =DELAY
             ldr r6, =GPIOA_IDR
             ldr r5, [r6]
             movs r4, #0x1
             ands r5, r5, r4
             cmp r5, #0x1
             bne five_led
             ldr r6, =GPIOB_ODR
             ldr r5, [r6]
             movs r4, #0x80
             orrs r5, r5, r4
             str r5, [r6]
             bl button_three_led

    five_led:
        ldr r6, =GPIOB_ODR
        ldr r5, [r6]
        movs r4, #0x7F
        ands r5, r5, r4
        str r5, [r6]
        subs r2, r2, #1
        bne five_led
        ldr r6, =GPIOB_ODR
        ldr r5, [r6]
        movs r4, #0x3E
        orrs r5, r5, r4
        str r5, [r6]
        button_five_led:
             ldr r2, =DELAY
             ldr r6, =GPIOA_IDR
             ldr r5, [r6]
             movs r4, #0x1
             ands r5, r5, r4
             cmp r5, #0x1
             bne seven_led
             ldr r6, =GPIOB_ODR
             ldr r5, [r6]
             movs r4, #0x80
             orrs r5, r5, r4
             str r5, [r6]
             bl button_five_led
```

```
seven_led:
     ldr r6, =GPIOB_ODR
     ldr r5, [r6]
     movs r4, #0x7F
     ands r5, r5, r4
     str r5, [r6]
     subs r2, r2, #1
     bne seven_led
     ldr r6, =GPIOB_ODR
     ldr r5, [r6]
     movs r4, #0x7F
     orrs r5, r5, r4
     str r5, [r6]
     button_seven_led:
          ldr r2, =DELAY
          ldr r6, =GPIOA_IDR
          ldr r5, [r6]
          movs r4, #0x1
          ands r5, r5, r4
          cmp r5, #0x1
          bne five2_led
          ldr r6, =GPIOB_ODR
          ldr r5, [r6]
          movs r4, #0x80
          orrs r5, r5, r4
          str r5, [r6]
          bl button_seven_led

temp:
bl no_led

five2_led:
     ldr r6, =GPIOB_ODR
     ldr r5, [r6]
     movs r4, #0x7F
     ands r5, r5, r4
     str r5, [r6]
     subs r2, r2, #1
     bne five2_led
     ldr r6, =GPIOB_ODR
     ldr r5, [r6]
     movs r4, #0x3E
     ands r5, r5, r4
     str r5, [r6]
     button_five2_led:
          ldr r2, =DELAY
          ldr r6, =GPIOA_IDR
          ldr r5, [r6]
          movs r4, #0x1
          ands r5, r5, r4
          cmp r5, #0x1
```

```
            bne three2_led
            ldr r6, =GPIOB_ODR
            ldr r5, [r6]
            movs r4, #0x80
            orrs r5, r5, r4
            str r5, [r6]
            bl button_five2_led

    three2_led:
         ldr r6, =GPIOB_ODR
         ldr r5, [r6]
         movs r4, #0x7F
         ands r5, r5, r4
         str r5, [r6]
         subs r2, r2, #1
         bne three2_led
         ldr r6, =GPIOB_ODR
         ldr r5, [r6]
         movs r4, #0x1C
         ands r5, r5, r4
         str r5, [r6]
         button_three2_led:
              ldr r2, =DELAY
              ldr r6, =GPIOA_IDR
              ldr r5, [r6]
              movs r4, #0x1
              ands r5, r5, r4
              cmp r5, #0x1
              bne one2_led
              ldr r6, =GPIOB_ODR
              ldr r5, [r6]
              movs r4, #0x80
              orrs r5, r5, r4
              str r5, [r6]
              bl button_three2_led

    one2_led:
         ldr r6, =GPIOB_ODR
         ldr r5, [r6]
         movs r4, #0x7F
         ands r5, r5, r4
         str r5, [r6]
         subs r2, r2, #1
         bne one2_led
         ldr r6, =GPIOB_ODR
         ldr r5, [r6]
         movs r4, #0x8
         ands r5, r5, r4
         str r5, [r6]
         button_one2_led:
              ldr r2, =DELAY
              ldr r6, =GPIOA_IDR
```

```
            ldr r5, [r6]
            movs r4, #0x1
            ands r5, r5, r4
            cmp r5, #0x1
            bne temp
            ldr r6, =GPIOB_ODR
            ldr r5, [r6]
            movs r4, #0x80
            orrs r5, r5, r4
            str r5, [r6]
            bl button_one2_led

/* this should never get executed */
nop
```

Figure 9. Code for Problem 1

1.5. CONCLUSION

In this problem, diamond pattern is implemented with external LEDs. 8 LED and 1 push button is used. The button is used to play or pause the pattern. When the button is pressed, pattern stops, status led lights on and when the button is released, the pattern continues from where it is stopped. To solve bouncing problem, 125ms delay is implemented.

2. PROBLEM 2

2.1. FLOW CHART

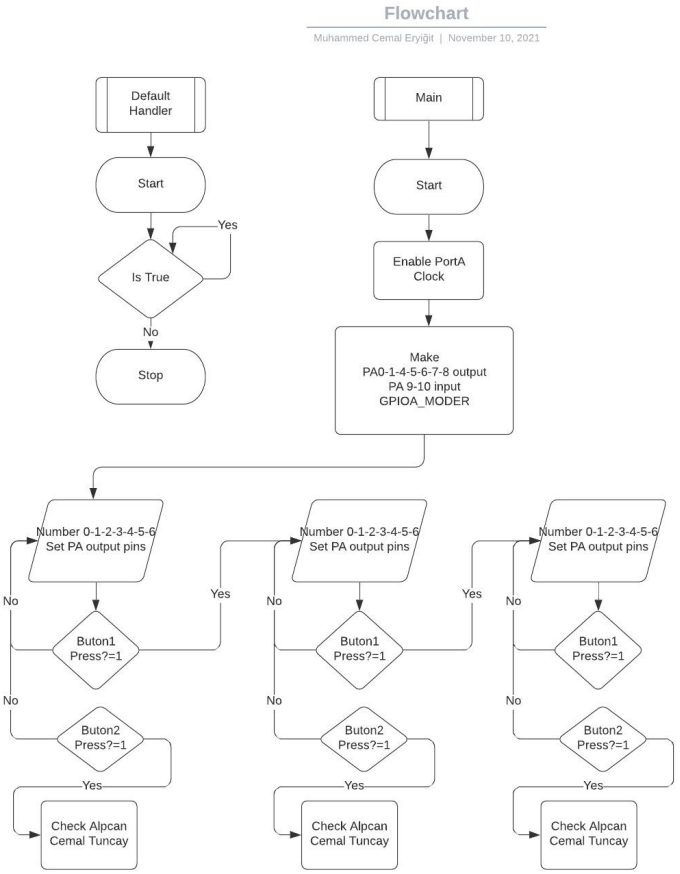

Figure 10. Flowchart

2.2. CONNECTION DIAGRAM

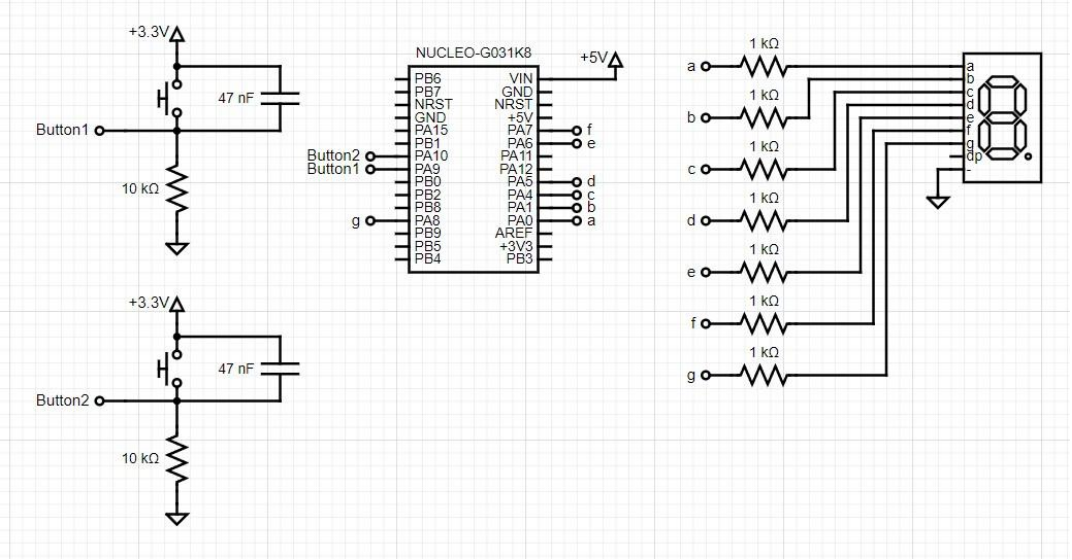

Figure 11. Connection Diagram

2.3. REQUESTED PICTURES FOR QUESTIONS

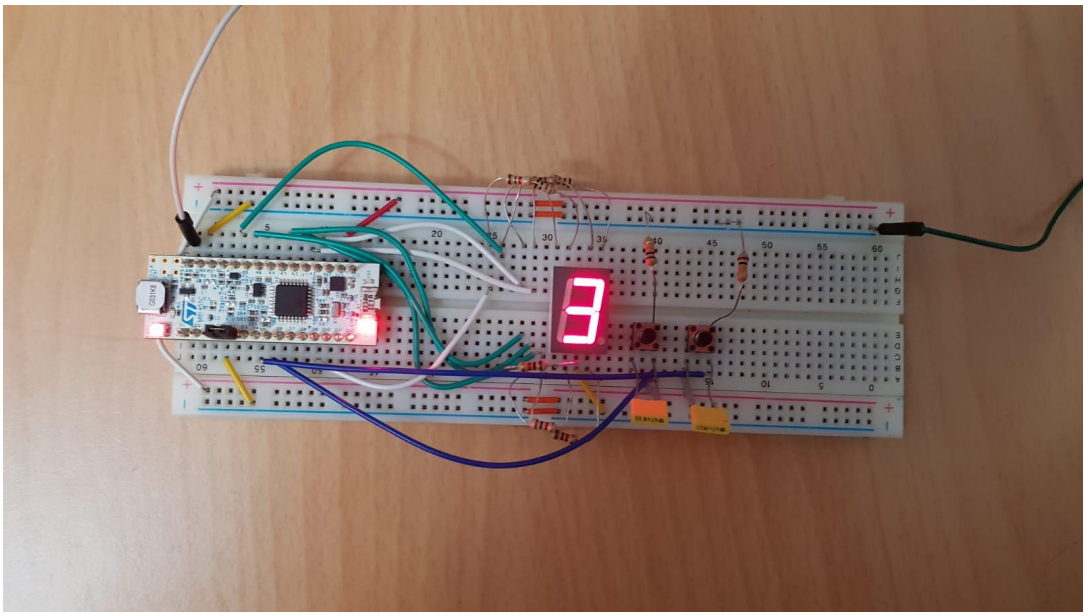

Figure 12. Circuit for Problem 2

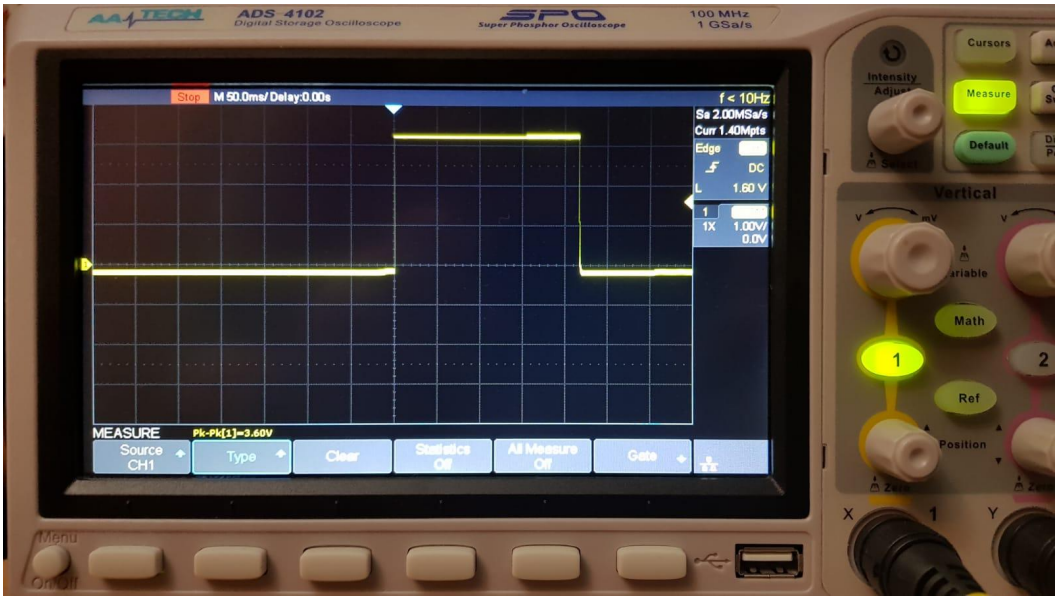

Figure 13. Oscilloscope Display for Problem 2 Button PA9

2.4. CODE

```
.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)          // GPIOC base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOC MODER register offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOC ODR register offset
.equ GPIOA_IDR,        (GPIOA_BASE +(0x10))
//.equ DELAY,               (0xC3500)
/* 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 GPIOC clock, bit2 on IOPENR */
     ldr r6, =RCC_IOPENR
```

```
ldr r5, [r6]
/* movs expects imm8, so this should be fine */
movs r4, 0x1
orrs r5, r5, r4
str r5, [r6]

/* setup PC6 for led 01 for bits 12-13 in MODER */
ldr r6, =GPIOA_MODER
ldr r5, [r6]
/* cannot do with movs, so use pc relative */
ldr r4, =0x3FFFFF
bics r5,r5,r4
ldr r4, =0x15505
orrs r5, r5, r4
str r5, [r6]
ldr r1,= 0xC3500
set:
movs r7,#0x0
b check


number0:
     subs r1, r1, #1
     bne number0
     ldr r6, =GPIOA_ODR
     ldr r5, [r6]
     ldr r4,= #0x0
     ands r5,r5,r4
     str r5, [r6]
     ldr r3,= 0xF3
     orrs r5,r5,r3
     str r5, [r6]
     ldr r1,= 0xC3500
     ldr r6,=GPIOA_IDR
     ldr r2, [r6]
     movs r4, #0x1
     lsls r4, #0xA
     ands r2, r2, r4
     lsrs r2, #10
     cmp r2,0x1
     beq check
     ldr r3, [r6]
     movs r4, #0x1
     lsls r4, #0x9
     ands r3, r3, r4
     lsrs r3, #7
     orrs r3,r3,r7
     cmp r3, 0x5
     beq number4
     orrs r3,r3,r7
     cmp r3, 0x6
     beq number6
```

```
      orrs r3,r3,r7
      cmp r3, 0x7
      beq number5
      bne number0
           number5:
      subs r1, r1, #1
      bne number5
      ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4,= #0x0
      ands r5,r5,r4
      str r5, [r6]
      ldr r1,= 0xC3500
      ldr r3,= 0x1B1
      orrs r5,r5,r3
      str r5, [r6]
      ldr r6,=GPIOA_IDR
      ldr r2, [r6]
      movs r4, #0x1
      lsls r4, #0xA
      ands r2, r2, r4
      lsrs r2, #10
      cmp r2,0x1
      beq check
      ldr r3, [r6]
      movs r4, #0x1
      lsls r4, #0x9
      ands r3, r3, r4
      lsrs r3, #9
      cmp r3, 0x1
      beq number4
      bne number5
number1:
    subs r1, r1, #1
      bne number1
      ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4,= #0x0
      ands r5,r5,r4
      str r5, [r6]
      ldr r1,= 0xC3500
      ldr r3,= 0x12
      orrs r5,r5,r3
      str r5, [r6]
      ldr r6,=GPIOA_IDR
      ldr r2, [r6]
      movs r4, #0x1
      lsls r4, #0xA
      ands r2, r2, r4
      lsrs r2, #10
      cmp r2,0x1
      beq check
```

```
      ldr r3, [r6]
      movs r4, #0x1
      lsls r4, #0x9
      ands r3, r3, r4
      lsrs r3, #9
      cmp r3, 0x1
      beq number0
      bne number1
           check:
      subs r1, r1, #1
      bne check
      ldr r1,= 0xC3500
      adds r7,0x1
      cmp r7,#0x1
      beq alpcan
      cmp r7,#0x2
      beq cemal
      cmp r7,0x3
      beq tuncay
      cmp r7,0x4
      beq set
           number6:
      subs r1, r1, #1
      bne number6
      ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4,= #0x0
      ands r5,r5,r4
      str r5, [r6]
      ldr r1,= 0xC3500
      ldr r3,= 0x1F1
      orrs r5,r5,r3
      str r5, [r6]
      ldr r6,=GPIOA_IDR
      ldr r2, [r6]
      movs r4, #0x1
      lsls r4, #0xA
      ands r2, r2, r4
      lsrs r2, #10
      cmp r2,0x1
      beq check
      ldr r3, [r6]
      movs r4, #0x1
      lsls r4, #0x9
      ands r3, r3, r4
      lsrs r3, #9
      cmp r3, 0x1
      beq number5
      bne number6
number2:
    subs r1, r1, #1
     bne number2
```

```
      ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4,= #0x0
      ands r5,r5,r4
      str r5, [r6]
      ldr r1,= 0xC3500
      ldr r3,= 0x163
      orrs r5,r5,r3
      str r5, [r6]
      ldr r6,=GPIOA_IDR
      ldr r2, [r6]
      movs r4, #0x1
      lsls r4, #0xA
      ands r2, r2, r4
      lsrs r2, #10
      cmp r2,0x1
      beq check
      ldr r3, [r6]
      movs r4, #0x1
      lsls r4, #0x9
      ands r3, r3, r4
      lsrs r3, #9
      cmp r3, 0x1
      beq number1
      bne number2
number4:
      subs r1, r1, #1
      bne number4
      ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4,= #0x0
      ands r5,r5,r4
      str r5, [r6]
      ldr r1,= 0xC3500
      ldr r3,= 0x192
      orrs r5,r5,r3
      str r5, [r6]
      ldr r6,=GPIOA_IDR
      ldr r2, [r6]
      movs r4, #0x1
      lsls r4, #0xA
      ands r2, r2, r4
      lsrs r2, #10
      cmp r2,0x1
      beq check
      ldr r3, [r6]
      movs r4, #0x1
      lsls r4, #0x9
      ands r3, r3, r4
      lsrs r3, #9
      cmp r3, 0x1
      beq number3
```

```
      bne number4

number3:
      subs r1, r1, #1
      bne number3
      ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4,= #0x0
      ands r5,r5,r4
      str r5, [r6]
      ldr r3,= 0x133
      orrs r5,r5,r3
      str r5, [r6]
      ldr r1,= 0xC3500
      ldr r6,=GPIOA_IDR
      ldr r2, [r6]
      movs r4, #0x1
      lsls r4, #0xA
      ands r2, r2, r4
      lsrs r2, #10
      cmp r2,0x1
      beq check
      ldr r3, [r6]
      movs r4, #0x1
      lsls r4, #0x9
      ands r3, r3, r4
      lsrs r3, #9
      cmp r3, 0x1
      beq number2
      bne number3



alpcan:
      subs r1, r1, #1
      bne alpcan
      ldr r7,= 0x1
      ldr r1,= 0xC3500
      b number4
cemal:
      subs r1, r1, #1
      bne cemal
      ldr r7,= 0x2
      ldr r1,= 0xC3500
      b number6
tuncay:
      subs r1, r1, #1
      bne tuncay
      ldr r7,= 0x3
      ldr r1,= 0xC3500
      b number5
```

```
          /* turn on led connected to C6 in ODR
ldr r6, =GPIOA_ODR
ldr r5, [r6]
movs r4, 0x40
orrs r5, r5, r4
str r5, [r6]*/

/* for(;;); */
b check

/* this should never get executed */
nop
```

Figure 14. Code for Problem 2

2.5. CONCLUSION

In this problem, decimal counter for three different number is implemented. 1 seven-segment display, 2 buttons are used. Also resistors are used to prevent our components from high current. Seven-segment displays holds the “4-5-6” numbers which are last digits of our school id. The first button counts from that numbers to zero. The second button changes the number. To solve bouncing problem, software and hardware solutions are combined. Software delay is implemented and a capacitor is wired to the circuit.