CIT 595 Spring 2016
Arduino Tutorial

Introduction

In this assignment, you will use the serial communication libraries to set up communication between an Arduino device and a Linux application written in C.

You may work in groups of **three**. Please note that all students need to submit their code at the end of the recitation session, as described below.

Before you begin

If you are working on one of the Linux machines in Moore 207 or Moore 100A, begin by following these steps:

1. Attach your Arduino board to the machine via the USB cable.
2. Run the command `arduino` from a terminal in Linux. This will start the Arduino development environment.
3. The first time you start Arduino, it may ask you to create a directory for your "sketches" (Arduino programs). Specify a directory (create a new one if need be) and then click "OK".
4. Once the IDE opens, go to "Tools" on the menu bar, then select "Board" then select the board you are using (which will probably be "Arduino Duemilanove w/ATmega328" or "Arduino Uno"); ask the instructor if you are unsure.
5. Then from "Tools" on the menu bar, select "Serial Port" then "/dev/ttyUSB10" (or something similar).
6. Now you will open up one of the sample programs. On the menu bar, select "File", then "Examples", then "Basic", then "Blink". This is a simple Arduino program that just makes one of the lights blink (hence the name!).
7. In the new window that opens up with the "Blink" program, click the "Verify" button on the top left (the checkmark icon). This will compile the program for your specified platform. You should see the status message "Compiling sketch" and then "Done compiling" on the bottom of the IDE.
8. Last, click the "Upload" button (arrow pointing right). You should see "Uploading" and then after a few seconds, you should see the RX (receive) and TX (transmit) lights on the board flashing. Then the status bar should read "Done uploading" and then the tiny little orange "L" light on the board will blink on and off once per second.

Note: If you're using your own machine, follow the "Getting Started with Arduino" tutorial for Windows (http://arduino.cc/en/Guide/Windows) or Mac OS X (http://arduino.cc/en/Guide/MacOSX) and be sure that you are able to get the "blink" program running on the board.

Part 1

First, you will create a simple Arduino program that writes data to the serial port. The following program reports the number of milliseconds that it has been running by sending the value to the serial port once every second:

```
unsigned long time;

// This method is called once when the Arduino reboots
void setup(){
  // sets the transfer rate to 9600 baud
  Serial.begin(9600);
}

// This method is called repeatedly
void loop(){
  Serial.print("Time: ");
  time = millis(); // gets the time since program started
  Serial.println(time);
  // wait a second so as not to send massive amounts of data
  delay(1000);
}
```

Using the Arduino development environment, create a new sketch, copy & paste this code and then compile this program and load it onto the board.

Then choose "Tools" from the menu bar and then "Serial Monitor". After a few seconds you should see the "time" output coming from the board.

At this point, your Arduino board is ready to start communicating with the outside world!

Part 2

Write a C program that gets the input coming from the Arduino board by reading from the serial port. Find which serial port it is using by looking in the /dev directory for a recently created file that will probably be named something like ttyUSB10.

Make sure that you include all of these headers when writing your C program:

```
#include <sys/types.h>
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <termios.h>
#include <unistd.h>
```

You will also need configure the port for the appropriate baud rate using the following (after calling “open” to get the file descriptor fd):

```
struct termios options;        // struct to hold options
tcgetattr(fd, &options);       // associate with this fd
cfsetispeed(&options, 9600);   // set input baud rate
cfsetospeed(&options, 9600);   // set output baud rate
tcsetattr(fd, TCSANOW, &options);  // set options
```

Your program should take the input coming from the board and write it to the screen using printf, so that you see the increasing values of time appearing once per second, e.g. "Time: 0", then "Time: 1000", and so on, just as in the Serial Monitor.

Be sure that you store the <u>entire</u> incoming message in a global string in your C program! That is, do not just print one character at a time as it arrives. Although this is not strictly necessary just to mimic the Serial Monitor functionality, you will need to be able to do this for your group project. So you might as well do it now!

This seems really simple, but it can be very tricky!! A few things to keep in mind:

- Do not assume that the "read" command will read the entire message coming from the Arduino as one string; it may be read in small parts. Remember that the "read" system call returns the number of bytes read.
- Also, the "read" system call will return 0 if there is nothing to read, i.e. it will not (necessarily) block and wait for data.
- Because the Arduino program is using the "println" command to send the message, it will end with a newline character, i.e. '\n'. That is how you will know you've received the end of the message.
- Do not attempt to synchronize your C program and the Arduino by having the C program sleep for one second and then do the read. Just have it continuously looping and reading the bytes as they become available.
- Make sure you have closed the Serial Monitor in the Arduino IDE while attempting to run your program; you can't have two programs reading from the same input simultaneously.
- When you start your program and open the serial connection, this restarts the Arduino program, so you should theoretically see it start with "Time: 0" every time. However, you may see some "garbage" appear before "Time: 0" because of buffered data. It is okay if your program prints this out first.

<u>What to submit</u>

When you are done with this assignment, show it to a member of the instruction staff that they can verify that it works correctly. Then **each** member of your group should submit the code in the “Recitation #4” assignment in Canvas before 12pm noon today.