Delay Analysis of NB-IoT Technology with Quectel Board

TESI DI LAUREA MAGISTRALE IN
Telecommunication Engineering

Author: **Karo Amini**

Student ID:	10707670
Advisor:	Giacomo Verticale
Academic Year:	2022-23

Abstract

The Internet of Things (IoT) is a rapidly growing technology that is expected to connect billions of devices and enable innovative applications. Narrowband IoT (NB-IoT) is a new cellular standard developed by the 3rd Generation Partnership Project (3GPP) to support low-power, wide-area networks for IoT. This study presents a study of NB-IoT technology and its deployment scenarios, with a focus on measuring network characteristics using an evaluation board kit from Quectel Wireless Solutions.

The work environment and measurement tools used in this study are described, including Python code for AT commands and remote server communication. Data was collected in various network conditions to evaluate the technology's performance. The collected data was analyzed using techniques such as histogram analysis and correlation analysis of delays with channel signal quality, signal power, signal-to-noise ratio, and reference signal received quality.

The results of this study provide insights into the performance of NB-IoT networks and can aid in improving their deployment and optimization. Our main objective was to evaluate the delay performance of NB-IoT under various system parameters measured at different locations. Our findings revealed that there is considerable variation in delay, even due to packet size, which may be attributed to non-linear behavior of the equipment. Additionally, we discovered a clear correlation between signal quality and delay, with extreme locations experiencing delays in the range of 5 to 10 seconds, while other locations experience delays ranging from 1 to 4 seconds.

Key-words: NB-IoT, Delay measurements, IoT, Internet of Things.

Abstract in lingua italiana

L'Internet delle Cose (IoT) è una tecnologia in rapida crescita che si prevede connetterà miliardi di dispositivi e abiliterà applicazioni innovative. Narrowband IoT (NB-IoT) è uno nuovo standard cellulare sviluppato dal 3rd Generation Partnership Project (3GPP) per supportare reti a bassa potenza e a vasta area per IoT. Questa tesi presenta uno studio della tecnologia NB-IoT e dei suoi scenari di implementazione, con un focus sulla misura delle caratteristiche di rete utilizzando un kit di valutazione dalla Quectel Wireless Solutions.

Sono descritti l'ambiente di lavoro e gli strumenti di misura utilizzati in questo studio, tra cui il codice Python per i comandi AT e la comunicazione con il server remoto. I dati sono stati raccolti in diverse condizioni di rete per valutare le prestazioni della tecnologia. I dati raccolti sono stati analizzati utilizzando tecniche come l'analisi dell'istogramma e l'analisi della correlazione dei ritardi con la qualità del segnale del canale, la potenza del segnale, il rapporto segnale-rumore e la qualità del segnale di riferimento ricevuto.

I risultati di questo studio forniscono informazioni sulle prestazioni delle reti NB-IoT e possono contribuire al miglioramento della loro implementazione e ottimizzazione. Il nostro obiettivo principale era quello di valutare le prestazioni del ritardo di NB-IoT in diverse condizioni di sistema misurate in diverse posizioni. I nostri risultati hanno rivelato che c'è una considerevole variazione nel ritardo, anche a causa delle dimensioni del pacchetto, che potrebbe essere attribuita al comportamento non lineare dell'attrezzatura. Inoltre, abbiamo scoperto una chiara correlazione tra la qualità del segnale e il ritardo, con posizioni estreme che sperimentano ritardi nell'intervallo da 5 a 10 secondi, mentre altre posizioni sperimentano ritardi nell'intervallo da 1 a 4 secondi.

Parole chiave: NB-IoT, Delay measurements, IoT, Internet of Things.

Contents

Contents

1. Introduction

The Internet of Things or IoT is a revolution's key technology. You have almost heard this term and are probably familiar with some IoT devices and technologies. IoT can transform businesses, makes cities greener, save energy, improve efficiency, make more informed decisions, and generate new revenue streams by connecting various devices and pieces of equipment through the Internet. [1]

1.1 What is IoT, and how does IoT work?

The Internet of Things (IoT) is a network of connected devices that are able to communicate with one another and provide data to users through the Internet. IoT devices have the ability to connect to the Internet and collect data using the sensors that are implemented. While an IoT device can be useful on its own, when combined with others, it becomes even more valuable.

Since the number of pieces of equipment that are connected to the internet increased dramatically, the size of the IoT expands more and more. The Internet of Things includes a variety of devices. It can encompass anything from electrical substations to buildings and infrastructure and factory machinery which the number of them increases daily basis. The Internet of Things is used by manufacturers, energy companies, city governments, and a variety of other types of organizations.

The Internet of Things enables you to collect data automatically from a variety of functions, such as how much energy a building's lighting consumes or how much water flows through a wastewater treatment plant. Through the Internet, IoT solutions and devices can transmit the data they collect to a central system. Managers can then use this data to help them make more informed decisions. By employing data analysis techniques, you can delve deeper into the data in order to uncover new insights and forecast future outcomes.

Moreover, you can use IoT technology to automate certain pieces of equipment and processes within your business. Intelligent sensors enable equipment to automatically adjust its operation to optimize energy consumption, traffic flow, and more. When they detect a particular input, smart sensors take action or signal for an action to occur.

Motion-activated lighting is a simple example. When those sensors detect movement, they activate the lights. Additionally, smart sensors can detect abnormal conditions or device operation and notify operators of potential issues. [2]

1.2 Different technologies of IoT

The different technologies of IoT can be categorized into several groups based on their functionality and application. Wireless communication technologies, cloud computing and data processing technologies, sensor and actuator technologies, security and privacy technologies, Industrial and automation technologies. Here we only study "Wireless communication technologies" which contains:

- Wi-Fi
- Bluetooth
- Zigbee
- LoRa
- NB-IoT
- Cat-M

and we focus on NB-IoT technology. [3]

1.3 What is NB-IoT?

Narrowband IoT (NB-IoT) is a cellular network technology designed for low-power IoT devices that require long battery life, low data rates, and long-range communication. It operates on a narrowband frequency, making it ideal for massive machine-type communication (mMTC) applications such as smart city infrastructure, remote monitoring and control, and industrial automation. NB-IoT uses advanced features such as power-saving modes, adaptive modulation, and channel coding to optimize performance and battery life for IoT devices. It provides wide area coverage and penetration through buildings and other obstacles, and its standardization and compatibility with existing cellular infrastructure make it a promising technology for the IoT, particularly in applications that require secure connectivity. [4]

1.4 Contribution of this work

Narrowband Internet of Things (NB-IoT) is a promising technology for the Internet of Things (IoT) that offers long-range connectivity with low power consumption and high coverage. [5] However, there is a need to evaluate the performance of NB-IoT technology in different situations to understand its suitability for IoT applications. In particular, there is a need to evaluate the delays of transmitting packets with different sizes in different situations to determine the impact of network congestion and other factors on the performance of NB-IoT. This study aims to address this need by evaluating the performance of Quectel NB-IoT board in different situations and comparing the delays for different packet sizes.

2. 3GPP

3GPP (Third Generation Partnership Project) is a global standards organization responsible for the development of wireless telecommunications standards, including those related to IoT technologies. The organization was formed in 1998 and is comprised of various telecommunications standards bodies from around the world. The first time IoT was introduced in 3GPP was with the release of 3GPP Release 12 in 2014, which included specifications for Machine-to-Machine (M2M) communications. The first introduction of NB-IoT in 3GPP was with the release of 3GPP Release 13 in 2016, which included specifications for narrowband IoT technology designed specifically for low-power, wide-area IoT applications. Since then, 3GPP has continued to develop and improve NB-IoT technology with subsequent releases, including Release 14 and Release 15. [6][7]

2.1 Improvements in releases:

- Release 13 (2016): This release defined the standard for NB-IoT as a new radio access technology for IoT devices. It introduced several features specifically designed for IoT, including ultra-narrowband operation, power-saving mode, and extended coverage in challenging environments. [8]

- Release 14 (2017): This release introduced several new features to enhance the performance and functionality of NB-IoT, such as higher data rates, improved coverage, and reduced latency. It also added support for new deployment scenarios, such as standalone operation, multi-operator support, and positioning services. [9]
- Release 15 (2018): This release introduced several enhancements to improve the efficiency and flexibility of NB-IoT, such as enhanced carrier aggregation, extended DRX (Discontinuous Reception), and more efficient signaling procedures. [10]

- Release 16 (2020): This release introduced several new features to address the emerging needs of IoT applications, such as higher reliability, better positioning accuracy, and support for massive IoT deployments. It also introduced support for new spectrum bands, including the 2.4 GHz band, to enable IoT operation in unlicensed spectrum. [11]

- Release 17 (under development): This release is expected to introduce further improvements and new features for NB-IoT, such as higher data rates, enhanced security, and support for new IoT services and applications. It is also expected to address new use cases, such as IoT operation in high-speed trains and satellite communications. [12]

2.2 NB-IoT Deployment Scenarios

NB-IoT is a radio interface implemented over the cellular licensed spectrum. It offers high deployment flexibility and integration with the existing architecture, minimizing costs and complexity at network and device sides, and providing performance in line with mMTC expectations. Below, NB-IoT operation modes, possible deployment strategies are explained. NB-IoT devices operate over either a 200 kHz GSM-like channel or an LTE physical resource block (PRB) of 180 kHz, allowing coexistence with both GSM and LTE [8]. The deployment scenario should be transparent to a user equipment (UE) when it is first turned on and searches for an NB-IoT carrier, similar to existing LTE UEs.

Three different operation modes are defined:

- **In Guard Band**: In guard-band mode of operation, NB-IoT will utilize new resource blocks within the guard-band of an LTE carrier. An illustration of this is shown in *Figure 1* It may be possible to allocate the NB-IoT PRB right next to the outer LTE PRB. This, however, will depend on the channel raster for NB-IoT. In addition, since the NB-IoT carrier has been placed in the LTE guard band, additional guard band for the adjacent carrier may be required.

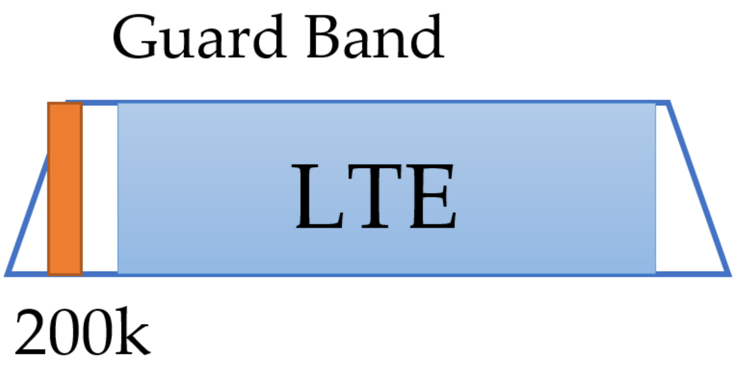


Figure 1 : Guard Band deployment NB-IoT

- **In Band**: For in-band operation, one or more PRBs are reserved for NB-IoT. This is shown in *Figure 2* where 1 PRB is reserved. Within this reserved region, NB-IoT signals must not be transmitted in time-frequency resources reserved for LTE. Sharing of PRBs between NB-IoT and LTE allows for more efficient use of the spectrum and seamless increase in NB-IoT capacity as more devices are added to the network.

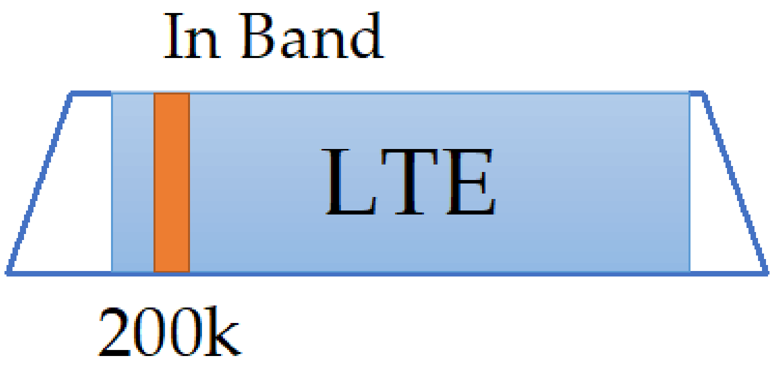


Figure 2 In Band deployment of NB-IoT

- **Stand Alone:** Standalone deployment mainly utilizes new bandwidth as shown in *Figure 3*. This option tends to offer the best performance in terms of improved indoor coverage. In standalone operation, NB-IoT can be used as a replacement of one or more GSM carriers since it occupies the same amount of bandwidth, 200 kHz. [13][14]

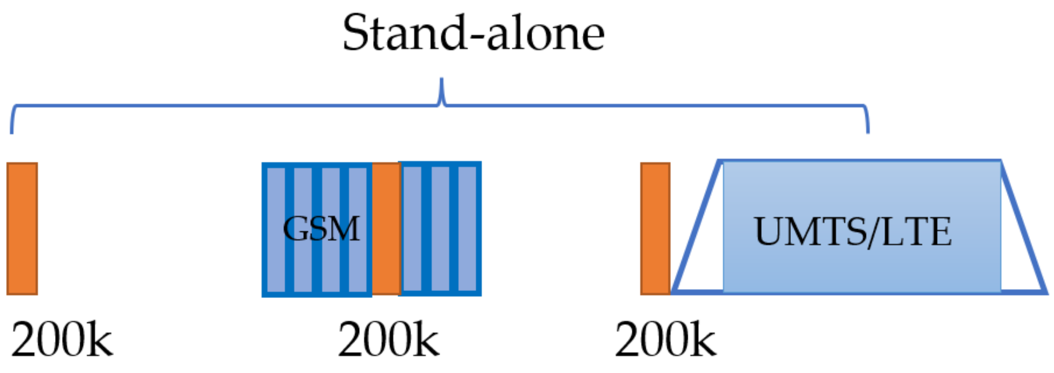


Figure 3 Stand Alone deployment NB-IoT

3. Description of work environment

In this chapter, the work environment for the NB-IoT technology evaluation will be described. The hardware utilized for the evaluation was the Quectel BC95 NB-IoT development board. The board features a Quectel BC95 module, which is a compact, high-performance module that supports both NB-IoT and eMTC (enhanced Machine-Type Communication) technologies. The board also features an ARM Cortex-M0 processor, which is used for running software applications. The software environment for the evaluation is Qcom, which is a development environment specifically designed for Quectel modules. Qcom provides a comprehensive set of tools for developing, testing, and debugging software applications for NB-IoT devices. In this chapter, the hardware and software components of the work environment will be described in detail, including the specifications of the Quectel BC95 development board and the features of the Qcom development environment.

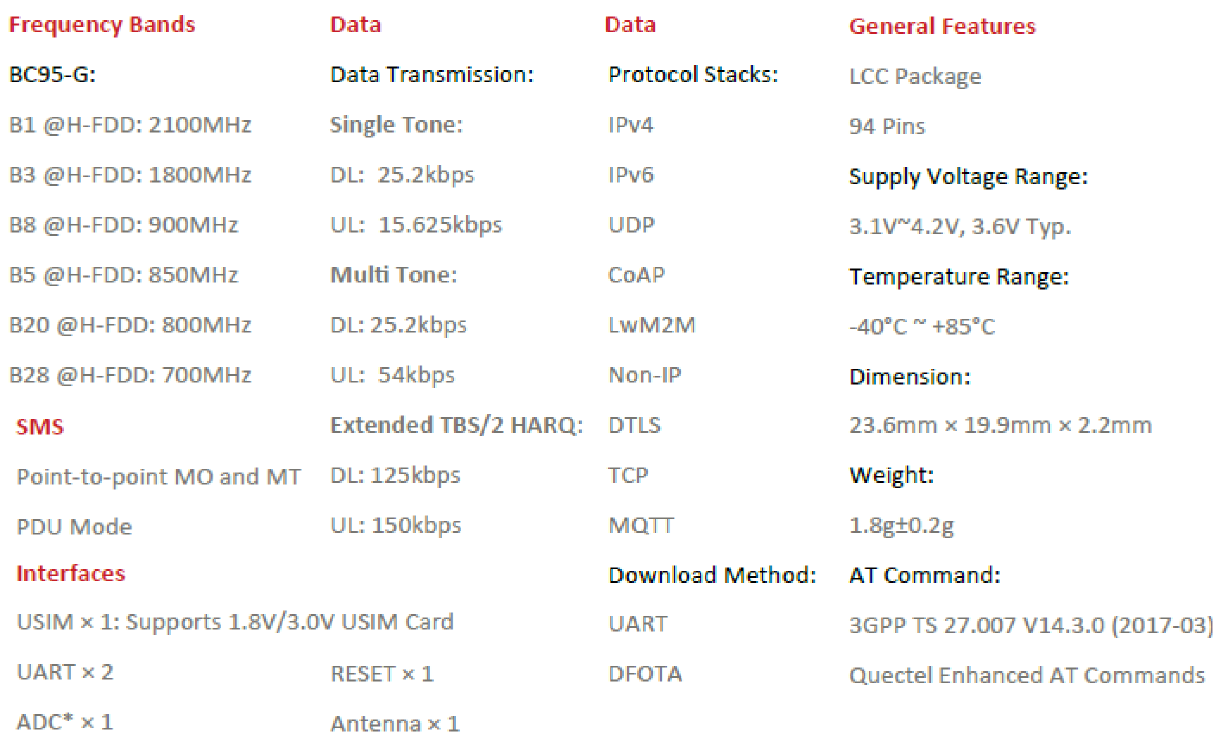
3.1 Vendor: Quectel Wireless Solutions

The work environment utilized for the evaluation of NB-IoT technology included the use of Quectel Wireless Solutions' cellular modules. Quectel is a global supplier of GSM/GPRS, UMTS/HSPA, LTE, LPWA, and GNSS modules, and is well-known for its expertise in IoT technology development. The cellular modules provided by Quectel are highly versatile and can be used in a wide range of IoT applications, including smart metering, asset tracking, wireless point-of-sale systems, and healthcare. For this particular experiment, the Quectel BC95 NB-IoT module was utilized, along with the UC15 GSM module. These modules are highly reliable and provide advanced cellular connectivity features that are ideal for low-power, wide-area IoT applications. Throughout the evaluation process, the Quectel cellular modules provided consistent and reliable connectivity, enabling the successful testing of NB-IoT technology in a variety of different scenarios. [15]

3.2 General Overview

The BC95 module from Quectel is a high-performance NB-IoT module that boasts extremely low power consumption, making it an ideal choice for IoT applications that require long battery life. Measuring just 23.6mm x 19.9mm x 2.2mm, the compact form factor of the BC95 makes it an excellent choice for size-sensitive applications. It is also designed to be compatible with the Quectel GSM/GPRS M95 module, providing a flexible and scalable platform for migrating from GSM/GPRS to NB-IoT networks. The BC95 module uses surface-mounted technology, which ensures its durability and ruggedness. Its low profile and small size LCC package enable easy integration into space-constrained applications, providing reliable connectivity with the applications. Due to its compact form factor, ultra-low power consumption, and extended temperature range, the BC95 module is an ideal choice for a wide range of IoT applications, including smart metering, bike sharing, smart parking, smart city, security, and asset tracking, home appliances, agricultural and environmental monitoring, and more. Additionally, the module can provide a complete range of SMS and data transmission services to meet the demands of client-side applications. General features can be seen in the ***Figure 4***. [16]

The BC95 NB-IoT module is capable of operating across multiple frequency bands, including 700 MHz, 800 MHz, 850 MHz, 900 MHz, 1800 MHz, and 2100 MHz. Its electrical characteristics set it apart from other technologies used for wireless data transmission. Notably, the module has a typical supply voltage of 3.6 V and is designed to operate within a wide temperature range of -40°C to +85°C. The standout feature of the BC95 module, however, is its exceptionally low power consumption. In standby mode, the module consumes a mere 5uA current, which is significantly lower than the current consumption of GSM/GPRS modules, which typically consume around 10-15mA current. This difference in power consumption allows the BC95 module to operate on battery power for up to 10 years. For this study, the "BC95-B20" module, which operates at 800 MHz, was utilized. It is shown on ***Figure 5***. [14]



Frequency Bands	Data	Data	General Features
BC95-G:	Data Transmission:	Protocol Stacks:	LCC Package
B1 @H-FDD: 2100MHz	Single Tone:	IPv4	94 Pins
B3 @H-FDD: 1800MHz	DL: 25.2kbps	IPv6	Supply Voltage Range:
B8 @H-FDD: 900MHz	UL: 15.625kbps	UDP	3.1V~4.2V, 3.6V Typ.
B5 @H-FDD: 850MHz	Multi Tone:	CoAP	Temperature Range:
B20 @H-FDD: 800MHz	DL: 25.2kbps	LwM2M	-40°C ~ +85°C
B28 @H-FDD: 700MHz	UL: 54kbps	Non-IP	Dimension:
SMS	Extended TBS/2 HARQ:	DTLS	23.6mm × 19.9mm × 2.2mm
Point-to-point MO and MT	DL: 125kbps	TCP	Weight:
PDU Mode	UL: 150kbps	MQTT	1.8g±0.2g
Interfaces		Download Method:	AT Command:
USIM × 1: Supports 1.8V/3.0V USIM Card		UART	3GPP TS 27.007 V14.3.0 (2017-03)
UART × 2	RESET × 1	DFOTA	Quectel Enhanced AT Commands
ADC* × 1	Antenna × 1		

Figure 4 General features for BC95 NB-IoT module

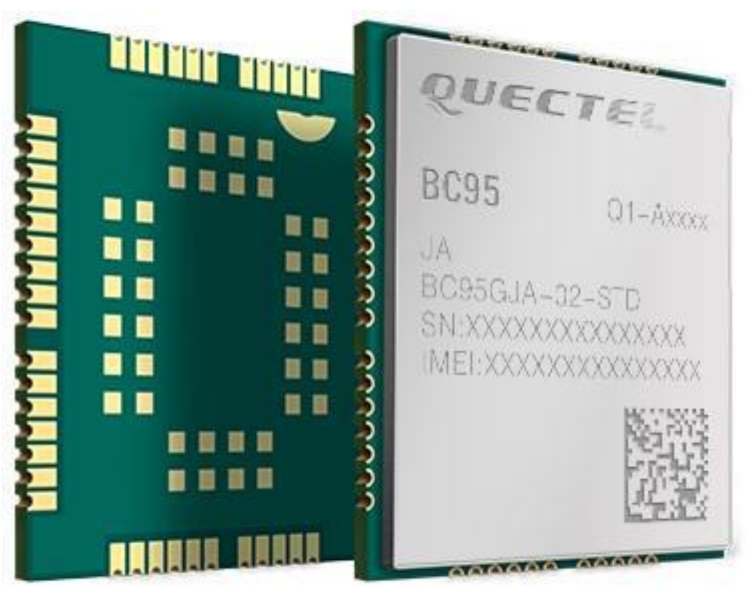


Figure 5 B20 module

3.3 The Evaluation Board Kit

1. The NB-IOT EVB bottom and top view are depicted in *Figure* 6 and Figure 7.

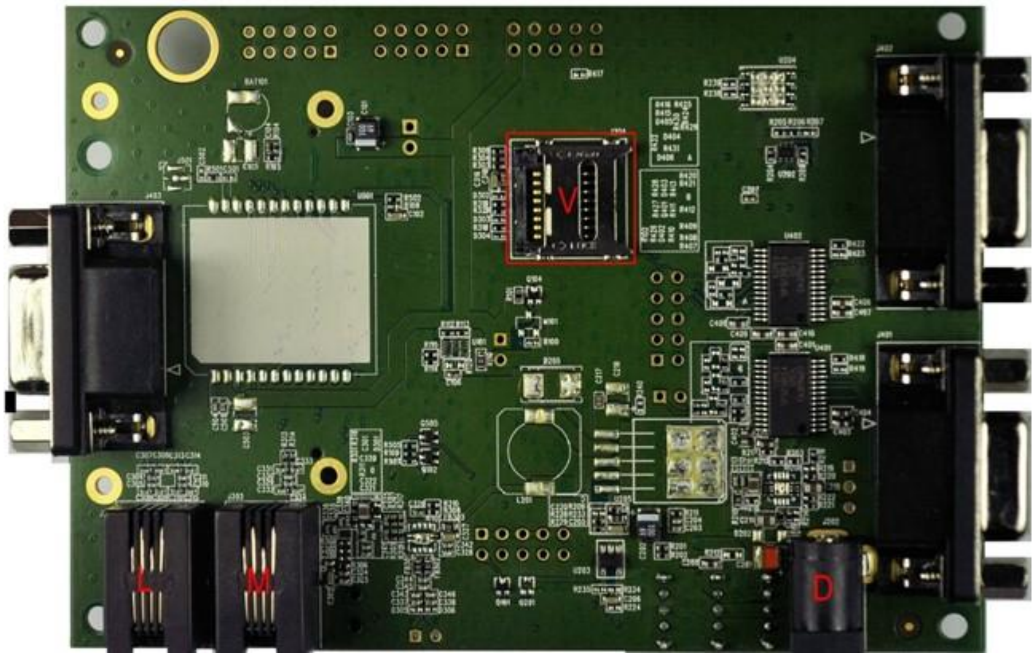


Figure 6 GSM EVB Bottom View

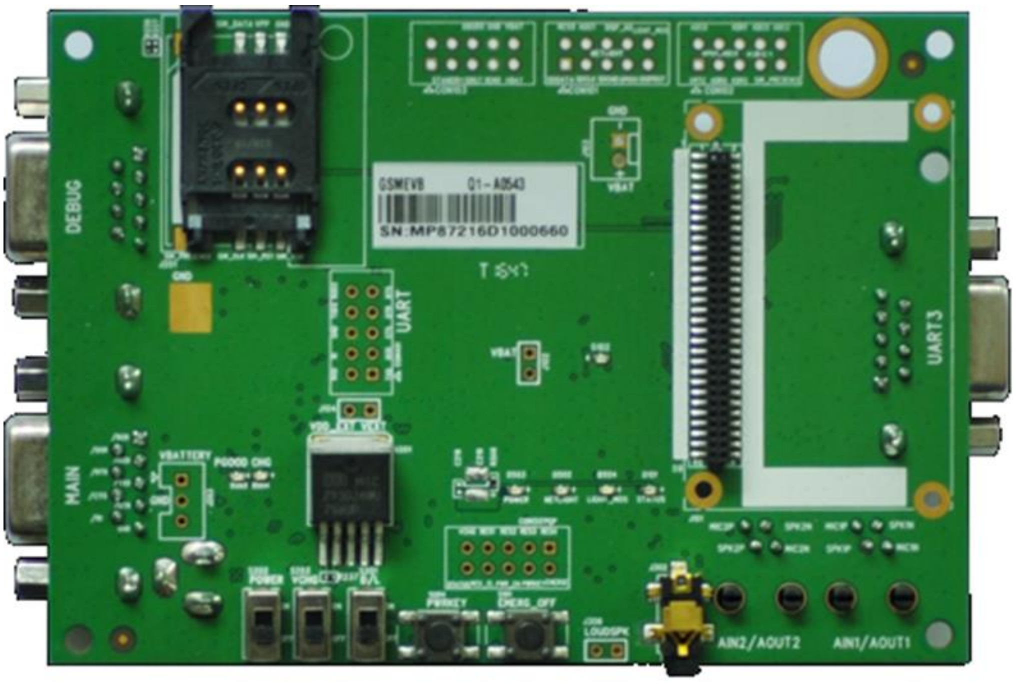


Figure 7 NBIOT EVB top view

2. **Cables:**

- USB to UART converter cable
- USB cable
- RF cable

3. Antenna

4. Adapter - 5V DC adapter

5. Audio – Earphone

6. Disk - Disk involving related documents and drivers

7. Instruction sheet - A sheet of paper giving instructions for EVB connection, details of EVB accessories, etc. See *Figure 8*.

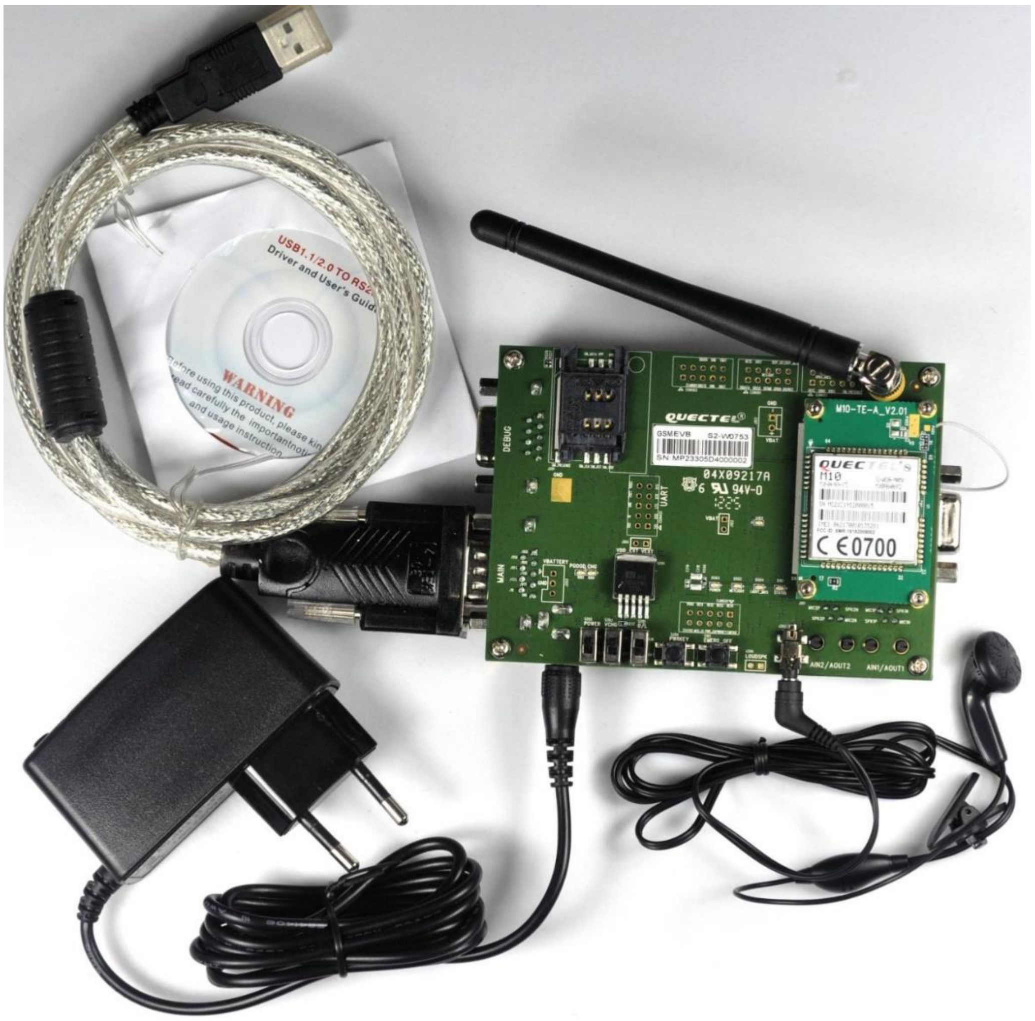
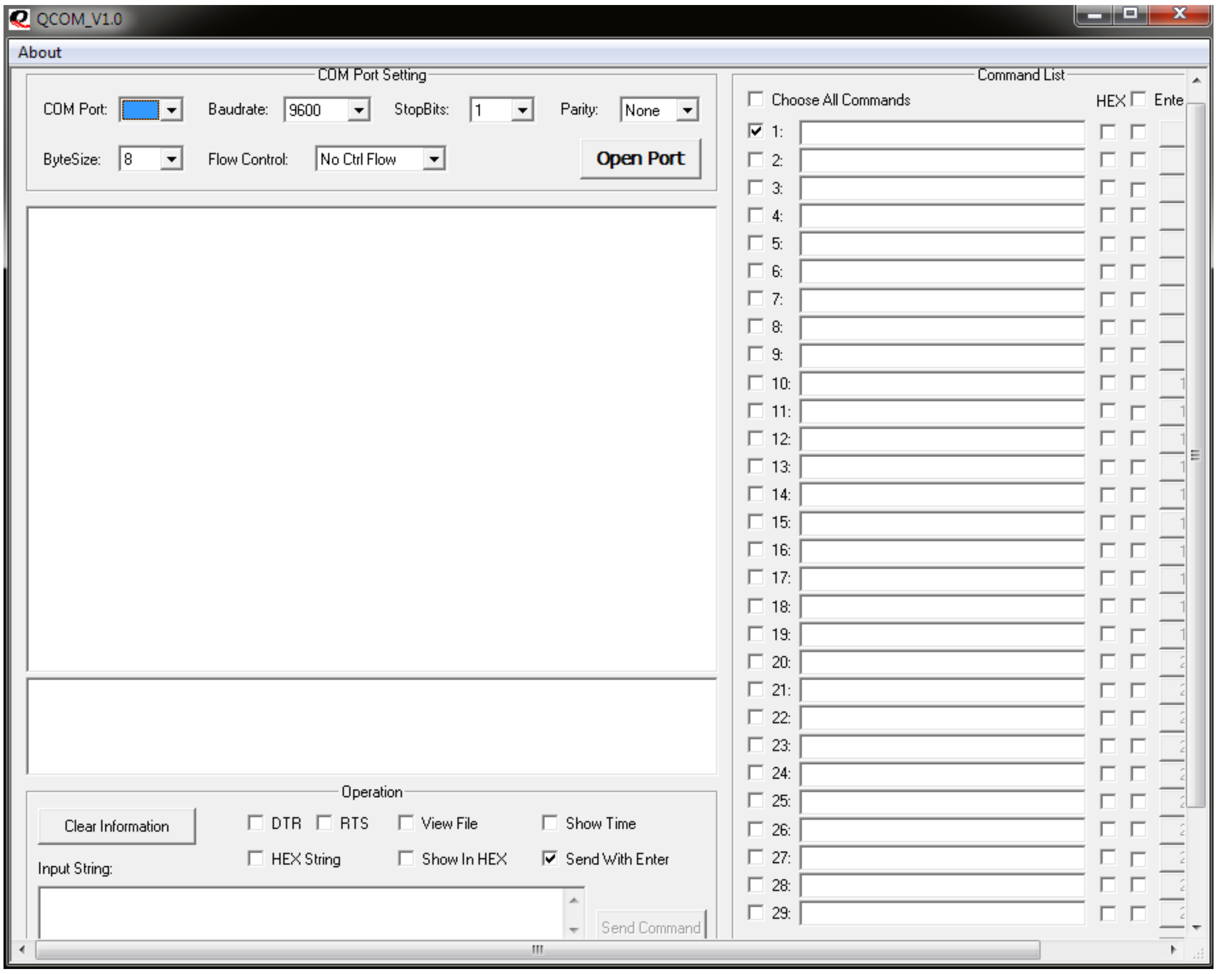


Figure 8 EVB and Accessories

3.4 Software

The BC95 module is accompanied by a software package known as QCOM. This software package provides a user-friendly interface for sending AT commands to the module, and its functionality is illustrated in *Figure 9*. By utilizing this software, users can easily configure and control the BC95 module to suit their specific requirements. This software is an essential component of the BC95 module, and its user-friendly interface simplifies the process of interfacing with the module, allowing for efficient and effective utilization of its capabilities. [16]



COM Port:
Baudrate: 9600
StopBits: 1
Parity: None
ByteSize: 8
Flow Control: No Ctrl Flow

[✓] 1:

[✓] Send With Enter

Figure 9 Qcom software view

Before using the QCOM software to communicate with the BC95 module, proper installation and configuration of the Com port settings are crucial. The success of the installation can be confirmed through the Device Manager of the computer. After connecting the module to the computer via the serial port, the COM Port will be automatically identified through the Operating System of the computer. The Com port settings consist of six parameters that need to be set appropriately: COM port, Baud rate, Stop bits, Parity, Byte size, and Flow control.

The COM port parameter identifies the serial port number that the device is connected to, such as COM1, COM2, COM3, etc. The recommended setting for the BC95 module is to use the serial port number that it is connected to. The Baud rate parameter refers to the speed at which data is transmitted and received over the serial port, and the recommended value for the BC95 module is 9600 bps. The Stop bits parameter specifies the number of bits used to signal the end of a character, and the recommended value for the BC95 module is 1.

The Parity parameter provides a method of error detection in serial communication and can be set to None, Even, Odd, Mark, or Space. For the BC95 module, the recommended value is None. The Byte size parameter specifies the number of data bits that make up each character, and the recommended value for the BC95 module is 8 bits. The Flow control parameter determines how data flow is controlled between the device and the computer and can be set to None, Hardware, or Software control flow. The recommended value for the BC95 module is None.

Once the Com port settings have been properly configured, the "Open Port" option can be selected in the QCOM software, and the Input String can be used to send AT commands effectively.

3.5 AT Commands

The effective operation of the Dev Board, which integrates the BC95 module, relies on communication via a serial line. To operate correctly, the Dev Board must be connected to a device with the appropriate software installed, and AT commands must be transmitted from this device to the board. The Dev Board can only receive commands from the serial line, and a specific set of AT commands must be transmitted to register the BC95 module to a network and enable it to transmit packets successfully. While the specific set of AT commands required may vary depending on the specific use case and network configurations, a typical set includes commands for network registration, setting the APN, defining the PDP context, activating the PDP context, setting the IP address, and initiating data transfer. It is essential to enter the AT commands accurately and in the correct sequence to ensure successful packet transmission, as the successful transmission of these commands is crucial for the proper operation of the Dev Board. [17][18]

1. AT+NRB: This command stands for "Network Reboot" and instructs the BC95 module to restart the network. It is often used to reset the module when it encounters an error or malfunction.

2. AT+CFUN=1: This command sets the module's function level to 1, which enables full functionality. This is often used to activate the module after it has been powered on or reset.

3. AT+CGDCONT=0,"IP","nb.xxxx.gdsp": This command is used to set the APN (Access Point Name) and the PDP (Packet Data Protocol) context for the module which in this case is Vodafone

4. AT+CEREG=2: This command configures the module to report cell registration and location information. The value "2" indicates that the module should report when it is registered on a network and when the location information changes.

5. AT+CSCON=1: This command enables the module to report the state of the circuit-switched domain. The value "1" indicates that the module should report when the circuit-switched domain is attached or detached.

6. AT+COPS=1,2,"22xxx": selects and registers the EPS network operator using the USIM card in the currently selected card slot. It has three parameters, with the first indicating manual network operator selection. The second parameter specifies numeric operator identification, followed by the operator identification number as the third parameter. This command is crucial for successful network registration and proper BC95 module operation.

7. AT+CSQ: This command is used to check the signal strength of the module's connection to the network. It returns a value between 0 and 31, with higher values indicating a stronger signal.

8. AT+NUESTATS: This command retrieves various statistics about the module's network connection, including the signal quality, network registration status, and data usage.

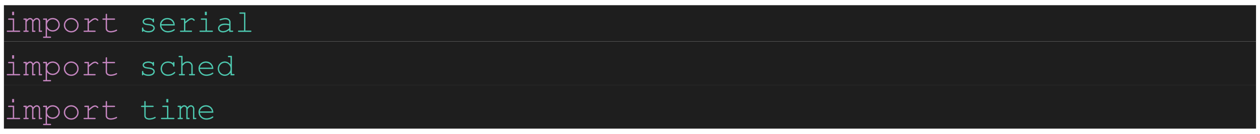
9. AT+NSOCR=DGRAM,17,3365,1: This command creates a UDP socket for the module to send and receive datagrams. The values "DGRAM" and "17" specify that the socket should be a datagram socket using the UDP protocol, and "3365" is the port number to be used for the socket.

10. AT+NSOST=0,131.175.120.22,8883,2,4f4b: This command sends data over the UDP socket created in the previous command. The values "0" and "131.175.120.22" specify that the data should be sent using the socket with ID "0" to the IP address "131.175.120.22," and "8883" is the port number to which the data should be sent. The remaining values "2" and "4f4b" specify the length of the data to be sent and the data itself, respectively.

3.6 Python Code of AT commands

In this work, we automated the procedure of testing the board using a custom-written code. This code replaced the Qcom software that was previously used for this purpose. The code uses Python's serial and scheduling libraries to open the serial port and connect the board to the network, send UDP packets, and close the socket and serial port when the testing is completed. The code simplifies the testing procedure by automating the repetitive tasks, saving time and effort. It also provides more flexibility and customization options compared to the Qcom software.

The first part of the code which involves importing necessary libraries and modules is a crucial step in automating the procedure of measurements which is shown in the *Figure 10*.



```
import serial
import sched
import time
```

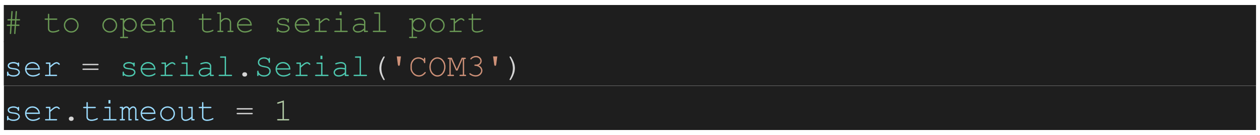
Figure 10 importing necessary libraries and modules.

import serial - This line imports the Python serial module, which provides access to the serial communication ports on a computer. The serial module allows Python programs to communicate with external devices (such as microcontrollers, sensors, and other hardware) over serial connections. [19]

import sched - This line imports the Python sched module, which provides a simple interface for scheduling tasks to run at specific times or intervals. The sched module is often used in combination with other Python libraries (such as time and datetime) to create timed events or periodic tasks. [20]

import time - This line imports the Python time module, which provides various time-related functions and data types. The time module can be used to measure elapsed time, delay program execution, generate timestamps, and perform other time-based operations in Python programs. [21]

The next step involves creating a serial object to connect to the device through the serial port. The 'serial' library is used for this purpose. In Windows, the COMx represents the number of the port that the system automatically allocates for the serial connection, whereas for Linux, it is `'/dev/ttyUSB0'`. The 'timeout' attribute sets the time in seconds to wait for data from the serial port. The code is represented in *Figure 11*.



```
# to open the serial port
ser = serial.Serial('COM3')
ser.timeout = 1
```

Figure 11 creating a serial object to connect to the device through the serial port.

The next part of the code is used to connect the board to the network using AT commands. The AT commands are sent as strings to the board through the serial connection. Each AT command is used to configure a specific aspect of the connection process. The following is a brief explanation of each AT command:

- AT+NRB: This command is used to reset the board to its default settings.
- AT+CFUN=1: This command sets the functionality level of the board to full functionality.
- AT+CGDCONT=1,"IP","nb.xxxx.gdsp": This command sets the APN (Access Point Name) to "nb.xxxx.gdsp".
- AT+CEREG=2: This command sets the board to register on the network and to automatically re-try registration if it fails.
- AT+CSCON=1: This command sets the board to automatically connect to the network.
- AT+COPS=1,2,"22xxx": This command sets the board to select the network operator with the MCC-MNC code "22xxx".
- AT+CSQ: This command is used to check the signal strength of the network.
- AT+NUESTATS: This command is used to check the network status.

The Python code is shown in ***Figure 12***.

```
# to connect the board to the network
ser.write(b'AT+NRB\R\N')
ser.write(b'AT+CFUN=1\r\n')
ser.write(b'AT+CGDCONT=1,"IP","nb.inetd.gdsp"\r\n')
ser.write(b'AT+CEREG=2\r\n')
ser.write(b'AT+CSCON=1\r\n')
ser.write(b'AT+COPS=1,2,"22210"\r\n')
ser.write(b'at+csq\r\n')
ser.write(b'at+nuestats\r\n')
```

Figure 12 the code to connect the board to the network using AT commands.

In the next step, the code sets up a socket to send UDP packets using the NB-IoT module. The **`ser.write()`** function sends an AT command to the module to open a socket using the command **`AT+NSOCR=DGRAM,17,3365,1\r\n`**. The command specifies that a UDP socket should be created (DGRAM), the protocol should be UDP (17), the local port number should be 3365, and the socket should be created in non-blocking mode (1). This command will return a socket ID which will be used in subsequent commands to send data through the socket. In summary, this code sets up a UDP socket to enable the NB-IoT module to send packets of data over the network see *Figure 13*.

```
#send UDP packets , first open the socket, then send UDP message
ser.write(b'AT+NSOCR=DGRAM,17,3365,1\r\n')
```

Figure 13 sets up a socket to send UDP packets.

The next line of the code creates a scheduler object using the sched module. This object is named s and it will be used later to schedule the sending of UDP packets at a specific interval of time. The **`sched.scheduler`** function takes two arguments: the first argument is a function that returns the current time, and the second argument is a function used for delaying a certain amount of time before executing a task. In this case, the functions **`time.time`** and **`time.sleep`** are passed as arguments respectively. The related code can be seen in the *Figure 14*.

```
s = sched.scheduler(time.time, time.sleep)
```

Figure 14 creates a scheduler object using the sched module.

The main part of the code is responsible for sending a packet via the NB-IoT network. It does so by defining a function **`do_something`** which retrieves the current timestamp using **`time.time()`** and then sends a 2-byte packet containing the value **`"OK"`** via the **`ser.write()`** function. The **`s.enter()`** function specifies the interval at which the packet is sent, while **`s.run()`** is responsible for starting the scheduler and allowing the packet to be sent at the specified interval. The IP address and port number to which the packet is sent are specified in the **`ser.write()`** function, with the values **`'131.175.120.22'`** and **`'8883'`** respectively.

This last part of the code is responsible for closing the socket and disconnecting the board from the network. The first line **`ser.write(b'AT+NSOCL=0\r\n')`** sends an AT command to close the socket which was opened before to send UDP packets. The argument 0 in **`AT+NSOCL=0`** refers to the socket ID, which in this case is 0 since there is only one socket used in the code. The second line **`ser.close()`** closes the serial port connection which was established at the beginning of the code using **`serial.Serial()`** function. This step is important to ensure that there is no data loss or corruption during the connection termination process. The way it should be used is shown in the *Figure 15*.

```
# to close the socket
ser.write(b'AT+NSOCL=0\r\n')
# close the serial port (disconnect the board)
ser.close()
```

Figure 15 closing the socket and disconnecting the board from the network.

4. Measurements tools

4.1 PySerial

An RS232 to USB cable is a type of cable that enables the connection of devices using serial communication protocols via a USB port. This type of cable is commonly used to connect legacy devices that use RS232 serial communication to modern computers or laptops. The cable includes a USB interface on one end and a DB9 or DB25 serial port interface on the other end. The DB9 or DB25 connector plugs into the device using the RS232 communication protocol, while the USB connector plugs into the computer or laptop.

The RS232 protocol is a standard communication protocol used for serial communication between devices. It defines the electrical signals, timing, and data format used in serial communication. The RS232 protocol defines the number of data bits, the number of stop bits, and the parity bit used for error checking. The protocol specifies the use of a fixed baud rate, which determines the speed at which data is transmitted between devices.

Using the PySerial library and the RS232 to USB cable, the developed python code is able to communicate with the development board and automate the measurement procedure. PySerial is a Python module used to interact with the serial port, allowing the user to read and write data from and to the serial port. PySerial provides support for different operating systems, including Windows, Linux, and macOS, making it a versatile and widely-used module for serial communication. In addition, PySerial supports different protocols, including the RS232. The use of PySerial and the RS232 to USB cable allowed for a reliable and efficient communication between the laptop and the development board, enabling the successful implementation of the measurement procedure. [22]

4.2 SSH protocol

Secure Shell (SSH) is a widely used network protocol for secure communication between two systems. It provides a secure channel over an unsecured network by encrypting all data transmitted between the client and the server. The SSH protocol has several versions, with the most commonly used being SSH-2.

SSH protocol is typically used for remote login to a computer or server, allowing users to securely execute commands on the remote system. It can also be used for secure file transfer and tunneling other network services. SSH is supported by most Unix-based operating systems, as well as Windows through third-party software.

To establish an SSH connection, the client and the server must both have SSH software installed. The client initiates the connection by specifying the server's IP address or domain name and providing authentication credentials such as a username and password or a public key. Once the connection is established, all communication between the client and the server is encrypted, providing a secure and private communication channel.

In summary, SSH is a widely used protocol for secure communication between two systems over an unsecured network. It provides a secure channel for remote login, file transfer, and tunneling of other network services. The protocol encrypts all data transmitted between the client and the server, ensuring a secure and private communication channel. [23]

BitVise software is utilized in this work to establish a communication channel between the laptop and the server for measuring delays in the network. The reception time of these packets can be viewed through BitVise software, facilitating the calculation of network delays by knowing the time of packet transmission and reception. Therefore, BitVise software serves as a crucial tool in this work for enabling the accurate measurement of network delays. [24]

4.3 Communication Design with Dev Board

The control of the Development Board via the serial line is performed by the PC. To eliminate the need for proprietary software such as "Q-COM" from Quectel to control the Dev Board, the PySerial library in Python is utilized. Upon issuing a sequence of commands, the board becomes registered and connected to the nearest Base Station (eNode B) within the LTE Network.

Additionally, a direct connection between the PC and the bonsai_16 machine is established, which serves as the Remote Server. This is achieved using an SSH connection on port 22, allowing for remote login and monitoring of the server's activity. It is depicted in the *Figure 16*.

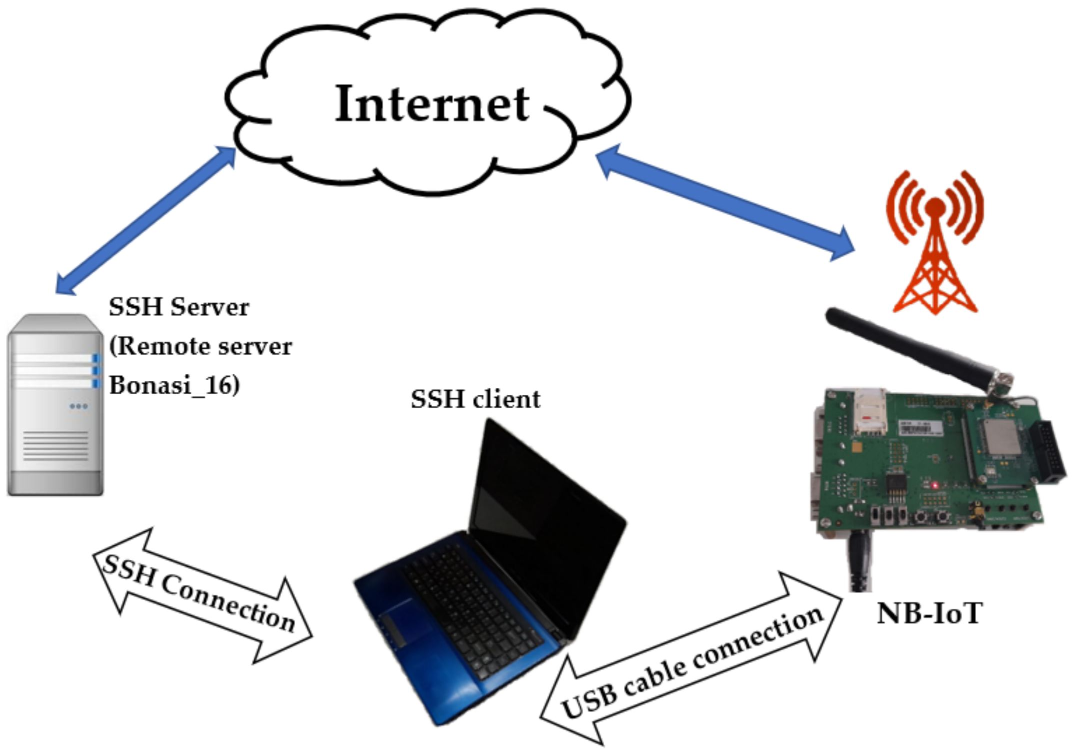
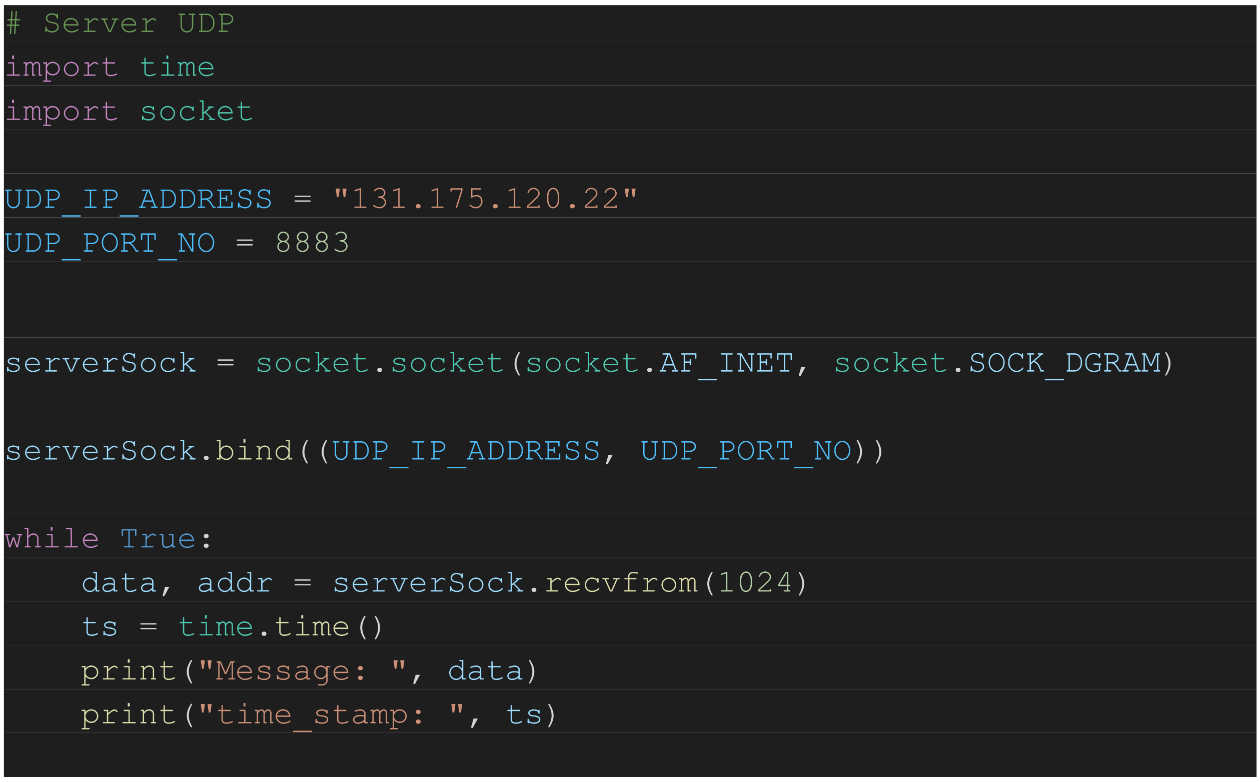


Figure 16 Communication Design with Dev Board. [25]

4.4 Python code on the remote server

In order to accurately measure the delays between the transmission and reception of UDP packets sent from the Development Board to the remote server, a Python code was developed to receive the incoming packets and determine their arrival timestamps. This section will discuss the details of this code, including its structure and functionality, as well as its role in the measurement process.



```
# Server UDP
import time
import socket

UDP_IP_ADDRESS = "131.175.120.22"
UDP_PORT_NO = 8883


serverSock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

serverSock.bind((UDP_IP_ADDRESS, UDP_PORT_NO))

while True:
    data, addr = serverSock.recvfrom(1024)
    ts = time.time()
    print("Message: ", data)
    print("time_stamp: ", ts)
```

Figure 17 Python code on the remote server.

The code provided in *Figure 17* is written in Python and is used to receive UDP packets on a specific IP address and port number.

The first step is to import the necessary modules for creating the UDP socket and measuring the timestamp of incoming packets. The code imports the "**`time`**" and "**`socket`**" modules.

Next, the code initializes two variables "**`UDP_IP_ADDRESS`**" and "**`UDP_PORT_NO`**" to the IP address and port number where the server will listen for incoming packets. After that, the code creates a UDP socket using the "**`socket.socket()`**" function and sets the socket type to "**`SOCK_DGRAM`**" for UDP packets. The code then binds the socket to the IP address and port number using the "**`serverSock.bind()`**" function.

The code then enters an infinite loop, listening for incoming UDP packets using the "**`serverSock.recvfrom()`**" function. When a packet arrives, the code stores the data and address of the sender in the "**`data`**" and "**`addr`**" variables, respectively. The code also measures the timestamp of the incoming packet using the "**`time.time()`**" function and stores it in the "**`ts`**" variable.

Finally, the code prints the received message and the corresponding timestamp using the "**`print()`**" function. Overall, this code allows the user to receive UDP packets and measure their arrival time using Python.

5. Measurements

5.1 Characteristics of the network

In this section, the findings of the measurements will be presented and analyzed. To begin with, the characteristics of the network were investigated using the "**`at+csq`**" and "**`at+nuestats`**" AT commands. The information obtained from these commands include:

The response to **`AT+CSQ`** is **`+CSQ: XX,YY`**

CSQ stands for "Channel Signal Quality" and represents the received signal strength indicator. XX value ranges from 0 to 31. *Table 1* represents these indications. YY is the signal quality index. It is a measurement of the bit error rate (BER), which is the number of bits that are received incorrectly over a certain period. The lower the value, the better the signal quality. The value of 99 indicates that the signal quality is not known or not detectable.

CSQ	Received Signal (dBm)	CSQ	Received Signal (dBm)	CSQ	Received Signal (dBm)
0	-113	**11**	-91	**22**	-69
1	-111	**12**	-89	**23**	-67
2	-109	**13**	-87	**24**	-65
3	-107	**14**	-85	**25**	-63
4	-105	**15**	-83	**26**	-61
5	-103	**16**	-81	**27**	-59
6	-102	**17**	-79	**28**	-57
7	-99	**18**	-77	**29**	-55
8	-97	**19**	-75	**30**	-53
9	-95	**20**	-73	**31**	-51
10	-93	**21**	-71	**99**	unknown

Table 1 Received Signal Strength Indicators (RSSI) for CSQ values.

The response to the **`AT+CESTATS`** command provides more information about the network and connectivity parameters of the NB-IoT board. Here is an explanation of the different parameters:

- **Signal power**: This represents the received signal power in centibels (the power ratio of a signal to a carrier signal expressed as dBc). A higher value indicates a stronger signal.
- **Total power**: This represents the total power in dBc that is being transmitted by the cell tower.
- **TX power**: This is the transmission power in dBc that is being used by the NB-IoT board.
- **TX time**: This represents the total time in milliseconds that the NB-IoT board has been transmitting data.

- **RX time**: This represents the total time in milliseconds that the NB-IoT board has been receiving data.
- **Cell ID**: This is the unique identification number of the cell tower that the NB-IoT board is currently connected to.
- **ECL**: This stands for "EC/N0 Cell Level" and represents the signal quality in terms of the ratio of energy per chip to background noise. The value ranges from 0 to 31, with 0 indicating the best signal quality.
- **SNR**: This stands for "Signal-to-Noise Ratio" and represents the ratio of the signal power to the noise power. A higher value indicates a better signal quality.
- **EARFCN**: This is the "E-UTRA Absolute Radio Frequency Channel Number" and represents the channel frequency that is being used by the NB-IoT board.
- **PCI**: This stands for "Physical Cell ID" and represents the unique identification number of the physical cell that the NB-IoT board is currently connected to.
- **RSRQ**: This stands for "Reference Signal Received Quality" and represents the quality of the received reference signal. A higher value indicates a better signal quality. The value is represented in dBc.

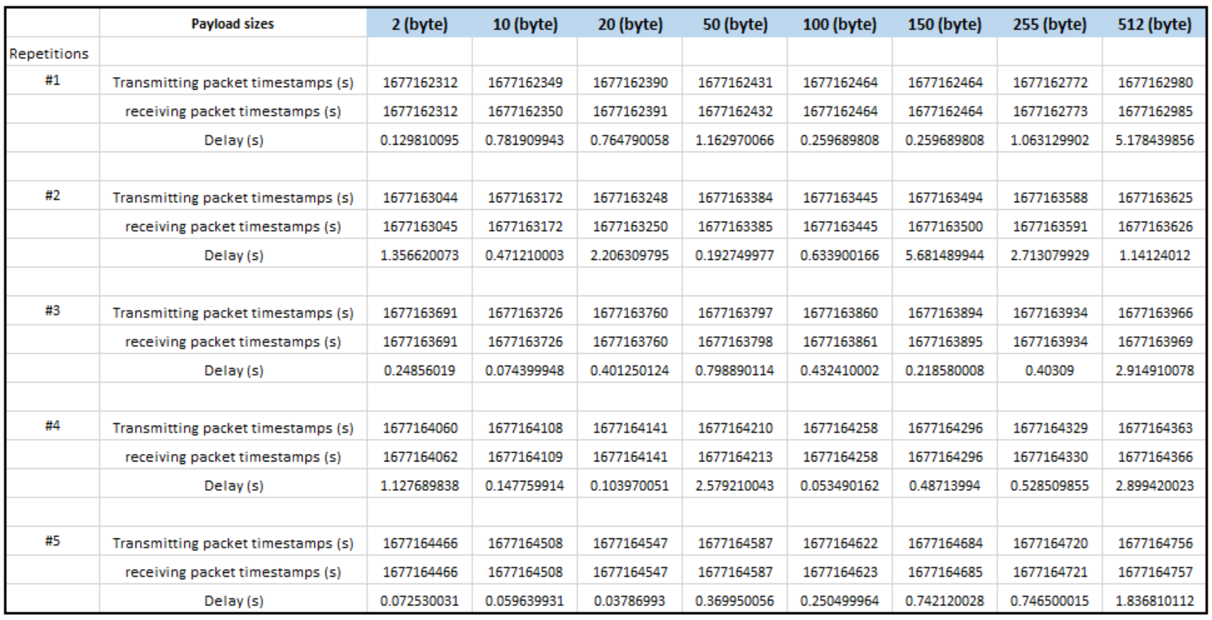
There are several parameters that are crucial and can impact the delay in IoT communication systems. These parameters include signal power, total power, TX power, SNR, and RSRQ.

5.2 Dataset

In this study, a total of 30 measurements were carried out under different network conditions to collect data on various network parameters. The dataset gathered during these measurements will be presented in this section. The dataset includes information on the time of measurement, location, delays for different packet sizes, and network characteristics.

5.2.1 Delays

For each measurement, the time, location, and network parameters were recorded, and a Python code was utilized to transmit packets of different sizes (2, 10, 20, 50, 100, 150, 255, 512 bytes) to the server. Note that the maximum length of received data with the Quectel BC95 module is 512 bytes. The transmission and receiving timestamps were then recorded, and the delay between them was calculated. To ensure the accuracy and reliability of the delay measurements, each packet was transmitted and recorded 20 times during each measurement. *Table 2* shows the transmitting and receiving timestamps, as well as the corresponding delays for each packet size in one of the experiments.



	Payload sizes	2 (byte)	10 (byte)	20 (byte)	50 (byte)	100 (byte)	150 (byte)	255 (byte)	512 (byte)
Repetitions									
#1	Transmitting packet timestamps (s)	1677162312	1677162349	1677162390	1677162431	1677162464	1677162464	1677162772	1677162980
	receiving packet timestamps (s)	1677162312	1677162350	1677162391	1677162432	1677162464	1677162464	1677162773	1677162985
	Delay (s)	0.129810095	0.781909943	0.764790058	1.162970066	0.259689808	0.259689808	1.063129902	5.178439856
#2	Transmitting packet timestamps (s)	1677163044	1677163172	1677163248	1677163384	1677163445	1677163494	1677163588	1677163625
	receiving packet timestamps (s)	1677163045	1677163172	1677163250	1677163385	1677163445	1677163500	1677163591	1677163626
	Delay (s)	1.356620073	0.471210003	2.206309795	0.192749977	0.633900166	5.681489944	2.713079929	1.14124012
#3	Transmitting packet timestamps (s)	1677163691	1677163726	1677163760	1677163797	1677163860	1677163894	1677163934	1677163966
	receiving packet timestamps (s)	1677163691	1677163726	1677163760	1677163798	1677163861	1677163895	1677163934	1677163969
	Delay (s)	0.24856019	0.074399948	0.401250124	0.798890114	0.432410002	0.218580008	0.40309	2.914910078
#4	Transmitting packet timestamps (s)	1677164060	1677164108	1677164141	1677164210	1677164258	1677164296	1677164329	1677164363
	receiving packet timestamps (s)	1677164062	1677164109	1677164141	1677164213	1677164258	1677164296	1677164330	1677164366
	Delay (s)	1.127689838	0.147759914	0.103970051	2.579210043	0.053490162	0.48713994	0.528509855	2.899420023
#5	Transmitting packet timestamps (s)	1677164466	1677164508	1677164547	1677164587	1677164622	1677164684	1677164720	1677164756
	receiving packet timestamps (s)	1677164466	1677164508	1677164547	1677164587	1677164623	1677164685	1677164721	1677164757
	Delay (s)	0.072530031	0.059639931	0.03786993	0.369950056	0.250499964	0.742120028	0.746500015	1.836810112

Table 2 Transmitting and receiving packet timestamps (in seconds) and their corresponding delays (in seconds) for experiment 2, which investigates the performance of the system under test. Experiment 2 comprises of 20 repetitions, and this table presents the transmitting and receiving packet timestamps and the resulting delay for five selected repetitions, as an illustrative example. The delay is calculated as the difference between the transmitting and receiving packet timestamps, both measured in seconds. The payload sizes are represented in bytes.

As delays are of primary interest in our analysis, we will focus exclusively on the delay measurements moving forward. An example of the delay measurements is provided in *Table 3*.

Payload size	2 (byte)	10 (byte)	20 (byte)	50 (byte)	100 (byte)	150 (byte)	255 (byte)	512 (byte)
Delay 01 (s)	0.12981	0.78191	0.76479	1.16297	0.25969	0.25969	1.06313	5.17844
Delay 02 (s)	1.35662	0.47121	2.20631	0.19275	0.6339	5.68149	2.71308	1.14124
Delay 03 (s)	0.24856	0.0744	0.40125	0.79889	0.43241	0.21858	0.40309	2.91491
Delay 04 (s)	1.12769	0.14776	0.10397	2.57921	0.05349	0.48714	0.52851	2.89942
Delay 05 (s)	0.07253	0.05964	0.03787	0.36995	0.2505	0.74212	0.7465	1.83681
Delay 06 (s)	1.34229	0.49099	0.15906	0.97731	0.41714	0.44721	0.28799	2.14692
Delay 07 (s)	0.27332	1.58739	0.27386	0.5529	0.18117	0.05989	0.76819	2.94459
Delay 08 (s)	2.23023	0.00175	0.15303	1.53444	0.50217	0.27687	3.11903	1.84097
Delay 09 (s)	0.20411	0.36889	0.21651	2.63164	0.64719	1.94938	1.01259	1.32982
Delay 10 (s)	0.7436	0.31214	0.18563	0.33709	0.64419	0.29162	0.83217	2.31599
Delay 11 (s)	0.61744	0.19072	0.44702	0.34491	0.80216	3.29296	0.65597	3.26385
Delay 12 (s)	0.15804	1.63985	0.36098	0.53514	0.73263	0.44884	2.3658	2.80867
Delay 13 (s)	0.84146	0.58549	0.39131	1.9867	0.64459	1.01589	0.77215	1.48285
Delay 14 (s)	2.64376	0.42782	0.42203	0.5196	0.77931	1.87166	0.86919	2.24272
Delay 15 (s)	0.80143	0.60355	0.25006	0.67214	0.66893	1.03577	1.68391	3.93886
Delay 16 (s)	0.574	0.52201	0.27922	0.66993	0.47283	2.78789	1.52128	2.26865
Delay 17 (s)	0.7806	1.71321	0.95323	0.96198	0.56545	2.93928	3.04256	3.22194
Delay 18 (s)	0.63179	2.34762	0.69243	1.70328	1.07618	0.78142	1.48134	2.14742
Delay 19 (s)	0.61409	0.97561	0.87355	0.96344	3.73413	5.24126	1.18494	2.56265
Delay 20 (s)	1.0369	1.07387	0.82766	0.66114	0.85825	0.7993	1.01768	5.23479

Table 3 Delays for different packet sizes in experiment 2. This table shows the delays, measured in seconds, for different payload sizes in bytes, obtained in experiment 2. The delays are computed as the difference between the transmitting and receiving packet timestamps, for 20 repetitions of the experiment 2.

To identify any outliers and ensure the reliability of the results, a box plot was generated for each set of delay measurements. The outliers were removed from the dataset to obtain a cleaned data set, which is used for further analysis and presentation of results. As an example, the Box Plot of Experiment 2 is presented in the *Figure 18*.

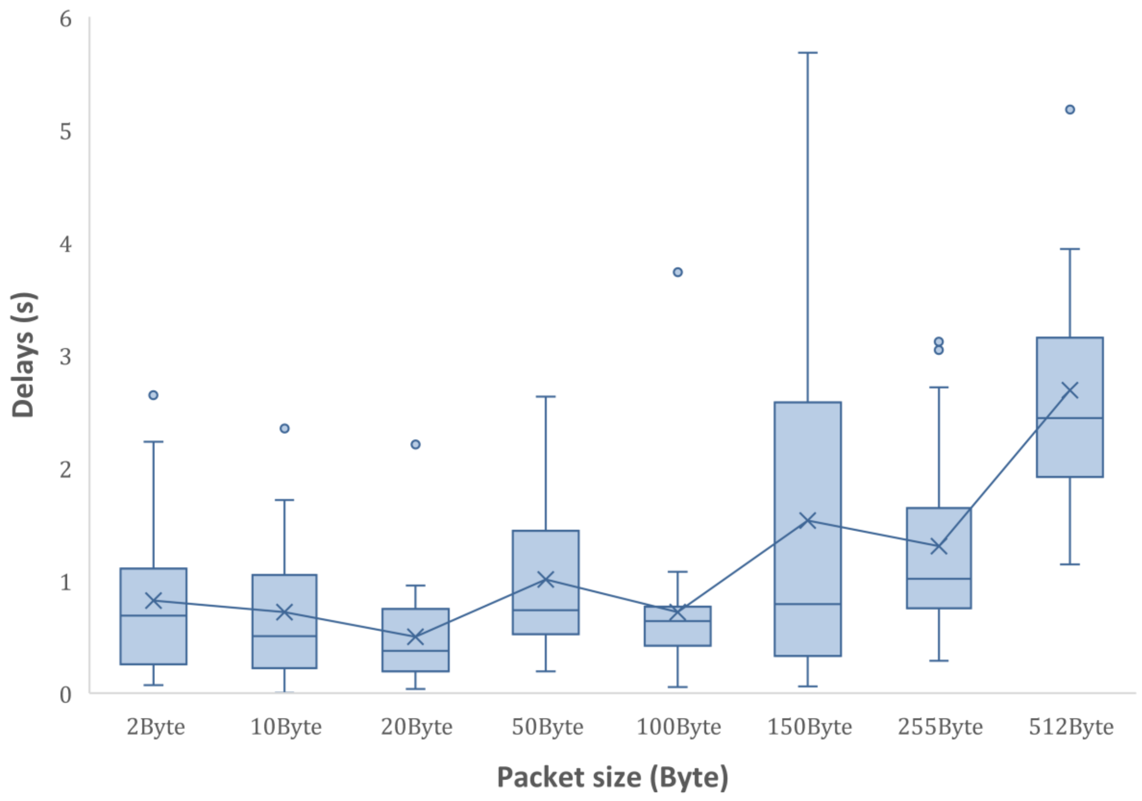


Figure 18 displays the box plot generated for experiment 2 in order to identify outliers and ensure the reliability of the results. The box plot illustrates the distribution of delay measurements for different packet sizes and shows the median, quartiles, and any outliers that were identified in the original data set. The line represents the average delay. The outliers have been removed from the dataset, resulting in a cleaned data set that is used for further analysis and presentation of results.

The results can be seen in the *Table 4*. The yellow cells were removed since they were considered outliers.

Payload size	2 (byte)	10 (byte)	20 (byte)	50 (byte)	100 (byte)	150 (byte)	255 (byte)	512 (byte)
Delay 01 (s)	0.12981	0.78191	0.76479	1.16297	0.25969	0.25969	1.06313	5.17844
Delay 02 (s)	1.35662	0.47121	2.20631	0.19275	0.6339	5.68149	2.71308	1.14124
Delay 03 (s)	0.24856	0.0744	0.40125	0.79889	0.43241	0.21858	0.40309	2.91491
Delay 04 (s)	1.12769	0.14776	0.10397	2.57921	0.05349	0.48714	0.52851	2.89942
Delay 05 (s)	0.07253	0.05964	0.03787	0.36995	0.2505	0.74212	0.7465	1.83681
Delay 06 (s)	1.34229	0.49099	0.15906	0.97731	0.41714	0.44721	0.28799	2.14692
Delay 07 (s)	0.27332	1.58739	0.27386	0.5529	0.18117	0.05989	0.76819	2.94459
Delay 08 (s)	2.23023	0.00175	0.15303	1.53444	0.50217	0.27687	3.11903	1.84097
Delay 09 (s)	0.20411	0.36889	0.21651	2.63164	0.64719	1.94938	1.01259	1.32982
Delay 10 (s)	0.7436	0.31214	0.18563	0.33709	0.64419	0.29162	0.83217	2.31599
Delay 11 (s)	0.61744	0.19072	0.44702	0.34491	0.80216	3.29296	0.65597	3.26385
Delay 12 (s)	0.15804	1.63985	0.36098	0.53514	0.73263	0.44884	2.3658	2.80867
Delay 13 (s)	0.84146	0.58549	0.39131	1.9867	0.64459	1.01589	0.77215	1.48285
Delay 14 (s)	2.64376	0.42782	0.42203	0.5196	0.77931	1.87166	0.86919	2.24272
Delay 15 (s)	0.80143	0.60355	0.25006	0.67214	0.66893	1.03577	1.68391	3.93886
Delay 16 (s)	0.574	0.52201	0.27922	0.66993	0.47283	2.78789	1.52128	2.26865
Delay 17 (s)	0.7806	1.71321	0.95323	0.96198	0.56545	2.93928	3.04256	3.22194
Delay 18 (s)	0.63179	2.34762	0.69243	1.70328	1.07618	0.78142	1.48134	2.14742
Delay 19 (s)	0.61409	0.97561	0.87355	0.96344	3.73413	5.24126	1.18494	2.56265
Delay 20 (s)	1.0369	1.07387	0.82766	0.66114	0.85825	0.7993	1.01768	5.23479

Table 4 Cleaned data. The yellow cells will be removed since they are considered as outliers.

From the cleaned data, the average and the standard deviation can be calculated. The results can be seen in the *Table 5*.

Payload size	2 (byte)	10 (byte)	20 (byte)	50 (byte)	100 (byte)	150 (byte)	255 (byte)	512 (byte)
Average Delay (s)	0.7255	0.63306	0.41018	1.00777	0.55906	1.53141	1.10597	2.40602
Standard Deviation (s)	0.53682	0.53605	0.27934	0.7209	0.25438	1.65797	0.64292	0.7345

Table 5 Average delay and standard deviation for different packet sizes. The table shows the results obtained from the cleaned dataset after removing the outliers. The average delay and standard deviation are calculated from 20 repetitions for each packet size.

To visualize the average delays and standard deviation, a scatter plot is presented in *Figure 19*. The blue line represents the average delay time, the dotted blue line shows the linear regression of the average delays, and the orange line represents the standard deviation. The standard deviation is an important statistical measure that indicates the amount of variation or dispersion of the data. A larger standard deviation indicates that the data points are spread out over a wider range, while a smaller standard deviation indicates that the data points are closer to the mean.

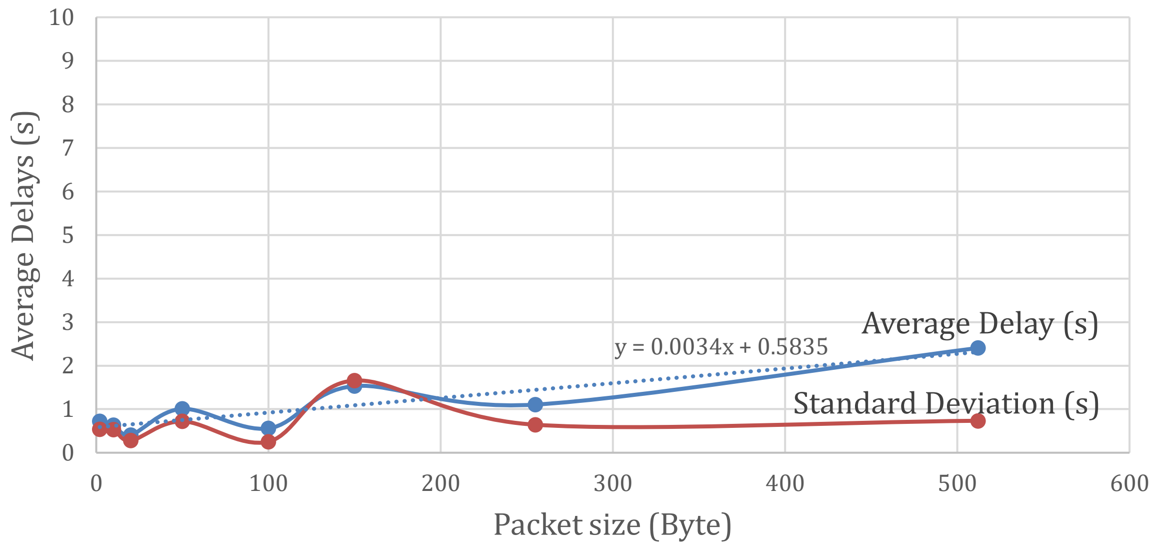


Figure 19 Scatter plot showing the average delays and standard deviation for the clean dataset. The blue line represents the average delay time, and the dotted blue line shows the linear regression of the average delays. The Brown line represents the standard deviation, which is a measure of the amount of variation or dispersion of the data. A larger standard deviation indicates that the data points are spread out over a wider range, while a smaller standard deviation indicates that the data points are closer to the mean.

5.2.2 Visualizing all the delays

A plot of all the delays recorded in the experiment is shown in *Figure 20*. Each point in the scatter plot represents an average of delay measurement for each packet size, while the x-axis indicates the packet sizes, and the y-axis represents the delay in second.

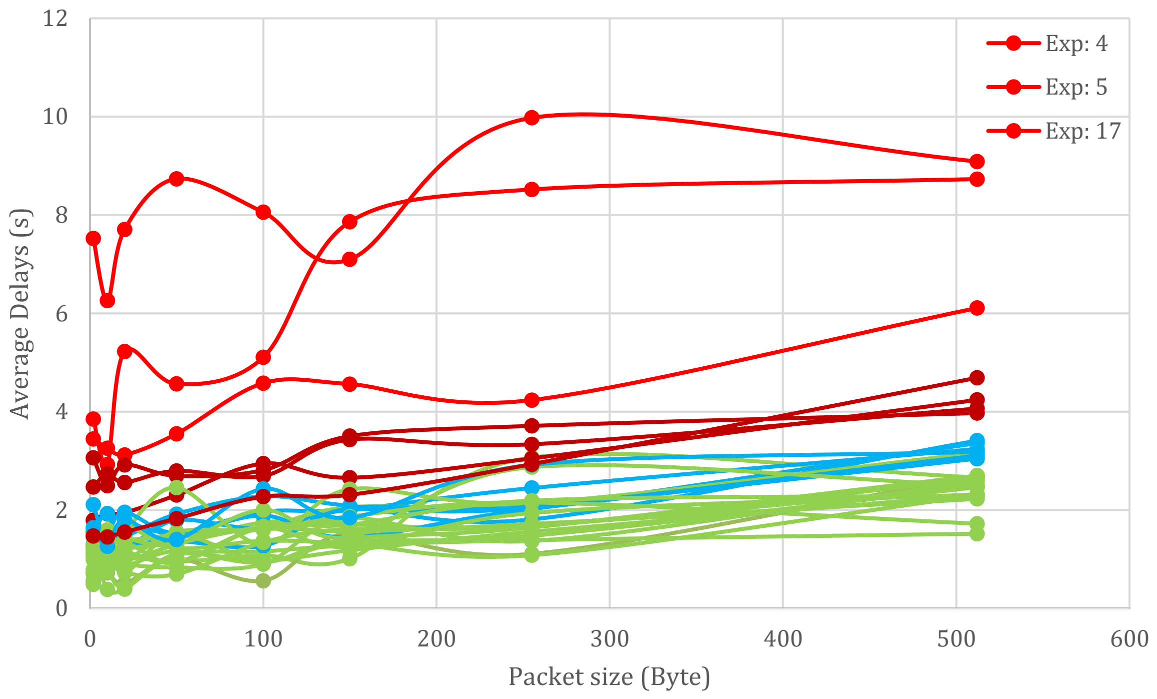


Figure 20 shows a scatter plot of delay measurements obtained from all 30 experiments conducted, each of which represents the average of 20 repetitions. Each point in the plot represents the average delay for a given packet size, with the x-axis representing the packet sizes in bytes and the y-axis indicating the delay in seconds. This plot allows for a visual inspection of the delay trends across different packet sizes and experiments.

Based on the plotted delays, it is observed that the average delays range from less than 1 second to about 10 seconds in the worst cases. Using the average delays as a criterion, the measurements can be grouped into four categories: the first group with the highest average delays ranging from 5 to 10 seconds, the second group with high delays, the third group with medium average delays, and the last group with the lowest average delays.

To enable better visualization of the delay groups, each group is depicted in a separate figure using the same scales for all figures. This allows for a direct comparison between the groups. *Figure 21* represents the group with the highest average delays, ranging from 5 to 10 seconds. *Figure 22* shows the group with high delays, while *Figure 23* presents the group with medium average delays. Finally, *Figure 24* shows the group with the lowest average delays which is the biggest group.

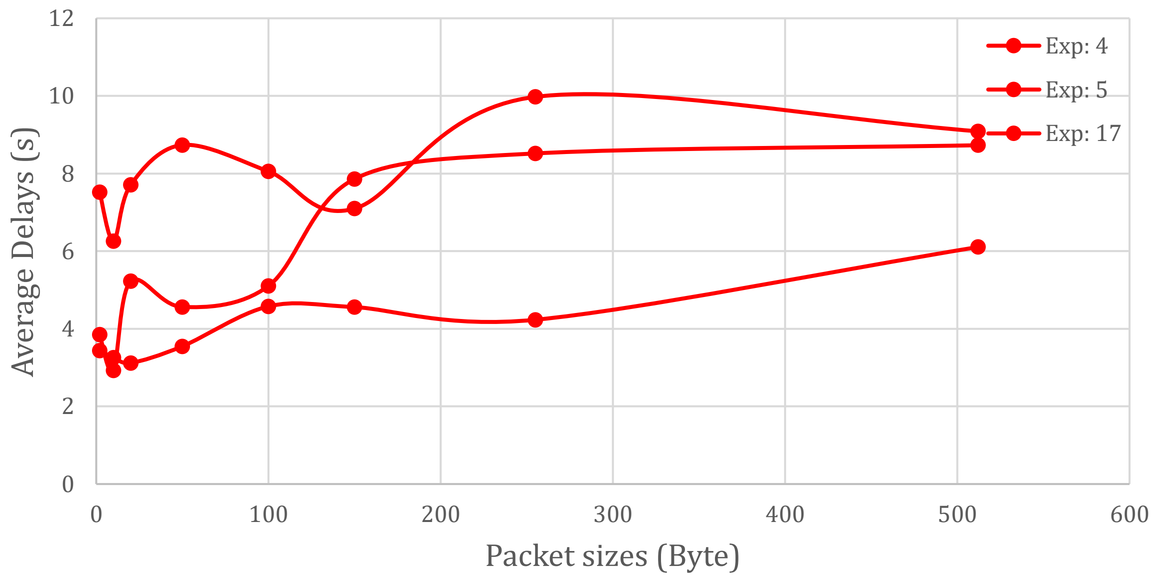


Figure 21 Experiments with highest average delays. The figure displays experiments with the highest average delays measured under extreme conditions. The signal quality was undetectable in two cases, and -87 dBm in another case, due to the measurements being conducted in basements and on floor -1 of the Polimi University's library. The x-axis represents the different packet sizes in bytes, and the y-axis represents the average delays in seconds.

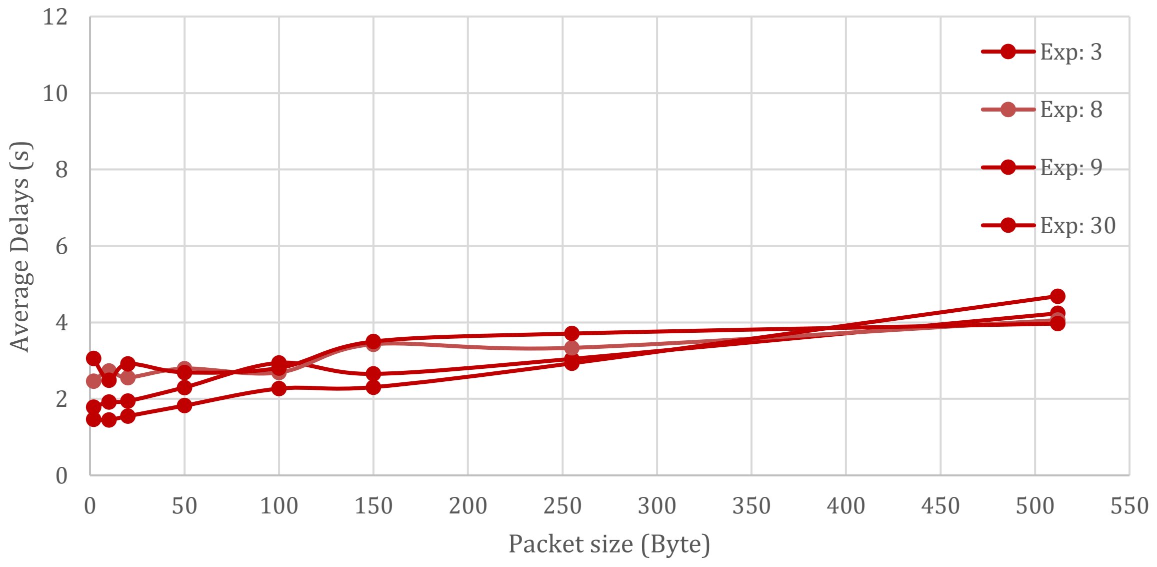


Figure 22 Experiments with Medium Average Delays. This figure displays the results of experiments with average delays ranging approximately from 3 to 5 seconds. The

experiments were conducted to investigate the performance of the system under various conditions, including different signal qualities and locations. The x-axis represents the different packet sizes in bytes, and the y-axis represents the average delays in seconds.

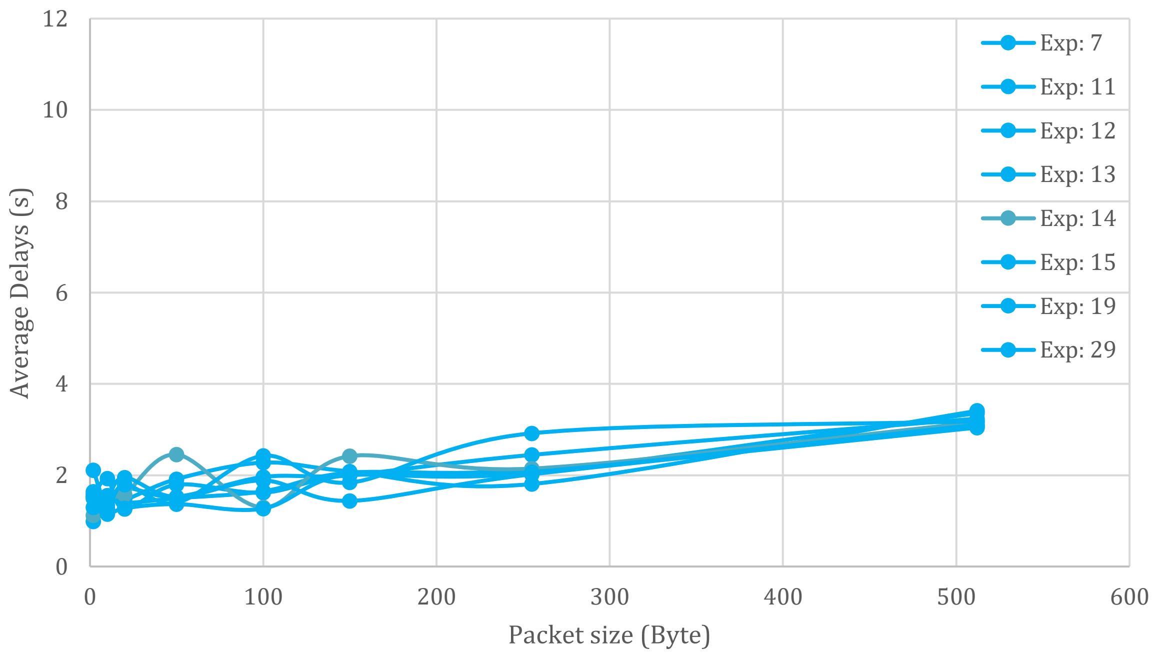


Figure 23 Experiments with Low Average Delays. The figure shows the average delay for experiments with delays ranging approximately from 1.5 to 3 seconds. The x-axis represents the different packet sizes in bytes, and the y-axis represents the average delays in seconds.

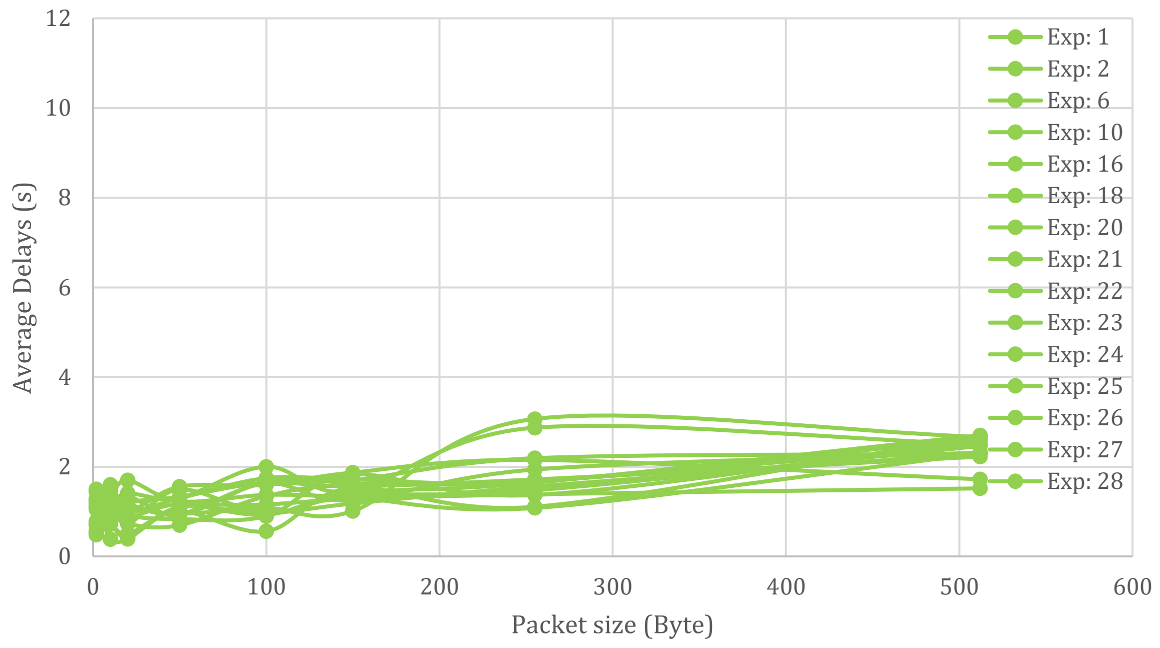


Figure 24 Experiments with Lowest Delays. This figure displays the experiments which have an average delay of approximately under 2 seconds. These experiments were conducted under good conditions with a good signal quality, including outdoor environments. The x-axis shows the packet sizes in bytes, and the y-axis represents the average delay in seconds.

5.3 Histogram analysis

A histogram is a graphical representation of the distribution of numerical data. It displays the frequency of occurrences of data values in a set of continuous or discrete intervals known as bins. A histogram can help to identify patterns, trends, and outliers in the data. It is particularly useful for understanding the shape of a dataset, including the location, and spread of the values. The histogram can also be used to determine the mode, median, and mean of the data. By examining the shape and characteristics of the histogram, one can draw inferences about the underlying population and make informed decisions about data analysis and modeling. Overall, histograms are a powerful tool for exploratory data analysis and are commonly used in fields such as statistics, data science, and machine learning.

5.3.1 CSQ (Channel Signal Quality)

The histogram of Cell Signal Quality (CSQ) shows the distribution of CSQ values across 30 measurements. The CSQ represents the quality of the received signal in decibels (dBm). *Table 1* presents the signal strength levels corresponding to each CSQ value. *Figure 25* displays the number of occurrences of CSQ values, with the x-axis representing the CSQ values and the y-axis representing the number of times each CSQ value was observed.

The histogram shows the values ranging from -87 dBm to -65 dBm, as well as two measurements where the CSQ value was unknown. The most frequently observed CSQ values are between -69 dBm and -73 dBm, with -69 dBm being the most common value within this range, observed nine times. This suggests that the network conditions during the measurements were generally good, although some measurements showed lower signal strength levels, with the lowest being -87 dBm, indicating poor signal quality. There were also two measurements where the CSQ value was unknown due to extreme conditions and low signal quality.

CSQ (dBm)	*Count*
-87	1
-85	0
-83	1
-81	1
-79	3
-77	2
-75	2
-73	4
-71	4
-69	9
-67	0
-65	1
Unknown	2

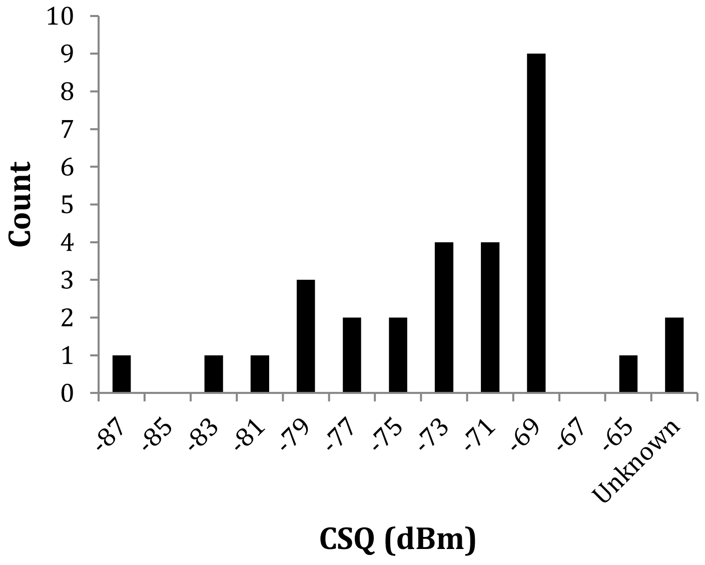


Figure 25 Histogram showing CSQ distribution, with two cases excluded due to low signal quality.

5.3.2 Signal power

Histogram of signal power shows the distribution of signal power values across the 30 measurements *Figure 26*. The x-axis represents the signal power values in centibels (dBc), and the y-axis represents the number of times each signal power value was observed in these ranges. The histogram shows that the most frequently observed signal power values are between -800 dBc and -700 dBc being the most common value. However, there were also some measurements where the signal power reported as -32768 which is an indication of a no signal or very weak signal.

Signal Power (dBc)	*Count*
-1100 to -1000	0
-1000 to -900	1
-900 to -800	1
-800 to -700	13
-700 to -600	13
undetectable	2

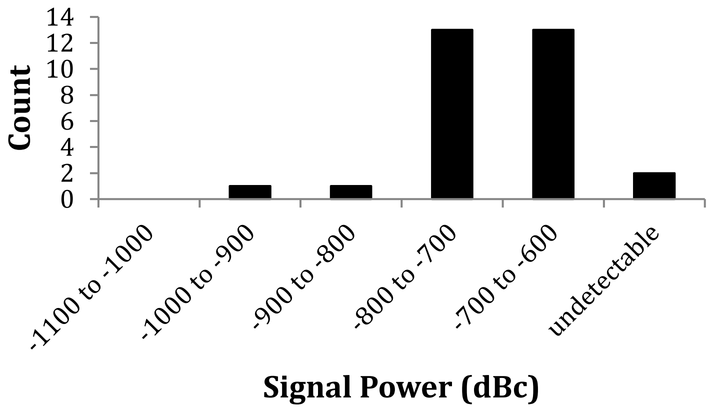


Figure 26 Histogram of signal power. The figure shows a histogram of signal power values in centibels (dBc), with the x-axis representing the signal power values and the y-axis representing the number of occurrences. Some measurements reported a signal power value of -32768, indicating no or very weak signal. These measurements are represented by a separate category in the histogram.

5.3.3 SNR

Figure 27 illustrate the histogram of signal-to-noise ratio (SNR). This histogram shows the distribution of SNR values across the 30 measurements. The x-axis represents the SNR values, and the y-axis represents the number of times each SNR value was observed in these ranges. The histogram shows that the most frequently observed SNR values are between 150 and 200. However, there were also some measurements where the SNR was reported as 0, which means that the signal and noise power are equal, and some measurements reported negative values, which means that the noise power is higher than the signal power. Additionally, there were two measurements where the SNR was reported as -32768, which is an indication of a no signal or very weak signal.

SNR	*Count*
-50 to 0	2
0 to 50	5
50 to 100	7
100 to 150	6
150 to 200	8
undetectable	2

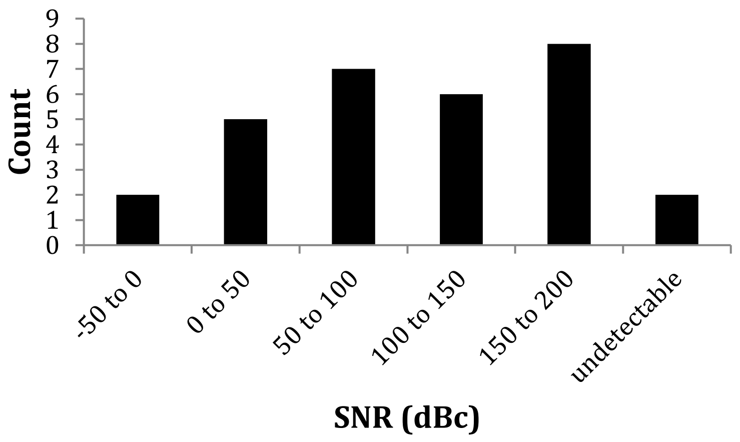


Figure 27 Histogram of SNRs. SNRs related to each experiment are divided into 5 ranges. There were two measurements where the SNR was reported as -32768, which is an indication of a no signal or very weak signal.

5.3.4 RSRQ

This histogram shows the distribution of RSRQ (Reference Signal Received Quality) values across the measurements. RSRQ is represented in dBc and measures the quality of the received reference signal. The most frequently observed RSRQ values are between -110 dBc and -100 dBc which is shown in *Figure 28*. There were also some measurements where the RSRQ value was -32768, which is an indication of a no signal or very weak signal. Overall, the distribution of RSRQ values appears to be bimodal, with one peak around -100 dBc and another peak around -110 dBc.

RSRQ (dBc)	*Count*
-170 to -160	1
-160 to -150	1
-150 to -140	0
-140 to -130	1
-130 to -120	0
-120 to -110	10
-110 to - 100	15
Undetectable	2

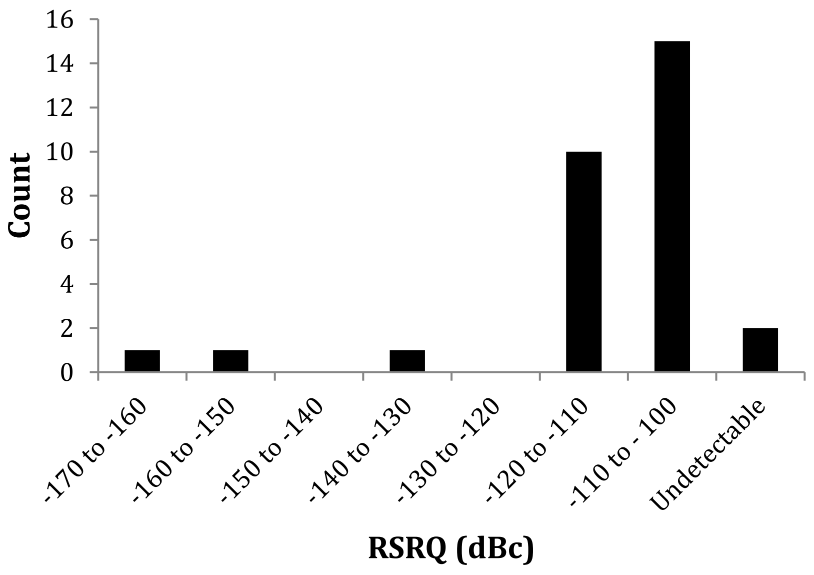


Figure 28 Histogram of RSRQ. This histogram displays the distribution of RSRQ values across all the measurements. The most observed RSRQ values range from -110 dBc to -100 dBc, indicating a fair signal quality. However, some measurements had RSRQ values of -32768, which implies no signal or very weak signal. The distribution of RSRQ values is bimodal, with one peak centered around -100 dBc and another peak centered around -110 dBc, suggesting that there are two types of signal qualities present in the dataset.

5.4 Correlation analysis

Correlation analysis is a statistical technique used to measure the strength and direction of the relationship between two variables. The correlation coefficient ranges from -1 to 1, where a value of -1 indicates a perfect negative correlation, a value of 0 indicates no correlation, and a value of 1 indicates a perfect positive correlation. In this study, Pearson correlation coefficient is used to measure the linear relationship between two variables. The formula for the Pearson correlation coefficient is given by *Equation 1*. [26]

$$\mathbf{r} = \frac{\mathbf{n}(\sum \mathbf{xy}) - (\sum \mathbf{x})(\sum \mathbf{y})}{\sqrt{[\mathbf{n}\sum \mathbf{x}^2 - (\sum \mathbf{x})^2][\mathbf{n}\sum \mathbf{y}^2 - (\sum \mathbf{y})^2]}}$$

Equation 1 Pearson correlation coefficient

Where,

- r = Pearson Coefficient
- n= number of pairs of the stock
- ∑xy = sum of products of the paired stocks
- ∑x = sum of the x scores
- ∑y= sum of the y scores
- ∑x2 = sum of the squared x scores
- ∑y2 = sum of the squared y scores

5.4.1 Correlation of Average Delays and CSQ

The correlations between delays of different packet sizes and channel signal quality (CSQ) were investigated, and it was found that the correlations were negative for all packet sizes. Specifically, the correlation coefficients ranged from -0.257 to -0.502, indicating a moderate to strong negative relationship between delays and CSQ. These results suggest that as the CSQ improves, the delays of packets of different sizes decrease.

The negative correlation between delays and CSQ is an expected result since better signal quality leads to faster transmission of data packets. The correlation coefficients provide a quantitative measure of the strength and direction of the relationship between the two variables. The moderate to strong correlations observed in this study suggest that the effect of CSQ on delays is substantial, and it is important to consider the CSQ when evaluating the performance of the network.

The packet size also appears to play a role in the relationship between delays and CSQ, as the correlation coefficients vary across different packet sizes. For instance, the correlation coefficient is stronger for larger packet sizes such as 255-byte and 512-byte packets, compared to smaller packet sizes such as 2-byte and 10-byte packets. This observation suggests that the impact of CSQ on delays may be more pronounced for larger packets. *Table 6* summarizing the correlations between delays of different packet sizes and channel signal quality CSQ.

Packet sizes	Correlation of Average Delays with CSQ
2-byte	-0.344
10-byte	-0.257
20-byte	-0.494
50-byte	-0.369
100-byte	-0.284
150-byte	-0.416
255-byte	-0.502
512-byte	-0.446

Table 6 Correlation coefficients between Average Delay of different packet sizes and Channel Signal Quality (CSQ), indicating stronger correlations for larger packet sizes such as 255-byte and 512-byte packets compared to smaller packet sizes such as 2-byte and 10-byte packets.

To enhance the visual representation, the correlation between CSQ and average delays for different packet sizes are depicted. Specifically, *Figure 29* through *Figure 36* correspond to correlations between CSQ and average delays of different packet sizes ranging from 2 to 512 bytes respectively.

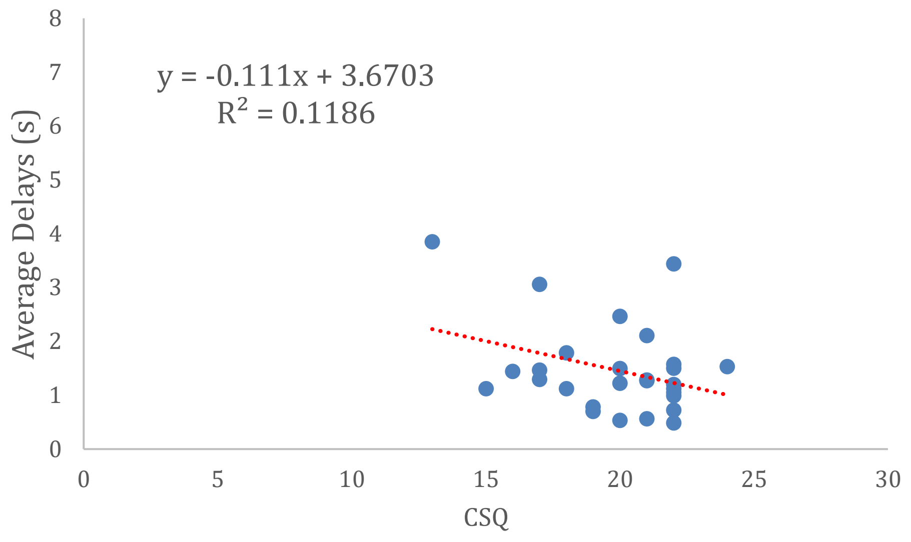


Figure 29 Regression line and correlation between CSQs and average delays of 2-byte packets. R^2 is equal to the square of the correlations in *Table 6.* CSQ is a Received Signal Strength indicator, and its corresponding Signal Strength is reported in the *Table 1*. For example, CSQ = 20 indicates Received Signal of -73 dBm and CSQ = 21 is -75 dBm.

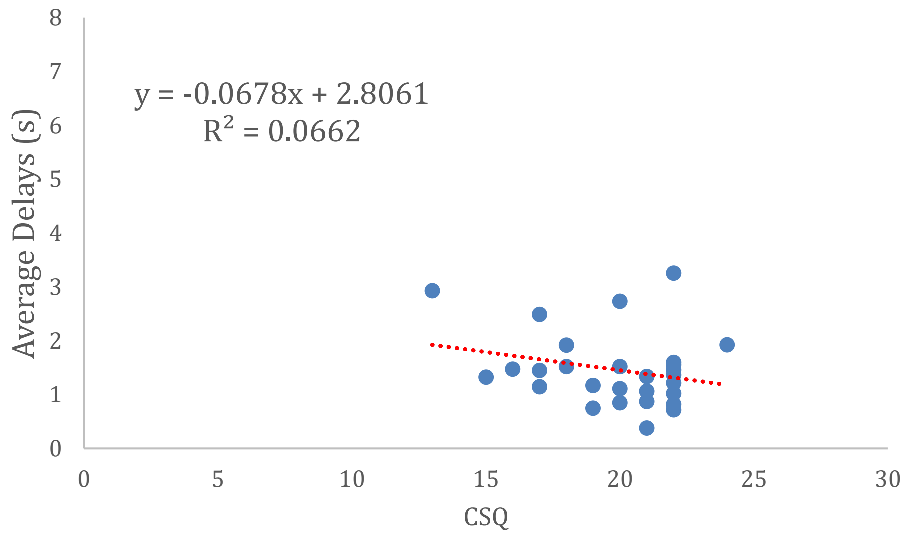


Figure 30 Regression line and correlation between CSQs and average delays of 10-byte packets. R^2 is equal to the square of the correlations in *Table 6.* CSQ is a Received Signal Strength indicator, and its corresponding Signal Strength is reported in the *Table 1*. For example, CSQ = 20 indicates Received Signal of -73 dBm and CSQ = 21 is -75 dBm.

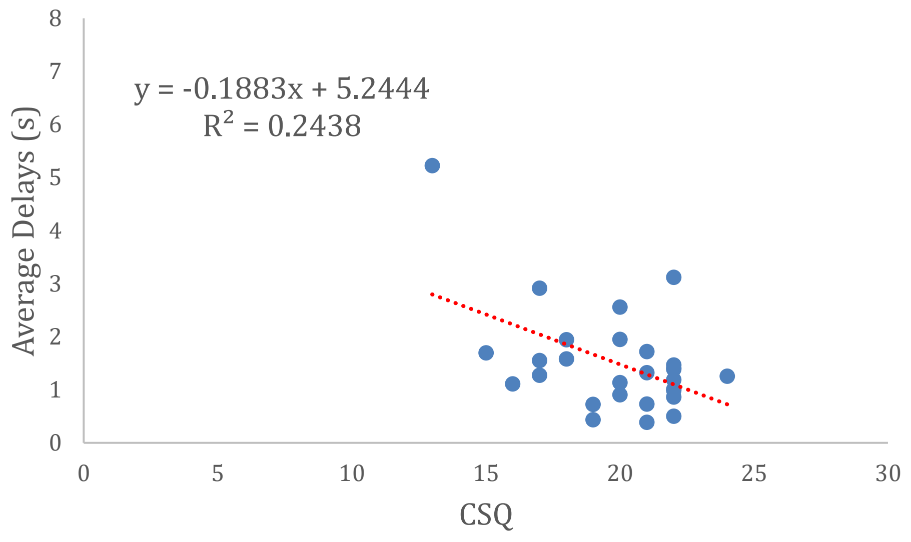


Figure 31 Regression line and correlation between CSQs and average delays of 20-byte packets. R² is equal to the square of the correlations in *Table 6.* CSQ is a Received Signal Strength indicator, and its corresponding Signal Strength is reported in the *Table 1*. For example, CSQ = 20 indicates Received Signal of -73 dBm and CSQ = 21 is -75 dBm.

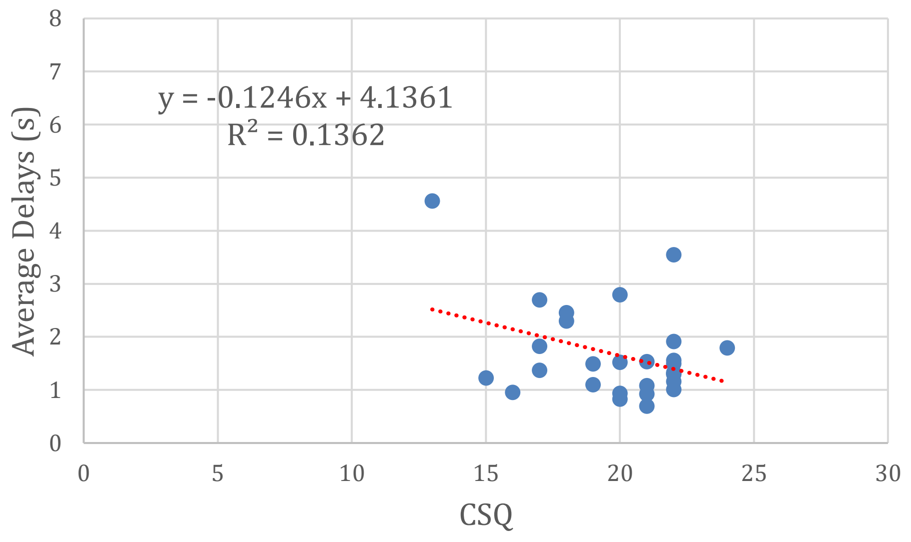


Figure 32 Regression line and correlation between CSQs and average delays of 50-byte packets. R² is equal to the square of the correlations in *Table 6.* CSQ is a Received Signal Strength indicator, and its corresponding Signal Strength is reported in the *Table 1*. For example, CSQ = 20 indicates Received Signal of -73 dBm and CSQ = 21 is -75 dBm.

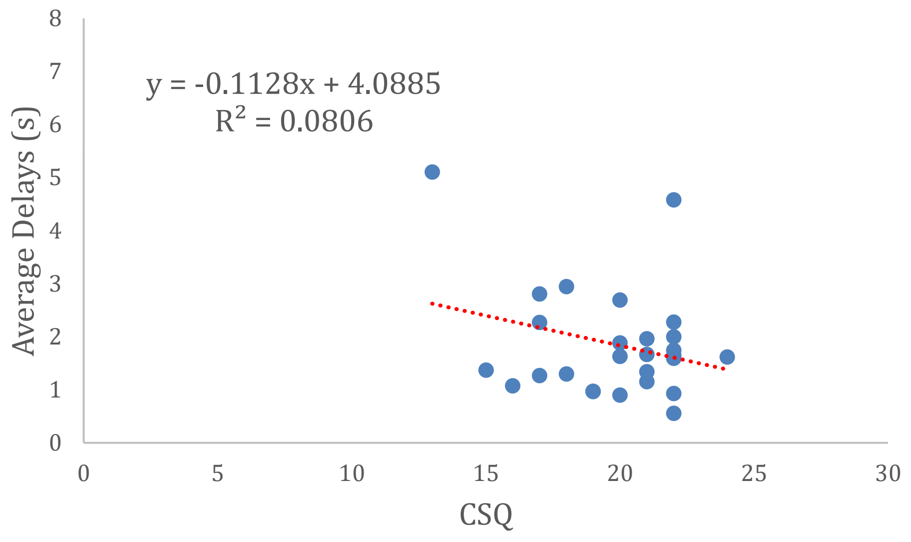


Figure 33 Regression line and correlation between CSQs and average delays of 100-byte packets. R^2 is equal to the square of the correlations in ***Table 6***. CSQ is a Received Signal Strength indicator, and its corresponding Signal Strength is reported in the ***Table 1***.

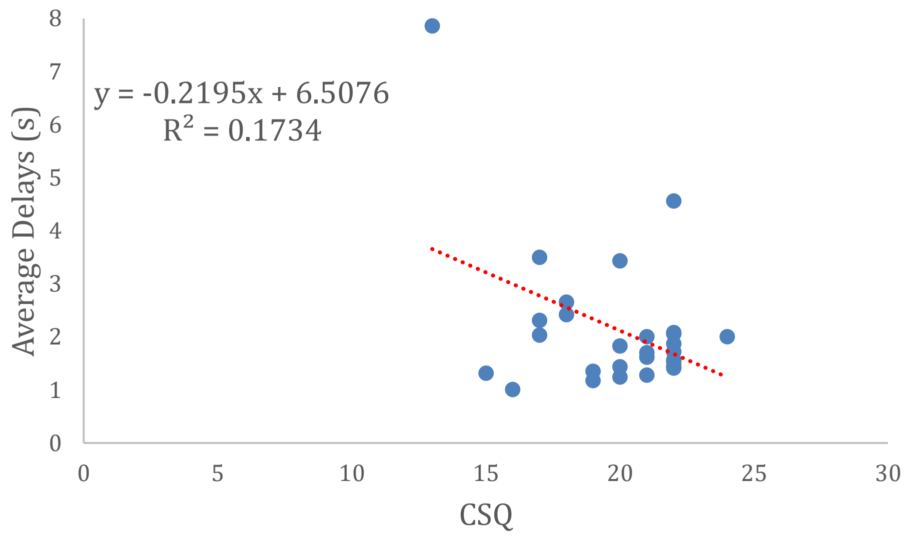


Figure 34 Regression line and correlation between CSQs and average delays of 150-byte packets. R^2 is equal to the square of the correlations in ***Table 6***. CSQ is a Received Signal Strength indicator, and its corresponding Signal Strength is reported in the ***Table 1***.

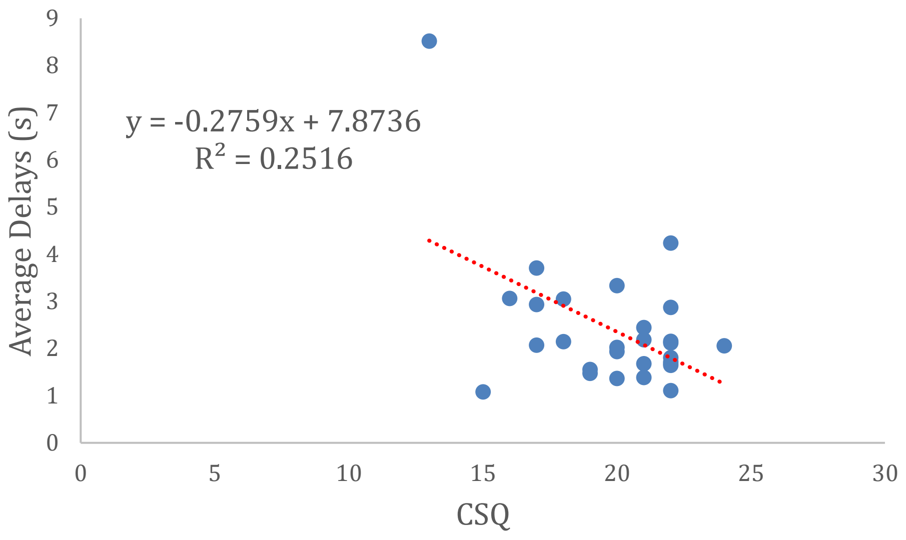


Figure 35 Regression line and correlation between CSQs and average delays of 255-byte packets. R^2 is equal to the square of the correlations in *Table 6*. CSQ is a Received Signal Strength indicator, and its corresponding Signal Strength is reported in the *Table 1*.

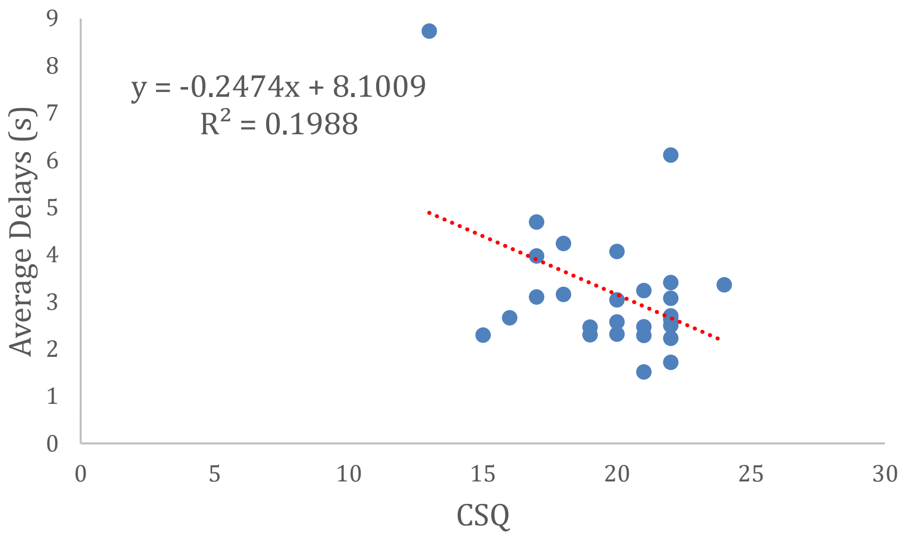


Figure 36 Regression line and correlation between CSQs and average delays of 512-byte packets. R^2 is equal to the square of the correlations in *Table 6*. CSQ is a Received Signal Strength indicator, and its corresponding Signal Strength is reported in the *Table 1*.

5.4.2 Correlation of Delays and Signal power

The results of the correlation analysis between Signal Power and delay for different packet sizes revealed a negative correlation between the two variables. This indicates that as the Signal Power increases, the delay for packet transmission decreases. The findings are consistent with previous studies that have shown a strong association between Signal Power and delay in wireless networks.

The analysis of the correlation coefficients showed that the correlation was strongest for the 255-Byte packet size, with a coefficient of -0.689. This suggests that Signal Power has a greater impact on the delay for larger packet sizes.

However, the analysis also showed weaker correlations for smaller packet sizes, such as the 2-Byte and 10-Byte packet sizes. *Table 7* represents the correlation of Delays of packets with different sizes and Signal Power.

Packet size	Correlation of average delay with Signal power
2-byte	-0.344
10-byte	-0.257
20-byte	-0.494
50-byte	-0.369
100-byte	-0.284
150-byte	-0.416
255-byte	-0.502
512-byte	-0.446

Table 7 Correlation between signal power and average delays for each packet size. The analysis found a strong negative correlation between signal power and delay for larger 255-Byte and 512-Byte packets, indicating that signal power has a greater impact on delay for larger packets. However, weaker correlations were found for smaller packet sizes, such as 2-Byte and 10-Byte packets.

5.4.3 Correlation of Delays and SNR

The analysis of the correlation between Delays for different packet sizes and SNR (Signal-to-Noise Ratio) revealed a negative correlation between these variables. This means that as SNR increases, the delay in packet transmission decreases. A higher SNR indicates a better signal quality and lower noise interference, which results in a faster and more reliable data transmission.

The correlation coefficients for different packet sizes showed that the strongest negative correlation was for the 512-Byte packet size, with a coefficient of -0.646. This indicates that for larger packet sizes, the impact of SNR on the delay is more significant. The correlation coefficients for smaller packet sizes, such as the 2-Byte and 10-Byte packet sizes, were weaker than those for larger packet sizes. *Table 8* shows the correlation coefficients between delays for different packet sizes and SNR.

Delay of each packet	Correlation with SNR
2-byte	-0.476770216
10-byte	-0.441815508
20-byte	-0.489958069
50-byte	-0.52528022
100-byte	-0.592782996
150-byte	-0.609079309
255-byte	-0.601858027
512-byte	-0.6466467

Table 8 Correlation between Delays of different Packet Sizes and SNR: Analysis shows negative correlation between SNR and average delays, where increasing SNR leads to a decrease in delay. The strongest negative correlation was observed for 512-Byte packet size, indicating the greater impact of SNR on larger packet sizes. The correlation coefficients for smaller packet sizes, such as the 2-Byte and 10-Byte packet sizes, were weaker than those for larger packet sizes.

5.4.4 Correlation of Delays and RSRQ

The analysis of the correlation between delays of different packet sizes and RSRQ (Reference Signal Received Quality) indicated a negative correlation between these variables. This means that as RSRQ increases, the delay in packet transmission decreases. A higher RSRQ indicates a better signal quality, which results in faster and more reliable data transmission.

The correlation coefficients for different packet sizes showed that the strongest negative correlation was for the 512-Byte packet size, with a coefficient of -0.792. This suggests that for larger packet sizes, the impact of RSRQ on delay is more significant. The correlation coefficients for smaller packet sizes, such as the 2-Byte and 10-Byte packet sizes, were weaker than those for larger packet sizes. *Table 9* represents the correlations of Delays and RSRQ.

Packet size	Correlation of average delays with RSRQ
2-byte	-0.565402992
10-byte	-0.474342591
20-byte	-0.629408969
50-byte	-0.63647146
100-byte	-0.688076458
150-byte	-0.725635475
255-byte	-0.74274506
512-byte	-0.791652793

Table 9 Correlation coefficients between RSRQ and average Delays for different packet sizes. The table displays the strongest negative correlation was for the 512-Byte packet size, with a coefficient of -0.792. This suggests that for larger packet sizes, the impact of RSRQ on delay is more significant. The correlation coefficients for smaller packet sizes, such as the 2-Byte and 10-Byte packet sizes, were weaker than those for larger packet sizes but still have a significant impact.

5.5 Histogram analysis of delays

To evaluate the performance of the NB-IoT, the delays for different packet sizes were analyzed under various experimental conditions. Delays were computed for packet sizes of 2, 10, 20, 50, 100, 150, 255, and 512 bytes, and 30 measurements with 20 repetitions for each packet size were conducted. The experimental conditions included parameters such as CSQ, signal power, total power, TX power, TX time, RX time, cell ID, ECL, SNR, EARFCN, PCI, and RSRQ.

To visualize the distribution of delays for each packet size, histograms were constructed. The same bin ranges were used for all histograms to facilitate comparison across different packet sizes. The bin ranges were set to 0-1 sec, 1-2 sec, 2-3 sec, 3-4 sec, 4-5 sec, 5-6 sec, 6-7 sec, 7-8 sec, 8-9 sec, and 9-10 sec.

The histograms showed that the delays for different packet sizes followed distinct distributions. For example, the delays for the 2-byte, 10-byte and 20-byte packets tended to be concentrated in the range of 0-4 sec, while the delays for the 255-byte and 512 packets were distributed from 2 to 9 seconds. Additionally, we observed that the delays tended to increase as the packet size increased, particularly for larger packet sizes such as 150, 255, and 512 bytes.

Overall, the histogram analysis of packet delay provided valuable insights into the performance of the packet network under different experimental conditions. By visualizing the distribution of delays for each packet size, we were able to identify patterns and trends that could help inform future improvements to the network.

The histograms and tables presented in this section provide a comprehensive analysis of the delays observed in the NB-IoT system under various experimental conditions. The histograms visually depict the distribution of delays for each packet size. These visualizations enable a better understanding of the performance of the system under different experimental conditions that can be seen in the *Figure 37* for 2-byte packet size, *Figure 38* for 10-byte packet size, *Figure 39* for 20-byte packet size, *Figure 40* for 50-byte packet size, *Figure 41* for 100-byte packet size, *Figure 42* for 150-byte packet size, *Figure 43* for 255-byte packet size, and finally *Figure 44* for 512-byte packet size.

2-byte packet size

Average Delays (s)	*Count*
0 to 1	7
1 to 2	16
2 to 3	2
3 to 4	3
4 to 5	0
5 to 6	0
6 to 7	0
7 to 8	0
8 to 9	0
More	0

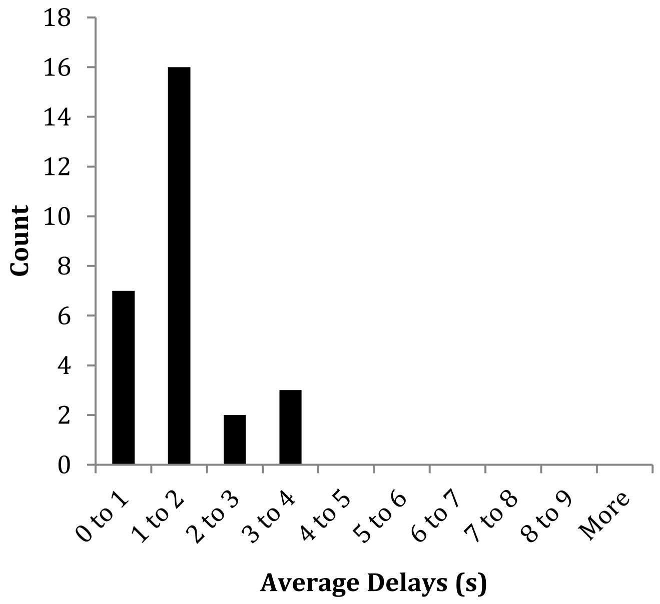


Figure 37 Histogram of Average Delays of 2-byte packet sizes. The range of delay is zero to 4 seconds with a pick in the center of 2 seconds delays.

10-byte packet size

Average Delays (s)	*Count*
0 to 1	6
1 to 2	18
2 to 3	3
3 to 4	1
4 to 5	0
5 to 6	0
6 to 7	0
7 to 8	0
8 to 9	0
More	0

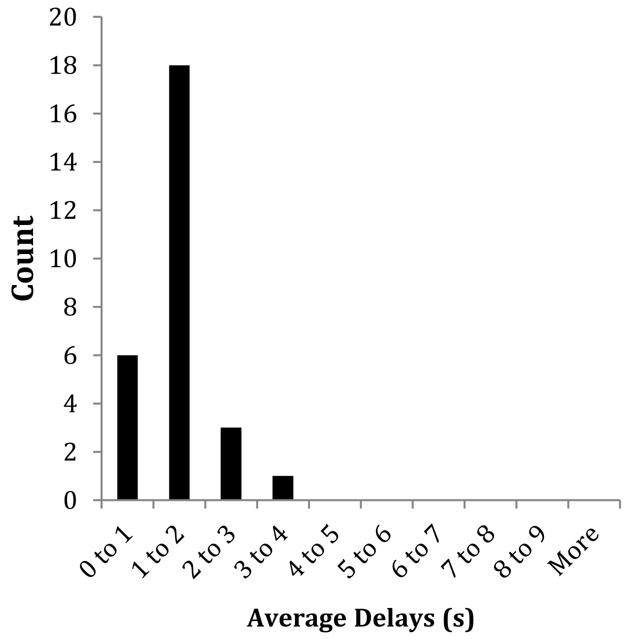


Figure 38 Histogram of Average Delays of 10-byte packet sizes. The range of delay is zero to 4 seconds with a pick in the center of 2 seconds delays.

20-byte packet size	
Average Delays(s)	*Count*
0 to 1	8
1 to 2	16
2 to 3	2
3 to 4	1
4 to 5	0
5 to 6	1
6 to 7	0
7 to 8	0
8 to 9	0
More	0

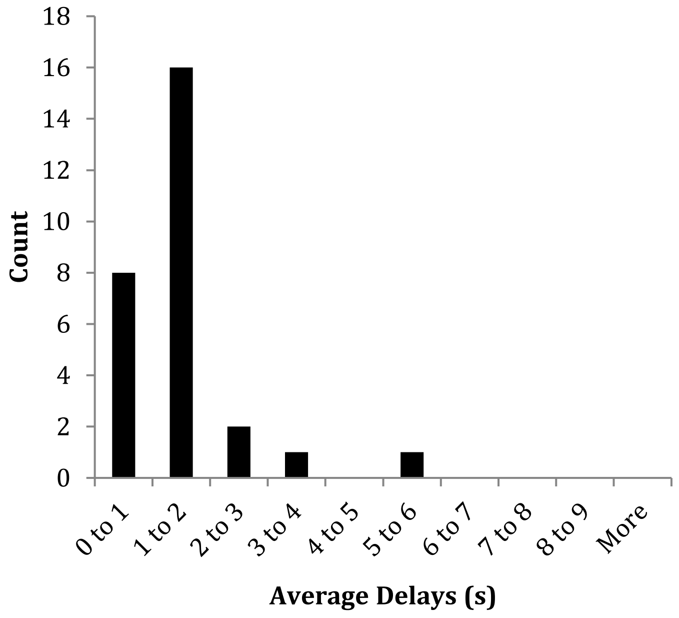


Figure 39 Histogram of Average Delays of 20-byte packet sizes.

50-byte packet size	
Average Delays (s)	*count*
0 to 1	5
1 to 2	17
2 to 3	4
3 to 4	1
4 to 5	1
5 to 6	0
6 to 7	0
7 to 8	0
8 to 9	0
More	0

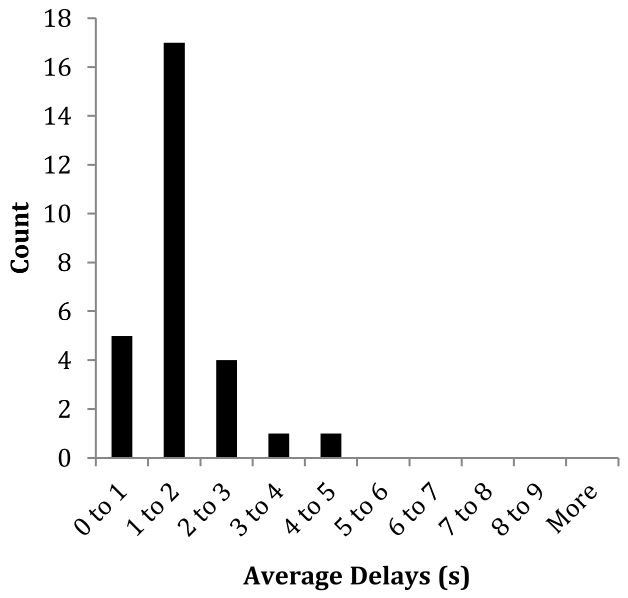


Figure 40 Histogram of Average Delays of 50-byte packet sizes. The range of delays increases to 5 seconds.

100-byte packet size

Average Delays (s)	*Count*
0 to 1	5
1 to 2	15
2 to 3	6
3 to 4	0
4 to 5	1
5 to 6	1
6 to 7	0
7 to 8	0
8 to 9	0
More	0

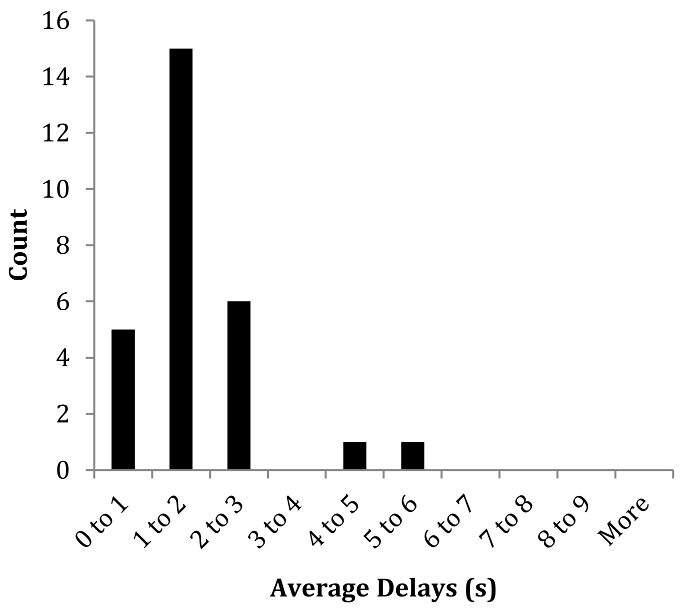


Figure 41 Histogram of Average Delays of 100-byte packet sizes. The range of delays increases to 6 seconds.

150-byte packet size

Average Delays (s)	*Count*
0 to 1	0
1 to 2	16
2 to 3	8
3 to 4	2
4 to 5	1
5 to 6	0
6 to 7	0
7 to 8	1
8 to 9	0
More	0

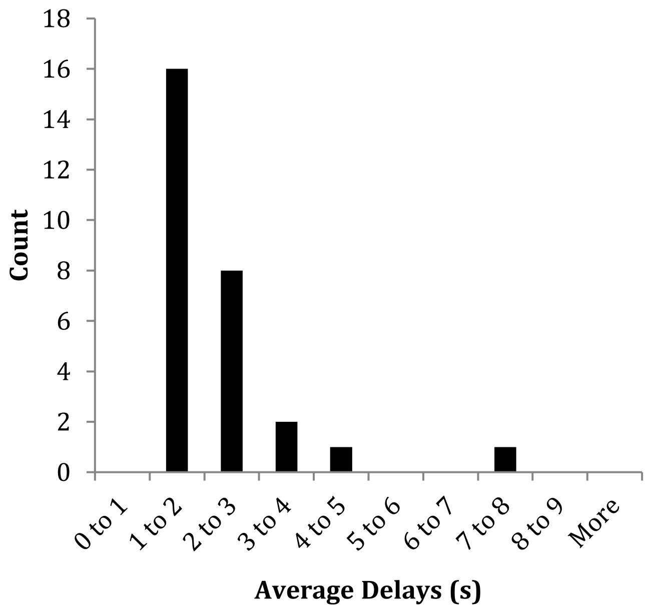


Figure 42. Histogram of Average Delays of 150-byte packet sizes.

255-byte packet size

Average Delays(s)	*Count*
0 to 1	0
1 to 2	12
2 to 3	10
3 to 4	4
4 to 5	1
5 to 6	0
6 to 7	0
7 to 8	0
8 to 9	1
More	0

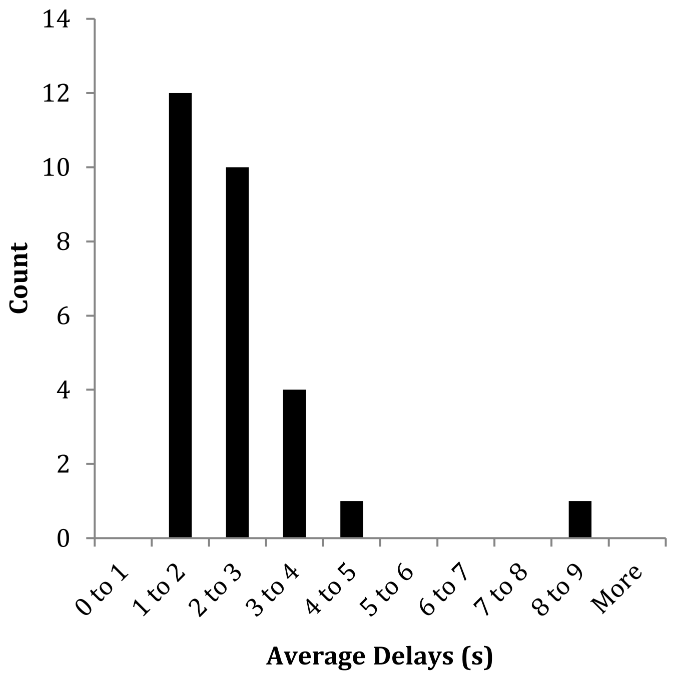


Figure 43. Histogram of Average Delays of 255-byte packet sizes.

512-byte packet size

Average Delays(s)	*Count*
0 to 1	0
1 to 2	2
2 to 3	13
3 to 4	8
4 to 5	3
5 to 6	0
6 to 7	1
7 to 8	0
8 to 9	1
More	0

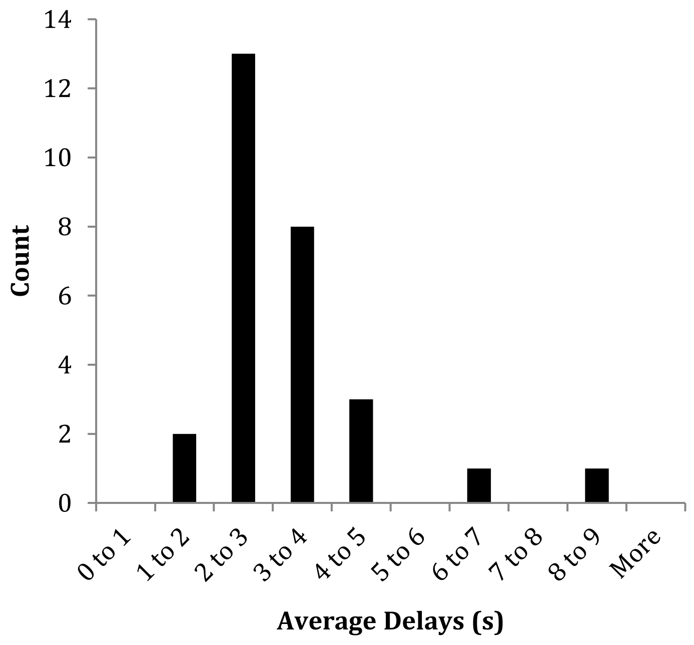


Figure 44 Histogram of Average Delays of 512-byte packet sizes.

6. Conclusion and future development

In this work, we have delved into the world of NB-IoT to understand how this technology performs under real and extreme conditions. We began by providing a demonstration of IoT technologies and specifically, what NB-IoT entails. This allowed the reader to gain a better understanding of the world of NB-IoT in the context of the Internet of Things.

Next, we explained the implementation of the hardware and software required for this technology, as well as the necessary commands and codes. We then introduced and described the measurement tools used in our assessment.

Our main objective was to evaluate the delay performance of NB-IoT under various system parameters measured at different locations. Our findings revealed that there is considerable variation in delay, even due to packet size, which may be attributed to non-linear behavior of the equipment. Additionally, we discovered a clear correlation between signal quality and delay, with extreme locations experiencing delays in the range of 5 to 10 seconds, while other locations experience delays ranging from 1 to 4 seconds.

While this work provides a detailed assessment of NB-IoT technology, future work could include a comparison of NB-IoT with other IoT technologies, such as Cat-M.

Bibliography

[1] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660. https://doi.org/10.1016/j.future.2013.01.010

[2] Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer Networks, 54(15), 2787-2805. https://doi.org/10.1016/j.comnet.2010.05.010

[3] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17(4), 2347-2376. https://doi.org/10.1109/COMST.2015.2444095

[4] Koubaa, A., Madani, A., & Ben Jemaa, M. (2018). Comparative analysis of IoT wireless technologies for industrial applications. In 2018 International Conference on Wireless Networks and Mobile Communications (WINCOM) (pp. 1-6). IEEE. https://doi.org/10.1109/WINCOM.2018.8686745

[5] Guan, X., & Zheng, K. (2018). Performance analysis of narrowband IoT. IEEE Internet of Things Journal, 6(1), 158-169. https://doi.org/10.1109/JIOT.2018.2811241

[6] Ratasuk, R., Mangalvedhe, N., & Ghosh, A. (2015). Overview of LTE enhancements for cellular IoT. In Proceedings of the IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC) (pp. 1729-1734). IEEE. https://doi.org/10.1109/PIMRC.2015.7343665

[7] 3GPP TR 23.887. (2013). Study on Machine-Type Communications (MTC) and other mobile data applications communications enhancements, v.12.0.0. December 2013. https://www.3gpp.org/ftp//Specs/archive/23_series/23.887/

[8] 3GPP. (2016). TS 36.300 - LTE; Overall description; Stage 2. V13.4.0. https://www.3gpp.org/ftp//Specs/archive/36_series/36.300/36300-f40.zip

[9] 3GPP. (2017). Technical Specification Group Radio Access Network; Study on enhancement of 3GPP support for 5G V2X services (Release 14). 3GPP TS 36.888. Retrieved from https://www.3gpp.org/ftp//Specs/archive/36_series/36.888/

[10] 3GPP. (2018). The 3GPP specifications: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (Release 15). [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/36_series/36.211/36211-f50.zip

[11] Meador, D. (2018). Semaphores in Operating System. Retrieved from https://www.tutorialspoint.com/semaphores-in-operating-system

[12] 3GPP. (n.d.). Narrowband IoT. Retrieved April 10, 2023, from https://www.3gpp.org/technologies/keywords-acronyms/98-nb-iot

[13] Kousias, K., Caso, G., Alay, Ö., Brunstrom, A., & D'Antonio, S. (2020). Coverage and Deployment Analysis of Narrowband Internet of Things in the Wild. IEEE Access, 8, 123830-123843. https://doi.org/10.1109/ACCESS.2020.3007533

[14] Mangalvedhe, N., Ratasuk, R., & Ghosh, A. (2016, September). NB-IoT deployment study for low power wide area cellular IoT. In 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC) Workshop (pp. 1-6). IEEE.

[15] Quectel Wireless Solutions. (n.d.). About Us. Retrieved April 10, 2023, from https://www.quectel.com/about/about-us.htm

[16] Quectel Wireless Solutions. (n.d.). NB-IoT BC95 module. https://www.quectel.com/product/bc95.htm

[17] Quectel. (2018). BC95-G&BC68 Hardware Design. Retrieved from https://www.quectel.com/UploadImage/Downlad/Quectel_BC95-G&BC68_Hardware_Design_V1.1.pdf

[18] Quectel. (2020). Quectel BC95-G&BC68 AT Commands Manual. Retrieved from https://www.quectel.com/UploadImage/Downlad/Quectel_BC95-G&BC68_AT_Commands_Manual_V1.3.pdf

[19] Python Software Foundation. (n.d.). serial — Serial port class. Python documentation. Retrieved April 6, 2023, from https://docs.python.org/3/library/serial.html

[20] Python Software Foundation. (n.d.). sched — Event scheduler. Python documentation. Retrieved April 6, 2023, from https://docs.python.org/3/library/sched.html

[21] Python Software Foundation. (n.d.). time — Time access and conversions. Python documentation. Retrieved April 6, 2023, from https://docs.python.org/3/library/time.html

[22] Electronic Industries Alliance. (1969). RS-232 - Recommended Standard 232: Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange.

[23] Ylonen, T., & Stenberg, P. (2006). The SSH protocol architecture. Internet Engineering Task Force. https://tools.ietf.org/html/rfc4251

[24] BitVise. (n.d.). SSH Client. https://www.bitvise.com/ssh-client

[25] PECIOSKI, Kliment. Narrow band IoT communication for Modbus/TCP devices. Thesis (Master of Science in Electrical Engineering) - Politecnico di Milano, School of Industrial and Information Engineering, 25-Jul-2018. Available at: http://hdl.handle.net/10589/141817.

[26] Agresti, A., & Finlay, B. (2009). Statistical methods for the social sciences (4th ed.). Pearson Education

List of Figures

List of Tables

Acknowledgements

Throughout the writing of this thesis, I have received a great deal of support and assistance. I would like to acknowledge and give my warmest thanks to my supervisor, Prof. Giacomo Verticale from Politecnico di Milano University, who made this work possible. His guidance and advice carried me through all the stages of writing this project.

In addition, I would like to thank my parents for their wise counsel and sympathetic ear. They have always been there for me and provided unwavering support.

Finally, I could not have completed this dissertation without the help of my friends. Their encouragement and assistance were invaluable and greatly appreciated.