Tenniel Takenaka-Fuller
Github root directory: https://github.com/TennielTakenaka/sturdy-carnival/tree/master/Lab2

Date Submitted: **11/10/19**

Task 01:

Youtube Link: https://youtu.be/hnmQxZ1FLU0

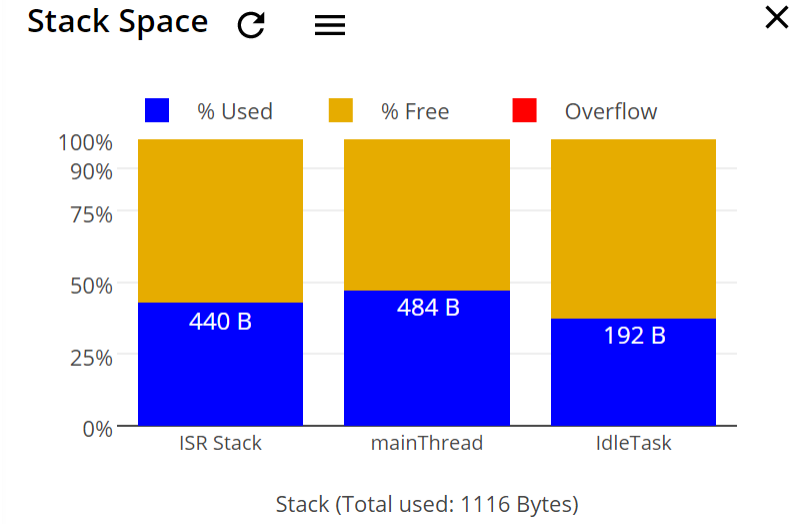
Sample output:

```
m ADC Reading 716
m ADC Reading 690
m ADC Reading 662
m ADC Reading 642
m ADC Reading 642
m ADC Reading 655
m ADC Reading 681
m ADC Reading 702
m ADC Reading 715
m ADC Reading 708
m ADC Reading 678
m ADC Reading 648
m ADC Reading 634
m ADC Reading 632
m ADC Reading 654
m ADC Reading 684
m ADC Reading 706
m ADC Reading 712
```

Stack space:

Task Detailed

address	label	priority	mode	fxn	arg0	arg1	stackPeak	stackSize	stackBase	curCoreId
0x2000150c	ti.sysbios.knl.Task.IdleTask	0	Running	ti_sysbios_knl_Idle_loop__E	0x0	0x0	192	512	0x200017c8	n/a
0x20000268		1	Blocked	mainThread	0x26f5	0x20000208	484	1024	0x200002b8	n/a



Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Tenniel Takenaka-Fuller
Github root directory: https://github.com/TennielTakenaka/sturdy-carnival/tree/master/Lab2

Log variables using ROV:

Hwi Detailed

address	halHwiHandle	label	type	intNum	priority	group	subPriority	fxn	arg	irp	status	coreId
0x20001700			Dispatched	20	224	7	0	ti_sysbios_family_arm_cc26xx_Timer_dynamicStub__E	0x0	0x2228	Enabled	0
0x20001214			Dispatched	16	224	7	0	PIN_hwi	0x0	0x4a8c	Enabled	0
0x20001a44			Dispatched	21	224	7	0	UARTCC26XX_hwiIntFxn	0x5b38	0x0	Enabled	0
0x200013b8			Dispatched	44	224	7	0	PowerCC26XX_auxISR	0x0	0x0	Enabled	0

Modified Code:

```
/* 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/SPI.h>
// #include <ti/drivers/UART.h>
// #include <ti/drivers/Watchdog.h>
#include <ti/drivers/ADC.h>
#include <ti/display/Display.h>

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


// Global Variables
uint16_t adcValue = 0;
uint16_t threshold = 100;
uint16_t trigger = 0;

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
    }
}


void gpioButtonFxn1(uint_least8_t index)
{
    /* Clear the GPIO interrupt and increment threshold */
    if(threshold > 4096){ // Ensure threshold doesn't go above max ADC range
        threshold = 4096;
    } else {
        threshold += 250; // increment by 250
    }
}

/*
 *  ======== mainThread ========
 */
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
void *mainThread(void *arg0)
{
    /* 1 second delay */
    uint32_t time = 100000;

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

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

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

    // ADC
    ADC_Handle adc;
    ADC_Params params;
    ADC_Params_init(&params);
    adc = ADC_open(Board_ADC0, &params);
    if (adc == NULL) {
        // ADC_open() failed
        while (1);
    }

    // UART
    Display_Handle    displayHandle;
    Display_Params    displayParams;
    Display_Params_init(&displayParams);
    displayHandle = Display_open(Display_Type_UART, NULL);

    /* 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);
        Display_printf(displayHandle, 1, 0, "ADC Reading %d", adcValue);
        Display_printf(displayHandle, 1, 0, "Threshold %d", threshold);

        if (res == ADC_STATUS_SUCCESS) {
            if(adcValue >= threshold){  // arbitrary threshold
                GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_ON);
                trigger = 1;
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

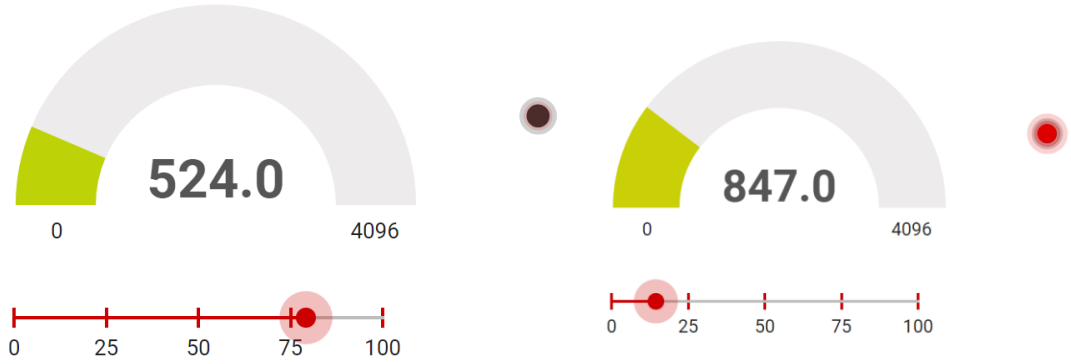
```
            } else{
                GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_OFF);
                trigger = 0;
            }
        }

        usleep(time);
    }
}
```

Task 02:

Youtube Link: https://youtu.be/3F80qL9KU4Y

Code is unchanged.



Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.