CPE403 – Advanced Embedded Systems

Design Assignment #3

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name:Cade Echevary

Email:echevary@unlv.nevada.edu

Github Repository link (root):
https://github.com/echevary/MicroController_proj/tree/master/TIRTOS/TIRTOS_Assignments

Youtube Playlist link (root): couldn't get compilation

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.

```
/*
 * 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.
// */
//


//---------------------------------------
// BIOS header files
//---------------------------------------
//#include <xdc/std.h>                        //mandatory - have to include
first, for BIOS types
#include <ti/sysbios/BIOS.h>                //mandatory - if you call APIs
like BIOS_start()
#include <xdc/runtime/Log.h>                //needed for any Log_info() call
#include <xdc/cfg/global.h>                 //header file for statically
defined objects/handles


//---------------------------------------
```

```
47. // TivaWare Header Files
48. //---------------------------------------
49. #include <stdint.h>
50. #include <stdbool.h>
51. 
52. #include "inc/hw_types.h"
53. #include "inc/hw_memmap.h"
54. #include "driverlib/sysctl.h"
55. #include "driverlib/gpio.h"
56. #include "inc/hw_ints.h"
57. #include "driverlib/interrupt.h"
58. #include "driverlib/timer.h"
59. #include "driverlib/adc.h"
60. #include "utils/uartstdio.h"
61. #include "driverlib/uart.h"
62. #include "driverlib/pin_map.h"
63. #include "driverlib/pwm.h"
64. 
65. //---------------------------------------
66. // Function Prototypes
67. //---------------------------------------
68. void hardware_init(void);
69. void HWI_Timer(void);
70. void adcTaskFxn(void);
71. void swReadTaskFxn(void);
72. void uartTaskFxn(void);
73. void InitConsole(void);
74. void ConfigureHeartBeat(void);
75. Void heartBeatFxn(UArg arg0, UArg arg1);
76. 
77. //---------------------------------------
78. // Define stmts and Global Variables
79. //---------------------------------------
80. 
81. #define PWM_FREQUENCY 55    // PWM frequency set to 55Hz
82. 
83. volatile int16_t counter;
84. uint32_t ui32ADC0Value[4];
85. uint32_t ui32ADCAvg;
86. uint32_t ui32Adjust;
87. volatile uint32_t ui32Load;
88. volatile uint32_t ui32PWMClock;
89. 
90. 
91. 
92. 
93. /*main function*/
94. 
95. int main(void)
96. {
97. 
98.     hardware_init(); // call function to initialize the hardware
99. 
100.         /* Start BIOS */
101.         BIOS_start();
```

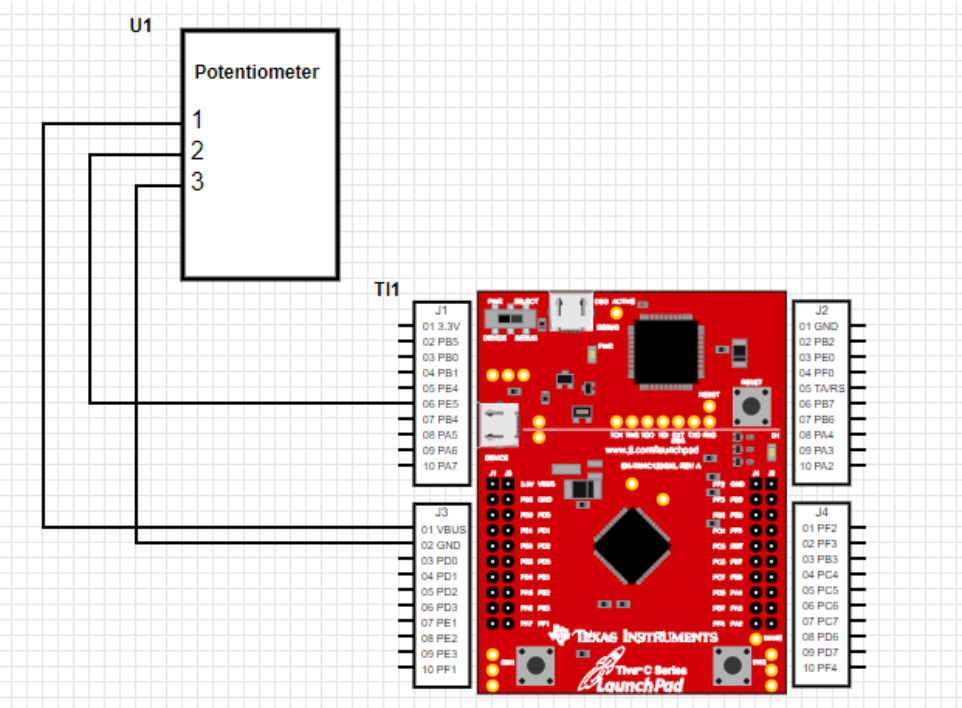
```
102.
103.      }
104.
105.
106.      void InitConsole(void){
107.          //Enable GPIO port A for UART pins
108.          SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
109.
110.          //Configure UART pins for Rx and Tx
111.          GPIOPinConfigure(GPIO_PA0_U0RX);
112.          GPIOPinConfigure(GPIO_PA1_U0TX);
113.
114.          //Enable UART0.
115.          SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
116.
117.          //Use the internal 16MHz oscillator
118.          UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);
119.
120.          //Select the alternate (UART) function for these pins
121.          GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
122.
123.          //Initialize the UART
124.          UARTStdioConfig(0, 115200, 16000000);
125.      }
126.
127.
128.      //-------------------------------------------------------------------
    -----
129.      // hardware_init()
130.      //
131.      // inits GPIO pins for toggling the LED
132.      //-------------------------------------------------------------------
    -----
133.      void hardware_init(void)
134.      {
135.          uint32_t ui32Period;
136.
137.          counter = 0; // initialize counter to 0
138.
139.
140.
141.          //Set CPU Clock to 40MHz. 400MHz PLL/2 = 200 DIV 5 = 40MHz
142.
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAI
    N);
143.          SysCtlPWMClockSet(SYSCTL_PWMDIV_64);
144.
145.          SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
146.          SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
147.
148.          // ADD Tiva-C GPIO setup - enables port, sets pins 1-3 (RGB) pins
    for output
149.          SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
150.          SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE); // enable analog input
    3 (PE0)
```

```
151.
152.          GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
   GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
153.          GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_0|GPIO_PIN_4);
154.          GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_0); // use PE0 (AIN3 -
   channel 3) for potentiometer
155.
156.          GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_0|GPIO_PIN_4,
   GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
157.
158.
159.          //initialize PWM
160.          ui32PWMClock = SysCtlClockGet() / 64;
161.          ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
162.
163.          GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0); //PD0 PWM pin
164.          GPIOPinConfigure(GPIO_PD0_M1PWM0);
165.
166.          PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
167.          PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);
168.
169.          PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
170.          PWMGenEnable(PWM1_BASE, PWM_GEN_0);
171.
172.
173.          //initialize ADC
174.          SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
175.          ADCHardwareOversampleConfigure(ADC0_BASE, 64);
176.          ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
177.
178.          // using channel 3 for the ADC samples
179.          ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_CH3);
180.          ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_CH3);
181.          ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_CH3);
182.          ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_CH3 | ADC_CTL_IE |
   ADC_CTL_END);
183.
184.          ADCSequenceEnable(ADC0_BASE, 1);
185.
186.          // Initialize Timer 2 for the HWI
187.          SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER2);
188.          TimerConfigure(TIMER2_BASE, TIMER_CFG_PERIODIC);
189.
190.          ui32Period = (SysCtlClockGet() / 500);  // period is around 1ms
191.          TimerLoadSet(TIMER2_BASE, TIMER_A, ui32Period);
192.
193.          TimerIntEnable(TIMER2_BASE, TIMER_TIMA_TIMEOUT);
194.
195.          TimerEnable(TIMER2_BASE, TIMER_A);
196.
197.
198.          // call function to initialize UART
199.          InitConsole();
200.
201.      }
```

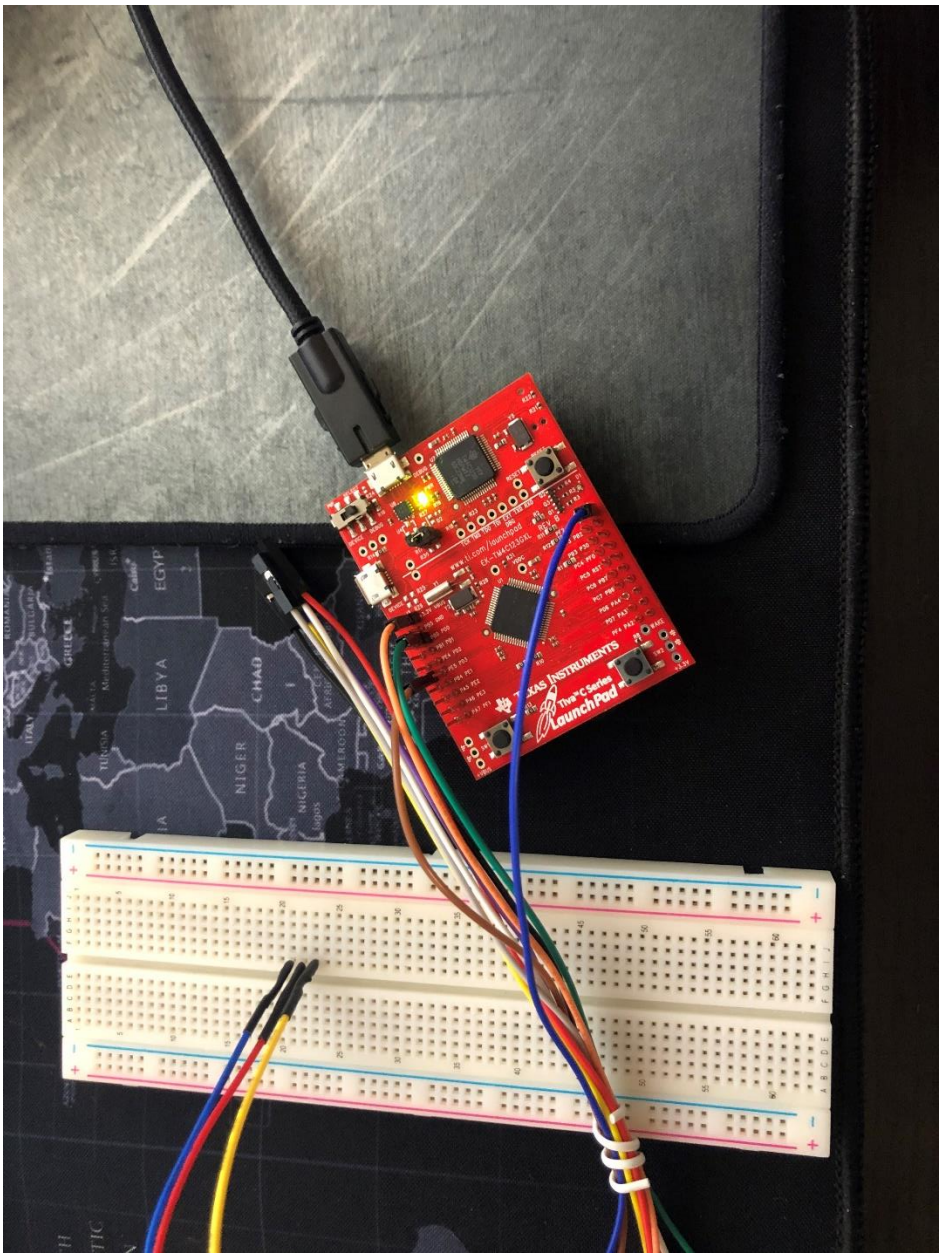
```
202.
203.
204.      void ConfigureHeartBeat(void)
205.       {
206.       SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);// enables the gpiof
207.       GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);
208.       }
209.
210.      void adcTaskFxn(void){
211.      // read ADC value, store into variable,
212.      // set the pulse width according to the ADC value
213.          while(1){
214.
215.              ADCIntClear(ADC0_BASE, 1);
216.              ADCProcessorTrigger(ADC0_BASE, 1);
217.
218.              while (!ADCIntStatus(ADC0_BASE, 1, false)) {}
219.
220.              ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
221.
222.              ui32ADCAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] +
    ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
223.              ui32Adjust = ui32ADCAvg; // store ADC avg value into the
    ui32Adjust variable
224.
225.              Semaphore_pend (sem_ADC, BIOS_WAIT_FOREVER);
226.          }
227.      }
228.
229.
230.      void uartTaskFxn(void){
231.      // display the current ADC value on terminal
232.          while(1){
233.
234.              UARTprintf("ADC Value: %d\n", ui32Adjust);
235.              Semaphore_pend (sem_UART, BIOS_WAIT_FOREVER);
236.          }
237.      }
238.
239.
240.      void swReadTaskFxn(void){
241.      // when the switch is pressed, the duty cycle of the PWM
242.      // changes according to the ADC value
243.          while(1){
244.              // if switch 1 is pressed down...
245.              if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4)==0x00)
246.              {
247.                  // set and adjust the width of the PWM using the ui32Adjust
    value
248.                  PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui32Adjust);
249.              }
250.              Semaphore_pend (sem_swRead, BIOS_WAIT_FOREVER);
251.
252.          }
253.      }
```

```
254.
255.       Void heartBeatFxn(UArg arg0, UArg arg1)
256.       {
257.       while (1) {
258.       Task_sleep((UInt)arg0);
259.       GPIO_toggle(Board_LED0);
260.       }
261.       }
262.
263.      void HWI_Timer(void){
264.      // HWI executes every 1ms
265.      // at every 5th instance, ADC task is executed
266.      // at every 10th instance, UART task is executed
267.      // at every 15th instance, swRead task is executed and reset the counter
268.          TimerIntClear(TIMER2_BASE, TIMER_TIMA_TIMEOUT); // clear Timer
    interrupt
269.          counter++; // increment counter every time HWI occurs
270.
271.
272.          // every time the pulse is high, turn on LED, else, turn off LED
273.          // the duration of the time that is high depends on the pulse width
    value
274.          // that was set in the sw read function
275.          if(GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0))
276.          {
277.              GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
    4);
278.          }
279.          else
280.          {
281.              GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
282.          }
283.
284.
285.          // execute ADC
286.          if (counter == 5){
287.              Semaphore_post (sem_ADC);
288.          }
289.
290.          // execute UART and display current ADC value
291.          else if (counter == 10){
292.              Semaphore_post (sem_UART);
293.          }
294.
295.          // execute sw Read task and read if the switch is pressed or not
296.          // if pressed, change the pwm pulse width according to the ADC value
297.          else if (counter == 15){
298.              Semaphore_post (sem_swRead);
299.              counter = 0; // reset counter
300.          }
301.
302.      }
```

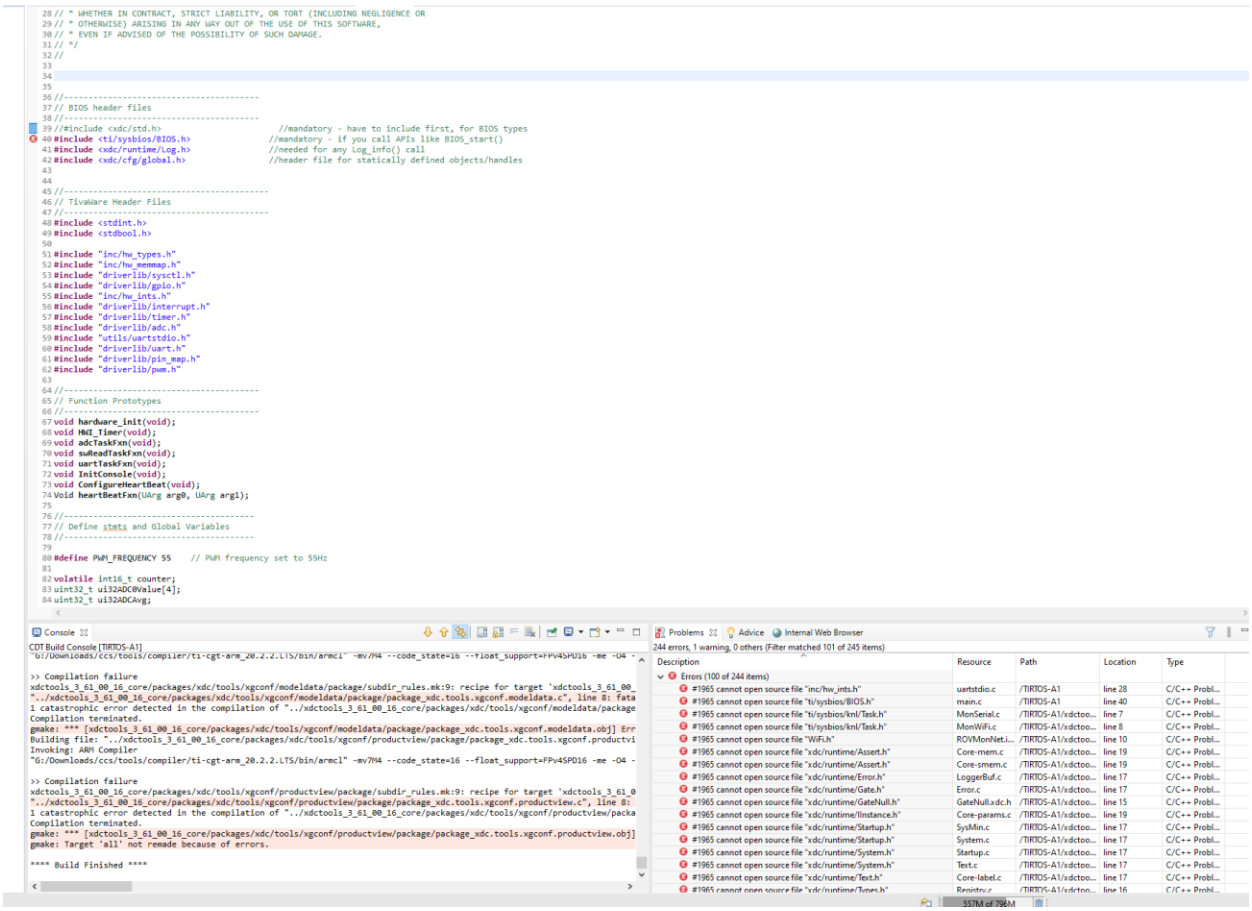
303. Block diagram and/or Schematics showing the components, pins used, and interface.



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



WE DIDN’T GET A POTENTIOMETER.



I couldn't get the code to compile due to not being able to link the xdc tool. So it threw 200 compilation errors.

305. Declaration

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

"This assignment submission is my own, original work".
Cade Echevary