

Lab02 Task1

Youtube Link: <https://youtu.be/HPGj5UYOLb0>

```
/* For usleep() */
#include <unistd.h>
#include <stdint.h>
#include <stddef.h>

/* Driver Header files */
#include <ti/drivers/GPIO.h>
// #include <ti/drivers/I2C.h>
// #include <ti/drivers/SDSPI.h>
// #include <ti/drivers/SPI.h>
// #include <ti/drivers/UART.h>
// #include <ti/drivers/Watchdog.h>

/* Board Header file */
#include "Board.h"

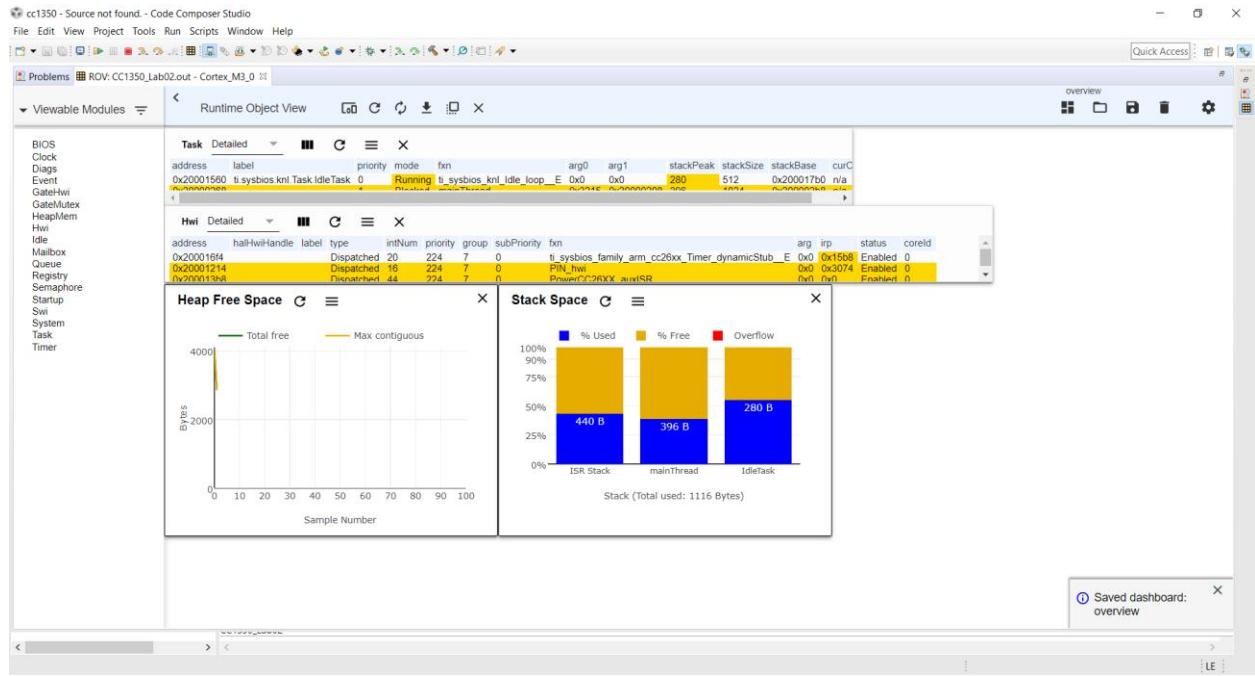
/*
 * ===== mainThread =====
 */
void *mainThread(void *arg0)
{
    /* 1 second delay */
    uint32_t time = 1;

    /* Call driver init functions */
    GPIO_init();
    // I2C_init();
    // SDSPI_init();
    // SPI_init();
    // UART_init();
    // Watchdog_init();

    /* Configure the LED pin */
    GPIO_setConfig(Board_GPIO_LED0, GPIO_CFG_OUT_STD | GPIO_CFG_OUT_LOW);

    /* Turn on user LED */
    GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_ON);

    while (1) {
        sleep(time);
        GPIO_toggle(Board_GPIO_LED0);
    }
}
```



Lab02 Task2&3

Youtube Link: <https://youtu.be/Rijw3vGCj7A>

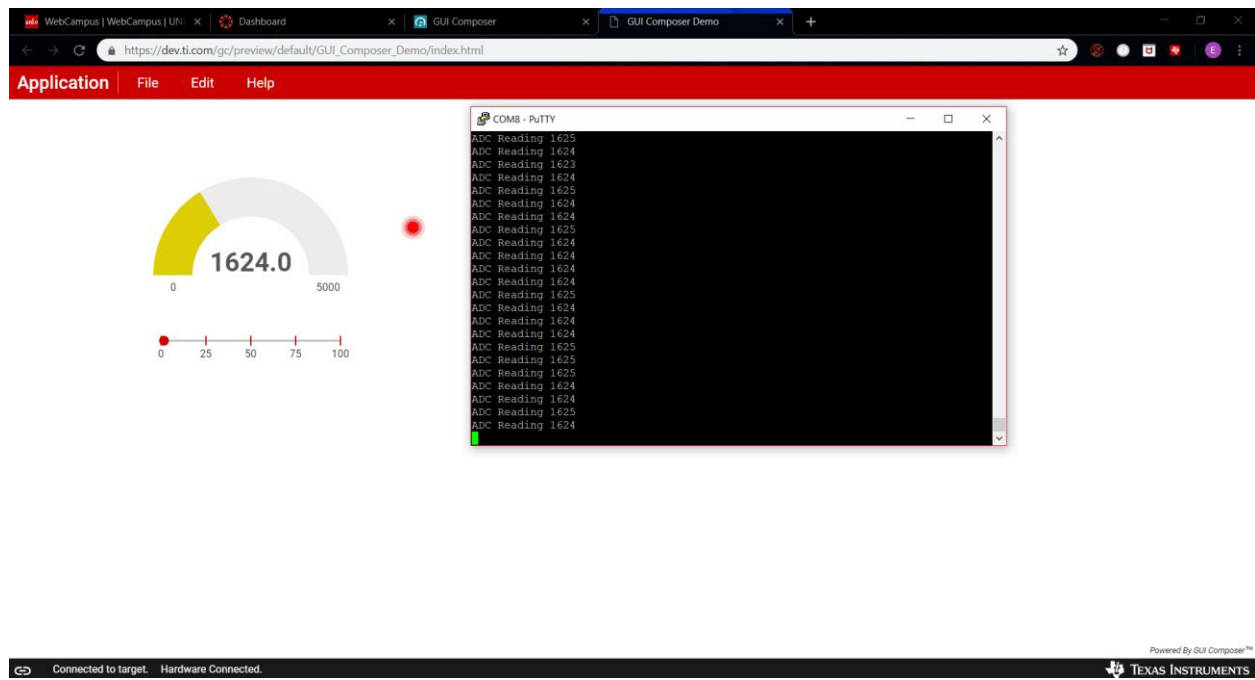
```
#include <unistd.h>
#include <stdint.h>
#include <stddef.h>

/* Driver Header files */
#include <ti/drivers/GPIO.h>
#include <ti/drivers/ADC.h>
#include <ti/display/Display.h>
// #include <ti/drivers/I2C.h>
// #include <ti/drivers/SDSPI.h>
// #include <ti/drivers/SPI.h>
// #include <ti/drivers/UART.h>
// #include <ti/drivers/Watchdog.h>
/* Board Header file */
#include "Board.h"
/* global variables FOR GUI COMPOSER */
uint16_t adcValue = 0;
uint16_t threshold = 100;
uint16_t trigger = 0;
/*
 * ===== mainThread =====
 */
void *mainThread(void *arg0)
{
    /* ~10 loops/second */
    uint32_t time = 100000; // update ~10/second
    /* Call driver init functions */
    GPIO_init();
    ADC_init();
```

```

// I2C_init();
// SDSPI_init();
// SPI_init();
// UART_init();
// Watchdog_init();
/* Open ADC Driver */
ADC_Handle adc;
ADC_Params params;
ADC_Params_init(&params);
adc = ADC_open(Board_ADC0, &params);
if (adc == NULL) {
    // Error initializing ADC channel 0
    while (1);
}
/* Open Display Driver */
Display_Handle displayHandle;
Display_Params displayParams;
Display_Params_init(&displayParams);
displayHandle = Display_open(Display_Type_UART, NULL);
while (1) {
    int_fast16_t res;
    res = ADC_convert(adc, &adcValue);
    if (res == ADC_STATUS_SUCCESS) {
        Display_printf(displayHandle, 1, 0, "ADC Reading %d", adcValue);
        if(adcValue >= threshold){
            GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_ON);
            trigger = 1;
        } else{
            GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_OFF);
            trigger = 0;
        }
    }
    usleep(time);
}
}

```



Lab02 Task4

Youtube Link: <https://youtu.be/LzrsFeVzCUw>

```
#include <unistd.h>
#include <stdint.h>
#include <stddef.h>
/* Driver Header files */
#include <ti/drivers/GPIO.h>
#include <ti/drivers/ADC.h>
#include <ti/display/Display.h>
// #include <ti/drivers/I2C.h>
// #include <ti/drivers/SDSPI.h>
// #include <ti/drivers/SPI.h>
// #include <ti/drivers/UART.h>
// #include <ti/drivers/Watchdog.h>
/* Board Header file */
#include "Board.h"
/* GLOBAL VARIABLES FOR GUI COMPOSER */
uint16_t adcValue = 0;
uint16_t threshold = 100;
uint16_t trigger = 0;
/*
 * ===== gpioButtonFxn0 =====
 * Callback function for the GPIO interrupt on Board_GPIO_BUTTON0.
 */
void gpioButtonFxn0(uint_least8_t index)
{
    /* Clear the GPIO interrupt and decrement threshold */
    if(threshold < 250){ // Ensure threshold doesn't go below zero
        threshold = 0;
    }
}
```

```

    } else {
        threshold -= 250; // decrement by 250
    }
}
/*
 * ===== gpioButtonFxn1 =====
 * Callback function for the GPIO interrupt on Board_GPIO_BUTTON1.
 * This may not be used for all boards.
 */
void gpioButtonFxn1(uint_least8_t index)
{
    /* Clear the GPIO interrupt and increment threshold */
    if(threshold > 16133){ // Ensure threshold doesn't go above max ADC range
        threshold = 16383;
    } else {
        threshold += 250; // increment by 250
    }
}
/*
 * ===== mainThread =====
 */
void *mainThread(void *arg0)
{
    /* ~10 loops/second */
    uint32_t time = 100000;
    /* Call driver init functions */
    GPIO_init();
    ADC_init();
    // I2C_init();
    // SDSPI_init();
    // SPI_init();
    // UART_init();
    // Watchdog_init();
    /* Open Display Driver */
    Display_Handle displayHandle;
    Display_Params displayParams;
    Display_Params_init(&displayParams);
    displayHandle = Display_open(Display_Type_UART, NULL);
    /* Open ADC Driver */
    ADC_Handle adc;
    ADC_Params params;
    ADC_Params_init(&params);
    adc = ADC_open(Board_ADC0, &params);
    if (adc == NULL) {
        // Error initializing ADC channel 0
        while (1);
    }

    GPIO_setConfig(Board_GPIO_BUTTON0, GPIO_CFG_IN_PU | GPIO_CFG_IN_INT_FALLING);
    GPIO_setConfig(Board_GPIO_BUTTON1, GPIO_CFG_IN_PU | GPIO_CFG_IN_INT_FALLING);
    /* install Button callback */
    GPIO_setCallback(Board_GPIO_BUTTON0, gpioButtonFxn0);
    GPIO_setCallback(Board_GPIO_BUTTON1, gpioButtonFxn1);
    /* Enable interrupts */
    GPIO_enableInt(Board_GPIO_BUTTON0);

```

```

GPIO_enableInt(Board_GPIO_BUTTON1);
while (1) {
    int_fast16_t res;
    res = ADC_convert(adc, &adcValue);
    if (res == ADC_STATUS_SUCCESS) {
        Display_printf(displayHandle, 1, 0, "ADC Reading %d", adcValue);
        if(adcValue >= threshold){
            GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_ON);
            trigger = 1;
        } else{
            GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_OFF);
            trigger = 0;
        }
    }
    usleep(time);
}
}

```

