LTE Cat-NB (Narrowband) Performance Evaluation

Daniel Robinson

Stellenbosch University, November 2019

Contents

List of Tables

List of Figures

1 Design and Methodology

As stated in §**??**, the aim of this study is to compare user equipment (UE) against mobile network operators (MNOs) with a set of tests that evaluate NB-IoT's performance according to a set of metrics which highlight striking differences due to the underlying complexities of LTE architecture.

Four mobile network operators (MNOs) are compared in South Africa according to the underlying vendor infrastructure used, namely Nokia and ZTE in the Cape/coastal regions and Ericsson and Huawei based in Gauteng/inland regions.

More than one UE is used to improve the accuracy of the result, namely Ublox and Quectel. A unit testing framework has been carefully prepared in Python in combination with a Hewlett Packard rotary RF attenuator in 10dBm steps. The results can be applied to multiple application use cases.

1.1 Preliminary Tests

These tests better orient the reader to the behavior of UE devices and LTE network.

1.1.1 Network Info and Behavior

This section looks at certain informative aspects and behavior of LTE networks.

1.1.1.1 System Information Blocks (SIB) SIBs carry relevant information for the UE, which helps UE to access a cell, perform cell re-selection, information related to INTRA-frequency, INTER-frequency and INTER-RAT cell selections. In LTE there are 13 types of SIBs as can be seen in Table 1.

See Appendix **??** for examples of NB-IoT SIB blocks.

- Downlink `systemInformationBlockType1`
- Downlink `systemInformation`
- Uplink `rrcConnectionRequest`
- Downlink `rrcConnectionSetup`

Table 1: System Information Blocks description [1]

SIB	Description
MIB	Master Information Block which sends essential information required to receive further SIBs
SIB-1	Cell access related parameters and scheduling of other SIBs
SIB-2	Common and shared channel configuration, RACH related configuration are present
SIB-3	Parameters required for intra-frequency, inter-frequency and I-RAT cell re-selections
SIB-4	Information regarding INTRA-frequency neighboring cells (E-UTRA)
SIB-5	Information regarding INTER-frequency neighboring cells (E-UTRA)
SIB-6	Information for re-selection to INTER-RAT (UTRAN cells)
SIB-7	Information for re-selection to INTER-RAT (GERAN cells)
SIB-8	Information for re-selection to INTER-RAT (CDMA2000)
SIB-9	Information related to Home eNodeB (FEMTOCELL)
SIB-10	ETWS (Earthquake and Tsunami Warning System) information (Primary notification)
SIB-11	ETWS (Earthquake and Tsunami Warning System) information (Secondary notification)
SIB-12	Commercial Mobile Alert Service (CMAS) information.
SIB-13	Contains the information required to acquire the MBMS control information associated with one or more MBSFN areas.

It is important to realize how intricate the underlying architecture of LTE is. For example, considering the singalling between UE and eNodeB using SIBs, we see this in action. This complexity hints that the probably cause of variation is due to the LTE network configuration.

1.1.1.2 Extended Coverage Level (ECL) Extended Coverage Levels increase the amount of repetitions between UE and eNodeB to increase range. Henceforth, this should mean that a weaker signal strength increases the ECL level. There are 3 levels, with level 0 being the least repetitions, and 2 being the most.

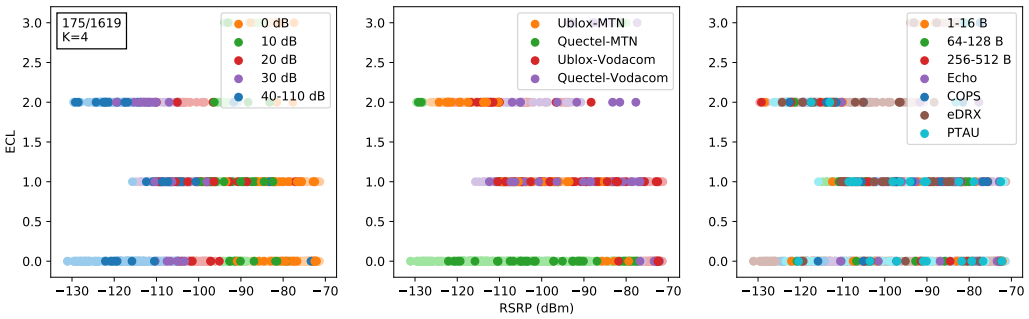


Figure 1.1: ECL levels shown against RSRP for Ubloxa and Quectel on ZTE-MTN and Nokia-Vodacom networks.

In this Fig. 1.1, ECL is shown as an example against two networks and it seems apparent that it is not determined by attenuation. Further investigation is necessary.

1.1.1.3 Cell ID, EARFCN, PCI These identifiers are related to the specific cell towers the UE is connected to.

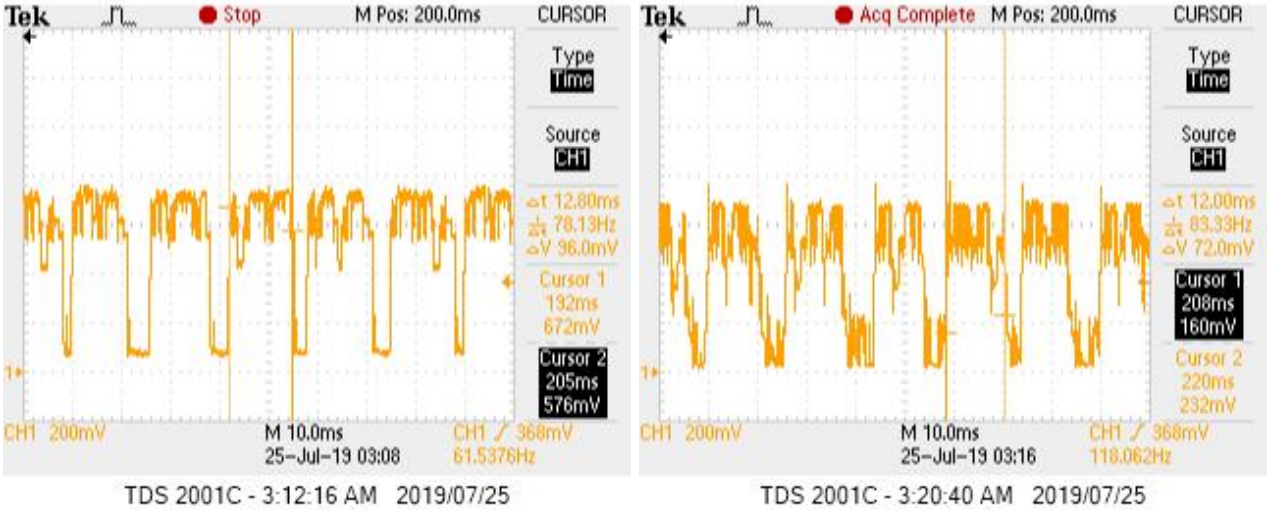
The Cell ID is the physical network cell ID. EARFCN uniquely identifies the LTE band and carrier frequency. PCIs, or Physical Cell Identifiers provide a psuedo-unique value for identifying eNodeBs and is a unique identifier for serving cells. The PCI value is created from two components - PSS and SSS. The PSS, Primary Synchronization Signal, has the value 0, 1, or 2. The SSS, Secondary Synchronization Signal, can have a value between 0 and 167.

Table 2: PCI, Cell ID and EARFCN count as a result of registrations with LTE networks. Tuples are in (Ublox, Quectel) format.

PCI	Cell ID	ZTE-MTN	Nokia-Vodacom
123	239882509	(34, 26)	
14	2671716	(13, 29)	
11	2672484	(1, 4)	
2	484196		(34, 32)
	EARFCN		
	3712	(48, 59)	
	3564		(34, 32)

In Table 2 we see three cell towers on the MTN-ZTE network. More than one tower at the same frequency or EARFCN proves that Intra-Frequency Cellular Reselection works as expected.

1.1.1.4 C-DRX mode On the Vodafone network in connected-DRX (C-DRX) mode, the UE is observed to show peaks spaced at regular 2.048s intervals [2]. On both Vodacom and MTN networks, these peaks are not visible and instead a steady stream of peaks can be seen as on the following images.



(a) C-DRX timing on MTN-ZTE-Ublox

(b) C-DRX timing on MTN-ZTE-Quectel

Figure 1.2: Timing measurement of two UEs on MTN-ZTE during C-DRX. Although the duty cycles vary in C-DRX mode, it can be estimated that pulses are roughly 12ms in length with 4ms idle between. This means that 75% of the time the UE device is drawing current.

In Fig. 1.2a, the Ublox UE uses 73.6mA at 110dB attenuation with the RF shield enclosure door slightly open and in Fig. 1.2b, with the same environment the Quectel UE uses 73.6mA. Observing C-DRX on the Nokia-Vodacom network, we have slightly different results as can be seen summarized in Table 3. It seems that on ZTE-MTN and Nokia-Vodacom that cycles are of length 16ms and 256ms respectively.

Table 3: Table showing a summary of C-DRX values

	ZTE-MTN	Nokia-Vodacom
Ublox		
Peak current	73.6 mA	72 mA
Transmit time	12.8 ms	56 ms
Idle time	4 ms	200 ms
Quectel		
Peak current	70.4 mA	66.4 mA
Transmit time	12 ms	80 ms
Idle time	4 ms	180 ms

On the MTN-ZTE network the peaks indicate an on time of roughly 12ms and idle of 4 seconds. With a cycle of 16ms, it fits the LTE requirements of between 10ms and 2560ms in terms of 1ms subframes. However, NB-IoT has a minimum requirement of 256ms to 9216ms for the interval length between C-DRX transmissions and Vodacom-Nokia is using this minimum value. MTN-ZTE is utilizing vastly more time on air than permitted by the 3GPP and it is having a detrimental effect on the estimated battery life. Vodacom-Nokia is using the minimum, but it is recommended to increase this value. Lastly, this does not bode well for the scaling up of devices due to the interference, especially on the shared uplink (NPUSCH) channel.

1.1.1.5 E-UTRAN Node B (eNB/eNodeB) Ericsson eNodeBs run Linux and their commands are accessible via MOShell, or the scripting language AMOS.

To get an idea of the complexity of a node (eNodeB) in a base station (BTS), running `$ get .` in the terminal of B06009-TESTPLANT returned 7037 `Managed Objects` (MOs) with 27989 parameters. See Appendix **??** for an example code snippet of the first two `Managed Objects`. This highlights how easy it is for a BTS to produce different results in this study depending on the network configuration and environment.

1.1.2 Range Field Test

This gives a good idea as to the range expected according to RSRP, with more information in §1.3.4.6.

Using a Quectel BG96, the following tests were taken on the rooftop described in Fig. 1.3[1].

Figure 1.3: Rooftop outside the HF RF lab on the 5th floor of the Electrical & Electronic Engineering building. The base station it connected to is on the General Building, and is just over 150m away at the same elevation with a single building blocking line-of-sight. The base station is situated on the bottom left of the picture at an altitude of approximately 138m.

The tests involve sending a set of 10 pings multiple times at a certain attenuation and resulting RSSI measurement using a Quectel BG96 modem.

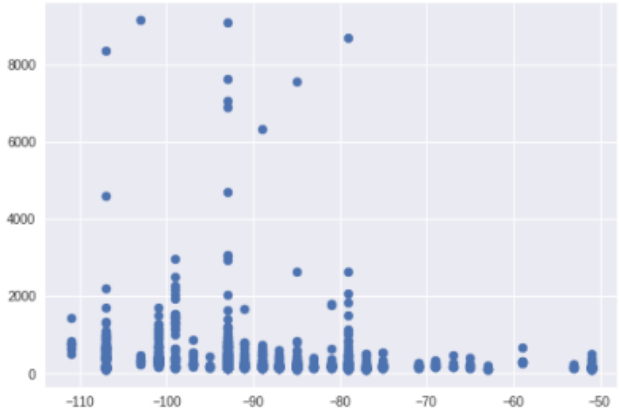
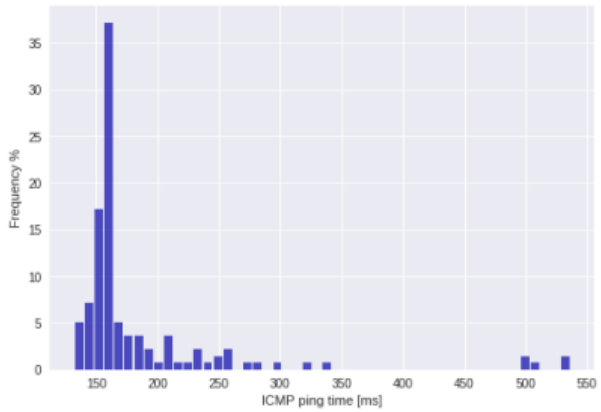
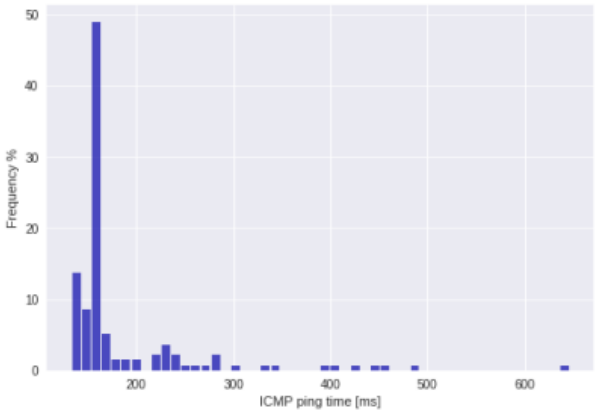


Figure 1.4: Looking at the ICMP ping response according to different RSSI values, we see high jitter of a few seconds from -80dBm or less. This means that in an urban area, NB-IoT satisfies the 2-5 km range specification.

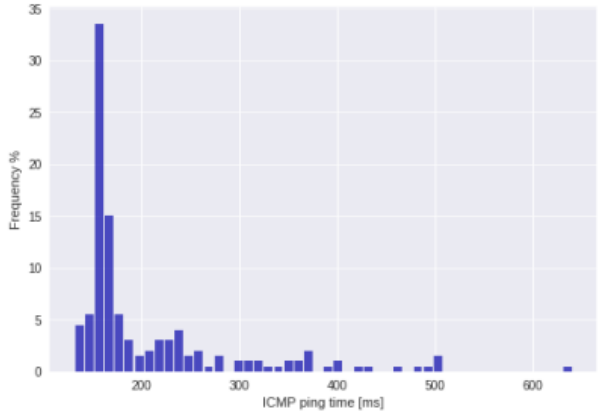
[1]The MTN-ZTE test dataset §1.7.1 was captured inside the RF enclosure inside the HF RF lab.



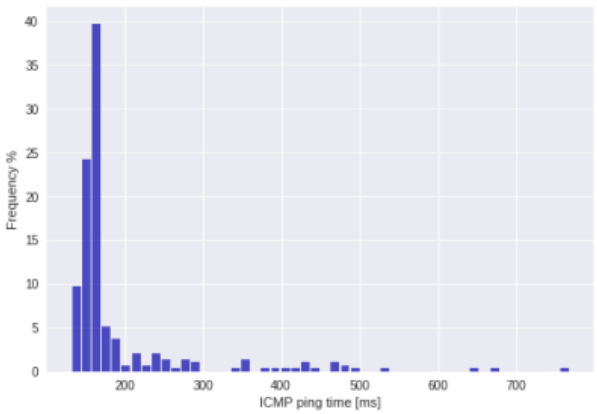
(a) 0 dB attenuation test in ECL class 0 at -51 dBm RSRP. With an antenna and the attenuator set to 0dB, we find most of the values around the mean of 185.2ms, except for the tail at around 500ms which is the time of the first ping in a set of 10.



(b) 110 dB attenuation test in ECL 0, at -85 dBm RSRP. Setting the attenuator to the max of 110dB, we see no change in the ping measurements which have a mean of 185.9ms. The tail has increased to a max of just over 600ms.



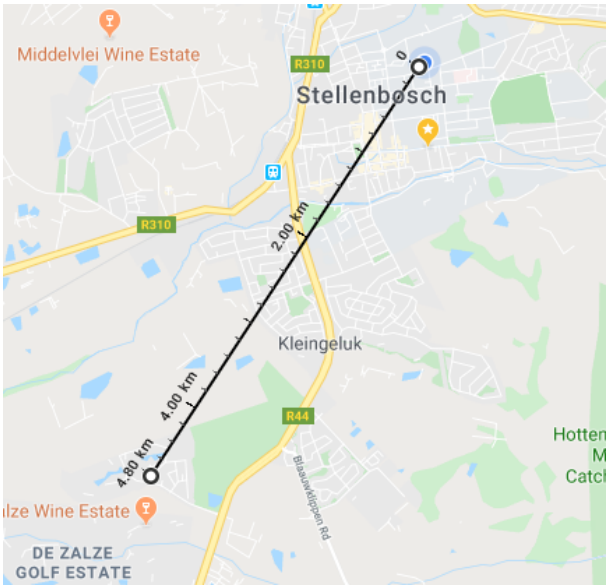
(c) Attenuator and no antenna test in ECL 0 at -91 dBm RSRP. Removing the antenna from the attenuator, we find that the data has a slightly thicker tail, and averages around 207.1ms.



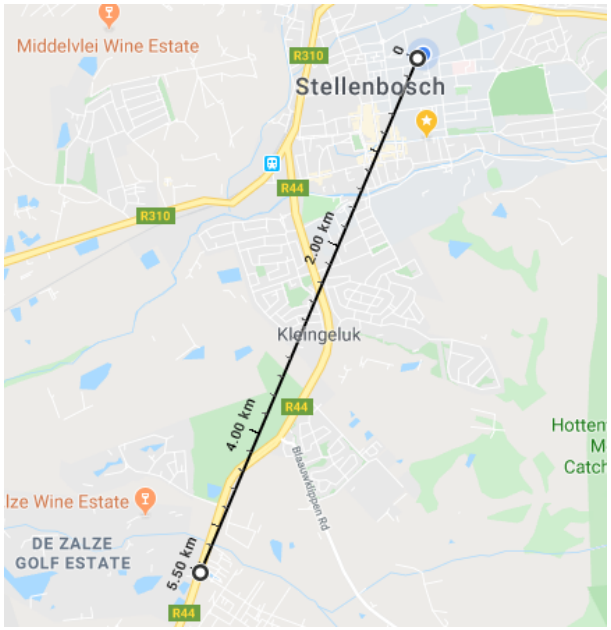
(d) No attenuator and no antenna test in ECL 1 at -107 dBm RSRP. Lastly, having no attenuator nor antenna we still have a connection at -107dBm with a mean of 190.6ms.

Figure 1.5: Ping tests on Engineering rooftop with time in milliseconds on x-axis and the percentage frequency on the y-axis. Here we see how ECL class 0 and ECL clsas 1 is quite similar.

To be able to attenuate the signal until disconnection, one must increase the range from the base station such that leakage transmission from traces, soldering and attenuator connectors do not interfere with the test. As such, there must not be a connection to the base station at all if the antenna or attenuator is disconnected or connected at maximum attenuation.

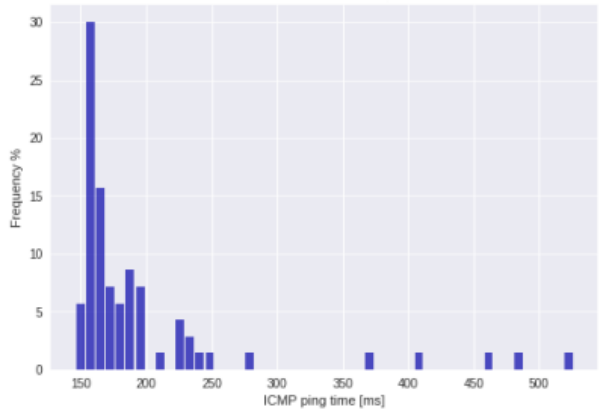


(a) A test was performed from 10pm onwards at Technopark on 14 March 2019. A connection was made at a range of 4.8 km at -93dBm and an altitude of 132m. This is a relative elevation of -6m to the base station.

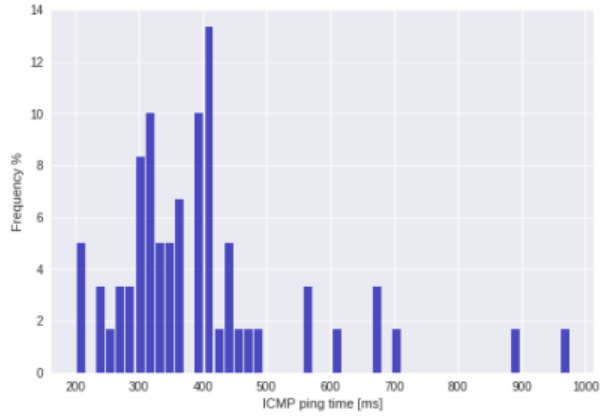


(b) The greatest distance measured was 5.5km from the intersection of the R44 and the turn-off to Stellenbosch Square or Jamestown at an altitude of 106m. This is a relative elevation of -32m to the base station and at an RSRP of -89dBm.

Figure 1.6: Long-distance tests were performed, and these two maps show the maximum distance that signals travelled.

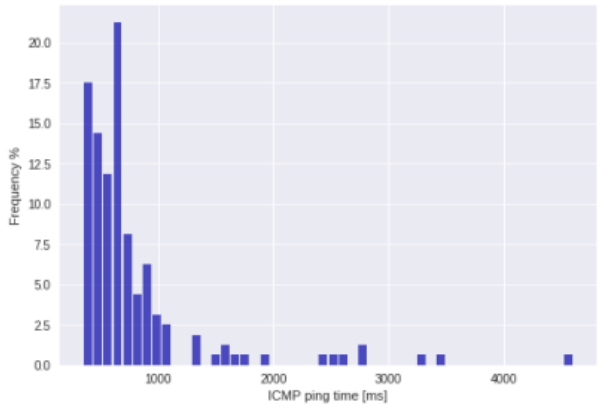


(a) 0 dB attenuation test in Technopark at -93 dBm RSRP in ECL class 0. With no attenuation, the data has a mean of 196.7ms and a tail just above 500ms.

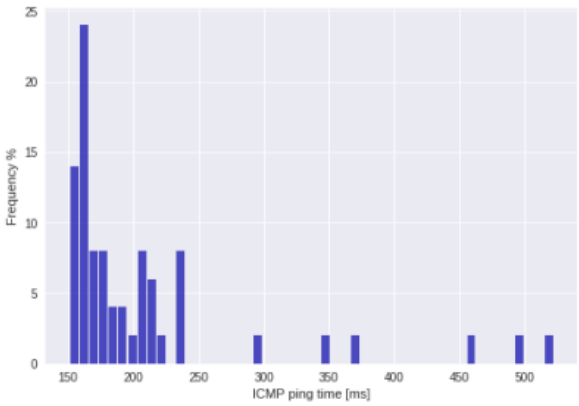


(b) 10 dB attenuation test in Technopark at -101 dBm RSRP in ECL class 1. In this condition, the data is more spread out from 200 - 500ms with a mean of 396.4ms and a tail at just under 1000ms.

Figure 1.7: Long-range ping tests in Technopark starting from 0 dB attenuation and adding 10 dB.

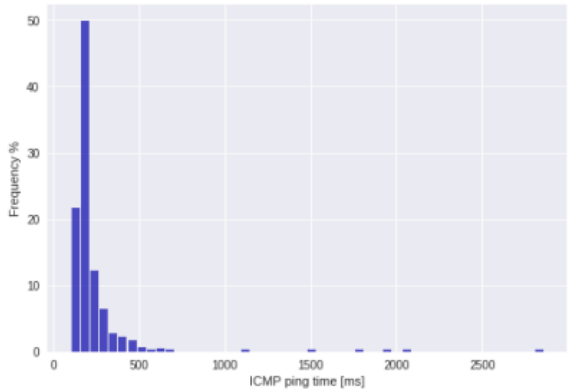


(a) 20 dB attenuation test in Technopark at -107 dBm RSRP in ECL 2. At 20dB attenuation, the data is more spread across 350 - 1000ms with a mean of 793.4ms and a tail that extends to over 4500ms. Any more attenuation and the signal is lost.

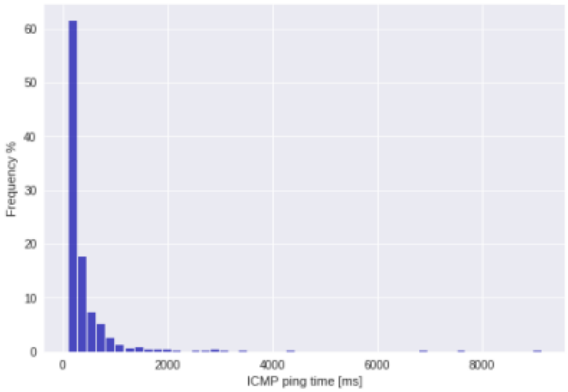


(b) 0 dB attenuation test at Stellenbosch Square R44 intersection at -89 dBm in ECL 0. At the furthest point in Fig. 1.6b, the signal strength increased to -89dBm and resumed a mean of around 209.6ms with a tail around 500ms.

Figure 1.8: Long-range ping tests in Technopark up to 20 dB attenuation and at Stellenbosch Square R44 intersection not surviving more than 0 dB attenuation.



(a) 0 dB attenuation test at Parmalat at -87 dBm in ECL 0. A similar pattern was seen 3.0 km away from the base station at Parmalat, although driving closer there were a few spots where connection was lost or many retries were needed such that the tail extended up to almost 3000ms for the ICMP ping time.



(b) Lastly, all the test data (including on the way to Technopark and back), we see a similar form except with a tail extending to almost 10 seconds, which is within 3GPP specifications.

Figure 1.9: Long-range ping tests show a few results at ECL class 2, which shows how different ECL class 2 from class 1 and 0, with the differentiating factor beign a couple of seconds latency as opposed to a few hundred milliseconds. and thus a factor 10 difference.

1.1.3 RF Spectrum Tests

Using an RTL2832 SDR dongle, we can capture RF signals. At the very least we can visualise how the signal propogates through the airspace.

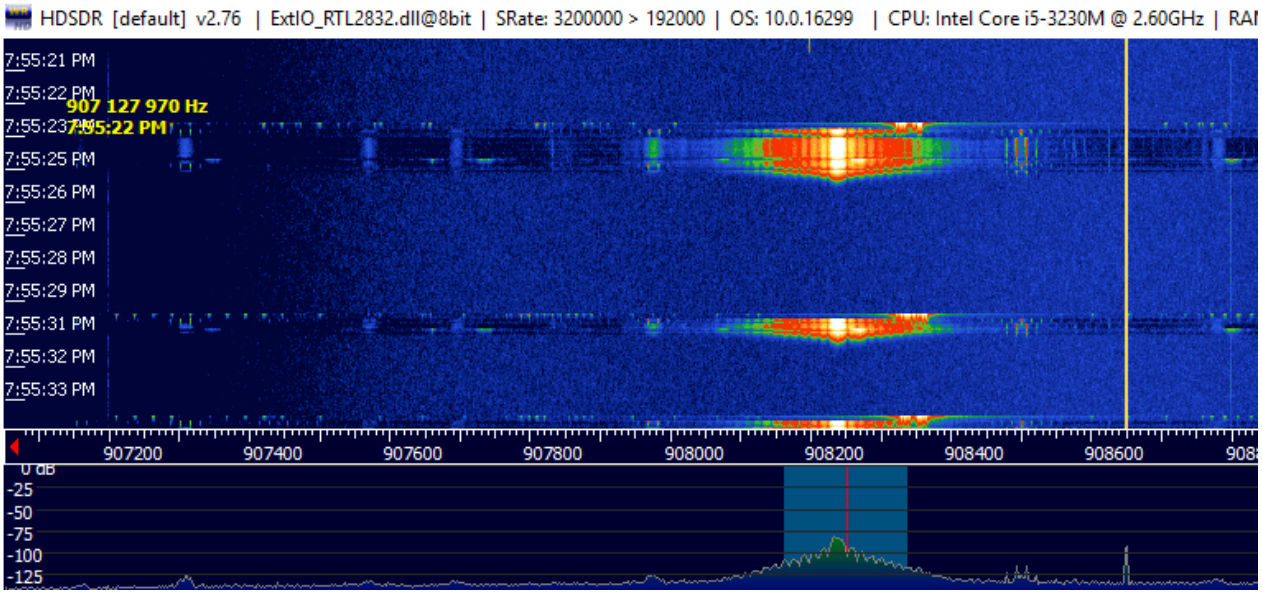


Figure 1.10: 5 dB SINR NB-IoT transmissions using Sierra Wireless WP7702 at 908.2 MHz and EARFCN 3734 of length 2282ms, 1560ms and 1380ms respectively.

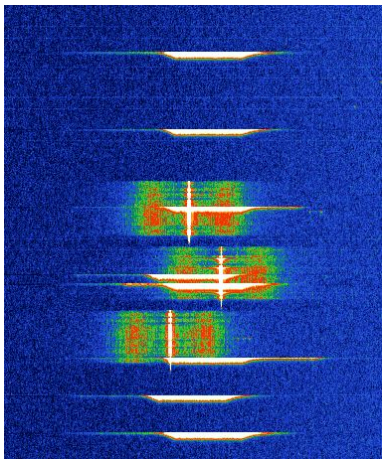


Figure 1.11: SigFox and LoRa RF signals @868 MHz on a waterfall diagram, with the x-axis showing frequencies at 868 MHz and the y-axis over time. The SigFox signals (vertical) take about 2 seconds to transmit, and the LoRa signals (horizontal) take a few hundred milliseconds.

Each technology has their own modulation scheme and unique features, and with that their own set of advantages and disadvantages. More can be found in §??.

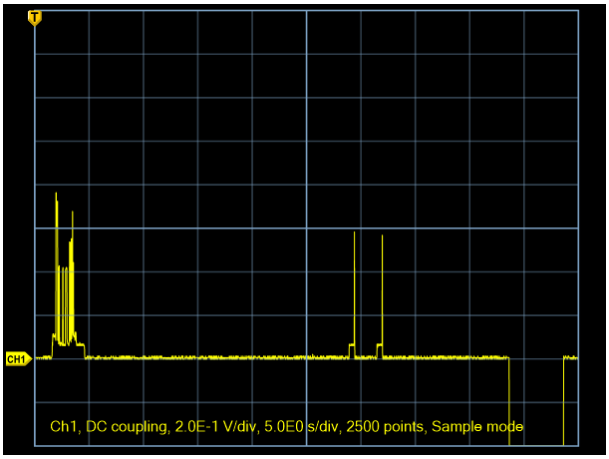
1.1.4 Power Saving Mechanisms

This section shows a brief investigation into the power saving mechanisms of NB-IoT as mentioned in §1.1.4.

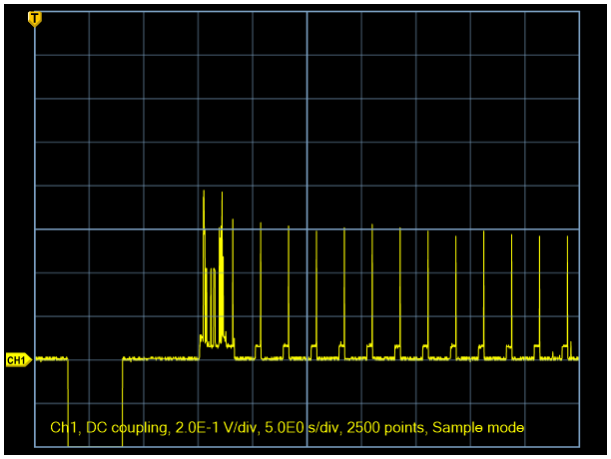
With a paging time window interval of 2.54s and 4 hyper-frames making up 40.96s, the following output is obtained.

```
AT+NPTWEDRXS=2,5,"0001","0011"
+CEREG: 5,1,"8CA7","28C464",7,,,"00011000","00101010"
```

AT+CEREG says that the T3324 active time is 48 seconds, or 2 seconds * 24 binary coded timer value. This is not the expected outcome, even according to "Table 10.5.5.32/3GPP TS 24.008: Extended DRX parameters informatio" as referenced in Ublox documentation, which expects 40.96s. Besides, the paging time interval is also not working as expected as in Figure 1.19a.



(a) Initial attempt at setting eDRX mode failed.



(b) Successful attempt at setting eDRX mode.

Figure 1.12: eDRX tests

The T3324 active timer value is then modified to 5.12s as in Figure 1.19b.

```
AT+NPTWEDRXS=2,5,"0001","0000"
```

AT+CEREG says that the timer is 32s, or 2 seconds * 16 binary coded timer value.

```
+CEREG: 5,1,"8CA7","28C465",7,,,"00010000","00101010"
```

In the debug logs we see the timer expires after exactly 30 seconds.

```
1400,00:07.952393,NAS_DBG_TIMER
    action=TIMER_START
    prim_id=USIM_STATUS_TIMER_EXPIRY
    duration=30
2092,00:37.952728,USIM_STATUS_TIMER_EXPIRY
    timer_handle=16871576
```

Increasing the T3324 active timer value to 10.24s, the following results are obtained. It is exactly the same as before.

```
AT+NPTWEDRXS=2,5,"0001","0001"
```

AT+CEREG says that the timer is 32s, or 2 seconds * 16 binary coded timer value.

```
+CEREG: 5,1,"8CA7","28C465",7,,,"00010000","00101010"
```

In the debug logs we see the timer expires after exactly 32 seconds.

```
2409,+00:00.400757,NAS_DBG_TIMER
    action=TIMER_START
    prim_id=USIM_STATUS_TIMER_EXPIRY
    duration=30
5981,+00:33.283905,USIM_STATUS_TIMER_EXPIRY
    timer_handle=16871576
```

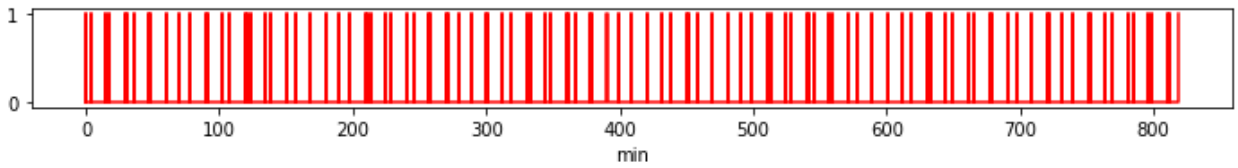


Figure 1.13: This test shows eDRX events until 800 minutes, measured externally. It shows an irregular eDRX time when not properly configured.

It is important to note that if eDRX time is not configured properly, then the outcome does not show as expected as in Fig. 1.13.

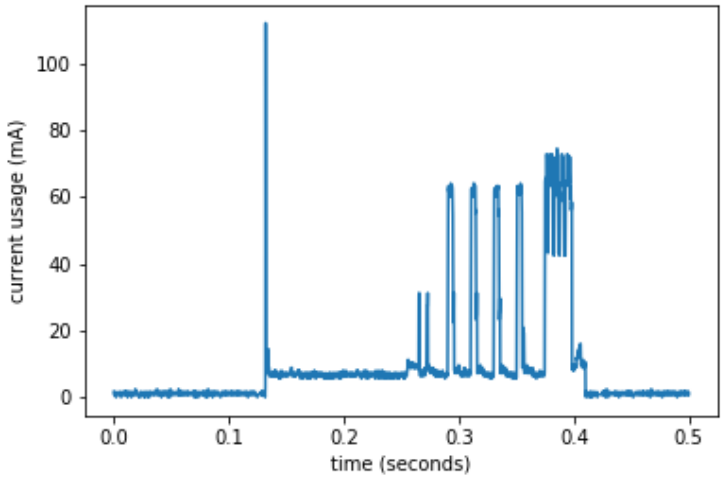


Figure 1.14: Typical eDRX current profile with a 277.8 ms duration. This shows how for the first few microseconds there is a large transmission current spike to synchronize, before receiving paging information from the cell tower.

Considering an eDRX event with a typical current profile as shown in Fig. 1.14, the debug trace shown here every 2.56 seconds for ZTE (same for Ericsson and Huawei but Nokia has a cycle every 10 seconds), shows the following information. Besides logs showing time synchronization and other network information, the serving cell logs show signal strength metrics.

```
102332,03:20.082306,LL1_EXIT_LOW_POWER_MODE
102334,03:20.082367,LL1_LOG_FAST_FORWARD_TIME_CALC
102360,03:20.165100,LL1_RESYNC
102362,03:20.165222,LL1_TIMING_ADJUST_LPM_WAKEUP
102364,03:20.165344,LL1_LOG_CURRENT_TIME_CALC
102370,03:20.205993,PROTO_LL1_SERVING_CELL_MEASUREMENT_IND
102372,03:20.206054,LL1_NRS_MEASUREMENT_LOG
```

```
102377,03:20.206573,LL1_LOG_CURRENT_TIME_CALC
102378,03:20.206665,LL1_CALC_PAGING_DATA
102381,03:20.207062,LL1_KV_CACHE_FLUSH
102382,03:20.207123,LL1_ENTER_LOW_POWER_MODE
102384,03:20.207336,LL1_SERVING_CELL_MEAS_IND
102385,03:20.207458,RRC_DBG_RESELECTION_MEASUREMENTS
```

1.1.5 Ultra-low Current Sleep Measurements

During deep sleep, UE devices typically use only a couple of microamps.

Using an MX 58HD DMM, one can measure the microamp sleep currents of UE devices. Testing the accuracy of the DMM, 4.501 mA is measured through a 4615 ohm resistor at 21.15V. Theoretically it should be 4.582 mA so that gives an error tolerance of 1.82% or ~2%.

The Ublox device consumed about 3.6 uA, and Quectel consumed 4.2 uA, which is close to manufacturer specifications.

1.1.6 Mobility Tests

There was a brief test done on mobility to show how NB-IoT devices can transition to different radio access technologies (RATs).

The Sierra Wireless 7702 has a Qualcomm 9206 modem which supports LTE Cat M1, NB1 and EC-GSM. Using a Sierra Wireless WP7702 on Ericsson Test BTS 'L06009A3' and EARFCN 3734/3752, the UE had to periodically ping an internet-facing server and the dead time was measured before it reconnected and received a response. The RSRP was in the range -50 to -80 dBm and in ECL 0. The tests took place within a faraday cage to isolate the test network from the live RAN, else by opening the door of the faraday cage it deregistered from the network and MME.

Table 4: NB-IoT and LTE Cat-M handover.

Mobility test	Time
Standalone to In-band	~ 11 s
In-band to Standalone	~ 11 s

1.1.7 Throughput

An initial throughput test was performed *not* using AT commands, but a linux script.

NB-IoT downloading was tested on the Sierra Wireless 7702 using the following script.

```
while [ 1 ]; do
    # wget --retry-connrefused --waitretry=1
    #   --read-timeout=20 --timeout=15 -t 0 --continue
    wget -t 0 -c http://speedtest.ftp.otenet.gr/files/test100k.db
    # check return value, break if not successful (0)
    if [ $? != 0 ]; then break; fi;
    sleep 1s;
done;
```

A 100 kb file is downloaded at a rate of around 3kB/s. The script continues download if stalled or other errors occur. Since it is a `Yocto` installation[2], the other `wget` arguments were not available.

[2]It's not an embedded solution. Rather, it creates a custom one for you, regardless of hardware architecture [3].

Table 5: Table showing preliminary throughput tests for GPRS, NB-IoT and LTE-M

	Uplink	Downlink
GPRS	158 kbps	254 kbps
NB-IoT	56 / 65 kbps	24 / 27 kbps
LTE-M	293 / 375 kbps	264 / 300 kbps

The Sierra Wireless 7702 is a powerful board, and it showed satisfactory throughput rates.

1.2 Example Application

An example application was built to test and understand NB-IoT. See schematic and board layout in Appendix **??**. The board includes not just NB-IoT but also LTE Cat-M, GPRS/EDGE, SigFox, LoRa, and Dash7. Initally designed to compare LPWANs, it was decided to focus more purely on NB-IoT as there is a great deal of variability among UEs and LTE vendors already.

Notable components include:

- Quectel BG96 cellular modem
- Murata CMWX1ZZABZ-078 which includes STM32L072CZ microcontroller and SX1276 transceiver
- Atmel SAMD21G18a microcontroller
- Microchip MIC29302WU 3A LDO Regulator **???**

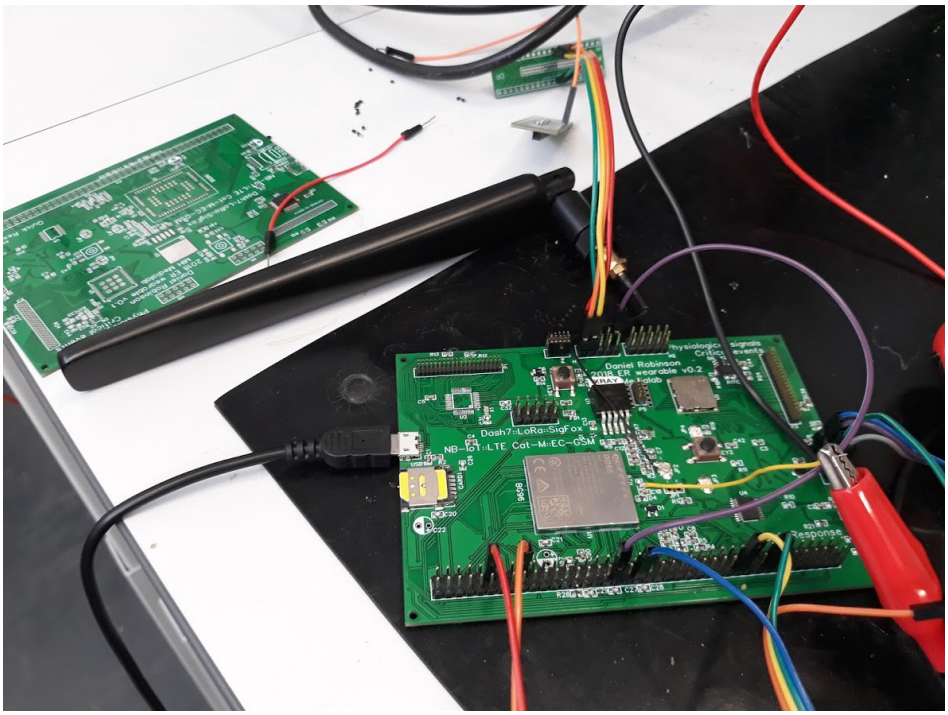


Figure 1.15: This is a PCB application developed as an example.

By adding a DHT22 temperature and humidity sensor, button and buzzer for and example application, we see the following dashboard result in Fig. 1.16.

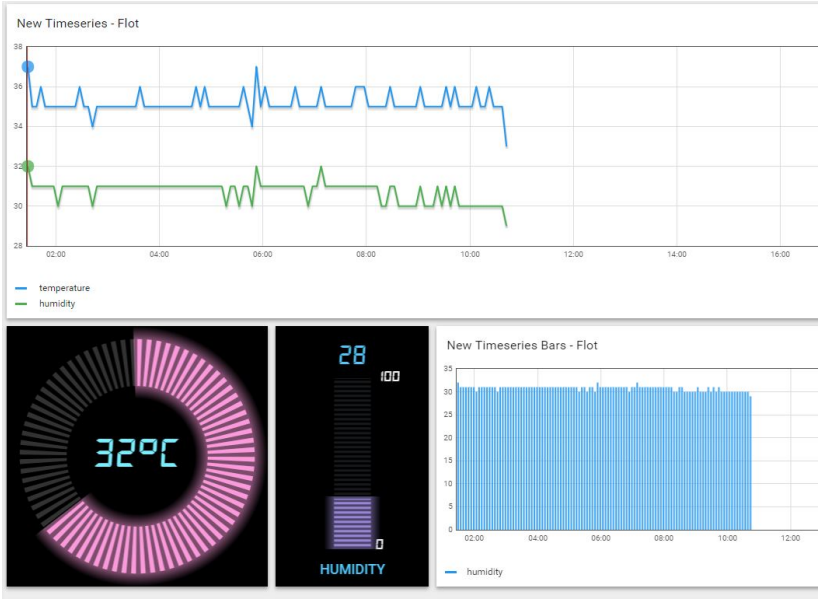


Figure 1.16: Dashboard showing communication between NB-IoT PCB and Thingsboard cloud via MQTT with temperature, humidity, push button and downlink buzzer control

Luckily UE manufacturers usually provide a development kit with open source schematics and board layouts. This study will use development kits so that tests are easily reproduceable.

1.3 Setup Procedure

Each field test will make use of various UE hardware and telemetry tests and this section outlines the steps taken to perform these field tests.

1.3.1 Hardware

This section outlines some of the hardware configurations required for field test captures.

1.3.1.1 Attenuator Two of these will be used in series: the HP8494B and the HP8496A. One has a range of 11dB in 1dB steps, and the other has a range of 110dB in 10dB steps, so it is possible to get a full range of 110dB in 1dB steps.

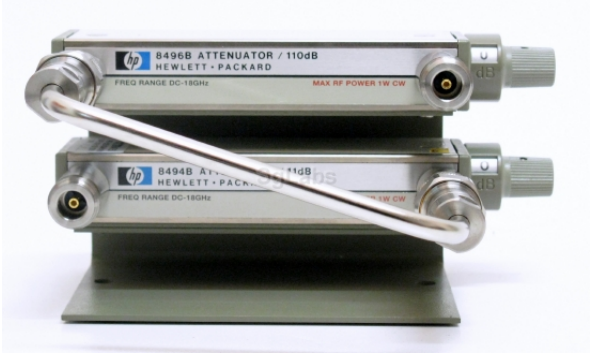


Figure 1.17: The Hewlett Packard attenuators used in this study to change the RF conditions for UE devices against multiple LTE vendors

The 1 dB attenuator is useful to attenuate the signal strength until the RSRP is on a decade multiple of 10. This way variation around the decade is more visible.

1.3.1.2 Current Measurements By measuring current, the field tests can measure the energy usage of each datagram packet.

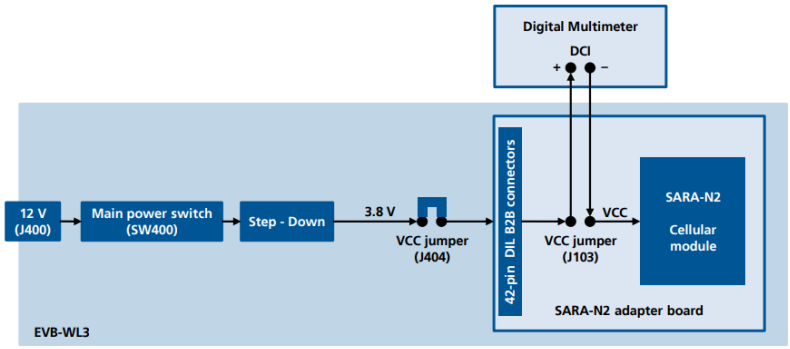
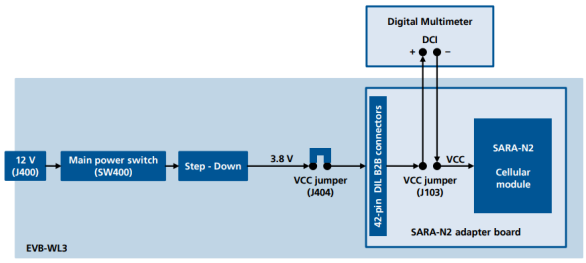
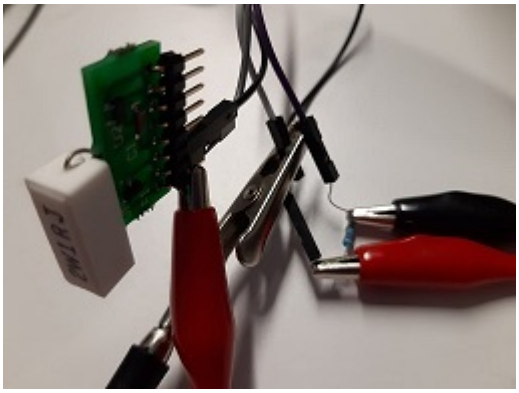


Figure 1.18: Block diagram of current consumption setup for SARA-N2 **??**

The digital multimeter in Fig. **??** is replaced with a ZXCT1008 high-side current monitor in series with the modem.



(a) ZXCT1008 high-side current monitor schematic



(b) ZXCT1008 high-side current monitor in action

Figure 1.19: Diagrams shows ZXCT1008 high-side current monitor which can be found on https://www.diodes.com/assets/Datasheets/ZXCT1008.pdf

Rs is set to a 1 ohm resistor and Rg is set as a 1k ohm resistor such that 100mA supplied to the modem makes 1V.

$$V_{out} = I_{load}[mA] * 10[\frac{V}{mA}] \tag{1}$$

1.3.1.3 Energy Capture Device The energy capture device measures the energy of each packet, and also returns the duration timings of each datagram packet for latency measurements.

`PlatformIO` compiles code for the microcontroller, and in this case it is a simple Atmel ATmega328P 8-bit microcontroller.

```
void energyLoop(boolean pause) {
    uint8_t reading = analogRead(A0);
    if (reading > 60) {
        if (reading > maxReading) maxReading = reading;
        if (!readCount++) {
            tStart = millis();
            idleTime = tStart - tEnd;
        }
        tEnd = millis();
        zeroM = tEnd;
        zeroCounter = 0;
        sum += reading;
        tStepCount += micros() - tStep;
    }
    else if (pause) zeroM = millis();
    else if (millis() - zeroM < 1000);
    else if (readCount) {
        txTime = tEnd - tStart;
        tStepCount /= 1000;
        energy = sum * 500 / 1023.0 * tStepCount / 1000 / 1000;

        buf.flush(); tx[0] = '\0'; // energyFlush();
        buf.print(idleTime); buf.print(",");
        buf.print(txTime); buf.print(",");
        buf.print(tStepCount); buf.print(",");
        buf.print(energy); buf.print(",");
        buf.println(maxReading/2);
        Serial.print(buf); // energyPrint();

        sum=idleTime=txTime=readCount=maxReading=energy=tStepCount= 0; // energySetup();
    }
    tStep = micros();
}
```

Code can be found on https://github.com/daniel-leonard-robinson/masters/tree/master/code/edge/src. It connects to the ZXCT1008 mentioned in §1.3.1.2 and converts the results to energy measurements. It also returns via serial to the PyTest framework the timings of each datagram packet.

1.3.2 Network Registration

Network registration is important before a device can send data to the internet. As mentioned in §??, the right SIM cards are necessary. It may even be possible to use e-SIMs as in Fig. 1.20.

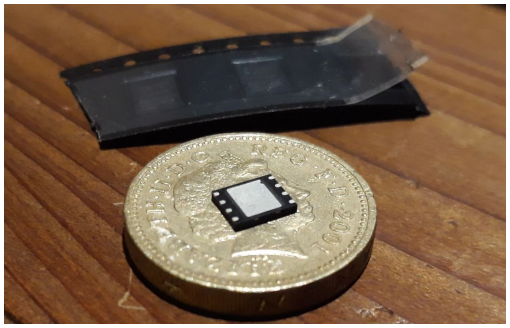


Figure 1.20: Hologram worldwide e-SIM

Finally, the right APNs are necessary. To use MTN's test network, the APN `rflab` is used. On Vodacom's network, the APN `nbiot.vodacom.za` is used.

1.3.3 PyTest Framework

A software framework was necessary to wrap AT commands into telemetry tests, include setup procedures.

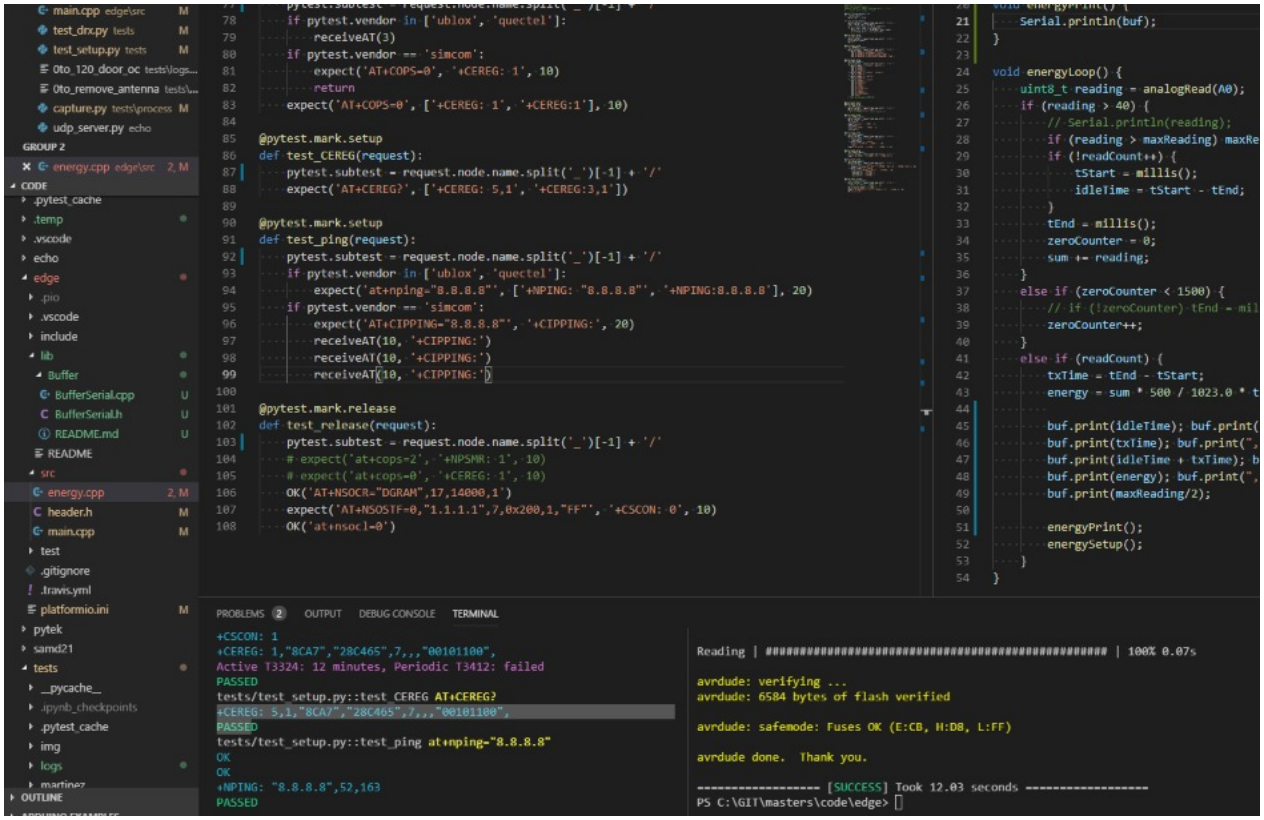


Figure 1.21: `Python PyTest` framework written in `Microsoft VS Code` and test output can be seen in bottom-left window. `PlatformIO` compiles microcontroller code and uploads via `avrdude` as can be seen in bottom-right window.

PyTest is a unit testing framework used to setup the UE for each test using AT commands and can be found

on https://github.com/daniel-leonard-robinson/masters/tree/master/code/tests. Although the testing framework is quite extensive, a few snippets of code will be discussed in this section to at least give an idea to the reader how this was developed.

Every test fixture includes the following setup and teardown code to open a serial connection to the UE. It automatically detects the COM port based on the USB vid.

```
def serialOpen():
    # setup for each test fixture
    global serAT, serTIM, serGPS, AT_PORT, uC_PORT
    ATcount = 0
    ports = serial.tools.list_ports.comports()
    for port, desc, hwid in sorted(ports):
        vid_pid = hwid.split('=')[1].split()[0]
        if vid_pid == '2341:8036':
            uC_PORT = port
        if vid_pid == '0403:6010' and not ATcount:
            AT_PORT = port
            ATcount += 1
            pytest.vendor = 'ublox'
        if vid_pid == '04E2:1414' and ATcount < 3:
            AT_PORT = port
            ATcount += 1
            pytest.vendor = 'quectel'
        if vid_pid == '0403:6001':
            AT_PORT = port
            pytest.vendor = 'simcom'
    try:
        serAT = serial.Serial(AT_PORT, 115200, timeout=1)
        serTIM = serial.Serial(uC_PORT, 115200, timeout=1)
        serGPS = serial.Serial(GPS_PORT, 9600, timeout=1)
    except serial.serialutil.SerialException as e:
        print(e)

def serialClose():
    # tear down for each test fixture
    global serAT, serTIM, serGPS
    serAT.close()
    serTIM.close()
    serGPS.close()
```

The setup and teardown functions are defined in a global file that is imported into each file of test fixtures. The location for new data in the database depends on chosen manufacturer (LTE vendor), location, file description and connected UE. The file description is the current RF attenuation and ranges from 0 to 110.

```
def setup_module(module):
    serialOpen()
    pytest.manufacturer = 'huawei' # 'ericsson', 'nokia', 'zte'
    pytest.loc = 'mtn/testplant_14th/'
    pytest.descr = '110'
    # pytest.lock = threading.Lock()

def teardown_module(module):
    serialClose()
```

See Appendix **??** to see how a Quectel or Ublox modem is set up. Running the following commands in Table 6 will set the device up.

Table 6: PyTest setup commands to be run in terminal

`pytest -svm apn`	Runs set APN fixture
`pytest -svm setup`	Runs all the setup fixtures
`pytest -svm reboot`	Reboots device if necessary

The following commands are wrappers for sending and receiving AT commands:

```
def OK(cmd, t=0):
    reply = sendAT(cmd, t)
    assert 'OK' in reply
    return reply

def expect(cmd, reply, t=1, output=True):
    replies = reply
    if str(type(reply)) == "<class 'str'>":
        replies = [reply]
    data = sendAT(cmd, t, replies, output)
    if not len(replies[0]):
        return data
    check = False
    for r in replies:
        if len(r):
            if True in [r in i for i in data]:
                check = True
                break
    if not check:
        print(magenta + str(replies), data)
    assert check
    return data

def sendAT(cmd, t=0, expect=['OK'], output=True):
    if output:
        print(yellow + cmd)
    serAT.write(bytes(cmd + '\r', 'utf-8'))
    return receiveAT(t, expect, output)

def receiveAT(t=0, expect=['OK'], output=True):
    if str(type(expect)) == "<class 'str'>":
        expect = [expect]
    c = 0
    data = []
    exp = expect[:]
    exp.append('ERROR')
    exp.append('FAILED')
    while True:
        d = serAT.readline().decode('utf-8')
        if not len(d):
            c += 1
        d = d.strip()
        if len(d) > 0:
            if output:
                print(cyan + d)
            out = converter(d)
            if out:
                print(magenta + out)
```

```
            data.append(d)
        if t > 0:
            if c >= t:
                data.append('timeout')
                return data
        for e in exp:
            if e in d:
                return data
```

Finally, the testing framework has a `capture` command which is blocking until an energy capture event. In this event the energy is sent via serial from the energy capture device (§1.3.1.3) and triggers the testing framework to extract information from the `AT+NUESTATS="RADIO"` command.

```
def receiveTIM():
    data = {}
    serTIM.flush()
    d = serTIM.readline().decode('utf-8') # d = '2300,260,2560,10.0,100,'
    if len(d):
        try:
            d = d.strip() # print(magenta + d)
            data['idleTime'] = int(d.split(',')[0])
            data['txTime'] = int(d.split(',')[1])
            data['totalTime'] = int(d.split(',')[2])
            data['energy'] = float(d.split(',')[3])
            data['maxCurrent'] = float(d.split(',')[4])
        except (ValueError, IndexError) as e:
            print(red + d)
            raise e
    return data
```

1.3.4 Telemetry Tests

The telemetry tests measure various aspects of the required metrics. Running the following commands in Table 7 will run through the desired telemetry test.

Table 7: PyTest telemetry test commands to be run in terminal

pytest -svm release	UDP test for multiple payload sizes
pytest -svk ptau	Run PTAU test
pytest -svk drx	Run eDRX test
pytest -svm reg	Run COPS test
pytest -svk echo	Runs Echo test

1.3.4.1 UDP UDP is used primarily for establishing low-latency and loss-tolerating connections between applications on the internet.

To test the capability of sending to the internet for multiple UEs, a simple protocol is necessary. TCP, MQTT, CoAP and other protocols are all based on the same IP infrastructure that UDP uses, yet not all UEs have this capability. UDP will be used and other protocols can be tested against it.

This test sends a UDP packet to an internet accessible IP address. The IP is 1.1.1.1 and it belongs to Warp which claims to be the fastest DNS resolver in the world, with OpenDNS, Google and Verisign taking the next respective rankings.

As an alternative, data can be sent to the u-blox echo server at `udp://echo.u-blox.com`. Because there is no DNS lookup function in the SARA-N2 module series, the server IP address that must be used is `195.34.89.241`.

UDP datagrams are sent with payloads of size 1, 16, 64, 128, 256 and 512 bytes.

Here is a snippet of one of the test fixtures for Ublox sending a 16 byte UDP payload with Release Assistance flags set.

```
@pytest.fixture(autouse=True)
def _config(request):
    pytest.test = 'release/'

...
@pytest.mark.release
def test_release_release16(request):
    pytest.subtest = request.node.name.split('_')[-1] + '/'
    if pytest.vendor == 'ublox':
        expect('at+nsocl=0', '')
        receiveAT(1)
        for i in range(5):
            OK('AT+NSOCR="DGRAM",17,14000,1')
            expect('AT+NSOSTF=0,"1.1.1.1",7,0x200,16,"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"',
                   '+CSCON: 0', 300)
            OK('at+nsocl=0')
            capture(1)
    ...
```

1.3.4.2 Periodic Tracking Area Update (PTAU)

This snippet sets up the eNodeB to schedule a PTAU event every 4 seconds (roughly ~5.5 seconds actual).

```
...
@pytest.fixture(autouse=True)
def _config(request):
    pytest.test = 'ptau/'

def test_ptau_set(request):
    pytest.subtest = request.node.name.split('_')[-1] + '/'
    setEDRX(4, 1, 0, 0, 3, 2) # 5.5 sec ptau
    capture(1)

def test_ptau_capture(request):
    pytest.subtest = request.node.name.split('_')[-1] + '/'
    capture(10)
```

1.3.4.3 Extended Discrete Reception (eDRX)

This snippet sets up the eNodeB to schedule a DRX event every 2.56 seconds.

```
...
@pytest.fixture(autouse=True)
def _config(request):
    pytest.test = 'drx/'

def test_drx_set(request):
    pytest.subtest = request.node.name.split('_')[-1] + '/'
    setEDRX(4, 1, 2, 5, 6, 2) # 2.56 continuous
    capture(1)

def test_drx_cap(request):
    pytest.subtest = request.node.name.split('_')[-1] + '/'
    capture(10000)
```

1.3.4.4 Cellular Operator Selection (COPS) Network Registration is necessary when the device is not yet connected. In Figure 1.22, An initial test was performed with AT+COPS=0 network registration until T3412 timeout of 270 seconds and peak current approximately 70mA.

Figure 1.22: `AT+COPS=0` network registration on MTN-ZTE network with lengthy inactivity timer setting of 270s.

This snippet registers the UE device on the network and as a workaround to shorten a long C-DRX inactivity timer of 10, 20 seconds or more (even up to ~265 seconds) it sends a UDP packet with a flag which tells the eNodeB that it would like to release the connection immediately, hence Release Assistance as mentioned in §??.

```
...
@pytest.fixture(autouse=True)
def _config(request):
    pytest.test = 'cops/'

############### reg release ###############
@pytest.mark.reg
def test_cops_register2(request):
    pytest.subtest = request.node.name.split('_')[-1] + '/'
    flushTIM()
    expect('AT+CFUN=1', 'OK', 3)
    expect('AT+COPS=0', ['+CEREG: 1', '+CSCON: 0', '+CEREG:1', '+CSCON:0'], 300)

@pytest.mark.reg
def test_cops_release(request):
    pytest.subtest = request.node.name.split('_')[-1] + '/'
    fetchTIM()
    if pytest.vendor == 'ublox':
        expect('at+nsocl=0', '')
        receiveAT(1)
        OK('AT+NSOCR="DGRAM",17,14000,1')
        receiveAT(1)
        expect('AT+NSOSTF=0,"1.1.1.1",7,0x200,1,"FF"', '+CSCON: 0', 100)
        OK('at+nsocl=0')
    elif pytest.vendor == 'quectel':
```

```
        ...
        ...
    receiveAT(1)
    receiveAT(1)
    fetchTIM()
    capture(1, 3)

############## dereg release ##############
@pytest.mark.reg
def test_cops_deregister(request):
    pytest.subtest = request.node.name.split('_')[-1] + '/'
    OK('AT+COPS=2', 5)
    receiveAT(300, ['+NPSMR:'])
    flushTIM()
    capture(1, 20)
...
```

1.3.4.5 Echo This test is designed to measure client and server initiated echo requests. The custom echo server replies immediately, then waits 10 seconds before sending another reply, hence a server initiated echo. Unfortunately, the inactivity timer resets and a UDP datagram with the Release Assistance flag set has to be sent.

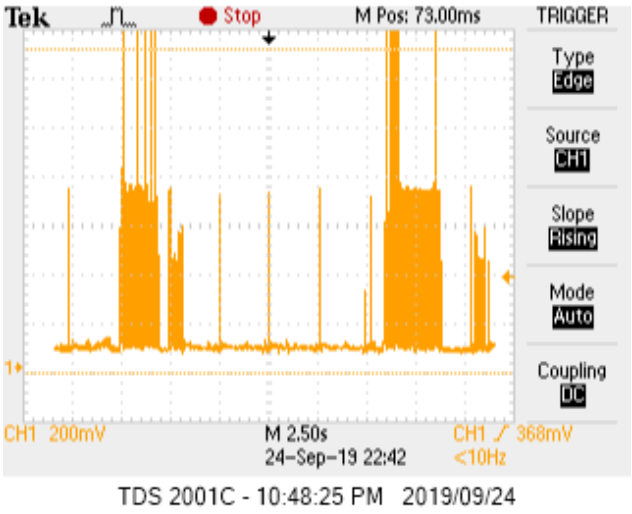


Figure 1.23: An oscilloscope measurement of the Echo test in progress. Notice the four DRX cycles before the second Echo response. Echoes worked successfully every time, and did not take another 2.56 seconds DRX cycle longer than the 10 second delay that the echo server induced.

The following snippet shows how the framework sends to a custom echo server (on `Google Cloud`) which responds immediately and then the echo server responds again after a ten second delay.

```
...
def test_echo_send(request):
    pytest.subtest = request.node.name.split('_')[-1] + ('512/' if big else '/')
    if pytest.vendor == 'ublox':
        expect('at+nsocl=0', '')
        receiveAT(1)
        OK('AT+NSOCR="DGRAM",17,4444')
        if big:
            expect('AT+NSOSTF=0,"34.74.25.60",5555,0x400,512,"3333333333333333333...
                   ... ... ... 333333333333333333 ... ... ...
                   ...3333333"', '+NSONMI: 0', 300)
        else:
            expect('AT+NSOSTF=0,"34.74.25.60",5555,0x400,3,"313232"', '+NSONMI: 0', 300)
        receiveAT(1, '+CSCON: ')
        OK('AT+NSORF=0,512', 3)
    ...
    capture(1, 8)
...
```

The custom echo server has a static IP (34.74.25.60) and is open on port 5555.

```
...
def receive_next(sock):
    "Repeatedly tries receiving on the given socket until some data comes in."
    logger.debug("Waiting to receive data...")
    while True:
        try:
            BUFFER_SIZE = 4096 # the buffer for receiving incoming messages
            return sock.recvfrom(BUFFER_SIZE)
        except socket.timeout:
            logger.debug("No data received yet: retrying.")
            pass

def receive_and_send_one(sock):
    "Waits for a single datagram over the socket and echoes it back."
    input_data, addr = receive_next(sock)
    message = input_data.decode()
    output_len = sock.sendto(input_data, addr)
    sleep(10) # 10 second delay before echoing back again
    output_len = sock.sendto(input_data, addr)

def start(args):
    "Runs the server."
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.settimeout(5) # seconds
    sock.bind((args.host, args.port))
    logger.info("Listening on %s:%s.", args.host, args.port)
    try:
        for i in itertools.count(1):
            receive_and_send_one(sock)
    ...
```

1.3.4.6 Ping The +NPING AT command can be issued to check if the module is able to send and receive data via the internet, or an internal network location. To ping Google's DNS server: AT+NPING="8.8.8.8"

The information text response to the +NPING AT command will be issued after a few seconds. If the information text response is +NPINGERR: 1, the ping has timed out (usually within 10 seconds). The first ping might fail because it can take a few seconds to connect to the base station. Use the +CSCON URC to show when the module is connected.

Whilst the simple `Ping` command is useful to measure connectivity and latency, it unfortunately has no way to release the inactivity timer by itself, which means the modem continues to consume current in receive-mode/C-DRX. That is why the `Echo` telemetry test was designed.

1.4 Primary Metrics

Primary metrics power efficiency and latency are investigated as mentioned in §**??**. Primary and secondary metrics have a few preliminary tests performed using Ublox and Quectel devices on MTN-ZTE and Vodacom-Nokia networks.

1.4.1 Power Efficiency

Power efficiency is one of the main metrics focused on in this study. This section outlines a few preliminary tests and the design for the final field tests comparing UEs and MNOs. Low power consumption is vital for battery longevity up to ~10 years or more. Power consumption is affect by various factors. In the hardware design, PCB layout, antenna matching and location will have an effect on the overall interference received by the module, SINR and ultimately transmit power. Transmit power also depends on the range and path loss to the UE device. With a weak signal, more repetitions are required, hence ECLs in §\ref{ECLs}. Other power saving mechanisms such as release assistance, PSM and eDRX mode work together to extend battery life as in §\ref{power-saving-mechanisms}.

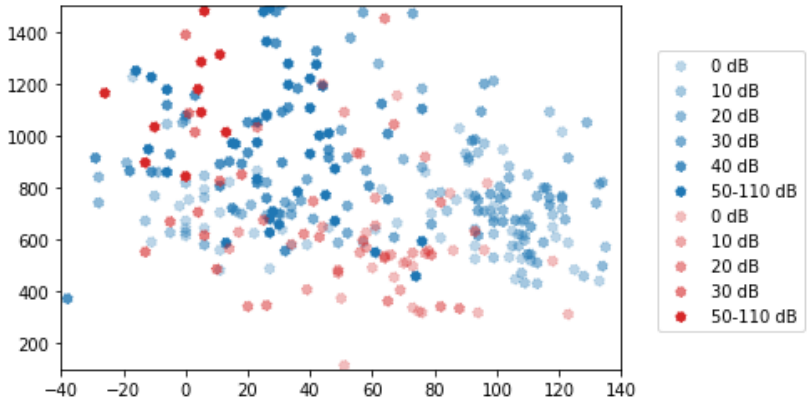


Figure 1.24: Ublox (blue) and Quectel (red) energy (J) per datagram as a function of the SINR (dB) as reported by the UE on the MTN-ZTE network limited to 1500 mJ. With the fading colour scheme and range just as in Martinez [2], Fig. 1.24 shows the impact of SNR on energy consumption. As observed in the figure, there is a trend of increasing energy with respect to lower SNR levels and high variability. Unfortunately, the effect of different ECLs is unclear.

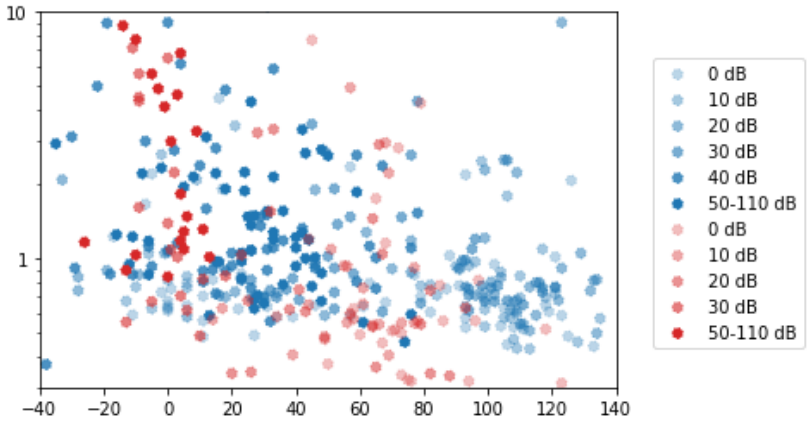
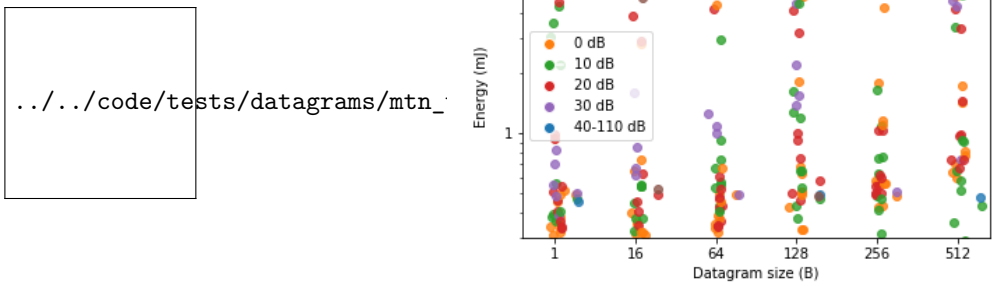


Figure 1.25: Ublox (blue) and Quectel (red) energy (mJ) per datagram as a function of the SINR (dB) as reported by the UE on the MTN-ZTE network. Increasing the range fully and using logarithms in Fig. 1.25, one can see that there is significant overshoot on the MTN-ZTE network. The trend mentioned in Fig. 1.24 continues.



(a) Datagram sizes of MTN-Ublox pair up to 1500 mJ.

(b) Datagram sizes of MTN-Quectel pair up to 10 J

Figure 1.26: UDP Datagram energy for different datagram packet sizes. Note the steady increase in energy consumption on the baseline, and the high variation. Although there is a slight trend in Fig. 1.26, it is not significant compared to the total variation for each datagram packet size.

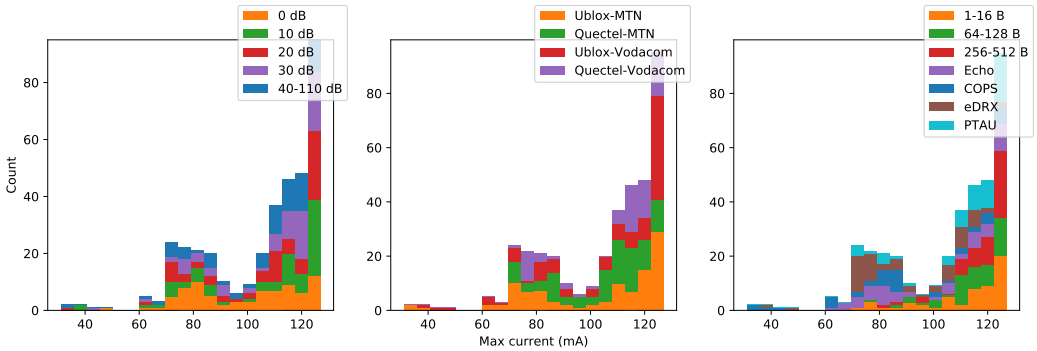
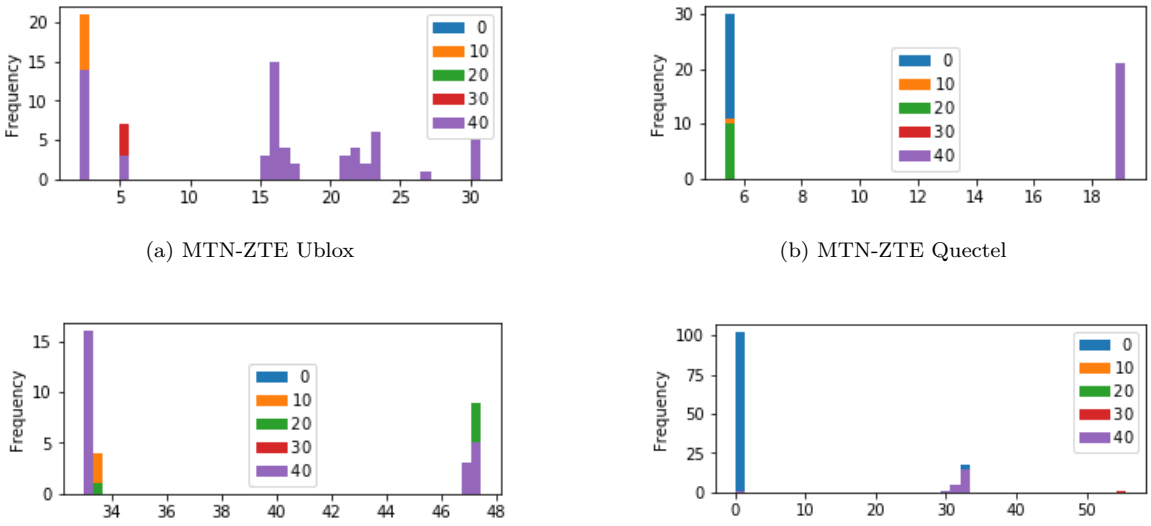


Figure 1.27: Peak energy measurements from roughly 70mA and skewed towards 128mA. These high peak energies shouldn't affect the average power much as peak current occurs only during the first few microseconds of the random access preamble (PRACH).



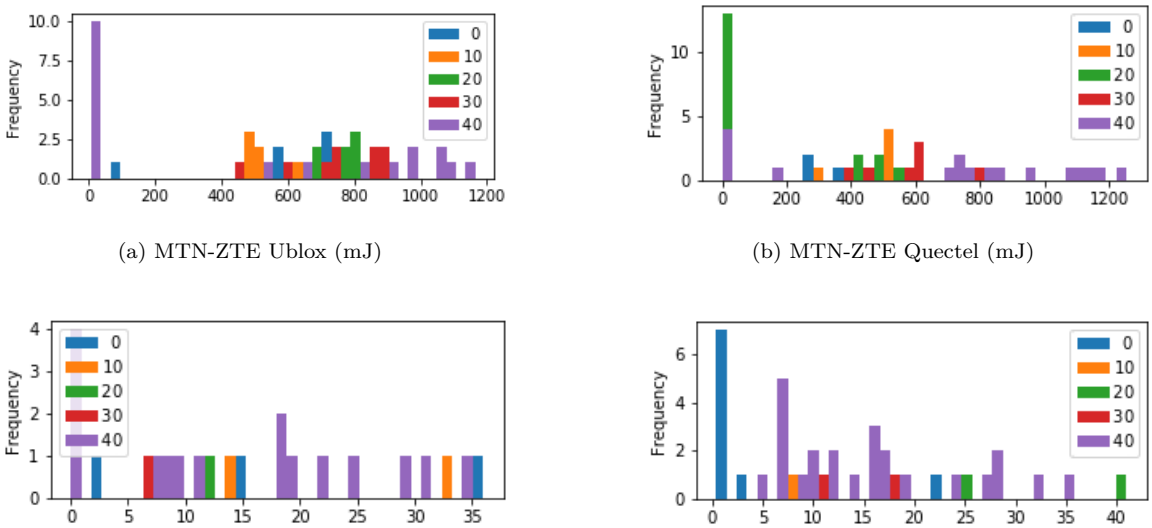
(a) MTN-ZTE Ublox

(b) MTN-ZTE Quectel

(c) Vodacom-Nokia Ublox

(d) Vodacom-Nokia Quectel

Figure 1.28: eDRX histograms of the energy produced by listening for paging occassions with respect to RF attenuation in dBm as mentioned in §??. Martinez [3] measured values under 10mJ for both Ublox and Quectel, yet on the MTN-ZTE and Vodacom-Nokia networks it is up to 5 times more. Thus, it is apparent that network configuration is the cause of these differences.



(a) MTN-ZTE Ublox (mJ)

(b) MTN-ZTE Quectel (mJ)

(c) Vodacom-Nokia Ublox (J)

(d) Vodacom-Nokia Quectel (J)

Figure 1.29: PTAU histograms of the energy produced when periodically updating UE devices and networks with tracking area IDs as mentioned in §??. PTAU uses about 20-100 times more power than eDRX for MTE-ZTE and up to 4000 times more energy on Vodacom-Nokia. Frequent PTAU updates should definitely be taken into consideration.

1.4.1.1 Average Power Average power is either measured, or estimated by assuming the following conditions: voltage is 3.6 V, transmit current is 250 mA and receive current is set to 60 mA.

$$Average\ Power\ =\ P \cdot t\ =\ V \cdot I \cdot t\ = V \cdot (I_{TX} \cdot t_{TX}\ +\ I_{RX} \cdot t_{RX}) \tag{2}$$

1.4.2 Latency and Timing

Latency and timing is also one of the main metrics focused on in this study. This section outlines a few preliminary tests and the final design of field tests. Latency was measured externally using energy capture device (§1.3.1.3) and UE reported values. UE reported values are extracted from `AT+NUESTATS="RADIO"` and relevant variables include `TX Time` and `RX Time` which are expressed in milliseconds since boot. `TX Time` is the duration for which the modem has been transmitting RF signals. `RX Time` is the duration for which the modem's receiver has monitored the downlink channel for activity. Together these time values can be used to assess latency and estimate the power consumed by the module.

Initial network registration will show `RX Time` increasing as it scans for a BTS. Once found, `TX Time` will start to increase until successful registration.

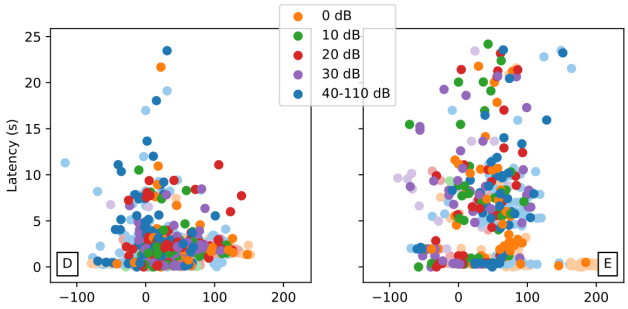


Figure 1.30: Latency per datagram as a function of the SINR (dB) as reported by the UE on the (D) MTN-ZTE and (E) Vodacom-Nokia network respectively. In Fig. 1.30, there is a poor distinction between attenuation zones as the SINR varies throughout the reported RSRP range. Grouping the data according to attenuation decade is important to see the effect of network conditions clearly.

1.5 Secondary Metrics

Secondary metrics, §??, are metrics of interest which have less of an impact than power or latency, namely signal strength, throughput and data overhead.

1.5.1 Signal Strength

Signal strength is affected by range and path loss. It also affects the ECL class that the network chooses depending on the signal strength that UE devices report to the network. Therefore, it is important to observe characteristics across a range of values.

		RSRP (dBm)	RSRQ (dB)	SINR (dB)
RF Conditions	Excellent	>=-80	>=-10	>=20
	Good	-80 to -90	-10 to -15	13 to 20
	Mid Cell	-90 to -100	-15 to -20	0 to 13
	Cell Edge	<=-100	<-20	<=0

Figure 1.31: LTE RSRQ and SINR RF Conditions. Tests were completed in good, mid cell and cell edge RF conditions as these would be the most common values in the field showing the typical network characteristics and variation.

Signal strength can be measured or reported from the UE device and the following metrics are all related: MCL, RSRP, RSSI, RSRQ, SNR, TX Power and ECLs.

1.5.1.1 MCL Maximum Coupling Link (MCL), as defined in §?? and by Eq. ??, is the greatest link between UE device and eNodeB. This can be calculated by using the minimum values of SINR obtained in the field capture datasets in §1.7.1, excluding outliers. Downlink SINR is calculated, because uplink SINR values as received on the BTS are unavailable.

P_{TX} is set to 43 dBm.

$Noise\ figure$ is defined as 5dB by 3GPP 45.820 7A [4].

With $Bandwidth$ in Eq. ?? set to 180 kHz as defined in §??, $Thermal\ Noise\ floor$ is equal to a value of -121.45 dB. Finally, P_{TX} is defined as

1.5.1.2 RSRP Reference Signal Receive Power (RSRP) or "Signal Power" is an average power measurement of the received power level in an LTE network, via a single reference signal in dBm.

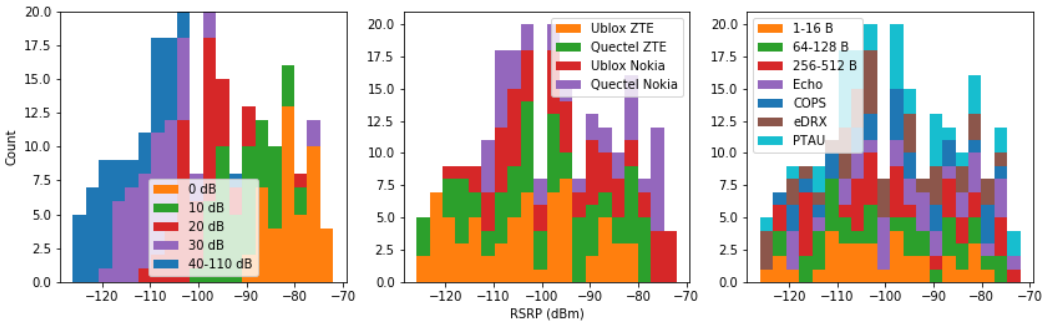


Figure 1.32: RSRP distribution using Ublox and Quectel UE on ZTE-MTN and Nokia-Vodacom infrastructure as well as attenuation and telemetry test set. RSRP and telemetry tests have a relatively even distribution, although RSRP still has about 20 dBm variability per attenuation decade. ZTE signals have higher MCL than Nokia.

1.5.1.3 RSSI Received Signal Strength Indicator (RSSI) or "Total Power", is the radio signal strength within the receive bandwidth. It usually combines the value of transmit power in the index. From this the signal to noise ratio (SINR) can be calculated.

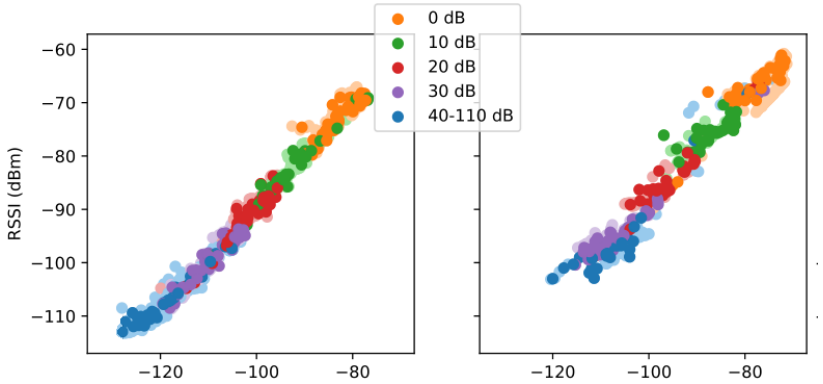


Figure 1.33: RSSI against RSRP for Ublox and Quectel devices (similar, thus combined) on MTN-ZTE (left) and Vodacom-Nokia (right). RSSI has a shorter 'range' than RSRP due to transmit power being included, and thus more information can be observed when comparing metrics against RSRP.

1.5.1.4 SINR Signal-to-interference-plus-noise ratio (SINR) is a measure of signal quality. It is proprietary to LTE vendors since it is not defined in 3GPP specs, yet still widely used to better quantify the relationship between RF conditions and throughput.

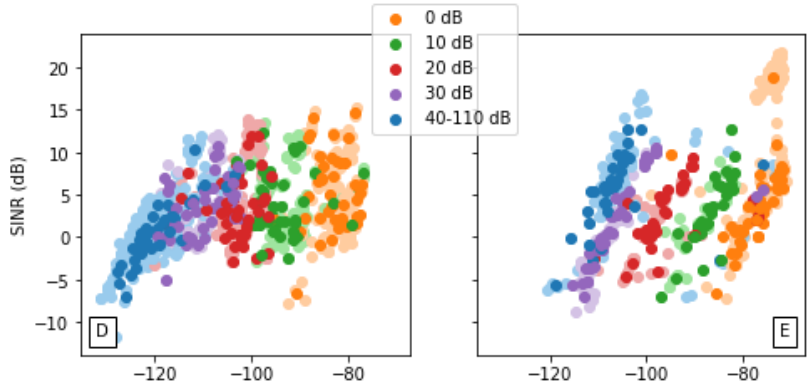


Figure 1.34: SINR against RSRP for Ublox and Quectel devices (relatively similar for now, thus combined) on MTN-ZTE (left) and Vodacom-Nokia (right). Each attenuation zone shows similar ranges of SINR, thus not as useful as RSRP when comparing other metrics against it. Ideally it should show a more linear relationship, and it is possible that these SINR readings are not accurate and contributing to variations in measured metrics.

Unfortunately, it has been implemented in various different ways which aren't trivially comparable. For example, using a synchronization signal (based off a PCI assigned by eNodeB) instead of a reference signal in estimation already results in a different SINR [5].

1.5.1.5 RSRQ Reference Signal Received Quality (RSRQ) is a measure of the quality of received power and is defined by Eq. 3.

$$RSRQ = N\ x\ \frac{RSRP}{RSSI} \tag{3}$$

Reported value	Measured quantity value	Unit
RSRQ_00	RSRQ < -19.5	dB
RSRQ_01	-19.5 ≤ RSRQ < -19	dB
RSRQ_02	-19 ≤ RSRQ < -18.5	dB
...	...	...
RSRQ_32	-4 ≤ RSRQ < -3.5	dB
RSRQ_33	-3.5 ≤ RSRQ < -3	dB
RSRQ_34	-3 ≤ RSRQ	dB

Figure 1.35: LTE RSRQ reporting range defined from -3...-19.5dB

In Eq. 3, N is the number of Physical Resource Blocks (PRBs) over which the RSSI is measured, typically equal to system bandwidth, and RSSI is defined as above.

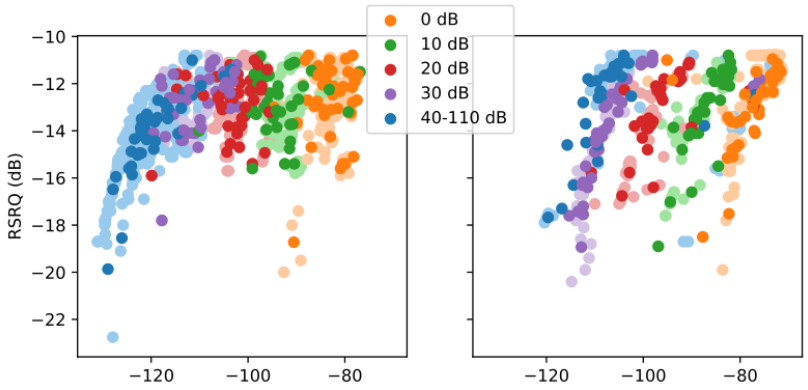


Figure 1.36: RSRQ against RSRP for Ublox and Quectel devices (similar, thus combined) on MTN-ZTE (left) and Vodacom-Nokia (right).

1.5.1.6 Transmit power Transmit power is the RF power output from the modem. It should be a lower number if within good coverage. Modems would typically consume ~230 mA for +23 dBm.

1.5.1.7 ECLs ECLs are equivalent to "PRACH coverage enhancement level" defined in 3GPP 36.321 [3] sub clause 5.1

As observed in §1.1.1.2, ECLs seem unaffected by RSRP. Due to repetition, it should have an impact on energy consumption and latency.

1.5.2 Throughput

Data can be extracted from UDP packet transmissions using latency and data size as in Eq. 4.

$$THP = \frac{Datagram\ Size}{Latency} \tag{4}$$

However, the UE reports RLC and MAC uplink and downlink separately, and therefore this will be used as these will be close to the theoretical maximum throughput of the UE device.

1.5.2.1 FOTA Upgrades GPRS and NB-IoT are able to offer FOTA upgrades to IoT devices, as Sigfox has limited bandwidth. This feature is supported by LoRaWAN, through the fragmentation of large payloads [22]. Since no correlation was found by Durand [6] with repect to downlink throughput and received signal power, that downlink latency is much less due to the higher bandwidth the eNodeB can provide and that FoTA updates are insignificantly infrequent, they will not be investigated in this study, although it can be in future.

1.5.3 Data Overhead

Variation in data overhead can be measured using TX, and RX byte counters. Usually the heavier the IoT protocol the more overhead, but we want to see the minimum overhead and therefore UDP is chosen out of TCP, CoAP and MQTT. UDP payloads are dynamic in size, but limited to 512 bytes. The UDP header is about 48-60 bytes in length, and so an application sending 100 bytes will actually send about 160 bytes. For devices in the extreme coverage class 2, this can be quite taxing on battery life and costly in terms of airtime. UDP sockets can remain open in PSM mode, so that the UE device may later resume the RRC connection with that context, thus saving considerable signaling overhead for the setup and transmission of infrequent small data packets.

1.6 Estimations

Two metrics are estimated in this study, including telemetry interval and battery longevity.

1.6.1 Telemetry Interval

The recommended telemetry interval or periodicity can be estimated for each telemetry type, including differently-sized UDP transmissions, Echo, COPS, eDRX and PTAU. In §**??**, telemetry interval is estimated in hours using the mean measured and UE reported values obtained in Appendix **??** and **??**.

1.6.2 Battery longevity

Similarily, the battery longevity can be estimated for each telemetry type, including differently-sized UDP transmissions, Echo, COPS, eDRX and PTAU. In §**??**, battery longevity is estimated in years using the mean measured and UE reported values obtained in Appendix **??** and **??**.

1.7 Field Test Captures

Ublox and Quectel data has been captured for:

- Nokia networks at Vodacom head office in Century City, Cape Town
- ZTE at the MTN Mobile Intelligence Lab, Stellenbosch inside an RF enclosure with the door slightly open (to ultimately reach the edge of received signal strength).
- Ericsson at MTN headquarters on 14th Avenue, Johannesburg
- Huawei on the Vodacom network in Quellerina, Randburg, Johannesburg

1.7.1 Dataset

The total dataset can be better represented visually in Figures 1.37 and 1.38.

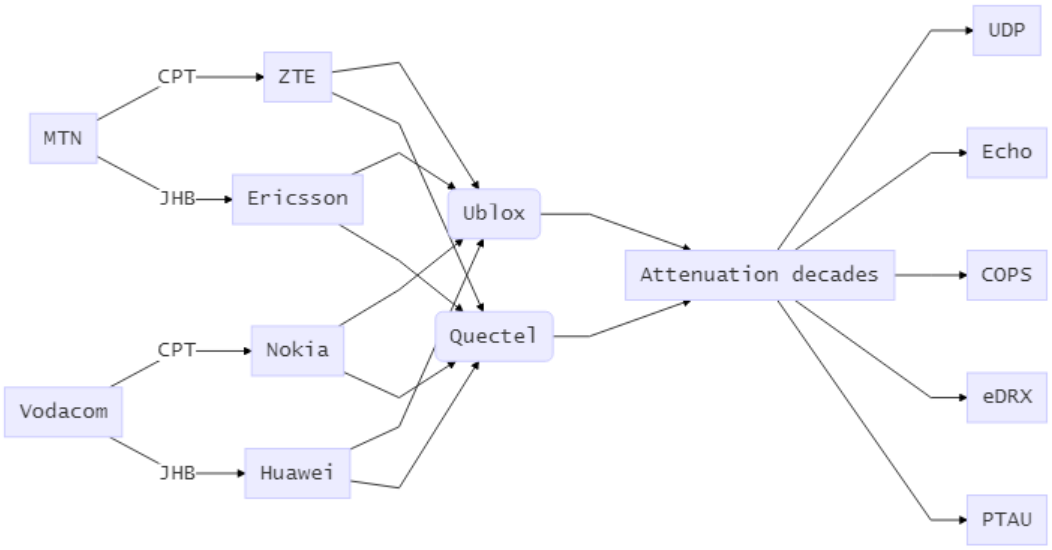


Figure 1.37: Visual representation of dataset. Five telemetry tests performed to at least five attenuation zone decades on two UE devices, four LTE vendors and two MNOs in Cape Town and Johannesburg.

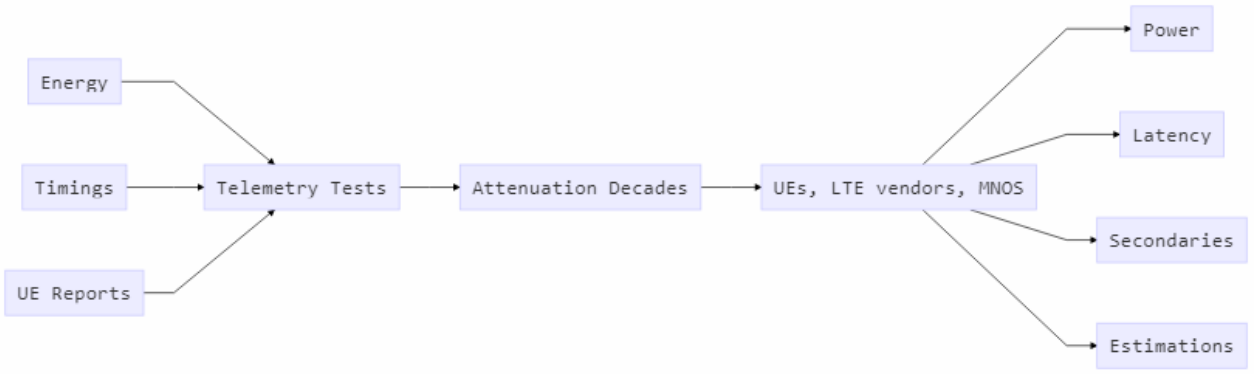


Figure 1.38: Each telemtry test takes in readings from the external energy capture device and UE reported values and through attenuation zone decades, UEs, LTE vendors and MNOs we extract power efficiency, latency, secondary metrics and estimations.

Every UE device and MNO pair (8 total) has 9 telemetry tests (of which 5 are differently-sized UDP datagrams) and each has its own attenuation zone decade (at least 5). The Cape Town dataset alone contains 424 files with 1811 trace entries, 40 possible submetrics (`AT+NUESTATS`, energy and timings combined in CSV files) and 79921 values. With the Johannesburg dataset included, there are 1390 CSV files in total.

The dataset is also heavily skewed towards lower latency entries in terms of a higher count. Multiple entries per test were captured, with the intent of increasing reliability, at a target of roughly 10 entries. Especially when entries takes a couple of seconds, but when an entry (unexpectedly) took up to 300 seconds it had a much lower chance of having 10 entries captured. Also considering that future dataset capture may be repeated in different locations, one does not necessarily want to spend more than a day on-site.

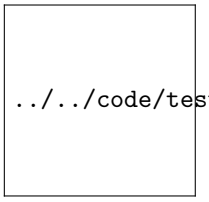
To solve for the skewness, each test can be normalized by taking a single mean of each of the associated trace entries and files. With a dataset of 140/1811 traces, it would make a minimum of 5600/79921 possible values.

Unfortunately this created the problem of lost features when multiple means are concerned, such as in ECL reports. To solve this, k-means clustering is applied as in §1.7.2.2.

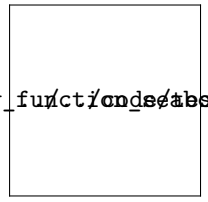
1.7.2 Post-processing

A method of post-processing the data for analysis is required for the dataset obtained in §1.7.1. Probability estimation was attempted, yet not considered as making histograms and kernel density estimations are two layers of manipulation on raw data before comparison. Instead, k-means clustering was performed on the dataset to lower low-latency skewedness and any other concentrated features to achieve a more normalized result with unique features for comparison. Finally, a plotting and extraction framework was developed in Jupyter with custom libraries developed in Microsoft Visual Studio Code (`vscode`). Plots are placed throughout this thesis, with "9-plots" that visualize the dataset in Appendix **??**, and measured or UE reported metrics and estimations in Appendix **??** and **??**, respectively.

1.7.2.1 Probability estimation Due to the large dataset and requiring a reasonable means of analysis, probability estimation is considered.



(a) The probability of the discrete data can be estimated in a continuous probability density function (PDF) with the kernel density estimation.



(b) Various types of kernel density estimation (KDE) all attempt to predict latency outcome with varying accuracy.

Figure 1.39: Probability estimation on latency histogram sample

Good practice would be viewing the data as is and not trying to analyze it from what is essentially a modified new perspective. Considering how various methods and techniques are used to estimate probability, there will always be a degree of inaccuracy, and in future work machine learning models can be considered to predict UE device behavior. In this thesis, we offer a two-pronged approach where we would like to view the data as close as possible to its raw form, yet as simple as possible for comparison.

1.7.2.2 K-Means Clustering Instead of finding a single mean for all the entries and associated files, at least three means are specified (K=3) to take into account the outliers that some tests produce or more for isolated regions (K=4+).

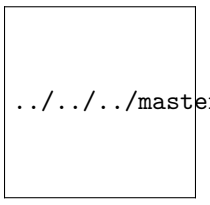


Figure 1.40: Trace entries per test (CSV file). In this example, the absolute maximum of 1811 traces has been filtered by removing duplicates and applying thresholds. K-means clustering achieves the desired effect of reducing dimensionality and skewness induced from low latency sampling on the dataset for different tests, yet keeps most of the features of the thresholded max. This allows for easier visualization and unbiased feature comparison.

1.7.2.3 Plot Visualization: Jupyter Jupyter is a python framework which is used for post-processing, and the following code snippet shows an example of the `9-plot` format used in the results (Chapter **??**):

```
import jupyterlib as j
import plotter as p
import plotter4 as p4

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

import importlib

def plot(*args, **kwargs):
    importlib.reload(j)
    importlib.reload(p)
    importlib.reload(p4)
    testl = ['1-16 B', '64-128 B', '256-512 B', 'Echo', 'COPS', 'eDRX', 'PTAU']
    K = kwargs.pop('K') if 'K' in kwargs else None
    p4.plot(db(), *args, **kwargs, K=K if K else 3, folder='plotterk', joburg=True,
        testl=testl)

...
...

plot('SNR', 'txTime', 'SNR (dB)', 'Latency (s)', scale=[10,1000], K=5)
plot('Signal power', 'energy', 'RSRP (dBm)', 'Power (uWh)', [10,3.6], K=6, log=True)
```

As there are numerous Jupyter files, most code resides in custom libraries which can be imported into each file to maintain consistency in case of duplication errors, and this can also be found on https://github.com/daniel-leonard-robinson/masters/tree/master/code/tests. During development on the custom libraries, Jupyter requires `importlib` to `reload` each library when a master function such as `plot(*args, **kwargs)` is called.

Table 8: Custom libraries imported by Jupyter and a description of their purpose

Library	Purpose
jupyterlib	processing CSV files, directories, tests, thresholds
plotter	gathering data into single dictionary database for plotting
plotter4	plotting data in 9-plot format, K-means clustering

Other plots were more specialized and code was kept within the Jupyter file it was developed in. Although the goal of plots in general is to investigate observations and comparisons, it is intended for visualisation of the results in Chapter **??**. A short description is occasionally written about the plots of the datasets captured in Cape Town, with the second set of `9-plots` showing the results in Johannesburg left up to the interpretation of the reader. If plots are omitted, they show quite similar information, and duplication is not necessary.

In the top left corner of the `9-plots` exists a box which shows the number of k-means cluster filtered data points out of the total number of possible filtered points, as well the 'K' value in the filtering process. Each diagram is marked from 'A' to 'I', with 'F' showing boxplots of Ublox and Quectel distributions on the relevant MTN and Vodacom network, depending on if the data was captured in Cape Town or Johannesburg.

- ABC shows UE device and MNO comparisons
- DE shows plots with data points by attenuation decade
- GH shows plots with data points by telemetry test set
- ADG and BEH shows two different types of LTE vendor, respectively.

[1] R. &. Schwarz, "Narrowband Internet of Things Whitepaper," p. 42, 2016.

[2] B. Martinez, S. Member, F. Adelantado, A. Bartoli, and X. Vilajosana, "Exploring the Performance Boundaries of NB-IoT."

[3] Yocto, "Yocto Project – It's not an embedded Linux distribution – it creates a custom one for you.".

[4] "CAT-M1 vs NB-IoT – examining the real differences - IoT Now - How to run an IoT enabled business.".

[5] "LTE RSRQ to SINR - CableFree.".

[6] T. Durand, L. Visagie, and M. Booysen, "Evaluation of next-generation low-power communication technology in IoT-applications," *IET Communications*, pp. 1–8, 2019.