Lab 1: Introduction to Arduino Microcontrollers, Data Types, and Serial Communication

Due Friday September 2, 2016 @ 5pm PT on bCourses

Images developed using Fritzing.

Part 1: Getting Started with Arduino

Exercise 1: Arduino Installation Guide

Items:

- 1x Arduino Uno Boards with USB Cables

Deliverables:

- None

Directions:

1. Download and install the Arduino IDE software (http://arduino.cc/en/Main/Software)
2. Follow the Getting Started Guide and complete the Blink Example (http://arduino.cc/en/Guide/HomePage).
    a. We are using Arduino UNO boards.
    b. Note: A dialog box may or may not appear when the Arduino Uno is first connected to the USB port.

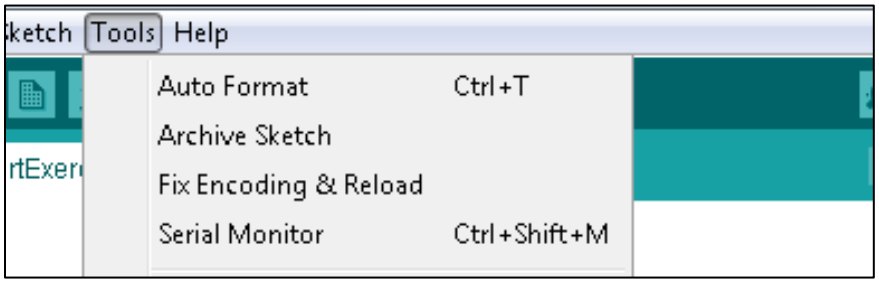
Exercise 2: The Serial Port Object

Deliverables:

- None

Directions:

1. Open the SerialPortExercise file (located in the Code folder of the Lab 1 Packet) and upload the program to your Arduino board.



2. In the Arduino IDE, open the Serial Monitor (ctrl+shift+M or shift-⌘-M). You should see a sequence of numbers printed to the screen.

Exercise 3: Expanding the Blink Sketch

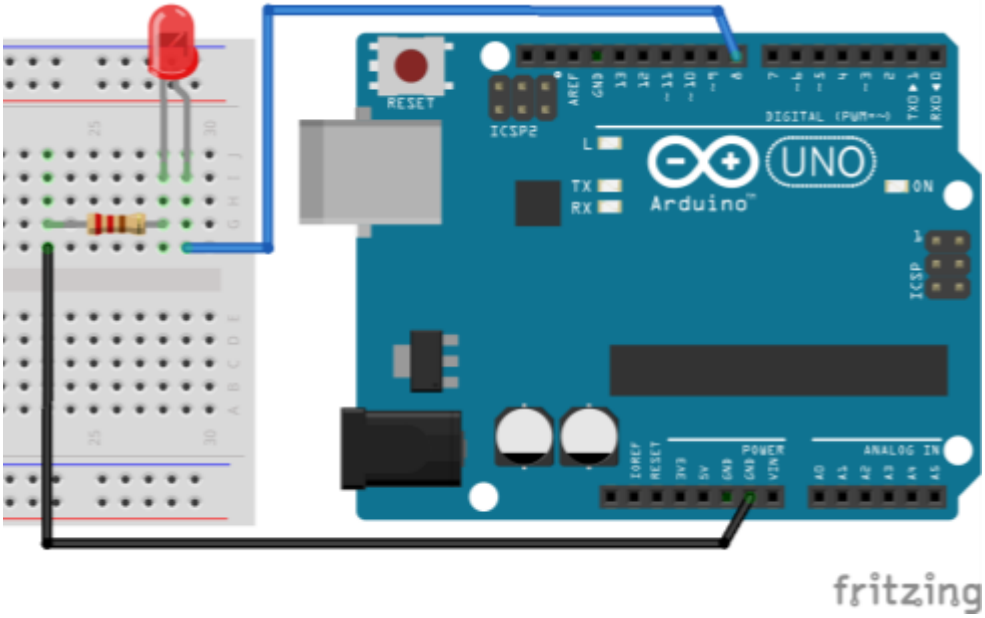
Additional Items:

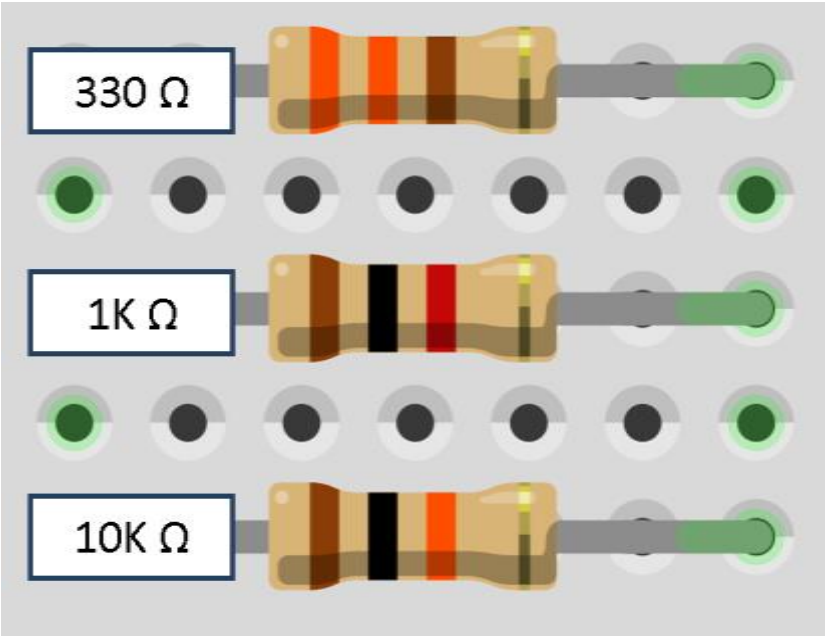
- 1x LED
- 1x Resistor (Between 200 and 1k Ohm)
- 1x Breadboard (A white plastic board with holes)
- Jumper Wires

Deliverables:

- Submit a file named BlinkLED.ino containing the program described below. Additionally, in your lab report, copy & paste your Arduino code into a code box.

Directions:





1. Add another LED to the Arduino, as shown in the image above. Connect the long pin (anode) of the LED to Pin 8 and the short pin (cathode) to one end of the resistor. Connect the other end of the resistor to ground (GND). If you are using an LED Breakout board, the + pin is the anode and the – pin is the cathode through a resistor. The image above shows the color codes for some of the most common resistors.
    a. Note: On a breadboard, each of the terminals in a numbered row are connected within the board. Thus, in the image above, the LED cathode (short pin) is connected to the resistor through the breadboard.
2. Open the Arduino Blink example (File->Examples->01.-Basics->Blink). Save the program to your computer with the name BlinkLED.
3. In the file, change the LED integer from 13 to 8.
4. Upload the program to the Arduino. The LED should begin blinking.

5. Edit the program such that the LEDs on Pins 8 and 13 both blink.
6. Define a function with the name "blinkLEDs". blinkLEDs() should:
    a. Return nothing.
    b. Accept one input: an integer specifying the duration of each blink in seconds.
    c. Turn on and off the LEDs attached to Pins 8 and 13.
    d. Write the function such that both LEDs are not on at the same time.
    e. HINT: Refer to https://www.arduino.cc/en/Reference/FunctionDeclaration
7. Edit the loop() function so that it calls the blinkLEDs() function and nothing else.

Exercise 4: The Serial Port Object

Deliverables:

- Submit a file named SerialPortForLoop.ino containing the program described below. Additionally, in your lab report, copy & paste your Arduino code into a code box.

Directions:

1. Create a program with the file name SerialPortForLoop.ino.
2. Use the code from SerialPortExercise.ino (located in the Code folder of the Lab 1 Packet) and write a program that prints the integers between 0 and 9 to the Serial port during each loop (i.e. add a "for loop" inside the loop() function).
    a. HINT: https://www.arduino.cc/en/Tutorial/ForLoop
3. After the new "for loop", print an empty line to the Serial port (i.e. Serial.println();).
4. Before the end of the loop() function, add a 1 second (1000 millisecond) delay to the program.
5. The output of your program should look similar to…

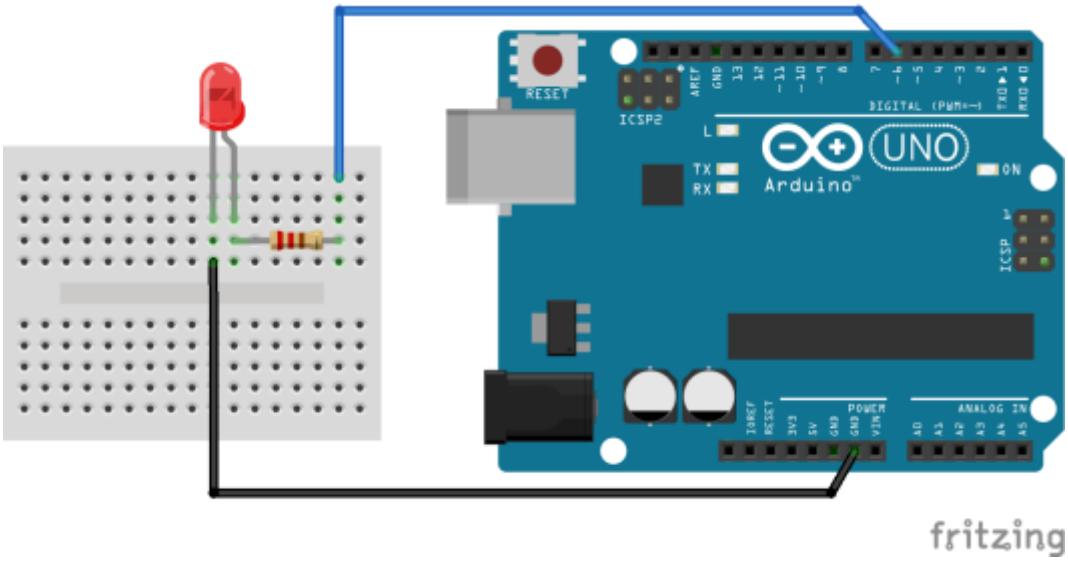
```
0
1
2
3
4
5
6
7
8
9

0
1
```

Exercise 5: Fade the LED with Pulse Width Modulation (PWM)

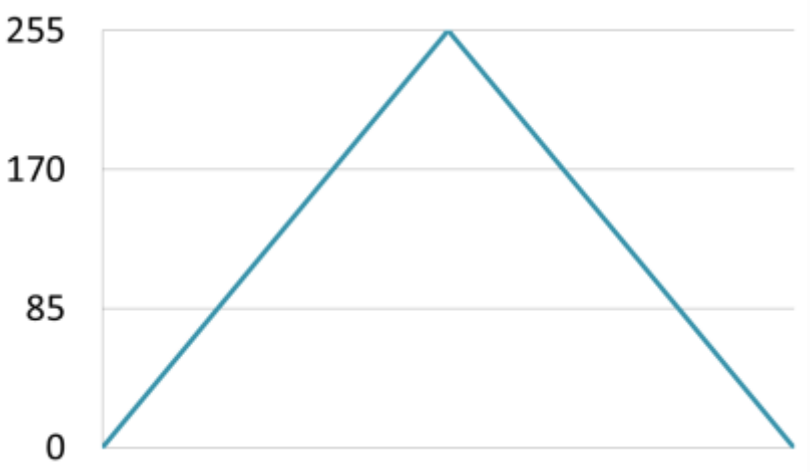
Deliverables:

- In your lab report, respond to Questions.

Directions:



1. The Arduino UNO has six digital pins capable of pulse width modulation (indicated with the tilde symbol, ~). With PWM, a digital pin can produce an analog signal rather than simply on (HIGH) or off (LOW). This allows the Arduino to output a specific voltage or a wave function. More explicitly, you can give the PWM pin an integer between 0 and 255 and the pin will output a voltage between (about) 0 and 5V. In this exercise, we will use a digital pin with PWM to incrementally fade an LED.
2. Connect the LED through the resistor to pin 6 (instead of pin 8).
3. Open the FadeLEDExercise file and upload it to your Arduino board. The LED should begin to blink with the intensity (roughly) represented by the function below.



4. Note the integer algebra in the FadeLEDExercise program. Microcontrollers are capable of floating point algebra, but it is generally slow and imprecise. Integer algebra is faster but may

not produce the results you expect. If you are doing algebra with integers and floats, pay attention to where integer rounding is (or is not) taking place.

```
// Calculate the LED brightness
// as a percentage (0 to 100)
// Note: int cannot have decimals
int percent = i*100/timesteps;
// Calculate the LED brightness
// as PWM output (0 to 255)
// Note: percent*255/100 does not = percent/100*255
int brightness = percent*255/100;
```

5. In the Arduino IDE, open the Serial Monitor (ctrl+shift+M or shift-⌘M). You will see a printout of each LED brightness value.
6. **Questions**:
    a. What happens if you increase or decrease the value of "timesteps"? Explain what "timesteps" represents.
    Hint: Try timesteps=1, timesteps=5, timesteps=100, timesteps=200.
    b. Logically, why could one argue that 256 is the maximum value of "timesteps"?
    Hint: See the Arduino documentation for analogWrite().
    c. Any problems with how FadeLEDExercise is implemented? Be specific.
    Hint: Check the variable types.
    Hint: If timesteps is 256, what are the possible values of percent and brightness?

Exercise 6: While…If…Else

Deliverables:

- Submit a file named SerialEchoExercise.ino containing the program described below. Additionally, in your lab report, copy & paste your Arduino code into a code box.
- In your lab report, respond to Questions.

Directions:

1. Upload WhileIfElseSerial program and open Serial Monitor in the Arduino IDE. Make sure you understand what the program is doing.
2. Open the program called SerialEchoExercise. Complete the code such that the Arduino reads characters being sent to the Serial port and then sends the same message back to the USB port. You will need to use the Serial object and the .available(), .read(), and .print() methods.
3. Edit the SerialEchoExercise code such that a period '.' is printed after every character when sent back to the USB port.
    a. Note: You may see "extra" periods appear. Not all characters, such as newline '\n', carriage return '\r', or tab '\t', are "visible" when printed to the monitor.

4. Add "if" statements to the SerialEchoExercise code such that the lower-case vowels 'a', 'e', 'i', 'o', and 'u', are replaced with the numbers '1', '2', '3', '4', and '5', respectively, when sent back to the USB port. For example, if you send "Hello" to the Arduino, "H.2.l.l.4." should be returned.
5. **Questions**:
    a. Type "engineering" into the serial monitor. What returns?
    b. Type "JacobsInstitute" into the serial monitor. What returns?

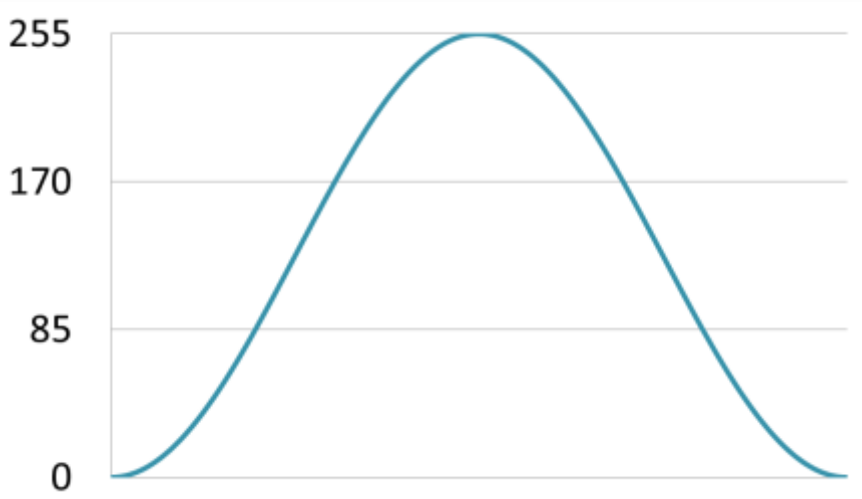
Exercise 7: Cosine Wave LED Fade

Deliverables:

- Submit a file named CosineFadeLEDExercise.ino containing the program described below. Additionally, in your lab report, copy & paste your Arduino code into a code box.

Directions:

1. Create a program with the file name CosineFadeLEDExercise.ino. Use the code from FadeLEDExercise .ino to write a program that fades an LED such that the output is a cosine wave like the one shown in the image below.
    a. Information about trig functions can be found at arduino.cc.
    b. You may find the approximation, radians = (degrees * 71) / 4068, to be helpful.
    c. Make sure to correct the integer math such that all integers between 0 and 255 could be sent to the PWM pin.
    d. Print each "brightness" value to the Serial port.



2. Edit the loop() function to allow a user to change the duration of the fade by sending an integer from the Arduino IDE's Serial Monitor.
    a. A variable representing the fade duration should be declared at the top of the sketch.
    b. You may find the Serial.parseInt() function to be helpful. (http://arduino.cc/en/Reference/Serial)
    c. It might help if the IDE's Serial Monitor is set to "No line ending."

Part 2: Variable Types and Serial Communication

Exercise 8: The Serial Port Object

Deliverables:

- None

Directions:

1. Open the ArduinoArithmetic program and upload it to your Arduino board.
2. The ArduinoArithmetic program includes 4 demonstrations, each showing some of the issues with using microcontrollers to perform basic arithmetic. Read through the code and make sure you understand what is being done.
3. In the Arduino IDE, open the Serial Monitor (ctrl+shift+M or shift-⌘-M). Send the numbers/characters '0', '1', '2', and '3' to run each demo.

Note: Each of the following exercises require a partner.

Exercise 9: Basic Software Serial Communication

Items:

- 2x Arduino Uno Boards with USB Cables
- Jumper Wires

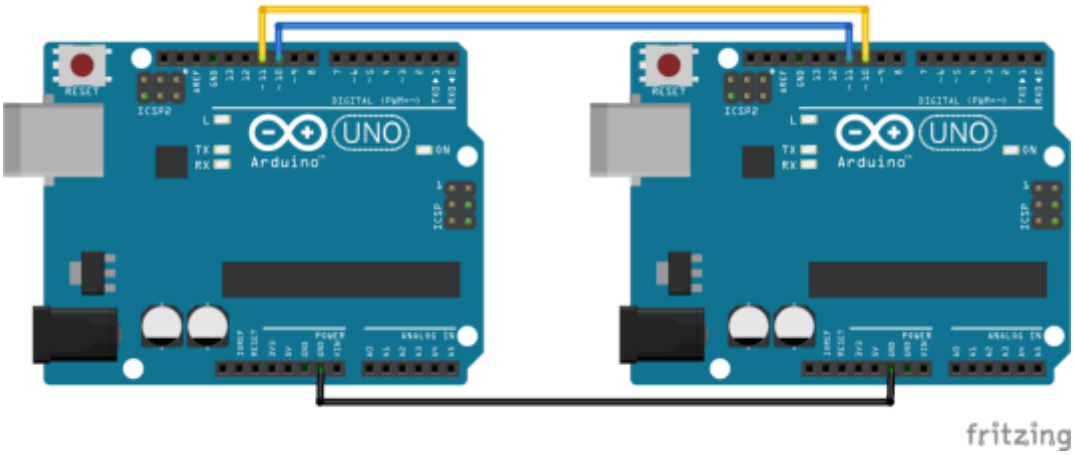
Deliverables:

- None

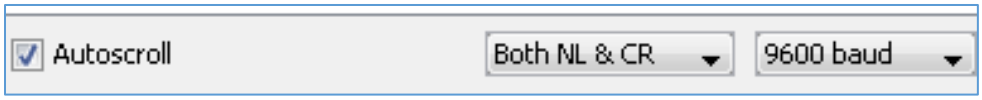
Directions:

1. With a partner, complete the following exercise.
2. Open the SoftwareSerialExercise.ino file with the Arduino IDE.
3. Notice that "include <SoftwareSerial.h>" is the first line of code. This is the syntax for importing the SoftwareSerial library into the sketch. Importing libraries allows us to extend the functionality of the Arduino environment with the functions and objects packaged into the library. See the Arduino Reference Page on Libraries for more.
   The Arduino Uno has only one hardware serial communication port, Digital Pins 0 and 1. Data is received by Pin 0 (RX) and transmitted by Pin 1 (TX). The SoftwareSerial library allows us to use 2 digital pins to create a software implementation of a serial port. In short, the library takes care of all the difficult programming and we can use a SoftwareSerial object just like the built-in serial object. Note: While it is possible to setup multiple SoftwareSerial ports, only one of these ports can receive messages (i.e. listen) at a time.

4. In this exercise, we will use Pin 10 for receiving (RX) and Pin 11 for transmitting (TX) data with the SoftwareSerial object.
5. With a partner, connect Pin 10 of one Arduino to Pin 11 of another Arduino and vice versa. For two devices to communicate over a Hardware Serial or Software Serial port, they must share a common ground. Therefore, connect the ground (GND) of each Arduino.



6. Upload the SoftwareSerialExercise program to your board and keep the Arduino connected.
7. Open the Arduino IDE's Serial Monitor (ctrl+shift+M or shift-⌘-M), type "Hello" into the text box, and click send. The text should appear on your partner's computer screen.
    a. Note: To break each message into a new line, change "No line ending" to "Both NL & CR" at the bottom of the Serial Monitor. Now, the Serial Monitor will send the "newline" and "carriage return" characters after every message you send.



8. Read the Arduino description of the .print() and .write() methods and make sure you understand the difference. In general, you should only use .write() to send a single byte or character.

Exercise 10: Basic Wireless Serial Communication

Additional Items:

- 2x XBee Series 2 (ZigBee) Radios with Power Regulator board

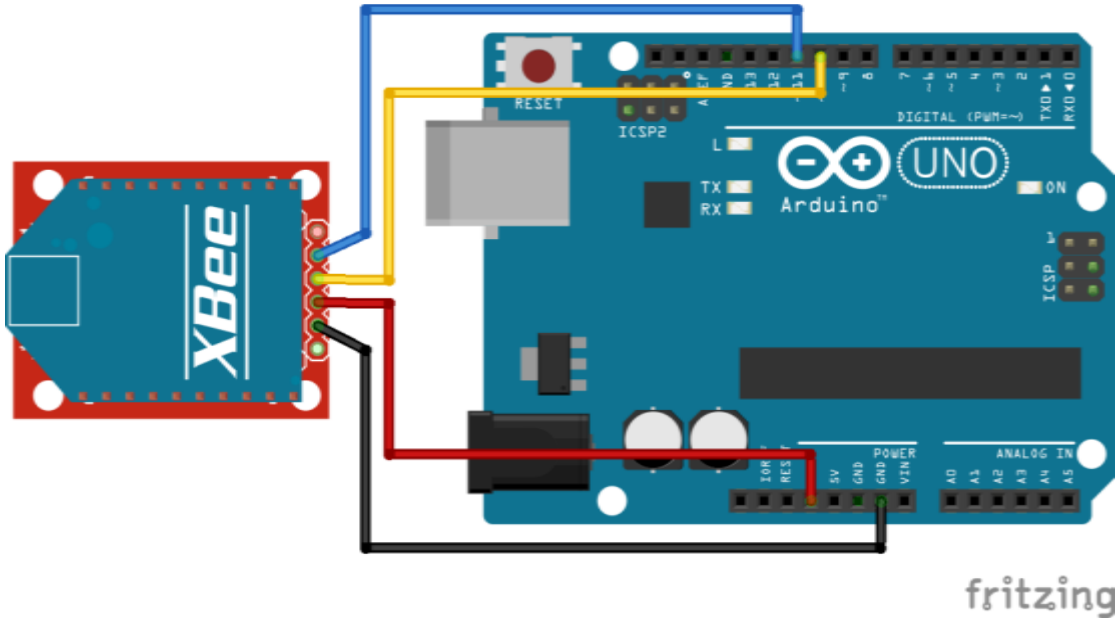
Deliverables:

- Submit a file named XBeeExercise.ino containing the program described below. Additionally, in your lab report, copy & paste your Arduino code into a code box.

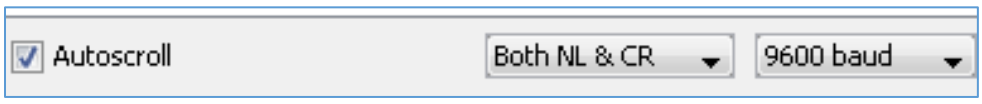
Directions:

1. With a partner, complete the following exercise.
2. (CE 186 Students: Skip This Step) Program two XBee S2s to communicate with each other in transparent mode.

   a. Radio #1 should have AT Coordinator firmware with the Address High and Low of Radio #2 set as the Destination High and Low.
   b. Radio #2 should have AT Router firmware with the Address High and Low of Radio #1 set as the Destination High and Low.
3. Using a Sparkfun XBee Regulated board, attach a radio to both Arduinos as shown below (UNO GND to XB GND, UNO 3.3V or 5V to XB 5V, UNO Pin 10 to XB DOUT, and UNO Pin 11 to XB DIN).



4. Open the SoftwareSerialExercise.ino file with the Arduino IDE and save as XBeeExercise.ino. In the program, change the variable name **mySerial** to **myXBee** to make it clear that the XBee is connected to the SoftwareSerial port (pins 10 and 11). Update the entire program accordingly.
5. Upload the program, open the Arduino IDE's Serial Monitor, type "Hello" into the text box, and click send. The chat program should now work wirelessly.
6. Edit the XBeeExercise such that the characters a through z and A through Z are transmitted as a hexadecimal string. In other words, when these characters are sent to the XBee by the SoftwareSerial port, the port should encode the character as a HEX string.
   a. Hint: mySerial.print( ) can accept more than one parameter.
   b. Check: If you send the string "Hi" from one Serial Monitor, the characters '4', '8', '6', '9' should appear on the other monitor.
   c. Note: If set to "Both NL and CR", the Serial Monitor will send the "newline" and "carriage return" characters after every message.



Exercise 11: Sending Commands

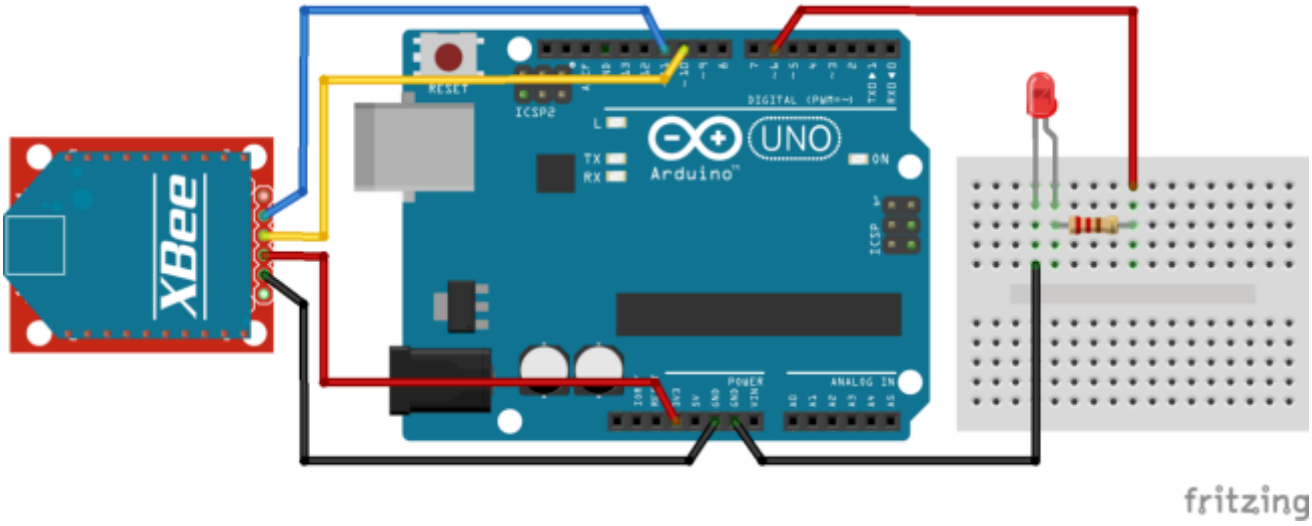
Additional Items:

- 1x LED
- 1x Resistor (200 to 1k Ohm)
- 1x Breadboard

Deliverables:

- Submit a file named XBeeLEDExercise.ino containing the program described below. Additionally, in your lab report, copy & paste your Arduino code into a code box.

Directions:



1. In this exercise, we will put together a system to remotely adjust the intensity of an LED. Using the Arduino IDE's Serial Monitor, users will send a command containing an integer between 0 and 255 to the Arduino connected to their computer. The Arduino will wirelessly send the command to another Arduino using two XBee radios. The receiving Arduino will interpret and implement the command (In this exercise, the receiving Arduino will dim the LED).
   You will again be working with a partner. The sending Arduino should be running the SoftwareSerialExercise program (unmodified) and the receiving Arduino should be programed according to the following directions.
2. Connect the short end (cathode) of an LED to ground and the long end (anode) to Pin 6 through a resistor. The XBee board should still be connected as described above.
3. Open the SoftwareSerialExercise program and save it as XBeeLEDExercise.ino.
4. To begin, we need to define the commands that the receiving Arduino will understand. To start, let's use the character "c" followed by an integer as the command to change the LED intensity. We will use ";" to mark the end of a message. In other words, sending "c0;" from the Serial Monitor will turn off the LED and "c255;" will turn the LED to full brightness.
5. To implement this behavior in the XBeeLEDExercise program, go to the while loop for the SoftwareSerial object and write an "if" statement that checks if the characters "c" or ";" have

been received. If "c" has been received, then we expect an integer to be coming. If ";" has been received, then we have reached the end of a message. You can use the parseInt() method instead of the read() method to receive the number as an integer instead of a series of characters. Finally, use the analogWrite() function to set the intensity of the LED.

6. Once you have this basic functionality working, add the following features:
    a. Report "OK" back to the Sending Arduino to verify that the message has been received and the LED intensity adjusted.
    b. Check that the integer is between 0 and 255. Report an error/warning message to the Sending Arduino otherwise.
    c. When a new intensity is received, adjust the LED brightness slowly/incrementally from the previous value to the new value using a "for" loop (or loops).
    d. Add parameters to the command to adjust the delay (in milliseconds) between each incremental change to the LED brightness (i.e. the command "c0d500;" dims the LED with a 500 millisecond delay between each adjustment). The total time to change the LED intensity should depend on the difference between the current and previous brightness. Lastly, the delay parameter should be optional (i.e. both "c0;" and "c0d500;" are valid commands).