Wireless Space Team

Due Date: Friday, November 3

Resources

THIS LAB REQUIRES READING DOCUMENTATION AND DATASHEETS. Before you ask an officer a question, make sure your answer is not in the slides, this document, the datasheet, or the documentation for the library.

Lecture 1 Slides: **Link**
NRF24L01p Datasheet: **Link**
RF24 Documentation: **Link**
Arduino SPI Documentation: **Link**
Teensy LC Pinout: **Link**
Arduino Pro Mini Pinout: **Link**
Teensyduino Installer: **Link**

Overview

In this lab, you will be implementing a game similar to the **Space Team** app for smartphones. You will be using two different microcontrollers connected wirelessly using the Nordic NRF24l01P modules. One microcontroller will be used to generate a random sequence of LED blinks, and the other microcontroller will have a series of tactile buttons which the player is expected to press in the same order as the sequence of LED flashes.

This lab is designed to teach you how to write to certain registers to set parameters, and to familiarize yourself with the radio modules, which will be used for the quadcopter.

Lab Outcomes and Expectations

By the end of this lab, you should be familiar with setting and clearing bits in designated registers and how data is transmitted via SPI. In order to be “checked off” for this lab, you must complete each checkpoint. Every team member is expected to have contributed to the lab, and may be quizzed on any of the information covered.

A. Assembling the Hardware

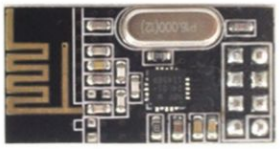
Your kit contains both a Teensy LC and an Arduino Pro Mini, but neither has headers attached to it. For the Arduino, you need to solder a row of male headers onto the left and right sides, and the angled male headers onto the front (the DTR, TX0, RXI, VCC, and GND pins). Do not attach headers on pins A4 to A7, or the GND pin on the back of the Arduino, or else you will not be able to breadboard it. For the Teensy, solder a row of headers on the left and right sides, but do not attach headers to pins 24, 25, AREF or the back row (except for pins 12 and 13; attach headers to those), or else you will not be able to breadboard it.

In order to program the Arduino, we must use an external FTDI programmer. Connect the programmer and the programming pins using a set of female-female jumper cables as follows:

Arduino Pro Mini Programming Pins	Programmer
VCC	VCC
GND	GND
DTR	DTR
Rx	Tx
Tx	Rx

To wire up the radio module using a set of male-female jumper cables (or a set of male-male jumper cables connected to a set of female-female jumper cables), refer to the following diagram. The pins are the same for both the Teensy and the Arduino.

Wiring Setup:



- CE, CSN do not need to be these pins
  - you initialize them in the constructor
  - RF24 radio(CEpin,CSpin);
- Use Female-Male jumper cables to connect

Signal	RF Module	COLOR	Arduino pin for RF24 Library
GND	1	Brown	GND
VCC	2	Red	3.3V
CE	3	Orange	9
CSN	4	Yellow	10
SCK	5	Green	13
MOSI	6	Blue	11
MISO	7	Violet	12

NOTE: FTDI programmers, MiniUSB cables, and batteries are lab stock. Once you are done using them in the lab, return them to their respective boxes on the shelf below the kits. DO NOT KEEP THEM.

On the Teensy Side:

The Teensy side of the Space Team game will be the “receiver”, which includes the red, yellow, and green LEDs. LEDs are in the AP parts drawers. Use this pinout to help you in selecting pins for the Teensy receiver. In order to upload code on the Teensy, you must install the Teensyduino plugin for Arduino. You can find installation instructions here.

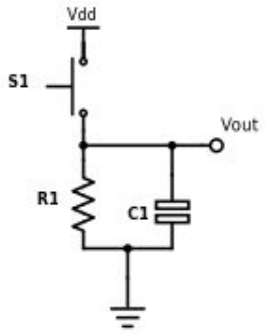
NOTE: LEDs will burn out if more than 20mA of current passes through them, and usually you want to limit current to <=5mA to save energy and to prevent the LED from being too bright. Make sure to include a current-limiting resistor in series with your LEDs.

On the Arduino Side:

The Arduino side of the Space Team game will be the “transmitter”, which includes the three push buttons corresponding to each of the LEDs. Push buttons are in the AP parts drawers. Use this pinout to help you in selecting pins for your Arduino transmitter.

NOTE: When you press or release a button, it will “bounce” several times before reaching its final state, which means you will register several button presses. One way to avoid this is to put a capacitor in series with the button, so that the button release will not register until after the capacitor is fully discharged, which should be after the button

has finished bouncing if the capacitor was chosen correctly. Below is the basic circuit for button debouncing:



Where V_{out} is the voltage read by the microcontroller pin. Choose the resistance and capacitance to get an RC time constant of approximately 1ms.

Using a Battery:

To power your microcontroller with a battery, connect the positive terminal of the battery to V_{in} (on the Teensy) or RAW (on the Arduino). DO NOT connect the battery to your microcontroller if the USB cable or FTDI programmer is connected. This will short your circuit and damage your computer and/or microcontroller. The AP parts drawer contains JST connectors that can be used to connect your battery to the breadboard.

Checkpoint 1: Solder headers onto both of your microcontrollers, and draw full schematics for both your receiver (Teensy side) and transmitter (Arduino side).

B. Editing the Library

In order to do the following steps to set up the radio modules, you must first finish the implementation for certain functions in the required library. The first part of the lab requires you to put the **folder** in the Arduino/libraries folder on your computer. In the folder, you will notice that certain sections in the "nRF24L01.h" and the "RF24.cpp" file have been marked as "TODO: START HERE" and "TODO: END HERE". DO NOT MODIFY ANYTHING ELSE. Your main source of filling out this information is the SPI commands, SPI timing, and register map sections of the NRF24l01P **datasheet.**

NOTE: While in the slides we used the name SS to refer to the slave select line, this datasheet uses the name CSN (chip select negative, because it is an active low line).

Other datasheets may use other terms, such as CS or nSS. No matter what the name, this pin always behaves the same way.

The write_register and read_register functions correspond to the W_REGISTER and R_REGISTER SPI commands respectively. When implementing those two functions, do not call any other functions from the file: you should only use the digitalWrite function and the Arduino SPI **library**. Specifically, you will need the SPI.beginTransaction, SPI.transfer, and SPI.endTransaction functions. The SPI.beginTransaction function requires you to know the maximum SPI clock speed, the bit order, and the phase/polarity: clock speed and bit order are specifically mentioned in the datasheet, while you can find the phase and polarity by looking at the timing diagrams, and comparing them to those in the lecture. To control the SS pin from within RF24.cpp, use the variable *csn_pin* with digitalWrite.

NOTE: To set bits, use bitwise OR. To specify which bits to set, use bitwise AND.

For example: If we have an 8-bit register *REGISTER* and we want to set the last bit to be a bit specified by some variable *bit*, do the following:

REGISTER = REGISTER | (bit & 0b00000001)

So if *REGISTER* is initially 0b10110000 and *bit* is 0b00000001, *REGISTER* will then be 0b10110001.

For the other four functions, you are modifying/accessing different registers on the module, so you should call the write_register and read_register functions within your implementation.

NOTE: On the Arduino, you must “#include printf.h” and call the printf_begin function before calling the printDetails function.

Checkpoint 2: Modify all marked parts of the RF24 library.

C. Communicating over the Radio Modules

Refer to **documentation** in order to do the following.

1. Initialize each radio using the constructor and the chosen chip enable (CE) and chip select (CSN) pins. CE allows for transitions between Tx and Rx mode. CSN is equivalent to SS in this application.
2. In the setup code, call the begin function to begin operation.
3. In the setup code, set the communication channel to be your team's number. This is an important step to avoid crosstalk.
4. In the setup code, set the power amplifier (PA) level to RF24_PA_MIN.
5. Check the datasheet for restrictions on the intended receiving and transmitting addresses. (For example, 0xc2c2c2c2c2 and 0xe7e7e7e7e7 are good values to use.) In the setup code, set the writing and reading pipes in both sets of code accordingly. Make sure the reading pipe address of the transmitter matches the writing pipe address of the receiver and vice versa.
6. Set the cyclic redundancy check (CRC) length to be RF24_CRC_16. CRC can be 1 or 2 bytes and is used for error checking purposes.
7. In the loop, begin listening when expecting data.
8. In the loop, call the available function to check when there are bytes to be read.
9. In the loop, stop listening when the bytes have been transmitted and received.

We suggest creating a simple sketch (for example, transmitting and receiving a single character) before beginning the code for Space Team. Since there are so many setup parameters, it would be a good idea to make a function initializing these parameters. This initialization function will look almost identically the same on both devices, except the reading and writing addresses should be swapped.

NOTE: Before uploading code, go to Tools->Board and set it to "Teensy LC" for the Teensy and "Arduino Pro or Pro Mini" for the Arduino. For the Arduino, also set Tools->Processor to "ATmega328P (3.3V, 8 MHz)".

D. Implementing Wireless Space Team

The final objective of this lab is to write a complete implementation of Space Team over the radio modules. The requirements for this game are as follows.

This game will center around the concept of “rounds”. In each “round”, the Teensy flashes a sequence of varying LED flashes to the player.

1. The very first round should have a sequence length of 1. Every subsequent round should increase the sequence length by 1, until the player fails to input the correct sequence.
2. With every round, the new sequence should be identical to the previous one, with the exception of the Teensy appending a new, randomly-generated item to the end of the sequence. (For example, if the current sequence is “RGGRYG”, then round 1 was “R”, round 2 was “RG”, round 3 was “RGG”, etc. The next sequence should be “RGGRYG*”, where * is any of R, Y, or G.)
3. A struct containing the sequence is sent to the Arduino over the radio module. The player must push the corresponding buttons.
4. If the player hits the correct sequence of buttons, the Arduino should send an indication to the Teensy, which should flash the green LED once before moving to the next round.
5. If the player hits the wrong sequence of buttons, the Arduino should send an indication to the Teensy, which should flash the red LED. You may choose to either have the program terminate here or restart from the beginning.

Checkpoint 3: Implement a fully working version of the Space Team adhering to the above requirements.

E. Summary

After finishing the lab, you should have completed the following:

---

1. Finished each checkpoint of the lab and were checked off by an Advanced Project lead for each. Your entire team must be present and understand how the code works.
2. Gained a solid understanding of how to apply SPI.