CPE403 – Advanced Embedded Systems

Design Assignment 05

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Do Le

Email: led2@unlv.nevada.edu

Github Repository link (root): https://github.com/DoVietLe/AES

Youtube Playlist link (root):
https://www.youtube.com/playlist?list=PLFfzhLPj7fvOz1lm2Vd9DevkHetoyvRQ6

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.

A lot of modifications were made to make the collector and sensors compatible. The following is the most important part which allows the collector to display data. The sensor was modified to increment a counter when the button was pressed.

Sensor

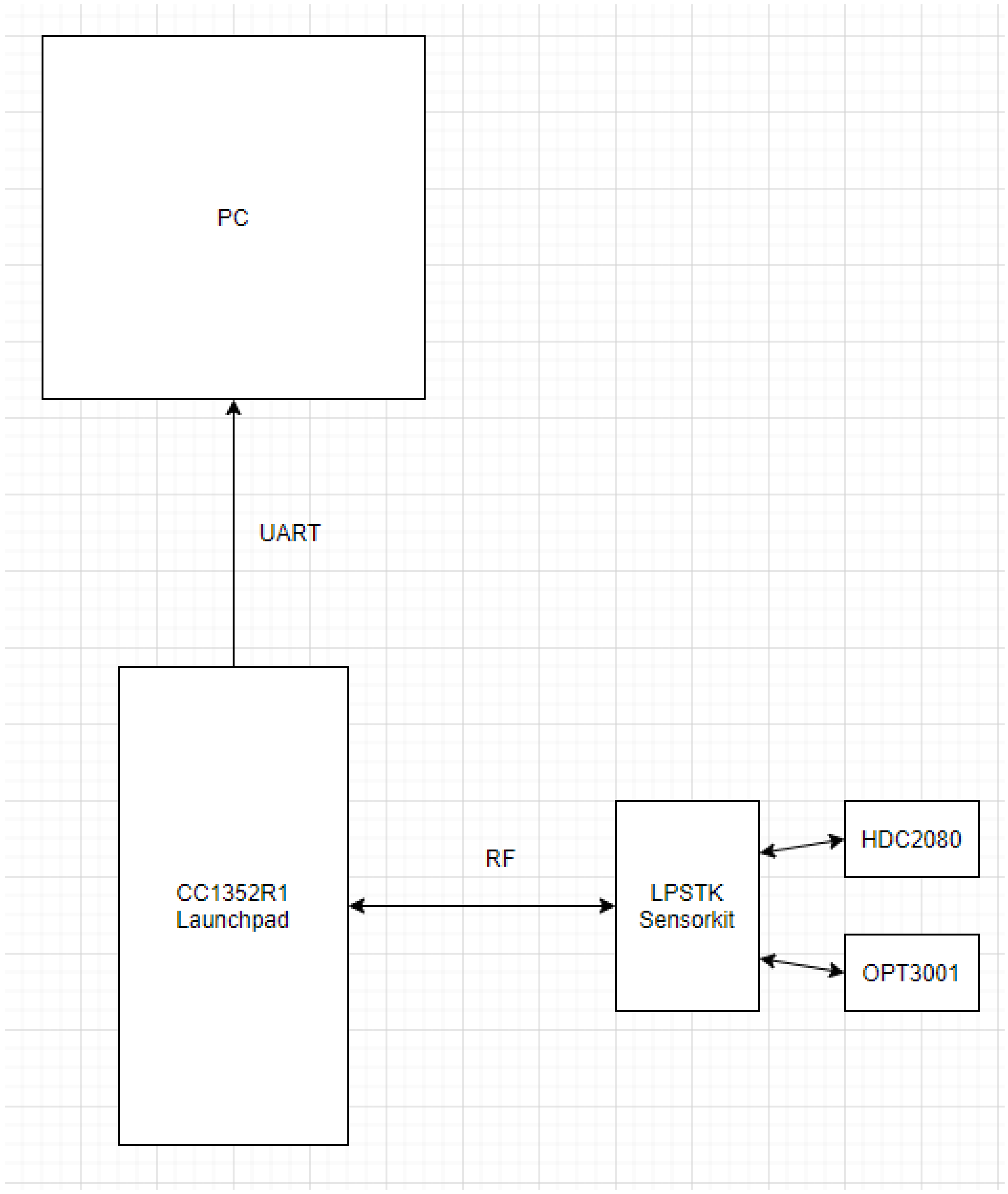
```
uint16_t generic_sensor_val = 0;

void Ssf_processEvents(void)
{
    /* Did a key press occur? */
    if(events & KEY_EVENT)
    {
        /* Right key press is a PAN disassociation request, if the device has started.
*/
        if((keys == gRightButtonHandle) && (started == true))
        {
            uint8_t provState = Jdllc_getProvState();
            if ((provState == Jdllc_states_joined) || (provState==
Jdllc_states_rejoined))
            {
                /* Send disassociation request only if you are in a network */
#ifndef CUI_DISABLE
                CUI_statusLinePrintf(ssfCuiHndl, sensorStatusLine, "Leaving");
#endif /* CUI_DISABLE */
```

```
                Jdllc_sendDisassociationRequest();
            }
        }
        /* Left key press is for starting the sensor network */
        else if(keys == gLeftButtonHandle)
        {

            if(started == false)
            {
#ifndef CUI_DISABLE
                CUI_statusLinePrintf(ssfCuiHndl, sensorStatusLine, "Starting");
#endif /* CUI_DISABLE */

                /* Tell the sensor to start */
                Util_setEvent(&Sensor_events, SENSOR_START_EVT);
                /* Wake up the application thread when it waits for clock event */
                Semaphore_post(sensorSem);
            }
            else
            {   /* Send LED toggle request to identify collector */
                Sensor_sendIdentifyLedRequest();
                generic_sensor_val ++;
            }
        }

        /* Clear the key press indication */
        keys = NULL;

        /* Clear the event */
        Util_clearEvent(&events, KEY_EVENT);
    }

#ifdef FEATURE_NATIVE_OAD
    /* Did a OAD event occur? */
#ifdef FEATURE_TOAD
    if(Sensor_events & SENSOR_OAD_TIMEOUT_EVT || Sensor_events &
SENSOR_TOAD_DECODE_EVT)
#else
    if(Sensor_events & SENSOR_OAD_TIMEOUT_EVT)
#endif
    {
        OADClient_processEvent(&Sensor_events);
    }
#endif //FEATURE_NATIVE_OAD

    if(events & SENSOR_UI_INPUT_EVT)
    {
#ifndef CUI_DISABLE
        CUI_processMenuUpdate();
#endif /* CUI_DISABLE */

        Util_clearEvent(&events, SENSOR_UI_INPUT_EVT);
    }

    if(events & SENSOR_SEND_COLLECTOR_IDENT_EVT)
    {
        Sensor_sendIdentifyLedRequest();

        Util_clearEvent(&events, SENSOR_SEND_COLLECTOR_IDENT_EVT);
    }

}
```

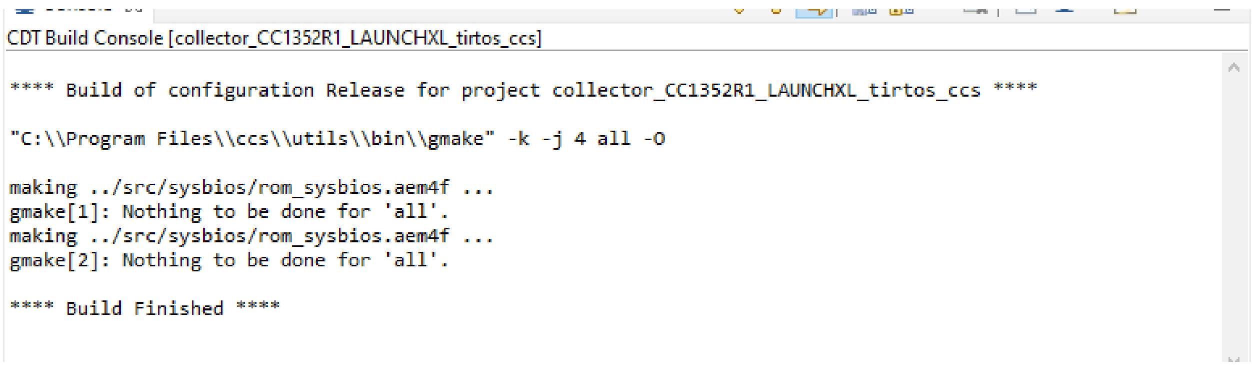
Collector

```
void Csf_init(void *sem)
{
#ifndef CUI_DISABLE
    CUI_clientParams_t clientParams;
#endif /* CUI_DISABLE */
#ifdef NV_RESTORE
    /* Save off the NV Function Pointers */
    pNV = &Main_user1Cfg.nvFps;
#endif

    /* Save off the semaphore */
    collectorSem = sem;

#ifndef CUI_DISABLE
    /* Open UI for key and LED */
    CUI_clientParamsInit(&clientParams);

    strncpy(clientParams.clientName, "154 Collector", MAX_CLIENT_NAME_LEN);
#ifdef LPSTK
  clientParams.maxStatusLines = 5;
#else
  clientParams.maxStatusLines = 5;
#endif

#ifdef LPSTK
  CUI_statusLineResourceRequest(csfCuiHndl, "LPSTK Data", true, &lpstkDataStatusLine);
#endif /* LPSTK */
  CUI_statusLineResourceRequest(csfCuiHndl, "Number of Joined Devices", false,
&numJoinDevStatusLine);
  CUI_statusLineResourceRequest(csfCuiHndl, "Generic Cnt", true, &genericStatusLine);
#if !defined(AUTO_START)
  CUI_statusLinePrintf(csfCuiHndl, collectorStatusLine, "Waiting...");
#endif /* AUTO_START */

#ifdef FEATURE_SECURE_COMMISSIONING
    clientParams.maxStatusLines++;
#endif /* FEATURE_SECURE_COMMISSIONING */
#ifdef SECURE_MANAGER_DEBUG
    clientParams.maxStatusLines++;
#endif /* SECURE_MANAGER_DEBUG */
#ifdef SECURE_MANAGER_DEBUG2
    clientParams.maxStatusLines++;
#endif /* SECURE_MANAGER_DEBUG2 */

    csfCuiHndl = CUI_clientOpen(&clientParams);
#endif /* CUI_DISABLE */

#ifdef FEATURE_SECURE_COMMISSIONING
#ifndef CUI_DISABLE
    /* Initialize the security manager and register callbacks */
    SM_init(collectorSem, csfCuiHndl);
#else
    SM_init(collectorSem);
#endif /* CUI_DISABLE */
#endif //FEATURE_SECURE_COMMISSIONING

    /* Initialize Keys */
    Button_Params bparams;
    Button_Params_init(&bparams);
    gLeftButtonHandle = Button_open(CONFIG_BTN_LEFT, processKeyChangeCallback,
&bparams);
    // Open Right button without appCallBack
    gRightButtonHandle = Button_open(CONFIG_BTN_RIGHT, NULL, &bparams);

    // Read button state
    if (!GPIO_read(((Button_HWAttrs*)gRightButtonHandle->hwAttrs)->gpioIndex))
    {
        /* Right key is pressed on power up, clear all NV */
        Csf_clearAllNVItems();
    }
    // Set button callback
    Button_setCallback(gRightButtonHandle, processKeyChangeCallback);

#if !defined(POWER_MEAS)
    /* Initialize the LEDs */
    LED_Params ledParams;
    LED_Params_init(&ledParams);
    gGreenLedHandle = LED_open(CONFIG_LED_GREEN, &ledParams);
    gRedLedHandle = LED_open(CONFIG_LED_RED, &ledParams);

    // Blink to indicate the application started up correctly
    LED_startBlinking(gRedLedHandle, 500, 3);
#endif /* POWER_MEAS */

#ifndef CUI_DISABLE
    CUI_registerMenu(csfCuiHndl, &csfMainMenu);

    CUI_statusLineResourceRequest(csfCuiHndl, "Status", false, &collectorStatusLine);
    CUI_statusLineResourceRequest(csfCuiHndl, "Device Status", true,
&deviceStatusLine);
#ifdef LPSTK
    CUI_statusLineResourceRequest(csfCuiHndl, "LPSTK Data", true,
&lpstkDataStatusLine);
#endif /* LPSTK */
    CUI_statusLineResourceRequest(csfCuiHndl, "Number of Joined Devices", false,
&numJoinDevStatusLine);
    CUI_statusLineResourceRequest(csfCuiHndl, "Generic Cnt", true,
&genericStatusLine);
#if !defined(AUTO_START)
    CUI_statusLinePrintf(csfCuiHndl, collectorStatusLine, "Waiting...");
#endif /* AUTO_START */
#endif /* CUI_DISABLE */

#if defined(MT_CSF)
    {
        uint8_t resetReseason = 0;

        if(pNV != NULL)
        {
            if(pNV->readItem != NULL)
            {
                /* Attempt to retrieve reason for the reset */
                (void)pNV->readItem(nvResetId, 0, 1, &resetReseason);
            }

            if(pNV->deleteItem != NULL)
            {
                /* Only use this reason once */
                (void)pNV->deleteItem(nvResetId);
            }
        }
        /* Start up the MT message handler */
        MTCSF_init(resetReseason, gLeftButtonHandle);

        /* Did we reset because of assert? */
        if(resetReseason > 0)
        {
#ifndef CUI_DISABLE
            CUI_statusLinePrintf(csfCuiHndl, collectorStatusLine, "Restarting...");
#endif /* CUI_DISABLE */

            /* Tell the collector to restart */
            Csf_events |= CSF_KEY_EVENT;
            Csf_keys = gLeftButtonHandle;
        }
    }
#endif
}

void Csf_deviceSensorDataUpdate(ApiMac_sAddr_t *pSrcAddr, int8_t rssi,
                                Smsgs_sensorMsg_t *pMsg)
{
#ifndef POWER_MEAS
    LED_toggle(gGreenLedHandle);
#endif /* endif for POWER_MEAS */
#ifndef CUI_DISABLE
    if(pMsg->frameControl & Smsgs_dataFields_bleSensor)
    {
        CUI_statusLinePrintf(csfCuiHndl, deviceStatusLine, "ADDR:%2x%2x%2x%2x%2x%2x,
UUID:0x%04x, "
                             "ManFac:0x%04x, Length:%d, Data:0x%02x", pMsg-
>bleSensor.bleAddr[5],
                             pMsg->bleSensor.bleAddr[4], pMsg->bleSensor.bleAddr[3],
pMsg->bleSensor.bleAddr[2],
                             pMsg->bleSensor.bleAddr[1], pMsg->bleSensor.bleAddr[0],
pMsg->bleSensor.uuid,
                             pMsg->bleSensor.manFacID, pMsg->bleSensor.dataLength,
pMsg->bleSensor.data[0]);
    }
    else
    {
        CUI_statusLinePrintf(csfCuiHndl, deviceStatusLine, "Sensor - Addr=0x%04x,
Temp=%d, Humidity=%d, Light=%d, RSSI=%d",
                      pSrcAddr->addr.shortAddr,
                      pMsg->humiditySensor.temp,
                      pMsg->humiditySensor.humidity,
                      pMsg->lightSensor.rawData,
                      rssi);
        CUI_statusLinePrintf(csfCuiHndl, genericStatusLine, "%d", pMsg-
>genericSensor);
#ifdef LPSTK
        CUI_statusLinePrintf(csfCuiHndl, lpstkDataStatusLine, "Humid=%d, Light=%d,
Accl=(%d, %d, %d, %d, %d)",
                             pMsg->humiditySensor.humidity, pMsg-
>lightSensor.rawData,
                             pMsg->accelerometerSensor.xAxis, pMsg-
>accelerometerSensor.yAxis,
                                    pMsg->accelerometerSensor.zAxis, pMsg-
>accelerometerSensor.xTiltDet,
                                    pMsg->accelerometerSensor.yTiltDet);
    CUI_statusLinePrintf(csfCuiHndl, numJoinDevStatusLine, "%x",
getNumActiveDevices());
#endif /* LPSTK*/
    }
#endif /* CUI_DISABLE */

#if defined(MT_CSF)
    MTCSF_sensorUpdateIndCB(pSrcAddr, rssi, pMsg);
#endif /* endif for MT_CSF */
}
```

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



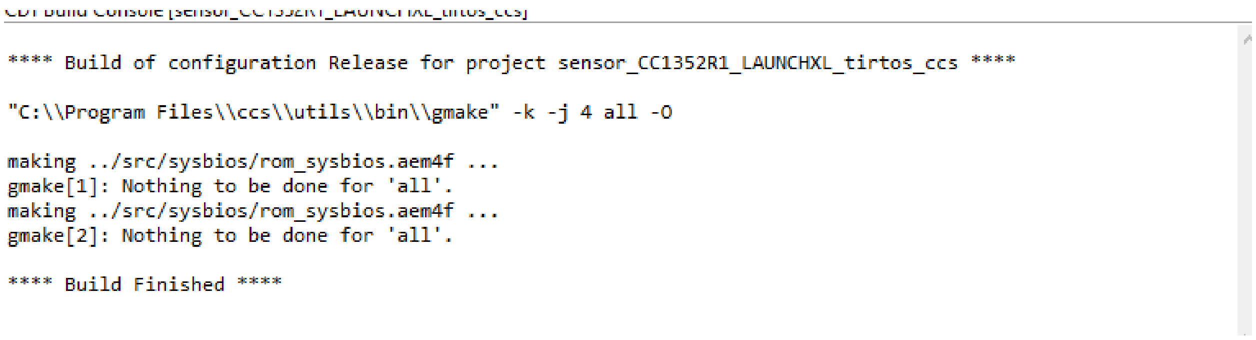
3. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.



CDT Build Console [collector_CC1352R1_LAUNCHXL_tirtos_ccs]

```
**** Build of configuration Release for project collector_CC1352R1_LAUNCHXL_tirtos_ccs ****

"C:\\Program Files\\ccs\\utils\\bin\\gmake" -k -j 4 all -O

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

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

Figure 1: Compilation of Collector Code



```
**** Build of configuration Release for project sensor_CC1352R1_LAUNCHXL_tirtos_ccs ****

"C:\\Program Files\\ccs\\utils\\bin\\gmake" -k -j 4 all -O

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

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

Figure 4: Compilation of Sensor Code

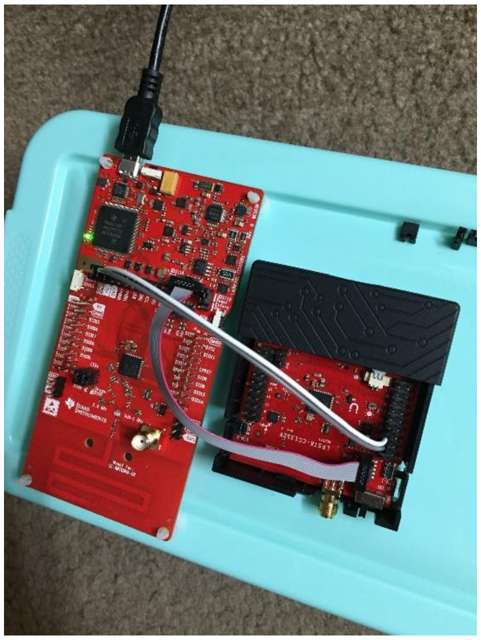


Figure 2: Programming Setup

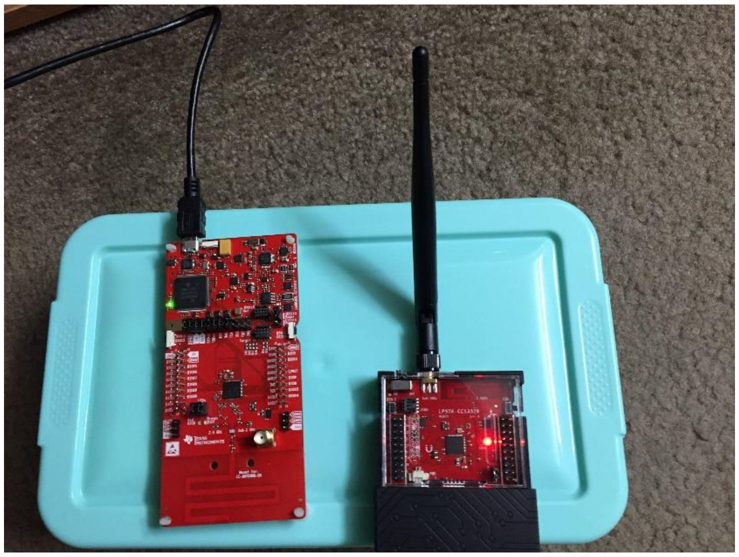
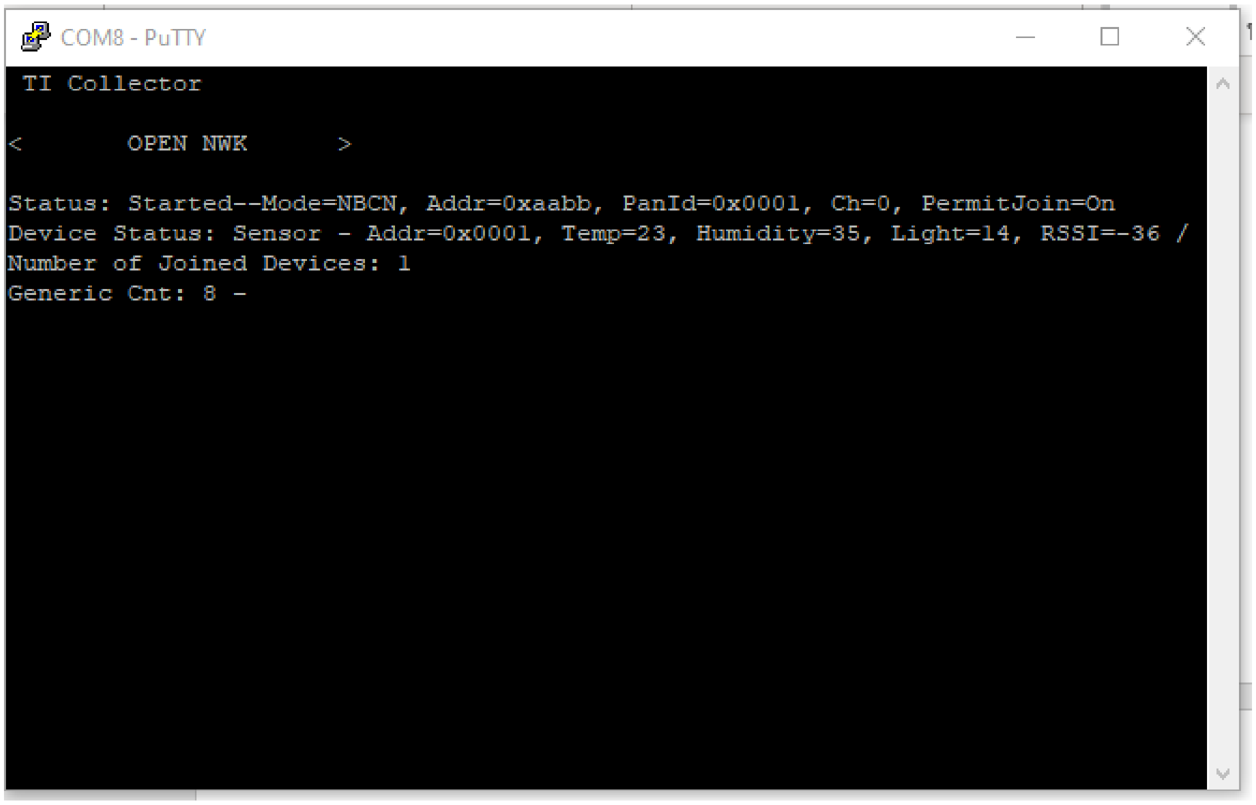


Figure 3: Connection Setup



```
COM8 - PuTTY
 TI Collector

<       OPEN NWK      >

Status: Started--Mode=NBCN, Addr=0xaabb, PanId=0x0001, Ch=0, PermitJoin=On
Device Status: Sensor - Addr=0x0001, Temp=23, Humidity=35, Light=14, RSSI=-36 /
Number of Joined Devices: 1
Generic Cnt: 8 -
```

Figure 5: Terminal Window

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

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

Do V. Le