listen_new.py Documentation

Important Notes:

Channel List: [26, 18, 24, 16, 22, 14, 20, 12]
Transmitter ID Numbering: Starts with 1 and ends at Total Number of Transmitters
Counter: Resets when it reaches 65,536 (16-bit unsigned int)
RSS Values: 8-bit signed int. Max = 127. Any values where nothing was read has 127 inserted

Input:

The input to the code is the raw output of the listener node in the radio sensing network. This is a stream of hexadecimal numbers. The program uses the indicator ‘beef’ to distinguish between new lines and wrap the string preceding ‘beef’ into a line to import line by line from the listener node. Below is a graphic of the location of relevant data points within the hexadecimal line:

Index	**1**	**2**	**3**	**4 to (#Tx+3)**	**#Tx+4**	**…**	**-2:end**
Data Stored	Counter (1st Half)	Counter (2nd Half)	Transmitter ID	RSS Values Stored	Channel ID	?	‘ef’,’be’

There is extra information stored in the hexadecimal line such as battery information of the transmitter which appears after the channel ID, however as it is irrelevant to the structure of the data packet, the program will omit looking at these numbers for the program.

Output:

The output of the listen_new.py program will be a buffered data stream which take in the last X lines of data streamed from the listener node (X being the total # of transmitters used), and outputs lines of data with the RSS values received from all other transmitters, along with the transmitter ID that sent those signals out, the channel it was sent on, and the Linux timestamp associated with that signal transmission. These values will be space-separated and put onto the same line. Below is the structure of this new data packet as it will be outputted onto the stream from listen_new.py:

Index	**1**	**2**	**3 to (#Tx+2)**	**#Tx+3**
Data Stored	Channel ID	Transmitter ID	RSS Values of Transmission	Timestamp

It is important to understand that the RSS Values being returned are not the values stored in that transmitter, but the RSS values from the other X-1 transmitters that were stored from the signal sent by the transmitter whose ID is listed (X being the total # of transmitters used).

Program Function:

The main function of this program is taking in lines with the RSS values stored by each node and converting them into lines that print out the RSS values of the signals sent out by the transmitter with the timestamp and counter number most accurate to the timing that the signal was sent out/received by the nodes. Because we are printing the RSS values stored in other nodes in order to compile all the values sent out in the same line, we must incorporate a buffered data stream which is buffered by the amount of lines equivalent to the number of transmitters used.

Below is an illustration to help visualize how exactly we are changing the RSS values being taken in and printed out (format [sending node] → [receiving node]):

RSS Values taken in:

TX ID	RSS_1	RSS_2	RSS_3	RSS_4	RSS_5	RSS_6	RSS_7	RSS_8	RSS_9	RSS_{10}
1	1→1	2→1	3→1	4→1	5→1	6→1	7→1	8→1	9→1	10→1

RSS Values outputted:

TX ID	RSS_1	RSS_2	RSS_3	RSS_4	RSS_5	RSS_6	RSS_7	RSS_8	RSS_9	RSS_{10}
1	1→1	1→2	1→3	1→4	1→5	1→6	1→7	1→8	1→9	1→10

In addition to reformatting the RSS values associated with each transmitter ID number, the program also makes sure that the counter and timestamp associated with the timepoint that the signal was sent out is printed along with the other information

Warnings:

Buffered Stream: Because the first 10 lines must be read before anything may be printed by the program, there is a 10-line delay between the data coming in and the output of the data stream, The values that will be printed in the first 10 lines of the output data stream will contain values of 0 for the RSS values on lines where all the data stored in the receivers were not of the new run and when the program is turned off, the last 9 lines of data will not ever be outputted to any extent due to the stream buffer.

Data Packet Loss: Sometimes a packet of data from the listener node may be lost, which results in line of the hexadecimal input not appearing as expected. The listen_new.py program considers the possibility of data packet loss and will insert values of '127' for any RSS values associated with that input line. Additionally, the transmitter ID and the channel ID will be filled in based off an interpolation of the lost packets between the last packet received and the current packet received. The timestamp unfortunately will be based when the lost packet is noticed and will not be the exact timing when the packet 'should've' been read.

New Data Stream: When starting a new data stream the program will recognize the transmitter and channel that the data stream starts on and will start printing data lines from that transmitter ID onwards, looping as normal. The starting transmitted ID will not always be 1 and the channel may be halfway run through when the program is first starting up. Any lines printed before 10 packets of data have been received will be output with values of 0 in some of the RSS values.