CSE466 LAB 5 – SimpliciTI Networks
AUTUMN 2012

OBJECTIVES

In this lab you will DO the following:

• Set up a static radio link between two eZ430-RF2500 2.4 GHz radio transceivers and the SimpliciTI network stack
• Implement a wireless UART relay system using the two transceivers
• Develop an interface for wireless control of the robotic blimp

In this lab you will LEARN the following:

• How to configure and use the SimpliciTI wireless networking stack from TI.

DELIVERABLES

At the beginning of the next lab period you will demo your working system to the TA. Be prepared to demonstrate any and all functionality developed throughout the lab assignment.

Prior to the beginning of your next lab period, each group should turn in the following to the course dropbox (one set per group):

• Your fully commented and neatly presented code, in a zipped format.
  
  **NOTE:** Only source files (*c, *.h, etc) are necessary. Make it obvious which files were used in which portion of the lab.
• A PDF description of any extra functionality you implemented, if applicable.

RESOURCES

These documents and web resources will be useful in completion of the lab and/or in answering the questions posed. Additional resources are listed in the text of the lab.

• MSP430F2013 Datasheet
• MSP430x2xx Family User’s Guide
• MSP430 Code Examples
• MSP430 Software Coding Techniques
• eZ430-F2013 User’s Guide
• Code Composer Wiki
• Code Composer v5 User’s Guide for MSP430
• MSP430 Optimizing C/C++ Compiler User’s Guide
PART 1: SIMPLE PEER-TO-PEER LINK

In this lab you will be working with the popular eZ430-RF2500 radio module, which uses the CC2500 radio chipset by TI along with a MSP430F2274. This radio operates in the 2.4 GHz band, and is considered an IEEE 802.15.4 compatible radio. You will be using a radio network stack provided by TI, known as SimpliciiTI.

The goal of this first section is to ensure that both of your radio transceiver boards are working, and to familiarize yourself with application development using the SimpliciiTI network stack.

If you’re using your own computer, you will need to download and install SimpliciiTI from: http://www.ti.com/tool/simpliciti
This download includes documentation and example applications.

PROCEDURE:

1. Find the SimpliciiTI installation directory at “C:\Texas Instruments\SimpliciiTI-CCS-XX.XX.XX\”
2. Copy this entire directory to your user folder of choice
3. Open “\Documents\SimpliciiTI Sample Application User’s Guide.pdf”, and navigate to section 2.3.2 (“Using Code Composer Studio”).
4. Follow the steps in this tutorial, but replace “C:\Texas Instruments\...” with the user folder to which you copied the SimpliciiTI components (You will eventually be making project-specific changes to the SimpliciiTI source files, so it’s useful to keep a separate copy of these files for each of your projects).
5. Read through the Simple Peer-to-peer sample application and be sure you understand the purpose of each line. Reference the SimpliciiTI API document (in “\Documents\”) for more information about SimpliciiTI calls.
6. Make sure you’re able to get your eZ430-RF2500 boards talking properly (synchronized blinking lights).

PART 2: STATIC LINKS WITH SIMPILCIITI

In this section you will use the “commission” functionality of the SimpliciiTI API to produce a static link between two SimpliciiTI end devices. Your two eZ430-RF2500 development boards will be these two devices.

The link procedure found in a typical SimpliciiTI application (e.g., the Chronos watch and USB transceiver) is not based on the device address of the nodes being linked but instead is based only on which nodes are currently in “Link” or “Link Listen” mode. A button push on the receiver followed by one on the transmitter causes the two devices to link together. This is a nice method for sensor networks, but there is an obvious downside to this method in cases where the link procedure needs to be more selective. Accidental unwanted links are very possible using this method, which is a serious concern/annoyance when the link is used to pass control information to a robot (your blimp). Instead of using this push-button linking method, we will hard-code the link procedure in our SimpliciiTI application using SMPL_Commission().

PROCEDURE:

1. Program one of your RF2500 target boards with the LinkListen code, and the other with the LinkTo code. Ensure that they reliably link and communicate using the unmodified demo application (you should have done this in Part 1).
2. Open the “SimpliciiTI API” from the Documents folder and thoroughly read the section about the SMPL_Commission() call.
3. Modify the LinkListen and LinkTo applications to add selectivity to the link procedure by using the SMPL_Commission() function call. Remove all unnecessary code from the demo application (starting from scratch will probably be easiest). You will want to rename the project, build configurations, and application source files to reflect the new nature of the two devices.
4. Write an application in which the button on each of the RF2500 boards controls one of the indicator lights on the other RF2500. If you want, you can use this to talk to your lab partner across the room via morse code.
PART 3: UART RELAY

The recommended method of implementing a wireless interface to the blimp is by using the eZ430-RF2500 boards as a sort of UART relay. The USCI module on the F2274 implements UART in hardware, so there is no need to use your software UART module.

The eventual purpose of this wireless UART relay is as a data/control link between your computer/controller and the blimp. In this section, you will simply develop the bidirectional wireless relay link.

GOALS:

- Implement a bi-directional wireless UART relay. You MUST use a static link method, as in Part 2.

HINTS:

- Only one of your debuggers has a UART <-> USB interface. You will need to either:
  - Find a creative solution to debug and test your system without having a PC on each end.
  - Get a FTDI USB <-> UART bridge from the TA and use it on one or both ends of the relay link.
  - Get a blimp board from the TA and write a simple UART “echo” application for the blimp microcontroller to ease testing of your UART relay.

EXTRA CREDIT SUGGESTIONS

- Use a baud rate higher than 9600. Keep in mind that your debugger will NOT support this higher baud rate, so you will need to find another testing method.
- Graph error rate (bits, bytes, or packets) as a function of transmit distance for your wireless UART relay.