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.
8. 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

/\* 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;

}

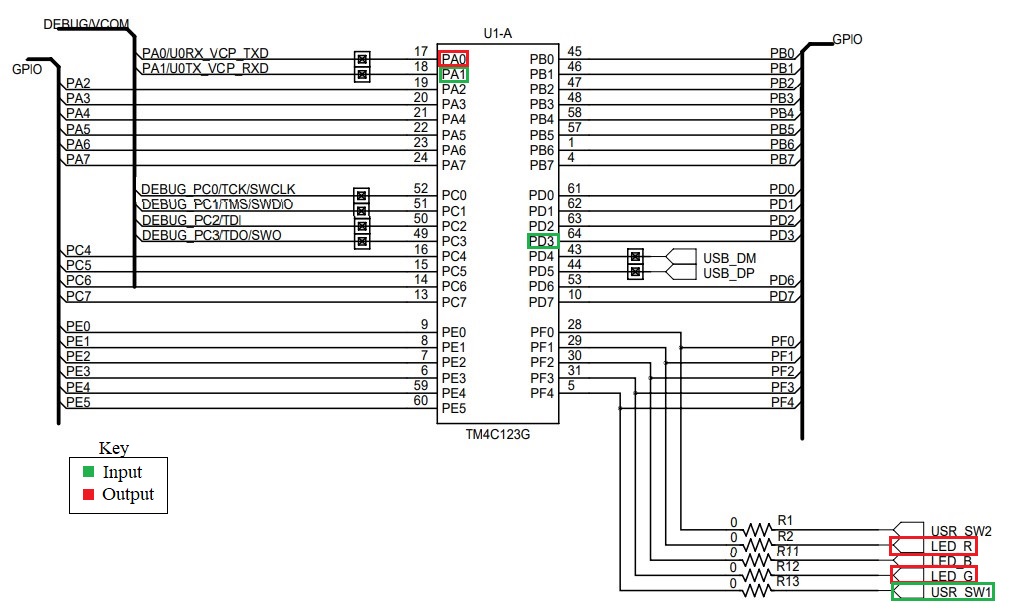
clearCount++;

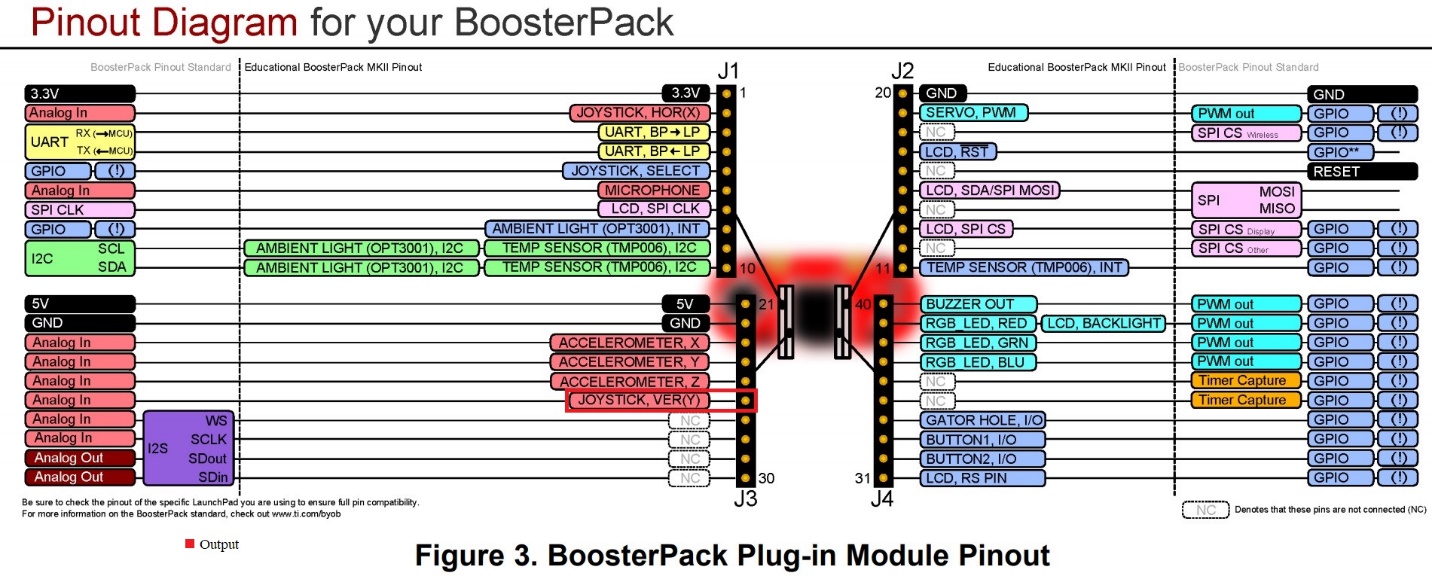
Semaphore\_pend(uart\_sem, BIOS\_WAIT\_FOREVER);

}

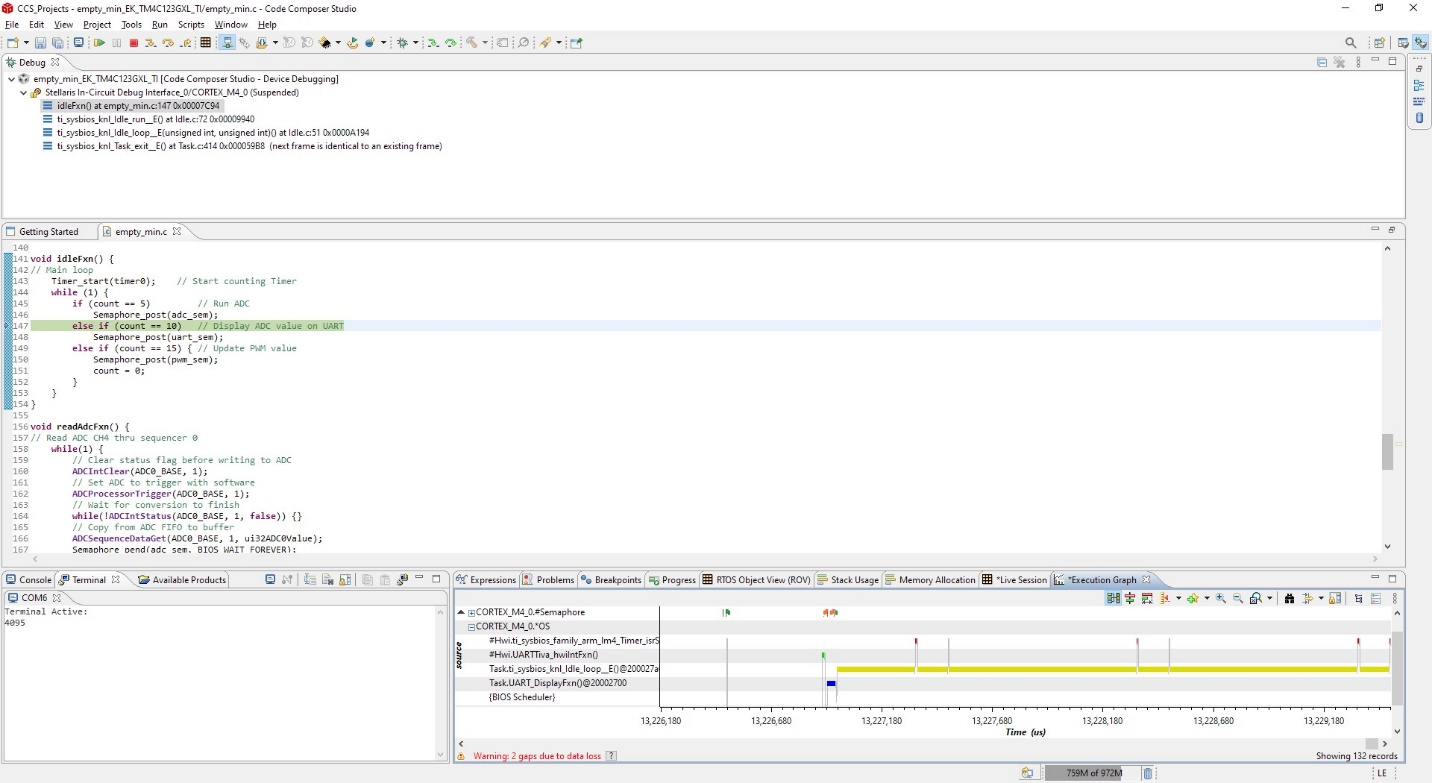
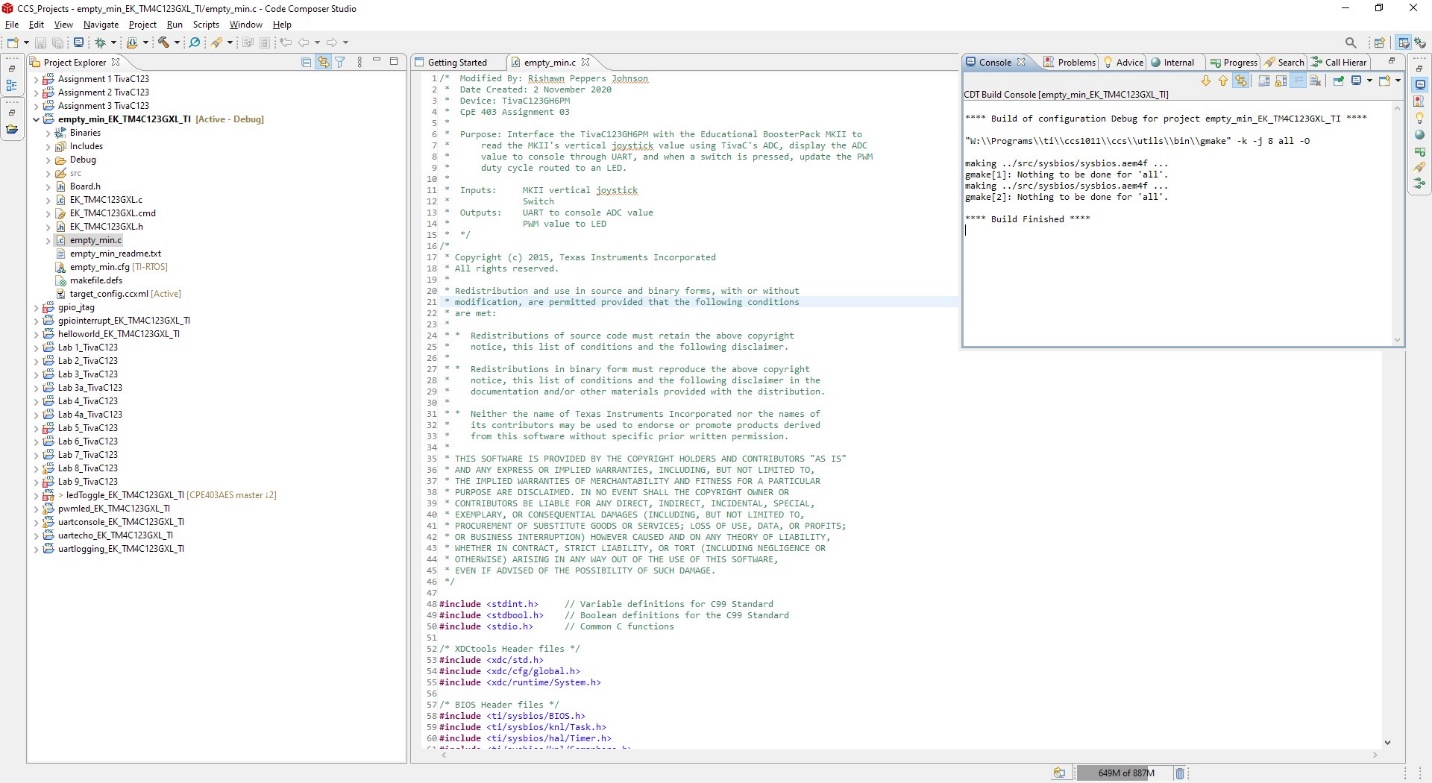
}

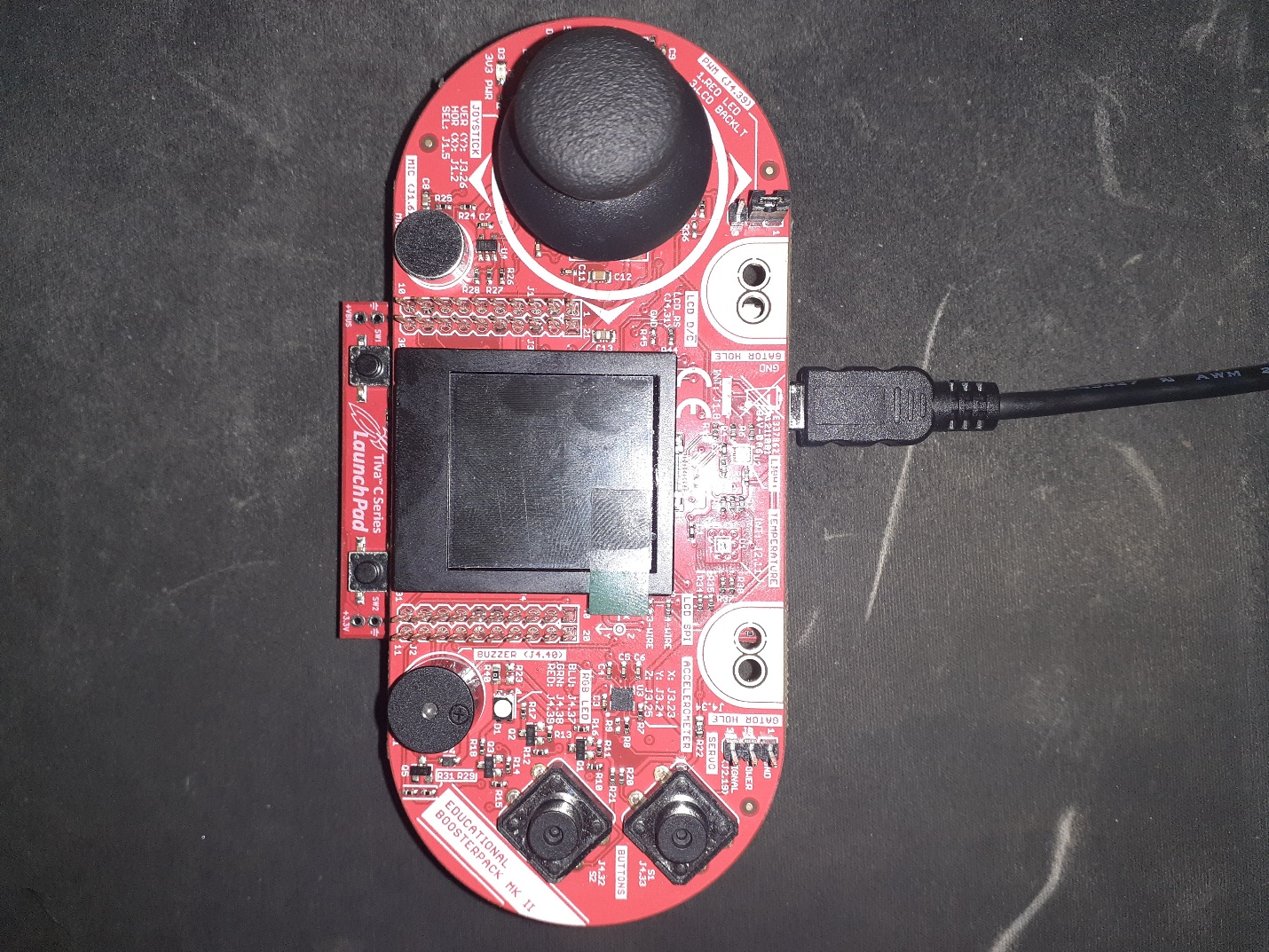
1. Block diagram and/or Schematics showing the components, pins used, and interface.





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





1. 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