CG1112 Engineering Principles and Practices II for CEG
Week 8 Tutorial

(Part 1)

1. In this question we explore the UART link-layer protocol:

   a. Show how odd and even parity bits works, and how they can be used to detect errors.

   b. Explain why error detection with parity bits is not reliable, and what engineers can do to overcome/improve this lack of reliability.

   c. Sketch the frames for the following pieces of 8-bit data in 8N1, 8E1, 7O1 and 7E1 frame formats. Where you are using 7-bit frames, discard the most significant (left most) bit.

      0b10110001
      0b01010011

2. We extend the idea of parity so that we set parities not just within a byte, but across bytes, effectively producing a parity byte. The table below shows this idea, using odd parity (0 is considered to have an even number of '1' bits):

Data bits ->	b7	b6	b5	b4	b3	b2	b1	b0	Parity (within byte)
Byte 1	1	0	1	1	0	1	0	0	**1**
Byte 2	0	1	0	0	0	1	0	1	**0**
Byte 3	0	1	1	1	0	0	0	0	**0**
Parity (across bytes)	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**

   This scheme can be used to correct single bit errors. Show how.

3. Our lectures do not look at flow control. Explain how XON and XOFF, and CTS/RTS flow control work and the differences between them.

4. How long does it take to transmit all 128 bytes in a buffer at 9600 bps? What implications does this have on how fast your program can write to your buffers?

5. Explain the difference between bit rate and baud rate. Explain why bit rate can sometimes be much higher than baud rate.

6. The serializing code provided to you uses a magic number of 0xFCFDFEFF. What is the purpose of the magic number? Give two more examples (not necessarily relating to communications or serializing) where magic numbers are used.

7. In this question we will explore the idea of checksums.

   a. Derive the checksum for the following sequence of bytes:

      0A 1C 42 3A

   b. Explain how to use this checksum to check for errors.
   c. The sequence in part a. was sent out by the transmitter but the receiver instead received:

      09 1C 41 3A

      Derive the checksum for this new sequence.

   d. From your answer in c., what is the main weakness of checksums?

(Part 2)

1. In the last few weeks, we learned quite a bit of serial communication and communication protocol. For this question, we are going to look at another source to reinforce our understanding. As you know (hopefully), the RPLidar unit uses serial communication too! Let us "dig around" in the source code to learn more.

   Please refer to the sdk source code given in week 8 studio 1, i.e. rplidar_sdk_v1.5.7.zip. Pay attention to the following two subfolders once you unzipped it:

   * sdk/sdk/include : Important defines and data type declarations

   * sdk/sdk/src: Implementation of the rplider driver

   Although rplidar driver is written in C++, the code is still largely understandable to a C programmer. Try to glean the key logical steps instead of worrying too much about unfamiliar syntax.

   Answer the following questions:

a. Describe the steps required to get the device information from the rplidar unit. Focus on the message format and meaning of the fields of the messages exchanged.
b. Similarly, describe the steps required to get the health information from the rplidar unit.

c. [Optional – not discussed] Find out how the scan data are retrieved from the rplidar unit.

Hints:

* Start from src/rplidar_driver.cpp, look for the relevant "functions", e.g. getDeviceInfo() and getHealth()

* Find out the relevant definitions from include/rplidar_protocol.h, inlucde/rplidar_cmd.h and other header files in that folder.