GEBZE TECHNICAL UNIVERSITY

ELECTRONIC ENGINEERING

ELEC335 – MICROPROCESSORS LABAORATORY

LAB 3

HAZIRLAYANLAR
1801022035 – Ruveyda Dilara Günal
1801022071 – Alperen Arslan
1901022255 – Emirhan Köse

PROBLEMS

Problem 1:

Write a program to blink an external LED at roughly 1 second intervals.

● Capture scope output.

● Is there any difference between the code size when you implemented this in assembly? What do you think accounts for that?

● Is the delay number different then the assembly implementation? Explain.

● Change the optimization to -O2, and try again, is there any change? If so, explain what happened. Is there any difference between the code sizes?

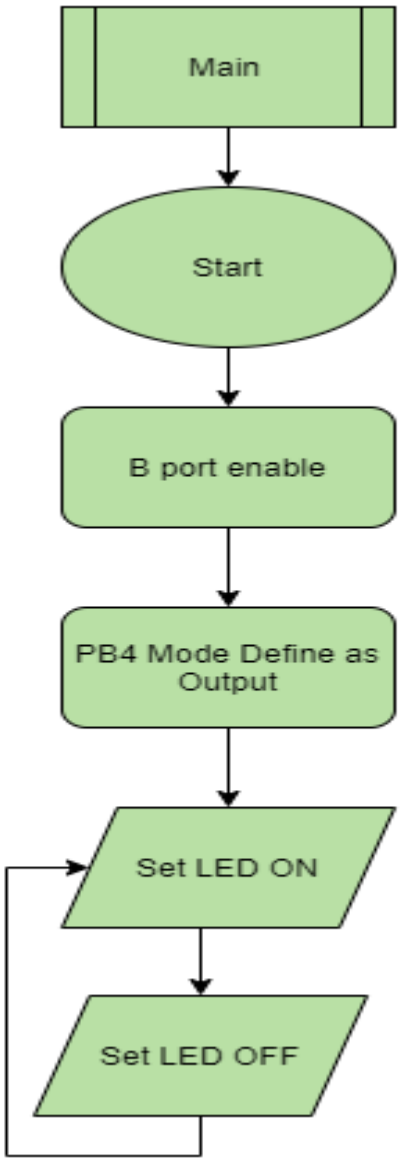

Figure 1: Problem 1 Flowchart

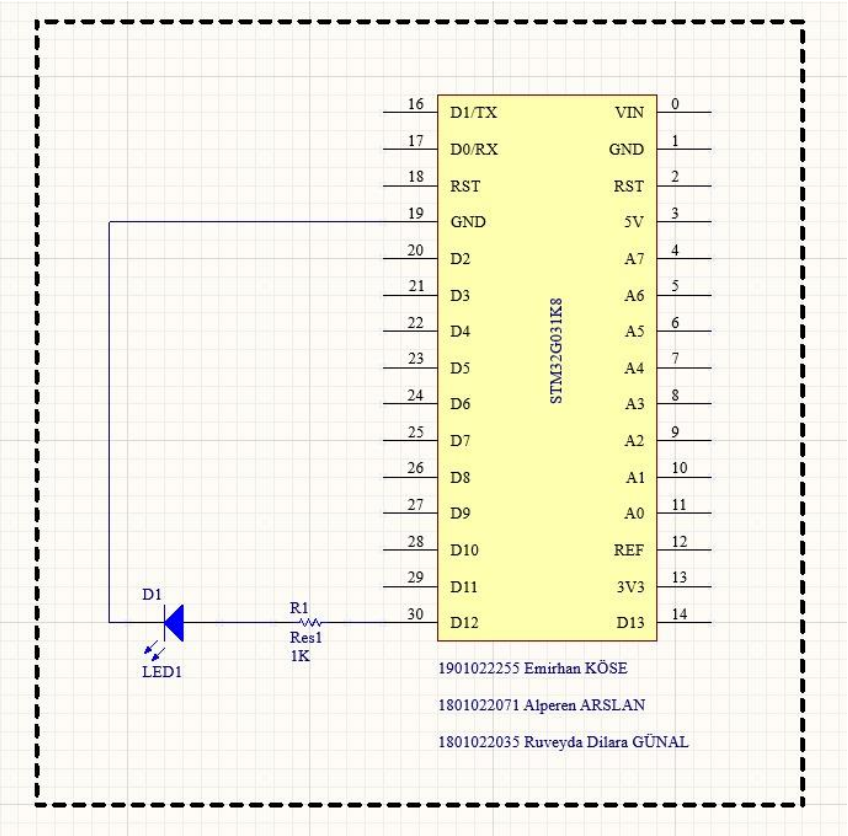

Figure 2: Problem 2 Block Diagram

```c
// main.c
// Author: Alperen Arslan, Emirhan Köse, Ruveyda Dilara Günal

#include "stm32g0xx.h"

#define LEDDELAY 1600000 //In 1 second, MCU process 16 mHz

void delay(volatile uint32_t);

int main(void) {

    // Enable GPIOB clock
    RCC->IOPENR |= 2U;

    // Setup PB4 as output
    GPIOB->MODER = 0xDFF;

    // Turn on LED
    GPIOB->ODR |= (1U << 4);

    while(1) {
        delay(LEDDELAY);
        GPIOB->ODR ^= (1U << 4); // Toggle LED
```

```
    }
    return 0;
}

void delay(volatile uint32_t s) {
    for(; s>0; s--);
}
```

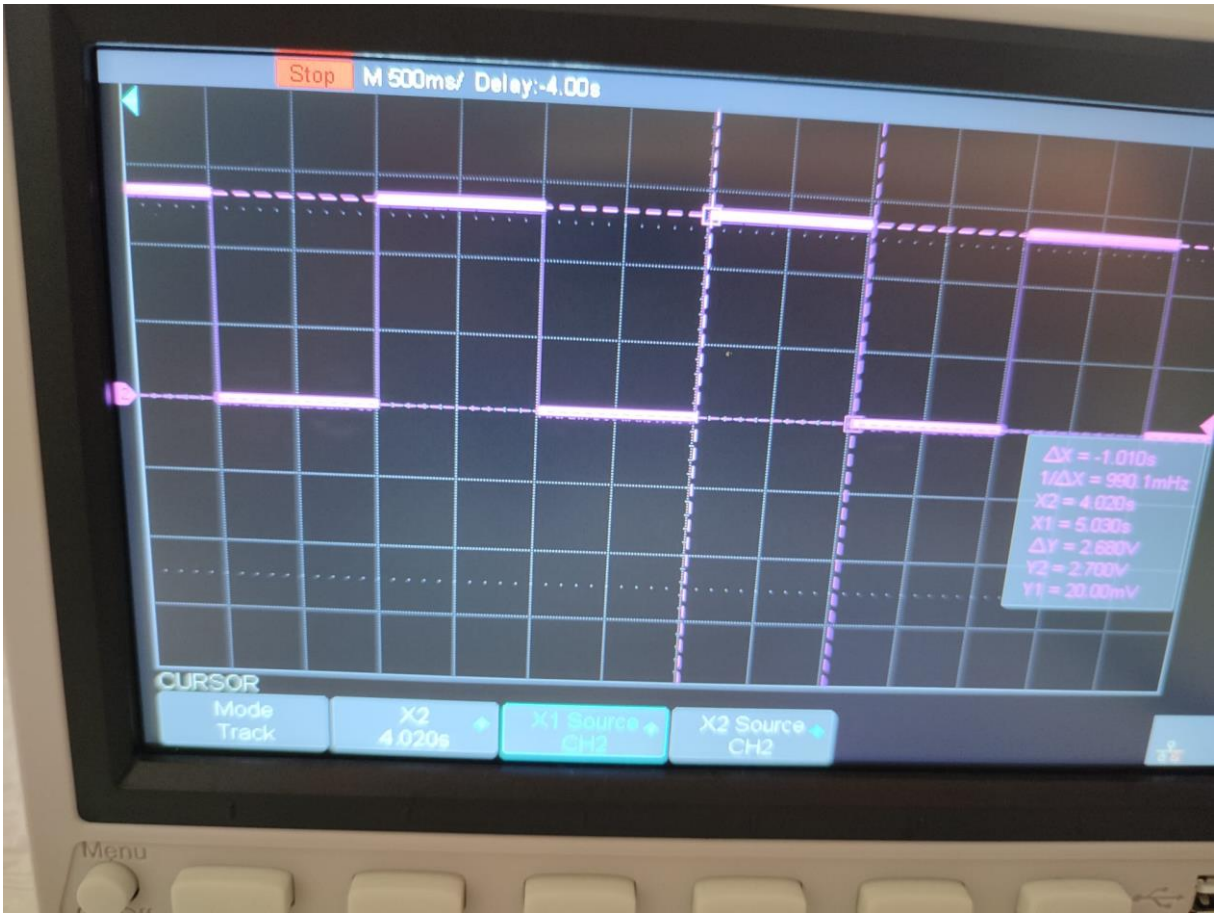

Figure 3: Led on

- Yes, there is obivious difference between assembly and c code because you need to arrange all moder, odr in assembly but in c you need to add library instead of these arragements.
- Yes, there is difference delay number. Because you need to calculate instructions delay in assembly. But in c programming it is enough to write the delay time to code.
- In normally the code give us warnings which is written roughly. But there is no difference between -o2 none in our code.

Problem 2:

Using **a state machine** blink the external LED at different intervals. Assign each speed to a mode, and attach a button to cycle through the modes. (Each button press will cycle through these modes.) You should do polling for the button press.

Modes:

- Mode 0 → No toggling, LED is off
- Mode 1 → LED is toggling at roughly 2 second intervals
- Mode 2 → LED is toggling at roughly 1 second intervals
- Mode 3 → LED is toggling at roughly .5 second intervals
- Mode 4 → LED is toggling at roughly .1 second intervals
- Mode 5 → No toggling, LED is on

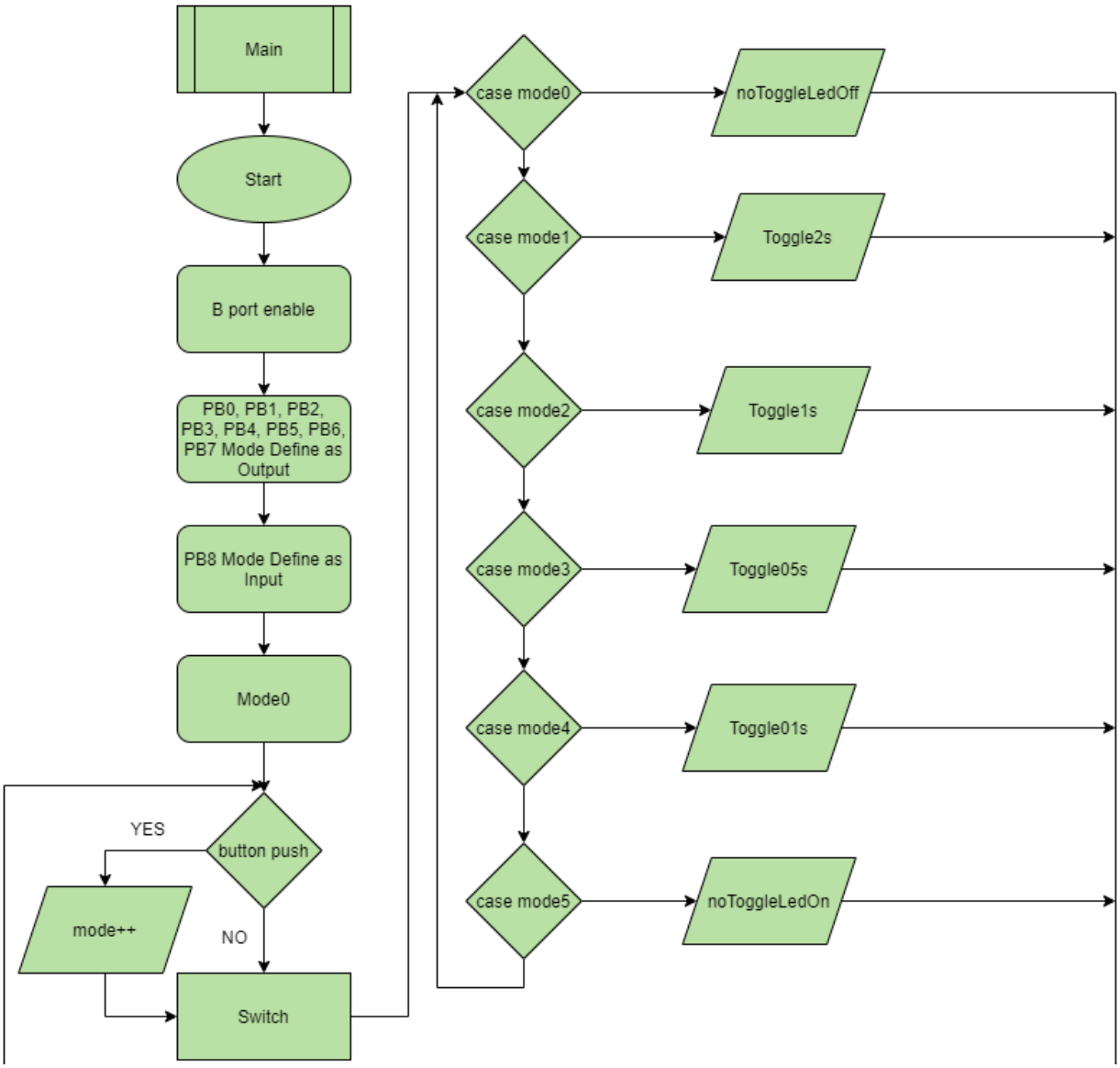

Figure 4: Problem 2 Flowchart

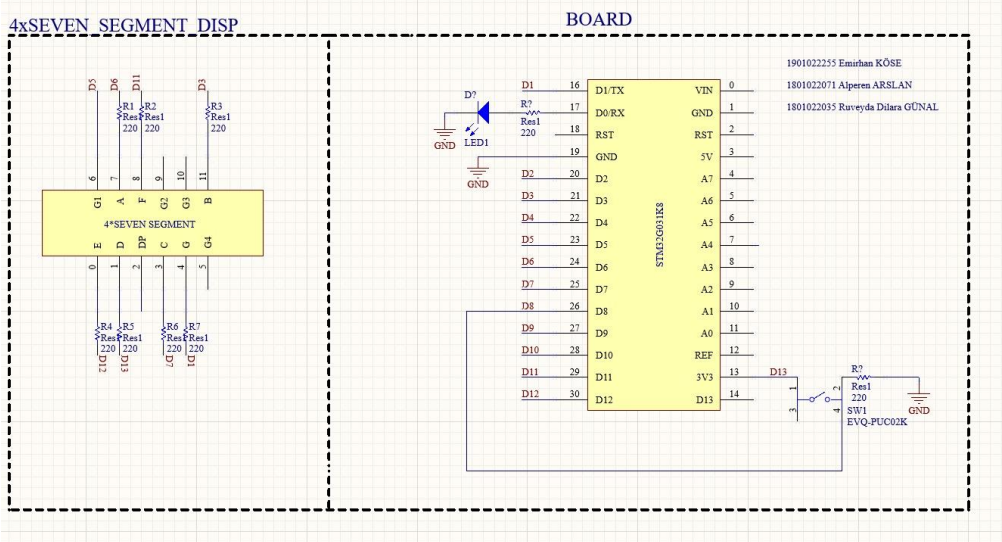

Figure 5: Problem 2 Block Diagram

```
//main.c
//
//Author: Alperen Arslan, Emirhan Köse, Ruveyda Dilara Günal
//Description: Changes the mode of the LED by pressing the button.
//At the same time, it shows which mode it is in with a 7-segment
display.

#include "stm32g0xx.h"

void delay(volatile uint32_t);
void noToggleLedOff();
void noToggleLedOn();
void Toggle2s();
void Toggle1s();
void Toggle05s();
void Toggle01s();
void display0mode();
void display1mode();
void display2mode();
void display3mode();
void display4mode();
void display5mode();

int main(void) {
    // Enable GPIOB clock
    RCC->IOPENR |= (1U << 1);

    // Setup PA9, PA10, PA15, PB0, PB1, PB4, PB6, PB7  as output
and PB5 as input
```

```
    GPIOB->MODER &= 0xFFFC5555;

    //Enum for change between modes
    enum changemode {mode0, mode1, mode2, mode3, mode4, mode5}mode;
    mode = mode0;

    while(1){
     int value = GPIOB->IDR &= (1U << 8);
         if (value == 0x100){
               if(mode == mode5) //If code at the last mode,
change to first mode
                     mode = mode0;
               else
                     mode ++; //Change mode
          }
     switch(mode){
          case mode0:
               display0mode(); //Display current mode to see mode
at the 7 segment
               noToggleLedOff(mode); //Set mode
               break;
          case mode1:
               display1mode();
               Toggle2s(mode);
               break;
          case mode2:
               display2mode();
               Toggle1s(mode);
               break;
          case mode3:
               display3mode();
               Toggle05s(mode);
               break;
          case mode4:
               display4mode();
               Toggle01s(mode);
               break;
          case mode5:
               display5mode();
               noToggleLedOn(mode);
               break;
     }
     delay(600000); //Delay for getting hand back from button
    }
}

void noToggleLedOff(){
     GPIOB->ODR &= ~(1U << 7);
}
```

```
void noToggleLedOn(){
	GPIOB->ODR |= (1U << 7);
}

void Toggle2s(){
	GPIOB->ODR |= (1U << 7);
	delay(8000000);
	GPIOB->ODR &= ~(1U << 7);
	delay(8000000);
}

void Toggle1s(){
	GPIOB->ODR |= (1U << 7);
	delay(4000000);
	GPIOB->ODR &= ~(1U << 7);
	delay(4000000);
}

void Toggle05s(){
	GPIOB->ODR |= (1U << 7);
	delay(2000000);
	GPIOB->ODR &= ~(1U << 7);
	delay(2000000);
}

void Toggle01s(){
	GPIOB->ODR |= (1U << 7);
	delay(400000);
	GPIOB->ODR &= ~(1U << 7);
	delay(400000);
}

void display0mode(){
	GPIOB->ODR &= 0x0000;
	GPIOB->ODR |= 0x003F;
}

void display1mode(){
	GPIOB->ODR &= 0x0000;
	GPIOB->ODR |= 0x0006;
}

void display2mode(){
	GPIOB->ODR &= 0x0000;
	GPIOB->ODR |= 0x005B;
}

void display3mode(){
```

```
      GPIOB->ODR &= 0x0000;
      GPIOB->ODR |= 0x004F;
}

void display4mode(){
      GPIOB->ODR &= 0x0000;
      GPIOB->ODR |= 0x0066;
}

void display5mode(){
      GPIOB->ODR &= 0x0000;
      GPIOB->ODR |= 0x006D;
}

void delay(volatile uint32_t s) {
    for(; s>0; s--);
}
```

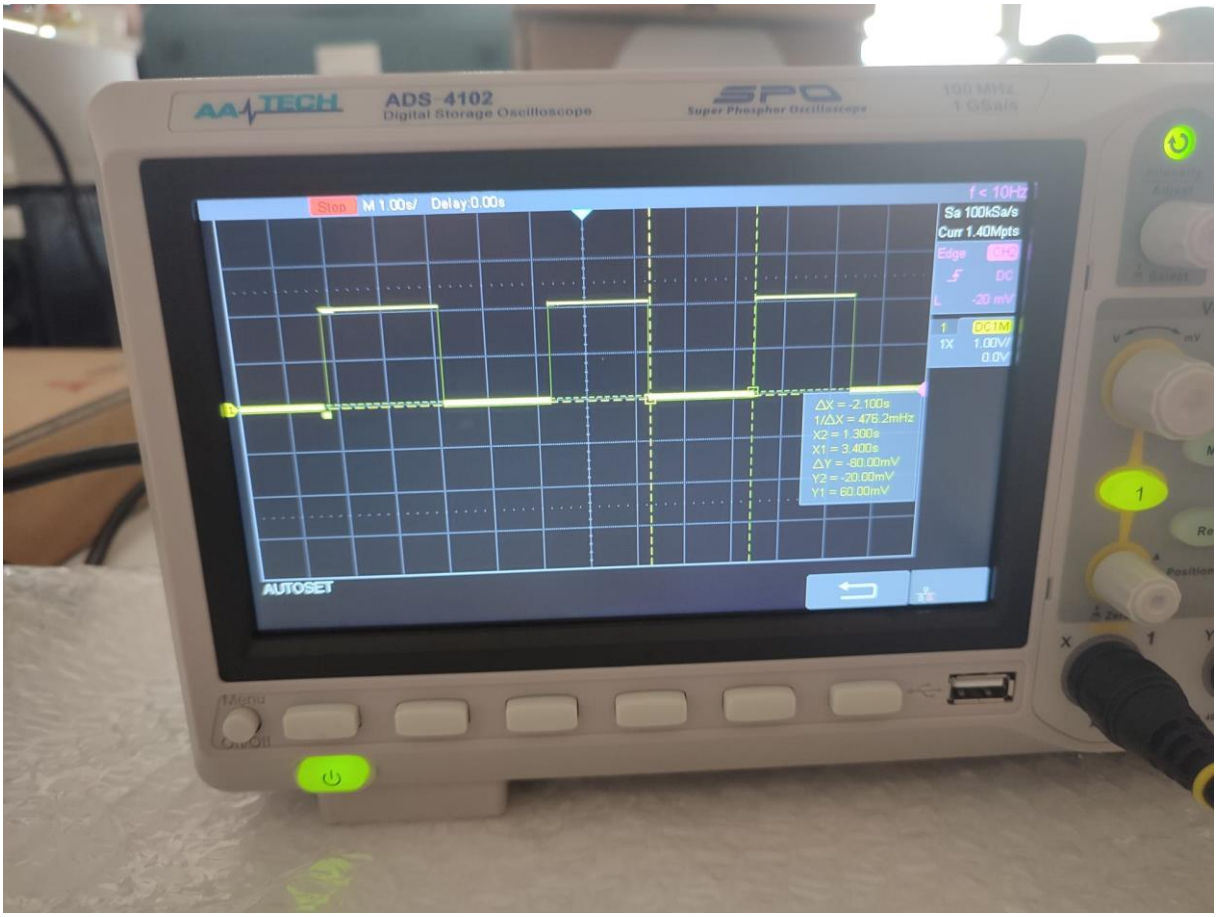

Figure 6: 2 seconds interval led off

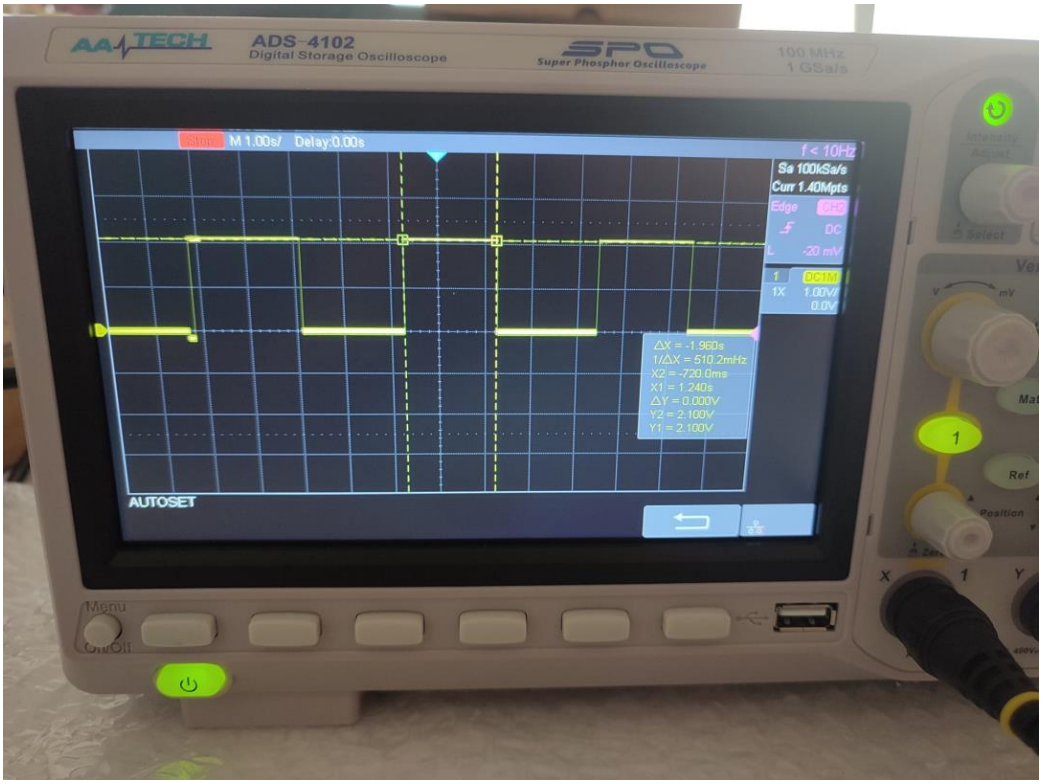

Figure 7: 2 seconds interval led on

Figure 8: 1 seconds interval led on

Figure 9: 1 seconds interval led off

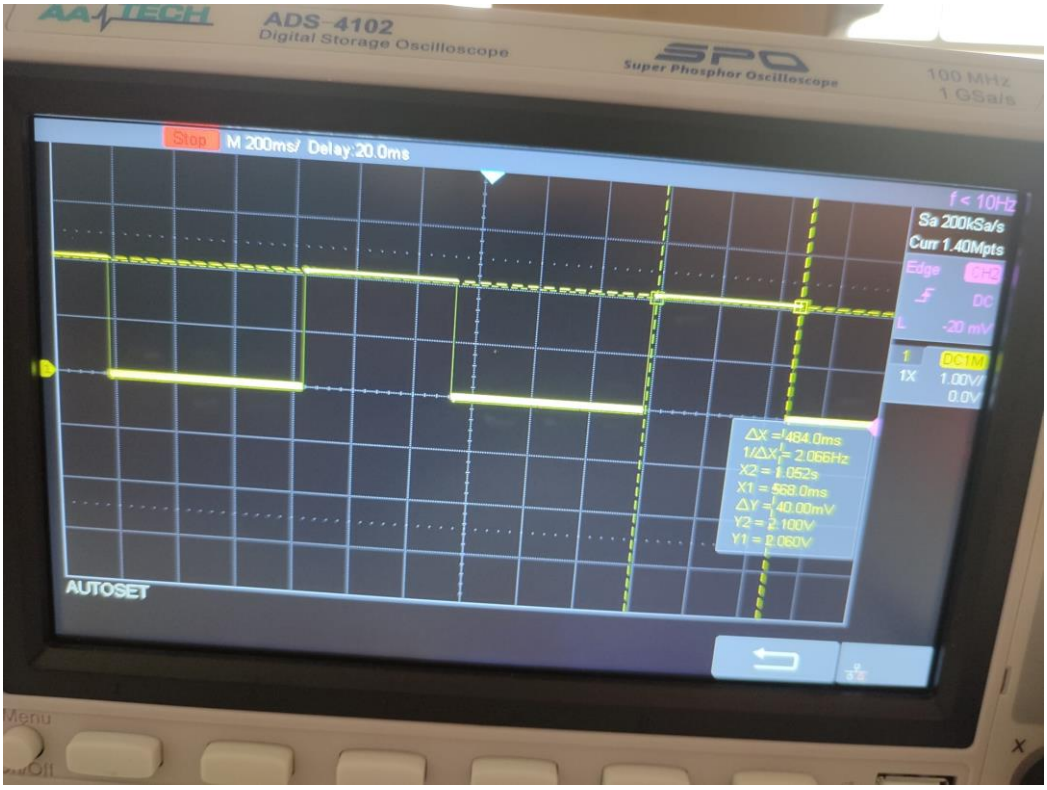

Figure 10: 0.5 seconds interval led on

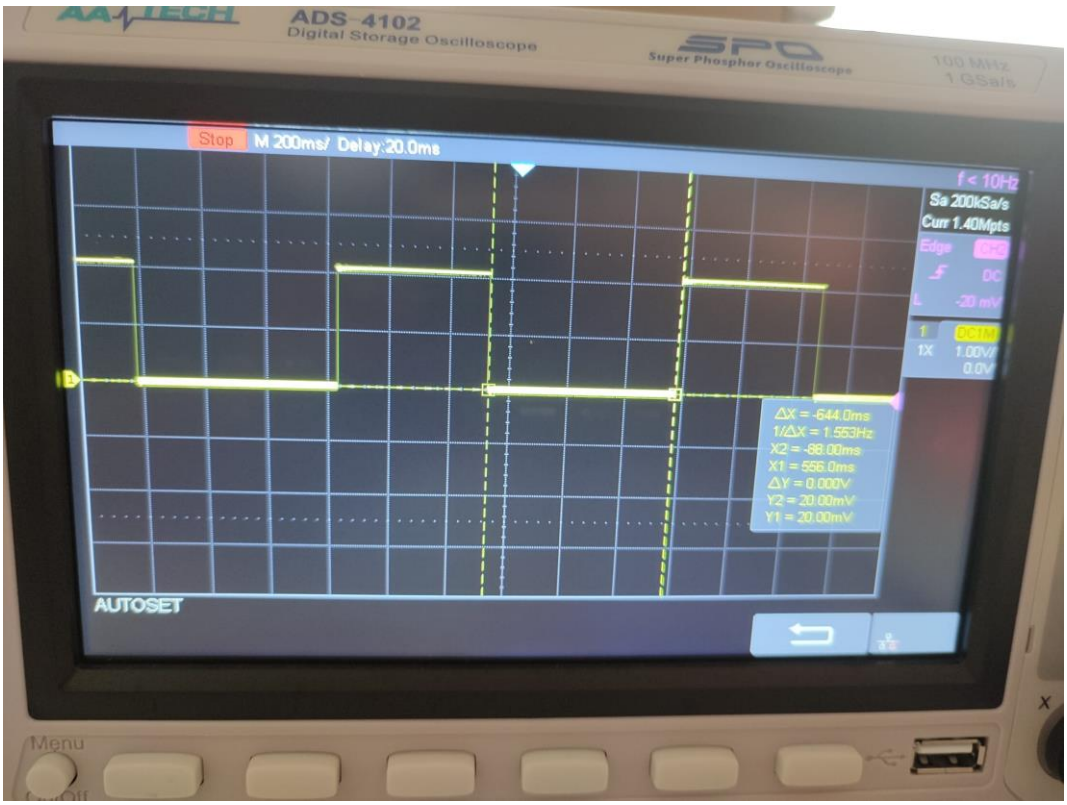

Figure 11: 0.5 seconds interval led off

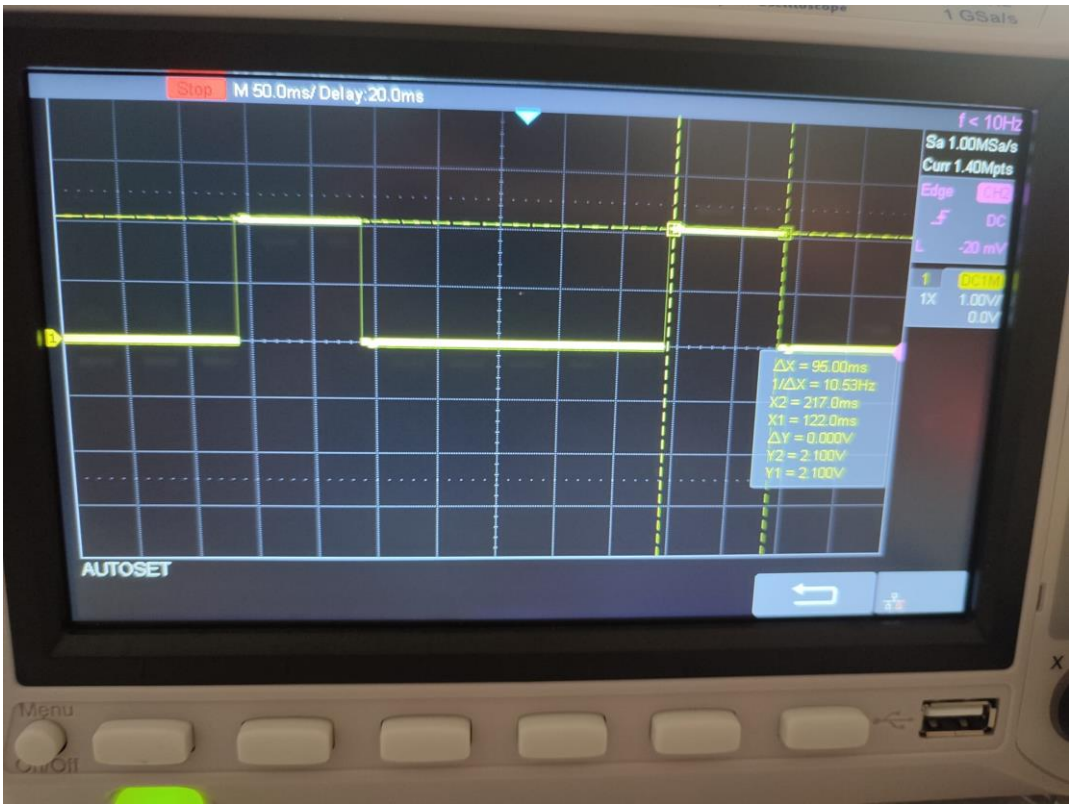

Figure 12: 0.1 seconds interval led on

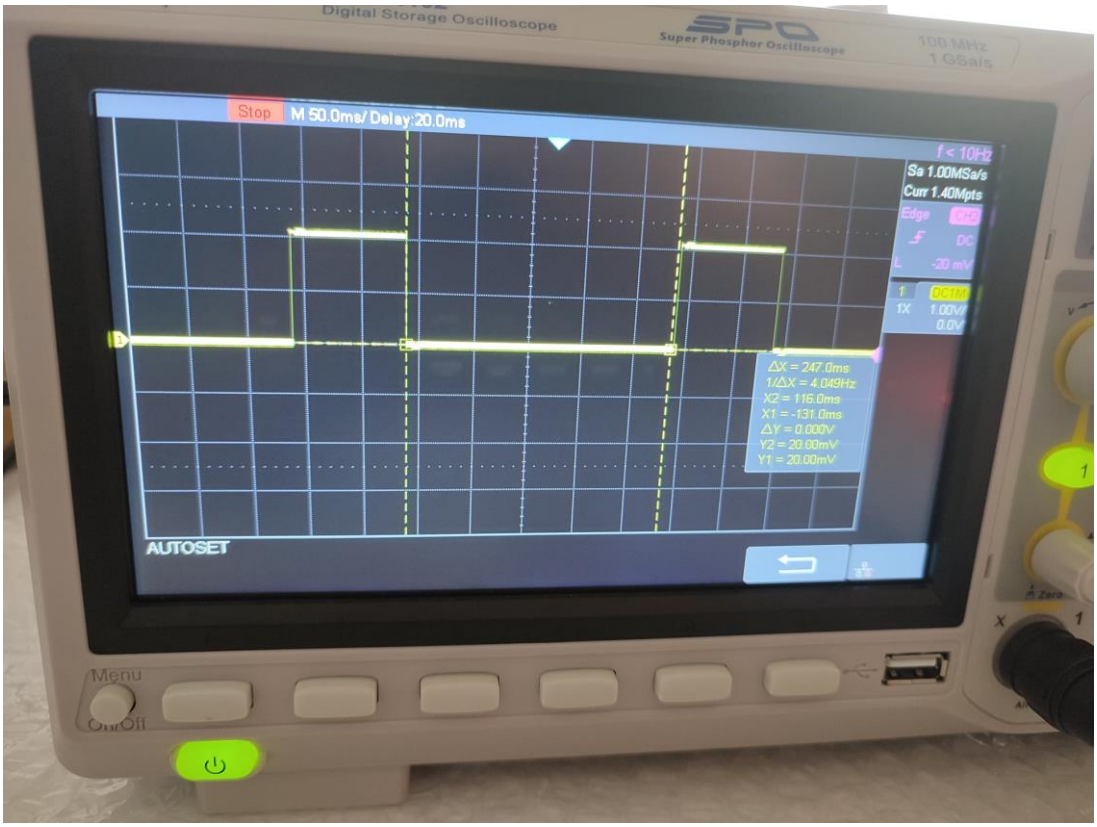

Figure 13: 0.1 seconds interval led off

Questions:

● What is the difference in code size when the optimization is enabled / disabled? How about the actual blinking speed of the LED? Is there any change? If so, what would be the difference?

● Compare the state machine approach to a regular super loop approach. What are the advantages / disadvantages? You need to give a pseudo-code for this comparison.

In normally the code give us warnings which is written roughly. But there is no difference between -o2 none in our code.

Problem 3:

Implement the same state machine in Problem 2, but this time use external interrupts to detect button press, and use the handler to change the state.

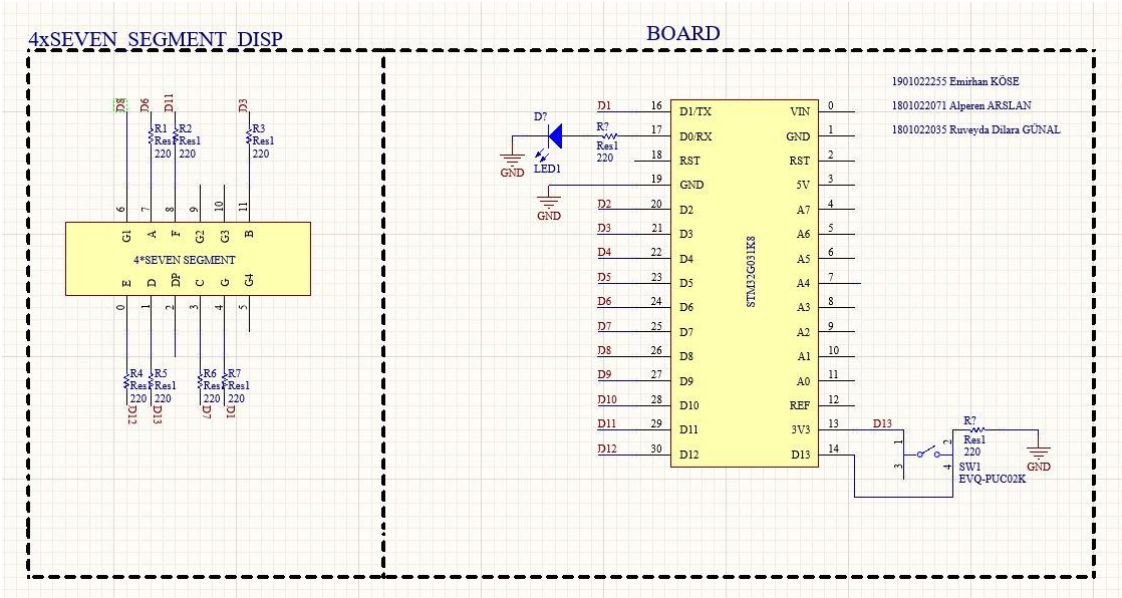

Figure 14: Problem 3 Block Diagram

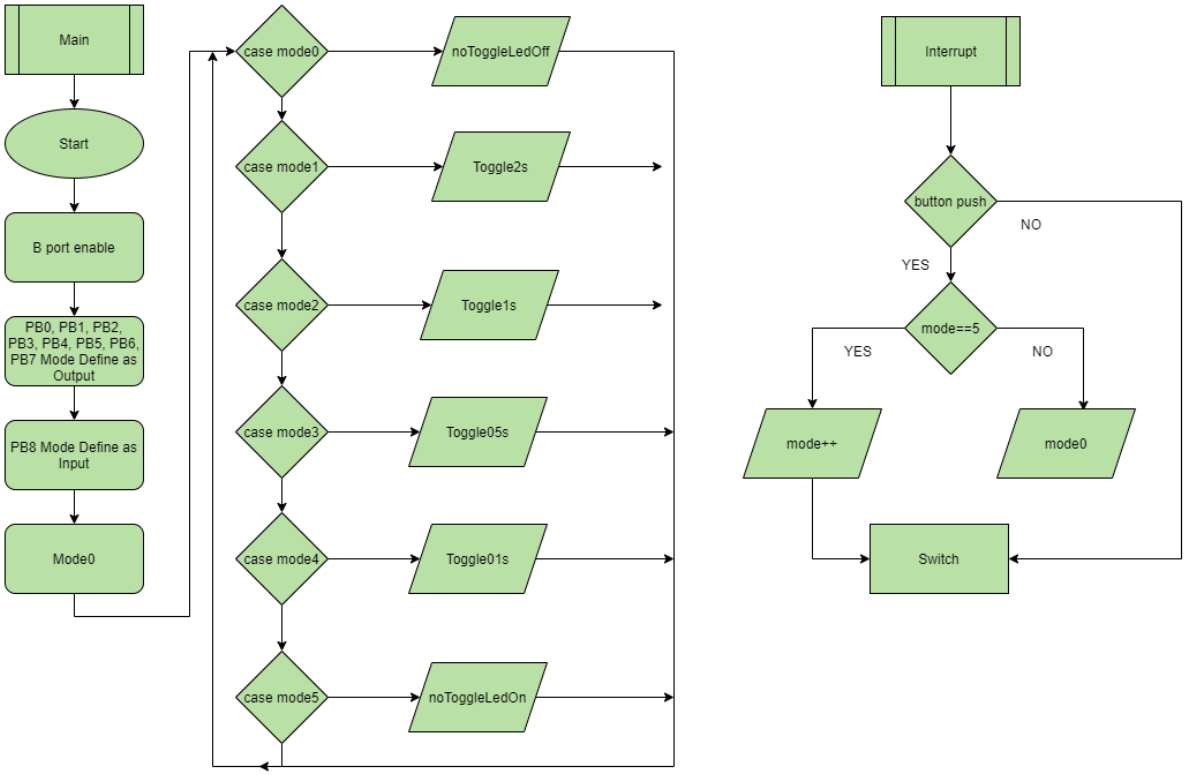

Figure 15: Problem 3 Flowchart

```
//main.c
//
//Author: Alperen Arslan, Emirhan Köse, Ruveyda Dilara Günal
//Description: Changes the mode of the LED by pressing the button.
//At the same time, it shows which mode it is in with a 7-segment
display.
//External Interrupt Used

#include "stm32g0xx.h"

void delay(volatile uint32_t);
void noToggleLedOff();
void noToggleLedOn();
void Toggle2s();
void Toggle1s();
void Toggle05s();
void Toggle01s();
void display0mode();
void display1mode();
void display2mode();
void display3mode();
void display4mode();
void display5mode();

int main(void) {
```

```
    // Enable GPIOB clock
    RCC->IOPENR |= (1U << 1);

    // Setup PA9, PA10, PA15, PB0, PB1, PB4, PB6, PB7  as output
and PB5 as input
    GPIOB->MODER &= 0xFFFC5555;
    GPIOB->PUPDR |= (2U << 2*8);

    //Enum for change between modes
    enum changemode {mode0, mode1, mode2, mode3, mode4, mode5}mode;
    mode = mode0;

    EXTI->EXTICR[2] |= (1U << 8*0); //PB8 3. Mux
    EXTI->RTSR1 |= (1U << 8); //Rising Edge PB8
    EXTI->IMR1 |= (1U << 8); //Mask for PB8
    NVIC_SetPriority(EXTI4_15_IRQn, 0);
    NVIC_EnableIRQ(EXTI4_15_IRQn);

    while(1){
      switch(mode){
            case mode0:
                  display0mode(); //Display current mode to see mode
at the 7 segment
                noToggleLedOff(mode); //Set mode
                break;
            case mode1:
                  display1mode();
                  noToggleLedOn(mode);
                break;
            case mode2:
                  display2mode();
                  Toggle2s(mode);
                  break;
            case mode3:
                display3mode();
                Toggle1s(mode);
                break;
            case mode4:
                  display4mode();
                  Toggle05s(mode);
                break;
            case mode5:
                  display5mode();
                  Toggle01s(mode);
                  break;
      }
      //delay(600000); //Delay for getting hand back from button
    }
}
```

```
void EXTI4_15_IRQHandler(int mode){
          if(mode == 5) //If code at the last mode, change to
first mode
                mode = 0;
         else
          mode ++; //Change mode
          EXTI->RPR1 |= (1U << 8);
}

void noToggleLedOff(){
     GPIOB->ODR &= ~(1U << 7);
}

void noToggleLedOn(){
     GPIOB->ODR |= (1U << 7);
}

void Toggle2s(){
     GPIOB->ODR |= (1U << 7);
     delay(3200000);
     GPIOB->ODR &= ~(1U << 7);
     delay(3200000);
}

void Toggle1s(){
     GPIOB->ODR |= (1U << 7);
     delay(1600000);
     GPIOB->ODR &= ~(1U << 7);
     delay(1600000);
}

void Toggle05s(){
     GPIOB->ODR |= (1U << 7);
     delay(800000);
     GPIOB->ODR &= ~(1U << 7);
     delay(800000);
}

void Toggle01s(){
     GPIOB->ODR |= (1U << 7);
     delay(160000);
     GPIOB->ODR &= ~(1U << 7);
     delay(160000);
}

void display0mode(){
     GPIOB->ODR &= 0x0000;
     GPIOB->ODR |= 0x003F;
```

```
}

void display1mode(){
      GPIOB->ODR &= 0x0000;
      GPIOB->ODR |= 0x0006;
}

void display2mode(){
      GPIOB->ODR &= 0x0000;
      GPIOB->ODR |= 0x005B;
}

void display3mode(){
      GPIOB->ODR &= 0x0000;
      GPIOB->ODR |= 0x004F;
}

void display4mode(){
      GPIOB->ODR &= 0x0000;
      GPIOB->ODR |= 0x0066;
}

void display5mode(){
      GPIOB->ODR &= 0x0000;
      GPIOB->ODR |= 0x006D;
}

void delay(volatile uint32_t s) {
    for(; s>0; s--);
}
```

Questions:

● What is the difference between Problem 2 and Problem 3 in terms of scalability, clarity and responsiveness? Compare the oscilloscope outputs for both of them and explain.

Problem 4:

Connect the keypad to the microcontroller, and using external interrupts detect button presses. Use an SSD to display the pressed button. Your main loop should only be used to display the SSDs.

Requirements:

● You should create a function that will display the given number on SSDs.

● Each time button is pressed, the number should slide in from the right of the SSD.

● When there are 4 digits already, the next number should erase the oldest number.

- For example, when you pressed the numbers 4, 2, 5, 7, 6 in that order, the SSD should display 4 → 42 → 425 → 4257 → 2576.

● There should be almost no difference in brightness between SSDs.

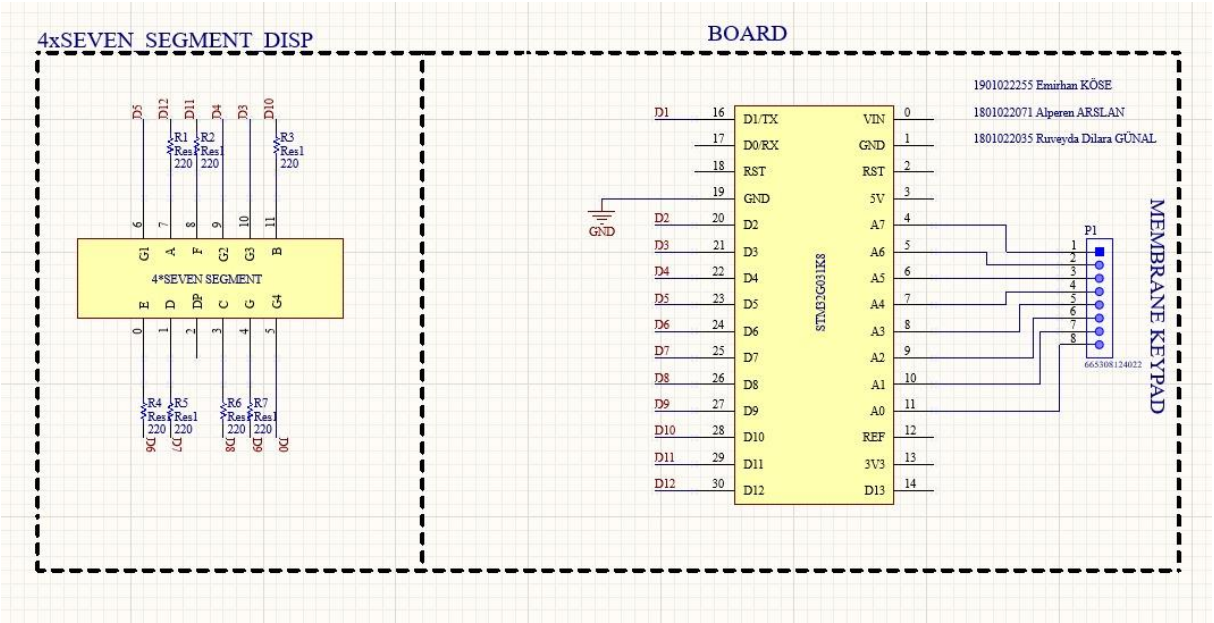

Figure 16: Problem 4 Bloack Diagram

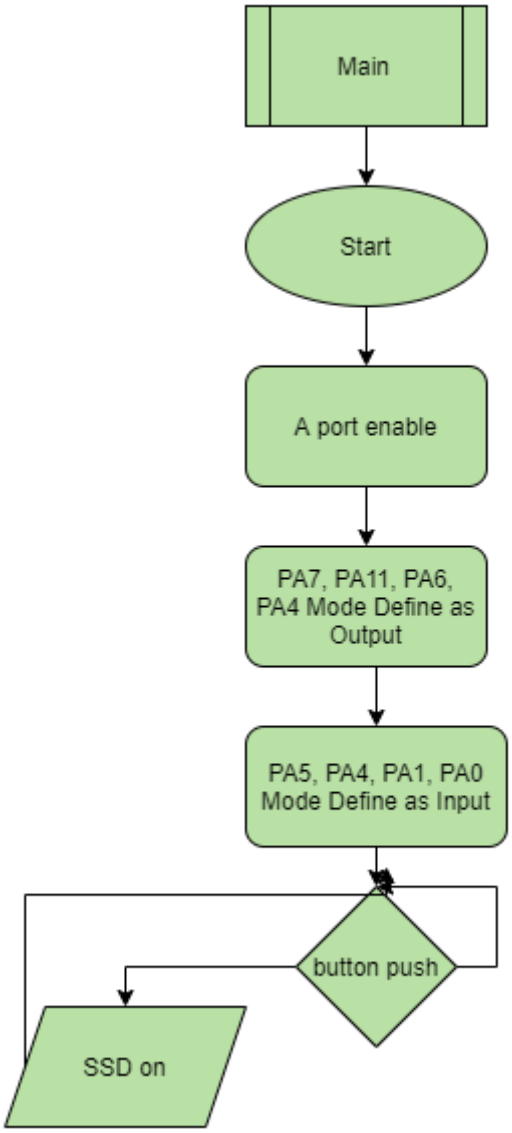

Figure 17: Problem 4 Flowchart

```
/*
 * main.c
 *
 * author: emirhan köse
 *
 * description: Blinks 1 on-board LED at roughly 1 second
intervals. system
 *   clock is running from HSI which is 16 Mhz. Delay function is
just a simple
 *   counter so is not accurate and the delay time will change
based on the
 *   optimization flags.
 */
```

```c
#include "stm32g0xx.h"

void clearSSD(void);
void clearRowsKeypad(void);
void setRowsKeypad(void);
void setSSD(int);

void EXTI0_1_IRQHandler(void){ //idr number 0 and 1 pins  c3 and c4
and no need to do c4 because we don't use it
	clearRowsKeypad();

	GPIOA->ODR ^=(1U<<7); //R1 high

	if((GPIOA->IDR>>1)&1){
		clearRowsKeypad();
		setSSD(3);
	}
	GPIOA->ODR ^=(1U<<7);

	GPIOA->ODR ^=(1U<<11);
	if((GPIOA->IDR>>1)&1){
		clearRowsKeypad();
		setSSD(6);
	}
	GPIOA->ODR ^=(1U<<11);

	GPIOA->ODR ^=(1U<<6);
	if((GPIOA->IDR>>1)&1){
		clearRowsKeypad();
			setSSD(9);
		}
	GPIOA->ODR ^=(1U<<6);

	EXTI->RPR1 &=0xFF; //clear interrupt flag

	setRowsKeypad();
}
void EXTI2_3_IRQHandler(void){}


void EXTI4_15_IRQHandler(void){

	clearRowsKeypad();

	GPIOA->ODR ^=(1U<<7);

	if((GPIOA->IDR>>2)&1){
		clearRowsKeypad();
		setSSD(2);
```

```
	}
	else if((GPIOA->IDR>>3)&1){
		clearRowsKeypad();
		setSSD(1);
	}
	GPIOA->ODR ^=(1U<<7);


	GPIOA->ODR ^=(1U<<11);
	if((GPIOA->IDR>>2)&1){
		clearRowsKeypad();
		setSSD(5);
	}
	else if((GPIOA->IDR>>3)&1){
		clearRowsKeypad();
			setSSD(3);
		}
	GPIOA->ODR ^=(1U<<11);


	GPIOA->ODR ^=(1U<<6);
	if((GPIOA->IDR>>2)&1){
		clearRowsKeypad();
		setSSD(8);
	}
	else if((GPIOA->IDR>>3)&1){
		clearRowsKeypad();
				setSSD(7);
			}
	GPIOA->ODR ^=(1U<<6);

	GPIOA->ODR ^=(1U<<12);
	if((GPIOA->IDR>>2)&1){
		clearRowsKeypad();
		setSSD(0);
	}
	GPIOA->ODR ^=(1U<<12);

	EXTI->RPR1 &=0xFF; //clear interrupt flag
	setRowsKeypad();
}

int main(void) {
RCC->IOPENR |=(3U<<0);


GPIOA->MODER &=~(3U<<0);
GPIOA->PUPDR |=(2U<<0);
```

```
GPIOA->MODER &=~(3U<<2);
GPIOA->PUPDR |=(2U<<2);

GPIOA->MODER &=~(3U<<2*4);
GPIOA->PUPDR |=(2U<<2*4);

GPIOA->MODER &=~(3U<<2*5);
GPIOA->PUPDR |=(2U<<2*5);

////////////////////////////////////////////

//OUTPUT MODER

GPIOA->MODER &=~(3U<<2*6); //PA 6 is adjusted as output
GPIOA->MODER |=(1U<<2*6);

GPIOA->MODER &=~(3U<<2*7); //PA 7 is adjusted as output
GPIOA->MODER |=(1<<2*7);

GPIOA->MODER &=~(3U<<2*8); //PA 8 is adjusted as output
GPIOA->MODER |=(1U<<2*8);

GPIOA->MODER &=~(3U<<2*9); //PA 9 is adjusted as output
GPIOA->MODER |=(1U<<2*9);

GPIOA->MODER &=~(3U<<2*10); //PA 10 is adjusted as output
GPIOA->MODER |=(1U<<2*10);

GPIOA->MODER &=~(3U<<2*11); //PA 11 is adjusted as output
GPIOA->MODER |=(1U<<2*11);

GPIOA->MODER &=~(3U<<2*12); //PA 12 is adjusted as output
GPIOA->MODER |=(1U<<2*12);

GPIOB->MODER &=~(3U<<2*0); //PB 0 is adjusted as output
GPIOB->MODER |=(1U<<2*0);

GPIOB->MODER &=~(3U<<2*1); //PB 1 is adjusted as output
GPIOB->MODER |=(1U<<2*1);

GPIOB->MODER &=~(3U<<2*2); //PB 2 is adjusted as output
GPIOB->MODER |=(1U<<2*2);

GPIOB->MODER &=~(3U<<2*4); //PB 4 is adjusted as output
GPIOB->MODER |=(1U<<2*4);

GPIOB->MODER &=~(3U<<2*5); //PB 5 is adjusted as output
GPIOB->MODER |=(1U<<2*5);
```

```
GPIOB->MODER &=~(3U<<2*7); //PB 7 is adjusted as output
GPIOB->MODER |=(1U<<2*7);

GPIOB->MODER &=~(3U<<2*8); //PB 8 is adjusted as output
GPIOB->MODER |=(1U<<2*8);

GPIOB->MODER &=~(3U<<2*9); //PB 9 is adjusted as output
GPIOB->MODER |=(1U<<2*9);

// KEYPAD'S INPUTS INTERRUPT (input pins 6 7 11 12)
EXTI->EXTICR[1] |= (0U<<8*0); //PA0
EXTI->EXTICR[1] |= (0U<<8*1); //PA1
EXTI->EXTICR[2] |= (0U<<8*0); //PA4
EXTI->EXTICR[2] |= (0U<<8*1); //PA5


// rising edge --> is it passing 0 to 1 or 1 to 0 ?

EXTI->RTSR1 |=(1U<<0);
EXTI->RTSR1 |=(1U<<1);
EXTI->RTSR1 |=(1U<<4);
EXTI->RTSR1 |=(1U<<5);

//MASK

EXTI->IMR1 |=(1U<<0);
EXTI->IMR1 |=(1U<<1);
EXTI->IMR1 |=(1U<<4);
EXTI->IMR1 |=(1U<<5);

/NVIC/
NVIC_SetPriority(EXTI0_1_IRQn, 0);
NVIC_EnableIRQ(EXTI0_1_IRQn);

NVIC_SetPriority(EXTI2_3_IRQn, 0);
NVIC_EnableIRQ(EXTI2_3_IRQn);

NVIC_SetPriority(EXTI4_15_IRQn, 0);
NVIC_EnableIRQ(EXTI4_15_IRQn);

//set all rows

GPIOA->ODR |=(1U<<7);  //R1 is high
GPIOA->ODR |=(1U<<11); //R2 is high
GPIOA->ODR |=(1U<<6);  //R3 is high
GPIOA->ODR |=(1U<<12); //R4 is high

clearSSD();
```

```
while(1){

}

return 0;
}

void clearSSD(){
	GPIOB->ODR &=~(1U<<4); //PB 4 A
	GPIOB->ODR &=~(1U<<9); //PB 9 B
	GPIOB->ODR &=~(1U<<8); //PB 8 C
	GPIOB->ODR &=~(1U<<2); //PB 2 D
	GPIOB->ODR &=~(1U<<0); //PB 0 E
	GPIOB->ODR &=~(1U<<5); //PB 5 F
	GPIOA->ODR &=~(1U<<8); //PA 8 G
}

void setSSD(int x){
	clearSSD();

	switch(x){

	case 0:
		GPIOB->ODR |=(1U<<4); //PB 4 A
		GPIOB->ODR |=(1U<<9); //PB 9 B
		GPIOB->ODR |=(1U<<8); //PB 8 C
		GPIOB->ODR |=(1U<<2); //PB 2 D
		GPIOB->ODR |=(1U<<0); //PB 0 E
		GPIOB->ODR |=(1U<<5); //PB 5 F
		break;
	case 1:
		GPIOB->ODR |=(1U<<9); //PB 9 B
		GPIOB->ODR |=(1U<<8); //PB 8 C
	    break;
	case 2:
		GPIOB->ODR |=(1U<<4); //PB 4 A
		GPIOB->ODR |=(1U<<9); //PB 9 B
		GPIOA->ODR |=(1U<<8); //PA 8 G
		GPIOB->ODR |=(1U<<0); //PB 0 E
		GPIOB->ODR |=(1U<<2); //PB 2 D
		break;
	case 3:
		GPIOB->ODR |=(1U<<4); //PB 4 A
		GPIOB->ODR |=(1U<<9); //PB 9 B
		GPIOA->ODR |=(1U<<8); //PA 8 G
		GPIOB->ODR |=(1U<<8); //PB 8 C
		GPIOB->ODR |=(1U<<2); //PB 2 D
		break;
	case 4:
```

```
            GPIOB->ODR |=(1U<<5); //PB 5 F
            GPIOA->ODR |=(1U<<8); //PA 8 G
            GPIOB->ODR |=(1U<<9); //PB 9 B
            GPIOB->ODR |=(1U<<8); //PB 8 C
            break;
      case 5:
            GPIOB->ODR |=(1U<<4); //PB 4 A
            GPIOB->ODR |=(1U<<5); //PB 5 F
            GPIOA->ODR |=(1U<<8); //PA 8 G
            GPIOB->ODR |=(1U<<8); //PB 8 C
            GPIOB->ODR |=(1U<<2); //PB 2 D
            break;
      case 6:
            GPIOB->ODR |=(1U<<4); //PB 4 A
            GPIOB->ODR |=(1U<<5); //PB 5 F
            GPIOA->ODR |=(1U<<8); //PA 8 G
            GPIOB->ODR |=(1U<<8); //PB 8 C
            GPIOB->ODR |=(1U<<2); //PB 2 D
            GPIOB->ODR |=(1U<<0); //PB 0 E
            break;
      case 7:
            GPIOB->ODR |=(1U<<4); //PB 4 A
            GPIOB->ODR |=(1U<<9); //PB 9 B
            GPIOB->ODR |=(1U<<8); //PB 8 C
            break;
      case 8:
            GPIOB->ODR |=(1U<<4); //PB 4 A
            GPIOB->ODR |=(1U<<9); //PB 9 B
            GPIOB->ODR |=(1U<<8); //PB 8 C
            GPIOB->ODR |=(1U<<2); //PB 2 D
            GPIOB->ODR |=(1U<<0); //PB 0 E
            GPIOB->ODR |=(1U<<5); //PB 5 F
            GPIOA->ODR |=(1U<<8); //PA 8 G
            break;
      case 9:
            GPIOB->ODR |=(1U<<4); //PB 4 A
            GPIOB->ODR |=(1U<<9); //PB 9 B
            GPIOB->ODR |=(1U<<8); //PB 8 C
            GPIOB->ODR |=(1U<<2); //PB 2 D
            GPIOB->ODR |=(1U<<5); //PB 5 F
            GPIOA->ODR |=(1U<<8); //PA 8 G
            break;
      }
}

void clearRowsKeypad(){
      GPIOA->ODR &=(0U<<6);  //PB 6
      GPIOA->ODR &=(0U<<11); //PB 11
      GPIOA->ODR &=(0U<<12); //PB12
```

```
      GPIOA->ODR &=(0U<<7); //PB7
}

void setRowsKeypad(){
      GPIOA->ODR |=(1U<<6);  //PB 6
      GPIOA->ODR |=(1U<<11); //PB 11
      GPIOA->ODR |=(1U<<12); //PB12
      GPIOA->ODR |=(1U<<7); //PB7
}
```

Questions:

● Try to figure out the processing delay of the interrupt looking at the scope output.

● Is there a brightness difference between the numbers Seven Segments? How did you solve it? Show the scope output of a single segment when you light up the same segment on all Seven Segments. What happens if you decrease the delay / increase the delay?