SNR-to-Error Evaluation of M-Ary PSK/QAM: Applied to Affordable Consumer SDRs

Ian Dwyer and Esraa Aldhufairi

Abstract

The goal of this article and testing is to verify the functionality of consumer available products, HackRF One SDR and GNURadio GUI, in comparison to theoretical values SNR values for simple uncoded QAM and PSK digital constellations and understand the value of gains such as coding gain, oversampling/processing gain, and constellation packing losses to improve SNR requirements. Text shows approximately 3dB more SNR is required for a Nyquist two samples per symbol for each increased order but interpolation from higher samples per symbol is suspected to provide some coding gain with unfortunately large processing overhead and is therefore a degree of freedom to investigate due to ease of implementation. GNU Radio, HackRF, antenna, a performance design workstation laptop and a bandpass filter were the essential testing equipment for the entire article. Text shows that 13dB SNR is required for M=2 and approximately M=4 at two samples per symbol for basic digital constellations to obtain a SER of 10^{-6} which is the lower end of media streaming expectations. The text and calculations also provides that there is 3dB SNR required for each harmonic higher order of M to ensure this error rate. Testing found that the OTA system required 2dB more than predicted at 5dB gain in SNR required to maintain SER of 10^{-6} and every doubling of samples per symbol approximately provided a 3dB SNR gain equivalent. Findings also confirmed that the minimum SNR was about 13dB at two samples per symbol for a BPSK, but also found that oversampling gain from sixteen samples per symbol reduced this minimum SNR requirement to 4dB for the BPSK at the expense of eight times the processing for the same data rate of two samples per symbol which is notably inefficient but effective. Results appeared to converge with expected values as samples per symbol increased. This study has provided insight to the functionality of consumer available SDRs being functional for custom systems where SNR has provided a close relationship to interpolation methods such as higher order constellations and higher samples per symbol but requires efficient designs to utilize the potential of these devices and that affordable consumer SDR devices will likely not have the efficiency to compete with specialized radio systems such as Wi-Fi modems that require 150MHz bandwidth with multiple access at high data rates.

I. Introduction

Software defined radio (SDR) devices have become more popular and despite massive price hikes up to 200% MSRP, from demand and shortages, modules such as the receiver modules like the RTL-SDR, half-duplex transceivers like the HackRF One and full-duplex high-quality systems

like USRP B200 are readily available to hobbyists, researchers, and developers alike. Unfortunately, dynamic functionality comes at a price and without deep pockets quality takes a backseat [5], so how well transceivers transmit and detect signals is a considerably practical application that should likely be done to characterize the functionality of any new device used in development.

One benefit to high-ordered constellations is the ability to fit more data in a single symbol, where data rates are costly due to limitations set by the FCC for an allotted band this provides the same bandwidth but with higher data density. PSK modulations are relatively easy to lock onto but higher densities on the same radian drastically reduces the minimum distance between symbols causing lower stability. QAM constellations offer higher density and more effectively utilize the constellation area allowing for nearest neighbor approximations to even be more accurate, which is the most common detector.

The most affordable platform for developers and hobbyists is GNURadio with HackRF One or the PlutoSDR. The HackRF has no filtering to ensure that the signal only operates in the intended band, so another expense is the bandpass filter to remove upper harmonics. An antenna and possibly an amplifier is needed, but other than these expenses GNURadio is free and open-source so unlike MATLAB no licensing or subscription is needed. The HackRF One by Michael Ossmann itself is open-source software and hardware, so modifications are welcome and there is plenty of documentation available. Using an unofficial clone HackRF, about a third of the cost of an official HackRF One, with the other equipment mentioned allows a hobbyist or developer to do testing for around 200 USD. Other than some inefficiencies in GNURadio such as dynamic blocks, blocks handled in Python, and only moderate, infrequent official support of the interface GNURadio Companion, it appears to be a helpful tool in quickly developing, simulating and testing radio systems. Tutorials on using GNU Radio are available through Wikipedia [4].

This study will provide many of the specification limitations and quantify functionality of the HackRF One using GNURadio. The goal is to test various constellations to observe at what SNR levels the HackRF One can still recover messages at a symbol error rate of 10^{-6} for sample per symbol interpolations of two, four, eight and sixteen in addition to testing the hypothesized SNR values in hopes to understand how this coding gain improves detection. It is proposed that the generally termed coding gain, while technically oversampling or processing gain, from increased doubled samples per symbol where SNR is proportional to the gain in samples per symbol, providing about a 4.76dB decrease in required SNR considering the process gain and bit density increase due to increasing the Nyquist zone [11] and [12]. These results are compared to text references for theoretical levels and empirical data from other research to supplement the findings. Other results of interest, with consideration to the qualitative review of affordable SDR devices, are the maximum sample rate achievable, the highest constellation with stable results, the maximum data rate achievable over-the-air (OTA) and the highest data rate achievable by the device connected to the radio.

II. Related Work

Given text reference from [1], using their equation for SER estimate with respect to SNR on an AWGN channel. For QAM the expected SNR required to achieve a specific SER can be calculated,

$$SNR = 10\log_{10}\left(\left(\frac{M-1}{3}\right)Q^{-1}\left(.\frac{Pe}{2\left(1-\frac{1}{sqrt(M)}\right)}\right)^{2}\right) \quad (1)$$

For MPSK,

$$SNR \approx 10\log_{10}\left(\left(\frac{Q^{-1}\left(\frac{Pe}{M-1}\right)}{sin\left(\frac{\pi}{M}\right)}\right)^{2}\right) \quad (2)$$

To account for SPS oversampling the commonly accepted SNR increase from oversampling using both Analog Devices [11] and Texas Instruments [12] it is proposed to apply a scaling of –(1.76-10*log10(D)) where D is the decimation being replaced by SPS/2 as the oversample factor where for QAM given that the results match [1] at samples per symbol without and RRC filter, so the modified equation used for QAM is,

$$SNR \approx 10\log_{10}\left(\left(\frac{M-1}{3}\right)Q^{-1}\left(.\frac{Pe}{2\left(1-\frac{1}{sqrt(M)}\right)}\right)^{2}\right) - 10\log_{10}\left(\frac{SPS}{4}\right) \quad (3)$$

Liberties were taken as the BPSK did not appear to follow this assertion and modifying for the binary constellation provided only 10log10(SPS) scaling to match results where this discrepancy should be investigated in later works.

Another work to mention is how the SNR is commonly estimated such as a Pilot-Based Time Domain SNR Estimation for Broadcasting OFDM Systems as (the most common method for low-overhead active SNR detection uses comb of uncorrelated (Zadoff-Chu sequence) pilot signal to minimize overhead and incorporate unique tone-based signals to ensure estimate of the local channel [8]. They propose a simple cyclical detector that uses the time domain power divided by the measured variance

Most of the challenges reported by other research were for achieving reasonable overhead, minimizing computation time and ensuring that the signals are uncorrelated so that the use of a unique pilot sequence for autocorrelation would be able to only detect SNR for the intended

channels signal. S. Manzoor et al. proposed that most SNR detectors do not consider ISI in their calculations [9] proposing front end. Others reported challenges in proper SNR detection such as A. Kahn et al. place emphasis on computational requirements [8] and overhead of the preamble used for detection. Unfortunately, one note of consideration, most of the research found did not have OTA testing where tested channels were not ideal channels but strictly using simulation brings practicality into question without further testing due to effectively timing the data where BER is measurable on an OTA channel that is not directly connected in the software. There are practical limitations on obtaining accurate BER estimates due to stream synchronization, lacking channel state information, repeatability and requirement of hardware intensive testing equipment such as anechoic/reverberation chambers for accurate results [7].

III. System Design and Experimental Setup

The transmitter is designed starting with the source, either an audio source or UDP source where the UDP source uses a local internal socket, not a Wi-Fi socket, to obtain byte messages from an external Python application. Ideally, the radio only takes in data at a specific rate, or it dynamically rescales the input rate from a UDP port to run on the radio, but there was not time to develop this. The signal is then packed as a payload in a packet with a checksum and access code header for reliable data recovery. The packet is then fed to the constellation modulator block which sparsely repacks the bits as bytes with b bits per symbol, it then zero-fills to interpolate, then applies a root raised cosine (RRC) filter for signal shaping. Normally this would be fed to the radio sink, but an RMS auto gain control (AGC) block soft clips the signal to minimize spectral regrowth from high PAPR at the ADC constraints [13] and the nonlinearity of the port's hardware. It should be noted that PAPR can be reduced by using less filter taps in the modulator, but that option is not available for the in-tree block available. AGC alpha step sizes below 0.001 were fast and minimized regrowth when transmit gain was scaled by 0.5. The maximum realizable signal from the 8-bit 15dBm (-15dB) max output is 49dB, this is the theoretical maximum of the ADC, but the reliable maximum has been 38dB with a true max without spectral regrowth at 45dB being occasionally achievable. The input is highly sensitive and can only receive -10dBm (-40dB) before high change of destroying the front-end circuitry.

The channel for testing is simple with the client in line-of-sight (LOS) to the base station. The tests were primarily performed in a 12'x12'x7' drywall room with wood framing and a single window so there is likely multipathing but no distinct fading. The transmitter and receiver were placed 6' apart for testing with gain at maximum on the transmitter and no external amplifier applied.

The receiver matches the transmitter except for the recovery blocks. It starts with an RMS AGC block to lock the gain, the SNR is probed at this output, it then uses a Zero-Crossing timing recovery with a maximum likelihood polyphase filter that utilizes the matching RRC taps [3] to

then allow for decimation of the signal. This is then fed to the phase recovery which also contains the symbol detector. The phase bandwidth edged were set large for connectivity, so phase deviation per sample was set to $\pi/2$ as $\pm\pi/4$ of the sample rate bandwidth. This is then checked for the access code and checksum, byte-aligned, then sent to the sink, either console print or the audio sink. The phase bandwidth itself was set for a relatively small size of 0.001 which is on the lower end of stable for connectivity where the opposite was done for the timing bandwidth using a larger bandwidth for the lock.

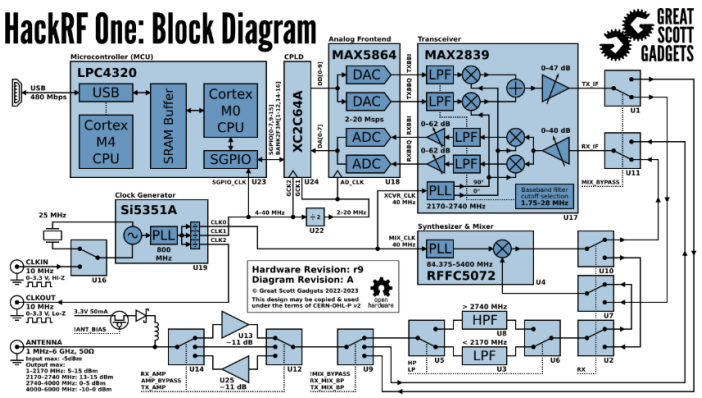

Figure 1: HackRF One Block Diagram

	P_{txmax}	P_{rxmax}	R_S	*linearity*	*ADC*	*TX Gain*	*RX Gain*
HackRF	15dBm	-10dBm	1-20MHz	low	8-bit	NA	NA
BPF	0dBm	0dBm	Multistage	high	NA	0dB	0dB
LNA	49.5dBm	0dBm	0.02-4GHz	high	NA	19.5dB	-10dB
Antenna	NA	NA	9.00MHz-9.24MHz	NA	NA	5.5dBi	5.5dB

Table 1: Hardware specifications of interest

The hardware used has unique nuances and basic parameters should be observed to ensure that the design was optimal for testing. One of the most important being the SDR device itself, so the hardware schematic in figure 1 and essential specifications in table 1 are essential in understanding the limitations of the equipment such as the ADC noise, max realizable PAPR/SNR, and sample rate limitations. As for the specifications of other essential equipment, antenna gain is 5.5dB, center of 912MHz, return loss of -18dB, and characteristic impedance of 40Ω (1.25 VSWR for 50Ω transmission line matching). The computer used for DSP is a 6-core 12-thread 2.6GHz Intel i7 and 32GB DDR5 with volk and multithreading enabled. An 8GB DDR5 Raspberry Pi4 and a 2019 MacBook Pro does not handle the program without massive distortion but a 2023 10-core 3GHz MacBook Pro powered by M1 with 64GB DDR6 appeared to operate GNURadio well with tuning but choppy as if sample rate was too low to manage. Speedups would be considerable and GNURadio would be manageable if only C++ compatible blocks were used and a workaround was used to operate bitwise before modulation, this alone is calculated to improve performance by a factor of 2 for the 16QAM, 4 for the QPSK and 8 for a BPSK.

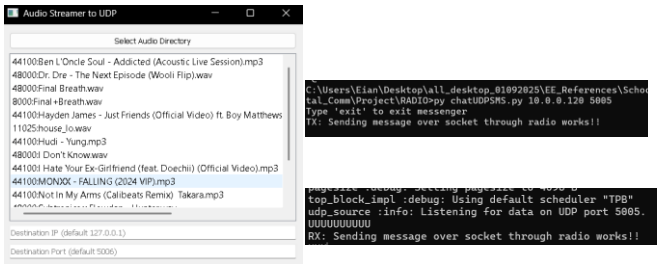

Figure 2: External Python scripts for IO

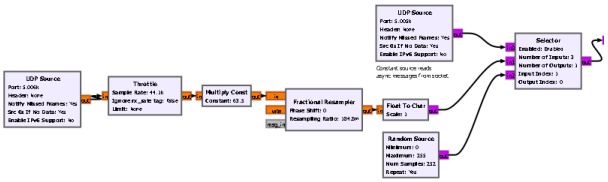

Figure 3: Input MAC layer simulation for messaging, audio streaming and random inputs to radio system

The stream starts with a UDP port that is fed into the flowgraph to generalize inputs. It also multiplexes other options such as random inputs and SMS inputs. This would normally be handled by a well-designed MAC layer with detailed header design to instruct packet management at the receiving MAC layer multiplexing all ports read by the radio, but this design simplified testing. The UDP ports both use an external Python script to either send messages using strict packet sizes that are zero-padded for stability and management and the other port uses an FFMPEG streamer to send music to the UDP socket in real time with settable port, IP and audio rate. Examples of the input interfaces are provided in figure 2.

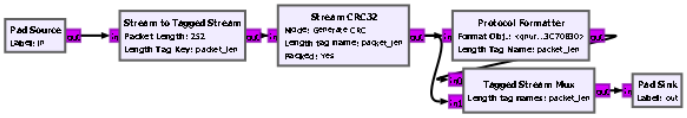

Figure 4: Packet assembly flow blocks

The packets are designed using a CRC32 header to provide a checksum for packet confirmation to ensure only error-free packets are received. As seen in the flowgraph section in figure 4 the header is generated separately and multiplexed into the stream. Careful consideration with fractional resampling is essential for ensuring device rates match to provide clean playback. Figure 5 shows that the constellation is then modulated using the in-tree module Constellation receiver which applies a zero-padded interpolation of samples per symbol (SPS) and pulse shapes using an RRC filter of a specified roll-off factor and options to truncate transient components and differential encoding. It was found that differential encoding significantly improved reliability. The signal then is clipped to remove spectral regrowth and fed into the radio block which required careful use of the gain values for a relatively stable signal.

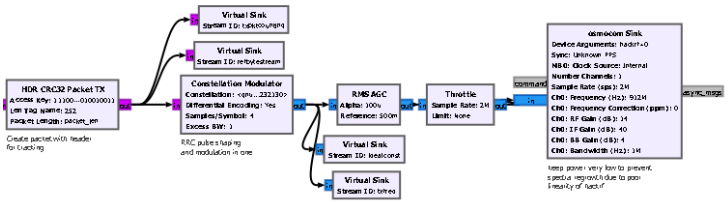

Figure 5: Signal modulation and pulse shaping blocks with soft clipper

The receiver required a shift with a DC block to remove constant noise from the DC hardware components and an interface on the GUI for center frequency ppm adjustment. Figure 7shows this is then fed into a custom demodulation block of the gain, timing, frequency, equalization and phase recovery to then provide differential decoding and repacking of the bits to recover the packed byte structure. After recovery, the flowgraph section in figure 8 depicts the CRC32 checker that parses the header, checks the access key, tags the packet's starting point with a length tag and sends it to an align stream block for relative synchronization using the appended tag. Synchronization was one of the most difficult tasks, and methods for this heavily outlined in [1] and [2]. The MAC layer equivalent in figure 9 is hardcoded for the same options as input except the audio and messages are directly decoded and played back in GNU Radio's GUI. The most important block is the SNR detector in figure 10 depicting the input parameters of the SNR detector and the direct coding application of SNR equation, equation 4.

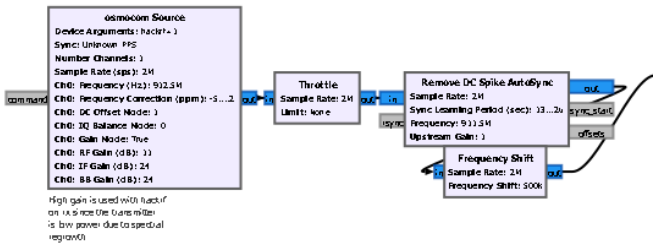

Figure 6: Receiver blocks for raw RF signal

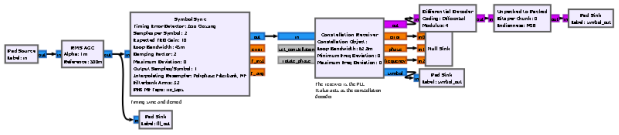

Figure 7: Synchronization and recovery for the received signal

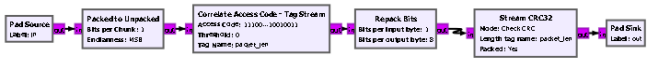

Figure 8: Packet preamble verification and payload extraction

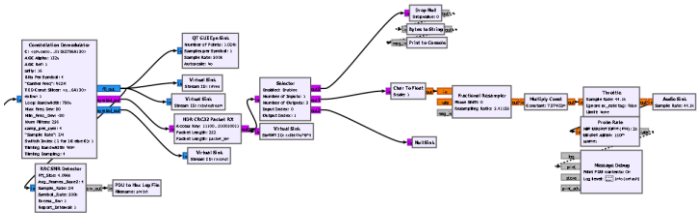

Figure 9: Simulated MAC output layer

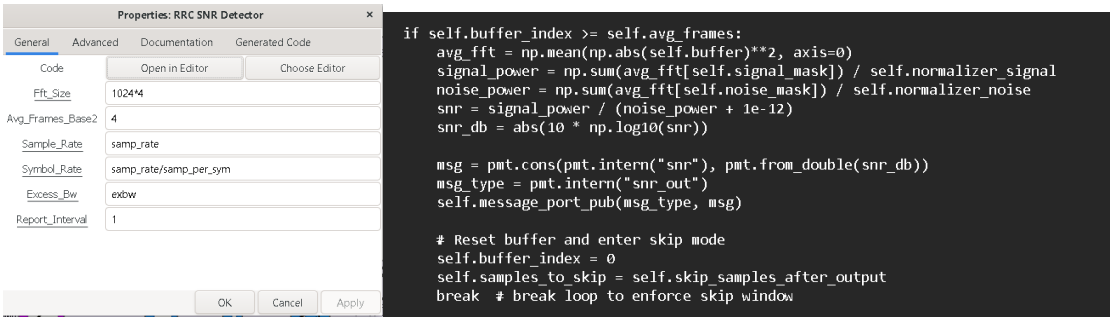

Figure 10: SNR detector interface in GNU Radio and the essential code block used to calculate SNR

Practical sample rates achievable with the HackRF One in GNU Radio are between 2MHz and 8MHz given that 8MHz is at the theoretical limit of USB2.0 considering 8×10^6 samples/sec×2, bytes/sample×8, bits/byte=128×10^6, therefore bits/sec=128Mbps. In testing, 4MHz is achievable, especially with higher samples per symbol to offset this bandwidth. The test program operates from 1.25MHz to 5MHz. The maximum bandwidth achieved was 4MHz with large error, operating bandwidth was 1MHz with an excess bandwidth of 1 at 4 samples per symbol, to allow room to shift DC signal, with the packet design and sparce byte representations of the symbols the max BPSK rate of data from the source is 60KSps (bytes in this case, so 477Kbps) with max 16QAM data rate of 239KSps (1.9Mbps) and which allows for audio (48KHz or 384Kbps for mono) when resampled properly. The max device rate achieved was 477KSps (3.8Mbps) with 16QAM at 4MHz and 4 samples per symbol. This provides that despite software limitations, the system is acceptable for testing purposes using the HackRF One on GNU Radio.

The constellations under investigation are a BPSK, QPSK, 8-PSK, 8-QAM, and a 16-QAM. These are achievable constellations that have unique roles in their design for testing. The QPSK qualifies as the lowest order QAM and a second order PSK which allows crossover between the BPSK and the 16QAM in results. The 8-PSK allows for understanding of the effects denser constellation packing on a single radian on detection. Lastly, the 8-QAM is essentially a bi-amplitude QPSK with the outer radian's points phase shifted by 90° allowing for symmetry with a maximized minimum distance at the cost of amplitude sensitivity. This acts as another interesting crossover between QAM and PSK where constellation density comes into consideration. Observing the results should provide effective correlation of SNR to error probability within increasing order and

varying complexities of constellations. Figures 11 to 15 display both the highest stable SNR for each constellation at 16Sps and the lowest SNR where streaming quality was exceptional with low error rates showing the saturation of their bounds by noise around the symbol.

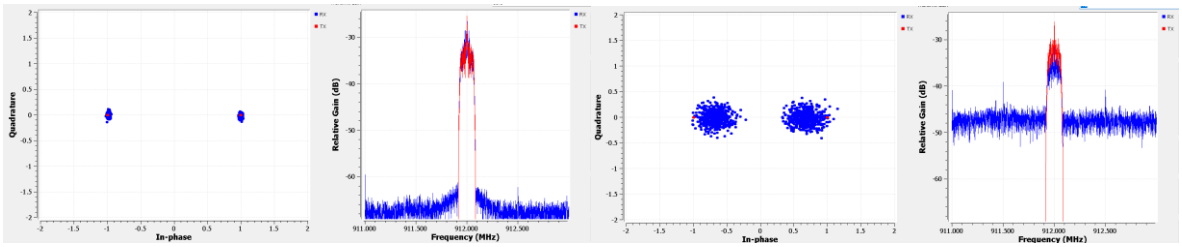

Figure 11: Results for BPSK at 16 SPS for both high SNR (left) and the lowest SNR (right) before error greater than 10^{-6}

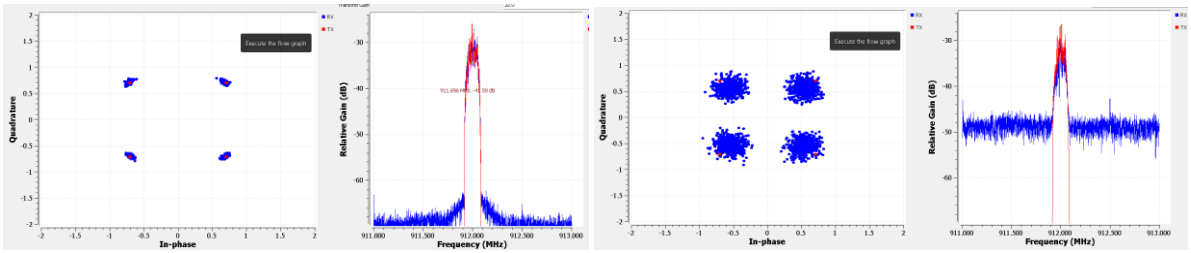

Figure 12: Results for QPSK at 16 SPS for both high SNR (left) and the lowest SNR (right) before error greater than 10^{-6}

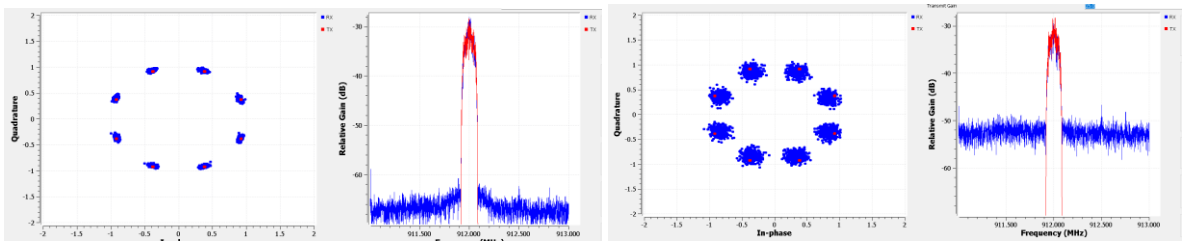

Figure 13: Results for 8PSK at 16 SPS for both high SNR (left) and the lowest SNR (right) before error greater than 10^{-6}

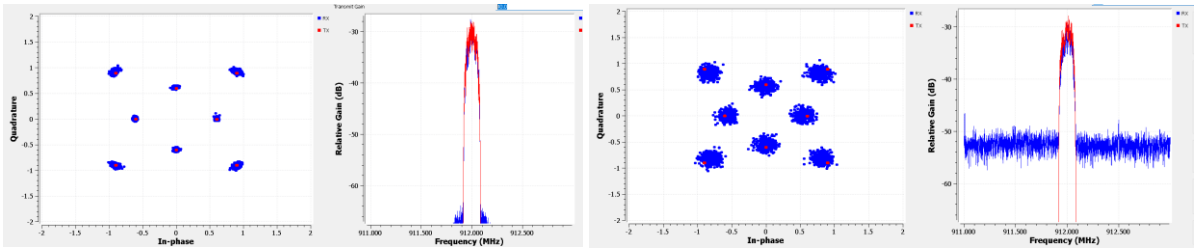

Figure 14: Results for 8QAM at 16 SPS for both high SNR (left) and the lowest SNR (right) before error greater than 10^{-6}

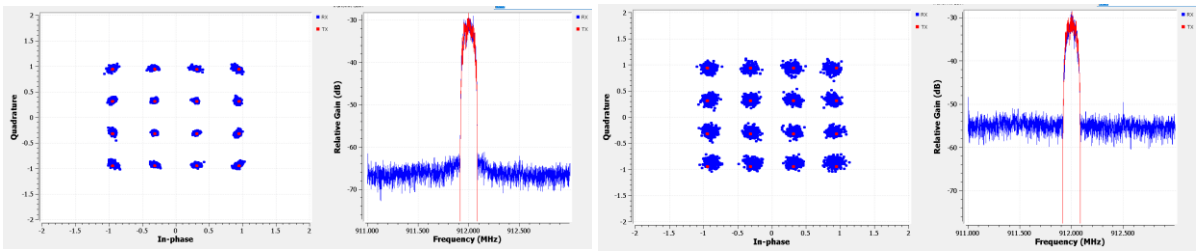

Figure 15: Results for 16QAM at 16 SPS for both high SNR (left) and the lowest SNR (right) before error greater than 10^{-6}

Signals were tested on the ISM band at 912MHz and held under 30dBm at the transmitter port with a 5.5dBi antenna with a bandpass filter to prevent harmonic resonance in higher frequency bands. To further ensure compliance, constant dense tones tested had a 500KHz bandwidth and low duty cycles with no frequency hopping for narrower bandwidths (250KHz at 32SPS) as the 20dB bandwidth compliance rule in FCC guidelines (Section 15.247(a)(2)). The densest constellation as seen in figure 16 achieved was a 64QAM requiring 28dB SNR for PER<10^-6 but appeared to require the entire transition band being detectable to achieve a reasonable signal.

A few options are available in GNU Radio's MPSK SNR block for SNR detection such as SVR, skewness, 2nd and 4th Moment and a simple detector, but none provided reasonable results that correlated to the observed data where the value was either inaccurately scaled without knowing the scaling factor or showed correct values until gain was reduced and no changes were measured. Due to this issue, a custom SNR detection module was developed using basic theory with respect to the known pulse shape in the frequency domain, so a simple FFT analysis was used to obtain SNR measurements with great accuracy.

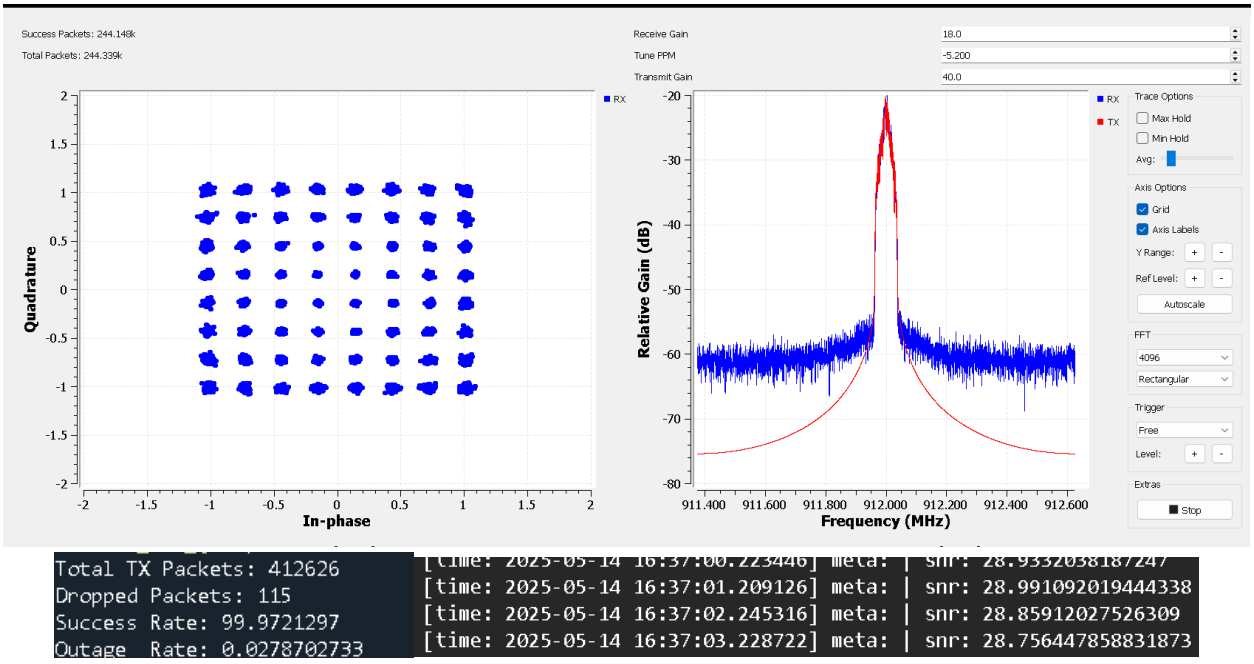

*Figure 16: Highest constellation order was 64QAM obtaining PER=3*10^-4 at 28dB SNR using 32SPS*

IV. Measurement Methodology

Error with respect to SNR was difficult given the system has evident issues with phase locking at high order constellations despite efforts. PER/BER/SER is determined based on a test run using

virtual sinks to determine the mean dropped packets, not OTA, and subtract this from the results from OTA testing. If there were no packets dropped, delay would be added to output and just measured, but this is neither available nor stable. Measurements are in the form of PER as the lowest precision measurable in the system designed due to real-time constraints. The duration of the test was timed where error becomes less than 10^{-5} approaching 10^{-6}, and instability in the system requires multiple tests to obtain a lock for this duration but it was possible for each constellation. It was further noted that the system appeared to have nearly flawless audio playback for $P_e = 10^{-6}$ and clipping was audibly evident for error greater than 10^{-5} and was about the same duration of a song at 48KHz, so this was used as a general reference for assuming conditions were suitable to continue running the test instead of restarting.

During each test, the SNR tests begin at 35dB SNR and end when the Pe>0.01. The lowest SNR for $P_e < 10^{-6}$ is recorded for each constellation for analysis. The SNR is not obtained using a conventional numerical method but instead uses a more rudimentary design where a custom embedded python block uses the RRC modulation parameters excess bandwidth (aka roll-off factor), sample rate and symbol rate with the moving average PSD filter parameters of FFT size and number of frames to average with an output duration to limit the load on the write and print console. These parameters are used to find the power under the passband, then the power in the noise region and find their ratio while omitting the transitional region from the calculations which would distort results from varying transitions widths. An outline of the bounds used for calculating SNR is provided in figure 17 and the standard equation for signal to SNR as equation 4 where the power per received signal's FFT sample is obtained for the passband and the noise by dividing by the number of samples in each region N_{pass} and N_{noise} respectively to then average the ratio of these two signals over N_{ave} FFT frames.

$$SNR = \frac{1}{N_{ave}} \sum_{a=0}^{N_{ave}-1} \left[\frac{\left(\frac{FFT_a^{pass}\{y\}}{N_{pass}} \right)}{\left(\frac{FFT_a^{noise}\{y\}}{N_{noise}} \right)} \right] \tag{4}$$

The running average smooths out non-persistent background signal and changes in the pulse's gain due to the modulation scheme used, such as with the QAM constellation. To simulate changes in SNR in a controlled manner, the transmit gain has a widget that is used for incrementally lowering the signal power at the receiver. Unfortunately, it should be noted that this method is highly process intensive in comparison to instantaneous time domain calculations, so active switching based on SNR with this method is inefficient and impractical outside of lab measurements despite its high accuracy. If implementing constant SNR monitoring methods such as pilot sequence time-based instantaneous calculations such as an improvement to the Zadoff-Chu pilot sequence that measures symbol by symbol [8] would be implemented to reduce overhead while maintaining accuracy. Due to time constraints, no coding gain is evaluated in this article, such as Reed-Solomon FEC convolutional encoding (combined is concatenated coding), but will likely be explored in the

future to ensure that the coding method provides a practical amount of overhead for the coding gain obtained compared to increasing samples per symbol.

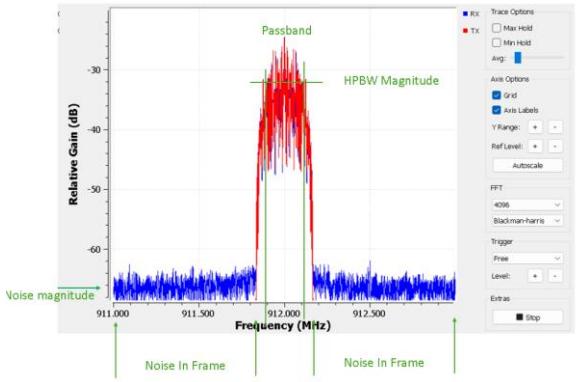

Figure 17: Regions outlined for reference of SNR calculations

For all data collected, the parameters used were a sample (of symbols for GNURadio and HackRF) rate of 2MHz, a true symbol rate of 500KHz (or 4 samples per symbol), an excess bandwidth of 1 to obtain a 1MHz channel width due to needing to shift the DC signal from the center frequency for improving signal integrity. The message packets payloads are 252 bytes in length with a CRC32 header and checksum multiplexed to make a packet length of 264 bytes. Excess bandwidth of 0.35 also aided in signal lock of higher ordered constellations and was used to aid in acquiring signals for the constellations of order greater than 4. The results have no redundancy, so no averaging occurred other than for SNR detection for removal of background by running a kernel which averages over N frames of the FFT size. SNR data is collected periodically, and tags allow for counting of success and total packets.

The PER is verified using a CRC32 checksum appended to each payload where delivered payloads are counted using tags appended to their stream after confirmation from the checksum and access code correlator. The tolerance is set to 0, allowing no errors in the header or payload, therefore PER is the lowest resolution of verification available and is used synonymously in this article.

Note that for M-Ary QAMs Y. Kim and G. Hwang found that their BER averaged 10^{-4} for SNR was lower than 30dB but appeared to quickly drop to 10^{-7} between 30dB and 40dB SNR, the results obtained for the QPSK and the 16QAM in the system used here provided similar results.

V. Results and Comparison with Literature

OTA Results

M-PSK	sps=2	ΔSNR	sps=4	ΔSNR	sps=8	ΔSNR	sps=16	ΔSNR	UNIT
2	11		7		4		2		dB
4	16	5	12	5	7	3	4	2	dB
8	21	5	17	5	13	5	8	4	dB
M-QAM	**sps=2**	**ΔSNR**	**sps=4**	**ΔSNR**	**sps=8**	**ΔSNR**	**sps=16**	**ΔSNR**	**UNIT**
4	16		12		7		4		dB
8	22	6	17	5	12	5	8	4.5	dB
16	27	5	22	5	17	5	11	3	dB

Calcluated Results

M-PSK	sps=2	ΔSNR	sps=4	ΔSNR	sps=8	ΔSNR	sps=16	ΔSNR	UNIT
2	10.5		7.5		4.5		1.5		dB
4	13.5	3	10.5	3	7.5	3	4.5	3	dB
8	19.5	6	16.5	6	13.5	5	10.5	6	dB
M-QAM	**sps=2**	**ΔSNR**	**sps=4**	**ΔSNR**	**sps=8**	**ΔSNR**	**sps=16**	**ΔSNR**	**UNIT**
4	13.5		10.5		7.5		4.5		dB
8	17.5	4	14.5	4	11.5	4	8.5	4	dB
16	20.5	3	17.5	3	14.5	3	11.5	3	dB

Figure 18: OTA and computed results for SPS of 2-16 of all constellations.

The results somewhat match the expected values. Unfortunately, there is not much documentation on the direct effect of oversampling with respect to RRC pulse shaped signals. What is particularly interesting is that the calculated results for 4, 8 and 16-QAM at 4 SPS provided matching results to the NNUB error approximation in [1] as seen in the results in figure 18. The OTA results obtained appear to converge to the calculated values as SPS increases towards infinity. The change in SNR in testing appears to have a 5dB change for each doubling of the interpolation, either decrease in M or increase in SPS but the calculations only show around 3dB increased SNR required for lower interpolations. This 2dB discrepancy at lower interpolations could either be due to hardware losses or lack of consideration for the losses due to inherent ISI introduced from the channel and RRC filtering where higher interpolations smooth this error. It appears, as expected, results match a Texas Instruments article [12] that many of the shortcomings that come from budget SDRs can be mitigated with oversampling (not directly as samples per symbol in this case, but still applicable) being described as “cheating physics”.

VII. Discussion

It was found that increasing the constellation density increased the probability of error while oversampling using samples per symbol decreased this SNR requirement. It was also found that there are likely larger SNR requirements than predicted for low SPS due to the channel and pulse shaping filters but appears to converge at high SPS providing detection like higher quality radios at the cost of overhead.

Despite this valuable SNR gain, it is apparent that a good signal generator is necessary to reduce overhead and reliability where SDRs are generalized to the point where both physical (requiring tuned bandpass filter, linear amplifier and DC block) and virtual overhead (signal correction, GUI/OS, and bytewise minimum handling) are required to compensate for their generality. This experiment has provided evidence that an SDR that can produce similar results to specialized radio

designs but will ultimately require more energy, processing, and memory to obtain lower quality results with similar specifications. For development of practical systems where a lock is required to be more stable and a significant portion of the overhead should not be designated to correction with no onboard DSP, it is speculated that a minimum quality requirement is essential where the HackRF one is not good for testing quality that would effectively verify signal power results for FCC certification, a module such as a USRP B200 which has a full duplex, 60MSps, 12-bit ADC and a lab-grade signal generator would be needed for practical use of an SDR to effectively compare to specialized radio systems.

There were various limitations on obtaining an ideal response including the hardware being non-linear and can only use the lower end of its sample rates. If the HackRF One was a full duplex with separate user and base station would have allowed for handshakes providing better synchronization and acknowledgement of when data streams begin and end for data collection purposes. Operating two devices on a single computer likely was a major bottleneck and was asynchronous at high rates, which contributed to lock failure when over and under runs occurred. Another limitation is two samples per symbol takes up the entire band and even with hardware and software DC blocking the program still has difficulty with the DC tone and distorts the results, where shifting excessively distorts the signal. Usually increasing the sample rate and interpolating before the radio would fix this, which was done, but 2MHz is already the low end of the functional sample rates so results were still somewhat distorted for two samples per symbol. Yet another limitation was the blocks available within GNU Radio. Due to not having matching blocks for C and Python the program was forced to be developed using Python due to limited time for design, so this software limitation is a major contributor to the hardware's realizable sample rate being severely limited. Finally, due to imprecise design of the loop recovery components, higher order constellations would lose lock, so SNR values were taken only when lock had been secured.

Various considerations are needed for future testing on the fundamental system and there are tests that should be done such as the SNR required for each in a Rician or Rayleigh channels given that these are more practical channels that the device will operate over for at least one of the radios. Multiple access channels such as OFDM (common in data networks), CDMA (common in cellular non-GSM networks) and TDMA (used in GSM networks) channels given that these are very commonly used.

If still using GNURadio for testing, the first note for speed to allow for real-time analysis is using C DSP backend but also more stable results should be possible if the stream is converted to polymorphic data units (PDUs) and would likely reduce overhead and bottlenecks due to being asynchronously passed, also making then better for stable real-time data analysis of SNR and BER or allowing for higher sample rates at the radio due to lower requirements of its physical buffer.

Furthermore, coding gain greatly affects the SNR required so convolutional (most common), turbo, Reed-Solomon and Viterbi, the easiest in GNU Radio is using their forward error correction (FEC) block to introduce coding, primarily convolutional. It was found, referencing figure 17, that 9dB

to 16dB gain can be achieved from interpolating by 8, but this requires substantially higher CPU but with linear complexity gain $O(SPS)$ in comparison to the gain obtained from memory intensive methods such as RS convolutional [15] (big-O complexity gain of $O(N^2 + L2^K)$ with Viterbi decoding for N bits where L is sequence length and K as memory depth) providing 6dB to 10dB or trellis coding (big-O complexity gain of Viterbi decoder only, $O(L2^K)$) proposed by Gottfried Ungerboeck [14] respectively one uses moderate redundancy and the other "mapping by set partitioning" providing 3dB to 6dB gain with only a fractional gain in processing requirements from the uncoded constellation.

VIII. Conclusion

Exploration of SNR requirements in basic digital modulation schemes such as PSK and QAM has provided insight into the functionality verification and the shortcomings of affordable SDR devices available to consumers and the need for effectively deciding on an SDR qualified for the task whether it is for hobby or research. This experiment has provided a relation that is valuable for research, design, quality comparison of devices, improving detection without increasing transmit power, and perspective that SDR's versatility provides a designer the ability to manage the signal quality in ways that specialized radio systems cannot compete with despite commonly being more effective devices. Given these degrees of freedom, it is important to consider that effective coding that produces requires minimal processing minimizes memory requirements and provides substantial detection gain for the overhead to reduce the SNR required for detection either allowing for greater distance or lower error transmissions. Further research should be placed on testing proposed SNR detectors OTA to choose a more practical design and apply it to various methods of increasing detection gain such as common coding methods, MIMO, more efficient modulation schemes like MSK/GMSK, various equalizers like Zero-Forcing or MMSE, optimization of the minimum Hamming constellation distance and general exploration into more methods.

References

[1] Cioffi, John M. "Modulation and Canonical Reception." Available: https://cioffi-group.stanford.edu/doc/book/chap1.pdf

[2] Ossmann, Michael. "Hardware Components — HackRF documentation," Readthedocs.io, 2021. https://hackrf.readthedocs.io/en/latest/hardware_components.html (accessed May 12, 2025).

[3] Lichtman, Marc. "Pulse Shaping — PySDR: A Guide to SDR and DSP using Python," CC BY-NC-SA 4.0, 2025. pysdr.org. https://pysdr.org/content/pulse_shaping.html

[4] "Tutorials - GNU Radio," wiki.gnuradio.org. https://wiki.gnuradio.org/index.php/Tutorials

[5] “Understanding USRP and Its Importance in SDR,” Apexwaves.com, 2025. https://www.apexwaves.com/blog/understanding-usrp-and-its-importance-in-sdr/ (accessed May 12, 2025).

[6] H. Meyr, M. Moeneclaey, and S. A. Fechtel, Synchronization Techniques for Digital Receivers. New York, NY, USA: Wiley, 1998.

[7] Y. Qi, G. Yang, S. Chen, and Y. Zhang, "Overview: 5G Over-the-Air Measurement Challenges," IEEE Trans. Microw. Theory Techn, vol. 67, no. 1, pp. 16-28, 2019.

[8] Khan, Abid Muhammad, Jeoti, Varun, Rehman, Muhammad Zaka Ur, Jilani, Muhammad Taha, Chugtai, Omer, Rehmani, Mubashir Hussain, Pilot-Based Time Domain SNR Estimation for Broadcasting OFDM Systems, Journal of Computer Networks and Communications, 2018, 9319204, 8 pages, 2018.

[9] Manzoor, Shahid & Jeoti, Varun & Kamel, Nidal & Khan, Muhammad. (2011). Novel SNR estimation technique in Wireless OFDM systems. Int. J. Future Generation Commun. Netw. 4.

[10] Bala, Diponkor & Waliullah, G. M. & Islam, Md & Abdullah, Md & Hossain, Mohammad. (2021). Analysis of the Probability of Bit Error Performance on Different Digital Modulation Techniques over AWGN Channel Using MATLAB. 7. 9-18.

[11] Hopper, RJ. “SSZTCA9 Technical article | TI.com,” Ti.com, 2015. https://www.ti.com/document-viewer/lit/html/SSZTCA9 (accessed May 12, 2025).

[12] Kester, Walter. “MT-001: Taking the Mystery out of the Infamous Formula, ‘SNR = 6.02N + 1.76dB,’ and Why You Should Care” Analog Devices, Inc 2019. https://www.analog.com/media/en/training-seminars/tutorials/MT-001.pdf

[13] T. Jiang and Y. Wu, "An Overview: Peak-to-Average Power Ratio Reduction Techniques for OFDM Signals," IEEE Commun. Surveys Tuts., vol. 10, no. 4, pp. 407-428, Fourth Quarter 2008.

[14] G. Ungerboeck, "Channel coding with multilevel/phase signals," IEEE Trans. Inf. Theory, vol. IT-28, no. 1, pp. 55-67, 1982.

[15] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," J. Soc. Indust. Appl. Math, vol. 8, no. 2, pp. 300-304, 1960.