Final Project Proposal

1. Project Functionality:

The aim of the project is to implement compression to speed debug prints over UART. Prior to transmitting a particular string over the serial line, the KL25Z will employ Huffman coding to compress debug messages prior to transmitting them over the UART.

On the PC side, the complementary code is used to decompress and display the debug prints.

- Huffman Coding: Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code.

  The variable-length codes assigned to input characters are Prefix Codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.

  The project flow is as follows,

1. Data from many sources is compiled into a corpus (covering the whole range of ASCII characters). This information is then used to calculate the likelihood of each symbol appearing (probability of occurrence).

2. A C header file containing the Huffman code for each symbol is constructed using the above-mentioned probability table.

3. The KL25Z serves as the transmitter for this project, which means it performs the encoding. The PC serves as a receiver, decoding the message received.

4. The transmitter and receiver each have their own main () function, as they must be run on the KL25Z and the PC separately. The cross-compiler compiles the software that will be run on the KL25Z, while the native compiler compiles the program that will be run on the PC.

5. The debug strings that are to be sent to the PC are first encoded by the KL25Z before being transmitted via UART. So, when the printf () statement is executed, it calls the __sys write () function. As a result, the system can also handle dynamic strings.

6. Within the __sys_write () function, the byte stream is encoded using Huffman Encoding and then enqueued onto the transmit buffer.

7. The main () function on the PC side takes a command line argument, the COM port on which the KL25Z is connected to. For instance, “/dev/ttyACM0”. Then, the decode function basically does a file open on "/dev/ttyACM0" and checks if any data exists in the buffer (file). If so, the decoding process is initiated.

8. At the PC side, once the encoded stream of bits has been received. The Huffman table is made use in order to decode.

9. In addition to all of the above: the PC side program keeps a hold of the following statistics, how many bytes were received over the UART, and how many bytes were emitted from the Huffman decode function. After the user terminates the program by pressing ctrl + c, the overall compression achieved is printed on screen.

2. Technologies Used:

- Circular Buffers
- UART
- Data Compression
- Systick

3. Reference Sources:

1. Huffman coding using C Programming: https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

   https://www.programminglogic.com/implementing-huffman-coding-in-c/

2. Access Serial Ports in Unix based OS: https://unix.stackexchange.com/questions/138342/how-to-read-write-to-tty-device

3. Signal handlers in C: https://www.geeksforgeeks.org/signals-c-language/

4. Additional Hardware:

- For this project, no additional hardware is necessary.

5. Testing Strategy:

- As described below, fully automated testing is used for this project.

- Separate from the main functionality which performs decoding of the received bit stream, within the PC code a test mode is incorporated which tests the functionality of encode and decode functions on a set of hardcoded strings.

- When this test mode is active, the test function is called every time the program is launched. For each string, it calls Huffman_encode and then Huffman decode. The resulting decoded output is then compared against the original string, and then asserts if they don't match.