CPE403 – Advanced Embedded Systems

Design Assignment 3

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Rishawn Peppers Johnson

Email: peppersj@unlv.nevada.edu

Github Repository link (root): https://github.com/PeppersJ/v4e0nk_i3

Youtube Playlist link (root): https://drive.google.com/drive/u/2/folders/1fJ029-AAWjTnN-QrRqNLd0iLwKGm6A08

Follow the submission guideline to be awarded points for this Assignment.

Submit the following for all Assignments:

1. In the document, for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only.
2. Create a private Github repository with a random name (no CPE/403, Lastname, Firstname). Place all labs under the root folder TIVAC, sub-folder named Assignment1, with one document and one video link file for each lab, place modified c files named as asng_taskxx.c.
3. If multiple c files or other libraries are used, create a folder asng1_t01 and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) with startup_ccs.c and other include files, c) text file with youtube video links (see template).
5. Submit the doc file in canvas before the due date. The root folder of the github assignment directory should have the documentation and the text file with youtube video links.
6. Organize your youtube videos as playlist under the name "cpe403". The playlist should have the video sequence arranged as submission or due dates.
7. Only submit pdf documents. Do not forget to upload this document in the github repository and in the canvas submission portal.

1. Code for Tasks. for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only. Use separate page for each task.

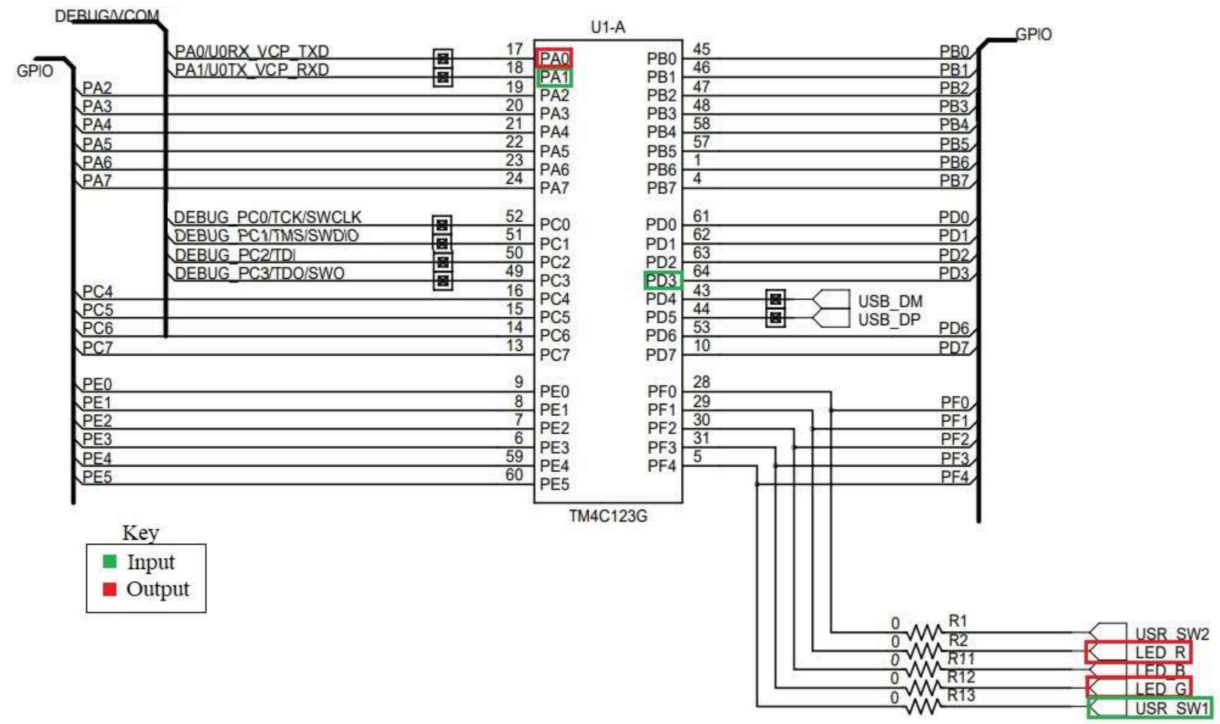
```
/*  Modified By: Rishawn Peppers Johnson
 *  Date Created: 2 November 2020
 *  Device: TivaC123GH6PM
 *  CpE 403 Assignment 03
 *
 *  Purpose: Interface the TivaC123GH6PM with the Educational BoosterPack MKII to
 *      read the MKII's vertical joystick value using TivaC's ADC, display the ADC
 *      value to console through UART, and when a switch is pressed, update the PWM
 *      duty cycle routed to an LED.
 *
 *  Inputs:     MKII vertical joystick
 *              Switch
 *  Outputs:    UART to console ADC value
 *              PWM value to LED
 *  */
/*
 * Copyright (c) 2015, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

#include <stdint.h>     // Variable definitions for C99 Standard
#include <stdbool.h>    // Boolean definitions for the C99 Standard
#include <stdio.h>      // Common C functions
```

```c
/* XDCtools Header files */
#include <xdc/std.h>
#include <xdc/cfg/global.h>
#include <xdc/runtime/System.h>

/* BIOS Header files */
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/sysbios/hal/Timer.h>
#include <ti/sysbios/knl/Semaphore.h>
#include <ti/sysbios/interfaces/ITimer.h>

/* TI-RTOS Header files */
#include <ti/drivers/GPIO.h>
#include <ti/drivers/UART.h>
#include <ti/drivers/PWM.h>
#include "driverlib/sysctl.h"   // Driver for SysTick
#include "driverlib/adc.h"      // Drivers for ADC
#include "inc/hw_memmap.h"      // Macros defining the memory map

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

#define TASKSTACKSIZE   512

// Heartbeat_fxn
Task_Struct task0Struct;
Char task0Stack[TASKSTACKSIZE];

uint16_t count = 0;          // 1ms timer interrupt count
uint32_t ui32ADC0Value[1];  // raw adc value from sequencer

// Function prototypes
void hwi_int();
void timer0_ISR_fxn();
void idleFxn();
void readAdcFxn();
void Switch_ReadFxn();
void UART_DisplayFxn();

Void heartBeatFxn(UArg arg0, UArg arg1) {
// Blink led every (arg0)ms
   while (1) {
        Task_sleep((UInt)arg0);
        GPIO_toggle(Board_LED2);
    }
}

int main(void) {
    Task_Params taskParams;

    /* Call board init functions */
    Board_initGeneral();
    Board_initGPIO();
    Board_initUART();
    Board_initPWM();
    hwi_int();  // Init SYSclck and ADC

    // Create hearBeatFxn
    Task_Params_init(&taskParams);
    taskParams.arg0 = 1000;
    taskParams.stackSize = TASKSTACKSIZE;
    taskParams.stack = &task0Stack;
    taskParams.priority = 3;
    Task_construct(&task0Struct, (Task_FuncPtr)heartBeatFxn, &taskParams, NULL);

    /* Start BIOS */
    BIOS_start();

    return (0);
}

void hwi_int() {
// Initializes sysclock and ADC0
    //System clock to 40Mhz (PLL= 400Mhz / 10 = 40Mhz)
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0); // Running at default rate of 1Msps

    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);   // Using ADC0,
sequencer 1, processor triggered, highest priority
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_CH4|ADC_CTL_IE|ADC_CTL_END); //
Read from CH4
    ADCSequenceEnable(ADC0_BASE, 1);    // Enable Sequencer 1
}

void timer0_ISR_fxn() {
// Timer 0 interrupt routine
// Runs every 1ms
    count++;
}

void idleFxn() {
// Main loop
    Timer_start(timer0);    // Start counting Timer
    while (1) {
        if (count == 5)         // Run ADC
            Semaphore_post(adc_sem);
        else if (count == 10)   // Display ADC value on UART
            Semaphore_post(uart_sem);
        else if (count == 15) { // Update PWM value
            Semaphore_post(pwm_sem);
            count = 0;
        }
    }
}

void readAdcFxn() {
// Read ADC CH4 thru sequencer 0
    while(1) {
        // Clear status flag before writing to ADC
        ADCIntClear(ADC0_BASE, 1);
        // Set ADC to trigger with software
        ADCProcessorTrigger(ADC0_BASE, 1);
        // Wait for conversion to finish
        while(!ADCIntStatus(ADC0_BASE, 1, false)) {}
        // Copy from ADC FIFO to buffer
        ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
        Semaphore_pend(adc_sem, BIOS_WAIT_FOREVER);
    }
}
```

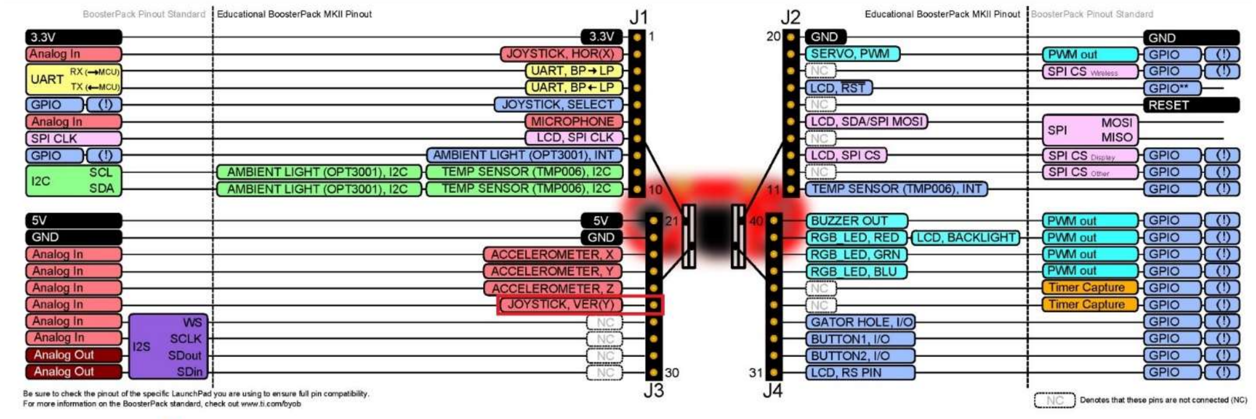
```
void Switch_ReadFxn() {
// Check if switch is pressed and update PWM duty cycle to LED
    PWM_Handle pwm1;
    PWM_Params params;
    uint16_t   pwmPeriod = 3000;      // Period and duty in microseconds
    uint16_t   duty = 0;

    PWM_Params_init(&params);
    params.period = pwmPeriod;
    pwm1 = PWM_open(Board_PWM1, &params);
    while (1) {
        if (!GPIO_read(Board_BUTTON0)) {    // Only when switch is pressed
            duty = ui32ADC0Value[0];    // 32bit to 16bit
            PWM_setDuty(pwm1, duty);
        }
        Semaphore_pend(pwm_sem, BIOS_WAIT_FOREVER);
    }
}
```

```
void UART_DisplayFxn(UArg arg0, UArg arg1){
//Output ADC value to console using UART
    UART_Handle uart;
    UART_Params uartParams;
    const char echoPrompt[] = "Terminal Active:\r\n";

    // Create a UART with data processing off.
    UART_Params_init(&uartParams);
    uartParams.writeDataMode = UART_DATA_BINARY;
    uartParams.readDataMode = UART_DATA_BINARY;
    uartParams.readReturnMode = UART_RETURN_FULL;
    uartParams.readEcho = UART_ECHO_OFF;
    uartParams.baudRate = 115200;
    uart = UART_open(Board_UART0, &uartParams);

    UART_write(uart, echoPrompt, sizeof(echoPrompt)); // Entry prompt

    char adcValue[6];
    uint32_t clearCount = 0;    // Count till clearing console
    // Loop forever echoing
    while (1) {
        sprintf(adcValue, "%d\r", ui32ADC0Value[0]);  // Convert int to string
        UART_write(uart, adcValue, sizeof(adcValue));
        if(clearCount == 80) {
            UART_write(uart, "     \r", sizeof(adcValue));  // Clear old value
            clearCount = 0;
        }
        clearCount++;
        Semaphore_pend(uart_sem, BIOS_WAIT_FOREVER);
    }
}
```

2. Block diagram and/or Schematics showing the components, pins used, and interface.



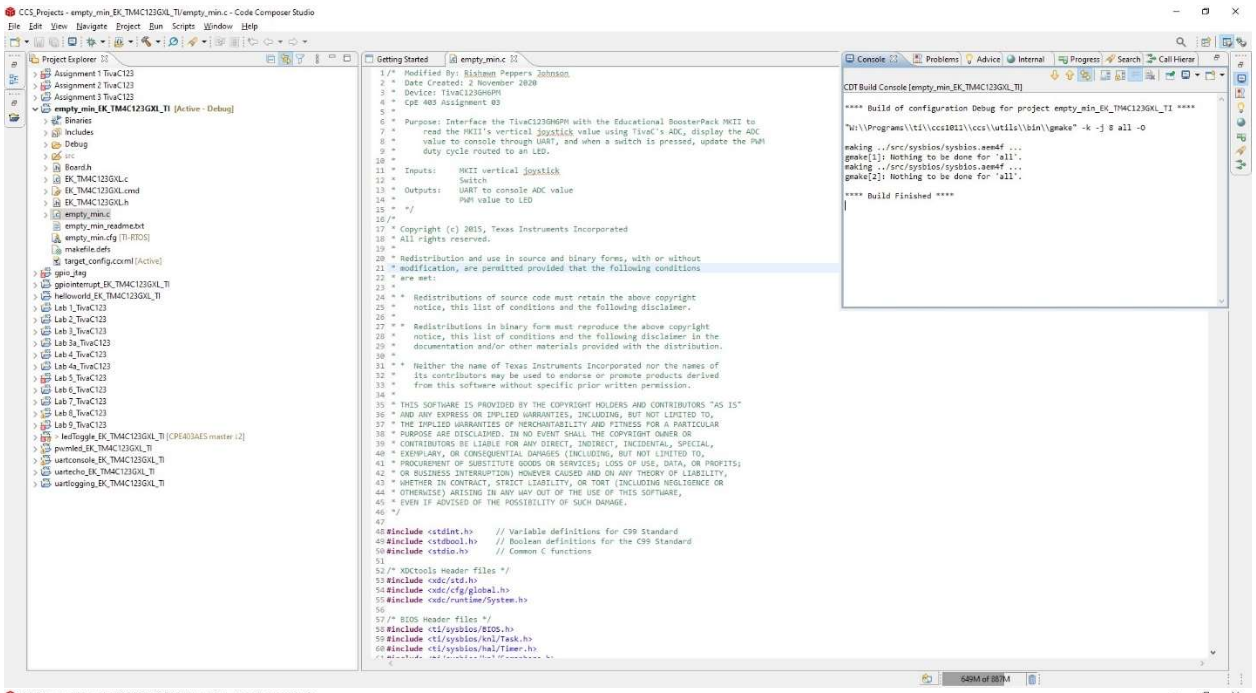
Pinout Diagram for your BoosterPack



🟥 Output

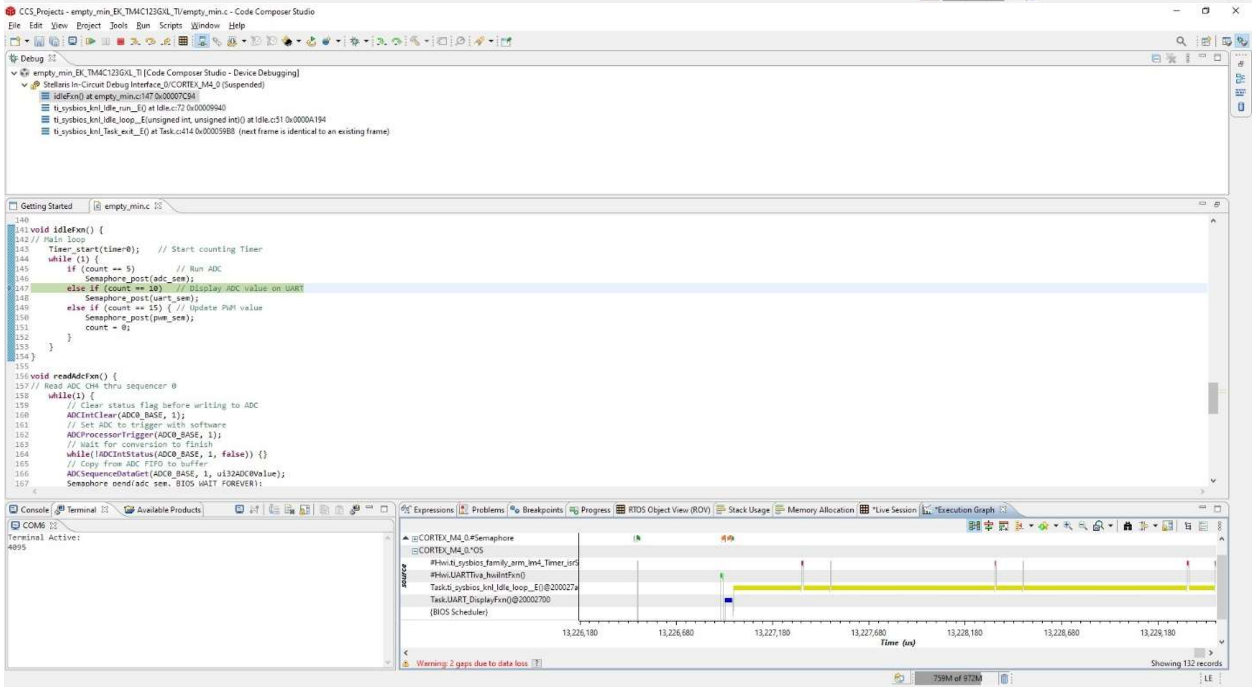
Figure 3. BoosterPack Plug-in Module Pinout

3. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.



```c
/*  Modified By: Rishawn Peppers Johnson
 *  Date Created: 2 November 2020
 *  Device: TivaC123GH6PM
 *  CpE 403 Assignment 03
 *
 *  Purpose: Interface the TivaC123GH6PM with the Educational BoosterPack MKII to
 *      read the MKII's vertical joystick value using TivaC's ADC, display the ADC
 *      value to console through UART, and when a switch is pressed, update the PWM
 *      duty cycle routed to an LED.
 *
 *  Inputs:     MKII vertical joystick
 *              Switch
 *  Outputs:    UART to console ADC value
 *              PWM value to LED
 *  */
/*
 * Copyright (c) 2015, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

#include <stdint.h>     // Variable definitions for C99 Standard
#include <stdbool.h>    // Boolean definitions for the C99 Standard
#include <stdio.h>      // Common C functions

/* XDCtools Header files */
#include <xdc/std.h>
#include <xdc/cfg/global.h>
#include <xdc/runtime/System.h>

/* BIOS Header files */
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/sysbios/hal/Timer.h>
```

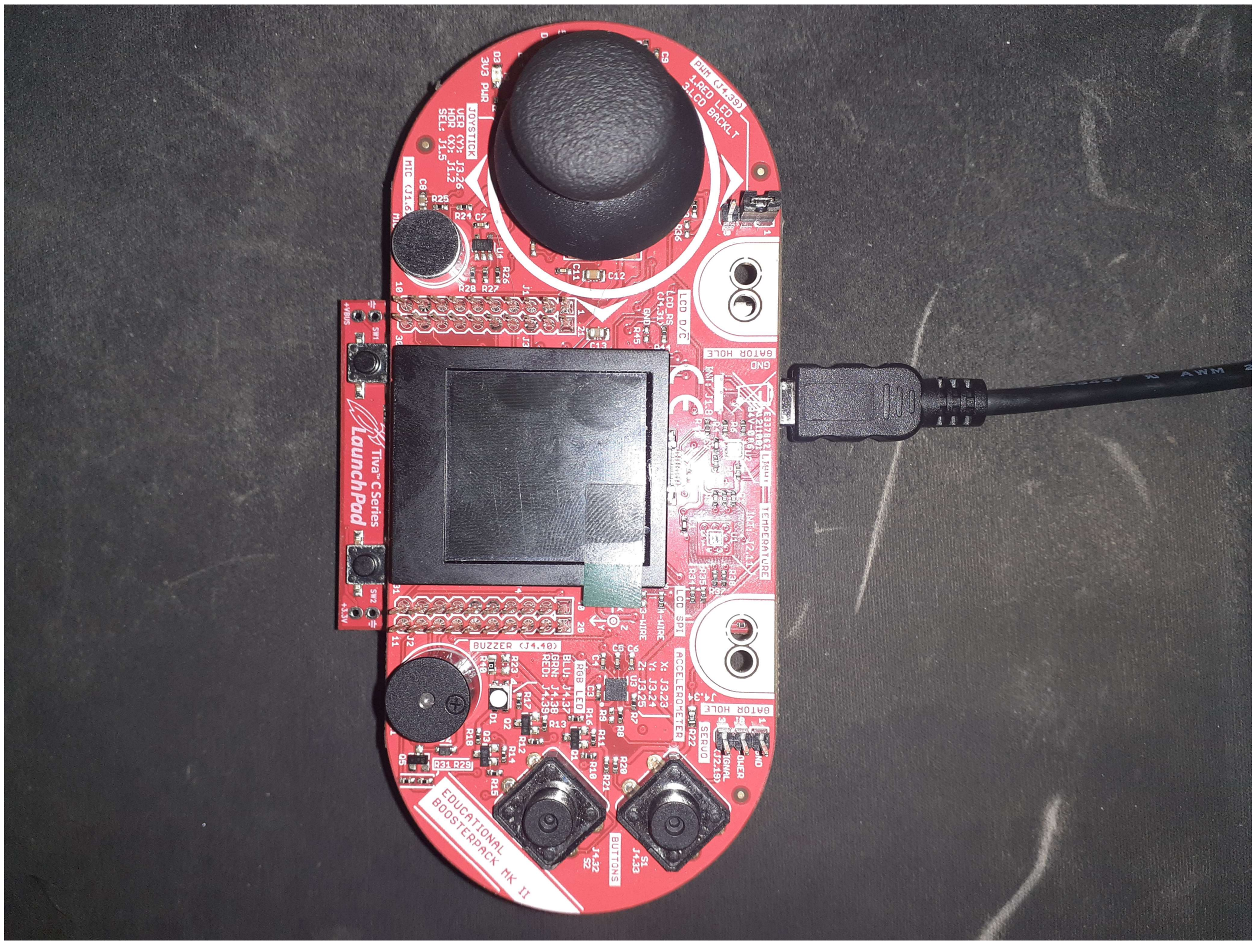
```
**** Build of configuration Debug for project empty_min_EK_TM4C123GXL_TI ****

"W:\\Programs\\ti\\ccs1011\\ccs\\utils\\bin\\gmake" -k -j 8 all -O

making ../src/sysbios/sysbios.aem4f ...
gmake[1]: Nothing to be done for 'all'.
making ../src/sysbios/sysbios.aem4f ...
gmake[2]: Nothing to be done for 'all'.

**** Build Finished ****
```



```
empty_min_EK_TM4C123GXL_TI [Code Composer Studio - Device Debugging]
  Stellaris In-Circuit Debug Interface_0/CORTEX_M4_0 (Suspended)
    idleFxn() at empty_min.c:147 0x00007C94
    ti_sysbios_knl_Idle_run__E() at Idle.c:72 0x00009940
    ti_sysbios_knl_Idle_loop__E(unsigned int, unsigned int)() at Idle.c:51 0x0000A194
    ti_sysbios_knl_Task_exit__E() at Task.c:414 0x000059B8  (next frame is identical to an existing frame)
```

```c

void idleFxn() {
// Main loop
    Timer_start(timer0);    // Start counting Timer
    while (1) {
        if (count == 5)         // Run ADC
            Semaphore_post(adc_sem);
        else if (count == 10)   // Display ADC value on UART
            Semaphore_post(uart_sem);
        else if (count == 15) { // Update PWM value
            Semaphore_post(pwm_sem);
            count = 0;
        }
    }
}

void readAdcFxn() {
// Read ADC CH4 thru sequencer 0
    while(1) {
        // Clear status flag before writing to ADC
        ADCIntClear(ADC0_BASE, 1);
        // Set ADC to trigger with software
        ADCProcessorTrigger(ADC0_BASE, 1);
        // Wait for conversion to finish
        while(!ADCIntStatus(ADC0_BASE, 1, false)) {}
        // Copy from ADC FIFO to buffer
        ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
        Semaphore_pend(adc_sem, BIOS_WAIT_FOREVER);
```

```
Terminal Active:
4095
```



4. Declaration

   I understand the Student Academic Misconduct Policy - http://studentconduct.unlv.edu/misconduct/policy.html

“This assignment submission is my own, original work”.
Rishawn Peppers Johnson