Computer System Design Lab # 1

Amy Guo
Lab Partner: Charlie Coleman

January 22, 2018

Pre-Lab:
N/A

Objective:
The objective of this lab was to use a PC to output and read serial data. We were to investigate different properties of serial communications, such as baud rate, parity, start & stop bits, data bits, ASCII, etc.

Circuit Diagram:
N/A

Outcome Predictions:
N/A

Equipment:
Oscilloscope, 9 pin RS232, RayTac MDBT40 Blue Tooth Low Energy radio, FTDI Friend and a Type-A to Mini-B USB cable.

Procedure:

1. We examined the 9 pin RS232 by using the oscilloscope to determine which pin corresponded to each function.
2. Using PuTTY, we created a continuous RS232 bit stream out of a PC serial port at the given baud rate: 9600 N81. We used the oscilloscope to collect the data from our bit stream.
3. We changed the baud rate and parity of our bit stream and analyzed the new traces of our characters.
4. To examine the TTL RS232, we went through the BlueFruit configuration. We used the oscilloscope to collect the data from the response.

Recalculations and Predictions:
N/A

Data and Observations:
While testing the 9 pin RS232 line, we examined the waveform output of various ASCII characters using an oscillascope. We attempted to record the waveforms output for the experiment, but the files were saved in an incorrect file format, so we are unable to insert waveforms in this lab document. We were able to identify many different ASCII characters, e.g. 'A'.
Initially, it was difficult to decode the letter waveforms as we did not take into account that the data was transmitted LSB first. Once this was realized, it was fairly easy to identify the letters correctly.

Analysis:
N/A

Discussion:
We were able to classify each pin of the 9 channel RS232 correctly. We were able to configure the BlueFruit to operate. On the oscilloscope, we observed the correct waveforms from the bit stream and the BlueFruit response.
At first the waveforms confused me but after analyzing, I realized start/stop bits were being outputted as well as the character. Not only were other bits were thrown in to confuse me, the data was sent out backwards. The data displayed from the least significant bit to the most significant bit. Overall this lab was successful.
Results:

Q: What baud rates is your terminal program capable of?

A: PuTTY is capable of any baud rates.

Q: What start/stop configurations is your terminal program capable of?

A: PuTTY only allows you to change the number of stop bits expected, and you can set this value to any number.

Q: What parity configuration is your terminal program capable of?

A: PuTTY includes no parity, even parity and odd parity.

Q: What word lengths is your terminal program capable of?

A: The number of data bits can be set to any value. The default is 8.

Q: From a serial bit stream, how would you determine the bit stream that is being transmitted?

A: It would be very difficult without some knowledge about what is being transmitted. You could look for square wave portion of the signal and measure the period.

Q: If a stream has a configure of 19200 N81, what is the maximum throughput in characters per second?

A: Each character is sent with 1 start bit and 1 stop bit. This means that each character (8 bits) requires 10 bits of data sent. The baud rate, or bits/second of the stream is 19200. This means that $\text{CPS(char./sec)} = 19200\frac{\text{bits}}{\text{sec}} \div 10\frac{\text{bits}}{\text{sec}} = 1920\frac{\text{char.}}{\text{sec}}$

Q: What is the throughput of your terminal program at the baud rates used to talk to the BlueFruit?

A: The BlueFruit ran at a baud rate of 9600 N81. Using the same formula as the previous question, we get a throughput of 960 characters/sec.

Conclusions:

This lab taught me how serial communication happens. I also learned how to read the waveforms that were outputted on the oscilloscope. The BlueFruit Configuration portion taught me how to set up and connect a bluetooth module. Hopefully this semester these skills will come in handy. Charlie and I joined the AUVSI team, so we will have to establish a way for the drone's computer to communicate with the ground server. Maybe we can use serial communication to solve this problem.