Communication Part 2: Communication Using IR

In our first exercise on serial communication, we investigated serial protocols designed to exchange information between computers. In this exercise, we will work with protocols used by Infra-Red (IR) remote controls. IR remote controls are ubiquitous; they control your TV, your Xbox, your garage door… IR is arguably the communication medium of choice for short-range wireless applications because it's simple and inexpensive. It has become very popular for use with embedded systems such as our Arduino, and, fortunately for us, this surge in popularity has been accompanied by a surge in good on-line documentation. We will begin by looking at the basics of communication using infra-red light. Read the attached article and be prepared to answer the following questions:

1. What is the wave length of infra-red light?
2. How do you increase the range of an IR transimitter?
3. Why is IR signal modulation necessary?
4. If the IR transmitter sends a high signal, does the receiver output a high signal?

Protocol

There are many different IR protocols. If you have a *universal* remote, you can set the protocol that it uses, but for your every-day garden variety remote, you must discover the protocol. One approach to "discovering" the protocol is to lookup your remote in the Linux Infrared Remote Control (LIRC) repository. Another approach is to decode the protocol using the IRremote library discussed in the section below. Using the second approach, the remote that you were given was found to use the NEC protocol. Check out the details of the NEC protocol and be prepared to answer the questions below.

1. What is the modulation frequency of the IR carrier signal?
2. In the original (non-extended) NEC protocol, how many bits are sent in the transmission generated for one key press?
3. Which is transmitted first, the least significant bit or the most significant bit?
4. In NEC's version of *pulse distance modulation*, what is the duration of a logical one? What is the duration of a logical zero?

Receiving Signals

We are going to use a library developed by Ken Shirriff to make it very easy to send and receive IR signals using a wide array of protocols---thanks Ken! Begin by downloading the IRremote library from Ken's page. As instructed on the page, unzip the library and install it in `arduino/hardware/libraries`.

In the Arduino IDE, open the program `IRrecvDemo` from the `examples` folder in the IRremote library installed above. Before running the program, connect your IR detector to the Arduino as described in the Hardware setup section of **Ken's page**.

Run the program and test it using the your remote. Be sure to point the plastic transmission window on the front of the remote directly at your IR detector as you depress the keys of the remote. Remember, IR is line-of-sight only. Note that each key on the remote sends a unique message. Referring back to the description of the protocol, how many Hex digits should be sent in each transmission---does that match what you see? Does the *address* or the *command* portion of the transmission change with each key that you depress?

Change It
As your first challenge, modify the code in `IRrecvDemo` to discover the protocol used by your remote. Use the available resources to figure out how to do that.

Change It Again
Now try something more fun; use your remote to control your Arduino. As a simple demonstration of what could be done, configure your Arduino to control a LED; do not remove the connection to your IR detector. Select 3 keys on the remote and modify `IRrecvDemo` so that depressing each key causes the LED to blink a unique number of times. For example, depressing key 1 would cause the LED to blink once, key 2 would cause 2 blinks, and key 3 would cause 3 blinks.

As you might imagine, you could use the IR remote to control a large number of arbitrarily complex Arduino functions.

Sending Signals

You can also send IR signals using the IRremote library. Begin by preparing your IR LED by shielding it with electrical tape before placing it on your breadboard. The shielding will greatly reduce the spread of your IR beam allowing better directional control. Connect your shielded IR LED to the Arduino as described in the Hardware setup section of **Ken's page**. Load the `IRsendDemo` program from the `examples` folder in the IRremote library to your Arduino. Use your instructor's Arduino and IR detector to test your `IRsendDemo` program. In other words, send IR signals to the instructor's IR detector and verify that they are received.

Change It
The next step might be to build an "IR walkie-talkie" that both sends and receives IR signals. But, unfortunately, the library does not support simultaneous sending and receiving of codes; transmitting disables receiving. So, we'll just play with sending as our final exercise.

Add three pushbuttons to your breadboard (leave the IR LED in place) and configure your system so that each pushbutton causes a different IR signal to be sent. (If you need help connecting the pushbuttons, refer to LEDs and Sensors Part 1 for a schematic.) Use

at least one protocol other than Sony. Use your instructor's Arduino and IR detector to test your new program.