CPE403 – Advanced Embedded Systems

Design Assignment #5

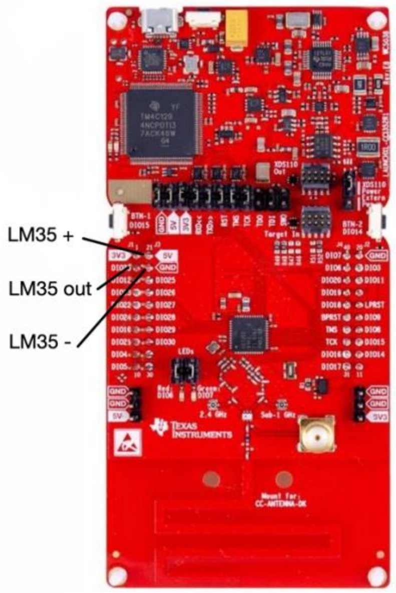
Name: Brian Wolak

Email: wolak@unlv.nevada.edu

GitHub Repository link (root): https://github.com/brianwolak/advanced_submissions

YouTube Playlist (root): https://youtube.com/playlist?list=PLl6a3M--0IcYnZIMGyucOLe8c-16wAClN

Goal of this assignment is to create a custom BLE profile to transmit any sensor data to a generic BLE app in Android or iPhone. A recommended BLE app is LightBlue by PunchThrough. The sensor data can be analog or i2c sensors (no digital sensors). The video should clearly demonstrate the BLE connectivity, services advertised, and data update on the BLE app. You could use the CC1352R1LP or CC1352R1STK for this assignment.



CC1352R1 Board Connections

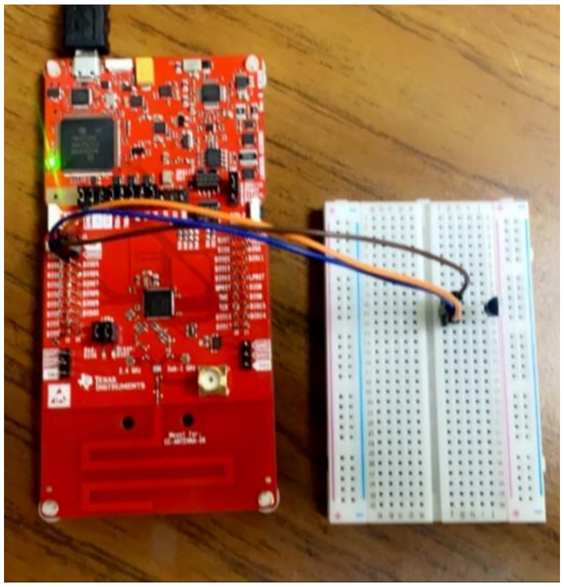
C Code:

```
/******************************************************************************

   @file  main.c

   @brief main entry of the BLE stack sample application.

   Group: WCS, BTS
   Target Device: cc13x2_26x2
```

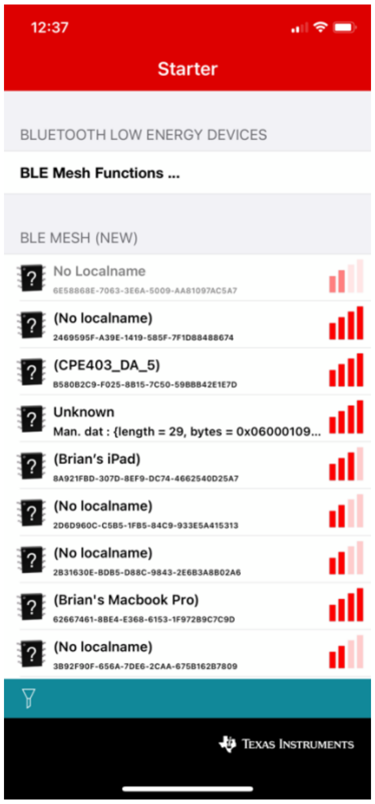
```c
 ******************************************************************************

 Copyright (c) 2013-2021, 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.

 ******************************************************************************


 *****************************************************************************/

/*******************************************************************************
 * INCLUDES
 */
#include <stdint.h>

#include <xdc/runtime/Error.h>

#include <ti/sysbios/knl/Clock.h>
#include <ti/drivers/Power.h>
#include <ti/drivers/power/PowerCC26XX.h>
#include <ti/sysbios/BIOS.h>
#include <ti/drivers/UART.h>

// Comment this in to use xdc.runtime.Log, but also remove UartLog_init below.
//#include <xdc/runtime/Log.h>
#include <ti/common/cc26xx/uartlog/UartLog.h> // Comment out to use xdc Log.
```

```
#include <common/cc26xx/flash_interface/flash_interface.h>
#include "find_stack_entry.h"

#include <icall.h>
#include "hal_assert.h"
#include "bcomdef.h"
#include "project_zero.h"

#ifndef USE_DEFAULT_USER_CFG
#include "ble_user_config.h"
// BLE user defined configuration
icall_userCfg_t user0Cfg = BLE_USER_CFG;
#endif // USE_DEFAULT_USER_CFG

/*******************************************************************************
 * MACROS
 */

/*******************************************************************************
 * CONSTANTS
 */

/*******************************************************************************
 * TYPEDEFS
 */

/*******************************************************************************
 * LOCAL VARIABLES
 */

/*******************************************************************************
 * GLOBAL VARIABLES
 */

// The stack's image header is found at runtime, available for entire app to use
const imgHdr_t *stackImageHeader = NULL;

/*******************************************************************************
 * EXTERNS
 */

extern void AssertHandler(uint8_t assertCause,
                          uint8_t assertSubcause);

/*******************************************************************************
 * @fn          Main
 *
 * @brief       Application Main
 *
 * input parameters
 *
 * @param       None.
 *
 * output parameters
 *
 * @param       None.
 *
 * @return      None.
 */
int main()
{
  /* Register Application callback to trap asserts raised in the Stack */
  RegisterAssertCback(AssertHandler);

  Board_initGeneral();

#if !defined( POWER_SAVING )
  /* Set constraints for Standby, powerdown and idle mode */
  // PowerCC26XX_SB_DISALLOW may be redundant
  Power_setConstraint(PowerCC26XX_SB_DISALLOW);
  Power_setConstraint(PowerCC26XX_IDLE_PD_DISALLOW);
#endif // POWER_SAVING

    /* Update User Configuration of the stack */
    user0Cfg.appServiceInfo->timerTickPeriod = Clock_tickPeriod;
    user0Cfg.appServiceInfo->timerMaxMillisecond = ICall_getMaxMSecs();

    /* Initialize the RTOS Log formatting and output to UART in Idle thread.
     * Note: Define xdc_runtime_Log_DISABLE_ALL and remove define UARTLOG_ENABLE
     *       to remove all impact of Log statements.
     * Note: NULL as Params gives 115200,8,N,1 and Blocking mode */
    UART_init();
    UartLog_init(UART_open(CONFIG_DISPLAY_UART, NULL));
    ADC_init();

    /* Initialize ICall module */
    ICall_init();

#ifndef STACK_LIBRARY
    {
        /* Find stack entry page */
        uint32_t stackAddr = findStackBoundaryAddr();

        if(stackAddr == 0xFFFFFFFF)
        {
            // If we cannot find the stack start address, exit
            ICall_abort();
        }

        /* set the stack image header based on the stack addr */
        stackImageHeader = (imgHdr_t *)stackAddr;

        /* Start tasks of external images - Priority 5 */
        const ICall_RemoteTask_t remoteTaskTbl[] =
        {
            (ICall_RemoteTaskEntry) (stackImageHeader->prgEntry),
            5,
            1000,
            &user0Cfg
        };
        /* Start tasks of external images - Priority 5 */
        ICall_createRemoteTasksAtRuntime((ICall_RemoteTask_t *) remoteTaskTbl,
                                         (sizeof(remoteTaskTbl) /
                                          sizeof(ICall_RemoteTask_t)));
    }
#else

    /* Start tasks of external images - Priority 5 */
    ICall_createRemoteTasks();
#endif

    ProjectZero_createTask();

    /* enable interrupts and start SYS/BIOS */
    BIOS_start();

    return(0);
}

/*******************************************************************************
 * @fn          AssertHandler
 *
 * @brief       This is the Application's callback handler for asserts raised
 *              in the stack.  When EXT_HAL_ASSERT is defined in the Stack Wrapper
 *              project this function will be called when an assert is raised,
 *              and can be used to observe or trap a violation from expected
 *              behavior.
 *
 *              As an example, for Heap allocation failures the Stack will raise
 *              HAL_ASSERT_CAUSE_OUT_OF_MEMORY as the assertCause and
 *              HAL_ASSERT_SUBCAUSE_NONE as the assertSubcause.  An application
 *              developer could trap any malloc failure on the stack by calling
 *              HAL_ASSERT_SPINLOCK under the matching case.
 *
 *              An application developer is encouraged to extend this function
 *              for use by their own application.  To do this, add hal_assert.c
 *              to your project workspace, the path to hal_assert.h (this can
 *              be found on the stack side). Asserts are raised by including
 *              hal_assert.h and using macro HAL_ASSERT(cause) to raise an
 *              assert with argument assertCause.  the assertSubcause may be
 *              optionally set by macro HAL_ASSERT_SET_SUBCAUSE(subCause) prior
 *              to asserting the cause it describes. More information is
 *              available in hal_assert.h.
 *
 * input parameters
 *
 * @param       assertCause    - Assert cause as defined in hal_assert.h.
 * @param       assertSubcause - Optional assert subcause (see hal_assert.h).
 *
 * output parameters
 *
 * @param       None.
 *
 * @return      None.
 */
void AssertHandler(uint8_t assertCause, uint8_t assertSubcause)
{
    Log_error2(">>>STACK ASSERT Cause 0x%02x subCause 0x%02x",
               assertCause, assertSubcause);

    // check the assert cause
    switch(assertCause)
    {
    case HAL_ASSERT_CAUSE_OUT_OF_MEMORY:
        Log_error0("***ERROR***");
        Log_error0(">> OUT OF MEMORY!");
        break;

    case HAL_ASSERT_CAUSE_INTERNAL_ERROR:
        // check the subcause
        if(assertSubcause == HAL_ASSERT_SUBCAUSE_FW_INERNAL_ERROR)
        {
            Log_error0("***ERROR***");
            Log_error0(">> INTERNAL FW ERROR!");
        }
        else
        {
            Log_error0("***ERROR***");
            Log_error0(">> INTERNAL ERROR!");
        }
        break;

    case HAL_ASSERT_CAUSE_ICALL_ABORT:
        Log_error0("***ERROR***");
        Log_error0(">> ICALL ABORT!");
        //HAL_ASSERT_SPINLOCK;
        break;

    case HAL_ASSERT_CAUSE_ICALL_TIMEOUT:
        Log_error0("***ERROR***");
        Log_error0(">> ICALL TIMEOUT!");
        //HAL_ASSERT_SPINLOCK;
        break;

    case HAL_ASSERT_CAUSE_WRONG_API_CALL:
        Log_error0("***ERROR***");
        Log_error0(">> WRONG API CALL!");
        //HAL_ASSERT_SPINLOCK;
        break;

    default:
        Log_error0("***ERROR***");
        Log_error0(">> DEFAULT SPINLOCK!");
        //HAL_ASSERT_SPINLOCK;
    }

    return;
}
```

```
/**************************************************************************
 */
```



CC1352 Board Connections to LM35 Temperature Sensor



12:37

Starter

BLUETOOTH LOW ENERGY DEVICES

BLE Mesh Functions ...

BLE MESH (NEW)

No Localname
6E58868E-7063-3E6A-5009-AA81097AC5A7

(No localname)
2469595F-A39E-1419-585F-7F1D88488674

(CPE403_DA_5)
B580B2C9-F025-8B15-7C50-59BBB42E1E7D

Unknown
Man. dat : {length = 29, bytes = 0x06000109...

(Brian’s iPad)
8A921FBD-307D-8EF9-DC74-4662540D25A7

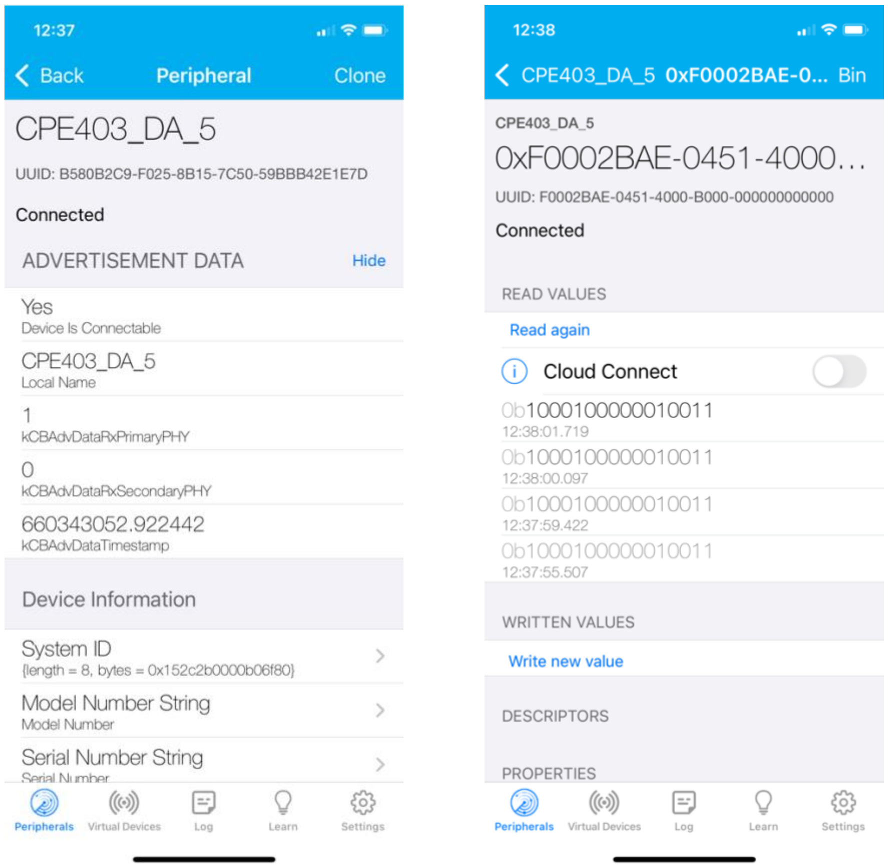
(No localname)
2D6D960C-C5B5-1FB5-84C9-933E5A415313

(No localname)
2B31630E-BDB5-D88C-9843-2E6B3A8B02A6

(Brian's Macbook Pro)
62667461-8BE4-E368-6153-1F972B9C7C9D

(No localname)
3B92F90F-656A-7DE6-2CAA-675B162B7809

Device Displayed in SimpleLink app



12:37

Back Peripheral Clone

CPE403_DA_5

UUID: B580B2C9-F025-8B15-7C50-59BBB42E1E7D

Connected

ADVERTISEMENT DATA Hide

Yes	Device Is Connectable
CPE403_DA_5	Local Name
1	kCBAdvDataRxPrimaryPHY
0	kCBAdvDataRxSecondaryPHY
660343052.922442	kCBAdvDataTimestamp

Device Information

System ID	{length = 8, bytes = 0x152c2b0000b06f80}
Model Number String	Model Number
Serial Number String	Serial Number

12:38

CPE403_DA_5 0xF0002BAE-0... Bin

CPE403_DA_5

0xF0002BAE-0451-4000…

UUID: F0002BAE-0451-4000-B000-000000000000

Connected

READ VALUES

Read again

Cloud Connect

0b1000100000010011	12:38:01.719
0b1000100000010011	12:38:00.097
0b1000100000010011	12:37:59.422
0b1000100000010011	12:37:55.507

WRITTEN VALUES

Write new value

DESCRIPTORS

PROPERTIES

Device Displayed and Reading Custom BLE Data in LightBlue App

GitHub: https://github.com/brianwolak/advanced_submissions/tree/main/DA_5

YouTube: https://youtu.be/5SGNBTDL-GQ

"This assignment submission is my own, original work".
Brian Wolak