GEBZE TECHNICAL UNIVERSITY

ENGINEERING FACULTY

ELECTRONICS ENGINEERING

ELEC 334

MICROPROCESSORS

PROJECT #1

Name - Surname	Yağmur DERYA
Student ID	171024011

1. Introduction

In this project, the goal is to implement a randomized counter in Assembly which will generate a random number when an external button pressed. When the counting ends, a LED will be turned off. The 4-digit numbers will be shown in a 4-digit seven segment display and when the system is idle, last 4-digits of the school ID will be shown. If the is pressed while counting, the counting will be paused and if it is pressed again, counting will be resumed.

2. Project

Parts list:

- Breadboard
- Nucleo-G031K8
- LED
- Jumper
- 4-Digit Seven Segment Display
- Push Button
- Resistor
- Micro USB Cable

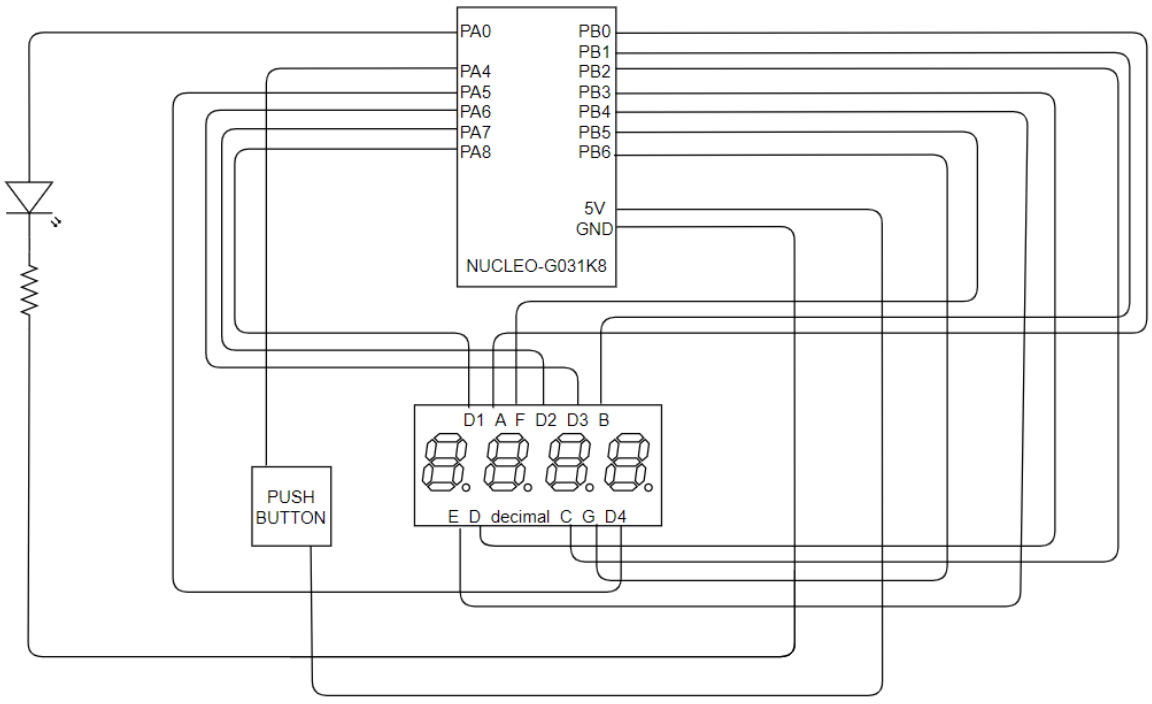

Figure 1. Block diagram.

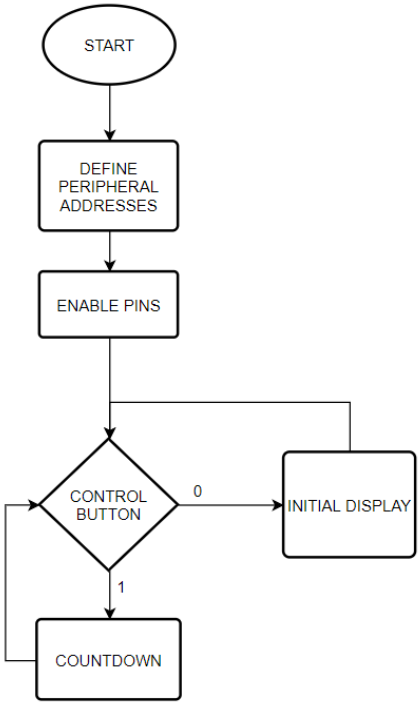

Figure 2. Flowchart.

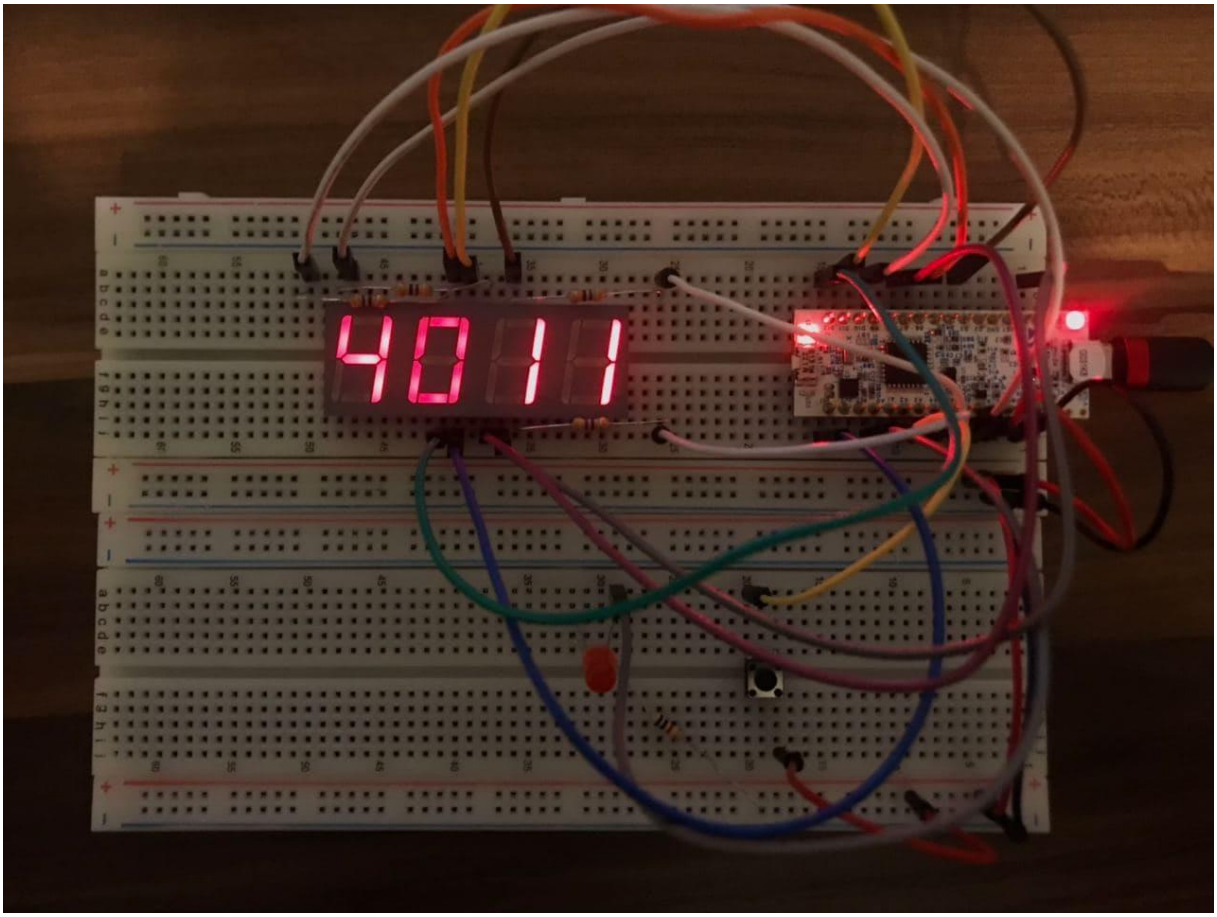

Figure 3.a. Project setup.

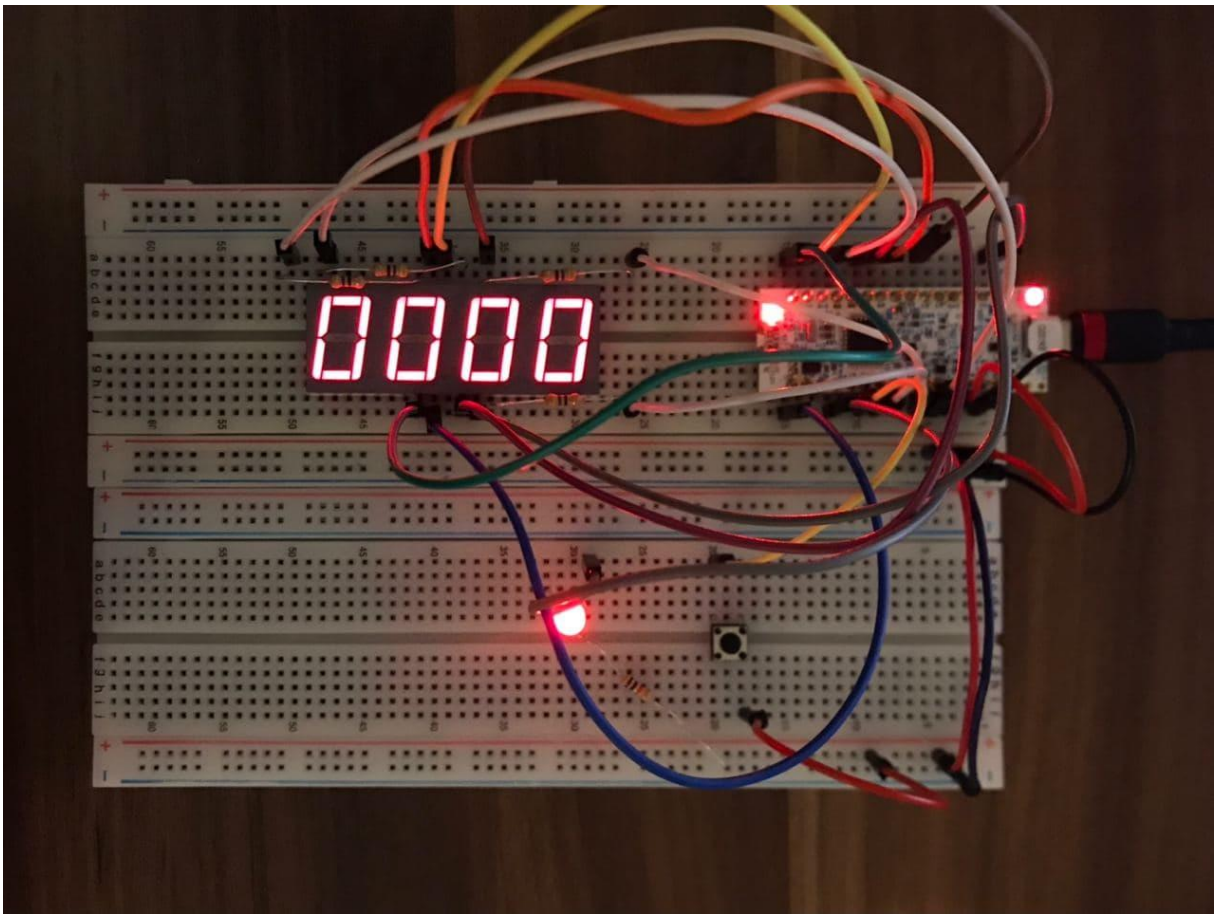

Figure 3.b. Project setup.

2.1. Tasks

1. Turn on 1 digit of the seven segment display.
2. Implement assembly code to display numbers.
3. Check each number.
   In this part, I recognized I need to clear data every time after display a number because since it writes on old data, numbers can not be displayed correctly.
4. Turn on all digits.
5. Try to display different numbers on digits.
   In this part, I discovered that to display different numbers, I have to turn on that digit while others off, show number with a pretty small delay and turn of that digit and turn on next digit and do the same things again.
   (Turn on digit > display > wait>turn of digit> turn on other digit>display…)
6. Add button.
7. Implement assembly code to start countdown when button = 1.
8. Add LED.
9. Implement assembly code to turn on LED when countdown ends.

2.2. Code

```
/* 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 offset

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

.equ GPIOB_BASE,       (0x50000400)          // GPIOB base address
.equ GPIOB_MODER,      (GPIOB_BASE + (0x00)) // GPIOB MODER offset
.equ GPIOB_ODR,        (GPIOB_BASE + (0x14)) // GPIOB ODR offset
.equ GPIOB_IDR,        (GPIOB_BASE + (0x10)) // GPIOB IDR offset
```

Main:

```
bcontrol:
       ldr r6, =GPIOA_IDR /* input data */
       ldr r5, [r6]
       lsrs r5, r5, #4
       movs r4, 0x1
       ands r5, r5, r4

       cmp r5, #0x1
       beq button1
       bne initialDisplay
```

This code controls if the button is pressed. If it is, goes to button1 function, if not, goes to initialDisplay function.

```
initialDisplay:
       /* write on D1 */
       bl turnonD1
       bl four
       bl delay8000
       bl resetstr /* clear data on mem[r6] */
       bl turnoffD /* turns off all digits */

       /* write on D2 */
       bl turnonD2
       bl zero
       bl delay8000
       bl resetstr
       bl turnoffD

       /* write on D3 */
       bl turnonD3
       bl one
       bl delay8000
       bl resetstr
       bl turnoffD

       /* write on D4 */
       bl turnonD4
       bl one
       bl delay8000
       bl resetstr
       bl turnoffD

       b bcontrol
```

initialDisplay function displays last four digits of my school ID which is 4011.

```
button1:
	bl turnonALL
	b countdown
```

button1 turns on all digits and goes to countdown.

```
countdown:
	bl nine
	bl delay1sec
	bl resetstr

	bl eight
	bl delay1sec
	bl resetstr

	bl seven
	bl delay1sec
	bl resetstr

	bl six
	bl delay1sec
	bl resetstr

	bl five
	bl delay1sec
	bl resetstr

	bl four
	bl delay1sec
	bl resetstr

	bl three
	bl delay1sec
	bl resetstr

	bl two
	bl delay1sec
	bl resetstr

	bl one
	bl delay1sec
	bl resetstr

	bl turnonLED

	bl zero
	bl delay1sec
	bl resetstr

	bl turnoffD
	bl delay1sec

	movs r1, #2 /* r1+1 toggle */
	b toggleZero
```

In countdown, all digits count down 9 to 0 at the same time. I could not implement the code to count the way it is asked.

I toggled zero 3 times with turned on LED.

```
toggleZero:
	bl turnonALL
	bl turnonLED
	bl zero
	bl delay1sec
	bl resetstr
	bl turnoffD
	bl delay1sec

	subs r1, r1, #1
	bne toggleZero
	b bcontrol
```

After count down ends and zero is toggled, the button is controlled. If it is pressed, count down starts again. If not, display “4011”.

I wrote functions for each number. For example, for zero;

```
zero:
	/* 1: (b c) 100_0000*/
	ldr r6, =GPIOB_ODR
	ldr r5, [r6]
	movs r4, 0x40 /* 0100_0000 */
	orrs r5, r5, r4
	str r5, [r6]
	bx lr
```

```
delay:
	subs r0, r0, #1
	bne delay
	bx lr
```

I wrote a delay function but defining r0 every single time seemed impractical so I wrote 2 different delay functions one for 1 second delay and 1 for very little delay.

```
delay8000: /* turns 8000 times */
	push {lr}
	ldr r0, =#8000
	bl delay
	pop {pc}
	bx lr

delay1sec: /* delay for one second */
	push {lr}
	ldr r0, =#3030303
	bl delay
	pop {pc}
	bx lr
```

```
resetstr: /* clears old data */
	movs r5, 0x0
	str r5, [r6]
	bx lr
```

To clear old data, write resetstr. In this way I don’t have to define it every time.

To turn on and off digits;

```
turnonD2:
	/* turn on D2 connected to A7 in ODR */
	str r5, [r6]
	ldr r6, =GPIOA_ODR
	ldr r5, [r6]
	ldr r4, =0x80 /* 0000_1000_0000 = lsrs r4, r4, #1 */
	orrs r5, r5, r4
	str r5, [r6]
	bx lr

turnonD3:
	/* turn on D3 connected to A6 in ODR */
	ldr r6, =GPIOA_ODR
	ldr r5, [r6]
	ldr r4, =0x40 /* 0000_0100_0000 = lsrs r4, r4, #1 */
	orrs r5, r5, r4
	str r5, [r6]
	bx lr

turnonD4:
	/* turn on D4 connected to A5 in ODR */
	ldr r6, =GPIOA_ODR
	ldr r5, [r6]
	ldr r4, =0x20 /* 0000_0010_0000 = lsrs r4, r4, #1 */
	orrs r5, r5, r4
	str r5, [r6]
	bx lr

turnonALL:
	push {lr}
	bl turnonD1
	bl turnonD2
	bl turnonD3
	bl turnonD4
	bl turnoffLED
	pop {pc}
	bx lr

turnoffD: /* turn off digits and LED */
	ldr r6, =GPIOA_ODR
	ldr r5, [r6]
	ldr r4, =0x0
	ands r5, r5, r4
	str r5, [r6]
	bx lr
```

Because the LED was connected PA, when turnonALL works, the LED was turned on too. To prevent that, I wrote turnoffLED function and called it in turnonALL.

```
turnoffLED:
	/* turn off led connected to A0 in ODR */
	ldr r6, =GPIOA_ODR
	ldr r5, [r6]
	movs r4, 0x1 /* 0001 */
	mvns r4, r4
	ands r5, r5, r4
	str r5, [r6]
	bx lr
```

```
turnonLED:
	/* turn on led connected to A0 in ODR */
	ldr r6, =GPIOA_ODR
	ldr r5, [r6]
	movs r4, 0x1 /* 0001 */
	orrs r5, r5, r4
	str r5, [r6]
	bx lr
```

3. Conclusion

When working on LEDs and buttons, old data was not that much important but it is needed to clear it when working on seven-segment display. Writing functions and calling these functions is more effective and takes less time comparing to write them over and over. Naming functions clearly is important for understandable code.

4. References

[1] RM0444 Reference manuel
[2] https://github.com/fcayci/stm32g0
[3] https://pdf.direnc.net/upload/14mm-4-lu-ortak-anot-display-datasheet.pdf