Wireless Sensor Networks

PRACTICAL TRAINING 2

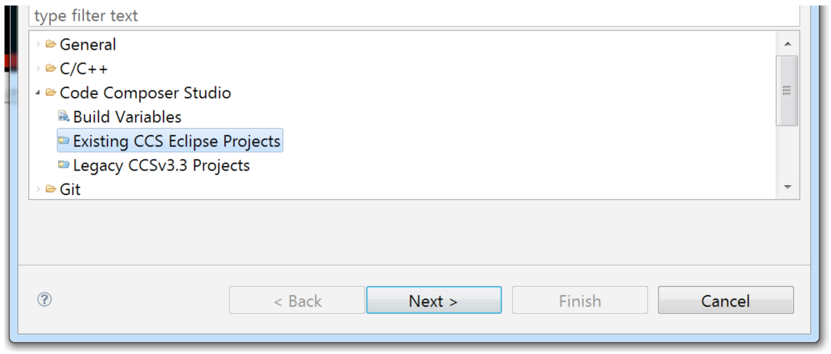
Introduction

After learning the basic functions of both the msp430 microcontroller and the Code Composer Studio IDE in the first part of the practical training, we will now focus on the simplest form of wireless communication with 2 target boards.

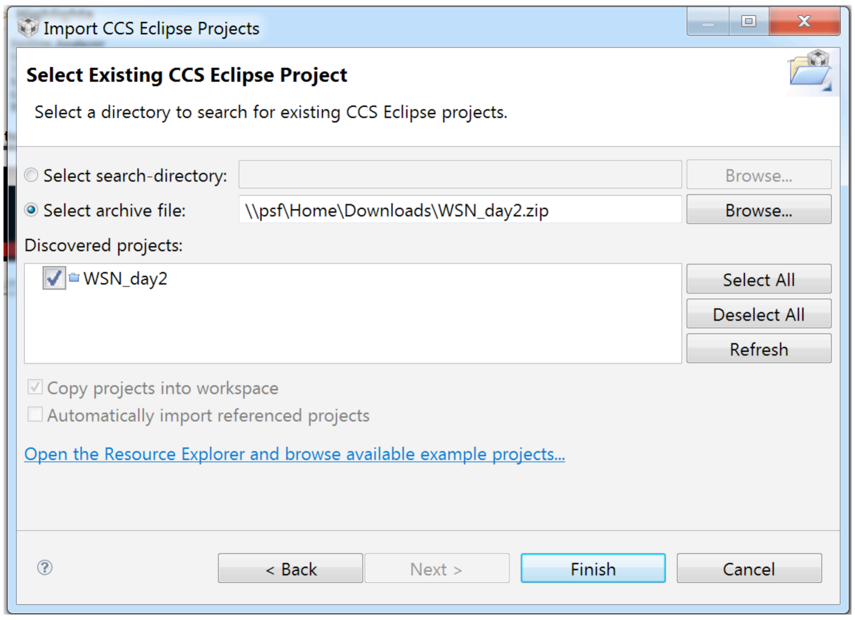
In part two of the practical training we will use the functions of the Data Link/Physical layer with its BSP and MRFI API. You will learn how to build a simple RF peer to peer network and get taught the function of basic RF standards like RSSI, CRC, CCA and LQI. At the end of this part you will have built a simple chat with all the groups.

MRFI Data Link/Physical Layer

1. In order to use the SimpliciTI Protocol and its lowest layer, we need some libraries provided by TI
2. Import CCS project with SimpliciTI libraries
    - Click on *File -> Import*
    - Choose *Code Composer Studio -> Existing CCS Eclipse Projects*



    - Click on *Browse* next to *Select archive file:* and browse to the WSN_day2.zip file


Import CCS Eclipse Projects

Select Existing CCS Eclipse Project

Select a directory to search for existing CCS Eclipse projects.

Select search-directory:

Select archive file: \\psf\Home\Downloads\WSN_day2.zip

Discovered projects:

☑ WSN_day2

Select All / Deselect All / Refresh

☑ Copy projects into workspace

☐ Automatically import referenced projects

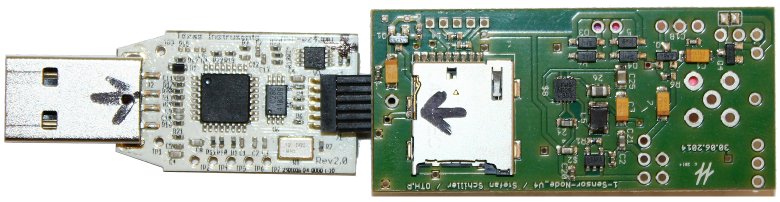
Open the Resource Explorer and browse available example projects...

< Back / Next > / Finish / Cancel

- Click on *Finish* to import the project to your workspace
- Now you have a new project with all necessary libraries and drivers included

3. As we will use 2 target boards, we also need 2 projects to write code for the microcontrollers. Therefore we will rename the first imported project and import it a second time.
   - Right click on the imported project and click on rename
   - Rename it to *WSN_part2_receiver*
   - Now repeat the step of importing the *WSN_day2.zip* project
   - Rename it this time to *WSN_part2_transmitter*

4. Connect the FET debugger + the target board to the pc and again wait for the driver installation to finish



The orientation of the debugger is very important, as the 6 pin debugging connecter will fit 2 ways. <u>Look carefully at picture and the arrows when connecting</u>!

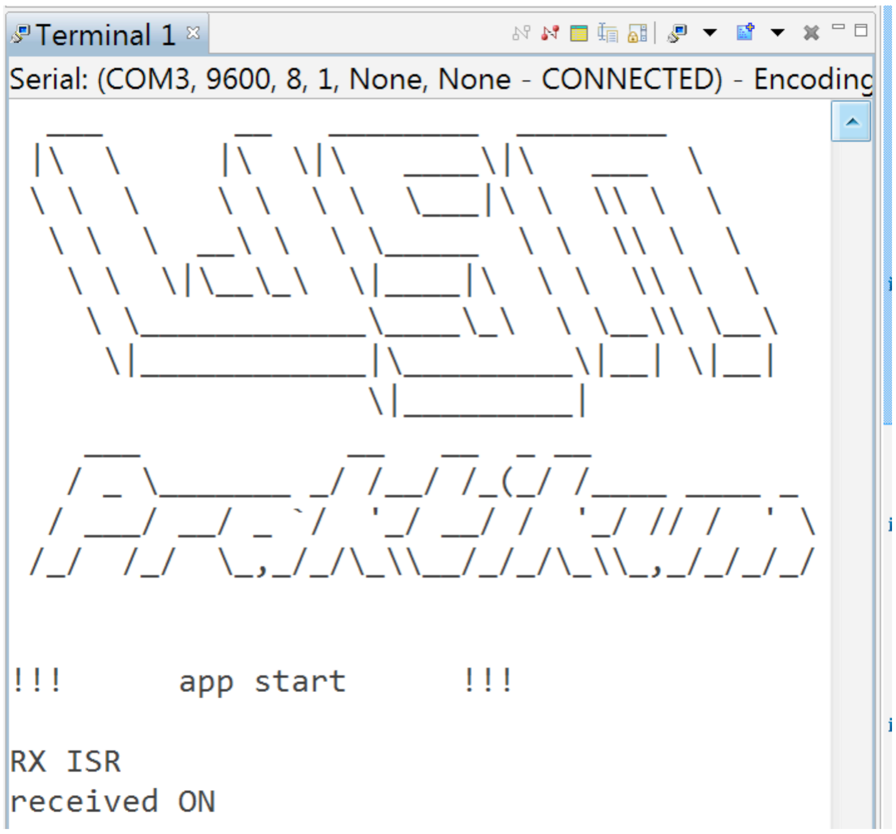
5. Figure out what the code is doing and try to understand the functions of the different code elements.

```
#include "mrfi.h"
#include "serial/serial.h"
#define msg_ON  0x01

void main (void)
{       WDTCTL = WDTPW+WDTHOLD;                      // Stop watchdog timer
        P1OUT  = BIT0;                          // P1.0 input+pullup
    P1REN |= BIT0;                              // P1.0 input+pullup
    P1IE  |= BIT0;                              // P1.0 interrupt enable
    P1IES |= BIT0;                              // P1.0 Hi/Lo edge
    P1IFG &= ~BIT0;                             // P1.0 IFG Flag cleared

        BSP_Init();
        MRFI_Init();                            // Init SPI com with CC2500
        MRFI_WakeUp();                          // wake up the radio
        MRFI_RxOn();                            // turn into Rx mode

        initSerial();                           // init Serial interface
        printf("\r\n\r\n!!!      app start      !!!\r\n\r\n");

        __bis_SR_register(LPM0_bits + GIE);     // Enter LPM0, interrupts enabled
}

void MRFI_RxCompleteISR_new()    // in Components/mrfi/radios/family5/mrfi_radio.c
{       printf("RX ISR\r\n");
        mrfiPacket_t packetreceived;
        MRFI_Receive(&packetreceived);

        if (packetreceived.frame[5]==msg_ON)
               printf("received msg_ON\r\n");
}

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{       P1IFG &= ~BIT0;                              // P1.0 IFG Flag cleared
        mrfiPacket_t packet;
        packet.frame[0]=0x14;
        packet.frame[5]=msg_ON;
        while(MRFI_TX_RESULT_SUCCESS!=MRFI_Transmit(&packet, MRFI_TX_TYPE_FORCED));
        printf("send msg_ON");
}
```

- You can use the UART interface with the standard *printf()* function for debugging and reading back states of your program code.

6. Load the code onto 2 target boards.
    - This time we use one target board with a battery holder.

      !***Always disconnect the battery while programming!***
    - Program the second target board and open the Terminal in CCS
    - Use the first target board to send a message to the second. You should see the received message in the Terminal.



```
Serial: (COM3, 9600, 8, 1, None, None - CONNECTED) - Encoding
 ___       __   ________  ________
|\  \     |\  \|\   ____\|\   ___  \
\ \  \    \ \  \ \  \___|\ \  \\ \  \
 \ \  \  __\ \  \ \_____  \ \  \\ \  \
  \ \  \|\__\_\  \|____|\  \ \  \\ \  \
   \ \____________\____\_\  \ \__\\ \__\
    \|____________|\_________\|__| \|__|
                  \|_________|

   ___             __   __  _ __
  / _ \_______ _/ /__/ /_(_) /____ ____ _
 / ___/ __/ _ `/  '_/ __/ /  '_/ // /  ' \
/_/  /_/  \_,_/_/\_\\__/_/_/\_\\_,_/_/_/_/


!!!      app start      !!!

RX ISR
received ON
```

- As every group will use the same RF Modules, you will not only receive messages from your sender, but also from the other groups.

7. Change the code so that you will only get a *received ON* message when your sender transmitted a packet. (you only need to change one #define)

Cyclic Redundancy Check (CRC)

1. Refer *to Cyclic Redundancy Check (CRC) in WSN_Theory.pdf*
2. Now you will use the following code to prove the importance of CRC

```
#include "mrfi.h"
#include "serial/serial.h"
#define msg_ON  0x01
       // RF Power, CRC off
void main (void)
{       WDTCTL  = WDTPW+WDTHOLD;             // Stop watchdog timer
        P1OUT   = BIT0;                      // P1.0 input+pullup
    P1REN |= BIT0;                           // P1.0 input+pullup
    P1IE   |= BIT0;                          // P1.0 interrupt enable
    P1IES |= BIT0;                           // P1.0 Hi/Lo edge
    P1IFG &= ~BIT0;                          // P1.0 IFG Flag cleared


        BSP_Init();
        MRFI_Init();                         // Init SPI com with CC2500

    MRFI_SetRFPwr(0);                 // RF transmitting power (0 to 2)
    mrfiRadioInterfaceWriteReg(PKTCTRL0,0x05);      //  CRC on (0x05)

        MRFI_WakeUp();                       // wake up the radio
        MRFI_RxOn();                         // turn into Rx mode

        initSerial();                        // init Serial interface
        printf("\r\n\r\n!!!      app start      !!!\r\n\r\n");

        __bis_SR_register(LPM0_bits + GIE);    // Enter LPM0, interrupts enabled
}

void MRFI_RxCompleteISR_new()    // in Components/mrfi/radios/family5/mrfi_radio.c
{       printf("RX ISR\r\n");
        mrfiPacket_t packetreceived;
        MRFI_Receive(&packetreceived);

        if (packetreceived.frame[5]==msg_ON){
                printf("received msg_ON \r\n");
                int i=0;
                for(i=0; i<12; i++)
                        printf("%c",packetreceived.frame[9+i]);
                printf("\r\n");
        }
}

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{       P1IFG &= ~BIT0;                             // P1.0 IFG Flag cleared
        mrfiPacket_t packet;
        packet.frame[5] = msg_ON;
        int i=0;
        for(i=0; i<12; i++)
                packet.frame[9 + i] = 'a' + i;
        packet.frame[0]= 0x14;

        while(MRFI_TX_RESULT_SUCCESS!=MRFI_Transmit(&packet, MRFI_TX_TYPE_FORCED));
        printf("send msg_ON\r\n");
}
```

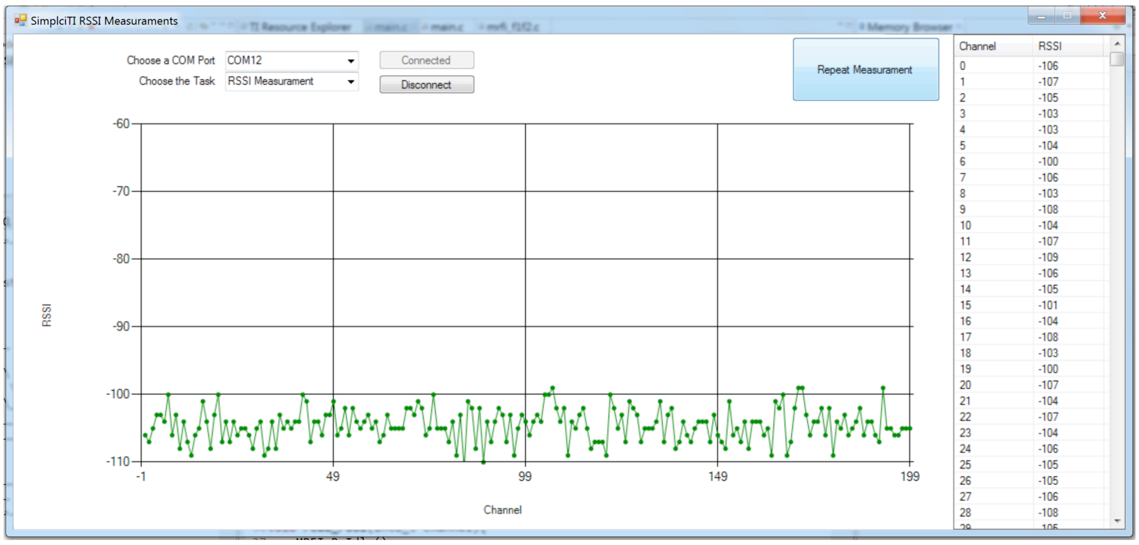
3. Flash the code on both target boards and open 2 terminals to log the communication of the target boards.
4. To deactivate the CRC of the target boards you have to change the Register PKTCTRL0. (*slau259e.pdf* page 715 and *WSN_Theory.pdf*) with the mrfiRadioInterfaceWriteReg(uint8_t addr, uint8_t, value).
5. Flash the target boards again and try to interfere the transmitter by putting your fingers on the antenna of the board while pressing the button.

Received Signal Strength Indicator (RSSI)

1. Refer to Received Signal Strength Indicator (RSSI) in *WSN_Theory.pdf*
2. Load the next code onto the receiver target board. The code measures the noise levels in dBm.

```
#include "mrfi.h"
#include "serial/serial.h"
#define msg_ON  0x01

void main (void){
       WDTCTL = WDTPW+WDTHOLD;              // Stop watchdog timer
              P1OUT  = BIT0;                       // P1.0 input+pullup
           P1REN |= BIT0;                          // P1.0 input+pullup
           P1IE  |= BIT0;                          // P1.0 interrupt enable
           P1IES |= BIT0;                          // P1.0 Hi/Lo edge
           P1IFG &= ~BIT0;                         // P1.0 IFG Flag cleared

              BSP_Init();
              MRFI_Init();                         // Init SPI com with CC2500
              MRFI_WakeUp();                       // wake up the radio
              MRFI_RxOn();                         // turn into Rx mode

              initSerial();                        // init Serial interface
              printf("\r\n\r\n!!!      app start      !!!\r\n\r\n");

              __bis_SR_register(LPM0_bits + GIE);    // Enter LPM0, interrupts
enabled
}

void MRFI_RxCompleteISR_new()    // in
Components/mrfi/radios/family5/mrfi_radio.c
{}

void read_rssi(int8_t channel){
       MRFI_RxIdle();
       mrfiRadioInterfaceWriteReg(CHANNR,channel);  // set channel number
       MRFI_RxOn();
       printf("%i\r\n", MRFI_Rssi());
}
```

3. Add a ISR to the program which will cycle 200 channels and measure the RSSI value. The ISR should be triggered from any character received by the UART interface.
   (look at your UART ISR code from Practical training 1)
4. Start the program *SimpliciTI RSSI Measurements.exe* and connect to the port of the receiver target board. Start a measurement to see the noisefloor.



Channel	RSSI
0	-106
1	-107
2	-105
3	-103
4	-103
5	-104
6	-100
7	-106
8	-103
9	-108
10	-104
11	-107
12	-109
13	-106
14	-105
15	-101
16	-104
17	-108
18	-103
19	-100
20	-107
21	-104
22	-107
23	-104
24	-106
25	-105
26	-105
27	-106
28	-108

5. With the same code repeat it several times and compare the data. Why are they different?
6. Now you have to prepare your second target board to be a transmitter. Set the channel to your group number multiplied by 10 and transmit a data packet continuously in a while loop.
7. Start *SimpliciTI RSSI Measurement.exe* again and repeat the measurement.
8. You can now change the transmission power and repeat the measurement

Link quality indicator (LQI)

1. Refer to Link quality indicator (LQI) *into WSN_Theory.pdf*
2. Based on the last tasks you should now be able to write your own program for a Link Quality measurement.

   Use the following instructions and code parts to build a program:

   Transmitter:

   - Send 100 packets when the button on the transmitter is pressed
   - Send a STOP packet 20 times (define a packet that will be identified by the receiver as the last packet e.g. 0xFF)

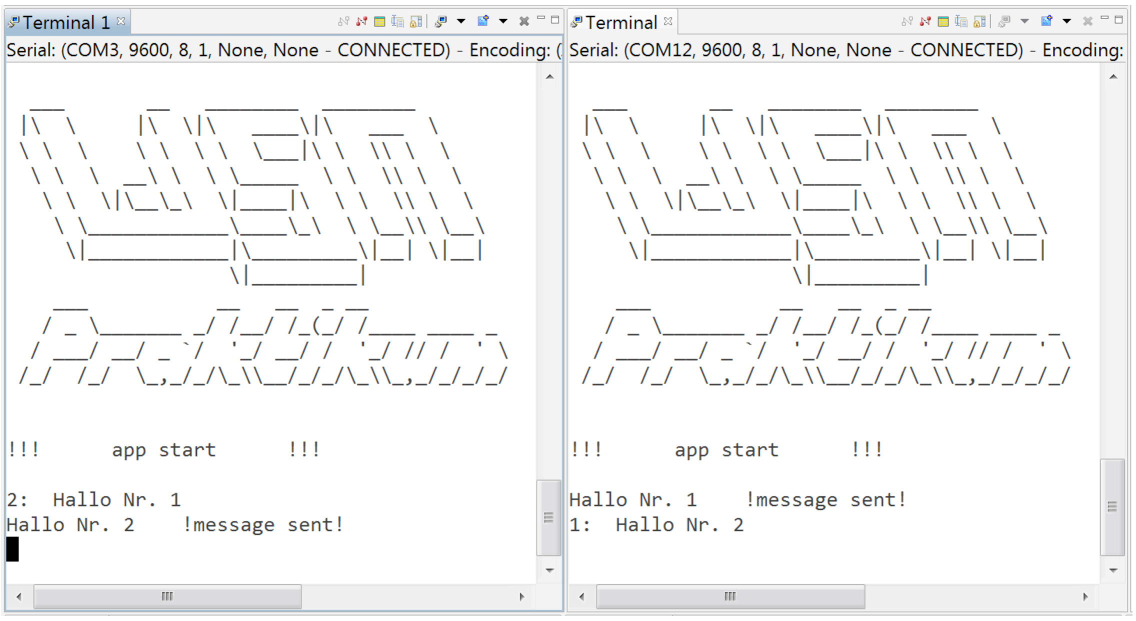
   Receiver:

   - Use the RSSI_calculate(uint8_t rawValue) function to get the RSSI value of the last received packet in dbm (The raw RSSI value is located at packet.rxMetrics[0])
   - Sum up the RSSI values till the STOP packet and divide them by the number of received packets
   - Print the result of the communication to the Terminal (the result of the division and the number of received packets)

```
int8_t RSSI_calculate(uint8_t rawValue)
{       int16_t rssi;
        if(rawValue >= 128)
                rssi = (int16_t)(rawValue - 256)/2 - 74;
        else
                rssi = (rawValue/2) - 74;

        if(rssi < -128)
                rssi = -128;
        return rssi;
}
```

Simple Chat

1. Realize a wireless chat between all of the groups.
   - Use an ISR of the UART and write every character received into packet.frame[i] and the number of received chars into packet.frame[0]. You should also use printf() or the UCA0TXBUF to send the command back to your own Terminal, so that you can see what you are typing
   - If the received character is 0x0D (carriage return) (Enter Key) you can send the packet with the MRFI_Transmit() function
   - As the first characters of every message, you should send your own ID (like "1: " for group one)
   - For receiving the message you must use the MRFI_RxCompleteISR_new( ) function. Use printf("%c",packet.frame[i]) in a for loop. After every message use printf("\r\n") to start a new line



```
Serial: (COM3, 9600, 8, 1, None, None - CONNECTED) - Encoding: (

 ___       __   ________  ________
|\  \     |\  \|\   ____\|\   ___  \
\ \  \    \ \  \ \  \___|\ \  \\ \  \
 \ \  \  __\ \  \ \_____  \ \  \\ \  \
  \ \  \|\__\_\  \|____|\  \ \  \\ \  \
   \ \____________\____\_\  \ \__\\ \__\
    \|____________|\_________\|__| \|__|
                 \|_________|

   ___           __   __  _ __
  / _ \_______ _/ /__/ /_(_) /____ ____ _
 / ___/ __/ _ `/  '_/ __/ /  '_/ // /  ' \
/_/  /_/  \_,_/_/\_\\__/_/_/\_\\_,_/_/_/_/


!!!      app start      !!!

2:  Hallo Nr. 1
Hallo Nr. 2    !message sent!
```

```
Serial: (COM12, 9600, 8, 1, None, None - CONNECTED) - Encoding:

 ___       __   ________  ________
|\  \     |\  \|\   ____\|\   ___  \
\ \  \    \ \  \ \  \___|\ \  \\ \  \
 \ \  \  __\ \  \ \_____  \ \  \\ \  \
  \ \  \|\__\_\  \|____|\  \ \  \\ \  \
   \ \____________\____\_\  \ \__\\ \__\
    \|____________|\_________\|__| \|__|
                 \|_________|

   ___           __   __  _ __
  / _ \_______ _/ /__/ /_(_) /____ ____ _
 / ___/ __/ _ `/  '_/ __/ /  '_/ // /  ' \
/_/  /_/  \_,_/_/\_\\__/_/_/\_\\_,_/_/_/_/


!!!      app start      !!!

Hallo Nr. 1    !message sent!
1:  Hallo Nr. 2
```

Now you can communicate with all other groups in a chat