

Project Final Report

Encrypted Chat

17 April 2011

Nayan Nandihalli

[Project Abstract](#)

[Status](#)

[Specification](#)

[Parts](#)

[Software Modules](#)

[Overall Design](#)

[Implementation & Construction](#)

[LCD Interface](#)

[PS/2 Interface](#)

[RF Interface](#)

[Flow Chart](#)

[Block Diagram](#)

[Picture](#)

[Retrospective](#)

[Attachments](#)

Project Abstract

The original project design consisted of 2 Zilog ZNEO development boards able to communicate over RF using RF transceivers (Part # RFM12B). Unfortunately, despite over 20 hours of debugging, I was unable to get the parts working. I decided to fall back on serial port communication.

One board will be connected to a PS/2 keyboard and the other will have an LCD to display text as it is received. These characters will be encrypted before being sent over the serial link and decrypted on the receiving side before being displayed on the LCD. Encryption will consist of Diffie-Hellman for the initial session key exchange and the following symmetric encryption algorithms:

- DES
- RC4

The development board pushbuttons will allow the user to resynchronize the session key and change the symmetric encryption algorithm.

Status

The project currently supports bi-directional communication between the two ZNEO boards. Pressing the pushbuttons on either board will initiate a key resynchronization and/or a symmetric cipher change. The ZNEO reads bytes input from the PS/2 keyboard and encrypts and transmits them over the serial link. The receiving side displays both the encrypted value and the decrypted character on its LCD.

I could not get the RF transceivers working. Because of this, I had to fall back on a serial link between the ZNEO boards.

Specification

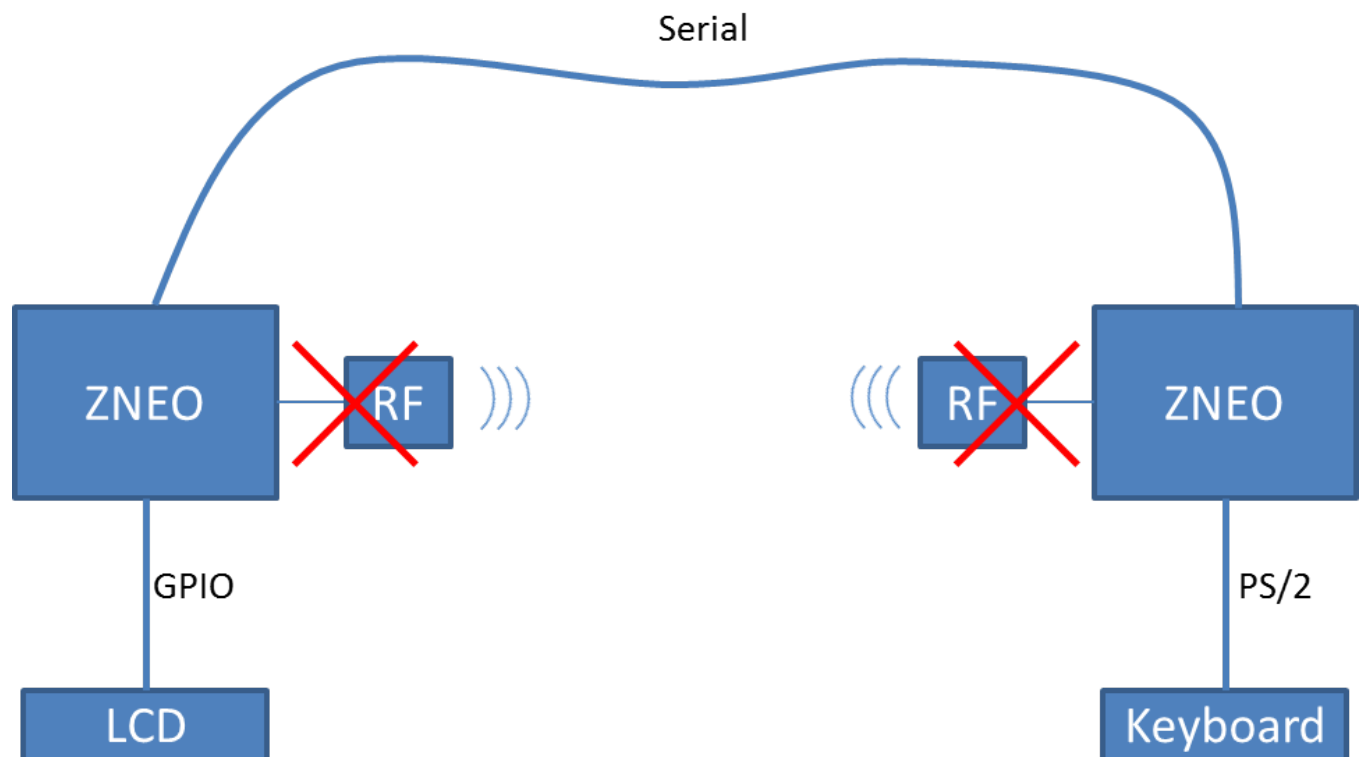
Parts

- ZNEO Z16F Development Board (2)
- MDL16166 - Liquid Crystal Display
- PS/2 Connector
- PS/2 Keyboard
- RFM12B - RF Transceiver (2) (not used)

Software Modules

- ChatComms - Chat protocol implemented over serial
- SerialComms - Low-level serial communications over GPIO
- Diffie-Hellman - Key exchange library
- LCD - Driver for LCD
- PS/2 - Keyboard driver that interprets scan codes
- RC4 - Encryption library (modified from internet)
- DES - Encryption library (modified from internet)
- SPI - Library to facilitate communication over SPI (not used)
- RFM12B - Wireless transceiver library (not used)

Overall Design



A keyboard is connected to one board via PS/2 and GPIO. This board reads the scan codes coming from the keyboard and translates them into ASCII characters, which it passes along for transmission. Another keyboard could easily have been added to the other board with another PS/2 connector.

An LCD is connected to the board without the keyboard and is used to display data received over the serial link. The 16-character LCD is divided into two 8-character sections. On the left

side, the actual decrypted characters are displayed. On the right side, the hex representation of the encrypted bytes received over the serial link are displayed. Depending on the encryption mode of the boards, this can be either:

1. The ASCII code of the character (Encryption off)
2. The RC4 encrypted character (RC4)
3. The first 4 bytes of the 8-byte DES encrypted block (DES)

Due to the limited space on the LCD, it was impossible to display the entire encrypted DES block. Displaying the first 4 bytes should be sufficient to prove that the board is operating DES in feedback mode since repeated key presses generate different encrypted blocks.

Both boards display the current encryption mode on their LED displays. This mode can be changed using the pushbuttons on the development board as follows:

- SW1 - Resynchronize session key
- SW2 - Switch to DES and resynchronize session key
- SW3 - Switch to RC4 and resynchronize session key

If there is an error during the key resynchronization, one or both boards will display "Err". In this case, simply attempting to resynchronize again should solve the problem.

Implementation & Construction

LCD Interface

The LCD used is part MDL-16166. This part has 16 pins that are used to interface with a microcontroller, although 2 of these pins (15-16) are related to a (nonexistent) backlight and are therefore unused. These pins were connected to the ZNEO GPIO ports as follows:

- 1 Vss GND
- 2 Vdd +5V
- 3 Vo (contrast) GND
- 4 RS PH3
- 5 R/W GND
- 6 E PH0
- 7-14 DB7:DB0 PB7:PB0

The LCD microcontroller understands commands that do things like move the cursor, shift the display, etc. The microcontroller is smart enough to understand ASCII characters, so conversion is not necessary like it is for the LED display.

PS/2 Interface

The PS/2 keyboard is connected to the ZNEO development board via GPIO, with the assistance of a PS/2 pin dangle. This dangle takes the PS/2 keyboard connector and rearranges the pins so that they can be connected to the GPIO pins. PS/2 consists of 6 pins, but I only connected 4 of them, as follows:

- CLK PD0
- GND GND
- DATA PD1
- Vcc Vcc

Once the keyboard was connected, modified the library I built for one of the class labs to work as needed for my final project.

RF Interface

The radio frequency transceivers I received for the project operated using an SPI interface. In order to get them working, I had to build both an SPI and RF library. I initially started by modifying the SPI library I built for one of the class labs. This code used the SPI interface on the ZNEO board to interface with the device. During the course of my debugging, due to the issues I had getting the SPI lab working, I decided to implement my own version of SPI using GPIO. This is the version that I turned in with all of the other code.

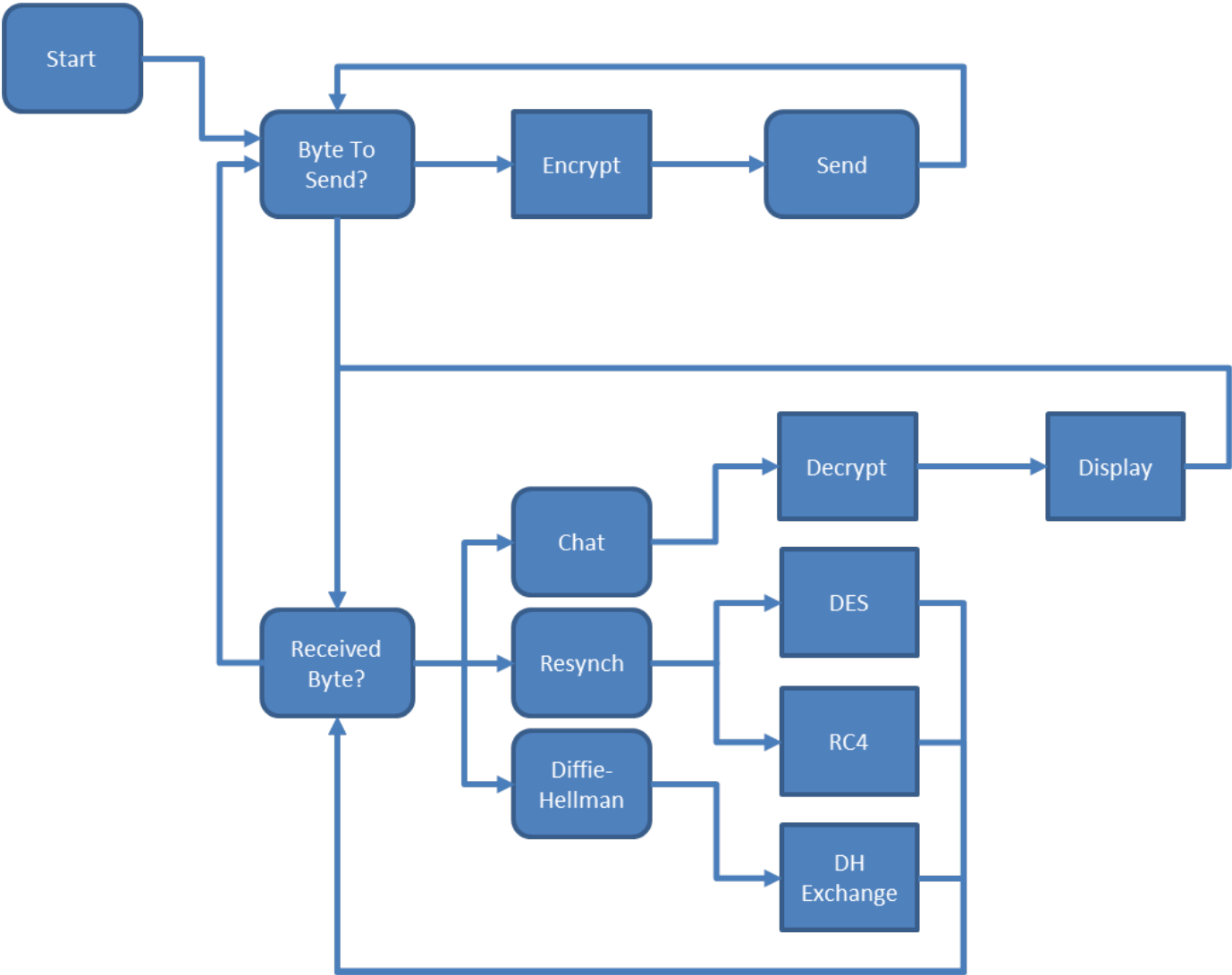
In addition to the SPI library I built, I also built a library to interface with the RF transceivers. Originally, this library was built upon the sample code found on the manufacturer's website¹. Unfortunately, I quickly discovered that this code was buggy. For example, it contains different functions for sending and receiving, but in both the code enables the "transmit" ability instead of transmit on one and receive on another. After scouring the internet, I found a website that claims to have fixed the bugs in the manufacturer's code², but still was not able to get this code working.

In the end, I suspect that the problems I encountered getting the transceivers working was related to the way I had connected them to the development boards. The transceiver pins were smaller and closer together than the development board pins, and therefore caused problems with short-circuits and loose connectors. A better way to connect these parts would probably be called for in the future.

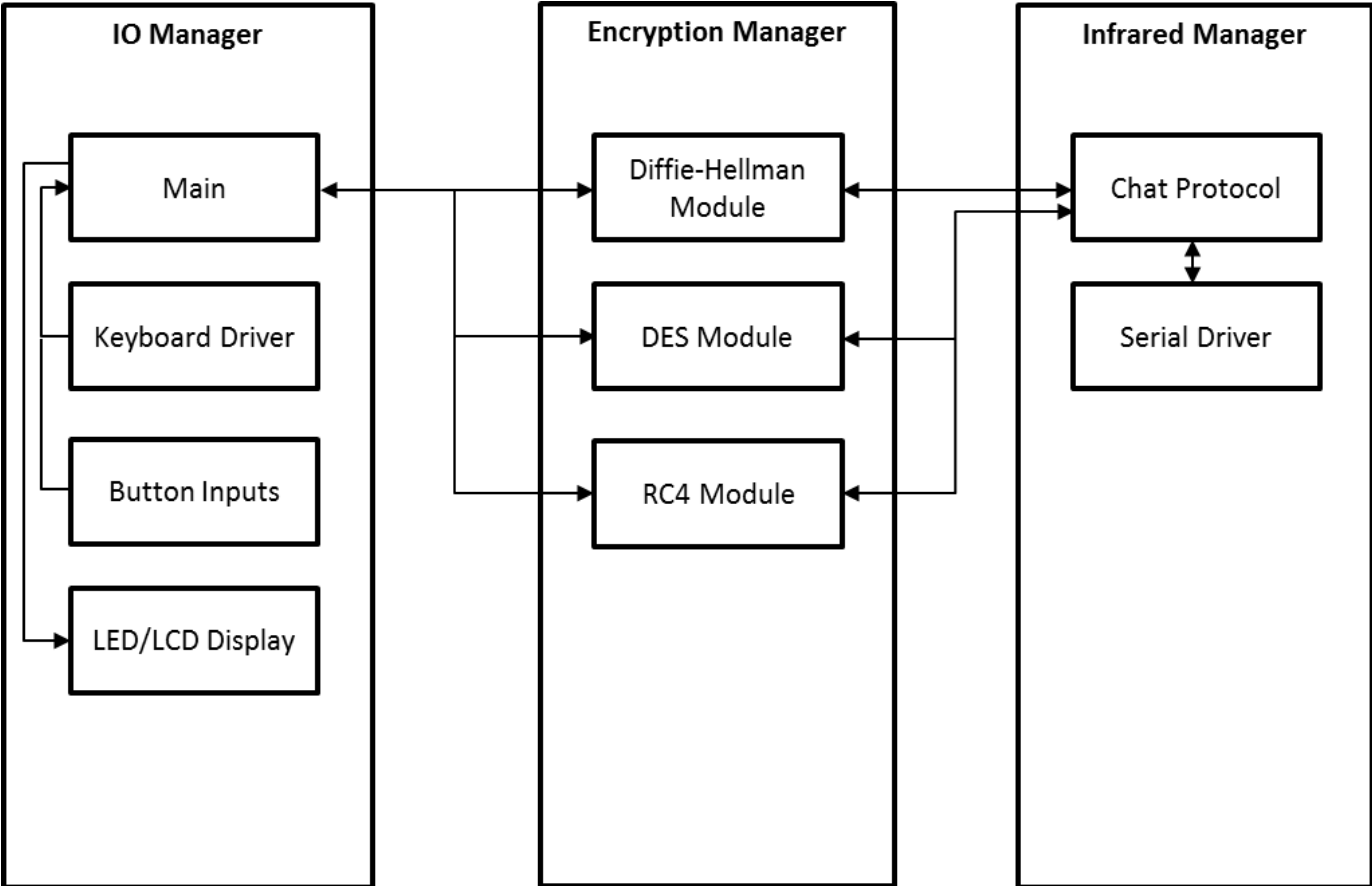
¹http://www.hoperf.com/upfile/RF12B_code.pdf

²http://loee.jottit.com/rfm12b_and_avr_-_quick_start

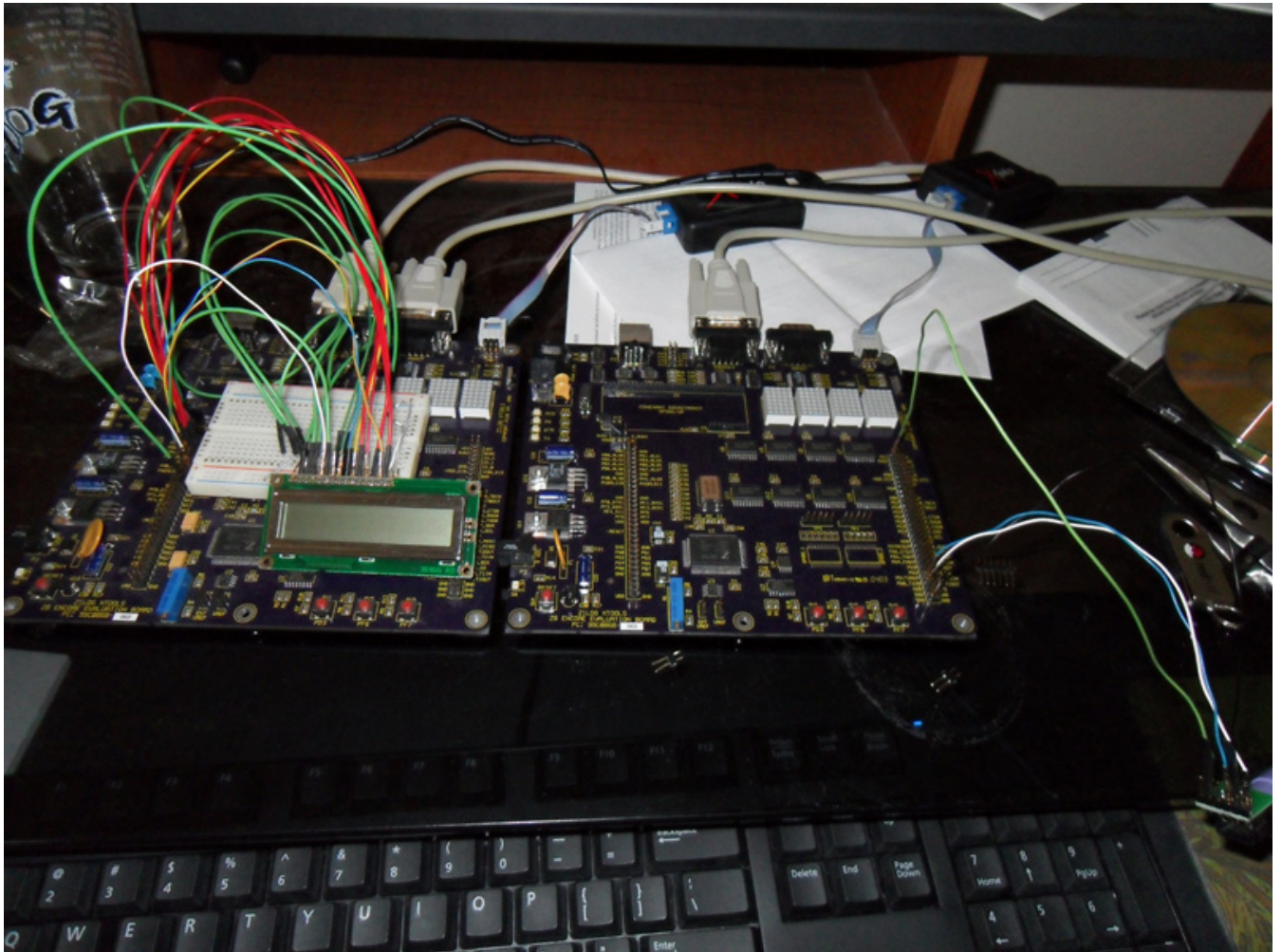
Flow Chart



Block Diagram



Picture



Retrospective

When I decided to write an encrypted chat program for the development boards, I knew that it would be a software-heavy design. The only variable would be how to communicate between the two boards. After deciding to use the RF transceivers to communicate between the boards, I realized that they used SPI as their interface. Unfortunately, SPI was the lab with which I had the hardest time and I knew there would be issues getting them working. Additionally, picking a hardware component in which I had no ability to debug (no way to capture wireless signals) was a mistake. Between these two issues, I was unable to get the parts working. If I had to do the project again, I would:

1. Pick a part that uses an interface technology I am familiar with and feel confident about working with.

2. Pick a part that I am able to fully control while debugging.

Attachments

Please find the following included in the zip package:

- All source code to the final project (Project & Project 2 use most of the same source code base)
- Schematics for LCD part
- Schematics for RF parts
- Pictures of project before and after giving up on RF