NTIA Technical Memorandum 19-539

Dual-band Tactical Radio Feasibility Study

Geoffrey A. Sanders
Anna E. Paulson
Douglas J. Anderson

technical memorandum

U.S. DEPARTMENT OF COMMERCE • National Telecommunications and Information Administration

NTIA Technical Memorandum 19-539

Dual-band Tactical Radio Feasibility Study

Geoffrey A. Sanders
Anna Paulson
Douglas Anderson

U.S. DEPARTMENT OF COMMERCE

June 2019

DISCLAIMER

Certain commercial equipment and materials are identified in this report to specify adequately the technical aspects of the reported results. In no case does such identification imply recommendation or endorsement by the National Telecommunications and Information Administration, nor does it imply that the material or equipment identified is the best available for this purpose.

CONTENTS

FIGURES

TABLES

ABBREVIATIONS/ACRONYMS

ARA	Antenna Research Associates
BAA	Broad Agency Announcement
BER	Bit-error-rate
COW	Cell On Wheels
DARPA	Defense Advanced Research Projects Agency
EIRP	Effective Isotropic Radiated Power
FFT	Fast-Fourier Transform
GGA	NMEA Global Positioning System Fix Data
GNU	GNU's Not Unix
GPS	Global Positioning System
GRC	GNU Radio Companion
GUI	Graphical User Interface
I/Q	In-phase and Quadrature
ITS	Institute for Telecommunication Sciences
NMEA	National Marine Electronics Association
NTIA	National Telecommunication and Information Administration
Network UP	Network Universal Persistence
PRBS-15	15th order Pseudo-Random Bit Sequence
PXA	Keysight N9030B Signal Analyzer
QPSK	Quadrature Phase Shift Keying
RMS	Root Mean Square
SNR	Signal-to-Noise Ratio
STA	Special Transmit Authority
STO	Strategic Technology Office
USRP	Universal Software Radio Peripheral
VSA	Vector Signal Analyzer
VSG	Vector Signal Generator

DUAL-BAND TACTICAL RADIO FEASIBILITY STUDY

Geoffrey A. Sanders, Anna Paulson, and Douglas Anderson[1]

In support of the Network Universal Persistence (Network UP) Broad Agency Announcement (BAA) sponsored by the Defense Advanced Research Projects Agency (DARPA), the Institute for Telecommunication Sciences (ITS) was asked to perform a dual-band tactical radio feasibility study. Propagation path loss and bit-error-rate (BER) were measured in different environments in the Boulder, CO, area at two different frequencies. The purpose was to examine how using a low frequency control channel while transmitting data over a higher frequency channel could potentially reduce tactical radio link outages. The results of this study are provided in this report and they indicate that it is feasible that a tactical dual-band radio using a low frequency control channel could have improved network connectivity.

Keywords: bit-error-rate, Network UP, path loss, propagation

1. INTRODUCTION

In January 2018, the Defense Advanced Research Projects Agency (DARPA) Strategic Technology Office (STO) issued a Broad Agency Announcement [1] (BAA) for the Network Universal Persistence (Network UP) program. The BAA calls for the development and demonstration of a tactical radio network that maintains reliability through difficult signal propagation conditions.

Current radio technologies use a single radio channel to transmit data and control messages. Both data and control messages can experience high bit error rates if the radio is operating at a high frequency and in an urban or residential environment subject to time-varying signal attenuation and multipath radio wave propagation impairments. High control message bit error rates are particularly troublesome because they can break network connections that take several minutes to reestablish. Transmitting the control messages over a separate radio channel at a lower frequency where these radio channel impairments are less likely can potentially increase radio network reliability.

The purpose of this study was to demonstrate the feasibility of such a “dual-band” radio network by measuring propagation path loss and bit-error-rate (BER) in urban and residential environments with a lower frequency simulated control channel and a higher frequency simulated data channel.

[1] The authors are with the Institute for Telecommunication Sciences, National Telecommunications and Information Administration, U.S. Department of Commerce, Boulder, CO 80305.

2. MEASUREMENT DESCRIPTION

The measurement system was comprised of a fixed position transmitter and a mobile receiver mounted in a vehicle. Signal, BER, time, and position data were collected as the vehicle was driven over planned routes. The signal data was used to calculate average received power and propagation path loss for comparison to BER at the same time or position.

Measurements were executed at two bit rates and two carrier frequencies over two routes for a total of eight drive tests summarized in Table 1. Quadrature phase shift keying (QPSK) modulation at 100 kbps and 1 Mbps bit rates were used at 286.375 and 1702.5 MHz carrier frequencies.[2] One route went through the Pearl Street pedestrian mall in downtown (urban) Boulder, CO, while the other route traversed two residential neighborhoods in southern Boulder.

Table 1. Summary of overall test cases amounting to a total of eight drive tests.

Channel	Bit Rate	Environment
Data (1702.5 MHz)	100 kbps	Urban
		Residential
	1 Mbps	Urban
		Residential
Control (286.375 MHz)	100 kbps	Urban
		Residential
	1 Mbps	Urban
		Residential

2.1 Measurement System

2.1.1 Transmitter

The measurement system transmitter was housed in a Cell on Wheels (COW) trailer with a climate controlled equipment cabinet, onboard generator, and 60 foot pneumatic mast at a fixed location on Kohler Mesa just west of the U.S. Department of Commerce Boulder Laboratories in Boulder, CO. Figure 1 provides a satellite image of Boulder showing the transmitter location (yellow X), the residential drive route area (red ovals), and the urban route area (green oval). Figure 2 shows a picture of the COW at the transmit location.

A block diagram of the transmit system is provided in Figure 3 and Table 2 provides the relevant specifications for the transmitter required to properly calculate effective isotropic radiated power (EIRP). The signal was created using an Agilent 8267D vector signal generator (VSG). An Antenna Research Associates (ARA) SAS-220 discone antenna radiated the signal through the radio channel to the receiver. The VSG output power was adjusted such that the EIRP was

[2] Both frequencies were subject to Special Transmit Authority (STA) approval. The 286.375 MHz used to represent the control channel was a new STA. The 1702.5 MHz used to represent the data channel was an existing STA.

37 dBm for each drive test. Transmit power was measured at the coupler output as seen in Figure 3 and EIRP was calculated from this measurement by taking into account all the gains and losses of the transmit system as shown in Table 2.

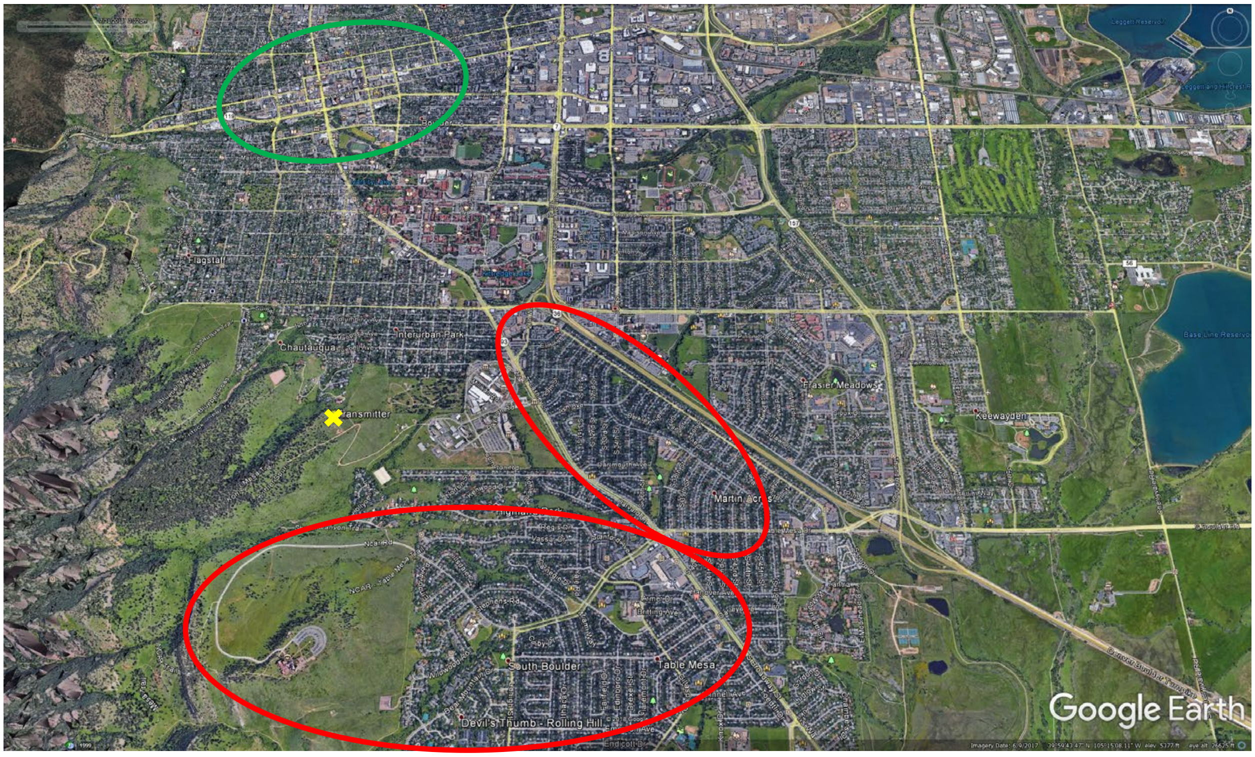


Figure 1. Transmitter location (yellow X), residential area (red circles), and urban environment (green circle) in Boulder, CO.

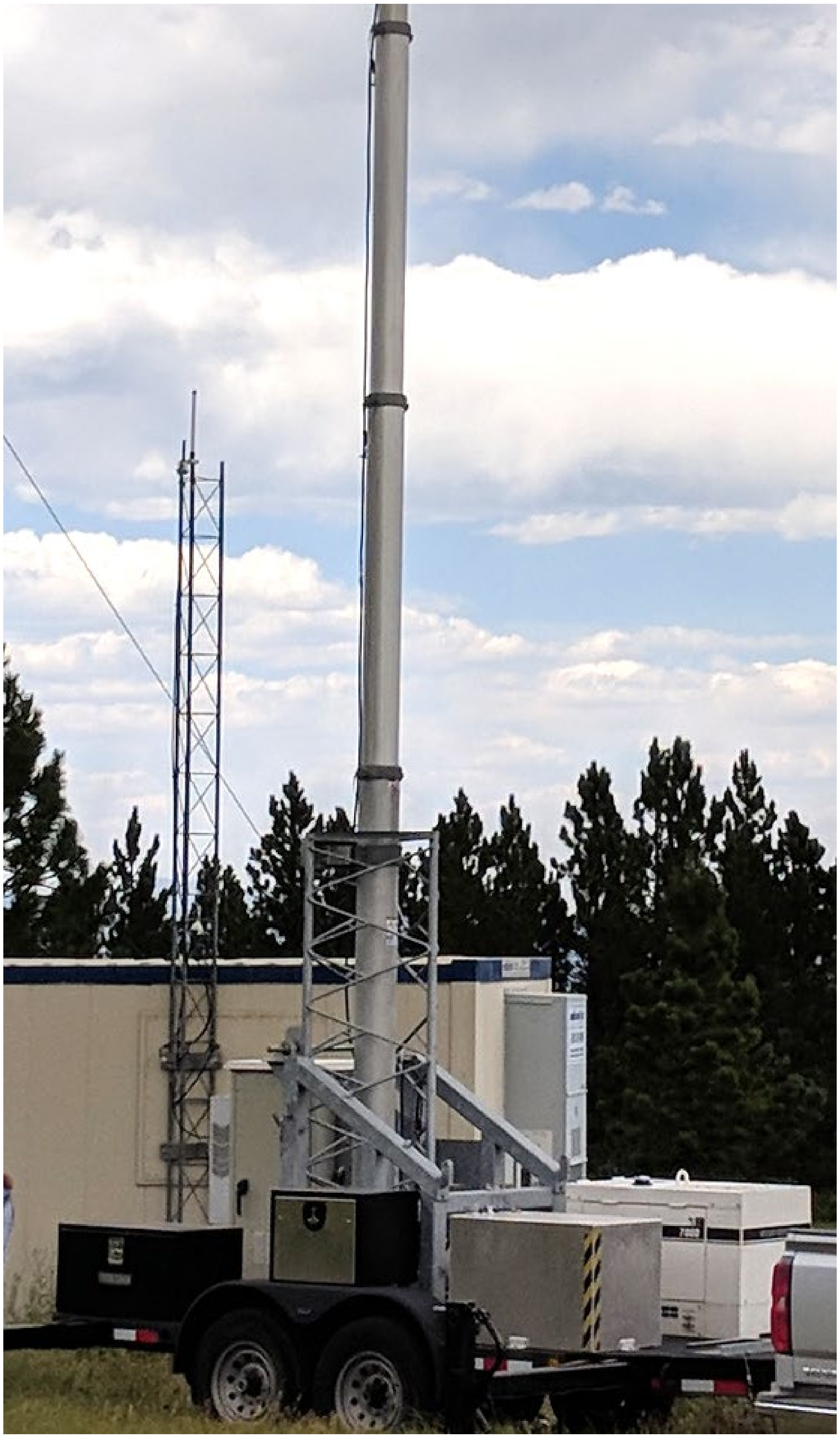


Figure 2. Picture of the COW set up at the transmit location on Kohler Mesa. The mast is extended 60 feet with the antenna mounted on the top of it. The equipment cabinet is seen near the rear of the trailer behind the mast.

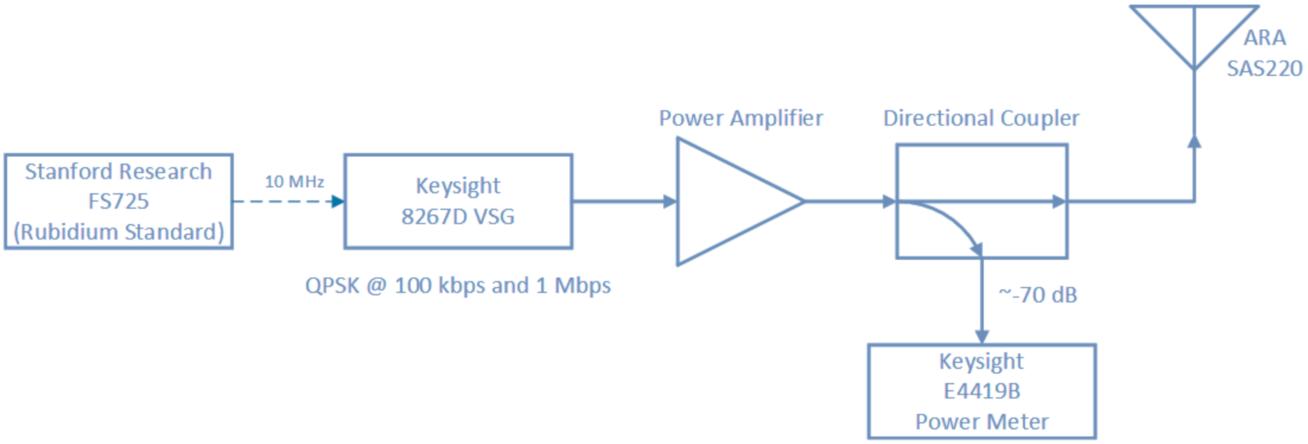


Figure 3. Transmitter block diagram.

Table 2. Transmitter system gains and losses used to determine EIRP.

	Data Channel (1702.5 MHz)		**Control Channel (286.375 MHz)**	
Description	**100 kbps**	**1 Mbps**	**100 kbps**	**1 Mbps**
Power Meter Measurement (dBm)	-27.44	-27.38	-32.44	-32.5
Coupler Loss (dB)	70.4	70.4	70.6	70.6
Cable (Coupler to Antenna) Loss (dB)	4.81	4.81	1.9	1.9
Azimuthal Antenna Gain (dBi)	-1.2	-1.2	0.8	0.8
EIRP (dBm)	36.95	37.01	37.06	37

2.1.2 Receiver

The measurement system receiver was housed in the ITS mobile measurement vehicle seen in Figure 4. A block diagram of the receive system is provided in Figure 5. The same type of antenna (ARA SAS-220) used to transmit the signal was used to receive the signal. The antenna output was split between the two primary measurement instruments. A Keysight N9030A (PXA) was operated in VSA mode to collect in-phase and quadrature (I/Q) data, and an Ettus Research Universal Software Radio Peripheral (USRP) B200 was used to demodulate the received signal and report BER.

The PXA collected 1 second long I/Q recordings as often as possible, which resulted in a recording every 2–3 seconds largely due to the time required to save the recordings to the hard drive. Associated with each PXA recording was a Global Positioning System (GPS) National Marine Electronics Association (NMEA) essential fix data (GGA) sentence containing a GPS time stamp, latitude, and longitude. This NMEA sentence was queried from the Trimble precision GPS receiver as seen in the receiver block diagram shown in Figure 5 at the beginning of each recording. Table 3 provides the settings that were used for the PXA recordings for each of the measurement test cases. Full descriptions of all the available NMEA sentences can be found in [2].

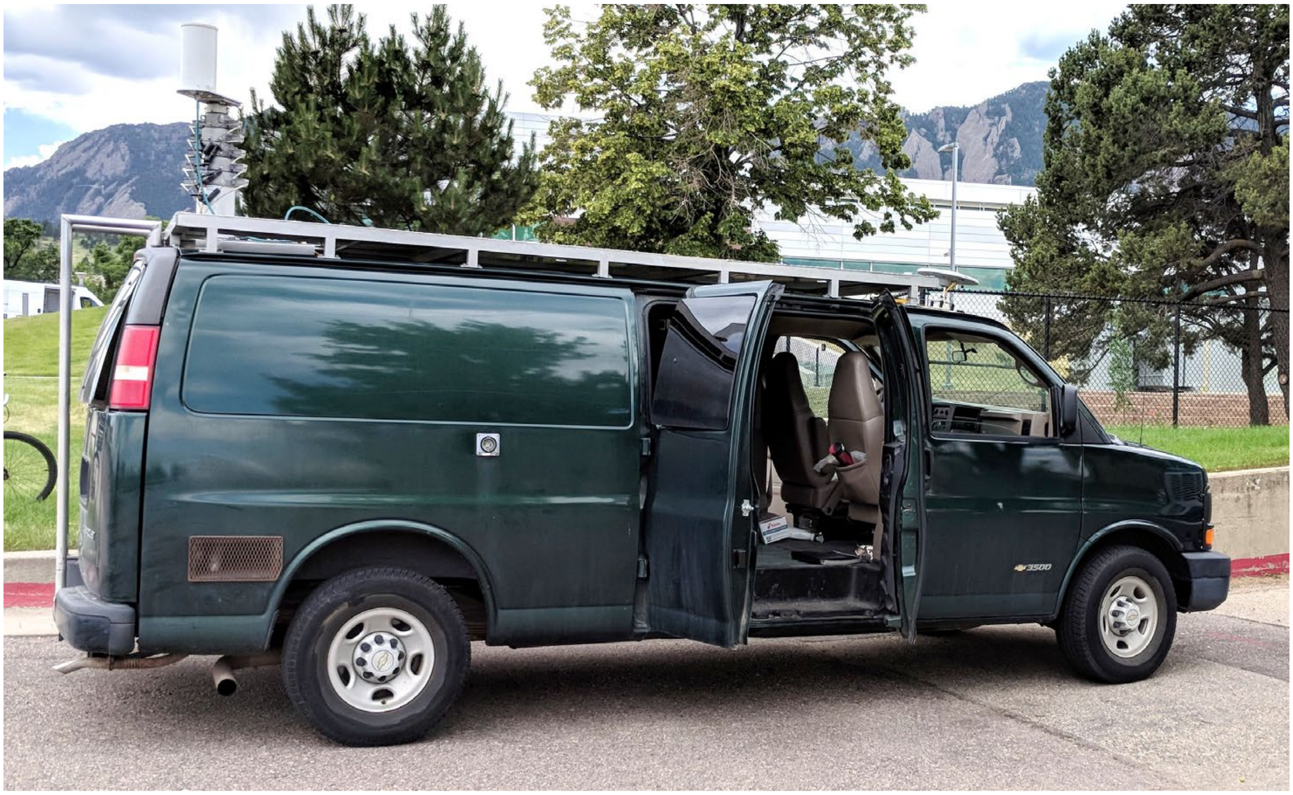


Figure 4. ITS mobile measurement vehicle.

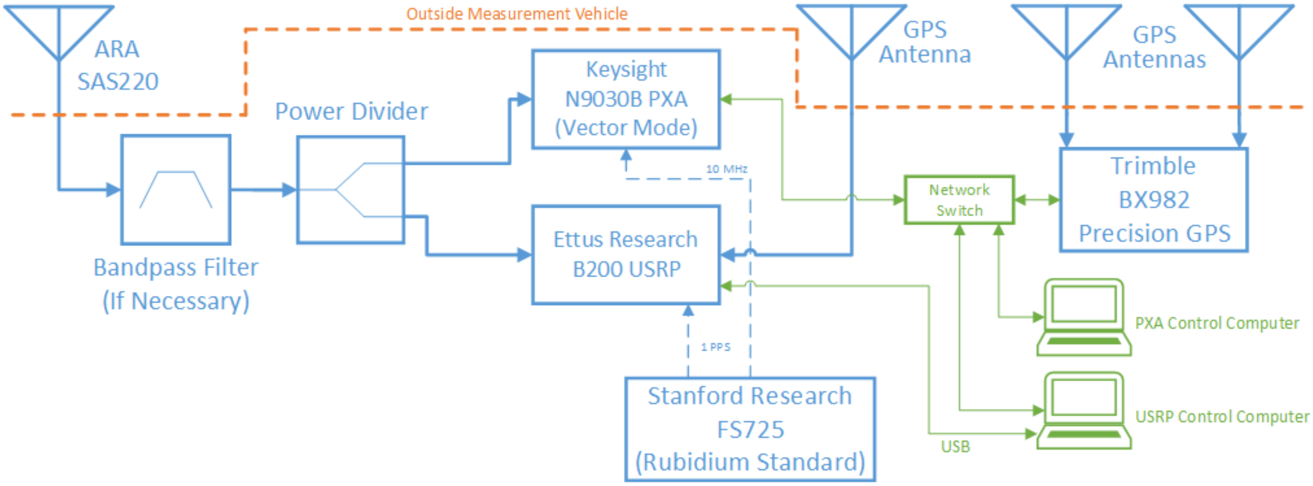


Figure 5. Receiver block diagram.

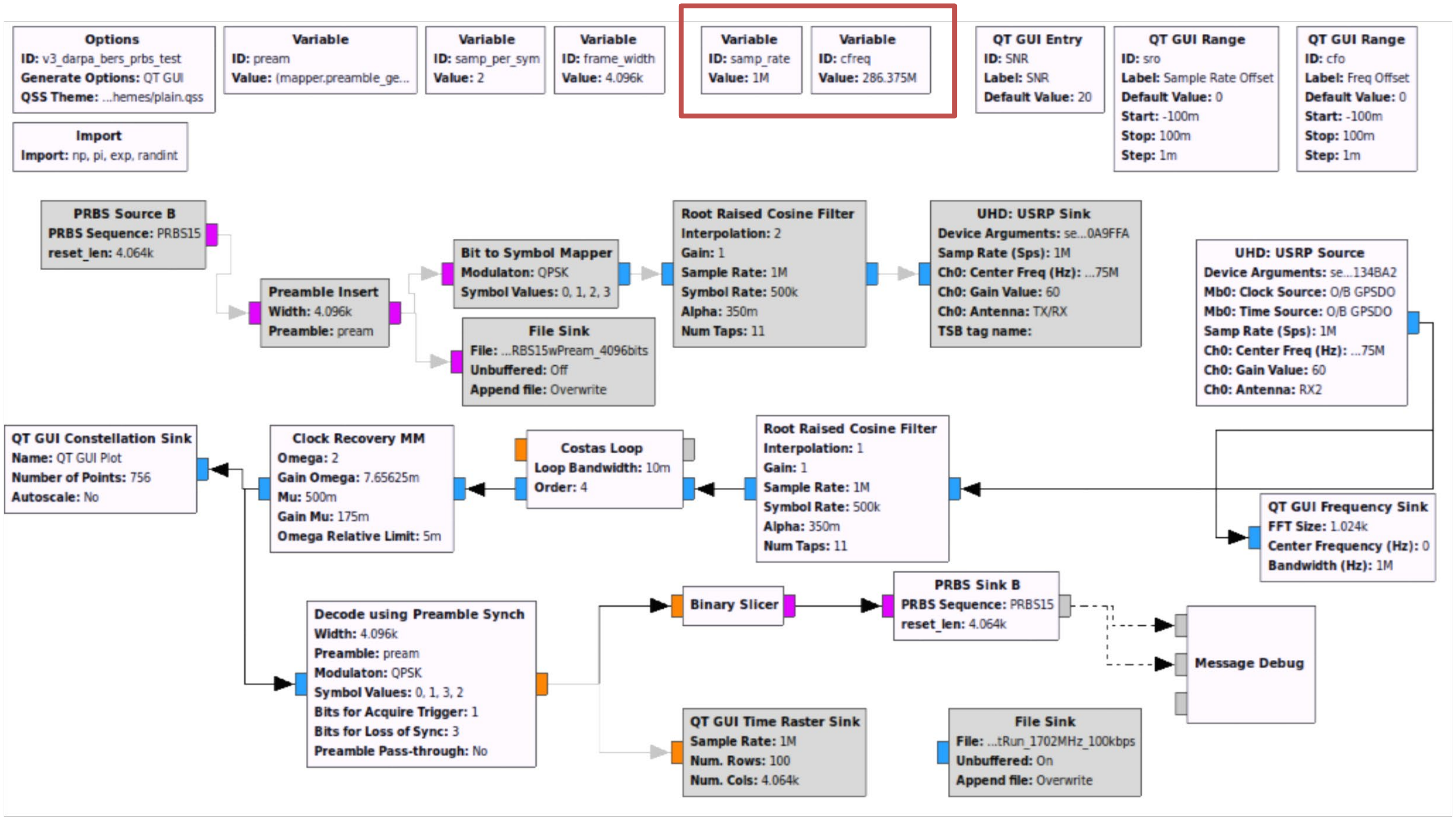
Table 3. Settings on the PXA in VSA mode to collect the I/Q recordings.

	Data Channel		**Control Channel**	
Setting	**100 kbps**	**1 Mbps**	**100 kbps**	**1Mbps**
Center Frequency (MHz)	1702.5	1702.5	286.375	286.375
Recording Length (Sec)	1	1	1	1
Span (MHz)	0.4	4	0.4	4
Range (dBm)	-42	-42	-42	-42

The USRP had a GPS module installed so that it was also able to report the GPS GGA sentence. GNU's Not Unix (GNU) Radio Companion (GRC) software application was used to configure and operate the USRP. Functions from an open source software library called *gr-mapper* available on GitHub [3] were used in conjunction with GRC for mapping bits and symbols and computing BER. Some of the blocks provided in *gr-mapper* were modified so BER was computed over one second and reported along with the GPS GGA sentence captured within that second. Unfortunately, the addition of the GPS reporting added a delay to the USRP reporting so these data points were only reported every 2–3 seconds.

The GRC flow graph that was used is shown in Figure 6. The key variables that were set for each test case are provided in Table 4. Any parameters not shown in Table 4 were held constant for all the test cases. While actively logging measurements, the USRP produced a text file containing BER values with associated GPS GGA string. An example of an output for a single data point is provided below. A full description of the BER measurement system including the complete configuration of the GRC flow graph and an explanation of the modifications that were made to *gr-mapper* can be found in Appendix A.

```
GPS_GPGGA: $GPGGA,174245.00,3959.6643,N,10515.6943,W,2,09,1.1,1666.1,M,-21.3,M,,*67
GPS_GPRMC: $GPRMC,174245.00,A,3959.6643,N,10515.6943,W,0.0,0.0,230818,,*29
Computer time: 2018-08-23 11:42:45.537337
NBits: 32512 NErrs: 0     BER: 0.0000E+00,    timestamp 1984.222851
******* MESSAGE DEBUG PRINT ********
((ber . 0) (timestamp . 1984.22))
************************************
```



Options
ID: v3_darpa_bers_prbs_test
Generate Options: QT GUI
QSS Theme: ...hemes/plain.qss

Import
Import: np, pi, exp, randint

Variable
ID: pream
Value: (mapper.preamble_ge...

Variable
ID: samp_per_sym
Value: 2

Variable
ID: frame_width
Value: 4.096k

Variable
ID: samp_rate
Value: 1M

Variable
ID: cfreq
Value: 286.375M

QT GUI Entry
ID: SNR
Label: SNR
Default Value: 20

QT GUI Range
ID: sro
Label: Sample Rate Offset
Default Value: 0
Start: -100m
Stop: 100m
Step: 1m

QT GUI Range
ID: cfo
Label: Freq Offset
Default Value: 0
Start: -100m
Stop: 100m
Step: 1m

PRBS Source B
PRBS Sequence: PRBS15
reset_len: 4.064k

Preamble Insert
Width: 4.096k
Preamble: pream

Bit to Symbol Mapper
Modulaton: QPSK
Symbol Values: 0, 1, 2, 3

File Sink
File: ...RBS15wPream_4096bits
Unbuffered: Off
Append file: Overwrite

Root Raised Cosine Filter
Interpolation: 2
Gain: 1
Sample Rate: 1M
Symbol Rate: 500k
Alpha: 350m
Num Taps: 11

UHD: USRP Sink
Device Arguments: se...0A9FFA
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): ...75M
Ch0: Gain Value: 60
Ch0: Antenna: TX/RX
TSB tag name:

UHD: USRP Source
Device Arguments: se...134BA2
Mb0: Clock Source: O/B GPSDO
Mb0: Time Source: O/B GPSDO
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): ...75M
Ch0: Gain Value: 60
Ch0: Antenna: RX2

QT GUI Constellation Sink
Name: QT GUI Plot
Number of Points: 756
Autoscale: No

Clock Recovery MM
Omega: 2
Gain Omega: 7.65625m
Mu: 500m
Gain Mu: 175m
Omega Relative Limit: 5m

Costas Loop
Loop Bandwidth: 10m
Order: 4

Root Raised Cosine Filter
Interpolation: 1
Gain: 1
Sample Rate: 1M
Symbol Rate: 500k
Alpha: 350m
Num Taps: 11

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

Decode using Preamble Synch
Width: 4.096k
Preamble: pream
Modulaton: QPSK
Symbol Values: 0, 1, 3, 2
Bits for Acquire Trigger: 1
Bits for Loss of Sync: 3
Preamble Pass-through: No

Binary Slicer

PRBS Sink B
PRBS Sequence: PRBS15
reset_len: 4.064k

Message Debug

QT GUI Time Raster Sink
Sample Rate: 1M
Num. Rows: 100
Num. Cols: 4.064k

File Sink
File: ...tRun_1702MHz_100kbps
Unbuffered: On
Append file: Overwrite

Figure 6. GRC flow graph used for configuration of the USRP for the BER measurement (Grayed-out blocks were disabled during testing).

Table 4. GRC variables that were changed depending on test case using the variable blocks in the red rectangle in Figure 6.

Setting	Flow Graph Variable Name	Data Channel		Control Channel	
		100 kbps	**1 Mbps**	**100 kbps**	**1Mbps**
Center Frequency (MHz)	cfreq	1702.5	1702.5	286.375	286.375
Sample Rate (ksps)	samp_rate	100	1000	100	1000

The USRP was characterized in the lab to measure noise figure, 1 dB compression point, and dynamic range for the settings that were used during testing. The results of this are provided in Table 5.

Table 5. Measured noise figure, 1 dB compression point, and dynamic range of the USRP.

Center Frequency (MHz)	Sample Rate (ksps)	Noise Figure (dB)	1 dB Compression Point (dBm)	Dynamic Range (dB)
286.375	1000	4.24	-48	61.76
286.375	100	5.29	-49	59.71
1702.5	1000	9.34	-46	58.6
1702.5	100	11.97	-46	56.03

2.2 Test Signals and Post-processing

The test signal was created by generating successive USRP data frames. The data frames were 4096 bits long consisting of a 32 bit preamble added to the first 4064 bits of the GRC 15th order pseudo-random (PRBS-15) bit sequence. Prior to transmission, the data frames were ray coded, QPSK modulated, and filtered with a 35% excess bandwidth root raised cosine filter [4]. The QPSK constellation before filtering is shown in Figure 7.

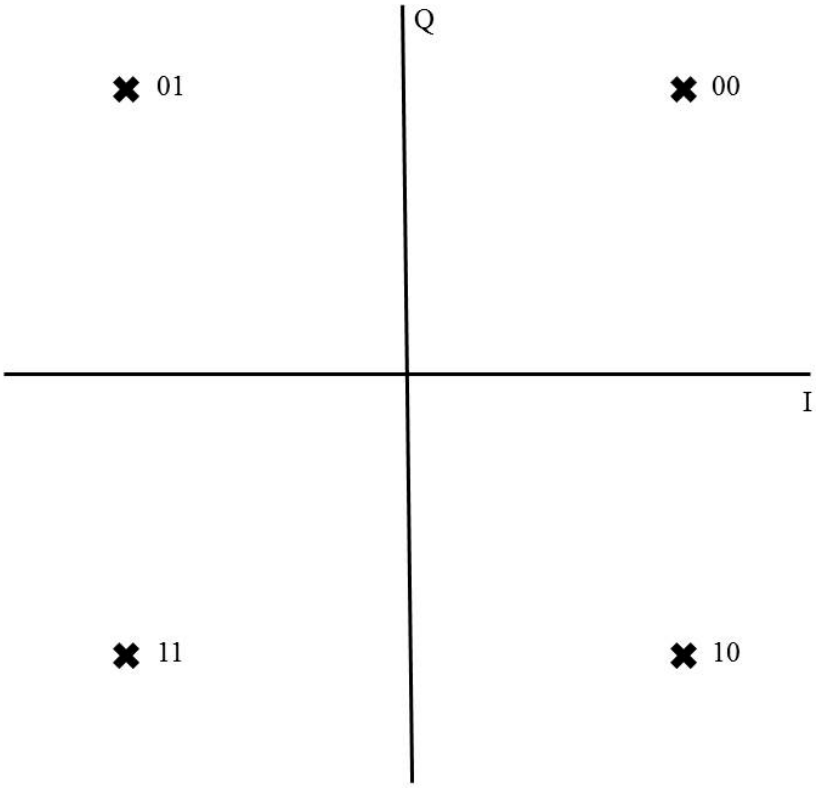


Figure 7. Gray Coded QPSK constellation showing symbol mapping used for this study.

BER and received signal power were computed over a one second interval. GRC computed the BER by generating the transmitted bit sequence,[3] comparing it to the demodulated bit sequence, and dividing the number of bit errors by the total number of information bits sent. Information in the preamble was not included in the BER calculation.

The I/Q data recorded by the PXA was used to calculate average received power. Prior to calculating the average received power, the PXA recording was filtered and resampled to the transmitted signal bandwidth equal to 1.35 times the symbol rate due to the 35% excess bandwidth root raised cosine filter.

The average received power was calculated by first converting the resampled peak voltage samples from the PXA recording to root mean square (RMS) power in milliwatts using (1) and then taking the average of these power values using (2). This average power was then converted to dBm. Compensating for the line losses and antenna gain, path loss was calculated using (3). Table 6 provides the values for this measurement system that were used to calculate path loss.

[3] The bit sequence was checked and it was noted that the gr-mapper PRBS Source block does not produce a proper maximum length PRBS-15 sequence. The BER measurement system was tested using a bit sequence with a known BER and it produced the correct values. A full description of the USRP configuration is provided in Appendix A.

PXA recording peak voltage samples are converted to RMS power by

$$P_{mW} = \left(\frac{{V_{Pk}}^2}{100}\right) * 1000 \tag{1}$$

where:

P_{mW} = Power in mW

V_{Pk} = Magnitude of peak Voltage I/Q samples from PXA recording in Volts

The average received power calculation is

$$P_{Avg} = \frac{1}{N}\sum_{n=1}^{N} P_{mW,n} \tag{2}$$

where N is the number of samples averaged over a 1 second period.

Path loss is calculated by

$$PL = EIRP - P_{Avg} + G_R - L_R \tag{3}$$

where:

PL = path loss in dB

$EIRP$ = effective isotropic radiated power (from Table 2) in dBm

G_R = gain of receive antenna in dBi

L_R = line losses of receive systems from antenna to instruments in dB

Table 6. Parameters describing the receive system required to calculate path loss.

	Data Channel (1702.5 MHz)		**Control Channel (286.375 MHz)**	
System Parameter	**100 kbps**	**1 Mbps**	**100 kbps**	**1 Mbps**
EIRP (dBm)	36.95	37.01	37.06	37
Receiver Line Loss (dB)	4.7	4.7	3.9	3.9
Receiver Antenna Gain (dBi)	-1.2	-1.2	0.8	0.8
PXA Sample Rate (kHz)	512	5120	512	5120
Resampled Sample Rate (kHz)	67.5	675	67.5	675

To find the maximum measurable path loss, the PXA was configured as it would be for the measurements, the input was terminated with a 50 Ω load and 10 recordings were made at each of the four PXA configurations. The average noise power in dBm was determined from the measured I/Q data using (1) and (2) and the maximum measureable path loss was calculated using (3). The maximum measurable path loss values are provided in Table 7.

Table 7. Maximum measurable path loss values derived from the measured PXA average noise power.

Channel	**Bit Rate**	**Measured Average Noise Power (dBm)**	**Maximum Measurable Path loss (dB)**
Data (1702.5 MHz)	100 kbps	-116.25	159.1
	1 Mbps	-106.25	149.16
Control (286.375 MHz)	100 kbps	-118.11	158.27
	1 Mbps	-108.17	148.27

2.3 Drive Routes

There were two drive routes used for this study. The first was through the urban downtown Boulder area near the Pearl Street Mall. This drive route was chosen as an example of an area that could have a lot of multi-path due to dense tall commercial buildings as well as traffic. These characteristics also contribute to clutter. Clutter causes additional path loss due to environmental factors such as buildings and vegetation in excess of propagation losses due to free-space and terrain effects [4].

The second drive route traversed two residential neighborhoods in South Boulder before heading up a hill next to the transmit site where the measurement vehicle had clear line-of-sight to the transmit antenna. The majority of this route was through neighborhoods with mature trees and single family homes. This route was chosen to represent an area that may experience high clutter losses through most of it.

2.3.1 Urban Downtown Boulder Drive Route

A map of the drive route is provided in Figure 8, followed by turn-by-turn instructions for the drive route.

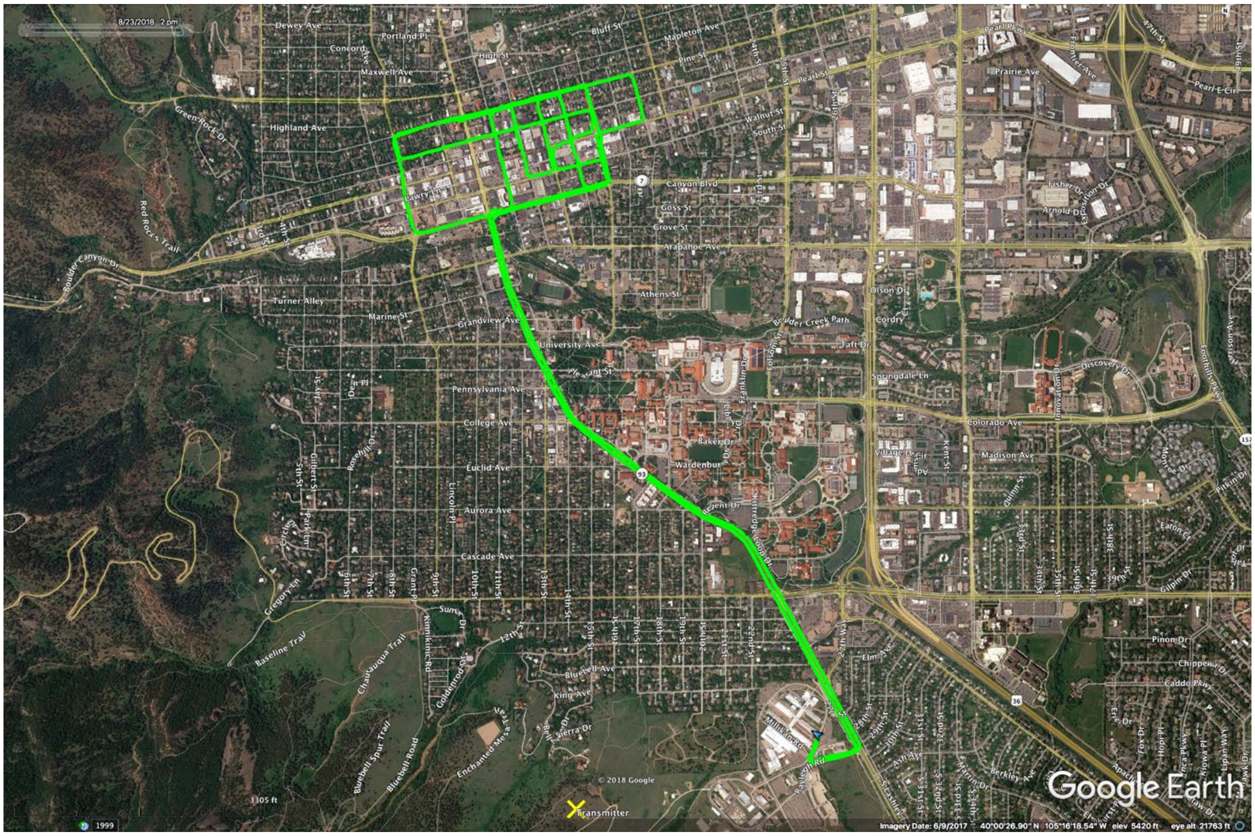


Figure 8. Urban Downtown Boulder drive route with the transmit antenna location shown as a yellow X.

1. Start at Boulder Labs
2. Head toward Broadway St.
3. Turn left at Broadway St.
4. Turn right at Canyon Blvd.
5. Turn left at 17th St.
6. Turn left on Pine St.
7. Turn left on 9th St.
8. Turn left on Canyon Blvd.
9. Turn left on 13th St.
10. Turn right on Pine St.
11. Turn right on 14th St.
12. Turn left on Walnut St.
13. Turn left on 15th St.
14. Turn right on Pine St.
15. Turn right on 16th St.
16. Turn left on Canyon Blvd.
17. Turn left on 17th St.
18. Turn left on Walnut St.
19. Turn right on 15th St.
20. Turn right on Pearl St.

21. Turn left on 17th St.
22. Turn left on Spruce St.
23. Turn right on 9th St.
24. Turn right on Pine St.
25. Turn right on 19th St.
26. Turn right on Pearl
27. Turn left on 17th St.
28. Turn right on Canyon Blvd.
29. Turn left on Broadway St.
30. Head back to labs.

2.3.2 Residential Boulder Drive Route

A map of the drive route is provided in Figure 9, followed by turn-by-turn instructions for the drive route.

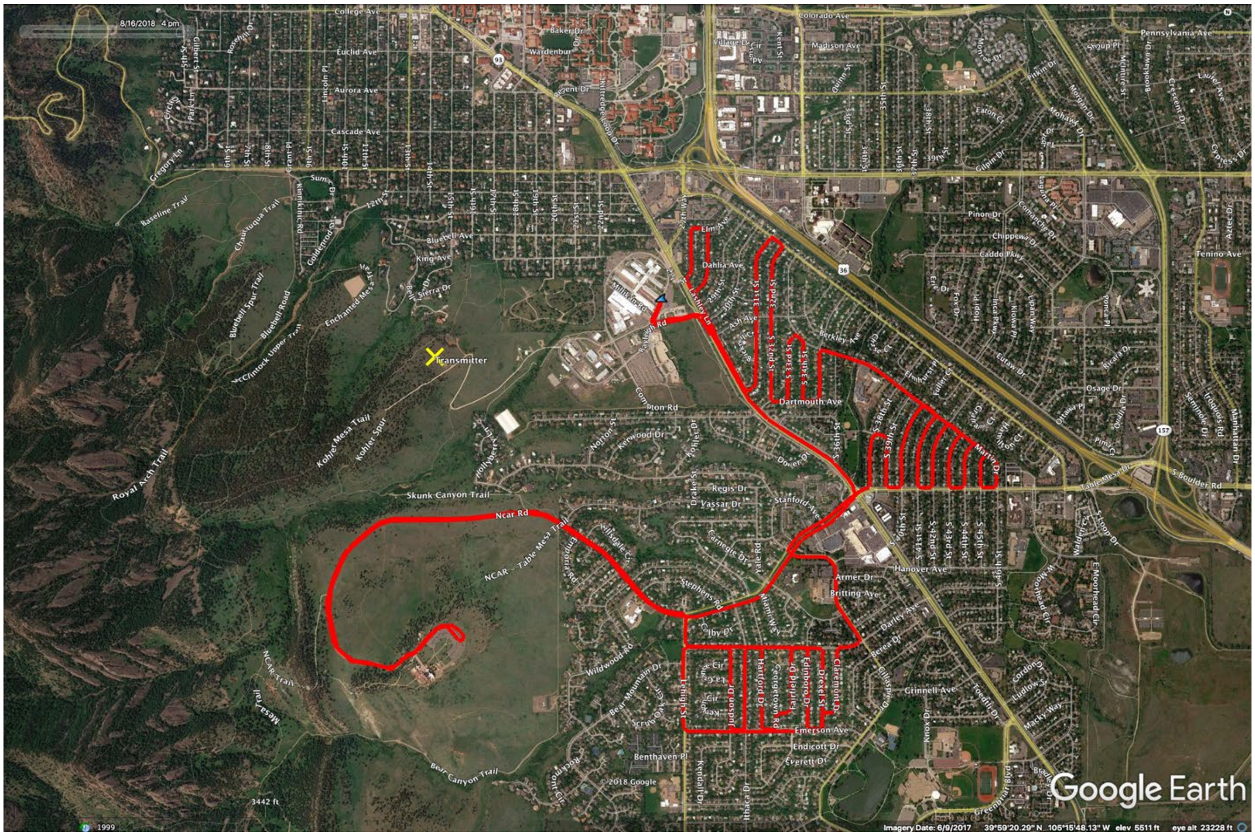


Figure 9. Residential Boulder drive route with the transmit antenna location shown as a yellow X.

1. Start at Boulder Laboratories
2. Turn right on Broadway St.
3. Take immediate left at entrance into Martin Acres Neighborhood
4. Turn left on Lashley Lane

5. Turn right on 28th St.
6. Turn left on Elm Avenue
7. Turn left on 27th St.
8. Turn left on Lashley Lane
9. Turn left on S. 31st St.
10. Turn right on Moorhead Ave.
11. Turn right on 32nd St.
12. Turn left on Dartmouth Ave.
13. Turn left on S. 33rd St.
14. Turn right on Martin Dr.
15. Turn right on S. 34th St.
16. Turn left on Dartmouth Ave.
17. Turn left on S. 35th St.
18. Turn right on Martin Dr.
19. Turn right on Table Mesa Dr.
20. Turn right on S. 45th St.
21. Turn left on Martin Dr.
22. Turn left on S. 44th St.
23. Turn right on Table Mesa Dr.
24. Turn right on S. 43rd St.
25. Turn left on Martin Dr.
26. Turn left on S. 42nd St.
27. Turn right on Table Mesa Dr.
28. Turn right on S. 41st St.
29. Turn left on Martin Dr.
30. Turn left on S. 40th St.
31. Turn right on Table Mesa Dr.
32. Turn right on S. 39th St.
33. Turn left on Dartmouth Ave.
34. Turn left on S. 38th St.
35. Turn right on Table Mesa Dr., head through intersection
36. Turn left on Gillespie Dr.
37. Turn right on Darley Ave.
38. Turn left on Claremont Dr.
39. Turn left on Drexel St.
40. Turn right on Emerson Ave.
41. Turn right on Edinboro Dr.
42. Turn left on Darley Ave.
43. Turn left on Fairfield Dr.
44. Turn left on Georgetown Rd.
45. Turn right on Emerson Ave.
46. Turn right on Hartford Dr.
47. Turn left on Darley Ave.
48. Turn left on Ithaca Dr.
49. Turn right on Emerson Ave.
50. Turn right on Judson Dr.

51. Turn left on Darley Ave.
52. Turn left on Lehigh St.
53. Turn left on Emerson Ave.
54. Turn left on Drexel St.
55. Turn left on Darley Ave.
56. Turn right on Lehigh St.
57. Turn left on Table Mesa Dr.
58. Head to NCAR
59. Turn around in parking lot at NCAR at east edge
60. Head back down Table Mesa Dr.
61. Turn left on Broadway St.
62. Head back to Boulder Laboratories

3. RESULTS

The results are divided into three sections. Section 3.1 presents the data for each drive test individually and Section 3.2 compares the control and data channel results taken over the same drive route with the same bit rate. Section 3.3 provides a summary of the results by presenting statistics for BER, measured path loss, and calculated free-space path loss for each measurement test case. Throughout this section, to facilitate comparisons, control channel data is plotted in red while the data channel is plotted in blue with the exception of the heat maps.

Section 3.1 data were collected at the same frequency and bit rate during a single drive test. While that data is displayed as distance traveled, the data is also time aligned via PXA and GRC GPS time information. Section 3.2 data were collected at the two different frequencies at the same bit rate over two different drive tests. The data is displayed as distance traveled via PXA and GRC GPS position information without the expectation of time alignment. The only deviations from distance traveled occurred during the residential routes at approximately 18 km. At this point, the measurement vehicle was turned around in a large parking lot and the same exact route through the parking lot was not always taken. In some instances a slight offset in the comparison plots in Section 3.2 can be seen after this point.

3.1 Individual Drive Test Results

Figure 10 provides the results of a data channel drive test through residential Boulder while transmitting at a bit rate of 100 kbps. Propagation path loss is shown in the upper plot as a solid line and BER is shown in the lower plot. The calculated free space path loss is shown as a dashed line. The maximum measurable path loss from Table 7 is displayed as a black line near the top of the upper plot.

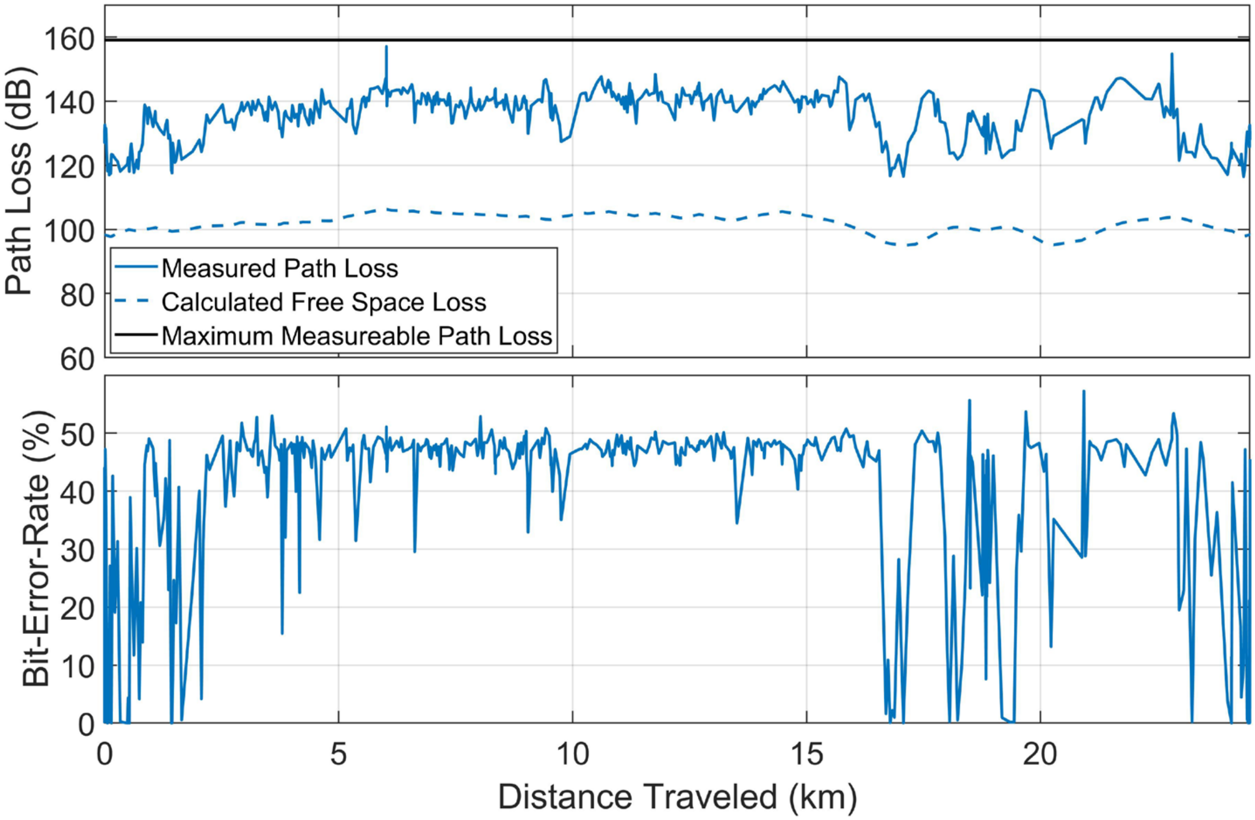


Figure 10. Path loss and BER for the data channel (1702.5 MHz) collected during the residential drive route with bit rate = 100 kbps.

Figures 11 and 12 provide heat maps of the path loss and BER data for the data shown in Figure 10. Brighter colors indicate higher values while the deeper red colors indicate lower values. The deep red line south of the transmitter location (indicated by a yellow X) is a part of the drive route with clear line of sight therefore both path loss and BER were relatively low through this part of the route.

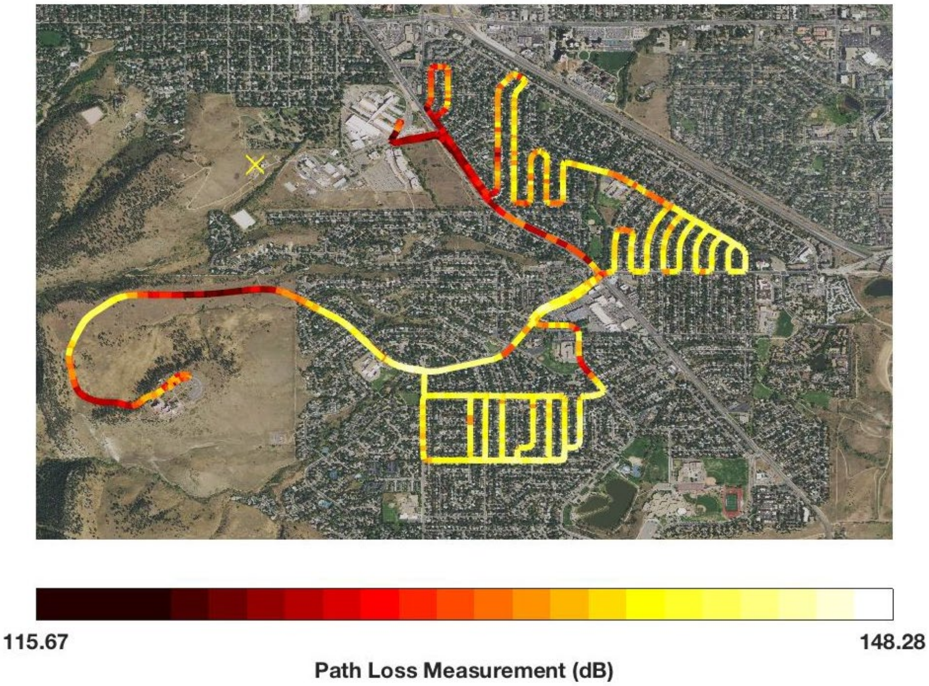


Figure 11. Heat map displaying path loss on the data channel (1702.5 MHz) through the residential drive route with bit rate = 100 kbps. The yellow X indicates the transmitter location.

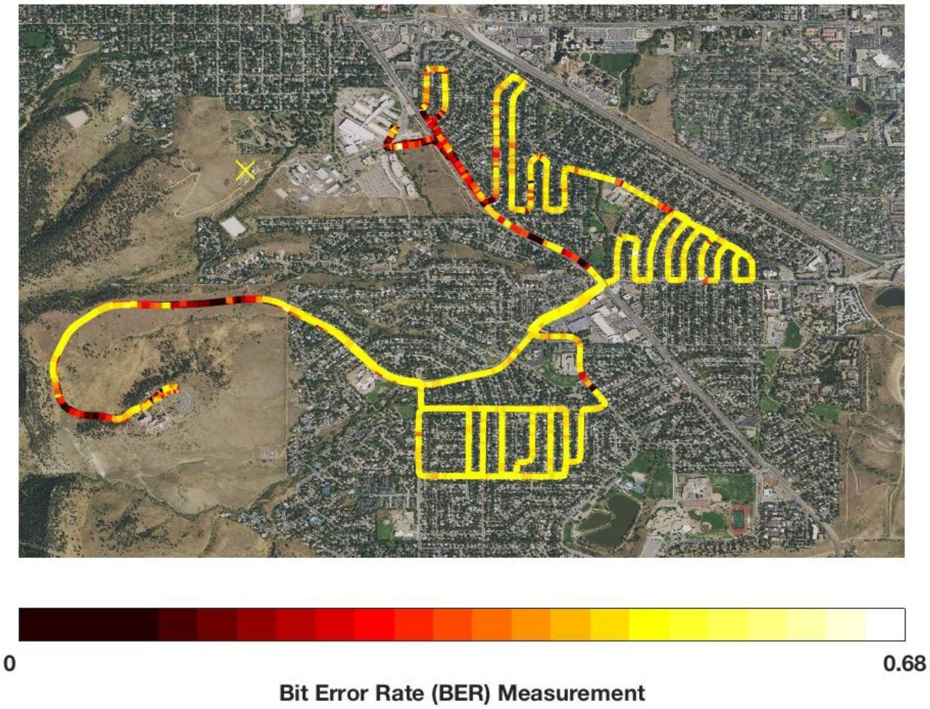


Figure 12. Heat map displaying BER on the data channel (1702.5 MHz) through the residential drive route with bit rate = 100 kbps. The yellow X indicates the transmitter location.

Figure 13 shows the results of a control channel drive test through residential Boulder with a bit rate of 100 kbps. It should be noted that the measured path loss falls far below the maximum measurable path loss line for the entire set of control channel measurements. This is due to improved propagation characteristics at the lower frequency.

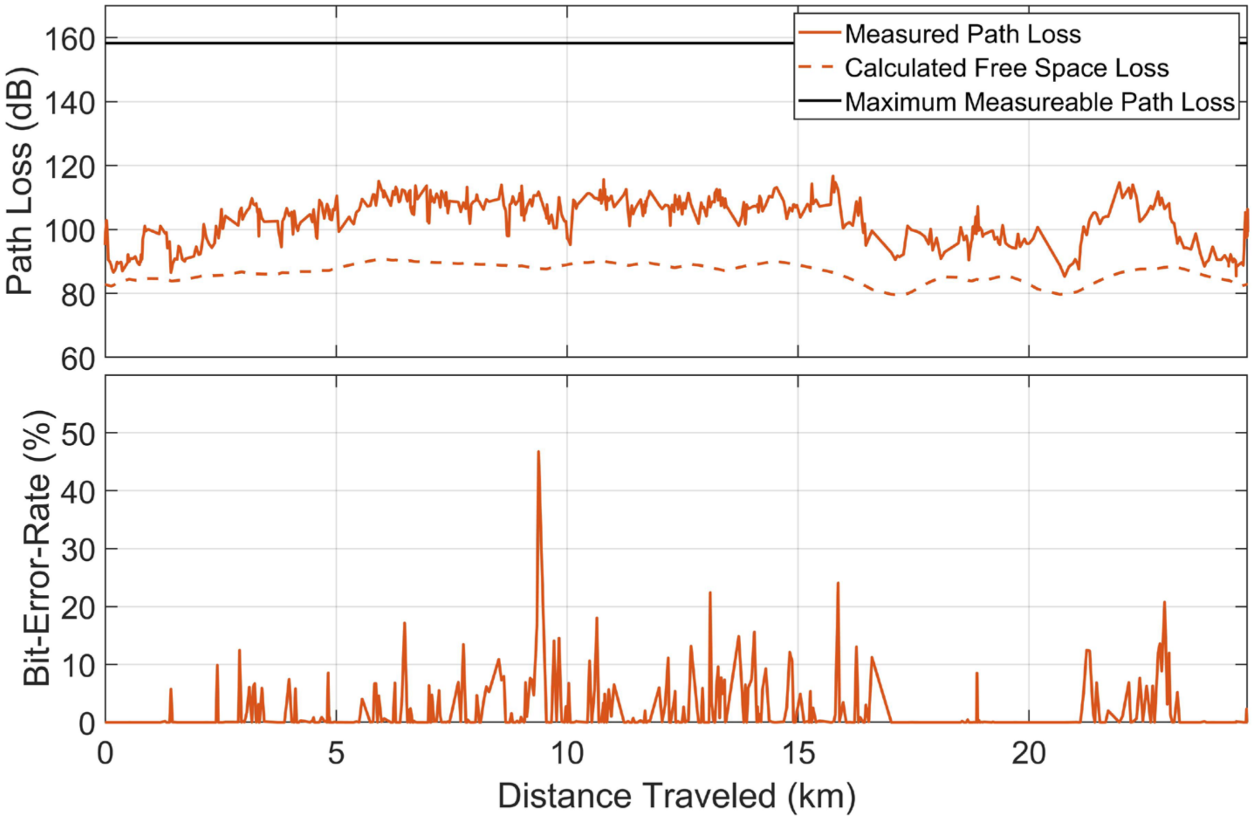


Figure 13. Path loss and BER for the control channel (286.375 MHz) collected during the residential drive route with bit rate = 100 kbps.

Figures 14 and 15 show heat maps for the path loss and BER data for the control channel along the residential drive route with a bit rate of 100 kbps.

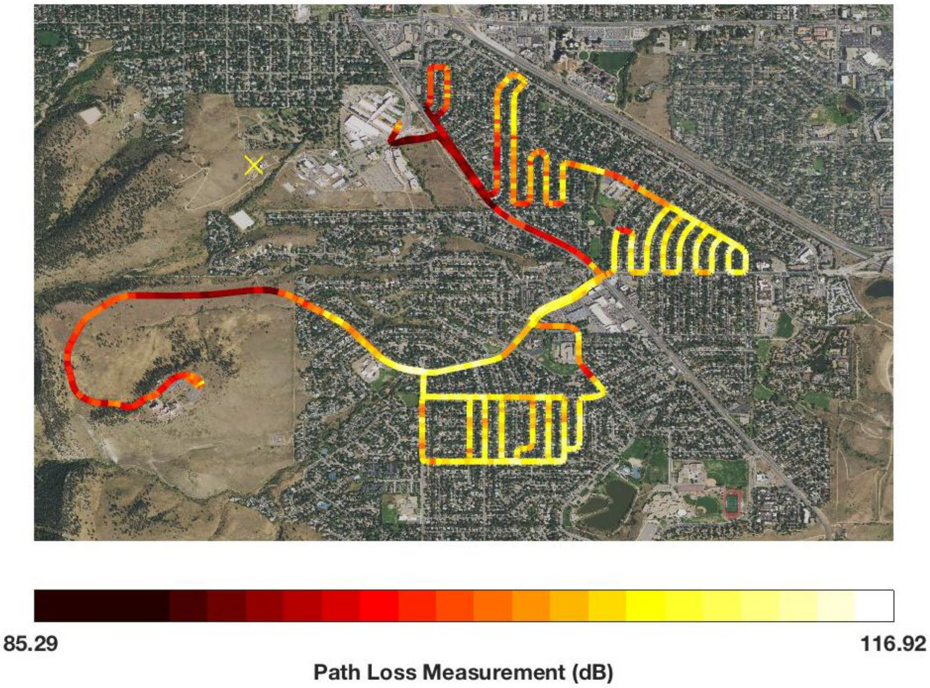


Figure 14. Heat map displaying path loss on the control channel (286.375 MHz) through the residential drive route with bit rate = 100 kbps. The yellow X indicates the transmitter location.

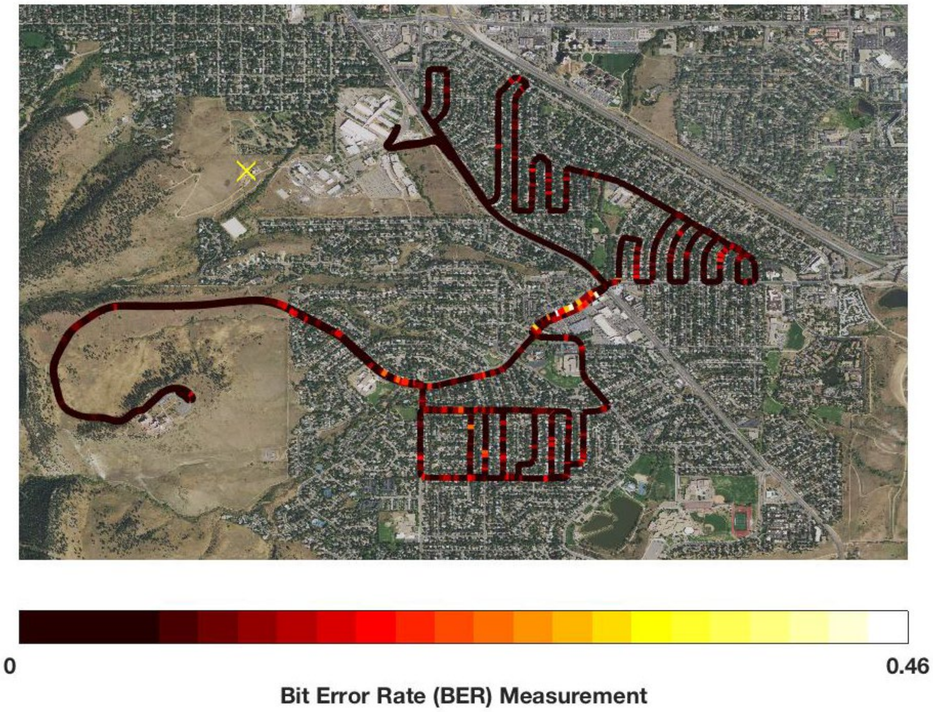


Figure 15. Heat map displaying BER on the control channel (286.375 MHz) through the residential drive route with bit rate = 100 kbps. The yellow X indicates the transmitter location.

Figure 16 provides the results of a data channel drive test through urban Boulder, with a bit rate of 100 kbps.

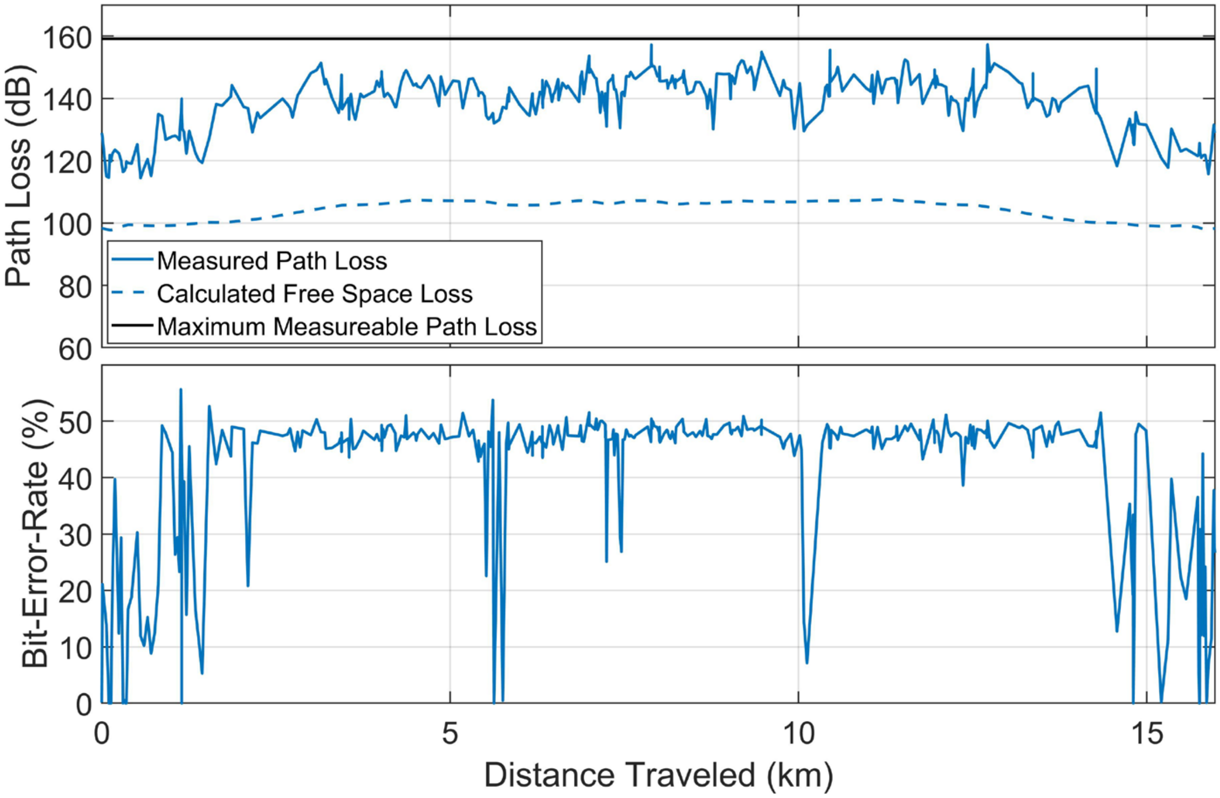


Figure 16. Path loss and BER for the data channel (1702.5 MHz) collected during the urban drive route with bit rate = 100 kbps.

Figures 17 and 18 provide heat maps of the path loss and BER data for the data channel along the urban drive route with a bit rate of 100 kbps.

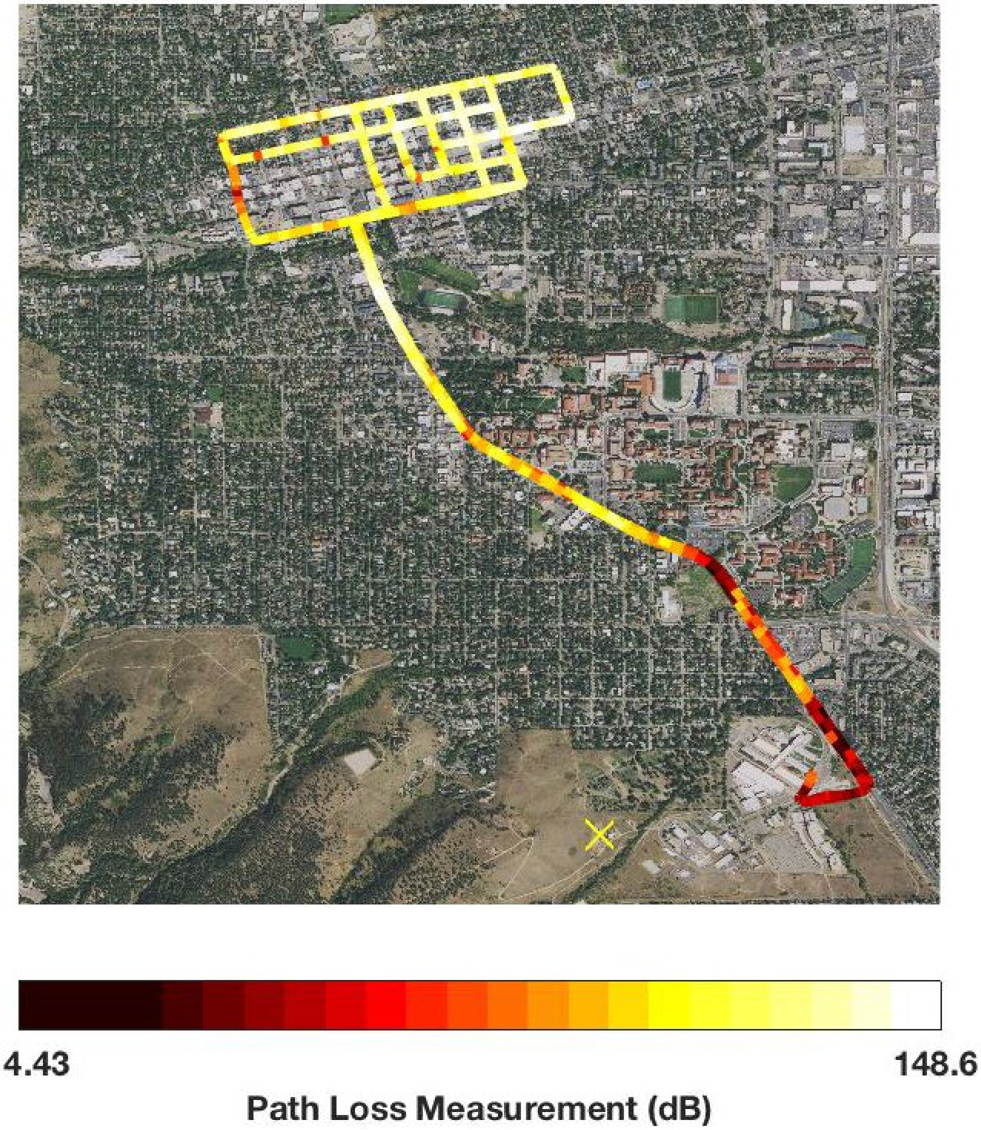


Figure 17. Heat map displaying path loss on the data channel (1702.5 MHz) through the urban drive route with bit rate = 100 kbps. The yellow X indicates the transmitter location.

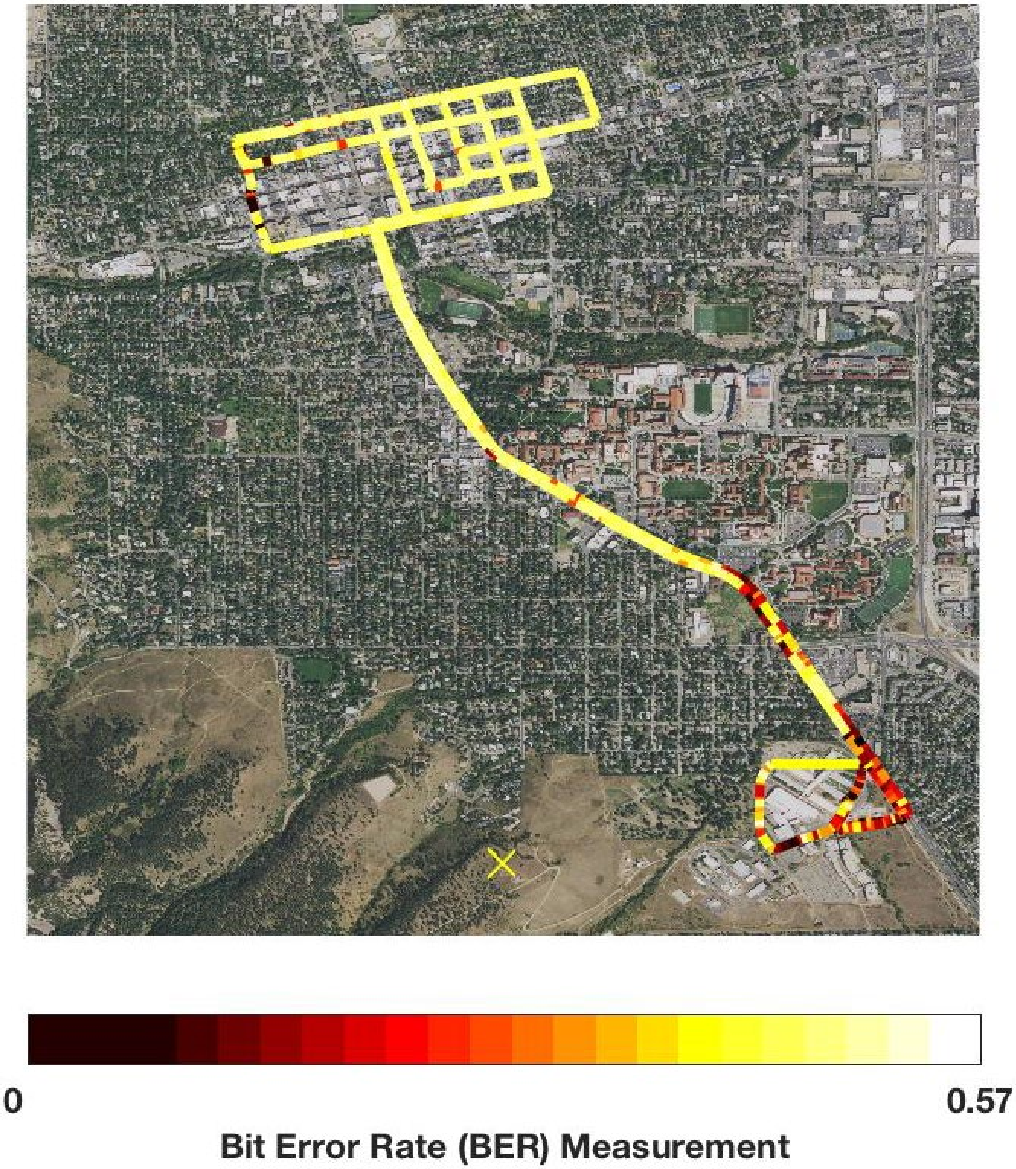


Figure 18. Heat map displaying BER on the data channel (1702.5 MHz) through the urban drive route with bit rate = 100 kbps. The yellow X indicates the transmitter location.

Figure 19 provides the results of a control channel drive test through urban Boulder with a bit rate of 100 kbps.

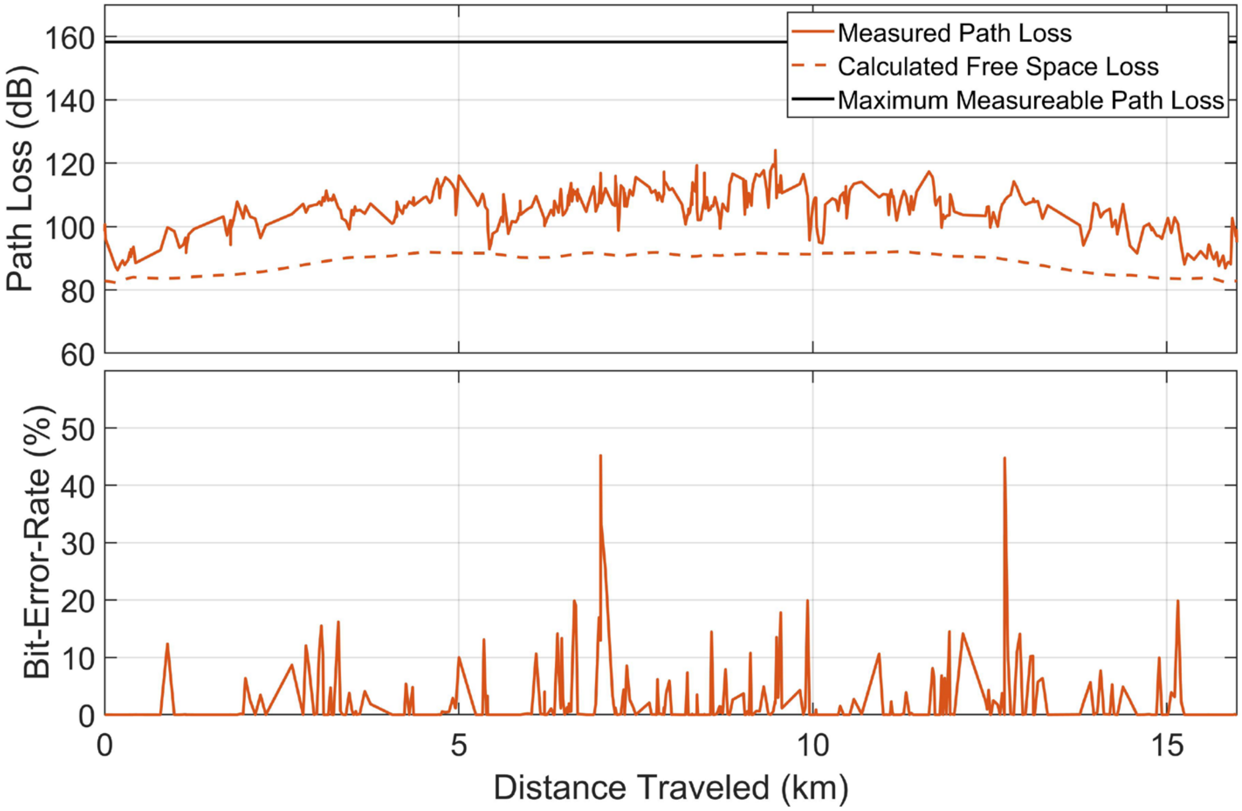


Figure 19. Path loss and BER for the control channel (286.375 MHz) collected during the urban drive route with bit rate = 100 kbps.

Figures 20 and 21 provide heat maps of the path loss and BER data for the control channel along the urban drive route with a bit rate of 100 kbps.

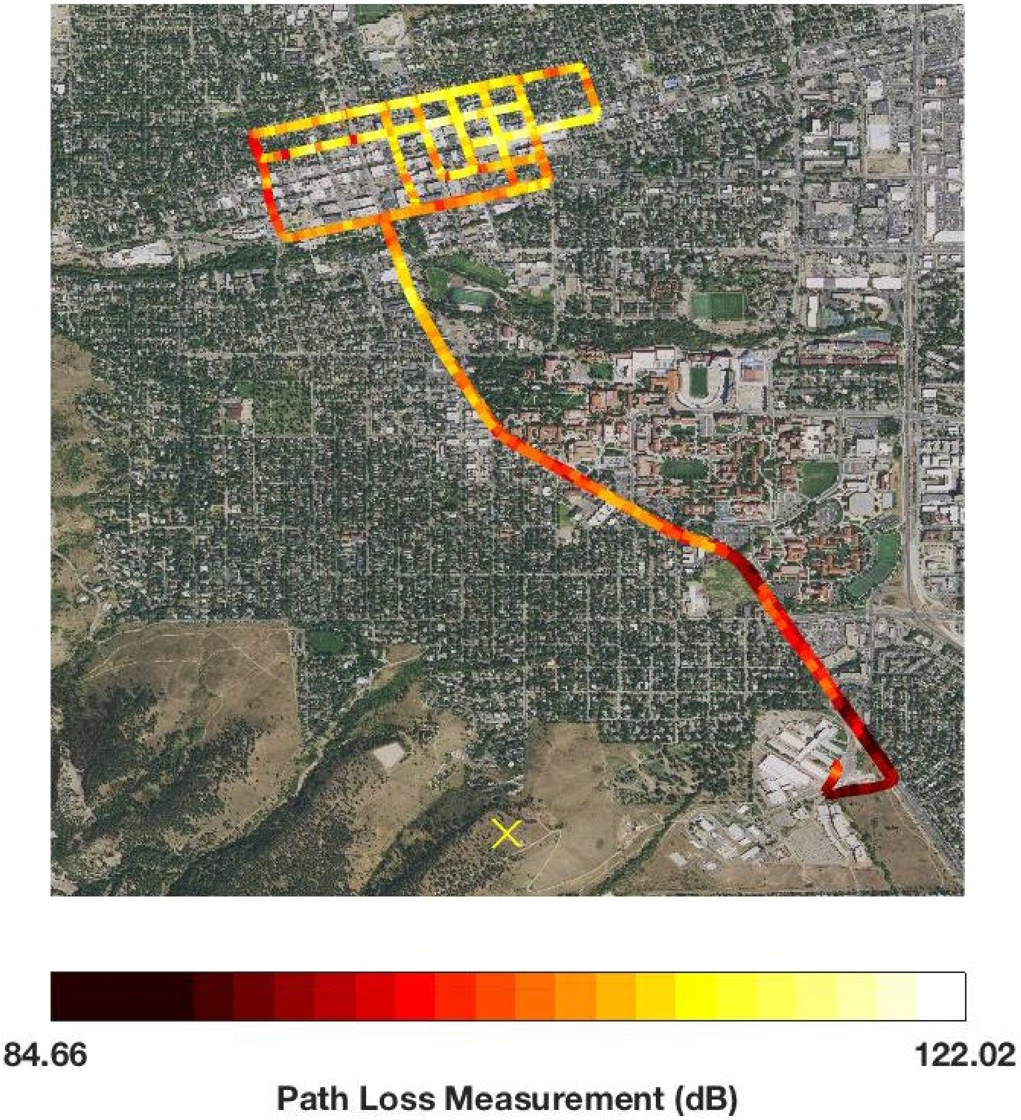


Figure 20. Heat map displaying path loss on the control channel (286.375 MHz) through the urban drive route with bit rate = 100 kbps. The yellow X indicates the transmitter location.

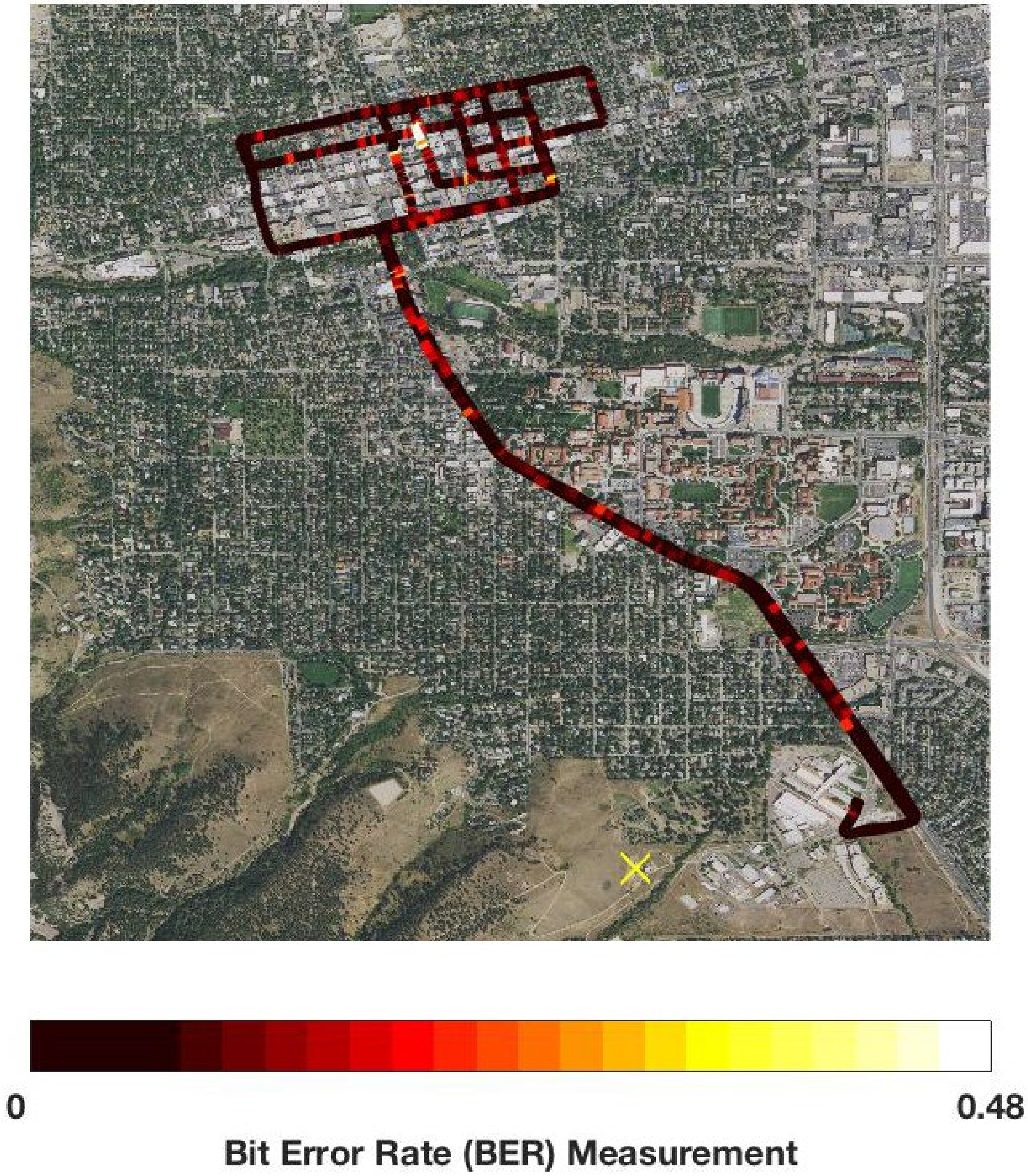


Figure 21. Heat map displaying BER on the control channel (286.375 MHz) through the urban drive route with bit rate = 100 kbps. The yellow X indicates the transmitter location.

Figure 22 provides the results of a data channel drive test through residential Boulder with a bit rate of 1 Mbps.

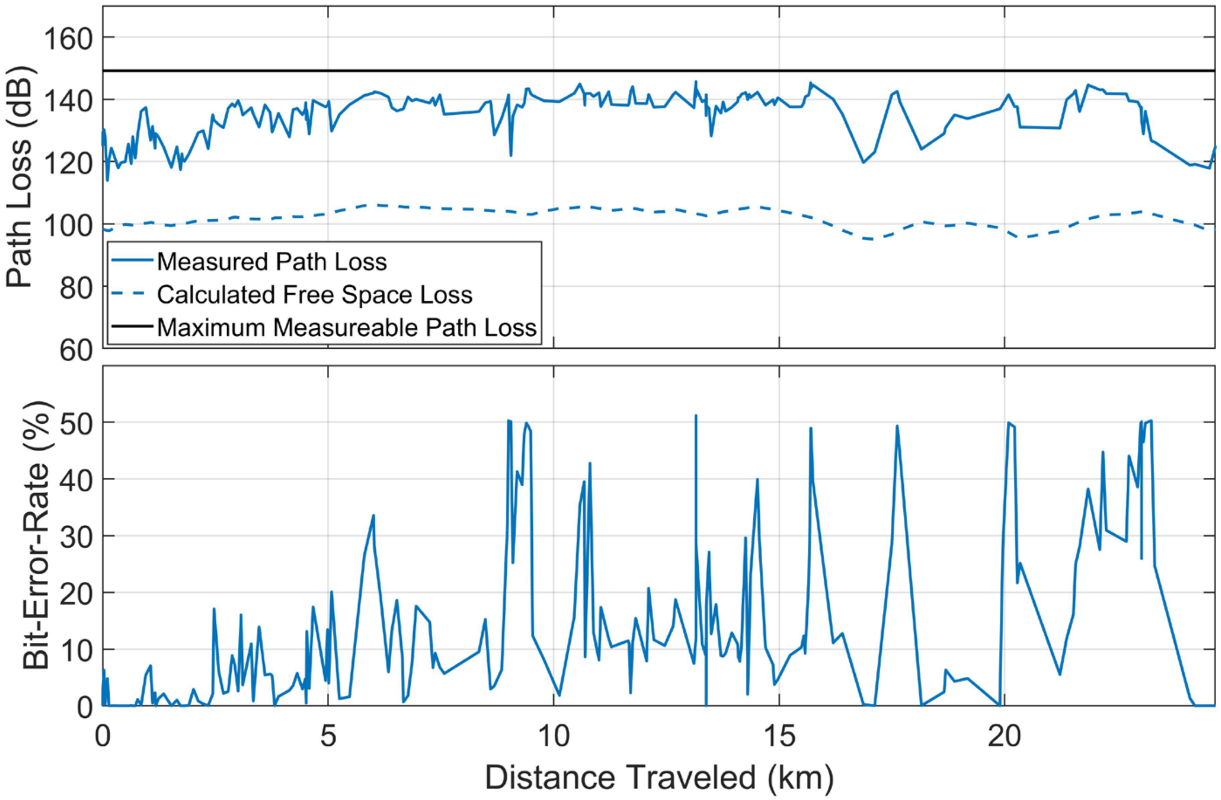


Figure 22. Path loss and BER for the data channel (1702.5 MHz) collected during the residential drive route with bit rate = 1 Mbps.

Figures 23 and 24 provide heat maps of the path loss and BER data for the data channel along the residential drive route with a bit rate of 100 kbps.

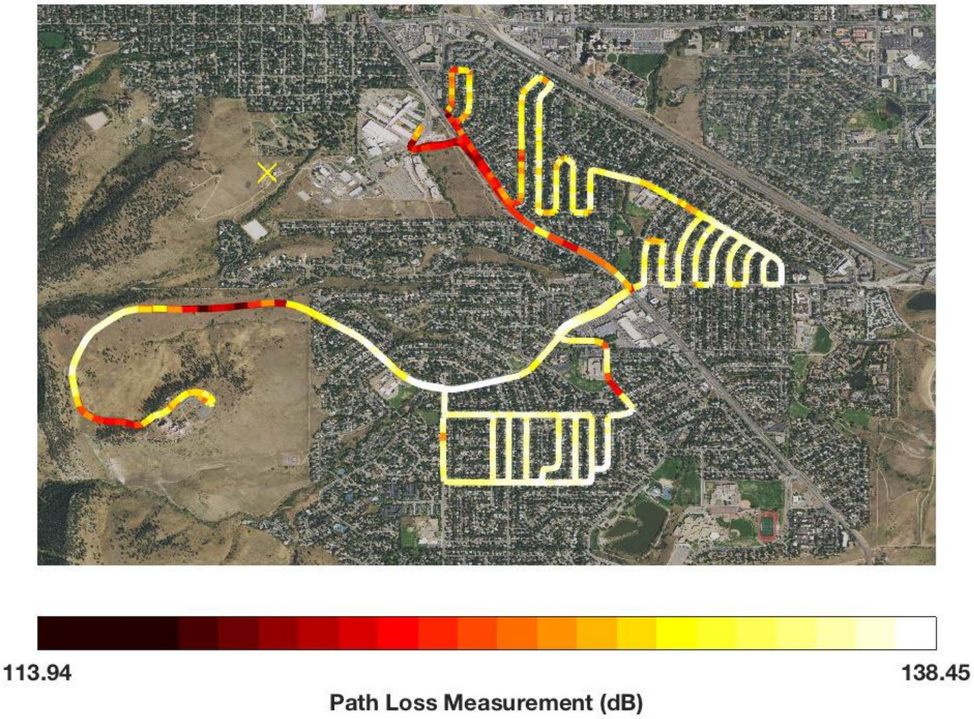


Figure 23. Heat map displaying path loss on the data channel (1702.5 MHz) through the residential drive route with bit rate = 1 Mbps. The yellow X indicates the transmitter location.

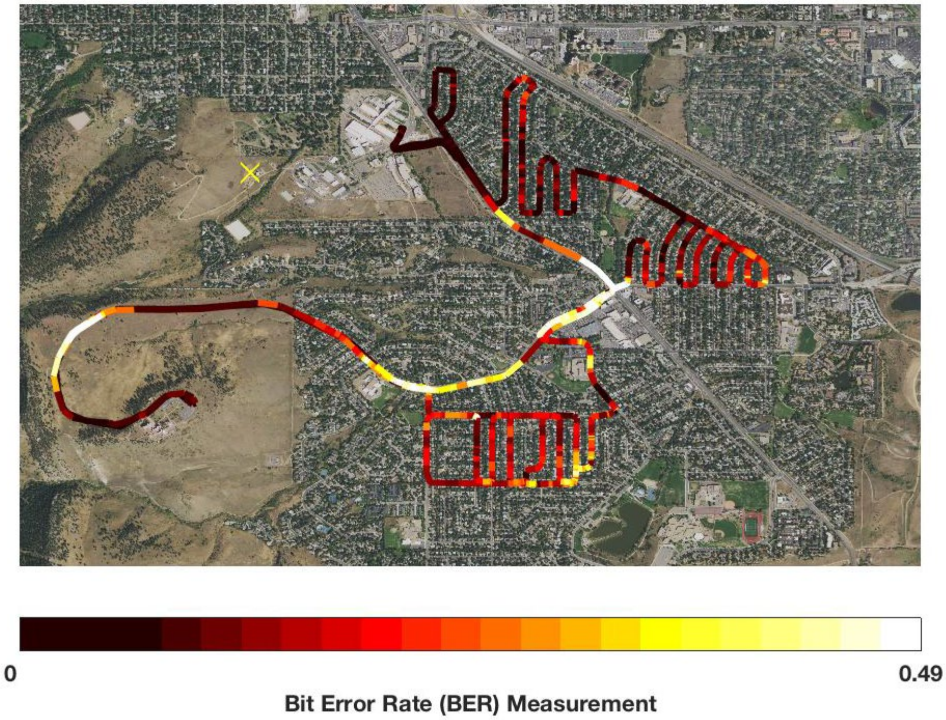


Figure 24. Heat map displaying BER on the data channel (1702.5 MHz) through the residential drive route with bit rate = 1 Mbps. The yellow X indicates the transmitter location.

Figure 25 provides the results of a control channel drive test through residential Boulder with a bit rate of 1 Mbps.

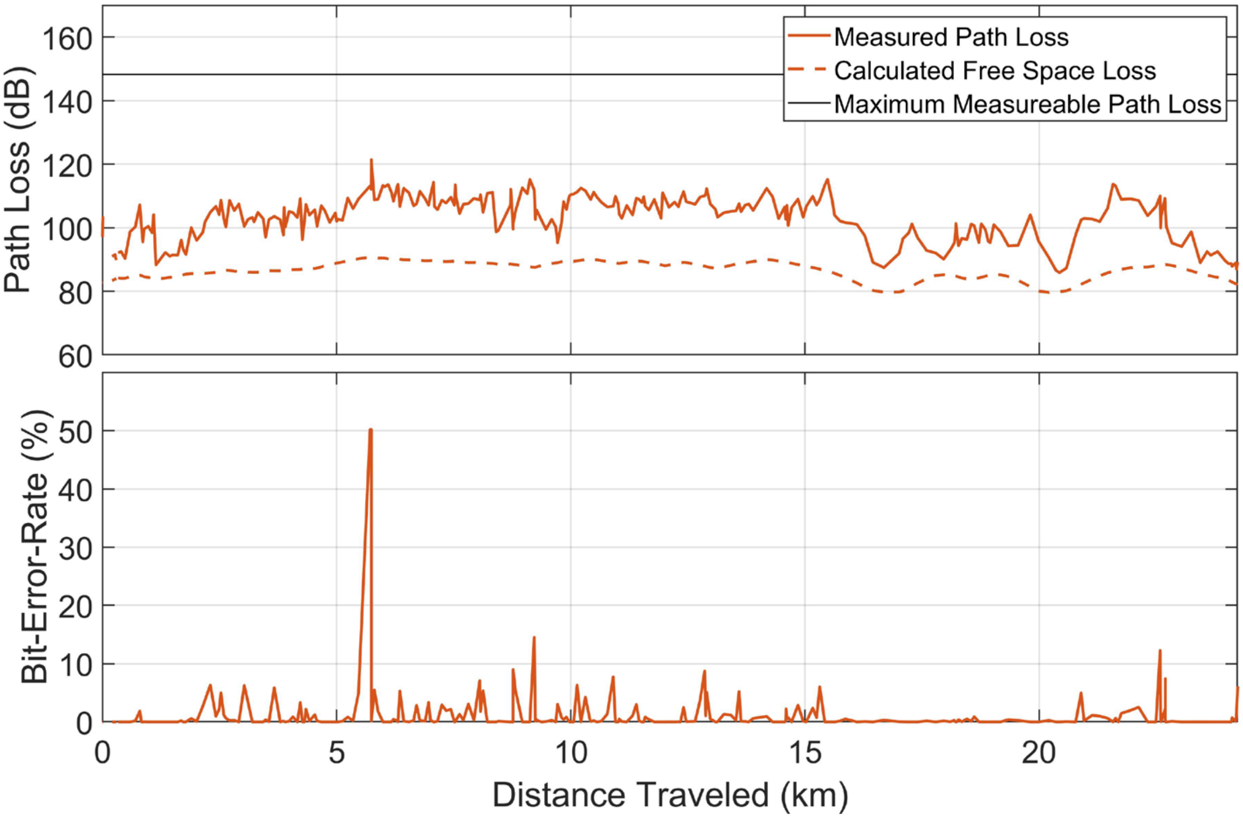


Figure 25. Path loss and BER for the control channel (286.375 MHz) collected during the residential drive route with bit rate = 1 Mbps.

Figures 26 and 27 provide heat maps of the path loss and BER data for the control channel along the residential drive route with a bit rate of 1 Mbps.

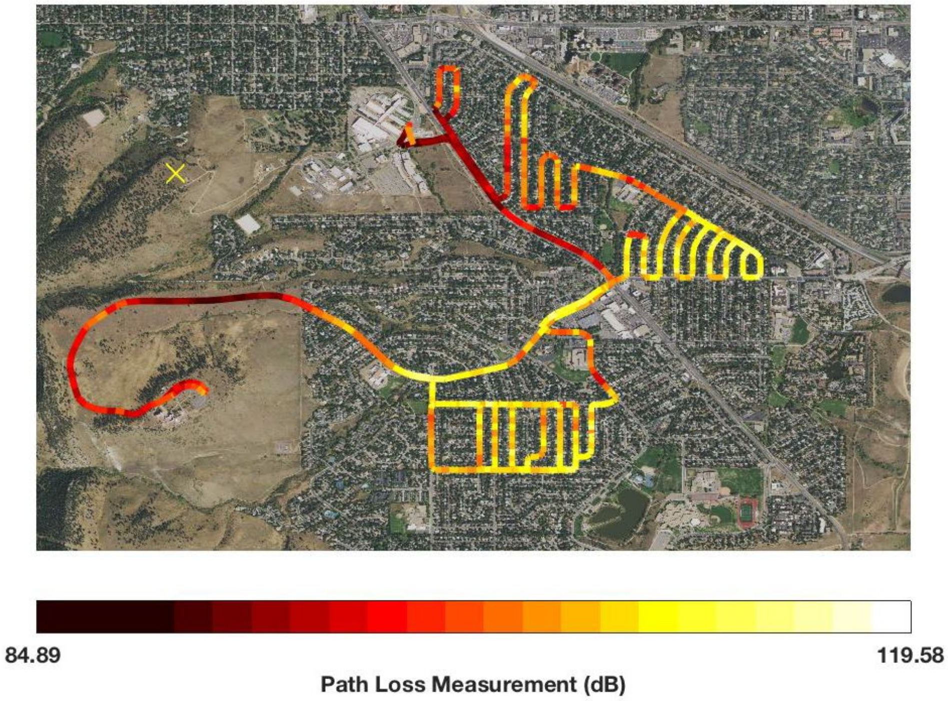


Figure 26. Heat map displaying path loss on the control channel (286.375 MHz) through the residential drive route with bit rate = 1 Mbps. The yellow X indicates the transmitter location.

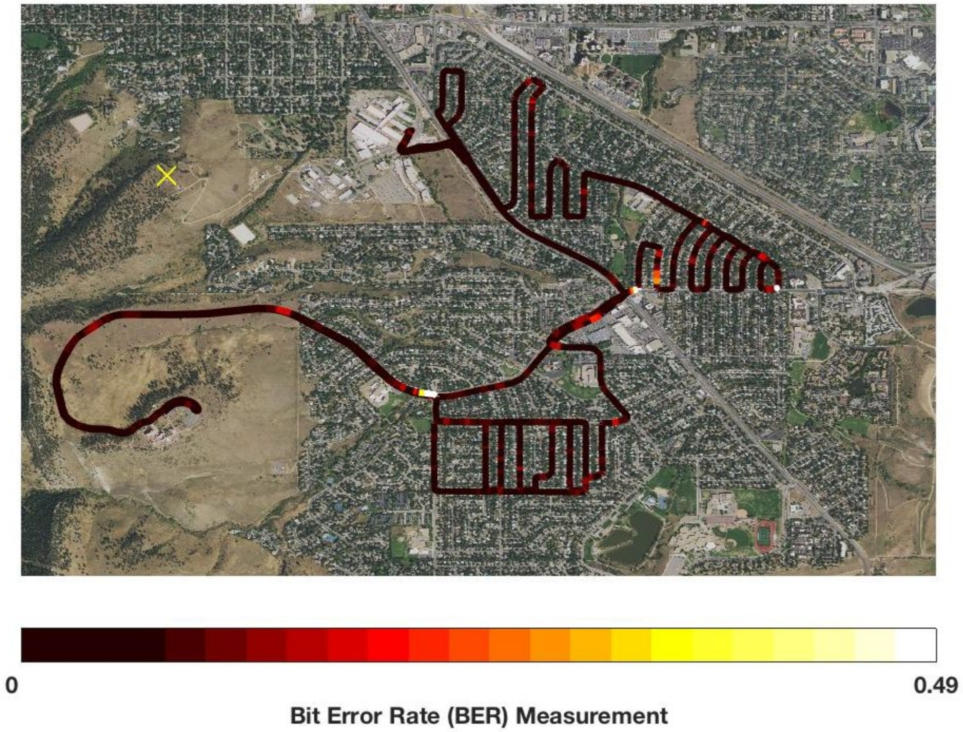


Figure 27. Heat map displaying BER on the control channel (286.375 MHz) through the residential drive route with bit rate = 1 Mbps. The yellow X indicates the transmitter location.

Figure 28 provides the results of a data channel drive test through urban Boulder with a bit rate of 1 Mbps.

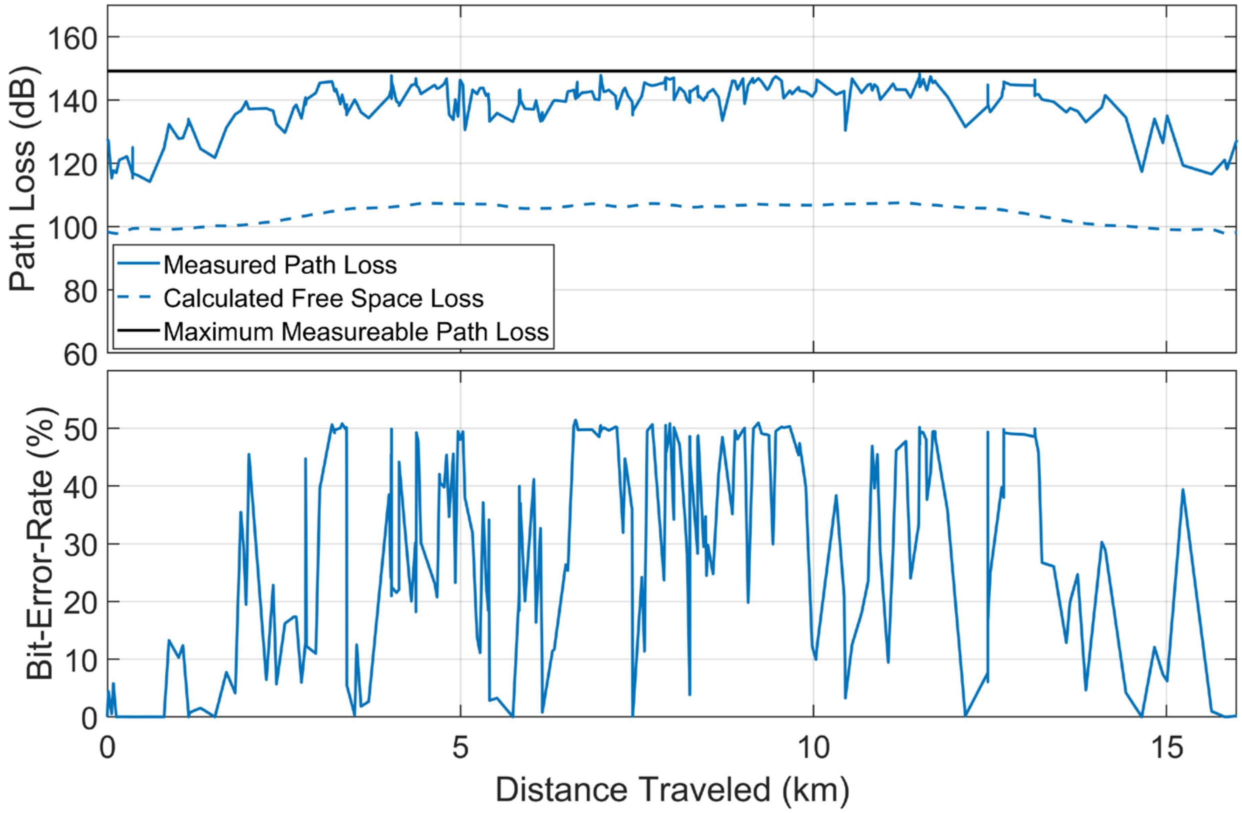


Figure 28. Path loss and BER for the data channel (1702.5 MHz) collected during the urban drive route with bit rate = 1 Mbps.

Figures 29 and 30 provide heat maps of the path loss and BER data for the data channel along the urban drive route with a bit rate of 1 Mbps.

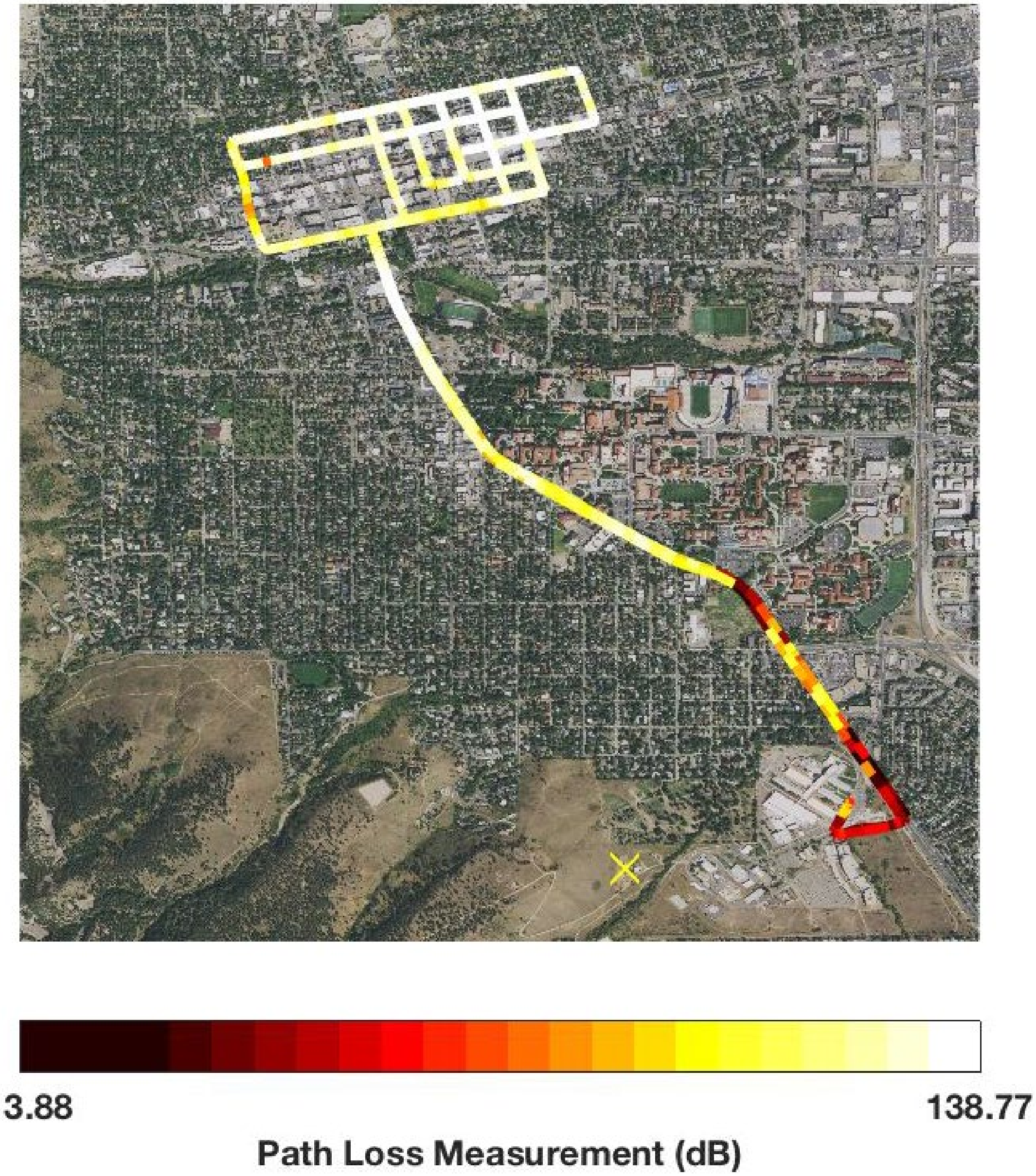


Figure 29. Heat map displaying path loss on the data channel (1702.5 MHz) through the urban drive route with bit rate = 1 Mbps. The yellow X indicates the transmitter location.

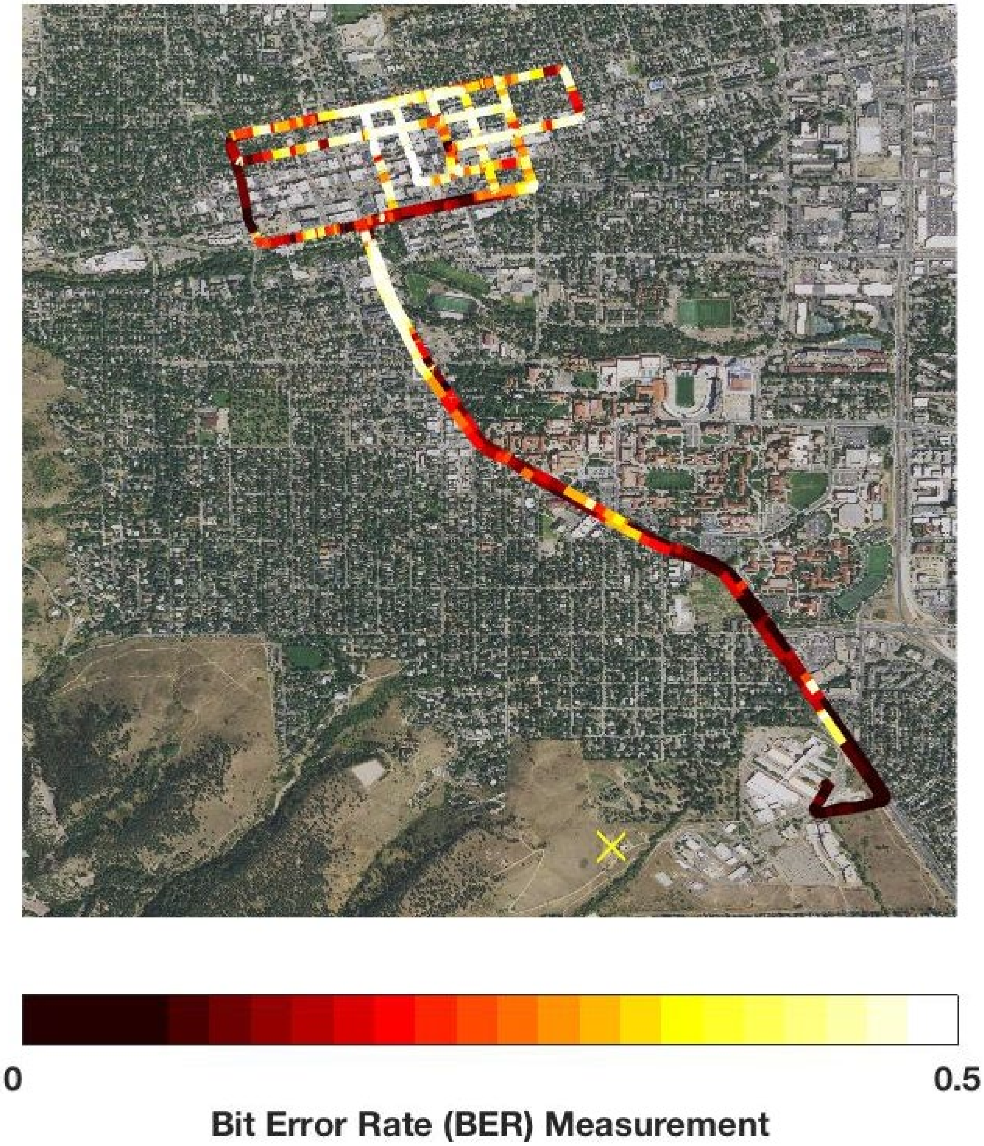


Figure 30. Heat map displaying BER on the data channel (1702.5 MHz) through the urban drive route with bit rate = 1 Mbps. The yellow X indicates the transmitter location.

Figure 31 provides the results of a control channel drive test through urban Boulder with a bit rate of 1 Mbps.

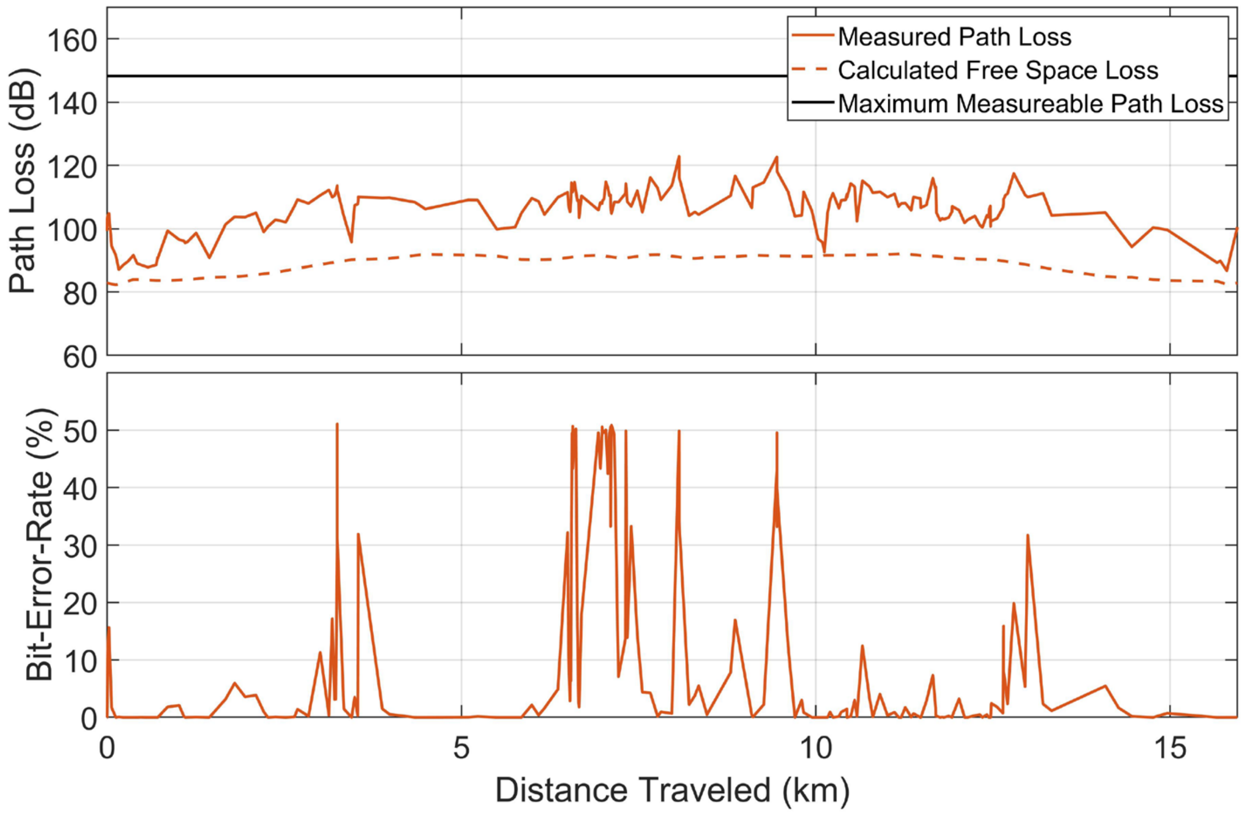


Figure 31. Path loss and BER for the control channel (286.375 MHz) collected during the urban drive route with bit rate = 1 Mbps.

Figures 32 and 33 provide heat maps of the path loss and BER data for the control channel along the urban drive route with a bit rate of 1 Mbps.

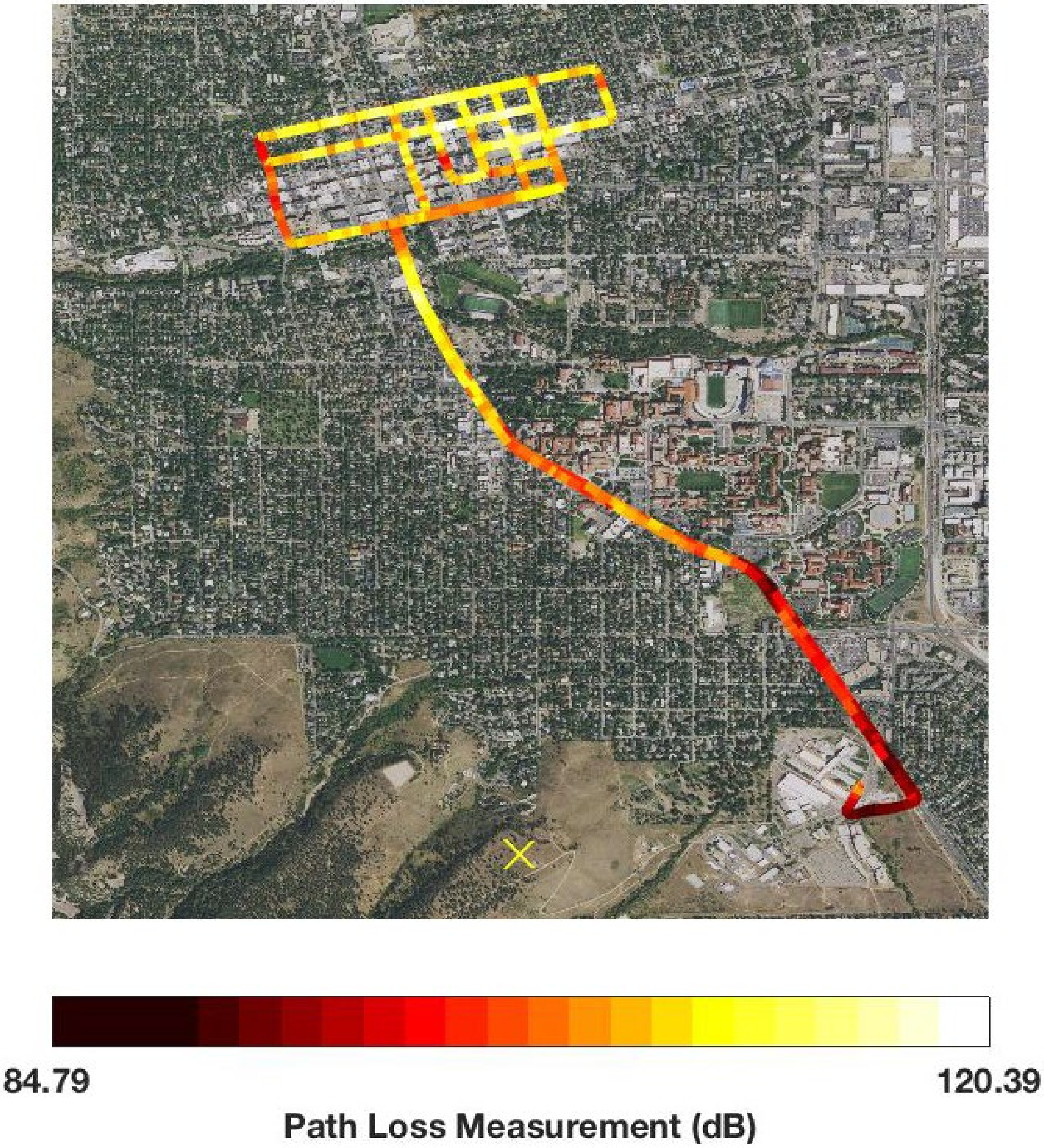


Figure 32. Heat map displaying path loss on the control channel (286.375 MHz) through the urban drive route with bit rate = 1 Mbps. The yellow X indicates the transmitter location.

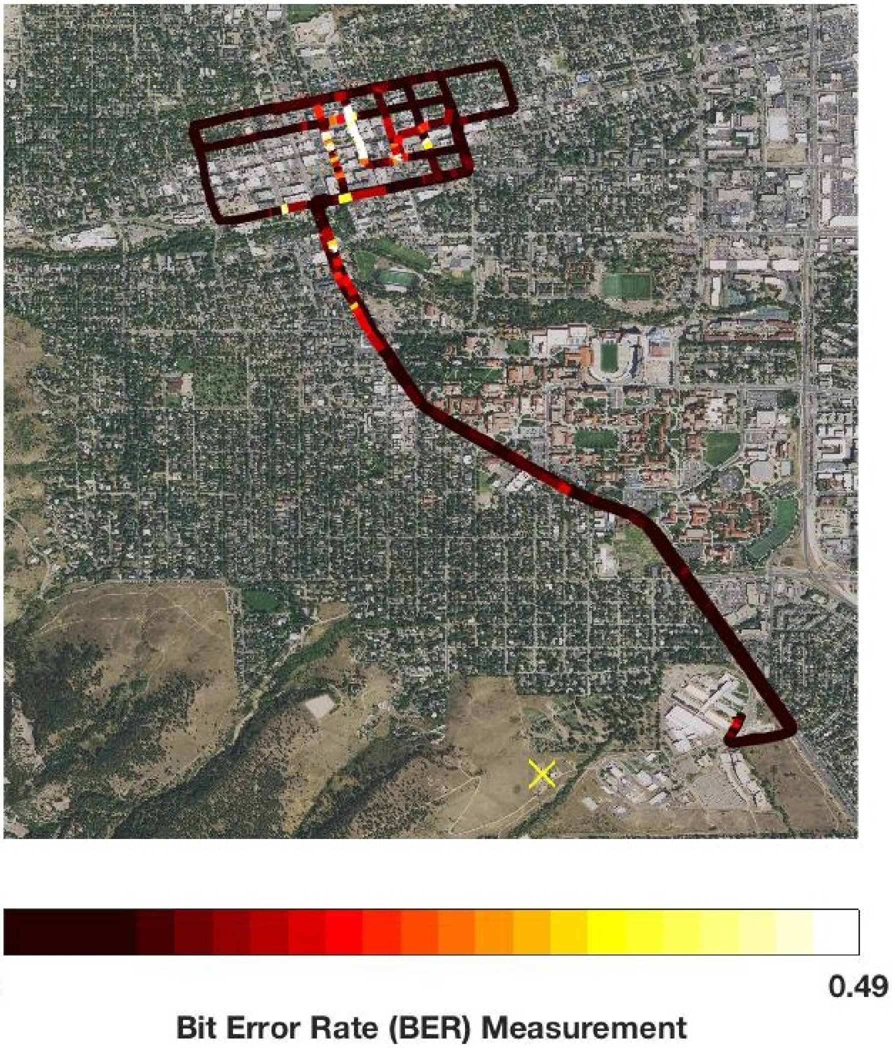


Figure 33. Heat map displaying BER on the control channel (286.375 MHz) through the urban drive route with bit rate = 1 Mbps. The yellow X indicates the transmitter location.

3.2 Comparison of the Data Channel to the Control Channel

In this section data is presented that compares the data channel to the control channel. Since the data was not collected at the same time, it was not possible to correlate points to the same position using the GPS time stamps. To correlate the data to the same position, the GPS position data was used to calculate distance traveled along the drive route and the path loss and BER data has been plotted as a function of distance as described previously. Figures 34 and 35 provide the comparison of path loss and BER for the residential drive route at a 100 kbps bit rate. Figures 36 and 37 provide the comparison of path loss and BER for the urban drive route at a 100 kbps bit rate. Figures 38 and 39 provide the comparison of path loss and BER for the residential drive route at a 1 Mbps bit rate. Figures 40 and 41 provide the comparison of path loss and BER for the urban drive route at a 1 Mbps bit rate.

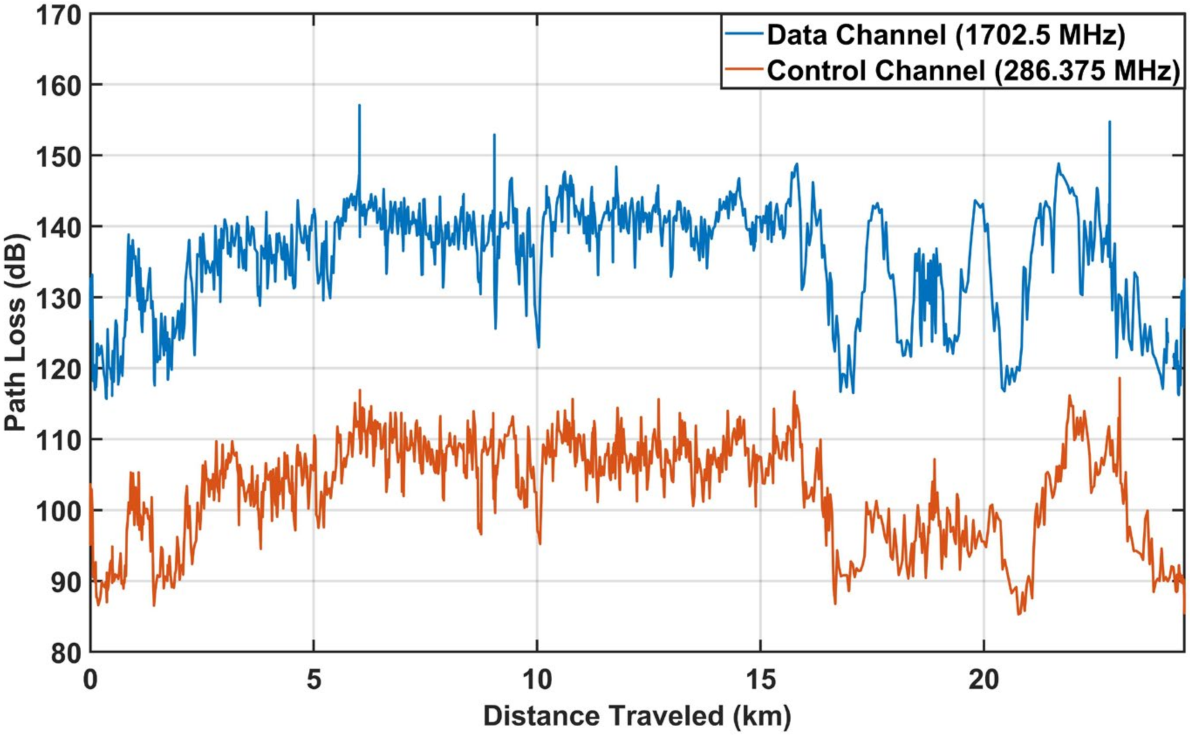


Figure 34. Comparison of path loss for the residential drive route and bit rate = 100 kbps.

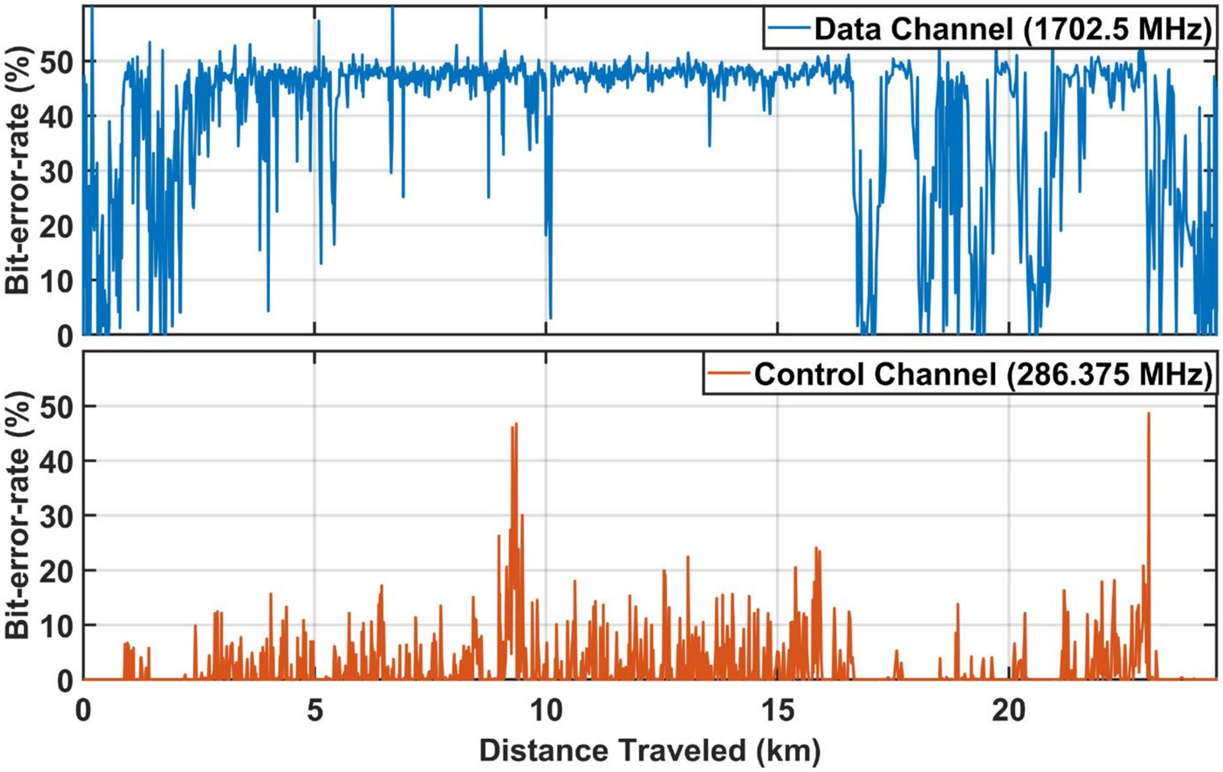


Figure 35. Comparison of BER for the residential drive route and bit rate = 100 kbps.

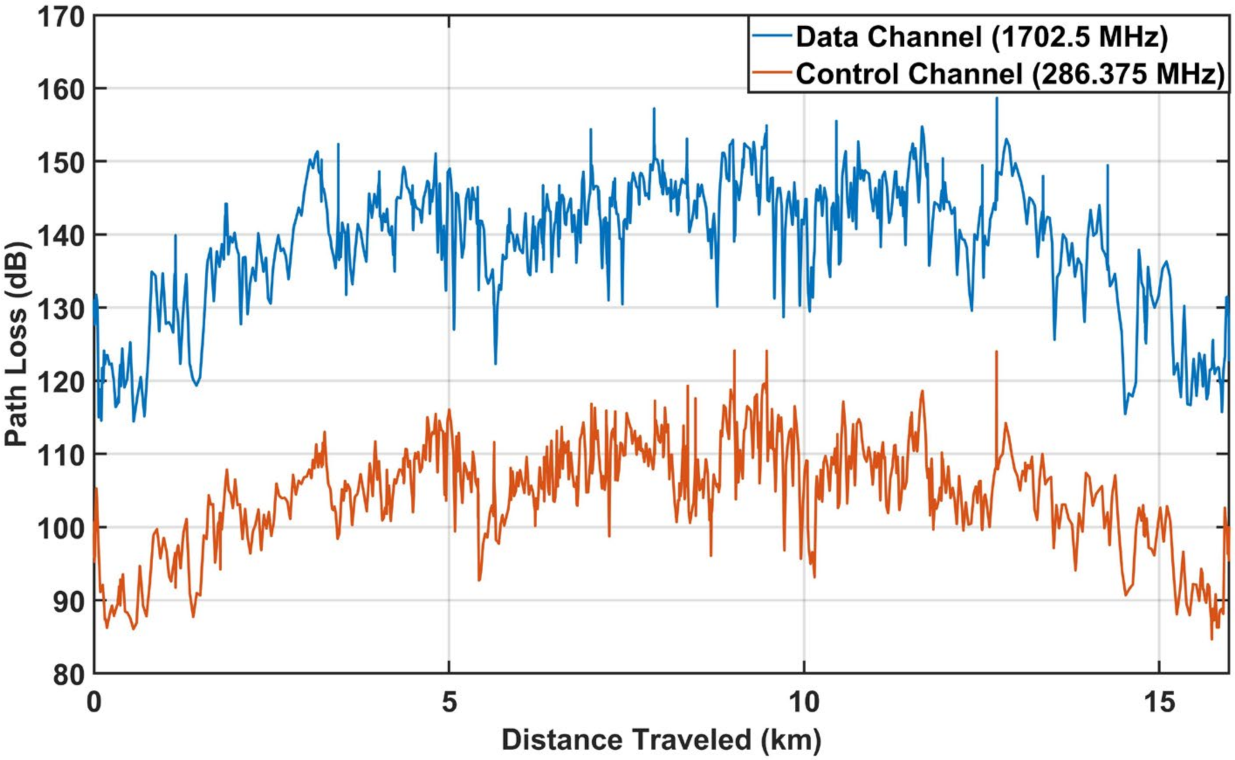


Figure 36. Comparison of path loss for the urban drive route and bit rate = 100 kbps.

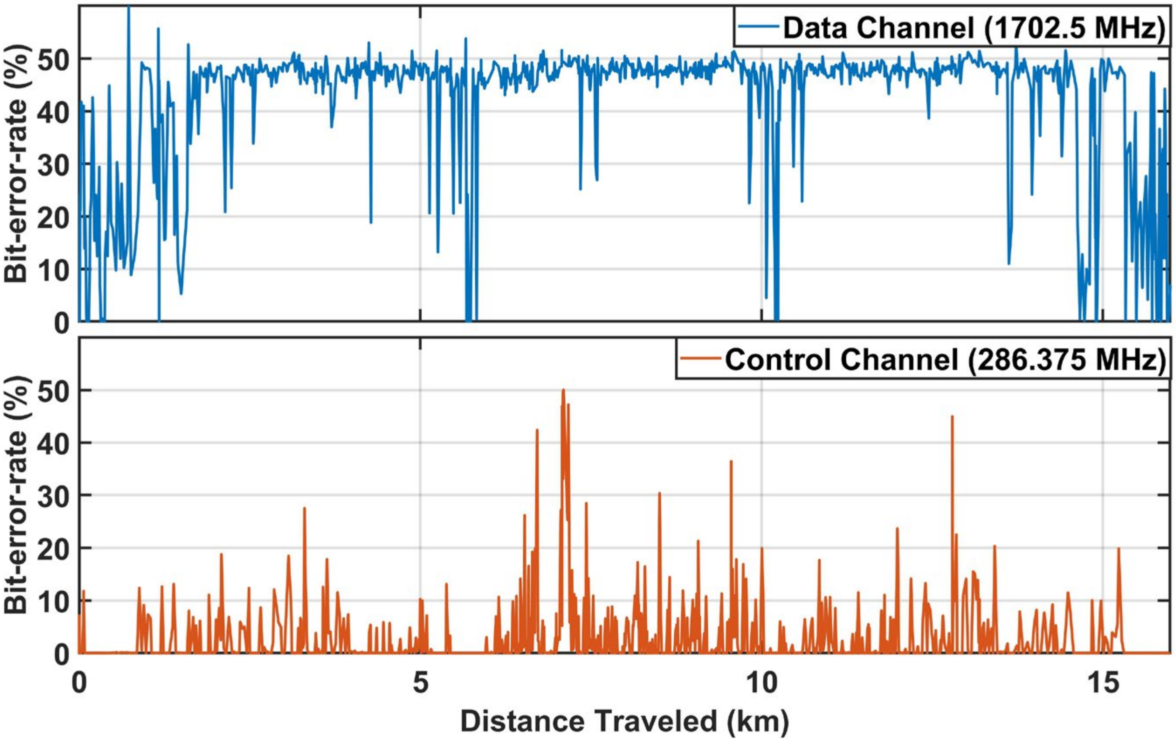


Figure 37. Comparison of BER for the urban drive route and bit rate = 100 kbps.

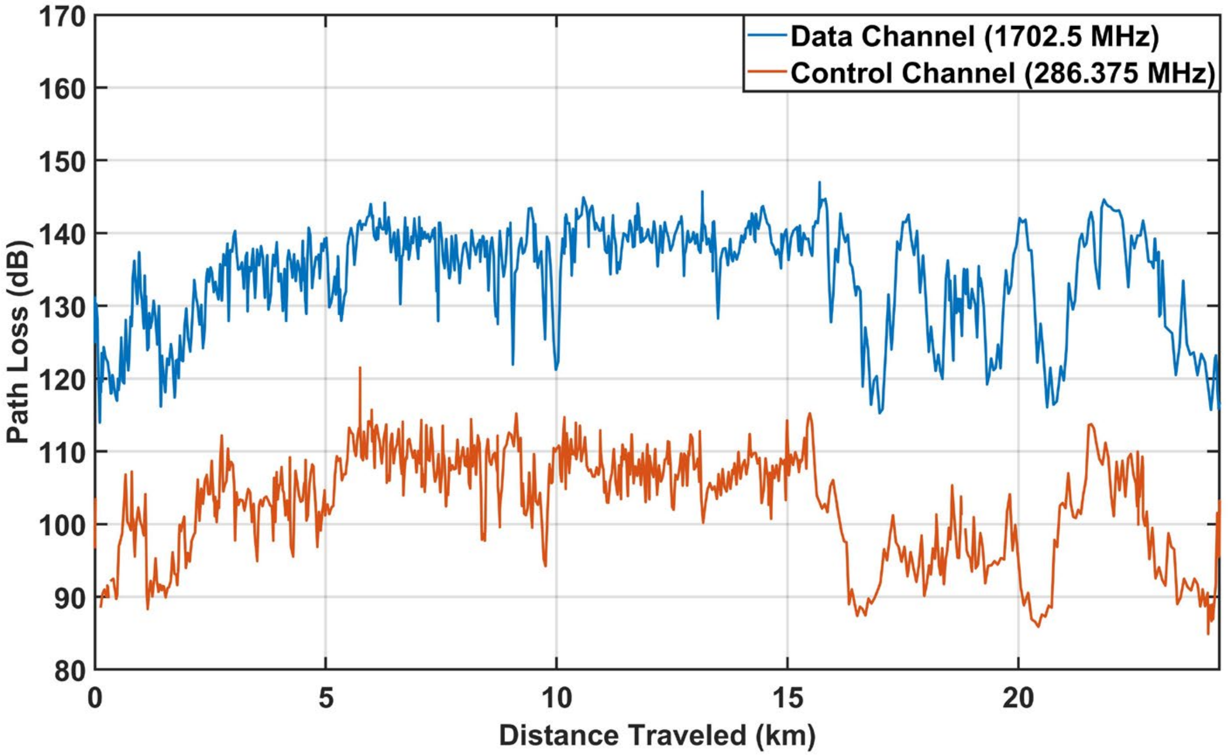


Figure 38. Comparison of path loss for the residential drive route and bit rate = 1 Mbps.

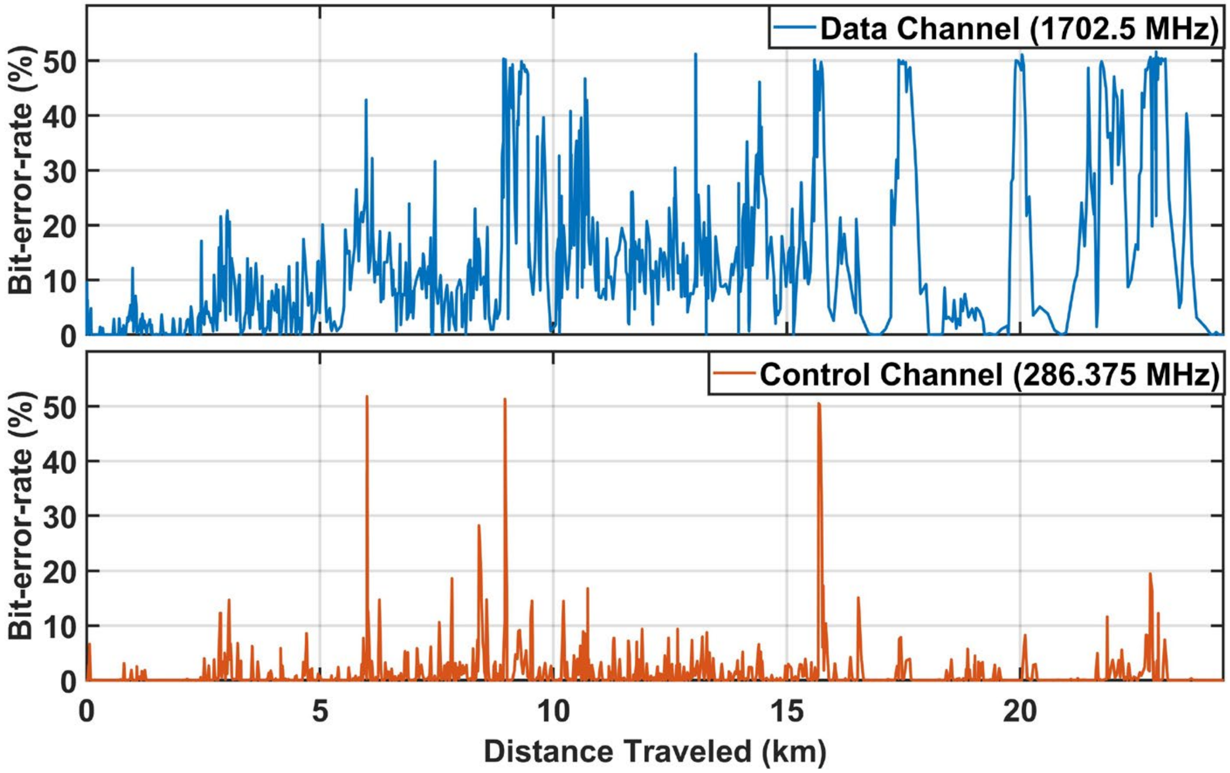


Figure 39. Comparison of BER for the residential drive route and bit rate = 1 Mbps.

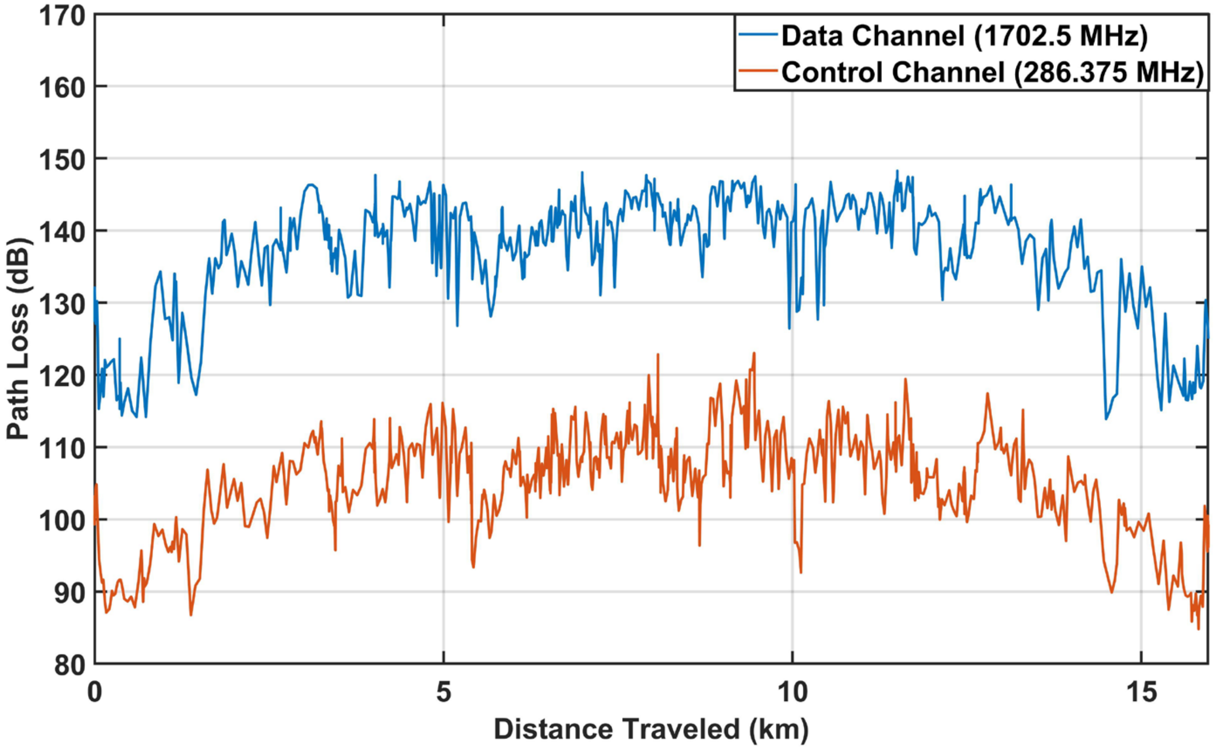


Figure 40. Comparison of path loss for the urban drive route and bit rate = 1 Mbps.

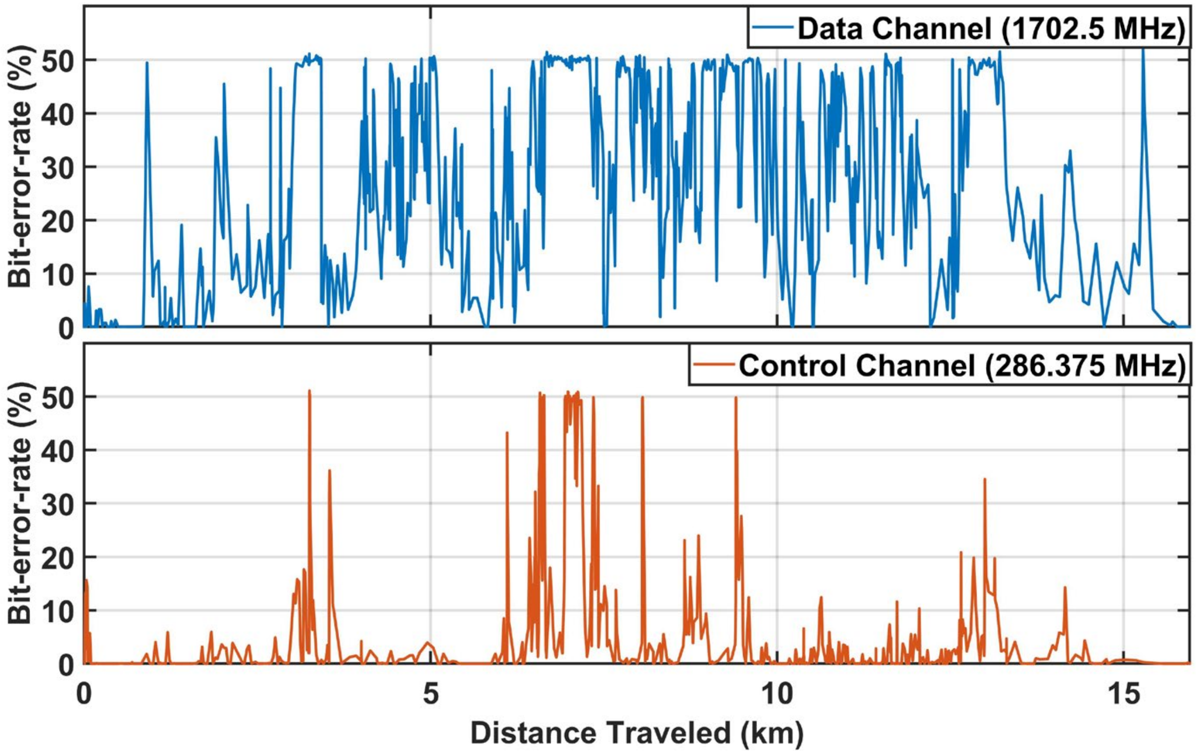


Figure 41. Comparison of BER for the urban drive route and bit rate = 1 Mbps.

3.3 Summary of Results

In order to summarize the results and to facilitate comparisons, the mean, median, maximum, and minimum (M4) statistics were computed for BER, measured path loss, calculated free-space path loss, and the difference between the measured and calculated path loss. It should be noted that effects of terrain are not accounted for in the free-space path-loss model. The free-space path loss model represents the least possible path loss between 2 points without considering anomalous propagation.

The results of this statistical analysis are presented in Tables 8–11. Each table provides a quick comparison between the data channel and the control channel for each of the 4 test scenarios.

- Table 8 provides the M4 statistics for the residential route with a bit rate of 100 kbps.
- Table 9 provides the M4 statistics for the residential route with a bit rate of 1 Mbps.
- Table 10 provides the M4 statistics for the urban route with a bit rate of 100 kbps.
- Table 11 provides the M4 statistics for the urban route with a bit rate of 1 Mbps.

Table 8. M4 Statistics of the data collected on the residential route with a bit rate of 100 kbps.

Parameter	Channel	Mean	Median	Maximum	Minimum
BER (%)	Data	39.33	46.49	57.259	0
	Control	1.67	0.004	46.75	0
Measured Path Loss (dB)	Data	140.89	138.4	157.08	116.39
	Control	106.88	104.22	116.71	85.29
Calculated Free-Space Path Loss (dB)	Data	103.1	103.1	106.21	95.11
	Control	87.41	87.241	90.72	79.61
Difference (dB) Measured - Calculated	Data	37.79	35.304	50.87	21.28
	Control	19.47	16.98	25.99	5.67

Table 9. M4 Statistics of the data collected on the residential route with a bit rate of 1 Mbps.

Parameter	Channel	Mean	Median	Maximum	Minimum
BER (%)	Data	16.77	10.62	51.181	0
	Control	1.4	0.05	50.22	0
Measured Path Loss (dB)	Data	138.71	137.46	145.72	113.95
	Control	107.76	104.22	121.42	85.87
Calculated Free-Space Path Loss (dB)	Data	103.02	103.05	106.21	95.1
	Control	87.67	87.81	90.72	79.65
Difference (dB)	Data	35.7	34.41	39.52	18.85

Parameter	Channel	Mean	Median	Maximum	Minimum
Measured - Calculated	Control	20.09	16.41	30.7	6.22

Table 10. M4 Statistics of the data collected on the urban route with a bit rate of 100 kbps.

Parameter	Channel	Mean	Median	Maximum	Minimum
BER (%)	Data	40.64	46.84	55.64	0
	Control	1.9	0.002	45.2	0
Measured Path Loss (dB)	Data	145	141.58	157.26	114.43
	Control	109.24	106.39	124.1	86.23
Calculated Free-Space Path Loss (dB)	Data	105.27	105.87	107.54	97.73
	Control	90.01	90.69	92.08	82.26
Difference (dB) Measured - Calculated	Data	39.72	35.71	49.72	16.7
	Control	19.23	15.7	32	3.97

Table 11. M4 Statistics of the data collected on the urban route with a bit rate of 1 Mbps.

Parameter	Channel	Mean	Median	Maximum	Minimum
BER (%)	Data	30.68	36	51.43	0
	Control	10.78	1.37	51.07	0
Measured Path Loss (dB)	Data	142.66	141.54	148.26	114.18
	Control	111.17	107.3	122.84	86.714
Calculated Free-Space Path Loss (dB)	Data	105.66	106.25	107.52	97.73
	Control	90.19	90.74	92.08	82.25
Difference (dB) Measured - Calculated	Data	37	35.29	40.75	16.46
	Control	20.98	16.55	30.76	4.47

As a final summary, Table 12 provides the difference between the median statistics for BER and measured path loss for each of the test case scenarios. This table is meant to provide an indication of the improvement gained in both BER and path loss on the control channel over the data channel.

Table 12. Summary of improvement of the median BER and path loss in the control channel over the data channel.

Test Case Scenario	Measured Paremeter	Difference in the Median (Data Channel – Control Channel)
Residential Route 100 kbps	BER (%)	46.686
	Path Loss (dB)	34.18

Test Case Scenario	**Measured Paremeter**	**Difference in the Median (Data Channel – Control Channel)**
Residential Route 1 Mbps	BER (%)	10.57
	Path Loss (dB)	33.24
Urban Route 100 kbps	BER (%)	46.84
	Path Loss (dB)	35.19
Urban Route 1 Mbps	BER (%)	34.63
	Path Loss (dB)	34.24

4. CONCLUSIONS

The objective of this study was to explore the feasibility of a radio network taking advantage of a dual-channel architecture to improve network connectivity by using a lower frequency to transmit network control messages and a higher frequency for the data transmission. Looking at the Table 12 in Section 3.3, the simulated control channel at the lower 286.375 MHz frequency experienced an improvement in median propagation loss of 33–35 dB over the simulated data channel at the higher 1702.5 MHz frequency.

In all test cases median BER was also improved on the control channel; however, this improvement was much more dependent on the test case scenario. The improvement in median BER ranged from approximately 10% on the residential route with a bit rate of 1 Mbps to over 45% for the two runs with a bit rate of 100 kbps. The improved BER is most likely due to improved signal-to-noise ratio (SNR) gained due to the reduced path loss.

The one unusual finding in the data was that for the higher frequency, the BER was higher for the lower bit rate than for the higher. This is probably caused by the higher bit rate having more preambles for the receiver to synchronize to. Further investigation is needed to verify this.

In conclusion, the data presented in this report indicates that it is feasible that a tactical dual-band radio network using a low frequency control channel will have better reliability than a single band radio network.

5. REFERENCES

[1] Defense Advanced Research Projects Agency (DARPA), Strategic Technology Office, Broad Agency Announcement (BAA) HR001118S0012, “Network Universal Persistence (Network UP),” January 5, 2018. Accessed https://www.fbo.gov/index?s=opportunity&mode=form&tab=core&id=1d9a6fd2130787c1e24c30140e4533ff&_cview=1 September 10, 2018.

[2] NMEA 0183: Standard for interfacing marine electronic devices, National Marine Electronics Association: Severna Park, MD

[3] Symbol to Bit Mapping and Demapping Blocks for GNU Radio, https://github.com/gr-vt/gr-mapper

[4] Couch, Leon W. *Digital and Analog Communications Systems.* 6th ed. Upper Saddle River, NJ: Prentice Hall, 2002, pp 141, 183-188,

[5] Chriss Hammerschmidt and Robert Johnk, “Extracting Clutter Metrics From Mobile Propagation Measurements in the 1755 – 1780 MHz Band,” in *2016 IEEE Military Communications Conference*, Baltimore, MD, Nov. 2016, pp. 213–218. https://www.its.bldrdoc.gov/publications/3166.aspx

[6] Symbol to Bit Mapping and Demapping Blocks for GNU Radio (NTIA Fork from [3]), https://github.com/NTIA/gr-mapper

ACKNOWLEDGEMENTS

The authors would like to thank Chriss Hammerschmidt for her support in conducting preliminary path loss measurements and for sharing her knowledge in conducting these types of measurements. The authors would also like to thank Peter Mathys and Robert Achatz for their support in troubleshooting the measurement system during its development. Finally the authors would like to thank Kenneth Brewster, Mike Chang, Savio Tran, and Erin Faubion for their support assembling the measurement system and in performing the measurements.

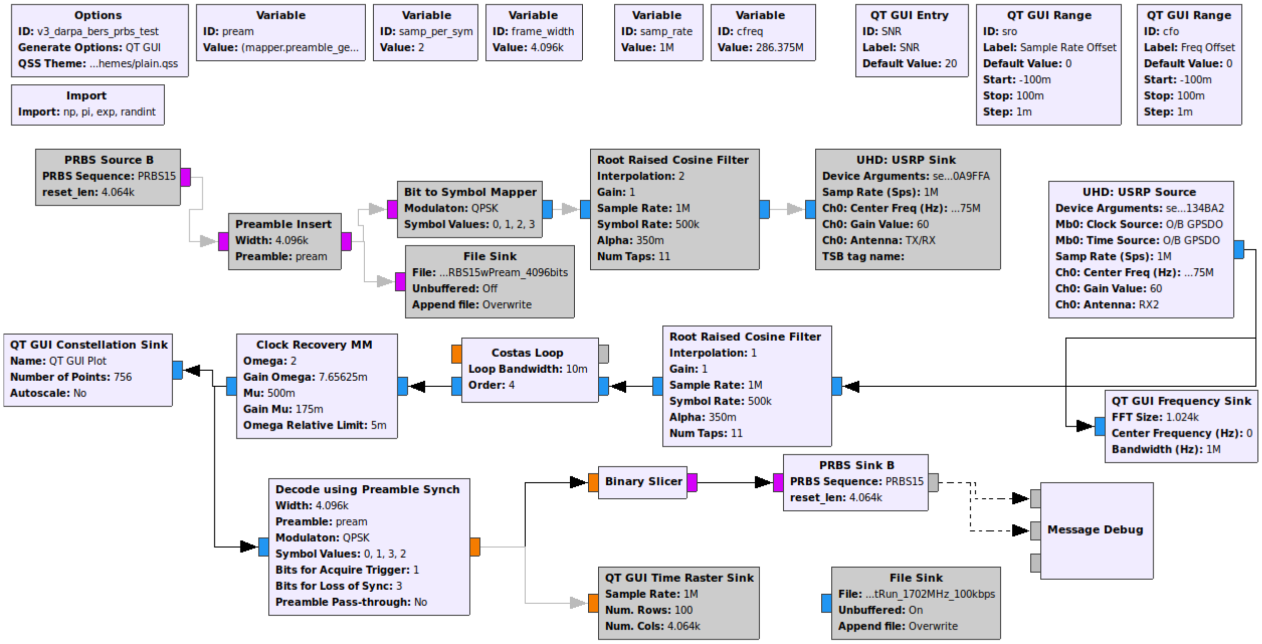
APPENDIX A. DESCRIPTION OF USRP BASED BER MEASUREMENT SYSTEM

This appendix describes the USRP based BER measurement system. The first section describes the configuration of the GRC flow graph while the second section describes the modifications that were made to the *gr-mapper* out-of-tree-module available on GitHub from [3]. The final section describes some performance tests that were conducted to measure BER as a function of signal-to-noise ratio (SNR) under controlled conditions.

One thing to note is that the PRBS generator included in this module does not produce a true maximum length sequence for any of the available PRBS options. Due to time constraints, it was not possible to debug the PRBS generator and fix the problem. Because the PRBS generator did not produce a standard PRBS-15 bit sequence, it was not possible to use the built-in PRBS sequence generator on the VSG. Instead, GRC was used to generate the sequence the way that the PRBS Sink B block expected and then capture the sequence using a file sink block. This captured bit sequence was then converted to a format that could be transferred to the VSG.

A.1 GRC Flow Graph Configuration

Figure A-1 shows the complete flow graph as used for the BER measurements described in this report. The grayed out blocks were used for testing purposes and for generating the bit sequence that was input into the VSG used as the transmitter but were not active during the measurements. In this section a description of each of the blocks in the flow graph and the settings that were used in each is provided.



Options
ID: v3_darpa_bers_prbs_test
Generate Options: QT GUI
QSS Theme: ...hemes/plain.qss

Import
Import: np, pi, exp, randint

Variable
ID: pream
Value: (mapper.preamble_ge...

Variable
ID: samp_per_sym
Value: 2

Variable
ID: frame_width
Value: 4.096k

Variable
ID: samp_rate
Value: 1M

Variable
ID: cfreq
Value: 286.375M

QT GUI Entry
ID: SNR
Label: SNR
Default Value: 20

QT GUI Range
ID: sro
Label: Sample Rate Offset
Default Value: 0
Start: -100m
Stop: 100m
Step: 1m

QT GUI Range
ID: cfo
Label: Freq Offset
Default Value: 0
Start: -100m
Stop: 100m
Step: 1m

PRBS Source B
PRBS Sequence: PRBS15
reset_len: 4.064k

Preamble Insert
Width: 4.096k
Preamble: pream

Bit to Symbol Mapper
Modulaton: QPSK
Symbol Values: 0, 1, 2, 3

File Sink
File: ...RBS15wPream_4096bits
Unbuffered: Off
Append file: Overwrite

Root Raised Cosine Filter
Interpolation: 2
Gain: 1
Sample Rate: 1M
Symbol Rate: 500k
Alpha: 350m
Num Taps: 11

UHD: USRP Sink
Device Arguments: se...0A9FFA
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): ...75M
Ch0: Gain Value: 60
Ch0: Antenna: TX/RX
TSB tag name:

UHD: USRP Source
Device Arguments: se...134BA2
Mb0: Clock Source: O/B GPSDO
Mb0: Time Source: O/B GPSDO
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): ...75M
Ch0: Gain Value: 60
Ch0: Antenna: RX2

QT GUI Constellation Sink
Name: QT GUI Plot
Number of Points: 756
Autoscale: No

Clock Recovery MM
Omega: 2
Gain Omega: 7.65625m
Mu: 500m
Gain Mu: 175m
Omega Relative Limit: 5m

Costas Loop
Loop Bandwidth: 10m
Order: 4

Root Raised Cosine Filter
Interpolation: 1
Gain: 1
Sample Rate: 1M
Symbol Rate: 500k
Alpha: 350m
Num Taps: 11

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

Decode using Preamble Synch
Width: 4.096k
Preamble: pream
Modulaton: QPSK
Symbol Values: 0, 1, 3, 2
Bits for Acquire Trigger: 1
Bits for Loss of Sync: 3
Preamble Pass-through: No

Binary Slicer

PRBS Sink B
PRBS Sequence: PRBS15
reset_len: 4.064k

Message Debug

QT GUI Time Raster Sink
Sample Rate: 1M
Num. Rows: 100
Num. Cols: 4.064k

File Sink
File: ...tRun_1702MHz_100kbps
Unbuffered: On
Append file: Overwrite

Figure A-1. GRC flow graph used to run the BER measurements described in this report.

The first block on the left at the top of the flow graph seen in Figure A-1 is the options block, which is used to set special parameters for the flow graph. The settings in this block that were used for the measurements described in this report are provided below. The majority of these settings were left as the defaults.

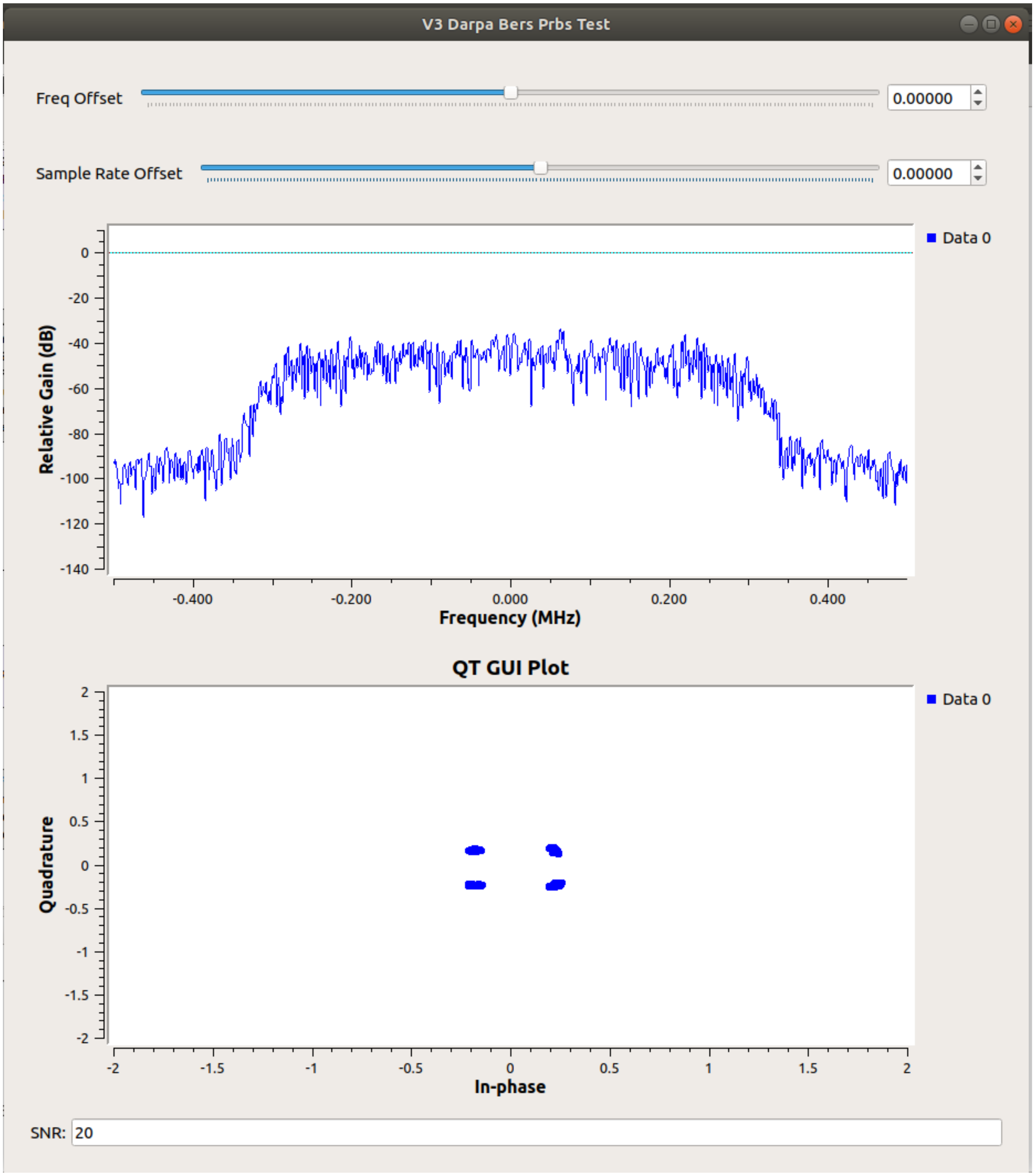
- ID: v3_darpa_bers_prbs_test
- Title: N/A
- Author: N/A
- Description: N/A
- Canvas Size: 1280, 1024
- Generate Options: QT GUI
- Run: Autostart
- Max Number of Output: 0
- Realtime Scheduling: Off
- QSS Theme: This is specific to the laptop and how the directory is set up but the plain.qss theme was used
- Thread-safe setters: Off
- Run Command: {python} -u{filename}
- Hier Block Source Path: .:
- Comment: N/A

The block beneath the options is the import block. This is used to add Python modules into the namespace. Using this block exp and pi were imported from cmath, radint was imported from random, and numpy was imported as np. These imports were given the ID import_0. The full line of imports used in this blocks was: from cmath import exp, pi; from random import randint; import numpy as np. Nothing was entered under the advanced settings.

There were 5 variable blocks used in the flow graph; three of which were held constant through all of the measurements. The two variables that changed depending on the measurement scenario were the sample rate (*samp_rate*) and the center frequency (*cfreq*). How these two variables were changed is described in Table 4. The preamble variable (*pream*) was created using a preamble generator within *gr-mapper*. Since the same seed was used for all tests, this always produced the same bit sequence to be used as the preamble for all of the measurement runs. All of the variables were set as follows:

- Pream: (mapper.preamble_generator(32,11,1033)).get_preamble()
  - This produced a 32 bit sequence of [1 0 1 0 0 1 1 0 1 0 1 0 1 0 1 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 1 1].
- Samp_per_symbol: 2
  - This is standard for a QPSK modulation
- Frame_width: 4.096 kbits (including the 32 bit preamble)
- Samp_rate: 1Msps for the 1 Mbps test cases and 100 ksps for the 100 kbps test cases
- Cfreq: 286.375 MHz for the control channel test cases and 1702.5 MHz for the data channel test cases

The three blocks in the upper right of Figure A-1 added features to the graphical user interface (GUI) that was shown while the measurements ran. A sample of this GUI is shown in Figure A-2. This GUI was primarily used to ensure the system was running properly while the measurements were being conducted. The data that were used to produce the results provided in this report all came from a text file that was produced when the GRC flow graph was executed. The QT GUI Entry block was used to add an input box that allowed the user to adjust the SNR using a channel simulation block that was eliminated for the final measurements.



Freq Offset 0.00000

Sample Rate Offset 0.00000

SNR: 20

Figure A-2. GUI produced by GRC while running the BER measurements described in this report.

The QT GUI Range blocks were used to add sliders to the GUI that would let the user adjust sample rate and center frequency slightly during testing. This was used while testing the system to see how these slight offsets would affect the BER. For the measurements both of these offsets were set to 0.

There are two other GUI blocks in the flow graph. The first is the QT GUI Frequency Sink block on the output of the UHD: USRP Source block. This block adds a plot to the GUI of the fast Fourier transform (FFT) of the sampled input signal received by the USRP. This provided immediate feedback to the user that the signal of interest was present. The second block called QT GUI Constellation Sink on the output of the Clock Recovery MM block created a constellation diagram in the GUI that provided the user with an indication of how well the signal was being demodulated. The settings for these two GUI blocks were as follows:

- QT GUI Frequency Sink
  - General
    - ID: qtgui_freq_sink_x_0
    - Type: Complex
    - Name: “”
    - FFT Size: 1024
    - Window Type: Blackman-Harris
    - Center Frequency (Hz): cfo
    - Bandwidth (Hz): samp_rate
    - Grid: No
    - Autoscale: No
    - Average: None
    - Y min: -140
    - Y max: 10
    - Y Label: Relative Gain
    - Y Units: dB
    - Number of Inputs: 1
    - Update Period: 0.10
    - GUI Hint: N/A
    - Show Msg Ports: No
  - Trigger
    - Trigger Mode: Free
    - Trigger Level: 0.0
    - Trigger Channel: 0
    - Trigger Tag Key: “”
  - Config
    - Control Panel: No
    - Legend: Yes
    - Axis Labels: Yes
    - Line 1 Label: N/A
    - Line 1 Width: 1
    - Line 1 Color: Blue
    - Line 1 Alpha: 1.0
  - Advanced

        - Not used
- QT GUI Constellation Sink
    - General
        - ID: qtgui_const_sink_x_0
        - Type: Complex
        - Name: QT GUI Plot
        - Number of Points: 840-84
        - Grid: No
        - Autoscale: No
        - Y min: -2
        - Y max: 2
        - X min: -2
        - X max: 2
        - Number of Inputs: 1
        - Update Period: 0.10
        - GUI Hint: N/A
    - Trigger
        - Trigger Mode: Free
        - Trigger Slope: Positive
        - Trigger Level: 0.0
        - Trigger Channel: 0
        - Trigger Tag Key: “”
    - Config
        - Legend: Yes
        - Axis Labels: Yes
        - Line 1 Label: N/A
        - Line 1 Width: 1
        - Line 1 Color: Blue
        - Line 1 Style: None
        - Line 1 Marker: Circle
        - Line 1 Alpha: 1.0
    - Advanced
        - Not Used

The first block in the chain of signal processing is the UHD: USRP Source. This block configures the USRP to act as a receiver and digitizer. One of the key parameters to set in this block, aside from center frequency and sample rate, is the gain. The gain setting determines the noise figure and dynamic range of the USRP. In other work being done at ITS an engineer performed an extensive study of USRP performance. During this study it was found that a gain setting of 40 provides the best trade-off between noise figure and dynamic range by maximizing dynamic range while providing a decent noise figure. However, for the measurements described in this report, a gain of 60 was used to get an even better noise figure at the expense of some dynamic range. This was to accommodate the measurements of the data channel. During preliminary testing it was discovered that using a gain value of 60 provided a slightly better SNR and therefore better BER measurements. In an effort to keep the majority of settings the same throughout all of the measurements, the gain was held at 60 throughout the measurement campaign. All of the settings in the UND: USRP Source block are provided below.

- General
  - ID: uhd_usrp_source_0
  - Output Type: Complex Float32
  - Wire Format: Automatic
  - Stream args: N/A
  - Stream channels: []
  - Device Address: “”
  - Device Arguments: “Serial=3134BA2”
    - This is device specific. The serial number of the USRP B200 being used must be entered here.
  - Sync: don’t sync
  - Clock Rate (Hz): Default
  - Num Mboards: 1
  - Mb0: Clock Source: O/B GPSDO
  - Mb0: time Source: O/B GPSDO
  - Mb0: Subdev Spec: N/A
  - Num Channels: 1
  - Samp Rate (Sps): samp_rate (variable described previously)
- RF Options
  - Ch0: Center Freq (Hz): cfreq (variable described previously)
  - Ch0: Gain Value: 60
  - Ch0: Gain Type: Absolute (dB)
  - Ch0: Antenna: RX2
  - Ch0: Bandwidth (Hz): 0
- FE Corrections
  - Ch0: Enable DC Offset Cc: “”
  - Ch0: Enable IQ Imbalance: “”
- Advanced
  - Show Command Port: No
  - Show LO Controls: No
  - Block Alias: N/A
  - Core Affinity: N/A
  - Min Output Buffer: 0
  - Max Output buffer: 0
  - Comment: N/A

The next block in the signal processing chain is the Root Raised Cosine Filter block. This block filters the incoming sample using the root raised cosine filter. The parameters of this block were set such that they matched the filter applied to the transmitted signal. The settings were as follows.

- General
  - ID: root_raised_cosine_filter_0_0
  - FIR Type: Complex->Complex (Interpolating)
  - Interpolation: samp_per_sym*0+1
  - Gain: 1

  - Sample Rate: samp_rate (variable described previously)
  - Symbol Rate: samp_rate/samp_per_sym (both variables described previously)
  - Alpha: 0.35
  - Num Taps: 11
- Advanced
  - Block Alias: N/A
  - Core Affinity: N/A
  - Min Output Buffer: 0
  - Max Output buffer: 0
  - Comment: N/A

The next block is the Costas Loop block which recovers the carrier frequency and downconverts to baseband. The settings for this block were:

- General
  - ID: digital_costas_loop_cc_0
  - Loop Bandwidth: 0.01
  - Order: 4
  - Use SNR: No
- Advanced
  - Block Alias: N/A
  - Core Affinity: N/A
  - Min Output Buffer: 0
  - Max Output buffer: 0
  - Comment: N/A

Following the Costas loop block is the Clock Recovery MM block. This block implements the Muller and Muller discrete time error tracking synchronizer according to the block documentation. For the most part the defaults were kept within this block. The settings for this block are provided below:

- General
  - ID: digital_clock_recovery_mm_xx_0
  - Type: Complex
  - Omega: samp_per_sym*(1+0.0) (variable described previously)
  - Gain Omega: 0.25*0.175*0.175
  - Mu: .5
  - Gain Mu: 0.175
  - Omega Relative Limit: 0.005
- Advanced
  - Block Alias: N/A
  - Core Affinity: N/A
  - Min Output Buffer: 0
  - Max Output buffer: 0
  - Comment: N/A

The Decode using Preamble Synch block that appears next in the signal processing chain is part of the *gr-mapper* out-of-tree module. The code for this block was edited slightly, as will be described in more detail in the next section of this appendix. This block locks to the preamble at the beginning of a frame, converts the symbols to bits, and removes the bits making up the preamble from the frame. These bits are then passed on to subsequent blocks to determine the BER. The settings for this block are provided below.

- General
    - ID: mapper_preamble_sync_demapper_0
    - Width: frame_width (variable described previously)
    - Preamble: pream (variable described previously)
    - Modulation: QPSK
    - Symbol Values: [0, 1, 3, 2]
    - Bits for Acquire Trigger: 1
    - Bits for Loss of Sync: 3
    - Preamble Pass-through: No
- Advanced
    - Block Alias: N/A
    - Core Affinity: N/A
    - Min Output Buffer: 0
    - Max Output buffer: 0
    - Comment: N/A

The next block after the decoding block is the binary slicer which was also part of the *gr-mapper* out-of-tree module. This block converts the floating point input to a one bit output that can be read by the following block to compute BER. All of the default settings were kept for this block as there aren't any general settings to change aside from the ID.

The PRBS Sink B block is next and this is the block that computes the BER. This block was also found in the gr-mapper module and the code was altered slightly. The original version of this block output the BER continuously and the BER was computed over the duration of the measurement. This was altered to compute BER over one second of bits and output this to the screen. Additional Python code was added to also output the GPS strings. This is described in more detail in the next section. The settings for this block are provided below.

- General
    - ID: mapper_prbs_sink_b_0
    - PRBS Sequence: PRBS15
    - Reset_len: fram_width-len(pream) (variables defined previously)
- Advanced – same as the last three listed blocks.

The final block in the flow graph is the Message Debug block. This block was added to capture tags that were added to the decode block containing BER information.

A.2 Alterations to GRC Blocks

In order to get the BER in the desired format and to tag each BER measurement with GPS strings the code for two of the blocks was altered. The first block that was changed was the Decode using Preamble Synch block. The file containing the C++ code for this block is called preamble_synch_demapper_impl.cc. Code was added to this file to catch tags coming from the USRP source block. The rx_rate tag (USRP sample rate), number of samples consumed, and the number of samples produced were used to compute the relative sample rate at the output of the block. In addition, once a second, an outgoing sample was tagged with the current time. The altered code, in its entirety is provided here.

```
/* -*- c++ -*- */
/*
 * Copyright 2015 Free Software Foundation, Inc
 *
 * This is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3, or (at your option)
 * any later version.
 *
 * This software is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this software; see the file COPYING.  If not, write to
 * the Free Software Foundation, Inc., 51 Franklin Street,
 * Boston, MA 02110-1301, USA.
 */

#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include <gnuradio/io_signature.h>
#include "preamble_sync_demapper_impl.h"
#include <stdio.h>
#include <stdexcept>
#include <cmath>


namespace gr {
    namespace mapper {

        preamble_sync_demapper::sptr
        preamble_sync_demapper::make(int width, const std::vector<unsigned char>
&preamble, modtype_t modtype,
                                     const std::vector<int> &symbol_map, int
acquire_bit_thresh, int loose_bit_thresh,
                                     bool passthrough)
        {
            return gnuradio::get_initial_sptr
                (new preamble_sync_demapper_impl(width, preamble, modtype, symbol_map,
acquire_bit_thresh, loose_bit_thresh, passthrough));
        }

        /*
         * The private constructor
         */
        preamble_sync_demapper_impl::preamble_sync_demapper_impl(int width, const
std::vector<unsigned char> &preamble, modtype_t modtype,
                                                                  const
std::vector<int> &symbol_map, int acquire_bit_thresh, int loose_bit_thresh,
                                                                  bool passthrough)
            : gr::block("preamble_sync_demapper",
                        gr::io_signature::make(1, 1, sizeof(gr_complex)),
                        gr::io_signature::make(1, 1,
sizeof(float))),//gr::io_signature::make(1, 1, sizeof(uint8_t))),
            d_const(modtype, symbol_map),
            d_preamble(preamble),
            d_state(0),
            d_modtype(modtype),
            d_symbol_map(symbol_map),
            d_acq_bt(acquire_bit_thresh),
            d_loo_bt(loose_bit_thresh),
            d_mapidx(0),
            d_offset(0),
            d_passthrough(passthrough),
            d_processed(0),
            d_last_reported_seconds(0),
            d_samp_rate(0),
            d_samp_last_time(0)
//        d_loose_count(0)
        {
            //printf("Made block.\n");
            d_bps = d_const.bps();
            assert(width%d_bps == 0);
            d_width = width;

            // rx_{time,rate} is meaningless on the other side of this block
            set_tag_propagation_policy(TPP_DONT);

            d_rx_time_tag_key = pmt::intern("rx_time");
            d_rx_rate_tag_key = pmt::intern("rx_rate");
        }

        /*
         * Our virtual destructor.
         */
        preamble_sync_demapper_impl::~preamble_sync_demapper_impl()
        {
        }

        void
        preamble_sync_demapper_impl::forecast (int noutput_items, gr_vector_int
&ninput_items_required)
        {
            /* <+forecast+> e.g. ninput_items_required[0] = noutput_items */
            int payload_sym_count = d_width/d_bps;
            int data_sym_count = payload_sym_count - d_preamble.size()/d_bps;
            int max_out;
            if(d_passthrough){
                max_out = (int) ceil((float)noutput_items / float(payload_sym_count *
d_bps));
            }
            else{
                max_out = (int) ceil((float)noutput_items / float(data_sym_count *
d_bps));
            }
            ninput_items_required[0] = max_out * payload_sym_count;//Needs at least
one full payload to output anything
            //printf("Forecasted\n");
        }

        int
        preamble_sync_demapper_impl::general_work (int noutput_items,
                                                   gr_vector_int &ninput_items,
                                                   gr_vector_const_void_star
&input_items,
                                                   gr_vector_void_star &output_items)
        {
            const gr_complex *in = (const gr_complex *) input_items[0];
            float *out = (float *) output_items[0];
            //uint8_t *out = (uint8_t *) output_items[0];

            int oo(0), ii(0);
            int consumed(0);
            int produced(0);
            int pream_len = (int) d_preamble.size();
            int preamble_space = pream_len/d_bps;
            int payload_space = d_width/d_bps;
            int data_space = payload_space - preamble_space;
            int M = (int) pow(2.0,(double)d_bps);
            int N = d_const.nmaps();
            int MN = M*N;

            size_t range_start = this->nitems_read(0);
            size_t range_stop = range_start + ninput_items[0];

            // extract sample rate from stream tag
            if (!d_samp_rate) {
                d_rx_rate_tags.clear();
                this->get_tags_in_range(d_rx_rate_tags, 0, range_start, range_stop,
d_rx_rate_tag_key);
                if (!d_rx_rate_tags.empty()) {
                    d_rx_rate_tag = d_rx_rate_tags.back();
                    d_samp_rate = round(pmt::to_double(d_rx_rate_tag.value));
                    printf("Sample rate set to %d\n", d_samp_rate);
                }
            }

            bool time_synced = false;
            // extract rx time from stream tag
            d_rx_time_tags.clear();
            this->get_tags_in_range(d_rx_time_tags, 0, range_start, range_stop,
d_rx_time_tag_key);
            if (!d_rx_time_tags.empty()) {
                // got an rx time tag, use it to correct time
                d_rx_time_tag = d_rx_time_tags.back();
                size_t offset = d_rx_time_tag.offset;

                if (!d_samp_last_time || offset > d_samp_last_time) {
                    d_rx_time = d_rx_time_tag.value;
                    d_seconds = pmt::to_uint64(pmt::tuple_ref(d_rx_time, 0));
                    d_fractional_seconds = pmt::to_double(pmt::tuple_ref(d_rx_time,
1));
                    d_samp_last_time = offset | 1;
                    printf("Time synced. ");
                    time_synced = true;
                }
            }

            if (!time_synced) {
                // interpolate time
                int samps_since_time = nitems_read(0) - d_samp_last_time;
                if (d_samp_rate && samps_since_time >= d_samp_rate) {
                    // 1 second has elapsed since last time calculation
                    d_seconds += 1;
                    d_samp_last_time += d_samp_rate;
                }
            }

            std::vector<uint8_t> comparer(d_preamble.size(),255);
            std::vector< std::vector<uint8_t> > comparing(MN,comparer);
            //printf("Working.\n");
            int n_symbols;
            if(d_passthrough){
                n_symbols = payload_space;
            }
            else{
                n_symbols = data_space;
            }

//        while((oo<noutput_items) && (ii < ninput_items[0]) && (ninput_items[0] >=
payload_space)) {
            while(1) {
                if(!(ninput_items[0] >= payload_space)){
//            printf("ninput break.\tninput = %d\tpayload = %d\n",ninput_items[0],
payload_space);
                    break;
                }
                if(!(oo < noutput_items - n_symbols*d_bps)){
//            printf("oo break.\too = %d\ttotal_out = %d\n",oo,noutput_items -
n_symbols*d_bps);
                    break;
                }
                if(!(ii < ninput_items[0] - data_space)){
                    //printf("ii break.\tii = %d\tninput = %d\tstate =
%d\n",ii,ninput_items[0],d_state);
                    break;
                }
                switch(d_state) {
                case 0: {
                    //printf("State 0:\n");
                    for(int idx=0; idx < MN; idx++){
                        for(int ind=d_bps; ind < pream_len; ind++){
                            comparing[idx][ind-d_bps] = comparing[idx][ind];
                        }
                    }
                    //printf("Compare has shifted.\t Pream_len = %d\n",pream_len);
                    for(int idx=0; idx < MN; idx++){
                        uint8_t symbol_bits[d_bps];
                        gr_complex symbol = in[ii] * exp(gr_complex(0,2*M_PI*idx/MN));
                        d_const.demap_hard(&symbol, &symbol_bits[0], 1);
                        for(int ind = d_bps; ind > 0; ind--){
                            //printf("comparing[%d][%d] = symbol_bits[%d] =
%d\n",idx,pream_len-ind,d_bps-ind,symbol_bits[d_bps-ind]);
                            comparing[idx][pream_len-ind] = symbol_bits[d_bps-ind];
                        }
                    }
                    //printf("Compare has updated.\n");
                    int ham_dist[MN];
                    for(int idx=0; idx < MN; idx++){ ham_dist[idx] = 0; }
                    for(int idx=0; idx < MN; idx++){
                        //printf("idx = %d of MN = %d\t",idx,MN);
                        for(int ind=0; ind < pream_len; ind++){
                            ham_dist[idx] += d_preamble[ind]^comparing[idx][ind];
                        }
                        //printf("dist = %d\n",ham_dist[idx]);
                    }
                    bool acq_met = false;
                    bool loo_met = false;
                    for(int idx=0; idx < MN; idx++){
                        //printf("ii = %d,\tham_dist[%d] =
%d\n",ii,idx,ham_dist[idx]);
                        if(ham_dist[idx] <= d_acq_bt){
                            d_offset = idx;
                            acq_met = true;
                            //               d_loose_count = 0;
                            d_state = 2;
                            //printf("Acquired @ ii = %04d\t%d\td_offset = %d\tdist =
%d\tidx = %d\n",ii,d_processed+ii,d_offset,ham_dist[idx],idx);
                            //printf("%d\n",d_processed+ii);
                        }
                        else if(ham_dist[idx] <= d_loo_bt){
                            if(!acq_met){
                                d_offset = idx;
                                loo_met = true;
                                d_state = 1;
                                //printf("Loose @ ii = %d\td_offset =
%d\n",ii,d_offset);
                                //printf("Loose @ ii = %04d\td_offset = %d\tdist =
%d\tidx = %d",ii,d_offset,ham_dist[idx],idx);
                            }
                        }
                    }
                    if(!acq_met && !loo_met) { ii++; }
                } break;
                case 1: {
                    //loose threshhold reached
                    //printf("\n");
                    //printf("Loose @ ii = %d\td_offset = %d\n",ii,d_offset);
                    //printf("State 1(LOO):\td_offset = %d\tin[%d] =
(%2.4f,%2.4f)\n",d_offset,ii,in[ii].real(),in[ii].imag());
                    //printf("Loose Thresh Met. Proceeding. loose_count =
%d\t",d_loose_count);
/*              if(d_loose_count >=100){
                ii += payload_space;
                d_loose_count = 0;
                }
                else{
                d_loose_count++;
                }*/
                    d_state = 2;
                } break;
                case 2: {
                    //acquire threshhold reached
                    //printf("State 2(ACQ):\td_offset = %d\tin[%d] =
(%2.4f,%2.4f)\n",d_offset,ii,in[ii].real(),in[ii].imag());
                    //printf("Acquiring 1 Payload.\n");
                    int start_pointer;
                    if(d_passthrough){
                        start_pointer = ii+1-preamble_space;
                    }
                    else{
                        start_pointer = ii+1;
                    }

                    std::vector<gr_complex> in_rotated(n_symbols);

                    for(int idx = 0; idx < n_symbols; idx++){
                        in_rotated[idx] = in[start_pointer+idx] *
exp(gr_complex(0,2*M_PI*d_offset/MN));
                    }
                    float soft_out[n_symbols*d_bps];
                    d_const.demap_soft(&in_rotated[0], &soft_out[0], n_symbols);
                    //uint8_t hard_out[n_symbols*d_bps];
                    //d_const.demap_hard(&in_rotated[0], &hard_out[0], n_symbols);
                    memcpy(&out[oo], &soft_out[0], sizeof(float)*n_symbols*d_bps);
                    //memcpy(&out[oo], &hard_out[0], sizeof(uint8_t)*n_symbols*d_bps);
                    /*else{
                      for(int idx = 0; idx < data_space; idx++){
                      in_rotated[idx] = in[ii+1+idx] *
exp(gr_complex(0,2*M_PI*d_offset/MN));
                      }
                      float soft_out[data_space*d_bps];
                      d_const.demap_soft(&in_rotated[0], &soft_out[0], data_space);
                      memcpy(&out[oo], &soft_out[0], sizeof(float)*data_space*d_bps);
                      }*/
                    consumed = ii+1+data_space;
                    produced += n_symbols*d_bps;
                    oo += n_symbols*d_bps;
                    ii += data_space+1;
                    comparing.assign(MN,comparer);
                    d_state = 0;
                } break;
                }
            }
            if((ii >= ninput_items[0] - data_space) && (oo == 0) && (ninput_items[0]
>= 2*payload_space)){
                //printf("No Sync Found. Dumping single payload.\n");
                consumed = payload_space;
                produced = 0;
            }
            //d_const.demap_soft(in,out,noutput_items/d_bps);
            // Do <+signal processing+>
            // Tell runtime system how many input items we consumed on
            // each input stream.
            if(consumed >= 3*payload_space){ printf("nin = %d,\tnout =
%d\n",ninput_items[0],noutput_items); }
            consume_each (consumed);
            d_processed += consumed;

            if (produced && d_seconds != d_last_reported_seconds) {
                // ship tag once a second
                //printf("Time %f at sample %lu\n", d_seconds + d_fractional_seconds,
d_samp_last_time);

                pmt::pmt_t rx_time;
                rx_time = pmt::mp(pmt::from_uint64(d_seconds),
pmt::from_double(d_fractional_seconds));
                add_item_tag(0, nitems_written(0), pmt::intern("rx_time"), rx_time);

                d_last_reported_seconds = d_seconds;
            }

            // Tell runtime system how many output items we produced.
            return produced;
        }

    } /* namespace mapper */
} /* namespace gr */
```

The other block that was modified was the PRBS Sink B block. The Python code for this block was edited such that the tag now created by the Decode block could be caught and BER could be computed on just one second of bits. The computed BER was tagged and reset so that BER could be computed over the next second. Additionally, GPS strings were read from the USRP and output. After the data was collected it was realized that reestablishing a connection to the USRP in this block of code is what introduced the 2 second delay discussed in the body of this report. ITS has published the updated *gr-mapper* module in a GitHub repository [6]. The code found in the NTIA repository is still being edited to see if the delay issue can be resolved. The edited PRBS Sink B block code as used for this study is provided here.

```
#!/usr/bin/env python

import numpy
from gnuradio import gr
import pmt
import prbs_base
from gnuradio import uhd #this seems to import uhd without causing compilation error
import datetime


class prbs_sink_b(gr.sync_block):
    def __init__(self, which_mode="PRBS15", reset_len=100000):
        gr.sync_block.__init__(self, name="prbs_sink_b",
                               in_sig=[numpy.int8], out_sig=[])

        self.base = prbs_base.prbs_base(which_mode, reset_len)
        self.nbits = 0
        self.nerrs = 0

        self.ber_port_id = pmt.intern("ber")
        self.message_port_register_out(self.ber_port_id)

    def work(self, input_items, output_items):
        inb = input_items[0]
        linb = len(inb)
        gen = self.base.gen_n(linb)
        self.uhd_usrp_source_0 = uhd.usrp_source(
          ",".join(("", "serial=3134BA2")),
          uhd.stream_args(
            cpu_format="fc32",
            channels=range(1),
          ),
        )
        tags = self.get_tags_in_window(0, 0, linb, pmt.intern("rx_time"))

        if tags:
            tag = tags[-1]
            rx_time = tag.value
            #print(tag.value)
            gpgga = self.uhd_usrp_source_0.get_mboard_sensor('gps_gpgga')
      gprmc = self.uhd_usrp_source_0.get_mboard_sensor('gps_gprmc')
            print('\n')
            print(gpgga)
            print(gprmc)
            seconds = pmt.to_uint64(pmt.tuple_ref(rx_time, 0))
            fractional_seconds = pmt.to_double(pmt.tuple_ref(rx_time, 1))
            timestamp = seconds + fractional_seconds
            if self.nbits > 0:
                ber = self.nerrs / float(self.nbits)
                print("Computer time: " + str(datetime.datetime.now()))
                print "NBits: %d \tNErrs: %d \tBER: %.4E, \ttimestamp
%f"%(int(self.nbits), int(self.nerrs), ber, timestamp)
                d = pmt.make_dict()
                d = pmt.dict_add(d, pmt.intern('timestamp'),
pmt.from_double(timestamp))
                d = pmt.dict_add(d, pmt.intern('ber'), pmt.from_double(ber))
                self.message_port_pub(self.ber_port_id, d)
    #print(query_gpsdo_sensors('gps_gpgga')) # need more to make this command work
get_mboard_sensor
                self.nerrs = 0
                self.nbits = 0

        self.nerrs += numpy.sum(numpy.bitwise_xor(inb, gen))
        self.nbits += len(inb)
        # if self.nbits > 0:
        #     print "NBits: %d \tNErrs: %d \tBER: %.4E"%(int(self.nbits),
int(self.nerrs), self.nerrs/self.nbits)

        return len(inb)
```

An example of the final output is provided below. This output was produced every 2–3 seconds.

```
GPS_GPGGA: $GPGGA,174245.00,3959.6643,N,10515.6943,W,2,09,1.1,1666.1,M,-21.3,M,,*67
GPS_GPRMC: $GPRMC,174245.00,A,3959.6643,N,10515.6943,W,0.0,0.0,230818,,*29
Computer time: 2018-08-23 11:42:45.537337
NBits: 32512 NErrs: 0      BER: 0.0000E+00,    timestamp 1984.222851
******* MESSAGE DEBUG PRINT ********
((ber . 0) (timestamp . 1984.22))
************************************
```

A.3 BER Measurement System Performance Tests

Under controlled conditions the BER measurement system was tested to measure how BER changed as a function of SNR to verify system performance. The test was configured as seen in Figure A-3. The VSG was set up to transmit the same test signals that were used for the measurements described in this report. The Keysight N5182B VSG was used to produce broadband Gaussian noise greater than the noise floor of the measurement device. This was to ensure both the PXA, which was used to measure the signals, and the USRP were receiving signals at the same SNR values. Four test cases were examined: control channel at 100 kbps, control channel at 1 Mbps, data channel at 100 kbps, and data channel at 1 Mbps. The results of the data channel tests can be seen in Figure A-4 and the results of the control channel tests are shown in Figure A-5.

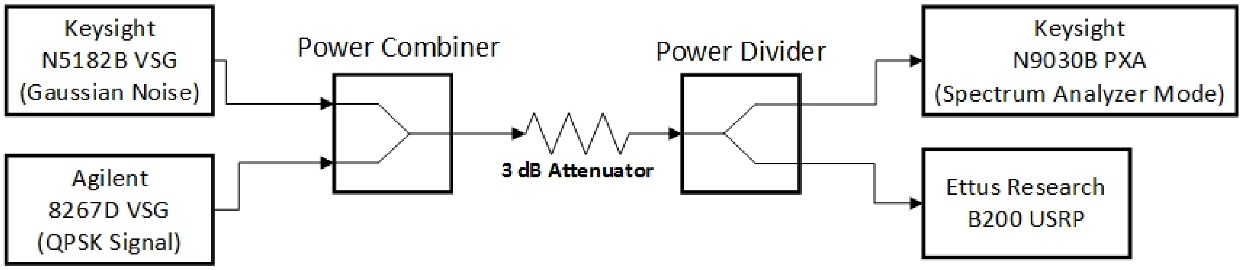


Figure A-3. Block diagram of the system used to measure the BER measurement system performance.

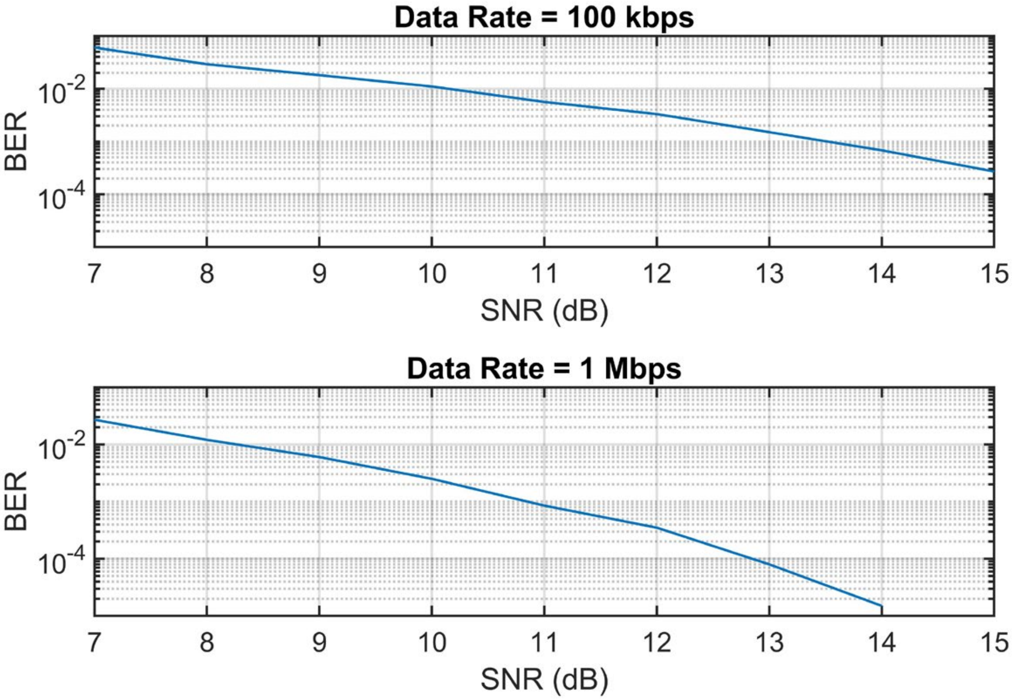


Figure A-4. Results of the BER measurement system performance tests for the data channel frequency 1702.5 MHz.

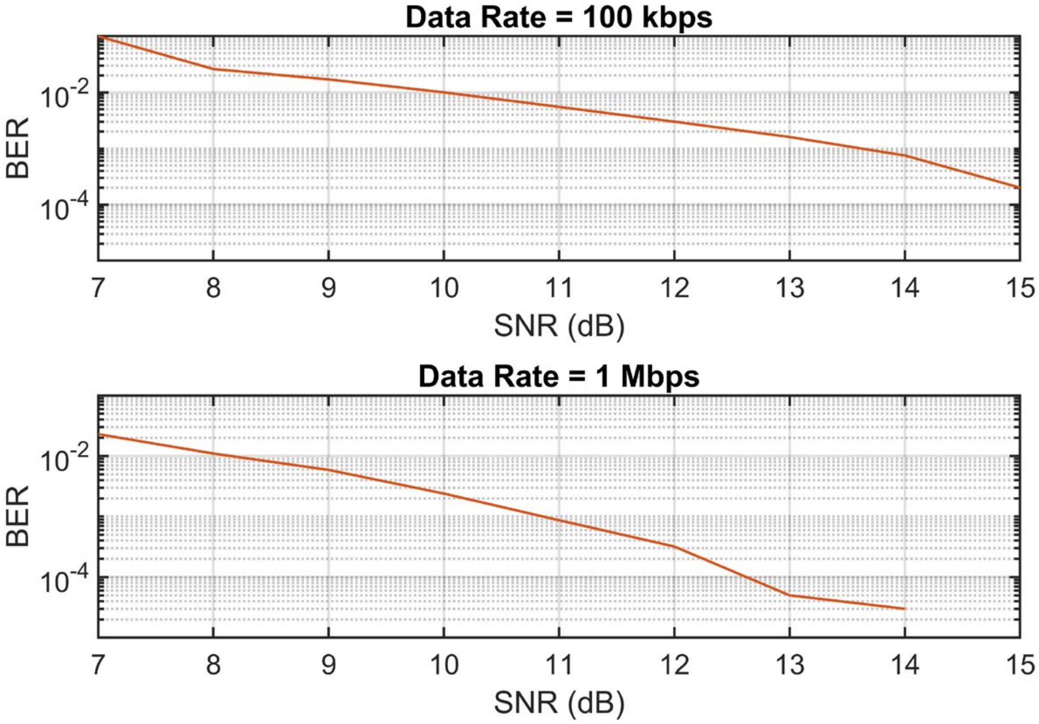


Figure A-5. Results of the BER measurement system performance tests for the control channel frequency 286.375 MHz.

These plots show that the USRP performs better at the higher 1 Mbps bit rate for both the control and data channels. The lower, 100 kbps bit rate, shows poorer performance. This agrees with what was seen during the measurements. To ascertain the reason for this would require further study of the USRP's performance.

NTIA FORM 29
(4-80)

U.S. DEPARTMENT OF COMMERCE
NATIONAL TELECOMMUNICATIONS AND INFORMATION ADMINISTRATION

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION NO. TM-19-539	2. Government Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE Dual-band Tactical Radio Feasibility Study	5. Publication Date June 2019
	6. Performing Organization Code NTIA ITS.M
7. AUTHOR(S) Geoffrey A. Sanders, Anna Paulson	9. Project/Task/Work Unit No. 6836000-300
8. PERFORMING ORGANIZATION NAME AND ADDRESS Institute for Telecommunication Sciences National Telecommunications & Information Administration U.S. Department of Commerce 325 Broadway Boulder, CO 80305	10. Contract/Grant Number.
11. Sponsoring Organization Name and Address Defense Advanced Research Projects Agency (DARPA) Strategic Technology Office (STO) 675 North Randolph St. Arlington, VA. 22203-2114	12. Type of Report and Period Covered

14. SUPPLEMENTARY NOTES

15. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)

In support of the Network Universal Persistence (Network UP) Broad Agency Announcement (BAA) sponsored by the Defense Advanced Research Projects Agency (DARPA) the Institute for Telecommunication Sciences (ITS) was asked to perform a dual-band tactical radio feasibility study. Propagation path loss and bit-error-rate (BER) were measured in different environments in the Boulder, CO, area at two different frequencies. The purpose being to examine how using a low frequency control channel while transmitting data over a higher frequency channel could potentially reduce tactical radio link outages. The results of this study are provided in this report and they indicate that it is feasible that a tactical dual-band radio using a low frequency control channel could have improved network connectivity.

16. Key Words (Alphabetical order, separated by semicolons)

bit-error-rate; path loss; propagation

17. AVAILABILITY STATEMENT ☒ UNLIMITED. □ FOR OFFICIAL DISTRIBUTION.	18. Security Class. (This report) Unclassified	20. Number of pages 81
	19. Security Class. (This page) Unclassified	21. Price: N/A

NTIA FORMAL PUBLICATION SERIES

NTIA MONOGRAPH (MG)

A scholarly, professionally oriented publication dealing with state-of-the-art research or an authoritative treatment of a broad area. Expected to have long-lasting value.

NTIA SPECIAL PUBLICATION (SP)

Conference proceedings, bibliographies, selected speeches, course and instructional materials, directories, and major studies mandated by Congress.

NTIA REPORT (TR)

Important contributions to existing knowledge of less breadth than a monograph, such as results of completed projects and major activities.

JOINT NTIA/OTHER-AGENCY REPORT (JR)

This report receives both local NTIA and other agency review. Both agencies' logos and report series numbering appear on the cover.

NTIA SOFTWARE & DATA PRODUCTS (SD)

Software such as programs, test data, and sound/video files. This series can be used to transfer technology to U.S. industry.

NTIA HANDBOOK (HB)

Information pertaining to technical procedures, reference and data guides, and formal user's manuals that are expected to be pertinent for a long time.

NTIA TECHNICAL MEMORANDUM (TM)

Technical information typically of less breadth than an NTIA Report. The series includes data, preliminary project results, and information for a specific, limited audience.

For information about NTIA publications, contact the NTIA/ITS Technical Publications Office at 325 Broadway, Boulder, CO, 80305 Tel. (303) 497-3572 or e-mail ITSinfo@ntia.doc.gov.