SANTA CLARA UNIVERSITY
Electrical and Computer Engineering Department

Real-Time Embedded Systems - ECEN 121
Lab 4

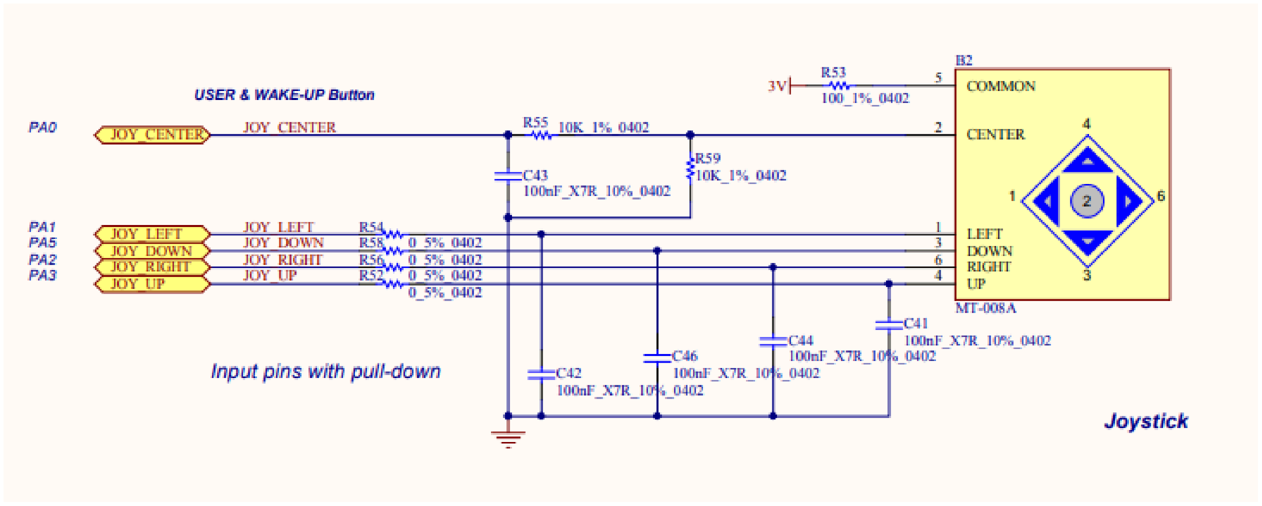
Dylan Thornburg & Kai Hoshide

Building an LCD Stopwatch

Andrew Wolfe

Prelab:

1) You will need to configure the joystick to control your stopwatch.
   a. Find the portion of the DISCO board schematic that shows the joystick circuit and copy it into your prelab.



   b. Explain what GPIO pins need to be configured in order to use the center and left switches on the joystick as inputs.

   We need to configure GPIO PA0 and GPIO PA1. We can tell this from looking at the diagram.

   c. Explain what needs to be configured on each of the GPIO pins.

   We have to enable the clock (RCC_AHB2ENR_GPIOAEN), configure the mode as input (GPIO_MODER for GPIOA), and we need to make sure all pins are pull down (using PUPDR).

2) You will need to configure a timer to support the stopwatch. We are going to use Timer 16 and wet it to interrupt every 100ms. The system clock and the timer clock will be set to 20MHz. What values should be used for the prescaler and the counter period registers?

Once every 100ms = 10hz; x=prescaler, y=counter period; both must be 0< here <65535.

20000000/(2*(xy))=10 -> xy=1000000

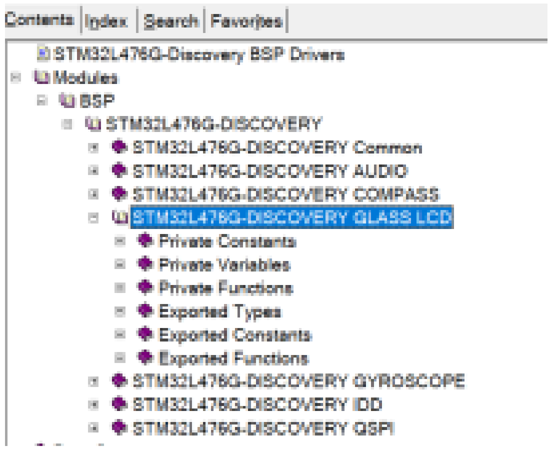
I choose x = 20000, and y = 50. But we have to do the minus one thing so the actual value in the registers is: prescaler = 19999, and counter period = 49.

3) You are also going to use the LCD. We are going to access it through the BSP libraries. The BSP LCD driver specifications are in the BSP manual. It is super finicky though. I suggest you copy the entire BSP directory to your own computer or account and use your local copy.

The manual is a file that you should be able to open with a browser. The file is called BSP\STM32L476G-Discovery\ STM32L476G-Discovery_BSP_User_Manual.chm If you cannot get it to work, there is a PDF version (BSP Glass Doumentation.pdf) in the Lab Handouts directory.

Open that manual and find the section on Glass LCD

1


Contents
Index
Search
Favorites
STM32L476G-Discovery BSP Drivers
Modules
BSP
STM32L476G-DISCOVERY
STM32L476G-DISCOVERY Common
STM32L476G-DISCOVERY AUDIO
STM32L476G-DISCOVERY COMPASS
STM32L476G-DISCOVERY GLASS LCD
Private Constants
Private Variables
Private Functions
Exported Types
Exported Constants
Exported Functions
STM32L476G-DISCOVERY GYROSCOPE
STM32L476G-DISCOVERY IDD
STM32L476G-DISCOVERY QSPI

Go to the Exported Functions section. These are the functions we will use.

Answer the following questions:

1) How many functions are exported in the BSP_LCD_GLASS API?
12

2) What parameters must be supplied to BSP_LCD_GLASS_Init() ?
None; it is void yo.

3) Which of those functions write to the frame buffer? List them.
BSP_LCD_GLASS_BarLevelConfig
This is the only function that actually says it WRITES to the frame buffer: "Configure the bar level on LCD by writing bar value in LCD frame buffer". Other functions deal with frame buffer or RAM buffer but only BarLevelConfig writes to the frame buffer.

Lab Procedure:

Step 1:

Create a new STM32CubeMX project for the discovery board. Call it LCD Timer. Be sure to go through all the steps for configuring a project and remember to check compiler and debugger options once you start Keil.

Configure the left and center joystick ports as interrupts. (We went over this in class). Give them useful pin names. Think about how GPIO Pull-up/Pull-down needs to be configured. No to the NVIC tab and enable the interrupts.

Step 2:

Enable Timer TIM16. Set the appropriate Prescaler and Counter Period Values. Use the default system and timer clock rate. Enable the interrupt in the NVIC settings tab.

Step 3:

Disable USB, QuadSPI Flash, I2C1, i2C2, SPI2, USART2, and USB_OTG_FS, and SAI1 in the same way you did in Lab 2.

Step 4:

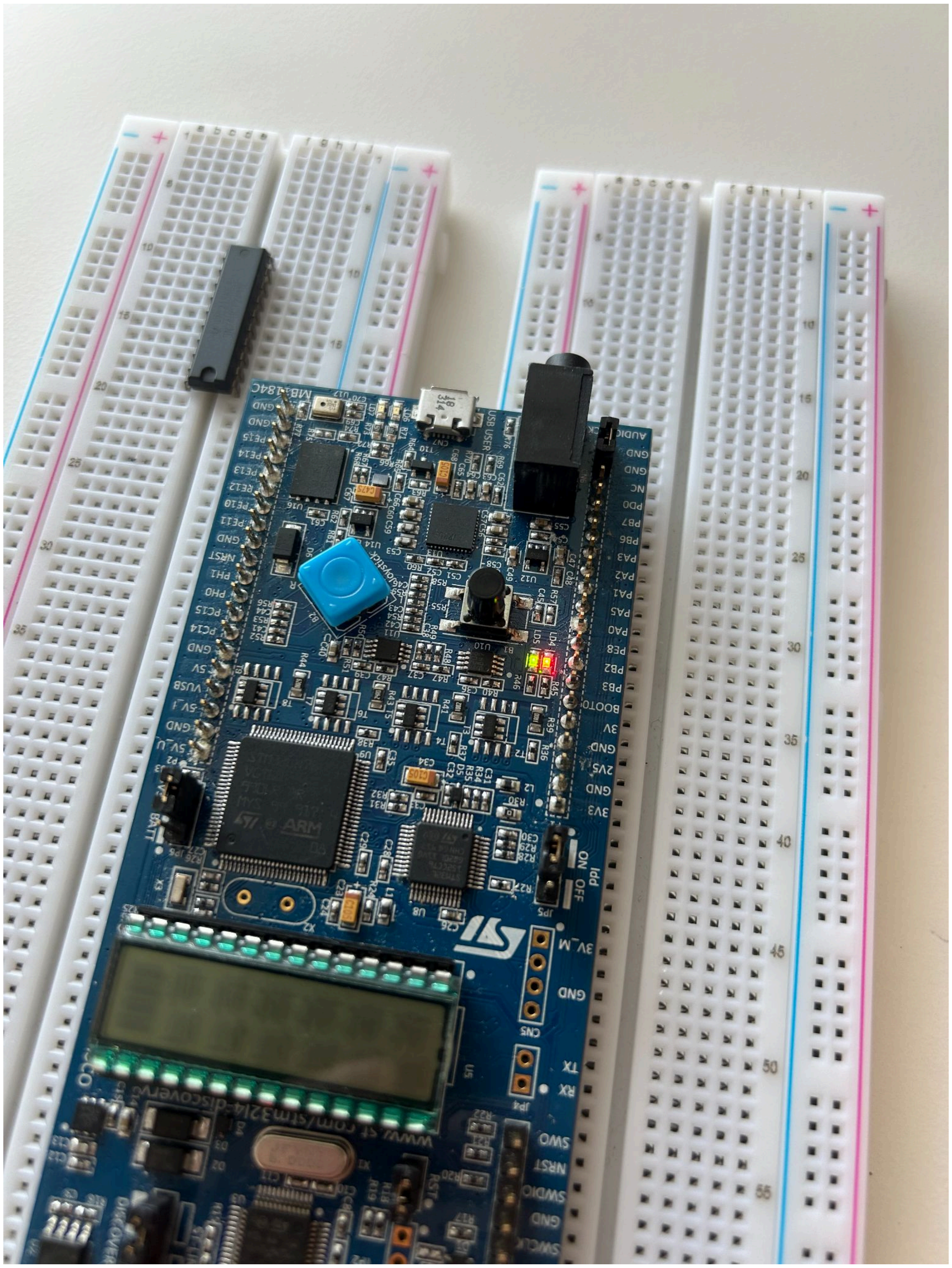
You must enable the real-time clock (RTC) in the Timers section and select "Activate Clock Source." This fixes a bug in these tool versions.

Step 5:

Generate code and open the project. Check your Keil settings. Compile the code to make sure it compiles.

Step 6:

Find the interrupt service routines for the center and left joystick buttons. Add code to those so that when you push the center joystick, the red LED toggles and when you push the left joystick, the green LED toggles. Test this code and make sure it works. **DEMO for the TA – DEMO 1.**



I started timer 16 in main and then added this in stm32l4xx_it:

```
void EXTI0_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI0_IRQn 0 */

  /* USER CODE END EXTI0_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(JOY_CENTER_Pin);
  /* USER CODE BEGIN EXTI0_IRQn 1 */
            HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);

  /* USER CODE END EXTI0_IRQn 1 */
}

/**
  * @brief This function handles EXTI line1 interrupt.
  */
void EXTI1_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI1_IRQn 0 */

  /* USER CODE END EXTI1_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(JOY_LEFT_Pin);
  /* USER CODE BEGIN EXTI1_IRQn 1 */
            HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
  /* USER CODE END EXTI1_IRQn 1 */
}
```

This code just toggles the red LED if I press center on the joystick and toggles the green LED if I press left on the joystick.

Step 7:

Create a new empty source code file called display.c in the project Src directory and add it to you project. My preferred way to do this is to use the “Add new item..” menu item in the Application/User group.

In display.c you should add the following variables:

```
volatile int seccount = 0;
volatile int mincount = 0;
volatile int tenth = 0;
```

Also include main.h

In stm32l4xx_it.c you should add external references to these variables.

Find the Timer 16 interrupt service routine and modify it so that it:

- Adds 1 to tenth each interrupt
- Resets tenth to 0 if it exceeds 9
- Toggles the green LED each time it resets tenth

Add the code to main to start timer 16.

Build and test the project.

Step 7:

Copy the files stm32l476g_discovery.c and stm32l476g_discovery_glass_lcd.c from the BSP\STM32L476G-Discovery\ directory to your Src directory. Include those into the item in the Application/User group. The file “stm32l476g_discovery_glass_lcd.c” will produce 2 warnings that are

inconsequential – but if you *substitute* the file “stm32l476g_discovery_glass_lcd_wolfemod.c” from the BSP directory, the warnings will go away.

Copy the files stm32l476g_discovery.h and stm32l476g_discovery_glass_lcd.h from the BSP\STM32L476G-Discovery\ directory to your Inc directory.

Add the following lines to display.c at the top and to main.c.

```
#include "stm32l476g_discovery.h"
#include "stm32l476g_discovery_glass_lcd.h"
```

Create a new function in display.c called display_test().

void display_test(void) {}

Add code to this function to call BSP_LCD_GLASS_DisplayChar() in order to display an X in character location 3.

Also create a new empty file called display.h in the project Inc directory. Don’t add it to the project. It will be found when you build the project. Include a prototype of display_test() in that file.

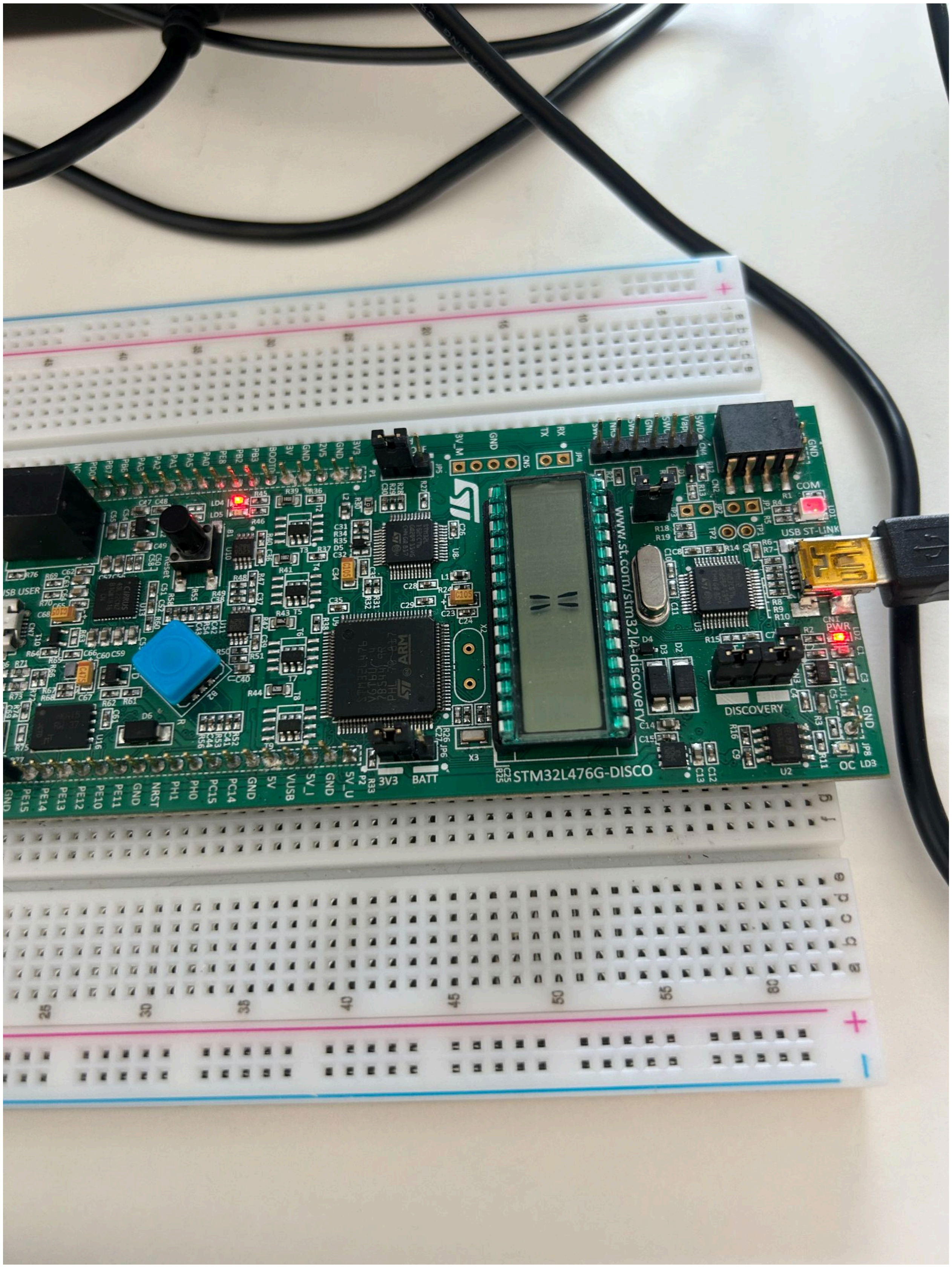
void display_test(void);

Also – include display.h in main.c

Call BSP_LCD_GLASS_Init () from main().

Call display_test() from main().

Compile and test. **DEMO for the TA – DEMO 2.**


STM32L476G-DISCO
www.st.com/stm32l4-discovery
DISCOVERY
USB ST-LINK
USB USER
COM
PWR
3V3
BATT

I feel like this is pretty basic. It's mostly just initializing and getting timing right. A tenth must repeat 10 times for one second.

MAIN CODE:

```
/* USER CODE BEGIN 2 */
        HAL_TIM_Base_Start_IT(&htim16);
        BSP_LCD_GLASS_Init();
        BSP_LCD_GLASS_ClearBar(0xFFFFFFFF);
        BSP_LCD_GLASS_Clear();
        display_test();
 /* USER CODE END 2 */
```

DISPLAY CODE:

```
#include "main.h"
#include "stm32l476g_discovery.h"
#include "stm32l476g_discovery_glass_lcd.h"

volatile int seccount = 0;
volatile int mincount = 0;
volatile int tenth = 0;
char ch = 'X';
void display_test(void)
{
        BSP_LCD_GLASS_DisplayChar((uint8_t*)&ch, POINT_OFF, DOUBLEPOINT_OFF ,3 );
}
```

STM32l4xx CODE:

```
tenth++;
                if(tenth > 9)
                {
                        tenth = 0;
                        HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
                }
```

Step 8:

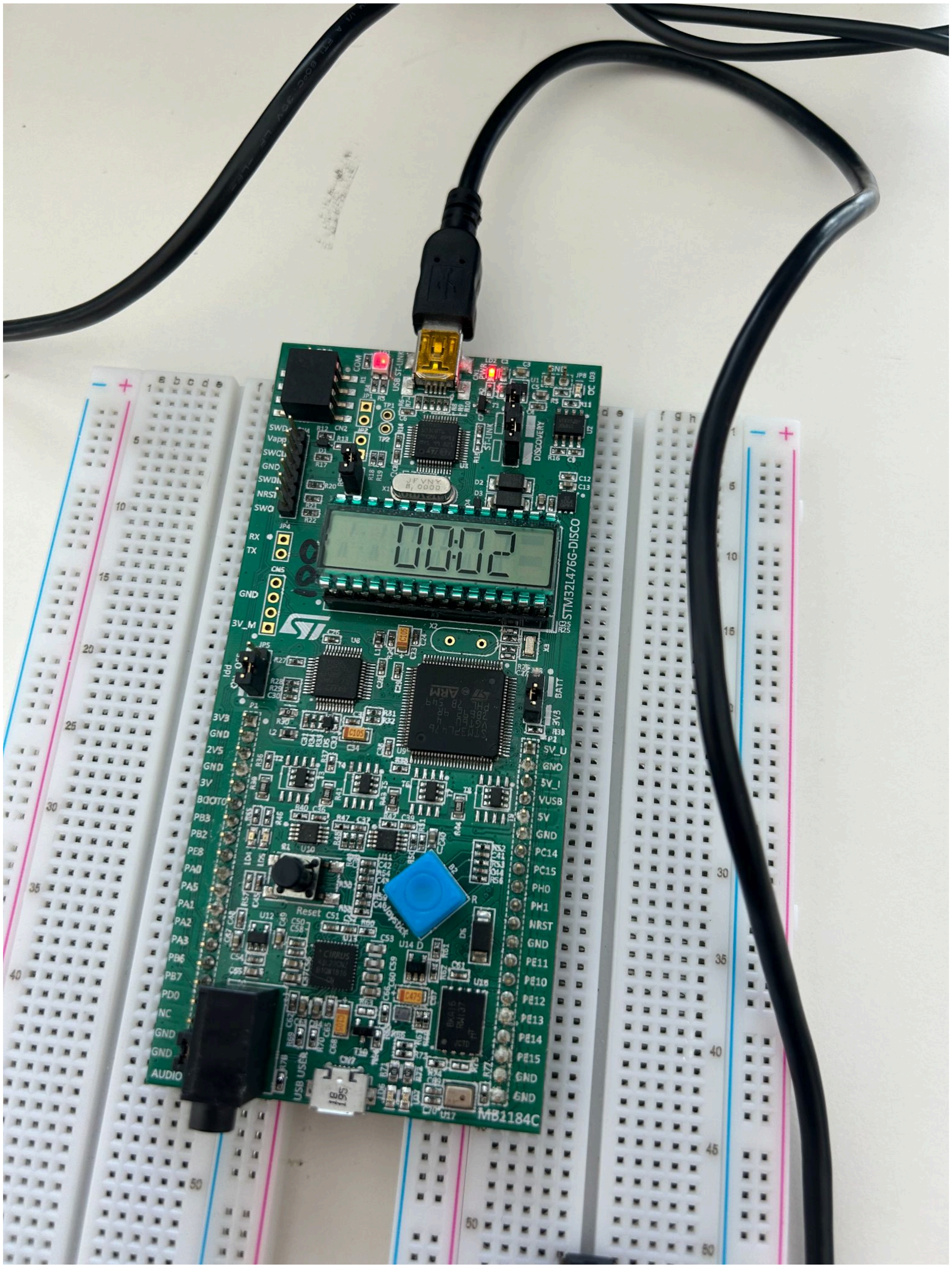
Finish the project.

Digits 3-6 of the LCD should display a stopwatch. 2 digit minutes, colon, 2 digit seconds. All leading 0s displayed. It should start counting when the board is reset.

When the center button is pressed, the stopwatch should pause. When it is pressed again, the count should resume.

When the left button is pressed, the count should reset to 0. The counting mode should not change.

DEMO for the TA – DEMO 3.


00:02
STM32L476G-DISCO
MB1184C

CODE:

```
void EXTI0_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI0_IRQn 0 */

  /* USER CODE END EXTI0_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(JOY_CENTER_Pin);
  /* USER CODE BEGIN EXTI0_IRQn 1 */
		cond2++;
		if(cond2 == 10) //reset case that makes sure cond2 doesn't get too big
		{
			cond2 = 0;
		}
  /* USER CODE END EXTI0_IRQn 1 */
}

/**
  * @brief This function handles EXTI line1 interrupt.
  */
void EXTI1_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI1_IRQn 0 */

  /* USER CODE END EXTI1_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(JOY_LEFT_Pin);
  /* USER CODE BEGIN EXTI1_IRQn 1 */
		uint8_t reset = 48;
		BSP_LCD_GLASS_DisplayChar(&reset, POINT_OFF, DOUBLEPOINT_OFF ,5 );
		BSP_LCD_GLASS_DisplayChar(&reset, POINT_OFF, DOUBLEPOINT_OFF ,4 );
		BSP_LCD_GLASS_DisplayChar(&reset, POINT_OFF, DOUBLEPOINT_ON ,3 );
		BSP_LCD_GLASS_DisplayChar(&reset, POINT_OFF, DOUBLEPOINT_OFF ,2 );
		seccount = 0;
		tenseccount = 0;
		tenmincount = 0;
		mincount = 0;
		tenth = 0;

  /* USER CODE END EXTI1_IRQn 1 */
}

/**
  * @brief This function handles TIM1 update interrupt and TIM16 global interrupt.
  */
void TIM1_UP_TIM16_IRQHandler(void)
{

  /* USER CODE BEGIN TIM1_UP_TIM16_IRQn 0 */

  /* USER CODE END TIM1_UP_TIM16_IRQn 0 */
```

```
 HAL_TIM_IRQHandler(&htim16);
 /* USER CODE BEGIN TIM1_UP_TIM16_IRQn 1 */
        if(cond2 % 2 == 0)
        {
        uint8_t ch;
        uint8_t ch2;
                uint8_t ch3;
                uint8_t ch4;
        tenth++;
                if(tenth > 9)
                {
                        seccount++;
                        tenth=0;
                        if (seccount> 9)
                        {
                                seccount = 0;
                                tenseccount++;
                                if(tenseccount > 5)
                                        {
                                                tenseccount = 0;
                                                mincount++;
                                                if (mincount > 9)
                                                {
                                                        mincount = 0;
                                                        tenmincount++;
                                                        if (tenmincount > 5)
                                                        {
                                                                mincount = 0;
                                                                tenmincount = 0;
                                                        }
                                                        ch4 = tenmincount + 48;
                                                        BSP_LCD_GLASS_DisplayChar(&ch4,
POINT_OFF, DOUBLEPOINT_OFF , 2);
                                                }
                                                ch3 = mincount + 48;
                                                BSP_LCD_GLASS_DisplayChar(&ch3, POINT_OFF,
DOUBLEPOINT_ON , 3);
                                        }
                                        ch2 = tenseccount + 48;
                                        BSP_LCD_GLASS_DisplayChar(&ch2, POINT_OFF,
DOUBLEPOINT_OFF ,4 );
                        }
                        ch = seccount + 48;
                        BSP_LCD_GLASS_DisplayChar(&ch, POINT_OFF, DOUBLEPOINT_OFF ,5 );

                }

        }
                /* USER CODE END TIM1_UP_TIM16_IRQn 1 */
```

```
}
```

In our code we counted up using variables that incremented anytime tenth reached 10. That part should be easy to see. Then we used a condition where if the center button was pressed, the condition would be incremented. When it was even our stopwatch worked, but when odd, it paused. For the reset we just set every variable we used to 0 (except cond2) and made the timer display everything as zero.