ALMA MATER STUDIORUM UNIVERSITY DI BOLOGNA

Facoltà di ingegneria
Dipartimento di ingegneria dell'energia elettrica e dell'informazione
Corso di Laurea in Telecommunications Engineering

Thesis in Mobile Radio Networks

Performance analysis and market evaluation of a low cost spectrum analyzer for IOT testing platform.

Prepared By:
Nour El Huda Al Daher

Submitted to:
Prof. Roberto Verdone

Bologna, Italy
March, 2021

Abbreviations

LoRa	Long Range
LPWAN	Low power and wide range network
M2M	Machine-to-Machine
IOT	Internet of things
SMA	Simple Spectrum Analyzer
SSA	Sweep Spectrum Analysis
BBB	BeagleBone Black element14
LTDZ	LTDZ 35-4400M SA
SA	Spectrum Analyzer
SSM	Spread Spectrum Modulation
CSS	Chirp Spread Spectrum
P2P	Point-to-Point
OEM	Original Equipment Manufacturer
RBW	Resolution Bandwidth
Fsh6	Handheld SA R&S ® FSH6
UNIBO	University of Bologna
VFO	Variable Frequency Oscillator
PLL	phase-locked loop
PIC	peripheral interface controller
ISM	Industrial, scientific and medical applications
EBI	Embit Binary Interface
GPIO	General Purpose Input/Output
UART	Universal Asynchronous Receiver-Transmitter

Premise/Abstract

This thesis is the final module of master’s program in telecommunications engineering at university of bologna.

The demand for connected devices, according to the Internet of Things (IoT) paradigm, is expected to grow considerably in the immediate and near future. Currently, there are many standards that are applied based on the need to connect every object which leads to have a massive connectivity. Among these standards, Low-Power Wide Area Networks (LPWANs) are continuously gaining importance and applied to many fields thanks to their ability to provide long-range coverage to devices, exploiting license-free frequency bands. The focus of this thesis is on one of the most prominent LPWAN technologies: LoRa™ and how to measure its performance and also about the main importance of studying and measuring the radio frequency spectrum for this standard and others too. Spectrum Analyzer “SA” is one of the most powerful equipment to study, analyze, and optimize the radio communication systems. These days due to spreading of these systems with the consequent growth of traffic demand it is basic to study and measure the radio spectrum.

Since always the time matter so we must find a way to increase the testing speed to analyze the communication signals. We found a fast and cheap way to read, analyze, and test the signal in both dimensions frequency and power.

This work has been done in collaboration with Embit SRL Company, where I did my thesis and collaborating with my university UNIBO, they usually assign us a project in order to let students achieve the experience they need to be ready for the real world.

The aim of my project was to increase and optimize the testing the signal of the EBi-LoRa. This is achieved by putting SA in parallel way to let the testing be simultaneously instead of one by one. The time consumed of testing will be decreased by number of SA times less than the current way of testing. Beside it can do all the tests process by itself without any monitor to stay in the bench, so with this idea we decrease the time and the cost needed to study the LoRa signals.

Acknowledgment

I would like to thank everyone who helped and supported me during my thesis work. Firstly, I would like to express my sincere gratitude to Prof. Roberto Verdone, my supervisor, for his advices, instruction and encouragements on my work, and direction for the thesis during my pursuit of the master's thesis. I have learnt a lot from him while joining his courses that was useful for my thesis as internship, in addition for writing and presenting myself as well.

I also would like to thank Dr. Bonizzi for his assessment in his own company, titled by EMBIT which gave me a lot of experience to face real life work, and the ability to work within a team inside his company and from them is Mr. Jonjan Hoxha and Mr. Augusto Sala to achieve the result goal.

Finally, I wish to extend my sincere thanks to my parents for their compassion and unlimited support.

"The Only Way To Do Great Work Is To Love What You Do. If You Haven't Found It Yet, Keep Looking. Don't Settle."

Steve jobs
1955 – 2015

Table of Contents

1. Introduction

Today there are multiple ways of connecting to multiple networks. The main parameters which give a full indicator about the performance of any technology or wireless system are such as the distance; connection time and amount of data transferred all necessitate different solutions. Each technology provides its own strengths, the weakness points too, and showing that is perfectly suited to various applications and situations.

In this work we will talk about the LoRa system and its overview besides the used ways to test the performance of a LoRa system. LoRa is known as a physical layer technology that works in unlicensed sub-GHz ISM band and is based on chirp spread spectrum (CSS) technique. LoRa can support various ISM frequencies, such as 868 MHz (for Europe), 915 MHz (for North America), and 433 MHz (for Asia).

This work starts by understanding the theory of how a LoRa system operates and how to use the right equipment for doing the measurements. The project has been carried out at the EMBIT Company [*http://www.embit.eu/]* in a cooperation project under the supervision of Dr. Fabio Bonizzi from EMBIT side and the Department of Electrical and Information Engineering Degree Program “Telecommunications engineering” at UNIBO, specifically in the RN (Radio Network) group headed by Professor Verdone.

The purpose of this project is to design, develop and control an electromechanical test bench which will be used for radio spectrum measurements. At the beginning we started to use this LTDZ 35-4400 MHz Spectrum Analyzer to perform the needed measurements, beside using the BeagleBone black element14 “BBB” with has a pre-installed debian Linux to create a user interface platform to program, and control the device during testing system performance.

After that, I moved to create my new user interface platform for the Spectrum analyzer to get more control over it and increase its efficiency to support our testing. I programmed it based on my experience on Python and by configuring all the parameters we need to build our own smart interface. Based on the program and code I created, I was able to control the SA in a different ways either remotely or automatically.

Finally, based on the measured data, I can easily saving and analyzing it to check how it works and improve the performance of the measured system.

1.1. Goals of The Thesis

For me, as a general goal of this work was to use the in house available equipments such as the normal SA to get accurate results and efficient test bench setup in terms of results take into account time and cost.

As initial step for this project, I started to show how I can specify the SA parameters like start frequency, stop frequency, span, samples number and step number in order to be able to build and create the application for the testing bench. The main of the testing bench is to read LoRa signals in parallel and without any manual interaction for monitor the process.

During this project, I got familiar with new things such as Linux environment, Python, C++ and J-Link beside the general knowledge about the LoRa and signal spectrum measurements, in addition to the way of connection networks such as LoRa.

The work has been done step by step, firstly to understand how to use Linux environment, then I started to program the code over the BBB using python, in order to use different connection ways to communicate with our devices such as the serial interface, GPIO, and UART.

As a general idea about the work done in the thesis, we started to send a list of command bytes to the LTDZ and then read the data bytes of EMB-LoRa signal in the input connector of LTDZ. Then we checked the output measured power at the specified frequency if it's matched with our expectations for the LoRa Signal. After that, going through saving the data, analyzing it, and give an overall feedback in order to continue the required process we will talk about them later in the thesis.

Based on the achieved results from the previous single LoRa test bench measurements, now we are going to integrate this process or setup to a multi LoRa devices in order to read and check the signals in parallel and simultaneously. So we increase the speed and optimizing the testing process.

This integration should be done in a bench that was already built, where we place a matrix of LoRa elements. The number of the LTDZ must be equal to the number of the LoRa in a row, so the test is done every row by row at a time.

By using the EBI library to communicate with the devices, and the python software for the SA, we can be able to get our measured signals. Based on these measurements, we can check if there is any device is not working properly as we expected or damaged for any reason, in this case we will get an alert such as a red led indicator pointed to the damaged device. In this way we can point directly where the damaged devices are. In general the communication has been done using UART, while the action is done using the GPIO pins over the BBB.

While checking the LoRa signals, we need to assign a unique ID for every tested LoRa device in addition to other information which you can see in details in Appendix [c]. To get such as information we needed to communicate with the server, so we can get all the information of the running test and also the needed specifications for each LoRa device which in this case will be known by its ID. The importance of this information as it will be delivered a kind of guidelines or user manual for the client/user together with the produced device.

1.2. Methodologies

As it was written before, that the work has been done inside Embit SRL Company where I got the access to use their facilities and equipment. The main part of the work depends on the SA, so we had to search for an efficient one that satisfied our needs. At the company, we used the Fsh6 SA to test the signal spectrum of their owned devices. This instrument used in a manual interaction on a test bench. The main disadvantage in this case that this way of testing will need an employee to monitor its process, so it costs a fixed employee/Lab technician to follow the process in presence in addition that testing is done device by device. In the recent years, many new and developed SA appears in the market with a very cheap price, controllable and having a small size. This kind of instrument should fit our goals.

The task of selecting a new efficient SA was delivered to me, so I did a tradeoff between many SA based on the price, size, frequency range, supported software and other parameters, and I came up with a shortlist of SA with their specifications and prices, then I submitted the paper to my colleagues to check and confirm which one will be bought. They decided to buy a new one from bankbook website named as LTDZ 35-440M [1]. This chosen LTDZ SA costs 27 euro and runs with a ready built-in application named winNWT4 with its last version 9.11 and this application is compatible with windows.

I began to use the BBB Computer to program this device, or in general to build the application needed to be used in our project as this device is a major part of it. The BBB Operating System is Linux Debian as we mentioned before. I started learning from scratch to find a way to run this SA in Linux.

It was difficult at the beginning to do this kind of programming for the SA as there were not many supported documents or references that could help, So I started to focus on this issue more and more until I found a way to reform it as we aim. I used another approach which they were being used for other instruments such as NWT4000 original one besides I follow Mr. Victor instructions that had the experiences in these kinds of devices [2].

In conclusion, it has not been used existing platform, but rather I needed to develop a new one by myself so I went deeply in the LTDZ structure to begin to understand how I can communicate with it, how to read the its data, and how to analyze the arrived information bytes in order to help myself achieving the goal to complete the building of the project. All the in detailed process and steps has been done will be present in the next chapters.

2. State of the Art

2.1. Telecommunications systems

Nowadays, telecommunication systems are spreading very fast everywhere, and becoming a fundamental part for every day in our lives, so we utilize the frequency to make more devices fit into our new daily applications, and there are many ways to exploit the way of using the frequency.

This popularization and large demand for electronic devices had led to an enormous demand of data, which had increases exponentially since the 90s, and it will increase more and more. Only in 2013, the global mobile traffic demand increased of 83% from 2012 [3]. The growth will be much larger in the next years due to the extension of telecommunications devices all over the world.

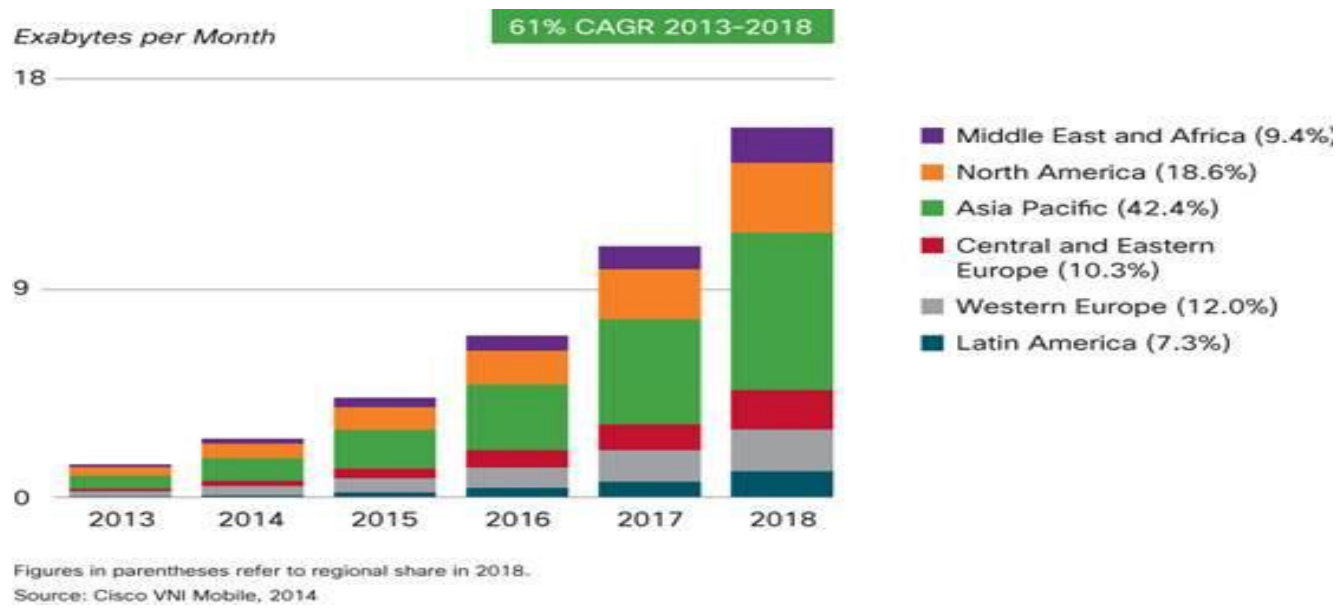


Fig 1. Global mobile traffic demand growth [1]

It is known that LoRa (Long Range) is a wireless technology and a part from the full image of the telecommunication systems that has been developed to enable low data rate communications to be made over long distances by sensors and actuators for different applications such as M2M and Internet of Things (IoT). Long range (LoRa) is one of the most successful low-power wide-area-network (LPWAN) technologies operating in unlicensed bands.

LoRa is one of the most important aspect in our days, LoRa is a LPWAN protocol developed by Semtech. It is based on SSM techniques derived from CSS technology. It was developed by Cycleo of Grenoble, France and acquired by Semtech, the founding member of the LoRa Alliance. It is the wireless technology mainly targeted for M2M and IOT networks. This technology will enable public or multi-users networks to connect many applications running in the one network.

2.2 Radio spectrum

Distinctive parts of the radio spectrum are distributed for diverse radio transmission innovations and applications. In a few cases, parts of the radio range are sold or licensed to administrators of private radio transmission administrations (for illustration, cellular phone operator, Wi-Fi or broadcast TV stations). The radio spectrum is constrained, it is exceptionally imperative to utilize an effective way to abuse the radio frequencies since in one hand, there's an expanding request of activity due to popularization of modern gadgets, and for the other hand, there's a radio range that's constrained and most of them possessed.

Conventional government arrangements donate licenses innovation and benefit particular, which has been seen that's a proficient strategy to utilize radio spectrum in a legitimately way, but it cannot bear the data request development. Hence, the government approaches change to progress this situation have three primary points:

- Spectrum reallocation: the reallocation of bandwidth from government and other long-standing users to new services, such as mobile communications, broadband Internet access and video distribution.
- Spectrum leases: the relaxation of the technical and commercial limitations on existing spectrum licenses by, for example, permitting existing licensees to use their spectrum for new or hybrid (for example, satellite and terrestrial) services and granting most mobile radio licensees the right to lease their spectrum to third parties.
- Spectrum sharing: the allocation of an amount of spectrum that could be used for unlicensed or shared services.

An LPWAN technology, the key performance and quality criteria of LoRa are RF performance under different RF conditions and configurations as well as power consumption. Testing some of the essential RF parameters is recommended before applying for the required LoRa certification and regulatory conformance certification in order to save time and money and to provide the best customer experience.

From the importance of the measurements of LoRa performance and its RF parameters, comes the importance of the basic and essential spectrum analyzer “SA”. Besides that as RF signals have become ubiquitous in the modern world, so too problems with interference between the devices that generate them. To overcome evolving challenges, it is crucial for today’s engineers and scientists to be able to reliably detect and characterize RF signals that change over time, something not easily done with traditional measurement tools.

A spectrum analyzer measures the magnitude of an input signal versus frequency within the full frequency range of the instrument. The primary use is to measure the power of the spectrum of known and unknown signals. Given the challenge of characterizing the behavior of today’s RF devices, it is necessary to understand how frequency, amplitude, and modulation parameters behave over short and long intervals of time.

Modern SAs can acquire a passband, or span, anywhere within the input frequency range of the analyzer. At the heart of this capability is an RF down converter followed by a wideband intermediate frequency (IF) section. An Analog to Digital Converter “ADC” digitizes the IF signal and the system carries out all further steps digitally. DSP algorithms perform all signal conditioning and analysis functions.

2.3 Overview of EMBIT Company

This work /Thesis were in cooperation with Embit, so in this part I will go to talk briefly about the company and its core devices. Embit was born in 2004 as an academic spin-off of the University of Modena and Reggio Emilia, in order to act as a bridge between academic research and local industries.

Embit offers support to the development of innovative ideas and solutions, offering designs, capacity and creativity, helping customers in the implementation of custom solutions with high technological content. In particular Embit assists customers throughout the design of embedded systems and the development of ad-hoc wireless solutions, from the feasibility study to the early stages of mass production. Supported by many years of RF expertise and know-how. Embit finds

and suggests the best solution that shortens the customer’s time-to-market and transforms his ideas in successful products.
Embit designs and produces high performance and cost-effective radio modules, embedded systems, and electronic devices to be used in several wireless scenarios. The company also provides its customer with application-specific ad-hoc designs.
The skill and expertise of Embit are spread over consumer, industrial, logistic and healthcare markets, with particular accent on radio-frequency technologies that operate on ISM (Industrial, Scientifical and Medical) bands (868MHz/915MHz and 2.4GHz) and metering bands (169MHz).
So Embit uses the EU regulation, so for the LoRa device the frequency that is legal to transmit on it is 868.1 MHz
Much obliged to these changes radio spectrum market will be more adaptable to adjust to the client and industry’s necessities and to permit the changes required to fulfill the existing and future request.

2.3.1 Embit Devices

Connectivity is at the heart of IoT. Devices talking to each other, the cloud and beyond are creating a new digital universe of interconnected technology. Embit SRL is one the main Italian partners to support the world with well made IOT devices and participation in new breakthrough technologies.

Embit offers a complete portfolio of wireless modules, system-in-package components and airtime options. The new LoRa and LoRaWAN technology from Embit is extremely flexible, scalable and secure and is quickly becoming the global standard for low energy radio communications. Widely supported by the industry, LoRa and LoRaWAN is truly empowering the future of the Internet of Things.

The EMB-LR1272 modules are 868/915 MHz wireless modules that combine high performance contained within a small form factor, with a low-cost price point. The system integrator is a simple and easy way to add LoRa/LoRaWAN long-range connectivity into existing products. Embit’s modules are compliant with the latest LoRaWAN specifications, allowing customers to develop firmware application and manage a compatible LoRa eco-system.

2.3.2 EMB-LR1276S

EMB-LR1276S as shown in Fig.2 is the new ultra-low power LoRaWAN based wireless OEM EMBIT module. The EMB-LR1276S have size 11.5x11.5 mm form factor, so it’s the smallest LoRaWAN available now in the market. It embeds a full LoRaWAN class A and C protocols and it is enabled to any LoRaWAN compliant network infrastructure. EMB-LR1276S is based on the new microchip SAMR34 Sip, a 32-bit ARM cortex Mo+.

Fig 2. EMB-LR1276S device

EMB-LR1276S can communicate with other devices through a wide range of serial interface: UART, 12C and SPI ports, up to 17 digital and 8 analog I/O ports useful for the managements of external devices and interfaces. Thanks to reduce power assumption of microchip SAMR34, the EMB-LR1276S is a perfect solution to implement long life battery powered devices.

Figure 2 illustrates the connections between the EMB-LR1276S and the peripherals (LEDs, push buttons, antenna, debugger, USB 2.0 Micro-B connector, and external battery) placed on the board as indicated in Figure 3.

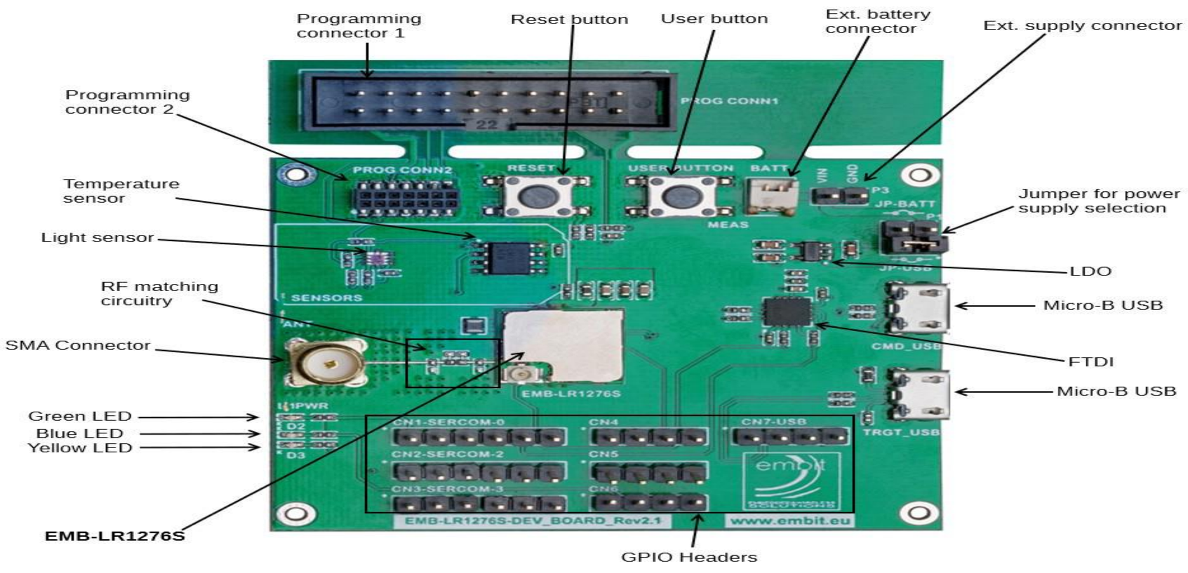


Fig 3. EMB-LR1276S Device Board Rev 2.1 layout configuration.

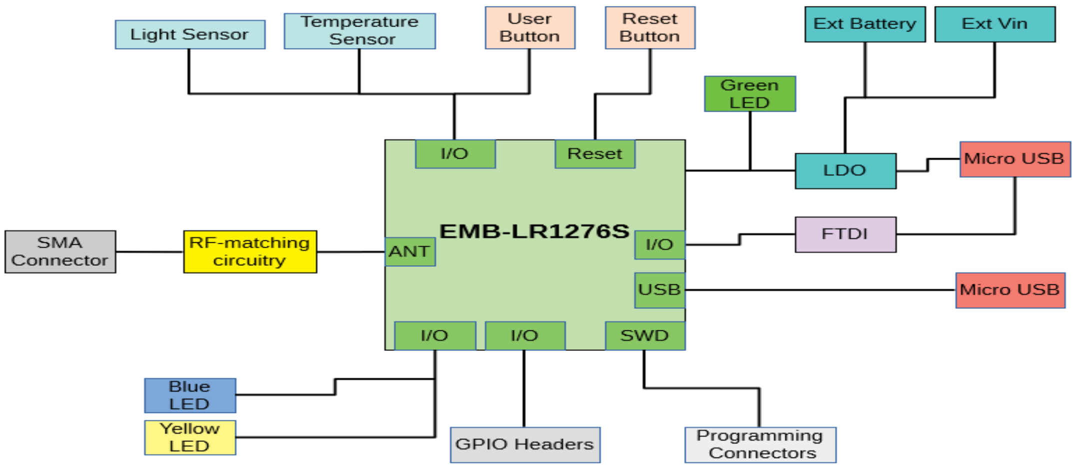


Fig 4. EMB-LR1276S Device Board Rev 2.1 block diagram

EMB-LR1276S's firmware can be easily developed starting from the Embit SDK, allowing establishing LoRaWAN compliant end-nodes or custom p2p network.

With LoRaWAN disruptive long range wireless technology, the EMB-LR1276S delivers dramatic performance improvements for the AMI, IOT, M2M mobile and consumer markets. It offers unique network scalability and 10x range extension over existing sub -1 GHz solution.

3. Experimental Methodology and Project Development

In this chapter I'm going to talk about the SA the original one and the one we bought "the clone of the original". The main problem was the protocol interface in serial port and how to compensate the errors in order to be used in our project. Then I'm going to talk about the system model used, in general (structure of the overall project).

3.1 NWT4000 SA

We already said that there are new and very cheap versions of SAs that could be used to analyze signals, the first one were produced in Germany titled by NWT4000. NWT series digital virtual frequency sweepers a high intelligent RF frequency analyzer. The equipment is composed of MCU, RF detection, RF generator(s), and the circuit board contains the linear detection, log detection, completely analyzing systems for amplitude versus frequency characteristics. The amplitude frequency characteristics can be fast and accurate measurement of RF devices in the 35MHz -4400MHz range.

NWT series digital virtual frequency sweeper can be widely used within radio, television, communication and other fields. The measured object includes RF Devices like coaxial cables, amplifiers, combiners, amplifier modules, filters, attenuators, splitters, loads, antennas, power-dividers, and tuners.

The hardware structure of the NWT4000 makes it possible to a limited extent. To represent a spectrum the measuring input RFin goes to the direct receiver with subsequent low pass 300 kHz. This principle always creates a double signal in the display for each single signal, with an upper and lower sideband. But you can live with that. The level of the IF signal is with detected by an AD8307. The mixer in the input channel RFin "is very over-control-proof". The dynamics of the input channel is almost 80dB, even at very high frequencies.

For example, I have a frequency of 150MHz with a level of [-10dBm] fed into RFin. With the function "Create peak list the peak marker is shown as text in the upper sideband.

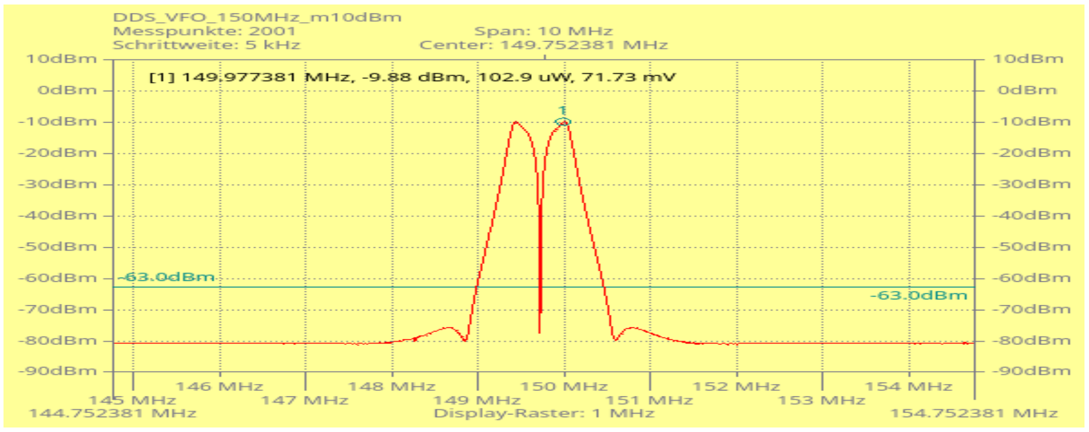


Fig 5. Signal at 150MHz with a level of - 10dBm

For all the devices that is clones for the NTW4000 for instance SMA/NWT/D6/LTDZ SA's have the same issue with the "notch" that appears on transponders/signals with less than 500 kHz bandwidth, as you can see in the figure above, and now I'm going to explain this problem in details.

To explain the concept, here in Fig.6a,b is the same signal captured with different RBW values

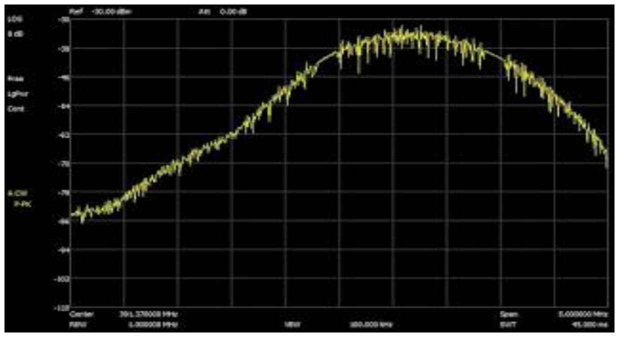


Fig 6a. x signal of RBW = 1 MHz

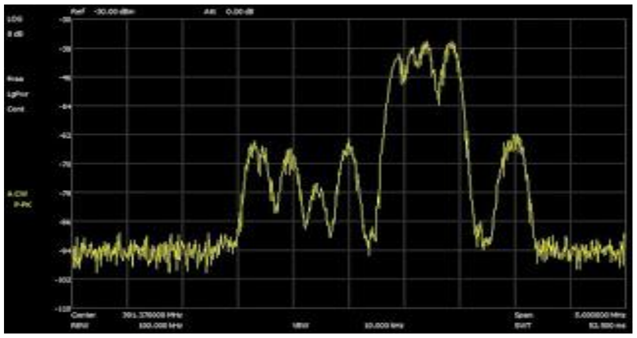


Fig 6b. x signal of RBW = 100 kHz

The SSA3021X is a real SSA and it permits to set the RBW as required. Imagine the RBW to be a window: the sample you get is the average of what you see through the window. The smaller the windows, the higher the determination, but the sweep will be longer.

In case of the SMA/NWT devices, we don't have a user configurable RBW. You can only determine a start frequency, a frequency step and the amount of samples you want to do. The device will then sweep the frequencies, but at a constant RBW. In this link you can see how the detector sweeps the RF spectrum [4].

The detector will average the power level inside its window. Tragically, the way the device and its IC's were designed, it includes a "blind spot" within the center frequency of the detector. This is the reason of the notches on signals with a narrow bandwidth. When the detector is perfectly in the middle of the signal the blind spot will mask a significant part of the signal, which causes the sample to really have a smaller power level than the neighboring samples as shown in Fig.7.

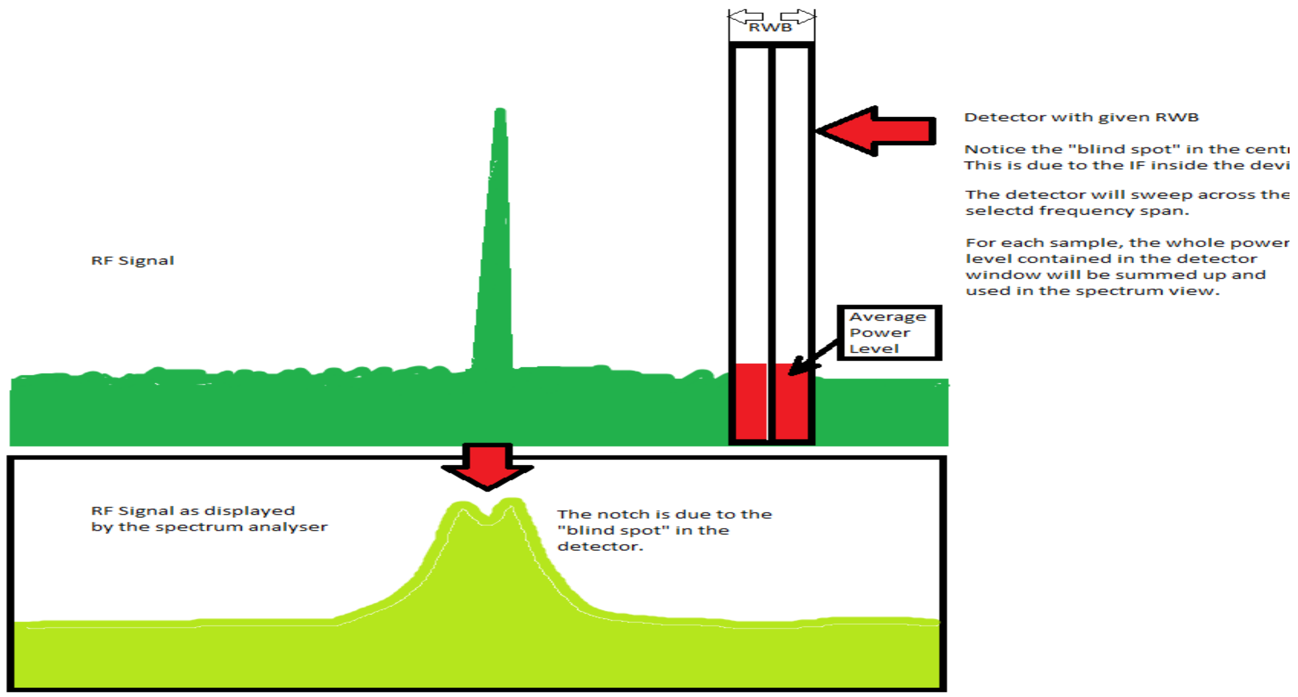

RWB
Detector with given RWB
Notice the "blind spot" in the cent
This is due to the IF inside the devi
The detector will sweep across the
selectd frequency span.
For each sample, the whole power
level contained in the detector
window will be summed up and
used in the spectrum view.
RF Signal
Average
Power
Level
RF Signal as displayed
by the spectrum analyser
The notch is due to the
"blind spot" in the
detector.

Fig 7. Sweep of the SA NWT4000

Note that the blind spot isn’t in the center of the frequency span that you see on the screen, but in the middle of the detector that is used to measure the power level in each sample.

Figure 8 shows that , When the narrow band signal is smaller than the RBW of the SMA/NWT device, the samples that totally contain the signal will measure a littler power level, than the samples where the signal isn't totally contained interior the detector window.

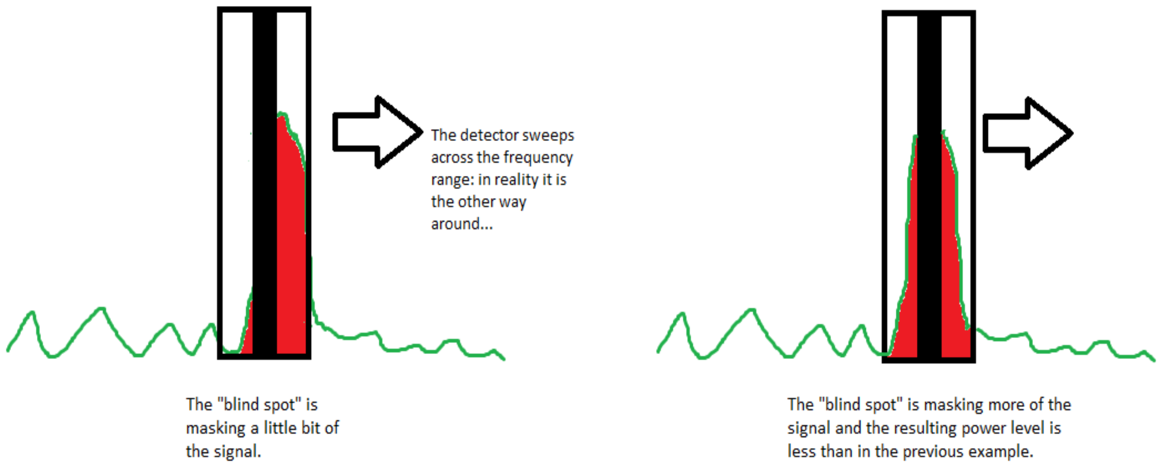

The detector sweeps across the frequency range: in reality it is the other way around...
The "blind spot" is masking a little bit of the signal.
The "blind spot" is masking more of the signal and the resulting power level is less than in the previous example.

Fig 8. Sweep of the SA NWT400

Here is some extra information on spectrum analysis:

The device will digitize a given number of samples in the Time Domain. This is much like a digital oscilloscope: you digitize for example 2000 points in one second. Imagine the signal is a sine at 100MHz (in other words, the time of the signal is 1/100000000 seconds). Capturing 2000 points will allow you to calculate the sine function matching the samples through FFT math. You can then render the function in Frequency Domain, as you know the amplitude (signal level) and frequency.

Here you take the input signal and convert it down to your BAND PASS FILTER. In a good device, the aperture of the filter is configurable (i.e. 10 kHz, 100 kHz, 1 MHz). The filter always operates at a fixed frequency span, so instead of sweeping the filter across the frequency span, the frequency is down converted accordingly (in the animation of my blog entry, the detector is fixed and it is the frequency ranges that moves). Next comes the detector: it measures the signal power of the signal leaving the band pass filter.

As final words The SMA/NWT are good devices, considering their prices, no doubts about it, but we need to accept the limitations and try to find a solution for them.

3.2. LTDZ 35-440 MHZ

In ideal situation, the LTDZ must be worked as NWT4000, but for real it's not the same, there are a lot of differences between these two for example the RBW, the protocol in the serial interface and many other parameters.

Following the instructions of the original NWT4000 protocol interface we know how the device must work, or in more clear words what we must send and what we must receive from it [5]. From the NWT4000 the original SA we have the notch problem, so all the clones that appears had the same problem, and many other problems in advanced than the original one depends in the limitation.

My collogues decided about buying one of them to initiate the project, so we took the advantage of the very cheap price and the wide range frequency and theses specification are good for our needs. Finally the Company decided to buy the "LTDZ 35-440MHz".

Here I put a timeline for my next steps of my work since I had all the equipment I needed. The first thing was, I wanted to do is to get familiar with the SA LTDZ, by studying the main characteristics, main capabilities, and operation methods. all the specification of the SA you can see them in Appendix A.

This type of Signal Tracking Source based on RF Frequency Domain Module and its Analysis Tool comes with Aluminum black Shell as you can see in Fig.9.

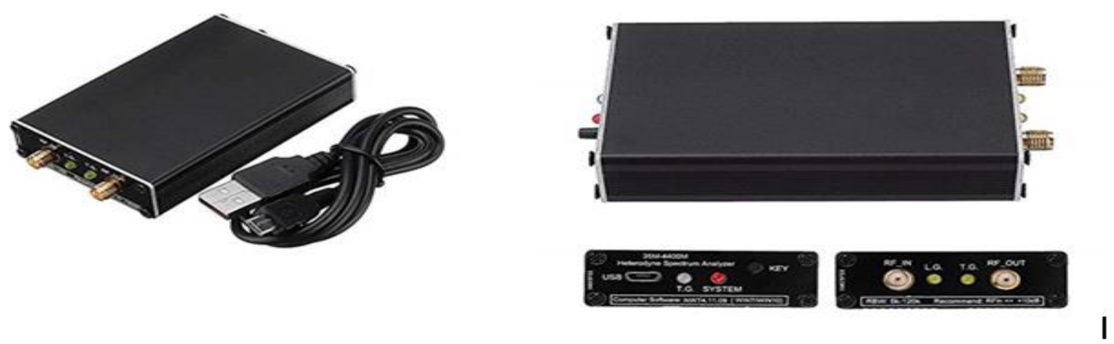


Fig. 9. SA LTDZ 35-4400M with metallic case

Figure 10 describes the board structure that is inside the metallic case and it shows the major components in the board. For the diagram block in details you can check Appendix B to have a good understanding of how the board works.

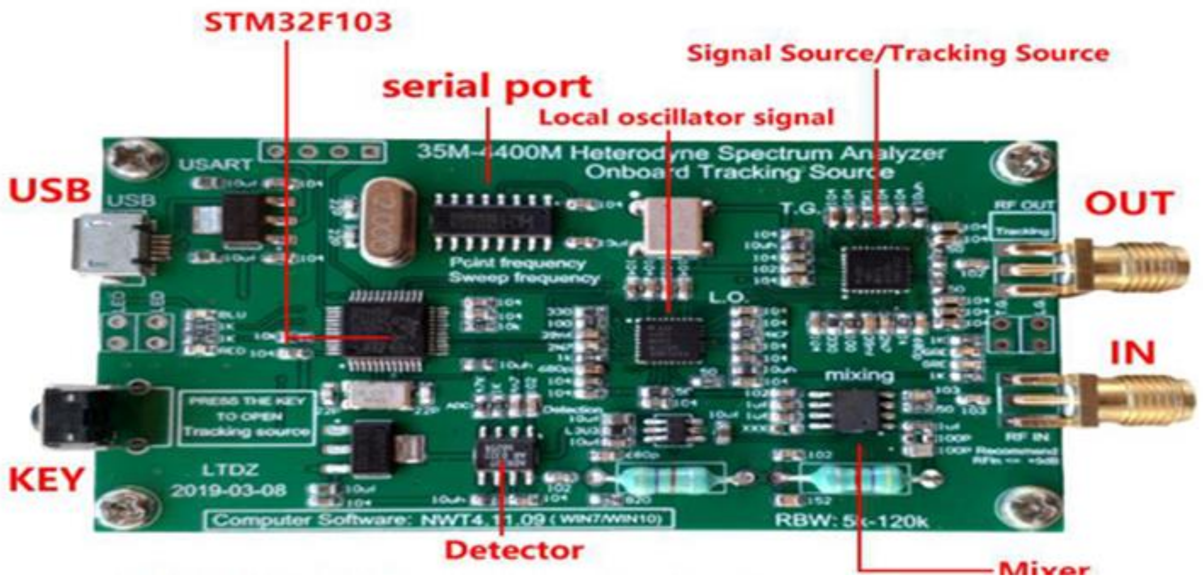


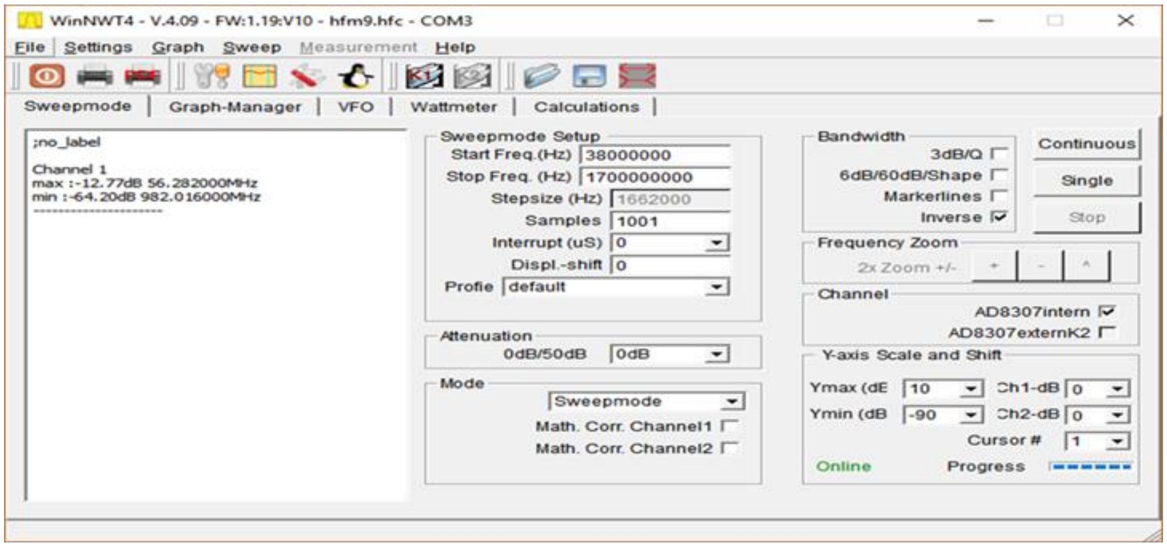
Fig 10. Board SA LTDZ 35-440M

3.2.1 Software

Figure 11 a,b show the winNWT4 version 9.11 application that comes with a software installation package for Windows computers . The CH341ER.exe is the driver for the USB to serial converter chip that the card uses. The device appears as COM port when connected to USB.

The software is WinNWT4 supports many similar devices, and it contains the most important parameters for sweeping, version and all the other parameters that SA uses.

Here is the control window:



```
;no_label
Channel 1
max :-12.77dB 56.282000MHz
min :-64.20dB 982.016000MHz
```

Sweepmode Setup
Start Freq.(Hz) 38000000
Stop Freq. (Hz) 1700000000
Stepsize (Hz) 1662000
Samples 1001
Interrupt (uS) 0
Displ.-shift 0
Profile default

Attenuation
0dB/50dB 0dB

Mode
Sweepmode
Math. Corr. Channel1 ☐
Math. Corr. Channel2 ☐

Bandwidth
3dB/Q ☐
6dB/60dB/Shape ☐
Markerlines ☐
Inverse ☑

Frequency Zoom
2x Zoom +/-

Channel
AD8307intern ☑
AD8307externK2 ☐

Y-axis Scale and Shift
Ymax (dB 10 Ch1-dB 0
Ymin (dB -90 Ch2-dB 0
Cursor # 1
Online Progress

Fig 11.a. control window of the winNWT4 version 4.09

Here is the measurement results window:

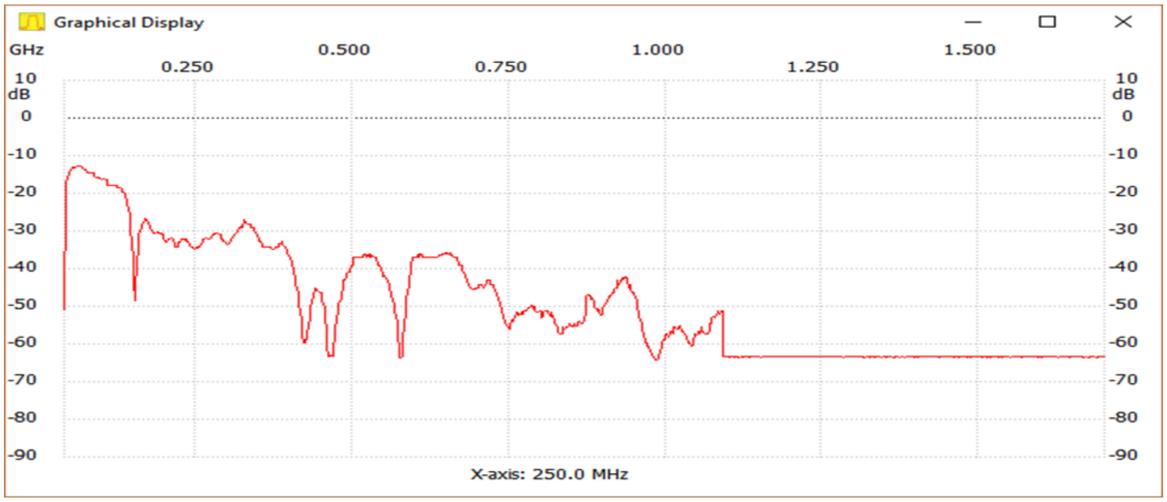


Fig 11.b. Measured graph of the winNWT4 version 4.09

To be able to run the SA in winNWT4 it must be connected to PC port and then choose the parameter and sweep out signal that is coming from input of SA.

After installing all the required drivers and libraries in python, I began the first step of the project by doing a simple communicate with the LTDZ via serial using python, so In order to communicate I needed to discover what are baud rate, connections mood, and other categories that are default for LTDZ.

The following list was elaborated with the parameters that it was thought that the platform should be able to configure.

- Frequency parameters: Central frequency and span & Stop and start frequency
- Bandwidth parameters: Resolution Bandwidth (RBW) value and mode
- Amplitude parameters: Rf. Attenuation,
- Attenuation mode: auto or manual, Reference Level.
- Trace parameters: Average. Sweep parameters: Single or continuous sweep, Sweep time, Sweep points.

3.3 Overview of the Project block diagram

In this section we are going to talk about the overall block diagram that explains the idea of the whole project. You will see in the block diagram in Fig.12 an unknown block, and these blocks are prohibited to disclose it based on internal rules for Embit, I didn’t get the permission to show any internal or detailed picture and this refer to the company policies that I can’t exceed.

The part that is prohibited to talk about is the blue block as you can see in the figure below, and this block contains the mechanical part in addition to the array of LoRa’s that needed to be tested, so the maximum I can said about this box is that it’s a electromechanical bench that do all the steps that must be done before starts testing LoRa, and sure it gets power from external device.

The arrow going down from the block diagram to the j-th number of SA’s. The arrow explain that there are j-th RF signal that must be send to the devices via ground-to-ground “G-S-G” , so the connection is done from the LoRa Output Pin to the SA connector.

Every RF signal is not just one signal, rather its i-th row signal that must be sent simultaneously to an i-th number of SA’s, for instance from the figure the first RF signal represent the i-th LORA signal, will be sent gradually to j-th SA’s, from 1-SA to the i-th SA to be tested.

The BBB with a 3 way arrows around it, the down arrow represent the communication between the SA and the BBB. Here the application I build in the BBB using python read the values coming from the LORA's, and we check signal by signal if the frequency versus power is as expected for every LoRa device.

After checking all signals, two actions must be done. first if the signal is expected so the upper link connect the BBB will communicate with the server to assign its specifications such as ID, test done in a screenshot picture, and then a paper is printed with all these specification for each LoRa, to be delivered with it when it will be sold.

Second action take place if the signal isn't as we expected, here the arrow between the blue block and the BBB represent the GPIO's doing action to turn on a led under the desire LoRa, and continue with the process normally.

After all the LoRa's have been checked and the specification have been printed, and the led turned on under the broken LoRa, so we take the unwanted one, and begin to check what could be the other problems of that specific one, and when we catch the problem we will fix that issue so in the future we don't face the same problems, and this how we optimize our test.

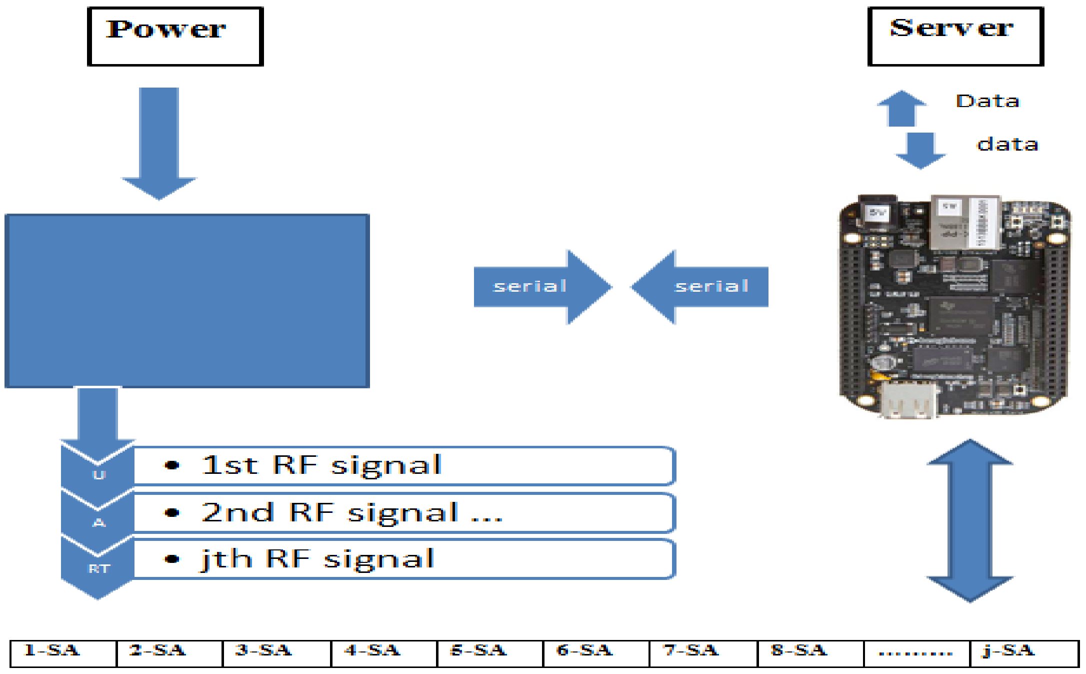


Fig 12. Overview structure of the project

3.4 Serial interface protocol

In this section I’m going to write about the serial interface protocol of the SA. What are the letters it understand, what are the aims of each letter, size of each letter, and what it must return back.

"r" letter aim for switching off the attenuators and it has data length 2 bytes, and we must receive 0 bytes in return. This command outputs for the various attenuators are switched as of firmware version 1.10, also the attenuator from the AMATEUR RADIO are driven.

"v" letter aim to Query the version of the firmware, it has a data length 1 byte and it must return 1 byte. This command displays the number of the currently used firmware is read in the PIC that Starts with the number 100 which means version 1.00.

"s" letter aim for Status queries the NWT it has data length 1 byte and we expect to get 4 bytes. The first byte containing the version number, second byte contains the settings of the attenuator (With this feedback, the attenuator is adjusted in the software as it is connected in NWT, and this is required if the SW was stopped and restarted without the NWT-HW has been restarted). The Bytes 3 and 4 are the result of A / D converter query on AN2 PIN4 the PIC. Byte 3 is LOW and byte 4 is HIGH the measurement result. With this measured value that the switch on SA is reported back by FA in the SW. In the FW version 4 to 7 of the command returns in byte 3 + 4 the value "0".

"e" letter define as oaks, and the all data length that must be sent with this letter are of 13 bytes, while no byte expected to be in return. This command allows an exact adjustment made to the clock frequency. The "e" followed by 10 characters all letters must be capitalized, except the command e! Then the first 2 characters yield a HEX number, while the last byte is the setting of the PLL in the DDS. In the byte interleaved the multiplier of the PLL and the steepness of the power source in the control loop. For these settings you have to look at the datasheet of the AD9951 accurately. The PLL byte is loaded from firmware version 104 in PIC at Power ON, but in the data telegram of the command it has to be send, otherwise the calibration does not work. From the FW version 1.14 the PLL Byte is only considered in the FW version 3. An example to explain this command:

e 0ABCC77 11800 corresponds to a clock frequency of 400.000000 MHz exactly
e 0ABCC6BCF300 corresponds to a clock frequency of 400.000400 MHz exactly

"m" letter aimed to measure value, it has data length 1 byte and we expect to return 4 bytes. This command is used to query the measurement channels. It will be returned two bytes to the PC for each channel. They contain the 10 bit A / D conversion results of the Measurement channels.

"f" letter known as VFO, and all data length that must be sent with this letter are of 10 bytes, and No Byte expected to be returned. The data must be sent to the pic are in the form: f 007030000 VFO is set to 7:03 MHz. With this command a relay in NWT500-special is switched. This NWT is an extra version with a relay which grinds in the SWV measurement head. With "0" it is turned on and off again with "1". Condition, the FW version 11 is PIC.

"x" Sweep n with the AD8307 10-bit converters. The data length that must be sent with this letter are is 22 bytes. The data must be sent to the pic are in the form: x 002000000 00014000 2000. x is 1 byte, the Initial frequency (Start frequency) is 2,000,000 Hz from bytes 2 to byte 10, the increment (step size) is 14000 Hz and it's from byte 11 to 18, the number of measurement points (number of samples) is from byte 18 to 22 example 2000 Hz. The Data that must be received from the PIC are 8000 bytes are given in response to the PC. Every sample there are 2 byte send two times channels, so 2*2*2000 is 8000 sample.

"w" letter is the same as "x" but is linear, where in our device this letter didn't work.

4. Test and Results

In this section we are going to talk about the software how it have been built step by step, the result of our tests, calibration and program of the LoRa device.

4.1. Build the software

We start testing the LoRa over the software winNWT4 using the device LTDZ. WinNWT4 is a build-in software compatible with the device over windows environment, but since our aim is different than just do a normal sweep, so we needed to build our own application. To achieve the goal we began writing the code for the software step by step and from scratch.

In order to build the functions for our application we used the instructions of the book linNWT [5] in chapter 6 under the title of protocol interface in serial. This book explains how a new software version in Linux was built for the NWT4000 device. We follow the instructions to be able to build the algorithms for our own application, so I started writing my code following this manual and open source software [6] that is written in C++ language. Under the Linux environment over the BBB I preferred to write in python language since it's more powerful for the machine learning environment, and I already had a good knowledge in this language so I can begin faster.

I began to search for the baud rate (Baud rate refers to number of signal or voltage level changes per second. Unit of baud rate is bauds per second. For example baud rate is 9600 means 9600 signal level changes are happening within a second) of this device and it was 576000.

The arguments of the serial interface needed to be set in a proper way for this device for example I needed to switch off any software (XON/XOFF) or hardware (rts/cts, dsr/dtr) handshake, since it is not supported by the device and/or cable and The commands need to be send without any EOF characters. This is why it did not work from the beginning, beside that I need to send them in same command the 8F & v because in python it takes the /n between two commands.

I began to search for all the existing building libraries that could help me in my project, and the most important was the "serial library" where it allows me to communicate through serial, but I need to include all the parameters like the name of the port, baud rate, byte size, parity, timeout and many other parameters that are related to the SA. Sure in addition to very important libraries for instance date time, platform, array, struct, and many others, each has its own functionality that could help me achieving my goal, this is an example of how the parameter must be set:

with serial.Serial('/dev/ttyUSB0', 57600 ,bytesize=8, parity='N' , stopbits=1 ,timeout=10, xonxoff=0, rtscts=0, dsrdtr=0) as ser:

So I continued searching other parameters of the Device from the open source library that was available in the website [6]. When all was founded, I set all the connection parameter that makes me able to communicate with the SA from the BBB easily.

Since each command in the direction NWT-Pic is initiated with a small letter. I start trying to send the small letters following the manual in binary, decimal and hexadecimal using the ascii table [7] until I assure that the device read the letters in hexadecimal form. I began trying with letter “v” but nothing responds, so i did a deep search until i discover that I must send 00x8F before doing any kind of communication I want to do with the SA, so I fix the problem and it was not working as well.

I started writing my code by closing the serial port then open it to check if it’s working properly and then flush the input buffer to discard all its contents inside and flush the output buffer, aborting current output and discard all that is in buffer.

After cleaning the buffer I needed to begin communicating with the SA following the manual as you can see in section {3.3}. I send the letter “v” which represents the version and since it’s the simplest one because it is just 1 byte to send and 1 byte to receive. I managed to do it by encode the hexadecimal that I need to send and the communication went well as expected, so I received 1 byte represented by the number 119 versions, then continue with the other commands till I reach the important ones that is sweeping the signal.

The first thing I did since I have multiple function is to create an input argument with a list to let the user choose which operation he would prefer to do, then depend on the entered argument we go to the specific function and begin executing.

Next command was letter “s” and here we receive 4 byes in the form b'\n\x00\x00\x00’.
Each byte has a different meaning from the other you can check section {3.3} to understand the meaning of this argument, and you can have a look at Appendix-A to see the 4 bytes output of the command and how to run this command in python.

The second letter was “m”, for this command the SA must returned 4 bytes as well but different meaning from the previous one letter “s” also you can check section {3.3} to understand the meaning of this argument, and you can have a look at the [appendix A] to see the 4 bytes output of this command and the function that run this command in python.

The most popular function between these commands and it is very important for my project was the sweeper and letter “x” or “w” represents this operation where the w is linear and x is logarithmic scales. I begin to follow the manual but it didn’t respond at first, then after trying many ways and study the structure of this device I discover that we need to add some other parameters to let it work.

The x parameter is not only one byte as “v” rather 22 bytes as written in the manual, but its 25 byte that must be sent to let the SA understand this command and it’s in the form: x 002000000 00014000 2000 100 You can check the section {3.3} to understand each byte in details.
First the user needed to enter the initial frequencies he wants to sweep then the end frequency, and after entering both of them the bandwidth is calculated automatically based on both frequencies, and printed out to let the user check his range he wants to sweep in another time.

Second the user must enter how many samples he wants to do in this range, and then we calculate the step size from the bandwidth and number of samples, the formula to calculate is the following: **step size = bandwidth/ number of sample.**

Note that I multiply the step size and the frequency by 10 because the bytes that must be sent to the SA is in a form of 9 bytes and to arrive till gigabytes we need 10 bytes, so we need to multiply by 10.

After printing the step size, we need to enter the last argument that was not written in the manual, and it is the scan time for each sample byte and its 3 bytes then sleep 0.1 second. here the user enter all the parameters that is needed to do the sweep, we can use them to run our program. So I begin with the continuous sweeping which have the role to continue sweeping as much as the user wants till he stop it manually. Every single time in the continuous sweep a 25 bytes is sending from the application to the SA via serial as explained before.

After confirming that we sent the correct commands, we did a delay for 1 second to let the device finish the process, so we gave it all the time needed to process the information and start to read from the serial cable what comes directly from SA. The length must be 4*samples bytes, and then print what I got from the spectrum. An example of how to read all the bytes that comes from serial is in the form:
Coming bytes=ser.read(4*num_samples) here it allows to read up to 4*samples number bytes.

I find a build-in function in python which unpacks the incoming data bytes, then put them in an array and then I print the maximum and minimum values of this array. The function to unpack bytes comes in the form of:
Unpack-array = np.array(struct.unpack('<'+str(len(arriving bytes))+'B', arriving bytes))

Reading the unpack array many times, I see that we need to flip these data to a real data that a human can understand, and these real data are the peak power of the signal. The rule to converts the unpack data coming into dBm is in the form of:
dBm=(LowByte*256+HighByte)*0.191187-87.139634 , it was not easy to find such a rule to do this calculation.

Then I begin to create a round up list for the dBm values that we got, Then I created a for loop for the single sweep counting the incoming data. Each sample is sending 4 bytes in the form of "high, low, high, low" encoding each analog to digital converter in 2 bytes(so it sends info twice "I don't know why") so I create a for loop which print the 4 bytes for each sample, so in this step I'm taking every 4 bytes as a 1 sample.

To find the frequency there were no data or anything that is coming from SA that explains how, so I need to set the frequency myself, so what I did is to assign a frequency for each sample power, and this can be achieved by multiply the step size by number of samples we are getting and then add to them the initial frequency which the user enters. So the calculation was done in the same loop of single sweep in parallel with the power sample.

This code below shows us the single sweep for all the samples and how I calculated the dBm and how to assign frequency in the same time for each power sample.

```
for i in range(0,sam,4):
        print(tmp[i:i+4])
        sleep(0.1)
        dBm= (((tmp[i])+ (256*tmp[i+1]))*0.191187)-87.139634
        round_dBm=round(dBm,3)
        sleep(0.1)
        lst.append(round_dBm)
        dBm_freq=((i*step_size)//4)+start_freq
```

after printing the readable values of power samples and at which frequency the software will find the frequency which has the maximum power “dBm” and its position in the list and print them, then check if the LoRa signal is between the frequency range that we are expected or no. after checking, if the device is not correct or doesn’t have the power we needed to see or is not at the frequency that we expect a led located below the desired LoRa is turning on using GPIO pins in the BBB, so we can go and check what is the specific problem of that device, depending in the problem we get we will go and fix what is wrong, so in the future work we don’t fall in the same problem and this how we optimize our test.

In the end we clean the buffer and close the port so in the next test for a new LoRa devices will be ready and clean of old information.

I have done some timings from the device and it appears to send output at around 4.2ms/sample. This means a 6,000 sample sweep will take ~25seconds (6000 * 0.0042). The time per sample doesn't seem to change with center frequency, number of samples or bandwidth.

For the rest of the command i tried didn't work as expected, actually I don't need all of them, but for example "w, o, n, m" they didn't work as expected you can check the section {3.3} to understand what they do.

This table below shows a brief description of the commands with their argument that must be sends to the device. There are some of them working and the others that doesn't.

Command	Description	Arguments	Returns
x	Receive in log (power) mode	`"%09d%08d%04d", frequency, step Size, numSamples`	Data(didn't work)
w	Receive in linear(power) mode	"%09d%08d%04d", frequency, step Size, numSamples`	no
f	CW frequency transmission	"%09d", frequency`	NONE
v	Get firmware version	No argument	BYTE
r	Set attenuation level	2 bytes	NONE
m/n	???	????	????
s	Get the status	No arguments	Version/aten /???
e	Frequency correction	???	NONE

Table 1. Briefly description of the commands that must be sent to SA

After I have finish the code now I can begin to test the devices like LoRa using my program to see the data received from the pic and compare this data with the data of winNWT4 version 9.11 and it must be similar if my code was well done.
To see the program done in details you can check Appendix-A my code section.

4.2 Testing the LoRa devices

I start to do the measurements in the available SA that exists in the company named FSH6, this SA is almost ideal. I used this SA FSH6 to study the LTDZ SA itself, and in addition to test the LoRa signal to assure that the devices is well before using my software or winNWT with LTDZ SA to test the LoRa.

Fig 13. The Handheld SA R&S ® FSH6 provides a frequency range of up to 6 GHz**.**

The signal that the SA FSH6 shows for LoRa was perfect as we expected and it contains the peak, sure not ideal but almost the same 14 dBm of power at 686.1 MHz as you can see in the Fig.14 below.

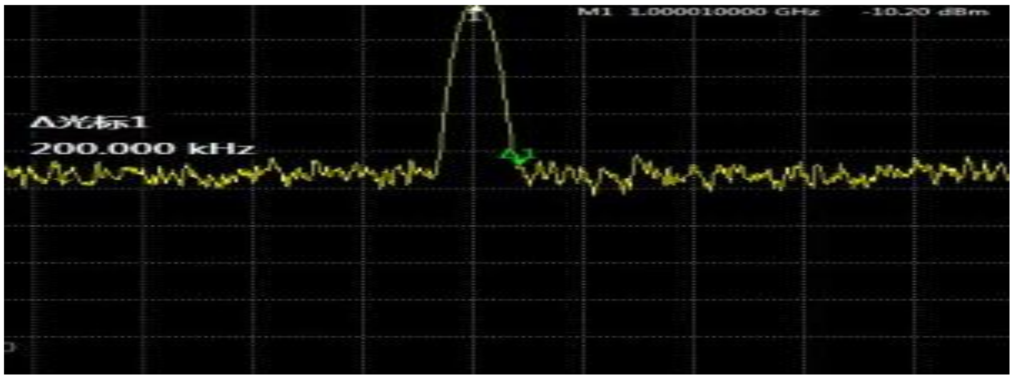


Fig 14. Signal of LR1276S in FH6S

After seeing the real signal of LoRa how must be, so I start to test the EBI-LoRa in the LTDZ using winNWT4 and I had these results for different scenarios.

First scenario was without any connection to the input of the LTDZ and the measured signal here is equal to the noise floor of the device, so around -60/-70 dBm as you see in Fig.15a.

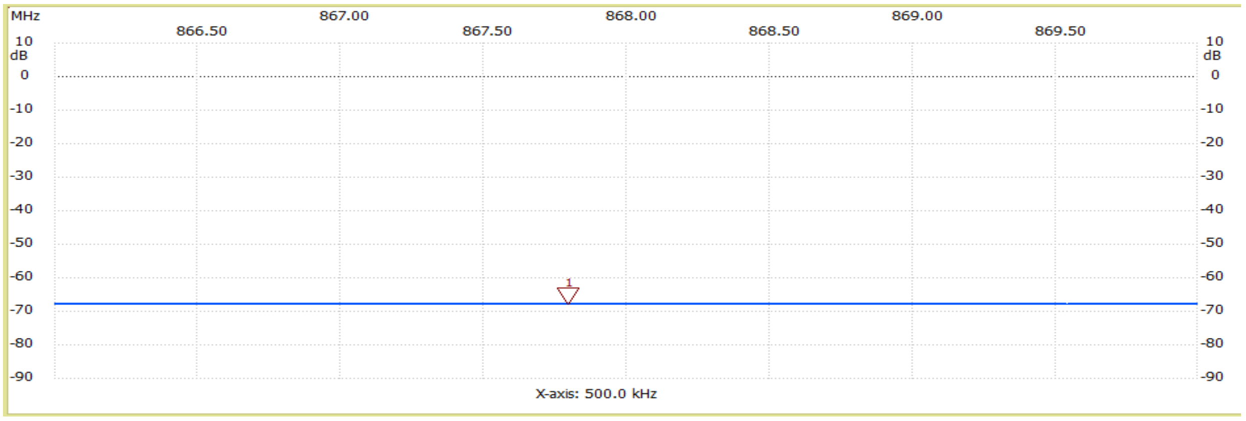


Fig 15.a. LTDZ with no connection

The second Scenario, I connect input to output directly to see how much is the device mistaken before trying the meaningful signal. In theory it must be zero, but in practice it was not at all, the median number of the sweep was around 5 dB, as shown below in Fig.15b.

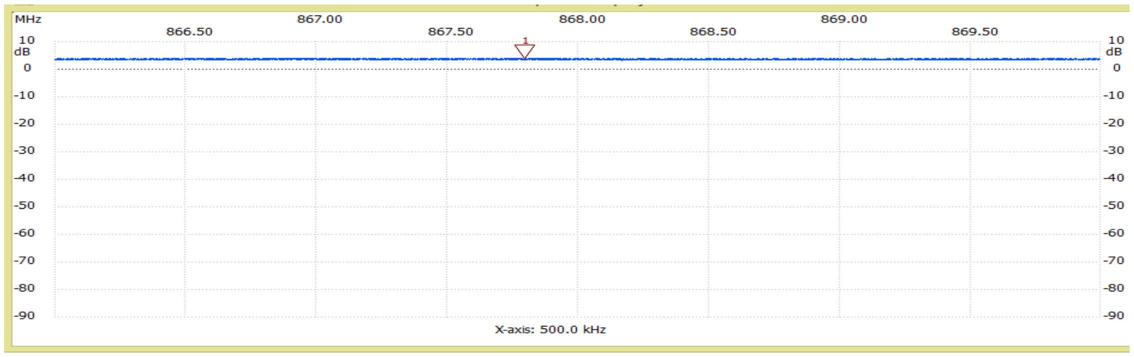


Fig 15b. LTDZ in-out connect

The third Scenario, I connect the LoRa device that is generate around 14 dB at 868.1 MHZ in theory, sure in practice we will not get the exact values as there will always be some issues that can affect the accuracy of the measurements from internal design of LoRa point of view, the LoRa operates by the EBI application and we got the result in Fig.15c below.

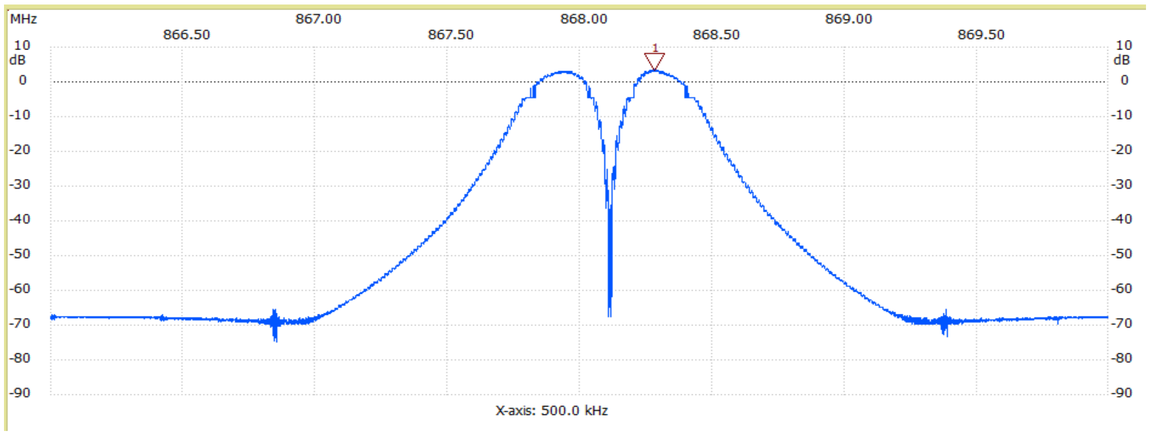


Fig 15c. LoRa device connect to LTDZ

So as we can see that the spectrum gives around +4 dB as a maximum, and it must be +14 in theory.

If we put an attenuator -20dBm between the LoRa and the spectrum reader analyzer in theory must be -6 dB, but in practice it was not and you can see in the figure below the signal gave maximum around 0.

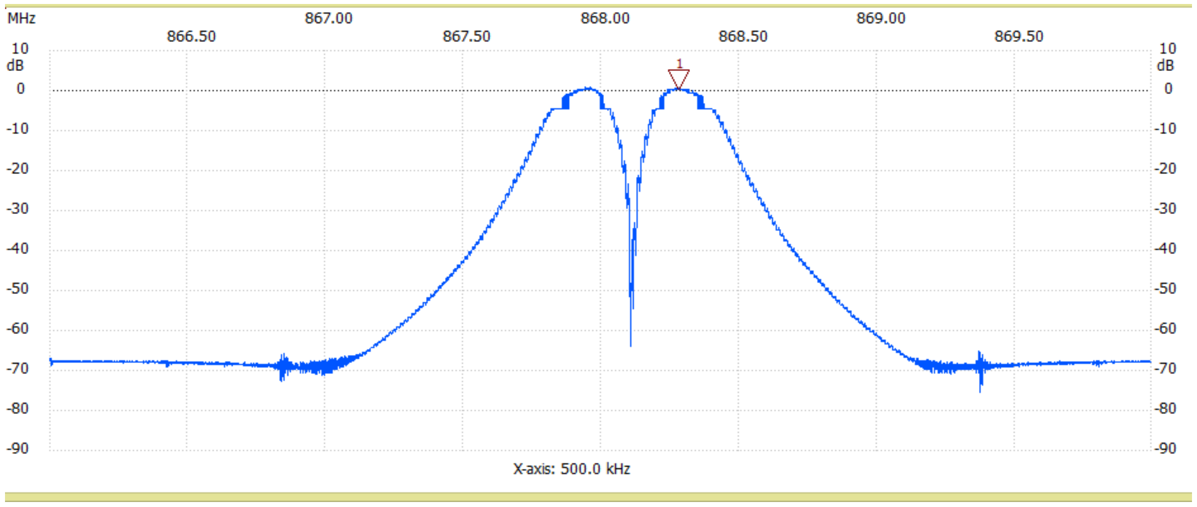


Fig 15.d. LoRa device connect to LTDZ with 1 20 dB attenuator

I decided to continue more in the SA to confirm that it’s not linear at all so I put 2 attenuator in series, and in theory the result must be around -26, but rather it was -35 dB as shown in Fig.15e below.

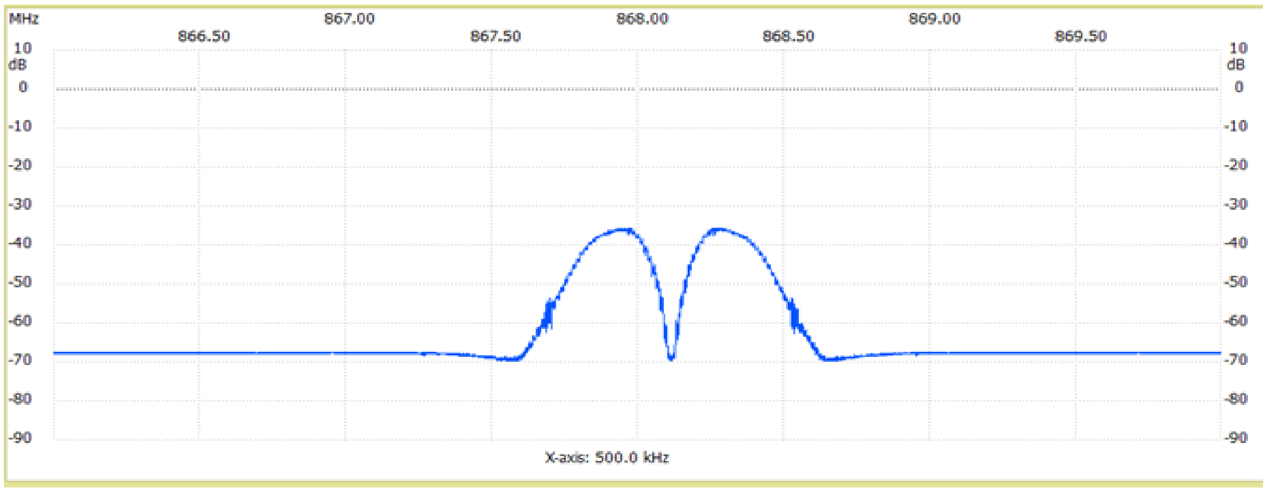


Fig 15.e. LoRa dev connect to LTDZ with double 20 dB attenuator

After all this different and crazy results, and non-constants outputs every time I declare more that this spectrum is not linear at all.

The table below shows the difference measurements between the winNWT4 and the FSH6 SA and my software that I build using different scenarios, so if you look at the table below you will see that my software have too much close value to the winNWT4, but the theory is closed to the FSH6 SA.

	Theory	**FSH6**	**NWT4**	**My software**
LTDZ (out-to-in)) directly	0 dB	0.75 dB	7.5 dB Or 5.5 dB	[5.58,9]dB
LTDZ (out+ att1)	-20 dB	-19.5 dB	-18 dB	[-14,-16] dB
LTDZ(out+att2)	-20 dB	-17.5 dB	-18 dB	[-14,-16] dB
LTDZ(out+att1,2)	-40 dB	NA	-48 dB	[-46,-48] dB
LoRa1 + att1	-6 dB	-6.5 dB	1.19 dB	[1-4]dB
LoRa1 + att2	-6 dB	-6.3 dB	1.2 dB	[2-4] dB
LoRa2+ att1	-6 dB	-7.26 dB	0.42 dB	[1.5-2]
LTDZ (out-to-in)) directly	0 dB	0.75 dB	7.5 dB Or 5.5 dB	[5.58,9]dB

LTDZ (out+ att1)	-20 dB	-19.5 dB	-18 dB	[-14,-16] dB
LTDZ(out+att2)	-20 dB	-17.5 dB	-18 dB	[-14,-16] dB
LTDZ(out+att1,2)	-40 dB		-48 dB	[-46,-48] dB
LoRa1 + att1	-6 dB	-6.5 dB	1.19 dB	[1-4]dB
LoRa1 + att2	-6 dB	-6.3 dB	1.2 dB	[2-4] dB
LoRa2+ att1	-6 dB	-7.26 dB	0.42 dB	[1.5-2] db

Table 2. Test table of different scenario with different SA

So after all the proofs and the data that we had saw until now my collogues decided that this cheap SA is not compatible with our project, since we need an exact value for the power of the signal, beside we need the peak so that we need to change it to a better SA that already exist in the company.
As overall the project it could be achieved if we compensate the notch problem and there was a way I found to do it for the original NWT4000 which I did a map algorithm in a table to remove the notch, but I tried the algorithm error map on this LTDZ SA it didn’t work since the notch is not linear, and every time we measure the same scenario it gives different values.

4.3 Program and calibrate the LORA

J-Link is the USB-to-ARM Single Wire Debug interface hardware developed by SEGGER, an embedded systems HW and SW company. Many different vendors of ARM Cortex M-series microcontrollers include SEGGER's J-Link HW on their development boards as a means of supporting software debugging.

My goal was to calibrate and program the LoRa device by myself using Linux, so I begin to calibrate first the ready programmed module using the EBi commands, and I wrote a python program to make this done via serial. [appendix D] following the manual of the EBI commands, and I was able to write a good code to similar job of EBI software, I tried it in the SA and it gave me the same signal we were getting using the EBi software.

Sure I used libraries, and serial com something similar to the code of the SA application, so in general it was easy.

Then I start to program the module itself, using the j-link segger and begin to assign to it some hexadecimal file to it using the segger application then assign another one using the load file command. In this step of the work my company asks me to stop thesis and begin writing the dissertation.

4.4 The Setup of Experimental work

Figures 16a,b show the initial setup while I was building the software and testing the LoRa's devices. The list of the test parts are listed in Table 3 below:

Number	Component
1	BBB
2	FTDI
3	USB cables
4	LTDZ SA
5	Attenuators
6	Lora's devices
7	Antennas

Table 3. Connection setup

So in brief the setup connections were as follow:

As shown in fig 16a. The PC running the BBB Linux via cable to provide power and connection where the python program that I used to build the application run. The EMB-LR1276S is connected from Micro-B-USB to the EBI application via USB cable where this application is compatible with the device and both them are done by Embit Company, then from the output Pin of the device we connect a cable to the input connector of LTDZ SA. , then the LTDZ SA is connected to the BBB USB host named "USB0" for power and connection where we read the signal, and control all the operations needed to be done. We use the FTDI to have a serial communication in ssh session.

Figure 16b shows the connection when I was doing the programming of the device using j-link.

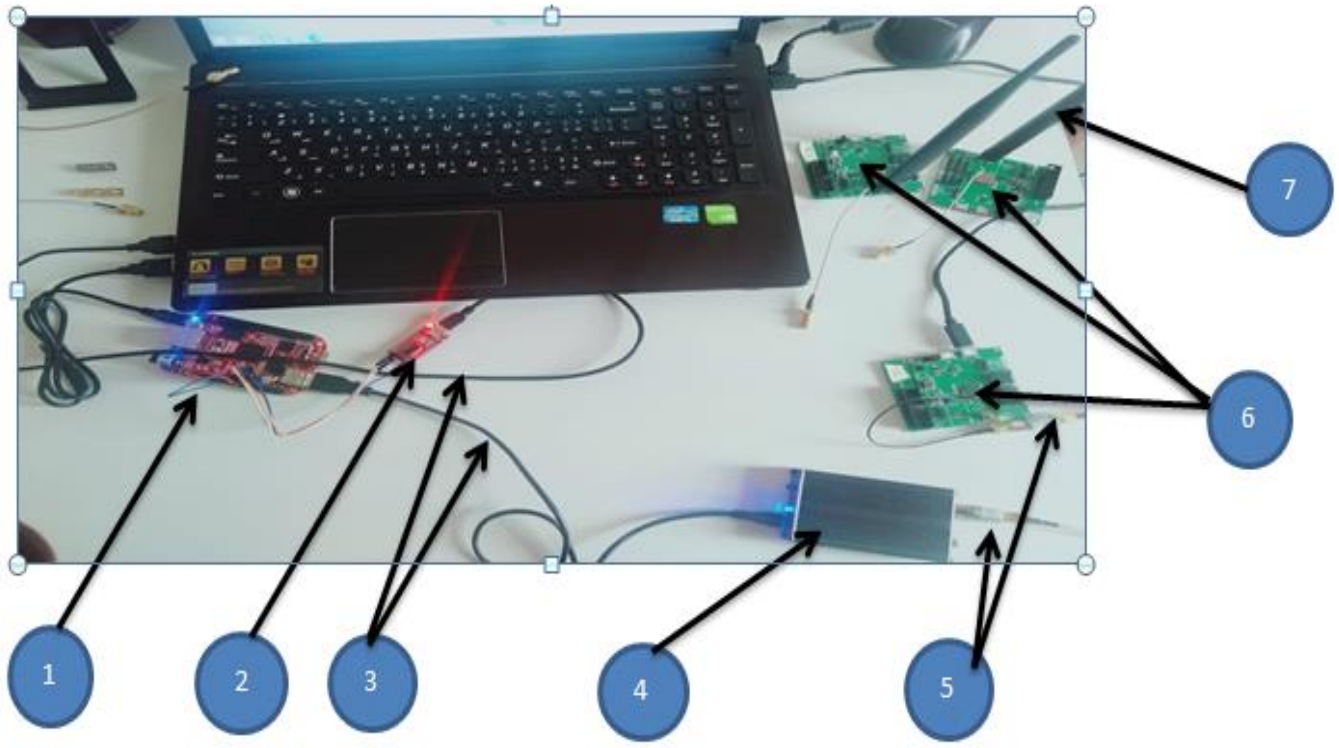


Fig 16a. Setup of my work

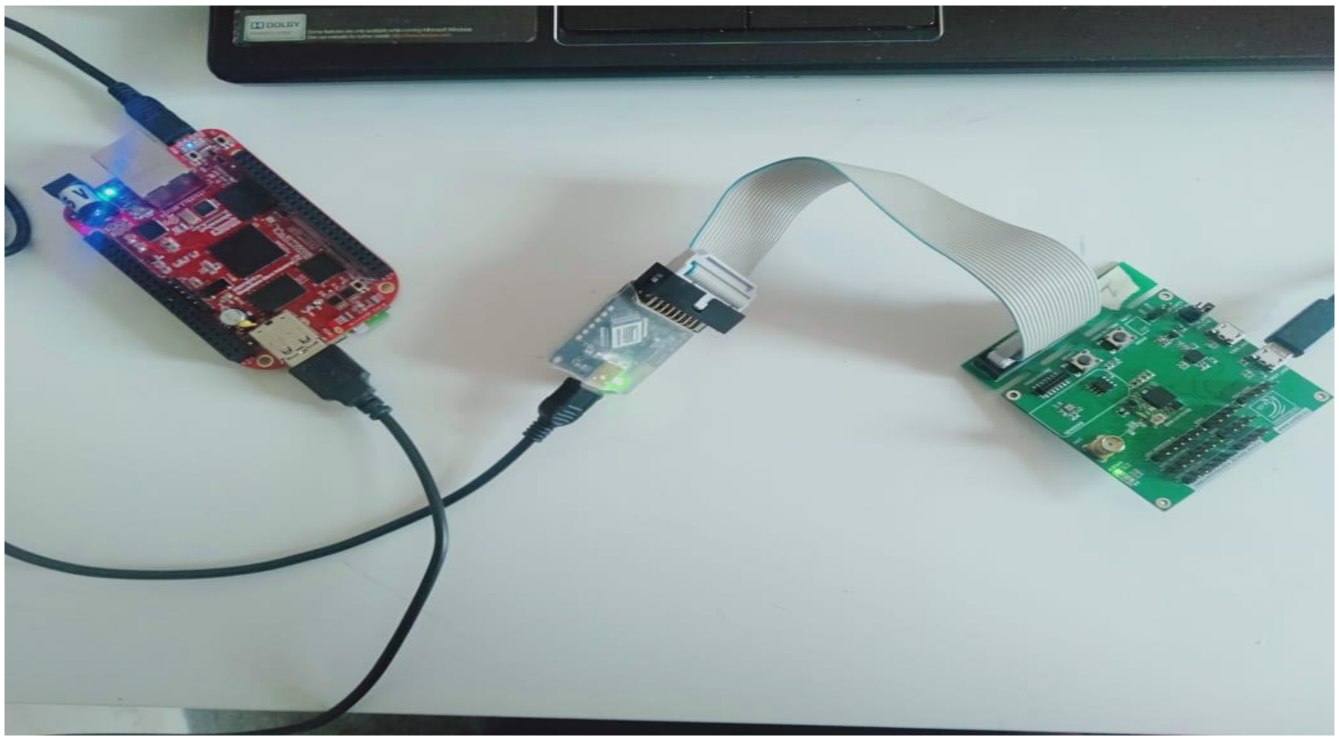


Fig 16b. Setup of my work

Conclusion

The main objective of this work thesis was to speed up the testing of the produced devices in an economical and improve the ending checkup processes. A thorough tests campaign was conducted from September till March in order to achieve the objectives of the project. In this project, I was focusing on the experimental methodologies that I applied it to the work to get positive results.

All done tests have shown a positive outcomes and measurable results based of the available facilities that I had, also depending on the studies, and the device performance. Even we didn’t have the full confidence for the first cheap SA device which was used at the beginning to carry out the measurements.

Moreover, As this project mainly focuses on testing real signals which should have an accurate measurement setup, even with that SA one we are able to study and build a program to read the signal correctly from its interior sample by sample, and then check if the measured signal is correct as was designed and we expect, and so continue to do the usual process.

Bibliography

[1] Geekcreit® SA usb ltdz 35-4400m signal source with tracking source module rf frequency domain analysis tool with aluminum shell Sale - Banggood.com

[2] VMA's Satellite Blog (vma-satellite.blogspot.com)

[3] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper (davidellis.ca)

[4]https://4.bp.blogspot.com/hPmLN1kn37I/Wd_992TViVI/AAAAAAAADmQ/nGgOtRJ4 4fEEUdW7i_xQQ_K7s90G1m9VA1wCLcBGAs/s320/ScreenCapture_13-10-2017-00.08.20.gif

[5] Microsoft Word - LinNWT.doc (giga.co.za)

[6]https://web.archive.org/web/20150620070032fw_/http://www.dl4jal.eu/linNWT4_V4_11_09.tar.gz

[7] https://www.sparkfun.com/news/2121

[5] http://www.embit.eu/products/wireless-modules/emb-lr1276s/.

[7] BeagleBoard.org - getting-started

[8] Microsoft Word - e14 BBB_SRM_rev 0.9.docx (sparkfun.com) “bbb manual”

APPENDIX A – codes and outputs

My code

```
import serial
import platform
import pickle
import getopt
import datetime
import struct
import array
import sys
import numpy as np
import binascii
import io
import os
import curses
from time import sleep
from datetime import date
from timeit import Timer
with  serial.Serial('/dev/ttyUSB0', 57600 ,bytesize=8,  parity='N' , stopbits=1 ,timeout=10,
xonxoff=0, rtscts=0, dsrdtr=0) as ser:

############################ check port ########################################

    ser.flushInput()  #flush input buffer, discarding all its contents
    ser.flushOutput() #flush output buffer, aborting current output #and discard all that is in buffer
    ser.close()
    ser.open()
    if ser.is_open:
        print("the port",ser.name,"is OPEN")
        sleep(1)

############################## Input ##########################################

    input1=input("enter the letter you want to send  ")
```

```
################################### "v" ##########################################
if input1== 'v' :
       ser.write(('\x8f' + 'v').encode())
       sleep(0.1)
 print("the version of SA is " ,ord(ser.read(1)))

######################################## "s" ######################################


    elif input1== 's' :
         ser.write(('\x8f' + 's').encode())
         data= ser.read(4)
         if len(data) ==4:
             print(data[0], data[1], data[2], data[3])
             print('version', data[0])
             print('anten', data[1])
             print('other', struct.unpack('<1H', data[2:])[0])
         else:
             print('Got', len(ss), 'bytes back from status command!')

################################## “m” ##########################################
 elif input1 == 'm' :
          ser.write(('\x8f' + 'm').encode())
          sleep(0.1)
          mm= ser.read(4)
          if len(mm)== 4:
             print(" command m return ",mm, "with", type(mm))
          else : print(" the number of bytes comming when send m is ", len(mm))

#####################################     "x"    ################################


    elif input1=='x' :


         start_freq=int(input("enter the initial frequency: "))
         bandwidth=int(input("enter the the bandwith: "))
         print ("the bandwith is : ",bandwidth, "Min freq = ", start_freq ,"and the Max freq = "
,start_freq + bandwidth)
         num_samples=int(input("enter the number of samples. it must be below 9999!!!: "))
         step_size = int(bandwidth / num_samples)*10
         print("the number of step size is :",step_size)
         timeout=int(input ("enter the time out:"))
 center_freq=(start_freq)+(bandwidth/2)
 for x in range(0, 1080):
       sleep(1)
       ser.write(('\x8f' + 'a' + format((start_freq), '09') +
                         format((step_size), '08') +
                         format(num_samples,'04') +
                         format(timeout, '03') ).encode())
       print('SA Sending command :','\x8f', 'a',
                format((start_freq), '09'),
                format((step_size), '08'),
                format(num_samples , '04'),
                format(timeout ,'03'))
       sleep(1)
       xx=ser.read(4*num_samples)
       sleep(0.5)
       print("number of bytes we receive is ", len(xx))
       sleep(1)
       tmp = np.array(struct.unpack('<'+str(len(xx))+'B', xx))
       print(tmp,"the max in matrix ", max(tmp),"the min in matrix ",min(tmp))
       sleep(0.5)
       sam=len(xx)
       lst= []
for i in range(0,sam,4):
             print(tmp[i:i+4])
             sleep(0.1)
             dBm= (((tmp[i])+ (256*tmp[i+1]))*0.191187)-87.139634
             round_dBm=round(dBm,3)
             sleep(0.1)
             lst.append(round_dBm)
             dBm_freq=((i*step_size)//4)+start_freq
             print(" the value of sample ",i//4,"in dBm is " ,
                           round_dBm,"dBm at frequency: ",  dBm_freq)
            sleep(1)
            maxvalue = max(lst)
            print("Maximum Value in the list is:", maxvalue)
            maxpos = lst.index(maxvalue)
       #maxxpos=maxpos+1
           print("\nPosition of max value in the list is:", maxpos)
           freqmaxpos= (maxpos*step_size)+start_freq
           print(" the maximum dBm is ",maxvalue," at frequency", freqmaxpos)
```

```
if  86700000<=freqmaxpos<=86900000:
   print("yes the max is between 867 MHz and 869 MHz ")
else: print( " no the max isnt here  check your signal")
```

Output of my codes

This is the output of my code, I make the number of samples small to decrease the output size because we always get 4 the times number of samples.

As you can see at the beginning we check if the port is open in the beaglebone serial interface, and then we ask the user to enter the parameter he need to see like start frequency, end frequency, and then the program calculate how much is boundaries to let the user check again if he mistake or something.

Also the lines specify what the bytes are received, then at every sample it calculate the dBm at which frequency it is , and it continue doing that till number of samples finishes.

At the end it check the position of the minimum and the maximum, in the list so it specify at which frequency and how much power, and then print the condition, if the peak of our signal is in the frequency region that we need, if not it print an alarm to notify the user.

Here I did only a test with 1 LoRa and 1 -20 dBm attenuation, no others because I don’t need to use many lines, but the idea is the same we can see it for winNWT4.

```
debian@beaglebone:~/test_spectrum$ python3 test_spectrum
The port /dev/ttyUSB0 is OPEN
Enter the letter you want to send v
Receive: b'w’ and this is the version of the device 119in decimal= w in hex.

debian@beaglebone:~/test_spectrum$ python3 test_spectrum
The port /dev/ttyUSB0 is OPEN
Enter the letter you want to send s
Receive: b'\n\x00\x00\x00’.

debian@beaglebone:~/test_spectrum$ python3 test_spectrum
The port /dev/ttyUSB0 is OPEN
Enter the letter you want to send m   #(109in decimal)
Receive: b'f\x00f\x00'

debian@beaglebone:~/test_spectrum$ python3 test_spectrum
The port /dev/ttyUSB0 is OPEN
```

```
Enter the letter you want to send e
Receive b’’ # we receive nothing from this command, so it’s not compatible with our device.


debian@beaglebone:~/test_spectrum$ python3 test_spectrum
The port /dev/ttyUSB0 is OPEN
Enter the letter you want to send   f 007030000
It returns 0 byte as it must be.
debian@beaglebone:~/test_spectrum$ python3 test_spectrum
The port /dev/ttyUSB0 is OPEN
Enter the letter you want to send x
Enter the initial frequency: 86700000
Enter the the bandwith: 200000
The bandwith is :  200000 Min freq =  86700000 and the Max freq =  86900000
Enter the number of samples. it must be below 9999!!!: 200
The number of step size is: 10000
Enter the time out: 200

SA Sending command: � a 086700000 00010000 0200 200

number of bytes we recieve is  800
[ 93   0  93   0 108   0 108   0 130   0 130   0 164   0 164   0 196   0
 196   0 239   0 239   0  35   1  35   1  94   1  94   1 175   1 175   1
 204   1 204   1 200   1 200   1 228   0 228   0 176   1 176   1 207   1
 207   1 176   1 176   1 113   1 113   1  42   1  42   1 251   0 251   0
 206   0 206   0 167   0 167   0 138   0 138   0 109   0 109   0 100   0
 100   0  95   0  95   0  98   0  98   0 100   0 100   0 102   0 102   0
 103   0 103   0 103   0 103   0 104   0 104   0 104   0 104   0 105   0
 105   0 105   0 105   0 104   0 104   0 105   0 105   0 104   0 104   0
 104   0 104   0 105   0 105   0 105   0 105   0 105   0 105   0 104   0
 104   0 104   0 104   0 106   0 106   0 105   0 105   0 105   0 105   0
 105   0 105   0 104   0 104   0 106   0 106   0 105   0 105   0 105   0
 105   0 105   0 105   0 105   0 105   0 105   0 105   0 104   0 104   0
 105   0 105   0 105   0 105   0 103   0 103   0 103   0 103   0 103   0
 103   0 104   0 104   0 104   0 104   0 103   0 103   0 103   0 103   0
 103   0 103   0 103   0 103   0 104   0 104   0 104   0 104   0 104   0
 104   0 103   0 103   0 104   0 104   0 103   0 103   0 103   0 103   0
 103   0 103   0 103   0 103   0 103   0 103   0 103   0 103   0 104   0
 104   0 103   0 103   0 104   0 104   0 104   0 104   0 103   0 103   0
 103   0 103   0 103   0 103   0 103   0 103   0 103   0 103   0 104   0
 104   0 104   0 104   0 102   0 102   0 103   0 103   0 102   0 102   0
 103   0 103   0 103   0 103   0 104   0 104   0 104   0 104   0 104   0
 104   0 103   0 103   0 103   0 103   0 104   0 104   0 104   0 104   0
 103   0 103   0 103   0 103   0 103   0 103   0 105   0 105   0 104   0
 104   0 105   0 105   0 104   0 104   0 105   0 105   0 105   0 105   0
 105   0 105   0 105   0 105   0 104   0 104   0 105   0 105   0 104   0
 104   0 106   0 106   0 105   0 105   0 105   0 105   0 105   0 105   0
 106   0 106   0 106   0 106   0 105   0 105   0 105   0 105   0 104   0
 104   0 105   0 105   0 104   0 104   0 104   0 104   0 104   0 104   0
 105   0 105   0 105   0 105   0 105   0 105   0 104   0 104   0 105   0
 105   0 105   0 105   0 105   0 105   0 104   0 104   0 106   0 106   0
 104   0 104   0 104   0 104   0 105   0 105   0 104   0 104   0 105   0
 105   0 105   0 105   0 105   0 105   0 105   0 105   0 105   0 105   0
 104   0 104   0 104   0 104   0 105   0 105   0 105   0 105   0 105   0
 105   0 105   0 105   0 104   0 104   0 105   0 105   0 104   0 104   0
 105   0 105   0 105   0 105   0 104   0 104   0 105   0 105   0 105   0
 105   0 104   0 104   0 103   0 103   0 103   0 103   0 104   0 104   0
 104   0 104   0 104   0 104   0 104   0 104   0 103   0 103   0 103   0
 103   0 103   0 103   0 103   0 103   0 104   0 104   0 103   0 103   0
 102   0 102   0 103   0 103   0 104   0 104   0 103   0 103   0 102   0
 102   0 103   0 103   0 103   0 103   0 103   0 103   0 103   0 103   0
 103   0 103   0 103   0 103   0 103   0 103   0 103   0 103   0 103   0
 103   0 104   0 104   0 103   0 103   0 104   0 104   0 102   0 102   0
 103   0 103   0 104   0 104   0 103   0 103   0 103   0 103   0 103   0
 103   0 103   0 103   0 103   0 103   0 104   0 104   0 104   0 104   0
 103   0 103   0 103   0 103   0] the max in matrix 251 the min in matrix 0


[93  0 93  0]
 the value of sample  0 in dBm is  -69.359 dBm at frequency:  86700000
[108   0 108   0]
 the value of sample  1 in dBm is  -66.491 dBm at frequency:  86710000
[130   0 130   0]
 the value of sample  2 in dBm is  -62.285 dBm at frequency:  86720000
[164   0 164   0]
 the value of sample  3 in dBm is  -55.785 dBm at frequency:  86730000
[196   0 196   0]
 the value of sample  4 in dBm is  -49.667 dBm at frequency:  86740000
[239   0 239   0]
 the value of sample  5 in dBm is  -41.446 dBm at frequency:  86750000
```

```
[35  1 35  1]
 the value of sample  6 in dBm is  -31.504 dBm at frequency:  86760000
[94  1 94  1]
 the value of sample  7 in dBm is  -20.224 dBm at frequency:  86770000
[175   1 175   1]
 the value of sample  8 in dBm is  -4.738 dBm at frequency:  86780000
[204   1 204   1]
 the value of sample  9 in dBm is  0.806 dBm at frequency:  86790000
[200   1 200   1]
 the value of sample  10 in dBm is  0.042 dBm at frequency:  86800000
[228   0 228   0]
 the value of sample  11 in dBm is  -43.549 dBm at frequency:  86810000
[176   1 176   1]
 the value of sample  12 in dBm is  -4.547 dBm at frequency:  86820000
[207   1 207   1]
 the value of sample  13 in dBm is  1.38 dBm at frequency:  86830000
[176   1 176   1]
 the value of sample  14 in dBm is  -4.547 dBm at frequency:  86840000
[113   1 113   1]
 the value of sample  15 in dBm is  -16.592 dBm at frequency:  86850000
[42  1 42  1]
 the value of sample  16 in dBm is  -30.166 dBm at frequency:  86860000
[251   0 251   0]
 the value of sample  17 in dBm is  -39.152 dBm at frequency:  86870000
[206   0 206   0]
 the value of sample  18 in dBm is  -47.755 dBm at frequency:  86880000
[167   0 167   0]
 the value of sample  19 in dBm is  -55.211 dBm at frequency:  86890000
[138   0 138   0]
 the value of sample  20 in dBm is  -60.756 dBm at frequency:  86900000
[109   0 109   0]
 the value of sample  21 in dBm is  -66.3 dBm at frequency:  86910000
[100   0 100   0]
 the value of sample  22 in dBm is  -68.021 dBm at frequency:  86920000
[95  0 95  0]
 the value of sample  23 in dBm is  -68.977 dBm at frequency:  86930000
[98  0 98  0]
 the value of sample  24 in dBm is  -68.403 dBm at frequency:  86940000
[100   0 100   0]
 the value of sample  25 in dBm is  -68.021 dBm at frequency:  86950000
```

```
[102   0 102   0]
 the value of sample  26 in dBm is  -67.639 dBm at frequency:  86960000
[103   0 103   0]
 the value of sample  27 in dBm is  -67.447 dBm at frequency:  86970000
[103   0 103   0]
 the value of sample  28 in dBm is  -67.447 dBm at frequency:  86980000
[104   0 104   0]
 the value of sample  29 in dBm is  -67.256 dBm at frequency:  86990000
[104   0 104   0]
 the value of sample  30 in dBm is  -67.256 dBm at frequency:  87000000
[105   0 105   0]
 the value of sample  31 in dBm is  -67.065 dBm at frequency:  87010000
[105   0 105   0]
 the value of sample  32 in dBm is  -67.065 dBm at frequency:  87020000
[104   0 104   0]
 the value of sample  33 in dBm is  -67.256 dBm at frequency:  87030000
[105   0 105   0]
 the value of sample  34 in dBm is  -67.065 dBm at frequency:  87040000
[104   0 104   0]
 the value of sample  35 in dBm is  -67.256 dBm at frequency:  87050000
[104   0 104   0]
 the value of sample  36 in dBm is  -67.256 dBm at frequency:  87060000
[105   0 105   0]
 the value of sample  37 in dBm is  -67.065 dBm at frequency:  87070000
[105   0 105   0]
 the value of sample  38 in dBm is  -67.065 dBm at frequency:  87080000
[105   0 105   0]
 the value of sample  39 in dBm is  -67.065 dBm at frequency:  87090000
[104   0 104   0]
 the value of sample  40 in dBm is  -67.256 dBm at frequency:  87100000
[104   0 104   0]
 the value of sample  41 in dBm is  -67.256 dBm at frequency:  87110000
[106   0 106   0]
 the value of sample  42 in dBm is  -66.874 dBm at frequency:  87120000
[105   0 105   0]
 the value of sample  43 in dBm is  -67.065 dBm at frequency:  87130000
[105   0 105   0]
 the value of sample  44 in dBm is  -67.065 dBm at frequency:  87140000
[105   0 105   0]
 the value of sample  45 in dBm is  -67.065 dBm at frequency:  87150000
```

```
[104   0 104   0]
 the value of sample  46 in dBm is  -67.256 dBm at frequency:  87160000
[106   0 106   0]
 the value of sample  47 in dBm is  -66.874 dBm at frequency:  87170000
[105   0 105   0]
 the value of sample  48 in dBm is  -67.065 dBm at frequency:  87180000
[105   0 105   0]
 the value of sample  49 in dBm is  -67.065 dBm at frequency:  87190000
[105   0 105   0]
 the value of sample  50 in dBm is  -67.065 dBm at frequency:  87200000
[105   0 105   0]
 the value of sample  51 in dBm is  -67.065 dBm at frequency:  87210000
[105   0 105   0]
 the value of sample  52 in dBm is  -67.065 dBm at frequency:  87220000
[104   0 104   0]
 the value of sample  53 in dBm is  -67.256 dBm at frequency:  87230000
[105   0 105   0]
 the value of sample  54 in dBm is  -67.065 dBm at frequency:  87240000
[105   0 105   0]
 the value of sample  55 in dBm is  -67.065 dBm at frequency:  87250000
[103   0 103   0]
 the value of sample  56 in dBm is  -67.447 dBm at frequency:  87260000
[103   0 103   0]
 the value of sample  57 in dBm is  -67.447 dBm at frequency:  87270000
[103   0 103   0]
 the value of sample  58 in dBm is  -67.447 dBm at frequency:  87280000
[104   0 104   0]
 the value of sample  59 in dBm is  -67.256 dBm at frequency:  87290000
[104   0 104   0]
 the value of sample  60 in dBm is  -67.256 dBm at frequency:  87300000
[103   0 103   0]
 the value of sample  61 in dBm is  -67.447 dBm at frequency:  87310000
[103   0 103   0]
 the value of sample  62 in dBm is  -67.447 dBm at frequency:  87320000
[103   0 103   0]
 the value of sample  63 in dBm is  -67.447 dBm at frequency:  87330000
[103   0 103   0]
 the value of sample  64 in dBm is  -67.447 dBm at frequency:  87340000
[104   0 104   0]
 the value of sample  65 in dBm is  -67.256 dBm at frequency:  87350000
```

```
[104   0 104   0]
 the value of sample  66 in dBm is  -67.256 dBm at frequency:  87360000
[104   0 104   0]
 the value of sample  67 in dBm is  -67.256 dBm at frequency:  87370000
[103   0 103   0]
 the value of sample  68 in dBm is  -67.447 dBm at frequency:  87380000
[104   0 104   0]
 the value of sample  69 in dBm is  -67.256 dBm at frequency:  87390000
[103   0 103   0]
 the value of sample  70 in dBm is  -67.447 dBm at frequency:  87400000
[103   0 103   0]
 the value of sample  71 in dBm is  -67.447 dBm at frequency:  87410000
[103   0 103   0]
 the value of sample  72 in dBm is  -67.447 dBm at frequency:  87420000
[103   0 103   0]
 the value of sample  73 in dBm is  -67.447 dBm at frequency:  87430000
[103   0 103   0]
 the value of sample  74 in dBm is  -67.447 dBm at frequency:  87440000
[103   0 103   0]
 the value of sample  75 in dBm is  -67.447 dBm at frequency:  87450000
[104   0 104   0]
 the value of sample  76 in dBm is  -67.256 dBm at frequency:  87460000
[103   0 103   0]
 the value of sample  77 in dBm is  -67.447 dBm at frequency:  87470000
[104   0 104   0]
 the value of sample  78 in dBm is  -67.256 dBm at frequency:  87480000
[104   0 104   0]
 the value of sample  79 in dBm is  -67.256 dBm at frequency:  87490000
[103   0 103   0]
 the value of sample  80 in dBm is  -67.447 dBm at frequency:  87500000
[103   0 103   0]
 the value of sample  81 in dBm is  -67.447 dBm at frequency:  87510000
[103   0 103   0]
 the value of sample  82 in dBm is  -67.447 dBm at frequency:  87520000
[103   0 103   0]
 the value of sample  83 in dBm is  -67.447 dBm at frequency:  87530000
[103   0 103   0]
 the value of sample  84 in dBm is  -67.447 dBm at frequency:  87540000
[104   0 104   0]
 the value of sample  85 in dBm is  -67.256 dBm at frequency:  87550000
```

```
[104   0 104   0]
 the value of sample  86 in dBm is  -67.256 dBm at frequency:  87560000
[102   0 102   0]
 the value of sample  87 in dBm is  -67.639 dBm at frequency:  87570000
[103   0 103   0]
 the value of sample  88 in dBm is  -67.447 dBm at frequency:  87580000
[102   0 102   0]
 the value of sample  89 in dBm is  -67.639 dBm at frequency:  87590000
[103   0 103   0]
 the value of sample  90 in dBm is  -67.447 dBm at frequency:  87600000
[103   0 103   0]
 the value of sample  91 in dBm is  -67.447 dBm at frequency:  87610000
[104   0 104   0]
 the value of sample  92 in dBm is  -67.256 dBm at frequency:  87620000
[104   0 104   0]
 the value of sample  93 in dBm is  -67.256 dBm at frequency:  87630000
[104   0 104   0]
 the value of sample  94 in dBm is  -67.256 dBm at frequency:  87640000
[103   0 103   0]
 the value of sample  95 in dBm is  -67.447 dBm at frequency:  87650000
[103   0 103   0]
 the value of sample  96 in dBm is  -67.447 dBm at frequency:  87660000
[104   0 104   0]
 the value of sample  97 in dBm is  -67.256 dBm at frequency:  87670000
[104   0 104   0]
 the value of sample  98 in dBm is  -67.256 dBm at frequency:  87680000
[103   0 103   0]
 the value of sample  99 in dBm is  -67.447 dBm at frequency:  87690000
[103   0 103   0]
 the value of sample  100 in dBm is  -67.447 dBm at frequency:  87700000
[103   0 103   0]
 the value of sample  101 in dBm is  -67.447 dBm at frequency:  87710000
[105   0 105   0]
 the value of sample  102 in dBm is  -67.065 dBm at frequency:  87720000
[104   0 104   0]
 the value of sample  103 in dBm is  -67.256 dBm at frequency:  87730000
[105   0 105   0]
 the value of sample  104 in dBm is  -67.065 dBm at frequency:  87740000
[104   0 104   0]
 the value of sample  105 in dBm is  -67.256 dBm at frequency:  87750000
```

```
[105   0 105   0]
 the value of sample  106 in dBm is  -67.065 dBm at frequency:  87760000
[105   0 105   0]
 the value of sample  107 in dBm is  -67.065 dBm at frequency:  87770000
[105   0 105   0]
 the value of sample  108 in dBm is  -67.065 dBm at frequency:  87780000
[105   0 105   0]
 the value of sample  109 in dBm is  -67.065 dBm at frequency:  87790000
[104   0 104   0]
 the value of sample  110 in dBm is  -67.256 dBm at frequency:  87800000
[105   0 105   0]
 the value of sample  111 in dBm is  -67.065 dBm at frequency:  87810000
[104   0 104   0]
 the value of sample  112 in dBm is  -67.256 dBm at frequency:  87820000
[106   0 106   0]
 the value of sample  113 in dBm is  -66.874 dBm at frequency:  87830000
[105   0 105   0]
 the value of sample  114 in dBm is  -67.065 dBm at frequency:  87840000
[105   0 105   0]
 the value of sample  115 in dBm is  -67.065 dBm at frequency:  87850000
[105   0 105   0]
 the value of sample  116 in dBm is  -67.065 dBm at frequency:  87860000
[106   0 106   0]
 the value of sample  117 in dBm is  -66.874 dBm at frequency:  87870000
[106   0 106   0]
 the value of sample  118 in dBm is  -66.874 dBm at frequency:  87880000
[105   0 105   0]
 the value of sample  119 in dBm is  -67.065 dBm at frequency:  87890000
[105   0 105   0]
 the value of sample  120 in dBm is  -67.065 dBm at frequency:  87900000
[104   0 104   0]
 the value of sample  121 in dBm is  -67.256 dBm at frequency:  87910000
[105   0 105   0]
 the value of sample  122 in dBm is  -67.065 dBm at frequency:  87920000
[104   0 104   0]
 the value of sample  123 in dBm is  -67.256 dBm at frequency:  87930000
[104   0 104   0]
 the value of sample  124 in dBm is  -67.256 dBm at frequency:  87940000
[104   0 104   0]
 the value of sample  125 in dBm is  -67.256 dBm at frequency:  87950000
```

```
[105   0 105   0]
 the value of sample  126 in dBm is  -67.065 dBm at frequency:  87960000
[105   0 105   0]
 the value of sample  127 in dBm is  -67.065 dBm at frequency:  87970000
[105   0 105   0]
 the value of sample  128 in dBm is  -67.065 dBm at frequency:  87980000
[104   0 104   0]
 the value of sample  129 in dBm is  -67.256 dBm at frequency:  87990000
[105   0 105   0]
 the value of sample  130 in dBm is  -67.065 dBm at frequency:  88000000
[105   0 105   0]
 the value of sample  131 in dBm is  -67.065 dBm at frequency:  88010000
[105   0 105   0]
 the value of sample  132 in dBm is  -67.065 dBm at frequency:  88020000
[104   0 104   0]
 the value of sample  133 in dBm is  -67.256 dBm at frequency:  88030000
[106   0 106   0]
 the value of sample  134 in dBm is  -66.874 dBm at frequency:  88040000
[104   0 104   0]
 the value of sample  135 in dBm is  -67.256 dBm at frequency:  88050000
[104   0 104   0]
 the value of sample  136 in dBm is  -67.256 dBm at frequency:  88060000
[105   0 105   0]
 the value of sample  137 in dBm is  -67.065 dBm at frequency:  88070000
[104   0 104   0]
 the value of sample  138 in dBm is  -67.256 dBm at frequency:  88080000
[105   0 105   0]
 the value of sample  139 in dBm is  -67.065 dBm at frequency:  88090000
[105   0 105   0]
 the value of sample  140 in dBm is  -67.065 dBm at frequency:  88100000
[105   0 105   0]
 the value of sample  141 in dBm is  -67.065 dBm at frequency:  88110000
[105   0 105   0]
 the value of sample  142 in dBm is  -67.065 dBm at frequency:  88120000
[105   0 105   0]
 the value of sample  143 in dBm is  -67.065 dBm at frequency:  88130000
[104   0 104   0]
 the value of sample  144 in dBm is  -67.256 dBm at frequency:  88140000
[104   0 104   0]
 the value of sample  145 in dBm is  -67.256 dBm at frequency:  88150000
```

```
[105   0 105   0]
 the value of sample  146 in dBm is  -67.065 dBm at frequency:  88160000
[105   0 105   0]
 the value of sample  147 in dBm is  -67.065 dBm at frequency:  88170000
[105   0 105   0]
 the value of sample  148 in dBm is  -67.065 dBm at frequency:  88180000
[105   0 105   0]
 the value of sample  149 in dBm is  -67.065 dBm at frequency:  88190000
[104   0 104   0]
 the value of sample  150 in dBm is  -67.256 dBm at frequency:  88200000
[105   0 105   0]
 the value of sample  151 in dBm is  -67.065 dBm at frequency:  88210000
[104   0 104   0]
 the value of sample  152 in dBm is  -67.256 dBm at frequency:  88220000
[105   0 105   0]
 the value of sample  153 in dBm is  -67.065 dBm at frequency:  88230000
[105   0 105   0]
 the value of sample  154 in dBm is  -67.065 dBm at frequency:  88240000
[104   0 104   0]
 the value of sample  155 in dBm is  -67.256 dBm at frequency:  88250000
[105   0 105   0]
 the value of sample  156 in dBm is  -67.065 dBm at frequency:  88260000
[105   0 105   0]
 the value of sample  157 in dBm is  -67.065 dBm at frequency:  88270000
[104   0 104   0]
 the value of sample  158 in dBm is  -67.256 dBm at frequency:  88280000
[103   0 103   0]
 the value of sample  159 in dBm is  -67.447 dBm at frequency:  88290000
[103   0 103   0]
 the value of sample  160 in dBm is  -67.447 dBm at frequency:  88300000
[104   0 104   0]
 the value of sample  161 in dBm is  -67.256 dBm at frequency:  88310000
[104   0 104   0]
 the value of sample  162 in dBm is  -67.256 dBm at frequency:  88320000
[104   0 104   0]
 the value of sample  163 in dBm is  -67.256 dBm at frequency:  88330000
[104   0 104   0]
 the value of sample  164 in dBm is  -67.256 dBm at frequency:  88340000
[103   0 103   0]
 the value of sample  165 in dBm is  -67.447 dBm at frequency:  88350000
```

```
[103   0 103   0]
 the value of sample  166 in dBm is  -67.447 dBm at frequency:  88360000
[103   0 103   0]
 the value of sample  167 in dBm is  -67.447 dBm at frequency:  88370000
[103   0 103   0]
 the value of sample  168 in dBm is  -67.447 dBm at frequency:  88380000
[104   0 104   0]
 the value of sample  169 in dBm is  -67.256 dBm at frequency:  88390000
[103   0 103   0]
 the value of sample  170 in dBm is  -67.447 dBm at frequency:  88400000
[102   0 102   0]
 the value of sample  171 in dBm is  -67.639 dBm at frequency:  88410000
[103   0 103   0]
 the value of sample  172 in dBm is  -67.447 dBm at frequency:  88420000
[104   0 104   0]
 the value of sample  173 in dBm is  -67.256 dBm at frequency:  88430000
[103   0 103   0]
 the value of sample  174 in dBm is  -67.447 dBm at frequency:  88440000
[102   0 102   0]
 the value of sample  175 in dBm is  -67.639 dBm at frequency:  88450000
[103   0 103   0]
 the value of sample  176 in dBm is  -67.447 dBm at frequency:  88460000
[103   0 103   0]
 the value of sample  177 in dBm is  -67.447 dBm at frequency:  88470000
[103   0 103   0]
 the value of sample  178 in dBm is  -67.447 dBm at frequency:  88480000
[103   0 103   0]
 the value of sample  179 in dBm is  -67.447 dBm at frequency:  88490000
[103   0 103   0]
 the value of sample  180 in dBm is  -67.447 dBm at frequency:  88500000
[103   0 103   0]
 the value of sample  181 in dBm is  -67.447 dBm at frequency:  88510000
[103   0 103   0]
 the value of sample  182 in dBm is  -67.447 dBm at frequency:  88520000
[103   0 103   0]
 the value of sample  183 in dBm is  -67.447 dBm at frequency:  88530000
[103   0 103   0]
 the value of sample  184 in dBm is  -67.447 dBm at frequency:  88540000
[104   0 104   0]
 the value of sample  185 in dBm is  -67.256 dBm at frequency:  88550000
```

```
[103   0 103   0]
 the value of sample  186 in dBm is  -67.447 dBm at frequency:  88560000
[104   0 104   0]
 the value of sample  187 in dBm is  -67.256 dBm at frequency:  88570000
[102   0 102   0]
 the value of sample  188 in dBm is  -67.639 dBm at frequency:  88580000
[103   0 103   0]
 the value of sample  189 in dBm is  -67.447 dBm at frequency:  88590000
[104   0 104   0]
 the value of sample  190 in dBm is  -67.256 dBm at frequency:  88600000
[103   0 103   0]
 the value of sample  191 in dBm is  -67.447 dBm at frequency:  88610000
[103   0 103   0]
 the value of sample  192 in dBm is  -67.447 dBm at frequency:  88620000
[103   0 103   0]
 the value of sample  193 in dBm is  -67.447 dBm at frequency:  88630000
[103   0 103   0]
 the value of sample  194 in dBm is  -67.447 dBm at frequency:  88640000
[103   0 103   0]
 the value of sample  195 in dBm is  -67.447 dBm at frequency:  88650000
[104   0 104   0]
 the value of sample  196 in dBm is  -67.256 dBm at frequency:  88660000
[104   0 104   0]
 the value of sample  197 in dBm is  -67.256 dBm at frequency:  88670000
[103   0 103   0]
 the value of sample  198 in dBm is  -67.447 dBm at frequency:  88680000
[103   0 103   0]
 the value of sample  199 in dBm is  -67.447 dBm at frequency:  88690000
```

the Position of max value in the list is: 13
the Position of min value in the list is: 0
the maximum dBm is 1.38 at frequency 86830000
the minimum dBm is -69.359 at frequency 86700000
yes the max is between 867 MHz and 869 MHz

APENDIX B - specification of the Devices used

BBB element14 overview

BBB a no less than a computer. Yes, you read it right. It is packed with everything you find in a desktop or a laptop. A powerful processor, memory, and graphics acceleration all shrunken down as chips and soldered into a single circuit board. Therefore, it is fair enough to call it a single-board computer.

This powerful microcontroller board can be hooked up with a display, speakers, Ethernet network, keyboard and mouse. Moreover, it can be used to boot up a LINUX operating system.
It is a powerful tool for hobbyists and researchers to build sophisticated projects and a good approach to learning more about LINUX based operating systems.

The BBB is a pocket-friendly, compact development platform with excellent support from its fast growing community. It is a perfect device for physical computing and smaller embedded applications.

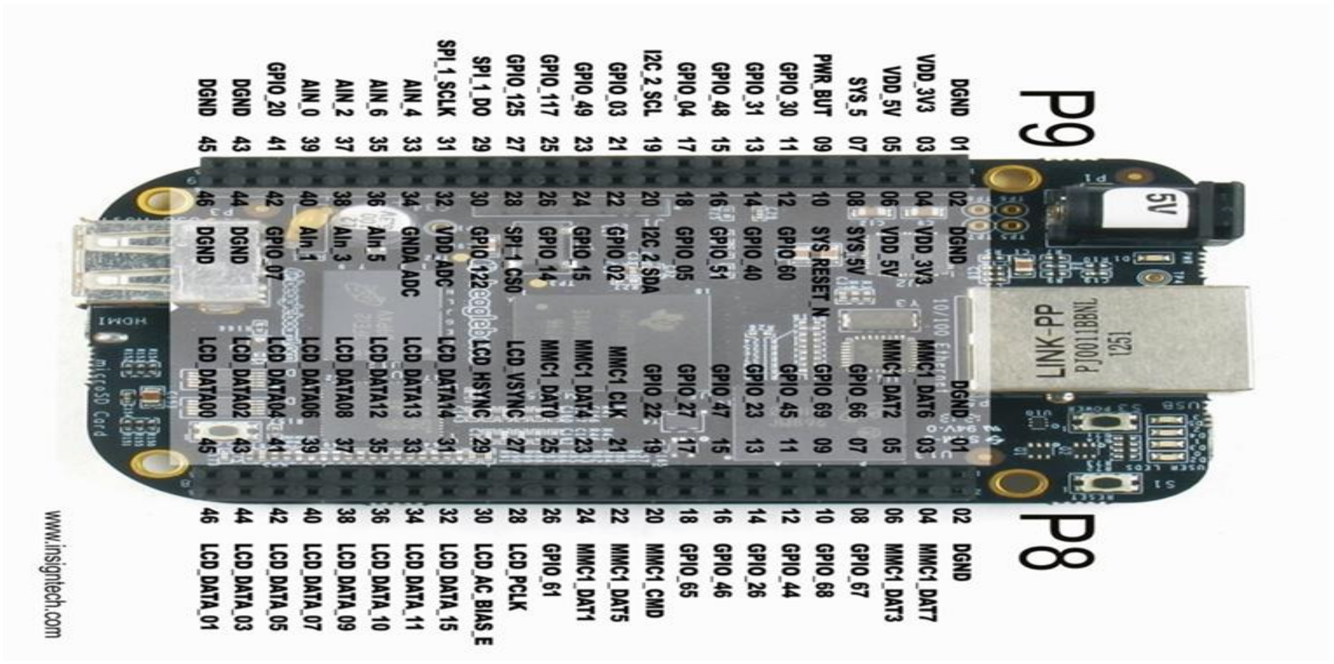
The best feature of BBB which makes is it a complete game changer is the feature to add different capes to it. Capes are plug-in boards which are added to BBB to enhance its functionality. Capes are available for motor control, VGA, camera, LCD, and other functionalities.

The BBB can be used when you need to run heavy operating systems with low power. There are many scenarios during a DIY project when Arduino is not enough. For example, during the boot of operating system and running heavy software’s, Arduino will require extra power. Here, Black comes in handy and does the same operation with low power.
When your project needs a lot of hardware to connect.

In terms of GPIO connectivity, the BBB knocks out Raspberry Pi. In Pi, we have a single 26-pin header to be used as 8 GPIO pins or serial bus. However, in BBB, we can find two 48-socket headers using which we can connect virtually n number of I/O hardware. It also features a number of analog I/O pins to connect sensors which out-of-the-box Pi lacks.

When you want your project to start quickly BBB takes very less time to get up and running. It comes with pre-installed LINUX distribution which saves a lot of time and prevents fuss.

As mentioned earlier also, that getting started with BBB is a very quick and easy process.
First, plug-in it into your computer using the included mini USB. This will power it up and boot into its LINUX distribution , Angstrom Hook it up to a display and USB peripherals you can install driver to connect BBB to web browser and control it with your computer
From here, there are no limits. You can get acquainted with the LINUX operating system or write custom software for BBB using Python and libraries to manage all GPIOs.
Applications: Robotics, Motor controller, Controlling and monitoring using Display cape, Automation
IOT, AWS, Bluetooth connectivity projects.

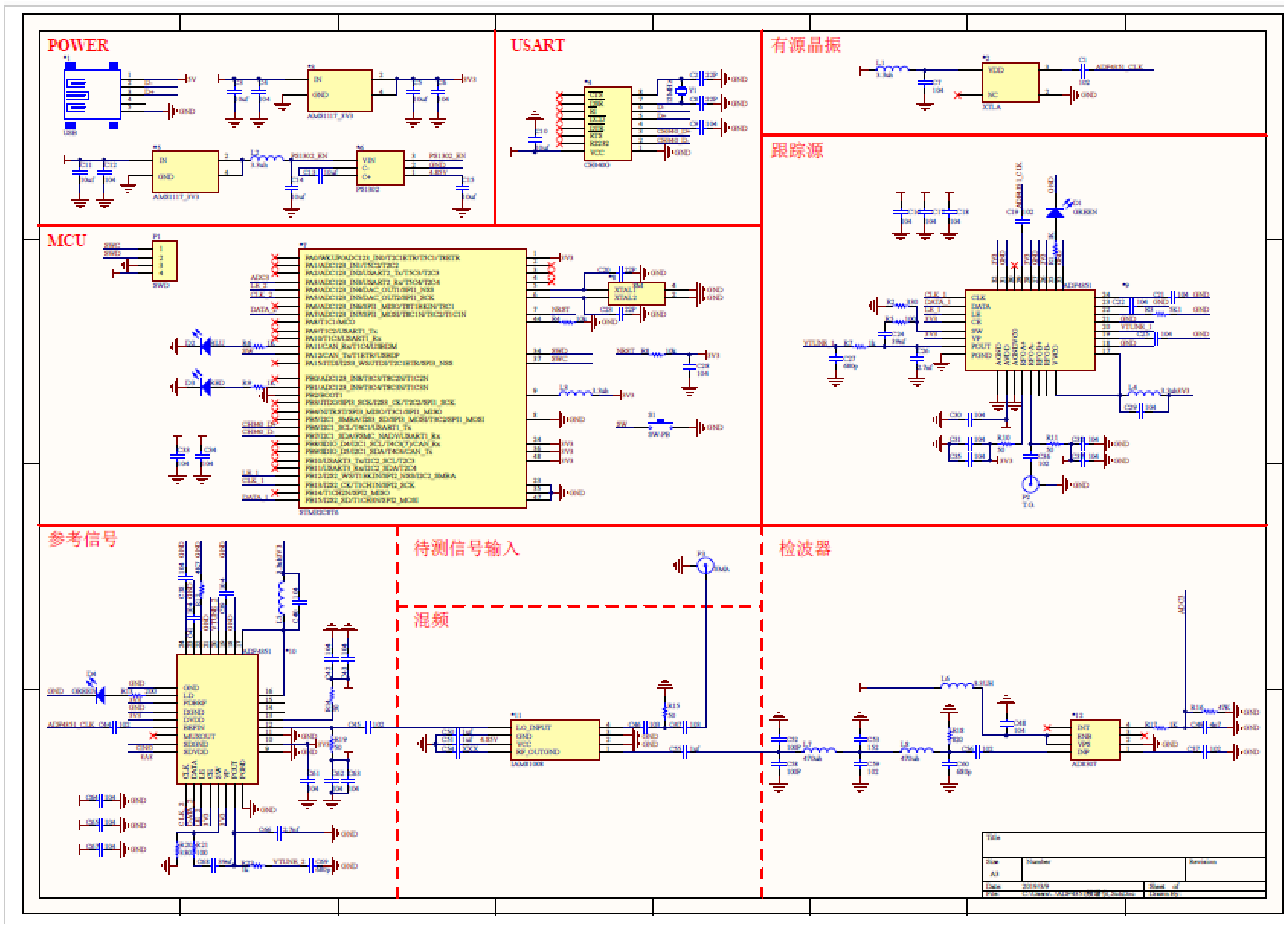


I used the BBB to program the SA via serial interface or UART, then I use it to program and configure the LoRa device.

Specification of LTDZ

This LTDZ Made of fine quality material has a long service life and Small and portable, it is an RF frequency domain analysis tool, it has a Sweep Bandwidth: 33 MHz---4400 MHz, and Turn on the tracking source output bandwidth 33 MHz---3000 MHz.

The sweep Step is: ≥ 33~68.75 mHz/125Hz, 68.75~137.5mHz/250Hz, 137.5~275mHz/500Hz, 275~550mHz/1kHz, 550~1100mHz/2kHz, 1100~ 2200mHz/4kHz, 2200~4400mHz/8kHz.
The software used on the computer side is NWT4.11.09 version for windows.
The Interface and the power supply is via USB and the Standby Current: ≤100mA, while the Sweep Current: ≤350mA ,
The Sweep Speed is more than 800 points/sec
Sweep Dynamic log ratio is more than 50dB
Point Frequency or sweep output power: ≈0 dBm
Input Detection: ≤10dBm
Background: ≈ -60dBm
The figure below shows the schematic of Device LTDZ



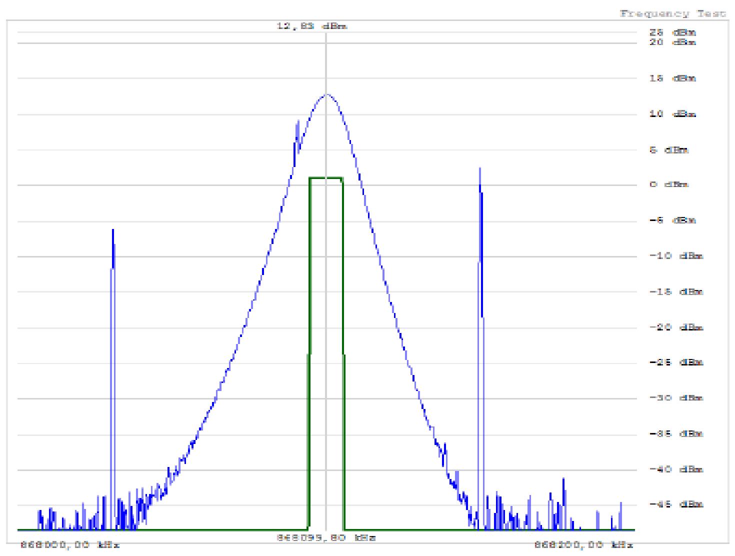
APPENDEX C - the specification of LoRa device

RF test report

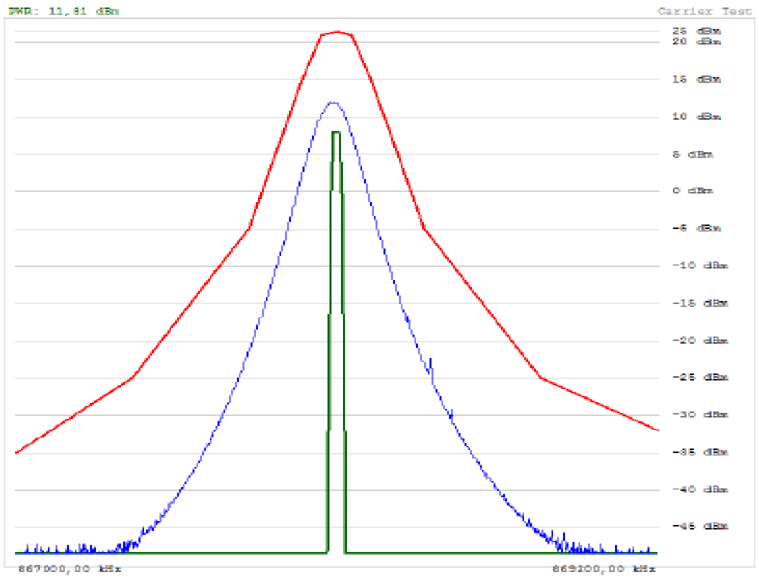
Date: 07/12/2020
Time: 14:32:50
Embit UID: 87C4B00E51514151
IEEE Address: 001BC50670115A76

```
Frequency                                 [PASSED]
Carrier                                   [PASSED]
2nd Harmonic                              [PASSED]
3rd Harmonic                              [PASSED]
Fullspan                                  [SKIPPED]
```

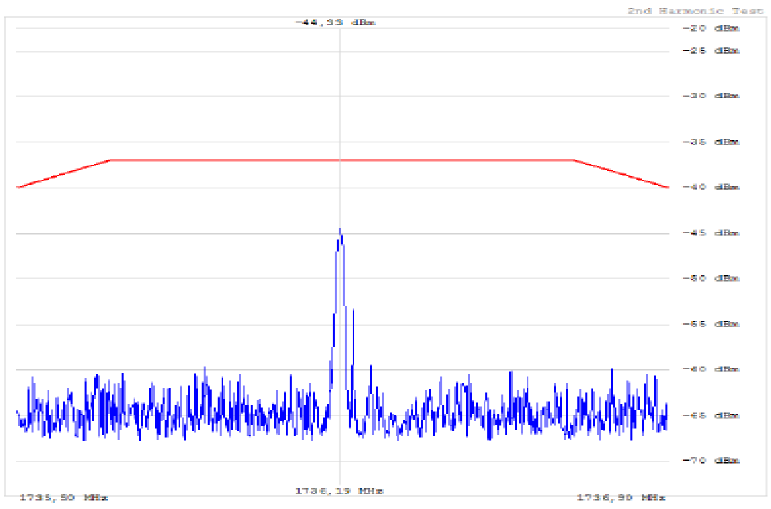
Frequency test screenshot



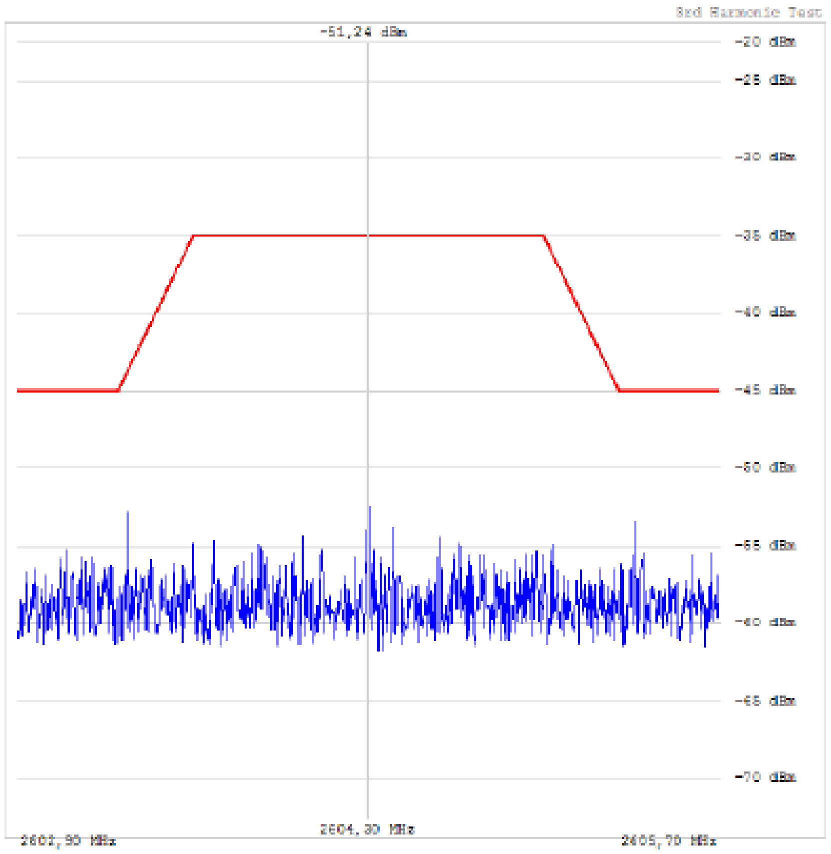
Carrier test screenshot



2nd Harmonic test screenshot



3rd Harmonic test screenshot



Appendix D calibrate the LoRa manually

```
import pickle
import getopt
import datetime
import struct
import array
import sys
import numpy as np
import binascii
import io
import serial
import os
import curses
from time import sleep
from datetime import date
from timeit import Timer
with   serial.Serial('/dev/ttyUSB0',  9600  ,bytesize=8,   parity='N'  ,  stopbits=1  ,timeout=10,
xonxoff=0, rtscts=0, dsrdtr=0) as ser:

    ser.flushInput()  #flush input buffer, discarding all its contents
    ser.flushOutput() #flush output buffer, aborting current output #and discard all that
    ser.close()
    ser.open()
    if ser.is_open:
       print("the port",ser.name,"is OPEN")
       sleep(1)
    input1=input("enter the operation you need to do  ")
    if input1 =='01':
        hex_data=[0x00,0x04,0x01,0x05]
        ser.write(hex_data)
        x=ser.read(100)
        b = binascii.b2a_hex(x)
        print ("the device information is :",b)
 elif input1=='04' :
        hex_data1=[0x00,0x04,0x04,0x08]
        ser.write(hex_data1)
        x1=ser.read(100)
        b1 = binascii.b2a_hex(x1)
        print ("the device state is :",b1)

    elif input1=='05' :
        hex_data2=[0x00,0x04,0x05,0x09]
        ser.write(hex_data2)
        x2=ser.read(100)
        b2 = binascii.b2a_hex(x2)
        print ("the device reset is :",b2)

    elif input1=='06' :
        hex_data4=[0x00,0x04,0x06,0x0A]
        ser.write(hex_data4)
        x4=ser.read(100)
        b4 = binascii.b2a_hex(x4)
        print ("the device state is :",b4)
  elif input1=='power' :
        for i in range (0,100) :
            hex_data4=[0x00,0x07,0x7E,0x02,0x01,0x0E,0x96]
            ser.write(hex_data4)
            x4=ser.read(100)
            b4  = binascii.b2a_hex(x4)
            print ("the device is transmiting 14 dBm at freq 868.1 MHZ :",b4)

    ser.flushInput()  #flush input buffer, discarding all its contents
    ser.flushOutput() #flush output buffer, aborting current output #and discard all that is
    ser.close()
```