CPE403 – Advanced Embedded Systems

Design Assignment 5

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.

Collector and Sensor: smsgs.h

```
/*! Length of the button press portion of the sensor data message */
#define SMSGS_BUTTON_PRESS_LEN 2
/*! Button Press Request message length (over-the-air length) */
#define SMSGS_BUTTON_PRESS_REQEUST_MSG_LEN 1
/*! Button Press Request message length (over-the-air length) */
#define SMSGS_BUTTON_PRESS_RESPONSE_MSG_LEN 2

/*!
 Message IDs for Sensor data messages.  When sent over-the-air in a message,
 this field is one byte.
 */
 typedef enum
 {
…
    /* Button Press request msg */
    Smsgs_cmdIds_ButtonPressReq = 18,
    /* Button Press response msg */
    Smsgs_cmdIds_ButtonPressRsp = 19

 } Smsgs_cmdIds_t;
/*!

 Frame Control field states what data fields are included in reported
 sensor data, each value is a bit mask value so that they can be combined
 (OR'd together) in a control field.
 When sent over-the-air in a message this field is 2 bytes.
 */
typedef enum
{
…
    /*! Button Press */
    Smsgs_dataFields_buttonPress = 0x0800,
} Smsgs_dataFields_t;


/*!
Button Press Sensor Field
*/
typedef struct _Smsgs_buttonpressfield_t {
…
 /*! Button Press count out of button press sensor */
 uint16_t count;
} Smsgs_buttonPressField_t;


/*!
 Sensor Data message: sent from the sensor to the collector
```

```
 */
typedef struct _Smsgs_sensormsg_t
{
…
    /*!
     Button Press field - valid only if Smsgs_dataFields_buttonPress
     is set in frameControl.
     */
    Smsgs_buttonPressField_t buttonPress;
} Smsgs_sensorMsg_t;
```

Collector: collector.h

```
/* Default configuration frame control */
#define CONFIG_FRAME_CONTROL (Smsgs_dataFields_tempSensor | \
                              Smsgs_dataFields_lightSensor | \
                              Smsgs_dataFields_humiditySensor | \
                              Smsgs_dataFields_msgStats | \
                              Smsgs_dataFields_configSettings | \
                              Smsgs_dataFields_buttonPress)
```

```
/*!
* @brief Build and send the button press message to a device.
*
* @param pDstAddr - destination address of the device to send the message
*
* @return Collector_status_success, Collector_status_invalid_state
*         or Collector_status_deviceNotFound
*/
extern Collector_status_t Collector_sendButtonPressRequest(ApiMac_sAddr_t *pDstAddr);
```

Collector: collector.c

```
/*!
 * @brief      Process the Sensor Data message.
 *
 * @param      pDataInd - pointer to the data indication information
 */
static void processSensorData(ApiMac_mcpsDataInd_t *pDataInd)
{
…
    if(sensorData.frameControl & Smsgs_dataFields_buttonPress)
    {
        sensorData.buttonPress.count = Util_buildUint16(pBuf[0], pBuf[1]);
        pBuf += 2;
    }
…
}
```

```
/*! Build and send the Button Press message to a device Public function defined in
collector.h */
Collector_status_t Collector_sendButtonPressRequest(ApiMac_sAddr_t *pDstAddr)
{
  Collector_status_t status = Collector_status_invalid_state; /* Are we in the right
state? */

  if(cllcState >= Cllc_states_started)
  {
      Llc_deviceListItem_t item;
       /* Is the device a known device? */
      if(Csf_getDevice(pDstAddr, &item))
      {
          uint8_t buffer[SMSGS_BUTTON_PRESS_REQEUST_MSG_LEN];
          /* Build the message */
          buffer[0] = (uint8_t)Smsgs_cmdIds_ButtonPressReq;
          sendMsg(Smsgs_cmdIds_ButtonPressReq, item.devInfo.shortAddress,
                  item.capInfo.rxOnWhenIdle,
                  SMSGS_BUTTON_PRESS_RESPONSE_MSG_LEN,
                  buffer);
          status = Collector_status_success;
      }
      else
      {
          status = Collector_status_deviceNotFound;
      }
  }
  return(status);
}
```

Collector: csf.c

```
/*!
 The application calls this function to indicate that a device
 has reported sensor data.

 Public function defined in csf.h
 */
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 */
    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, Button 1 Press Count=%d, RSSI=%d",
                        pSrcAddr->addr.shortAddr,
                        pMsg->humiditySensor.temp,
                        pMsg->humiditySensor.humidity,
                        pMsg->lightSensor.rawData,
                        pMsg->buttonPress.count,
                        rssi);
#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);
#endif
    }
    CUI_statusLinePrintf(csfCuiHndl, numJoinDevStatusLine, "%x",
getNumActiveDevices());


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

Sensor: sensor.c

```
/* Keep track of number of times button has been pressed */
extern uint16_t button_press_val;
static Smsgs_buttonPressField_t buttonPress =
{
0
};


/*!
 Initialize this application.

 Public function defined in sensor.h
 */
#ifdef OSAL_PORT2TIRTOS
void Sensor_init(uint8_t macTaskId)
#else
void Sensor_init(void)
#endif
```

```
{
…
#endif /* LPSTK */
    configSettings.frameControl |= Smsgs_dataFields_msgStats;
    configSettings.frameControl |= Smsgs_dataFields_configSettings;
#ifdef DMM_CENTRAL
    configSettings.frameControl |= Smsgs_dataFields_bleSensor;
#endif
    configSettings.frameControl |= Smsgs_dataFields_buttonPress;
…
}

/*!
 * @brief      MAC Data Indication callback.
 *
 * @param      pDataInd - pointer to the data indication information
 */

static void dataIndCB(ApiMac_mcpsDataInd_t *pDataInd)
{
…
            case Smsgs_cmdIds_ButtonPressReq:
              if(pDataInd->msdu.len == SMSGS_BUTTON_PRESS_REQEUST_MSG_LEN)
              {
                  /* send the response message directly */
                  cmdBytes[0] = (uint8_t) Smsgs_cmdIds_ButtonPressRsp;
                  cmdBytes[1] = *((uint8_t*)button_press_val);
                  Sensor_sendMsg(Smsgs_cmdIds_ButtonPressRsp,
                          &pDataInd->srcAddr, true,
                          SMSGS_BUTTON_PRESS_RESPONSE_MSG_LEN,
                          cmdBytes);

              }
…
}

#if !defined(OAD_IMG_A)
/*!
 @brief   Build and send sensor data message
 */
static void processSensorMsgEvt(void)
{
…
    if(sensor.frameControl & Smsgs_dataFields_buttonPress)
    {
      memcpy(&sensor.buttonPress, &buttonPress,
             sizeof(Smsgs_buttonPressField_t));
    }
}

/*!
 * @brief   Manually read the sensors
 */
static void readSensors(void)
```

```
{
#if defined(TEMP_SENSOR)
    /* Read the temp sensor values */
    tempSensor.ambienceTemp = Ssf_readTempSensor();
    tempSensor.objectTemp =  tempSensor.ambienceTemp;
#endif
#ifdef LPSTK
    Lpstk_Accelerometer accel;
    humiditySensor.temp = (uint16_t)Lpstk_getTemperature();
    humiditySensor.humidity = (uint16_t)Lpstk_getHumidity();
    hallEffectSensor.flux = Lpstk_getMagFlux();
    lightSensor.rawData = (uint16_t)Lpstk_getLux();
    Lpstk_getAccelerometer(&accel);
    accelerometerSensor.xAxis = accel.x;
    accelerometerSensor.yAxis = accel.y;
    accelerometerSensor.zAxis = accel.z;
    accelerometerSensor.xTiltDet = accel.xTiltDet;
    accelerometerSensor.yTiltDet = accel.yTiltDet;
#endif /* LPSTK */
    buttonPress.count = button_press_val;
}

/*!
 * @brief   Build and send sensor data message
 *
 * @param   pDstAddr - Where to send the message
 * @param   pMsg - pointer to the sensor data
 *
 * @return  true if message was sent, false if not
 */
static bool sendSensorMessage(ApiMac_sAddr_t *pDstAddr, Smsgs_sensorMsg_t *pMsg)
{
…
    if(pMsg->frameControl & Smsgs_dataFields_buttonPress)
    {
      len += SMSGS_BUTTON_PRESS_LEN;
    }
…
        if(pMsg->frameControl & Smsgs_dataFields_buttonPress)
        {
          pBuf = Util_bufferUint16(pBuf, pMsg->buttonPress.count);
        }
…
}

/*!
 * @brief   Filter the frameControl with readings supported by this device.
 *
 * @param   frameControl - suggested frameControl
 *
 * @return  new frame control settings supported
 */
static uint16_t validateFrameControl(uint16_t frameControl)
{
…
```

```
    if(frameControl & Smsgs_dataFields_buttonPress)
    {
     newFrameControl |= Smsgs_dataFields_buttonPress;
    }
…
}
```

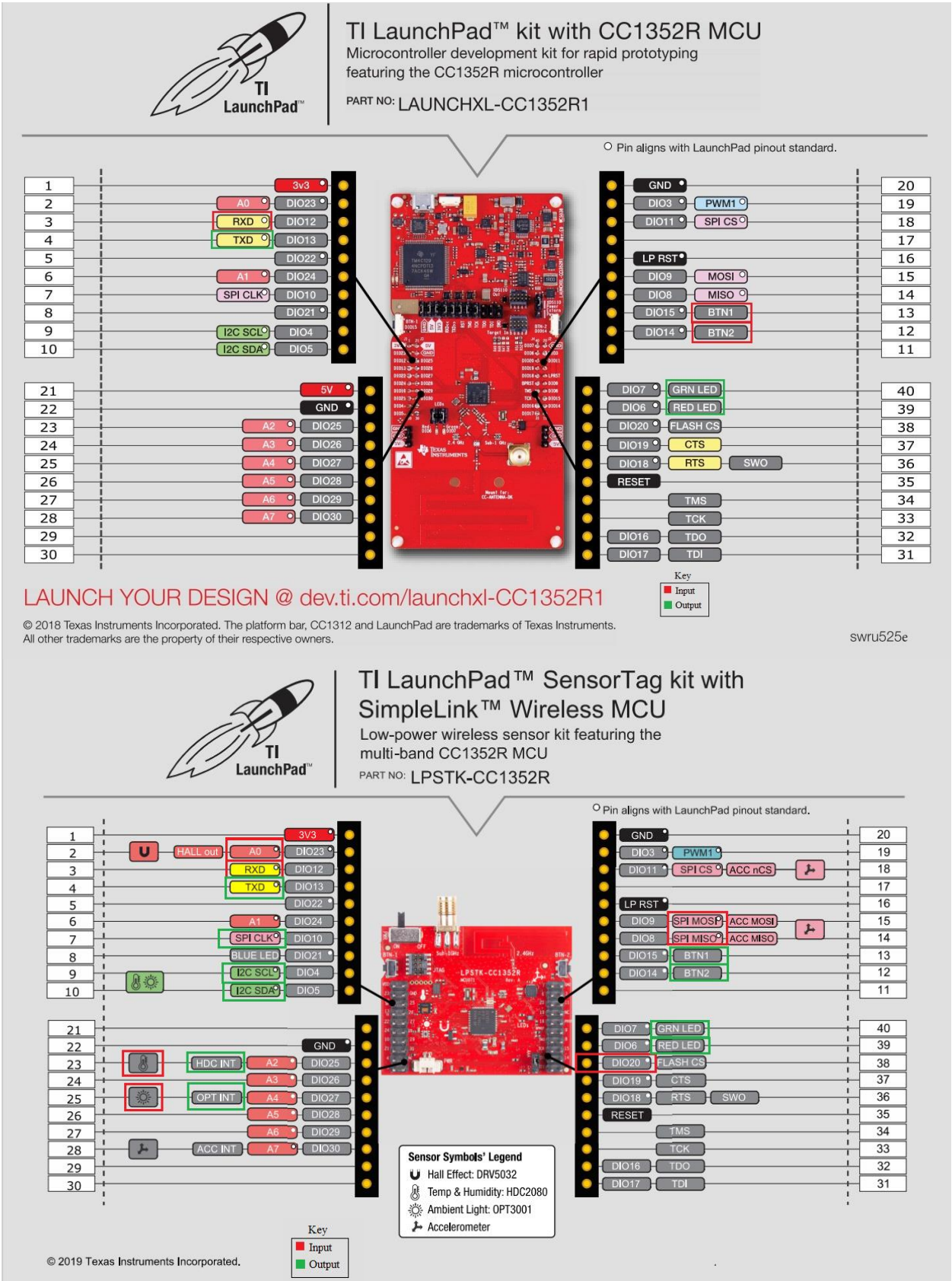
Sensor: ssf.c

```
/*********************************************************************
 Public variables
 *********************************************************************/
/*! Number of times button 1 has been pressed */
uint16_t button_press_val = 0;

/*!
 The application must call this function periodically to
 process any events that this module needs to process.

 Public function defined in ssf.h
 */
void Ssf_processEvents(void)
{
…
        /* Left key press is for starting the sensor network */
        else if(keys == gLeftButtonHandle)
        {

            if(started == false)
            {
                CUI_statusLinePrintf(ssfCuiHndl, sensorStatusLine, "Starting");

                /* 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();
                button_press_val ++;
            }
        }

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

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

Sensor: Sensor.opts

```
-DLPSTK
-DTEMP_SENSOR
…
```

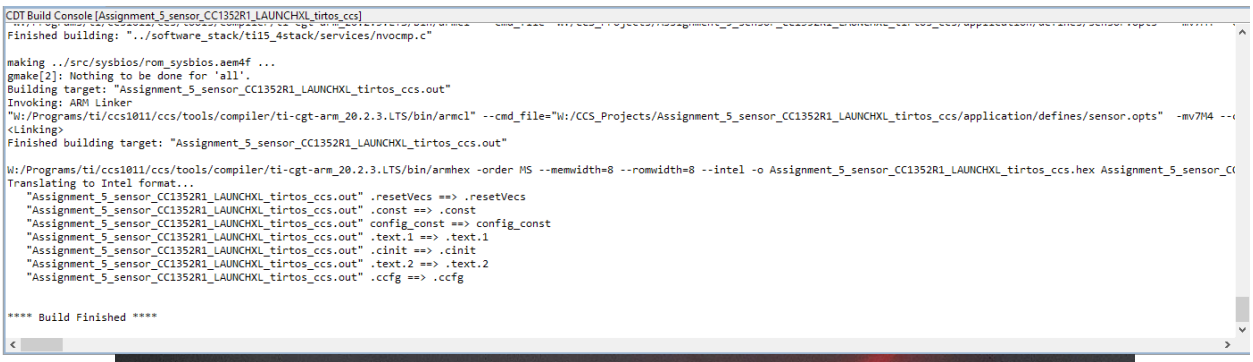
2. Block diagram and/or Schematics showing the components, pins used, and interface.

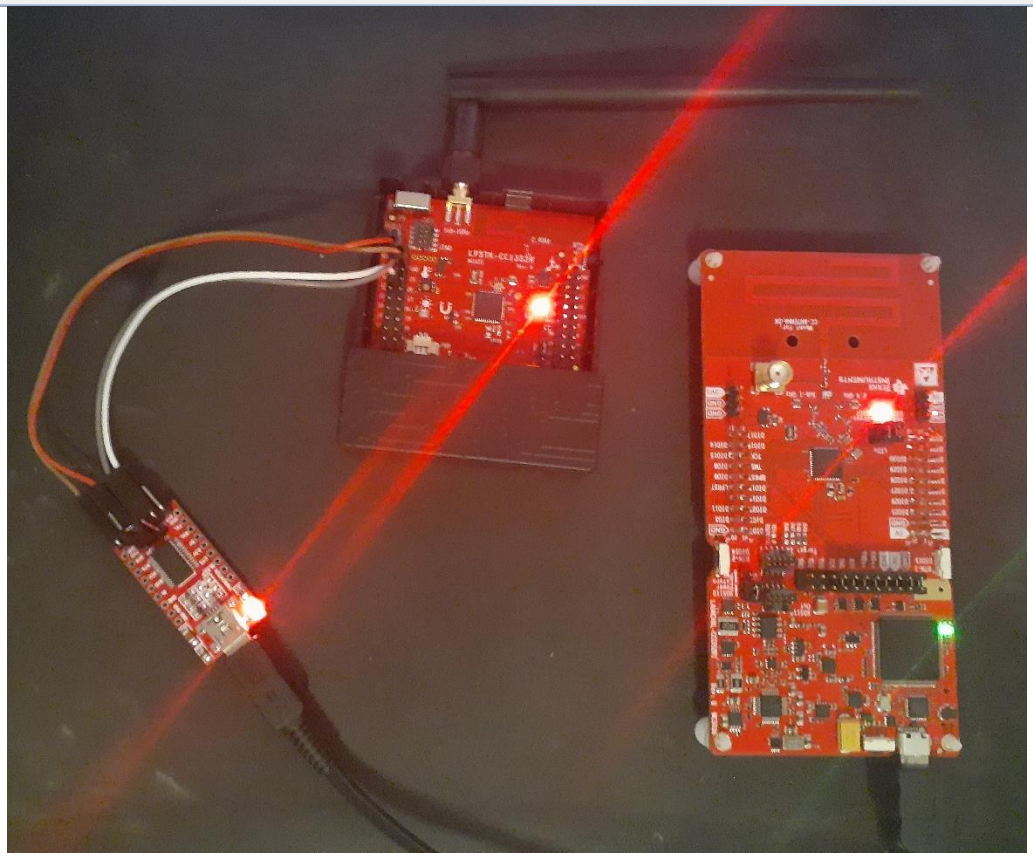


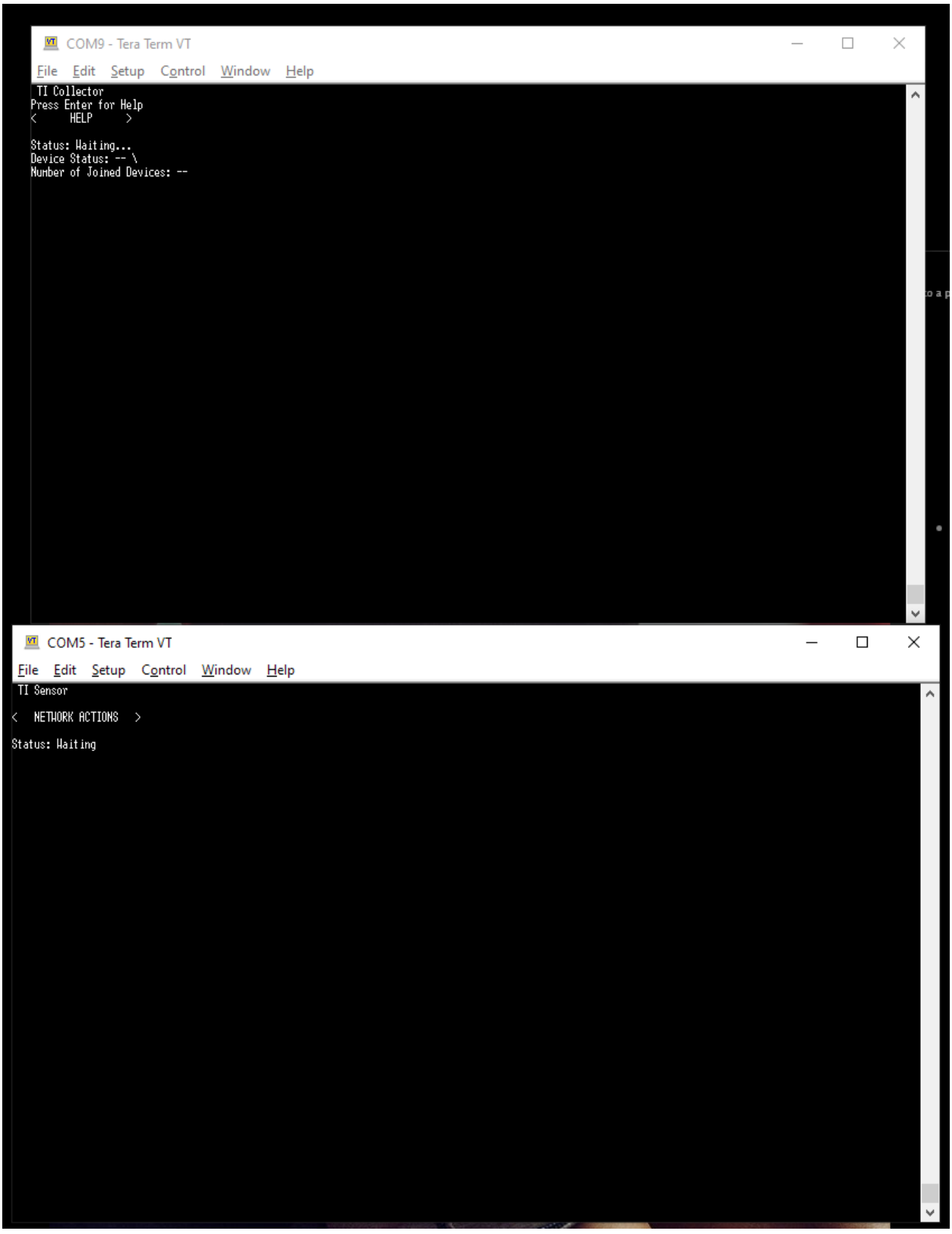
3. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.

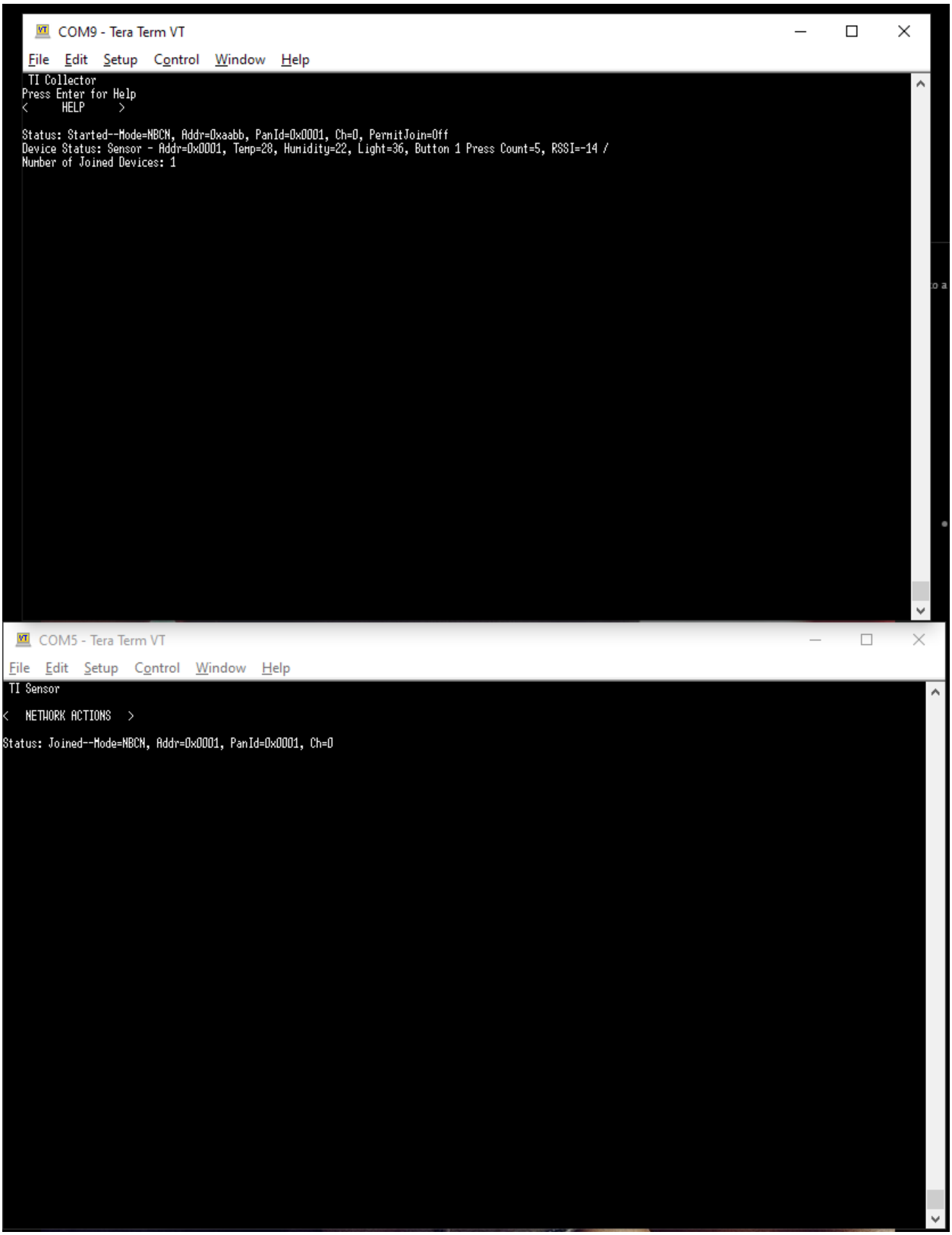
Collector

Sensor






COM9 - Tera Term VT
File Edit Setup Control Window Help
TI Collector
Press Enter for Help
< HELP >
Status: Waiting...
Device Status: -- \
Number of Joined Devices: --
COM5 - Tera Term VT
File Edit Setup Control Window Help
TI Sensor
< NETWORK ACTIONS >
Status: Waiting


COM9 - Tera Term VT
File Edit Setup Control Window Help
TI Collector
Press Enter for Help
< HELP >
Status: Started--Mode=NBCN, Addr=0xaabb, PanId=0x0001, Ch=0, PermitJoin=Off
Device Status: Sensor - Addr=0x0001, Temp=28, Humidity=22, Light=36, Button 1 Press Count=5, RSSI=-14 /
Number of Joined Devices: 1
COM5 - Tera Term VT
File Edit Setup Control Window Help
TI Sensor
< NETWORK ACTIONS >
Status: Joined--Mode=NBCN, Addr=0x0001, PanId=0x0001, Ch=0

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