SANTA CLARA UNIVERSITY
Electrical and Computer Engineering Department

ELEN 120 – Embedded Computing Systems

Lab 3 – GPIO

Andrew Wolfe

Assignment: In this assignment, you will learn to setup the STM32L476G-DISCO Discovery kit, configure general purpose I/O pins (GPIO) and use them.

Learning Objectives: Learn to program the GPIO registers as a first example of I/O register programming and learn to create loop-based timing delays.

Lab Procedure:

Test – Before Problem 1:

Plug in the STM32L476G-DISCO Discovery board. Use the process described in the *Discovery Board Programming – Keil* document with the provided test file to create a project, build it, and debug it.

Problem 1

In this problem you are going to configure a GPIO and use it to light up the red LED on the Disco board. The schematic for the LED's on this board is shown below:

The red LED is connected to GPIO port B pin 2. The green LED is connected to GPIO port E pin 8. In this circuit driving a high voltage to the GPIO pin turns on the LED.

As discussed in class and in the book, each GPIO port is controlled and accessed via a bank of control and status registers. Each of these registers is 32-bits and includes a set of bit-field for configuration and/or status. The specifications of the GPIO control registers are provided in a file called GPIO register map that is provided along with the lab 3 handout. More details are in section 9 of the STM32L476VGT6 Reference Manual.
In order to use the GPIO as an output to drive an LED, we must configure it as follows: (Using port B as an example)

- Enable the GPIO port clock using the RCC_AHB2ENR register (which is described in Section 8, not section 9)
- Set the pin mode to digital output using the GPOIB_MODER register
- Make sure the pin is in push-pull mode using the GPIOB_OTYPER register (the default) • Set a pin output speed using the GPIOB_OSPEEDR register (low-speed default is fine) • Select any pull-up or pull-down resistors using the GPIOB_PUPDR register (default "none" is fine)
- Set the pin to high using the GPIOB_ODR register

The steps shown in blue can be skipped in this lab since the default values are

fine. You need to write a program that performs the 3 required steps.

Step 1:

Since the STM32L476VGT6 is designed to be a low-power part, pretty much everything is turned off are

reset. You need to turn hardware on before using it, including the ports. The AHB2 peripheral clock enable register (RCC_AHB2ENR) is described in section 8.4.17 of the STM32L476VGT6 Reference Manual.

The reset and control registers are in a block of registers that are defined by their offset from a base address. This base address is defined as RCC_BASE in the stm32l476xx_constants.s file. Since you are including this file in your code, you can use all of these mnemonics.

Section 8.4.17 of the STM32L476VGT6 Reference Manual explains that the offset of RCC_AHB2ENR is 0x4C. This is already defined as the value RCC_AHB2ENR in stm32l476xx_constants.s. By adding this offset to the register back base address, you get the address of the AHB2 peripheral clock enable register. The manual explains that bit 1 of this register must be set to 1 to enable the GPIO B clock. To do this you must read this register, set bit 1 to 1 without changing any other bit, and write the value back. Write code to do this.

Step 2:

Next we need to set the pin mode to digital output using the GPOIB_MODER register. The GPIO registers are all offset from a base address that is unique to each port. For example, the registers to configure GPIO B are offset from the value GPIOB_BASE. By adding the appropriate register offset to this value, we get the address for each GPIOB_XXXXX register. For example, the address of GPOIB_MODER is GPIOB_BASE + 0x00. The mnemonic for this offset value is GPIO_MODER. We can either do this math on the CPU or the assembler can usually do it for us.

As shown below, the program pin 2 of the GPOIB_MODER register for General purpose output mode, we must set bit 4 to 1 and bit 5 to 0 without changing any other bits. Again, this requires reading the register, modifying key bits, then storing the register. Write code to do this that runs after your clock enable code.

Step 3:

Next, we need to set the pin to high to turn on the LED. To do this, we must set bit 2 of the GPIOB_ODR register to high. The mnemonic for this offset value is GPIO_ODR. Add that code. Also add an infinite loop at the end of your program. Now test it and make it work. You can test in the debugger or on the hardware by downloading and hitting reset.

9.4.6 GPIO port output data register (GPIOx_ODR) (x = A..H)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port output data bit (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A..F).

Demo for the TA and turn in commented code along with a picture of the LED

on.

```
;******************** (C) Andrew Wolfe
  ****************************************
; @file    main_hw_proto.s
; @author  Andrew Wolfe
; @date    August 18, 2019
; @note
;         This code is for the book "Embedded Systems with ARM Cortex-M
;         Microcontrollers in Assembly Language and C, Yifeng Zhu,
;         ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara
  University
;*****************************************************************
  **************

  INCLUDE core_cm4_constants.s                ; Load Constant Definitions
  INCLUDE stm32l476xx_constants.s

                AREA    main, CODE, READONLY
                EXPORT          __main
                ENTRY

__main   PROC

                LDR r0, =(RCC_BASE + RCC_AHB2ENR) ;activates red
  clock
                LDR r1, [r0]
                ORR r1, #RCC_AHB2ENR_GPIOBEN
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_MODER) ;sets up digital
```

```
 output
                LDR r1, [r0]
                BIC r1, r1, #(0x03<<(2*2))
                ORR r1, r1, #(1<<(2*2))
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_OTYPER) ;sets reset states
                LDR r1, [r0]
                BIC r1, r1,#GPIO_OTYPER_OT_2
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_ODR) ;puts the led bit to
 high
                LDR r1, [r0]
                ORR r1, r1,#GPIO_ODR_ODR_2
                STR r1, [r0]
endless         b               endless

        ENDP
                ALIGN
                AREA    myData, DATA, READWRITE
                ALIGN

counter         DCD             10

 END
```

Problem 2

As shown above, the green LED is at port E pin 8. Add additional code to enable the clock for port E, configure pin 8, and turn that pin on. This should now turn on both LEDs.

Demo for the TA and turn in commented code along with a picture of the LEDs on.

```
;******************** (C) Andrew Wolfe ****************************************
; @file    main_hw_proto.s
; @author  Andrew Wolfe
; @date    August 18, 2019
; @note
;         This code is for the book "Embedded Systems with ARM Cortex-M
;         Microcontrollers in Assembly Language and C, Yifeng Zhu,
;         ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara University
;******************************************************************************



 INCLUDE core_cm4_constants.s                   ; Load Constant Definitions
 INCLUDE stm32l476xx_constants.s
```

```
                AREA    main, CODE, READONLY
                EXPORT          __main
                ENTRY

__main  PROC

                LDR r0, =(RCC_BASE + RCC_AHB2ENR) ;activates red clock
                LDR r1, [r0]
                ORR r1, #RCC_AHB2ENR_GPIOBEN
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_MODER) ;sets up digital output
                LDR r1, [r0]
                BIC r1, r1, #(0x03<<(2*2))
                ORR r1, r1, #(1<<(2*2))
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_OTYPER) ;sets reset states
                LDR r1, [r0]
                BIC r1, r1,#GPIO_OTYPER_OT_2
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_ODR) ;puts the led bit to high
                LDR r1, [r0]
                ORR r1, r1,#GPIO_ODR_ODR_2
                STR r1, [r0]
                ;part 1^^^^^
                LDR r0, =(RCC_BASE + RCC_AHB2ENR) ;activates green clock
                LDR r1, [r0]
                ORR r1, #RCC_AHB2ENR_GPIOEEN
                STR r1, [r0]

                LDR r0, =(GPIOE_BASE+GPIO_MODER) ;sets up digital output
                LDR r1, [r0]
                BIC r1, r1, #(0x03<<(2*8))
                ORR r1, r1, #(1<<(2*8))
                STR r1, [r0]

                LDR r0, =(GPIOE_BASE+GPIO_OTYPER) ;sets reset states
                LDR r1, [r0]
                BIC r1, r1,#GPIO_OTYPER_OT_8
                STR r1, [r0]

                LDR r0, =(GPIOE_BASE+GPIO_ODR) ;puts the led bit to high
                LDR r1, [r0]
                ORR r1, r1,#GPIO_ODR_ODR_8
                STR r1, [r0]
endless         b               endless

        ENDP
                ALIGN
                AREA    myData, DATA, READWRITE
                ALIGN
```

```
counter         DCD         10

  END
```

Problem 3

A loop that takes many iterations in the code with little or nothing inside can be used to insert a delay. In general, this is a bad plan, but it will work until we discover a better one. Add in a loop that takes about 1 second to execute then turn off both LEDs. Remember that the clocks are already configured and the ports are already configured, so you only need to change the output value on each port using the output data register every second. How do you figure out 1 second? Trial and error (along with some math).

Demo for the TA and turn in commented code along with a picture of the LEDs on. **In your lab report, explain how you determined the right code values for a 1 second delay. Provide a quantitative analysis of the loop that you developed based on computing how many clock cycles each loop iteration requires.**

```
;******************** (C) Andrew Wolfe ****************************************
; @file    main_hw_proto.s
; @author  Andrew Wolfe
; @date    August 18, 2019
; @note
;         This code is for the book "Embedded Systems with ARM Cortex-M
;         Microcontrollers in Assembly Language and C, Yifeng Zhu,
;         ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara University
;******************************************************************************



  INCLUDE core_cm4_constants.s              ; Load Constant Definitions
  INCLUDE stm32l476xx_constants.s

                AREA    main, CODE, READONLY
                EXPORT          __main
                ENTRY

__main    PROC

                LDR r0, =(RCC_BASE + RCC_AHB2ENR) ;activates red clock
                LDR r1, [r0]
                ORR r1, #RCC_AHB2ENR_GPIOBEN
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_MODER) ;sets up digital output
                LDR r1, [r0]
                BIC r1, r1, #(0x03<<(2*2))
                ORR r1, r1, #(1<<(2*2))
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_OTYPER) ;sets reset states
                LDR r1, [r0]
                BIC r1, r1,#GPIO_OTYPER_OT_2
```

```
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_ODR) ;puts the led bit to high
                LDR r1, [r0]
                ORR r1, r1,#GPIO_ODR_ODR_2
                STR r1, [r0]
                ;part 1^^^^^
                LDR r0, =(RCC_BASE + RCC_AHB2ENR) ;activates green clock
                LDR r1, [r0]
                ORR r1, #RCC_AHB2ENR_GPIOEEN
                STR r1, [r0]

                LDR r0, =(GPIOE_BASE+GPIO_MODER) ;sets up digital output
                LDR r1, [r0]
                BIC r1, r1, #(0x03<<(2*8))
                ORR r1, r1, #(1<<(2*8))
                STR r1, [r0]

                LDR r0, =(GPIOE_BASE+GPIO_OTYPER) ;sets reset states
                LDR r1, [r0]
                BIC r1, r1,#GPIO_OTYPER_OT_8
                STR r1, [r0]

                LDR r0, =(GPIOE_BASE+GPIO_ODR) ;puts the led bit to high
                LDR r1, [r0]
                ORR r1, r1,#GPIO_ODR_ODR_8
                STR r1, [r0]

                LDR r2, =0x000DBBA0  ;loops 1000000 times
loop            SUBS r2, #1
                CMP r2, #0
                BNE     loop

                LDR r0, =(GPIOB_BASE+GPIO_ODR) ;turns off red light
                LDR r1, [r0]
                EOR r1, r1,#GPIO_ODR_ODR_2
                STR r1, [r0]

                LDR r0, =(GPIOE_BASE+GPIO_ODR) ;turns off green light
                LDR r1, [r0]
                EOR r1, r1,#GPIO_ODR_ODR_8
                STR r1, [r0]


endless         b               endless

        ENDP
                ALIGN
                AREA    myData, DATA, READWRITE
                ALIGN

counter         DCD             10

  END
```

We used the software to determine our time by measuring the time it took for one instruction to execute. However, ended up having to use a smaller value for our loop than what was calculated. The timer used in the debug mode doesn't take into account CPU interrupts.

Problem 4

Now make a program that turns on the red LED, then every 1 second will toggle between the red LED and the green LED.

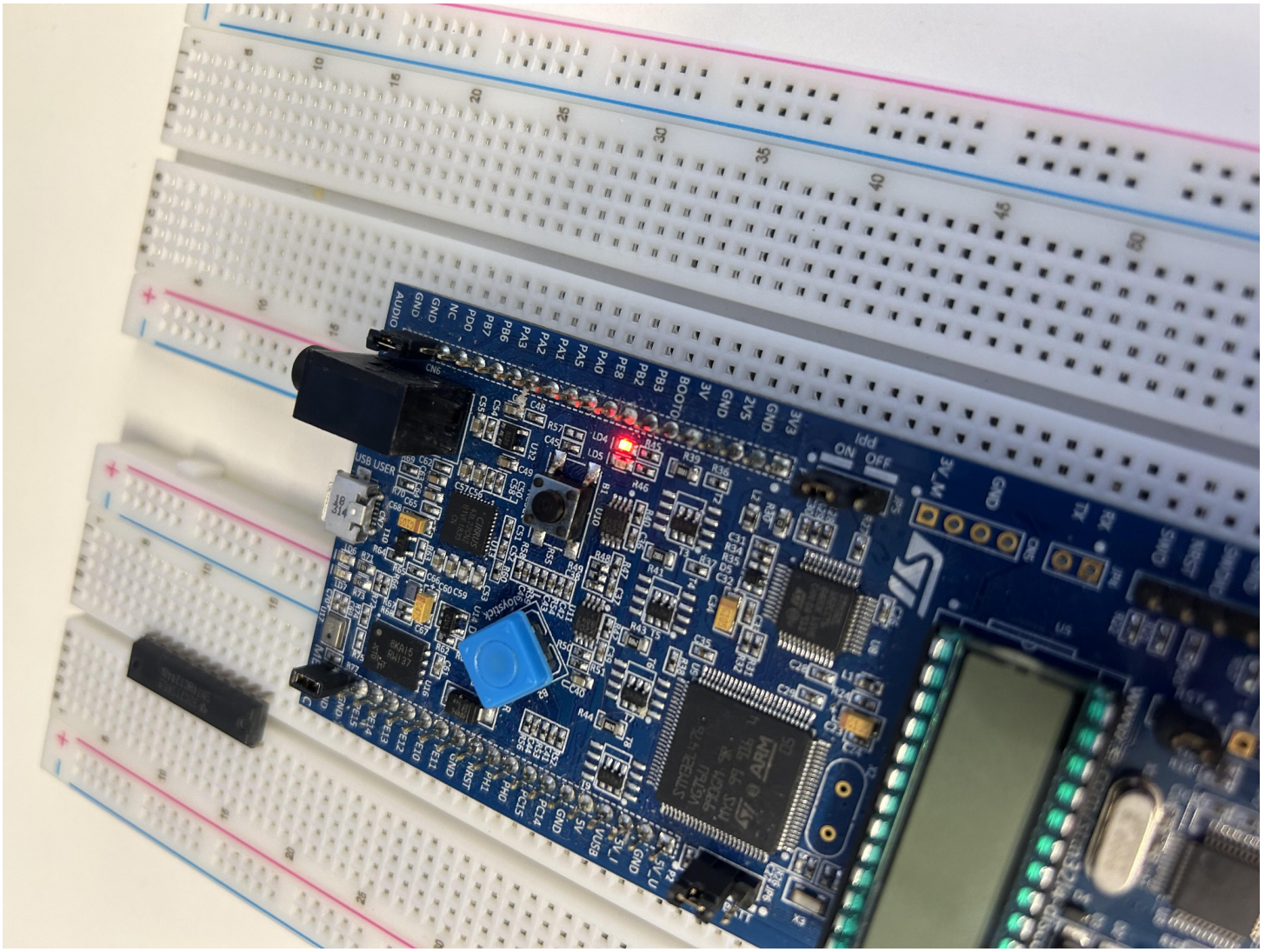
Demo for the TA and turn in commented code along with a video.

```
;******************** (C) Andrew Wolfe ********************************************
; @file    main_hw_proto.s
; @author  Andrew Wolfe
; @date    August 18, 2019
; @note
;         This code is for the book "Embedded Systems with ARM Cortex-M
;         Microcontrollers in Assembly Language and C, Yifeng Zhu,
;         ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara University
;**********************************************************************************



 INCLUDE core_cm4_constants.s                    ; Load Constant Definitions
 INCLUDE stm32l476xx_constants.s

                AREA    main, CODE, READONLY
                EXPORT          __main
                ENTRY

__main   PROC

                LDR r0, =(RCC_BASE + RCC_AHB2ENR) ;activates red clock
                LDR r1, [r0]
                ORR r1, #RCC_AHB2ENR_GPIOBEN
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_MODER) ;sets up digital output
                LDR r1, [r0]
                BIC r1, r1, #(0x03<<(2*2))
                ORR r1, r1, #(1<<(2*2))
                STR r1, [r0]

                LDR r0, =(GPIOB_BASE+GPIO_OTYPER) ;sets reset states
                LDR r1, [r0]
                BIC r1, r1,#GPIO_OTYPER_OT_2
                STR r1, [r0]
```

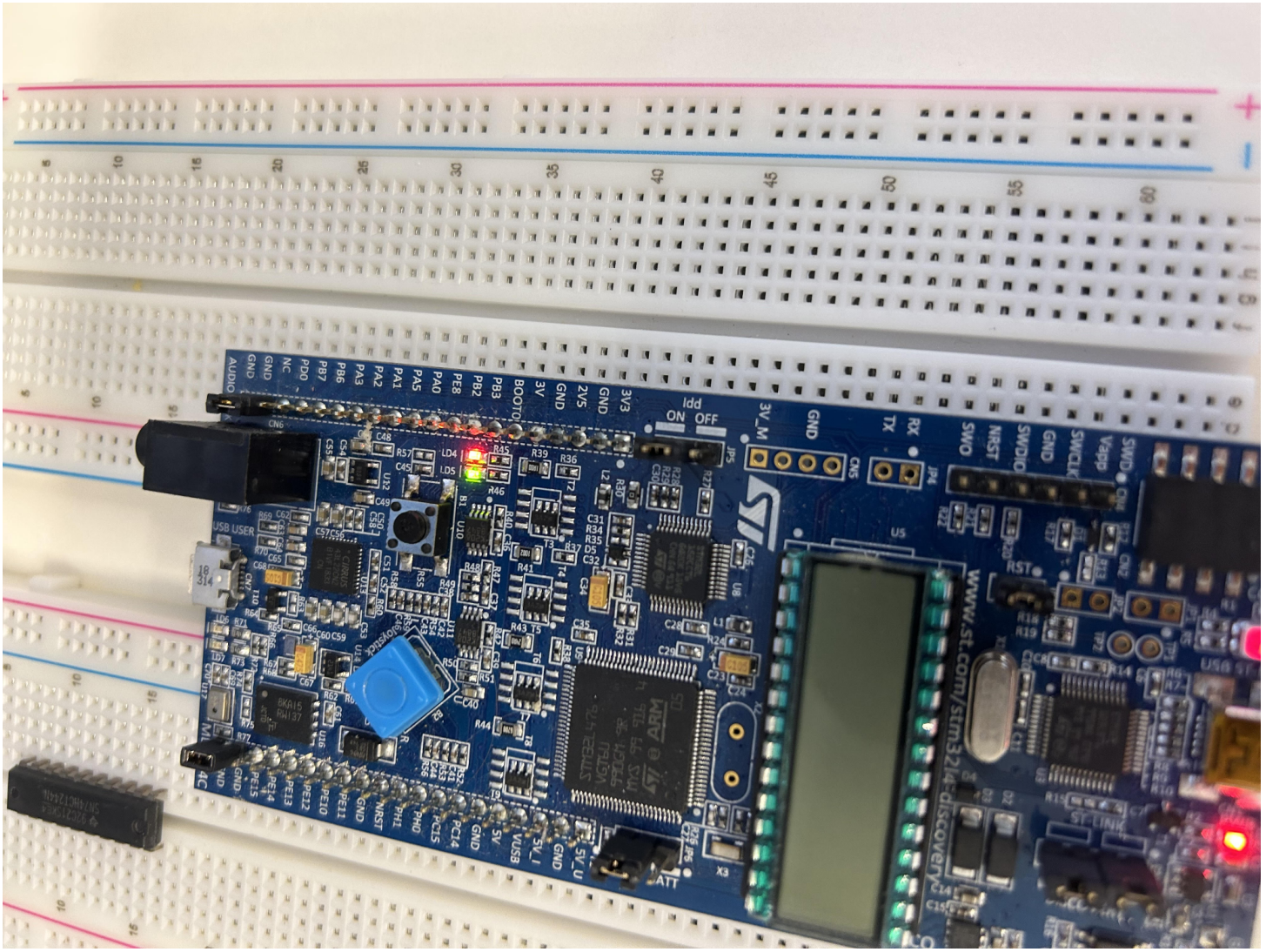
```
                ;part 1^^^^^
                LDR r0, =(RCC_BASE + RCC_AHB2ENR) ;activates green clock
                LDR r1, [r0]
                ORR r1, #RCC_AHB2ENR_GPIOEEN
                STR r1, [r0]

                LDR r0, =(GPIOE_BASE+GPIO_MODER) ;sets up digital output
                LDR r1, [r0]
                BIC r1, r1, #(0x03<<(2*8))
                ORR r1, r1, #(1<<(2*8))
                STR r1, [r0]

                LDR r0, =(GPIOE_BASE+GPIO_OTYPER) ;sets reset states
                LDR r1, [r0]
                BIC r1, r1,#GPIO_OTYPER_OT_8
                STR r1, [r0]


endless         LDR r0, =(GPIOB_BASE+GPIO_ODR) ;puts the red led bit to high
                LDR r1, [r0]
                ORR r1, r1,#GPIO_ODR_ODR_2
                STR r1, [r0]

                LDR r2, =0x000E09C0  ;loops 1000000 times
loop            SUBS r2, #1
                CMP r2, #0
                BNE     loop

                LDR r0, =(GPIOB_BASE+GPIO_ODR) ;turns off red light
                LDR r1, [r0]
                EOR r1, r1,#GPIO_ODR_ODR_2
                STR r1, [r0]

                LDR r0, =(GPIOE_BASE+GPIO_ODR) ;puts the green led bit to high
                LDR r1, [r0]
                ORR r1, r1,#GPIO_ODR_ODR_8
                STR r1, [r0]

                LDR r2, =0x000E09C0  ;loops 1000000 times
loop2           SUBS r2, #1
                CMP r2, #0
                BNE     loop2

                LDR r0, =(GPIOE_BASE+GPIO_ODR) ;turns off green light
                LDR r1, [r0]
                EOR r1, r1,#GPIO_ODR_ODR_8
                STR r1, [r0]
                b                   endless


        ENDP
                ALIGN
                AREA    myData, DATA, READWRITE
```

```
                ALIGN
counter         DCD         10
  END
```

Problem 1 picture:



Problem 2 picture:


www.st.com/stm32f4-discovery
ON OFF
Idd
USB USER
RST
ST-LINK