18-747 Wireless Device Architecture
Analysis of low power audio surveillance system using LoRa

Sai Vineeth Kalluru Srinivas, Joseph Liba, Jamal Davis
Carnegie Mellon University - Silicon Valley
December 8, 2018

***Abstract*—Internet of Things involves sensors that scan the environment for data. The data is sent over a wireless channel to an access point which then processes this data and optionally saves it in the cloud. During this process, an IoT system faces multiple set of challenges including sensing, synchronization, power optimization, packet creation, channel interference, ambient noise, free space path loss, receiver decoding and many others. This project is an attempt to simulate the challenges over a use case, which is, audio surveillance. The project has been designed as a part of 18-747 course semester project. The idea of the project is to design a wireless system that senses the environment, processes the data and sends it over LoRa to a receiver. Further, the goal of the project is to compete with other teams in designing a wireless system that uses least energy for processing and transmits data to receiver in lowest time. This report discusses about the journey of our team in analyzing and developing strategies that could overcome the challenges. The team (referred as *IoTeam*) have been successful in transmitting data by consuming less than a joule of energy in the final round of competition.**
Sai Vineeth Kalluru Srinivas

I. Introduction

Sai Vineeth Kalluru Srinivas
The Internet of Things introduces four main challenges, they are: Power challenge, Programming challenge, Time challenge and the Federation challenge. The semester project is designed to cover power, programming and time challenges. A brief description of the competition is as follows. All teams will be presented with a pair of PowerDue board(manufactured in CMU-SV) one of them is used for transmitting and the other one for receiving. The transmitting board has a microphone input that would sense the audio data in the environment, saves it into a audio buffer, performs FFT and captures the two major peaks of frequencies from the FFT data. The two frequencies are then transmitted (using LoRa protocol) over a wireless channel to a receiver placed in a building approximately 240 meters away. The team that consumes the lowest energy and takes least amount of time to perform this task wins the competition.

Our team has encountered a variety of challenges in designing a low power audio surveillance system using LoRa. This report discusses each of the challenges in detail and the strategy we used for overcoming it. However, in this section we would cover each of the challenge in brief.

A. Input noise filtering challenge

The audio input to the microphone is provided with dual tone frequencies that vary over a range of 200Hz to 20Khz. However, the dual tone frequencies is added with Gaussian noise that follows a particular pattern (sawtooth). The dual tone frequencies f_1 and f_2 remain constant in the FFT plot. However the frequencies added by the Gaussian noise will keep varying on the FFT plot. The challenge is to eliminate the varying frequencies (i.e the noise) and pick only the two dominant constant frequency peaks of the FFT plot. This process is known as noise filtering. The process is a computationally intensive process and therefore challenges the low power consumption requirement.

B. Interference challenge

Once the dual tone frequency components are determined, the frequency values are translated into a LoRa packet and transmitted to the receiver board. Inherent factors like Free space path loss and antenna gain pattern would determine the ability of the packet to reach the receiver. However, this project also introduces an Interferer board which uses maximum power to send lora packets with garbage or incorrect frequency values. The challenge is to receive and decode only the packets that would correspond to the correct transmitter and eliminate the packets from the interferer. The interferer also introduces the challenge of corrupting the transmitter's data and making it indecipherable at the receiver end. Teams are encouraged to design a system that could use either 915Mhz or 920Mhz frequency bands with a bandwidth of 125Khz and overcome the challenges imposed by the interferer. The process if configured improperly can lead to increased time delay in receiving the packet which poses a challenge to lowest time criteria.

The above challenges are presented as a part of the problem statement of the project. However, the team has encountered multiple other smaller challenges in solving the bigger ones. These challenges would be covered in length and breadth during their context in the upcoming sections.

Contributions: This work is a submission comprising of a set of strategies to overcome challenges in a typical IoT environment. Our team has employed *data-driven approach* in solving the challenges. Our team was successful in transmitting data in two out of three trails in the final round of competition. The system designed by our team consumed an average of 0.6 joules of energy for performing the task. The system has taken 2.2 seconds in average to transmit the data to the receiver(including reporting the data to a TCP server). Our team was appreciated for designing a robust FFT algorithm that ensured consistency in our system performance.

II. Approach

A. Robust FFT

Joseph Liba

The two peak values of an FFT need to be calculated and sent over the server. Given an FFT, it is trivial to calculate the two peak frequencies. The MajorPeak function in the FFT library was modified in such a way that during the linear scan of the frequencies, instead of keeping track of one peak frequency, two peak frequencies were returned. Because the MajorPeak function has O(n) complexity compared to the FFT calculation function which is O(nlogn), the MajorPeak function takes very little time. Therefore, this modification does not significantly affect time or energy performance.

In the first round of the competition, there was no noise in the audio channel that would make decoding the FFT difficult. FFT could be calculated once and sent immediately. However, in the second round of the competition, significant noise was added in the audio channel. Specifically, the noise amplitude was a Sawtooth wave that decreased over time. At the peak noise levels, the two peak frequencies returned by the MajorPeak functions may not correlate to the actual desired audio signals.

The following steps were taken for the final round of the competition to deal with the noise in the channel:

1) Create a boolean array which is the same size as vReal. This will be passed into the FFT function. Note that a boolean array is the only additional array that can be created because there is a memory limitation in the PowerDue. Trying to create another double array would result in hitting a memory limit and errors.
2) Calculate the cutoff point as the average value in the FFT. If any frequency is below the threshold value, the corresponding index in the boolean array is set to false.
3) When returning the two peak frequencies in the MajorPeak function, any index whose value in the boolean array is false will be skipped. Effectively, this means that for the first iteration of the FFT, even if a noise level was over the threshold and returned as the peak value, so long as the amplitude of that noise frequency eventually randomly goes below the threshold, the boolean value for that frequency will be set to zero. Over time, the noise frequencies would have their boolean values set to zero while the actual signals would always remain above the noise threshold.
4) If for a new FFT, the threshold value is lower than an old threshold value from a previous FFT, the old threshold value should be used. This will increase the number of noise signals eliminated as the noise level decreases. Since the audio signal will not change in amplitude, it will not be eliminated.
5) The FFTs will conclude in the case that for each of the two peak frequencies, they have appeared twice in the FFT calculations.

Finally, it is important to quantify the robustness of this methodology under different noise conditions. Although there is an elimination process and a requirement to find signals twice, it is still possible to return incorrect values for the FFT. For example, if it is the case that for two FFTs in a row, the peak frequency is a noise level that always remained above the threshold, then it will be returned. Two measures will be recorded: 1) The success rate of the FFT calculation at different noise amplitudes. A success means that both signals decoded were correct. A failure means that at least one of the frequencies returned by MajorPeak was incorrect and appeared twice. 2) The number of FFT iterations it takes in order to converge on the solution. In the best case scenario, it will take two FFTs since the frequencies need to be reported twice.

1) Modify the Arduino Code to report the number of iterations the FFT took in order to converge on the correct answer. On every new set of FFT data, the boolean array and iteration counter should reset themselves.
2) Set the noise amplitude in GNU Radio to 0.0.
3) Turn on the PowerDue and allow between 10 to 20 iterations of finding the peak frequencies. For each time the peak frequencies are found, record in a spreadsheet whether the frequencies were correct and how many iterations were required to converge on the correctly answer.
4) Repeat this process for noise amplitudes 0.15, 0.25, 0.35, and 0.5.

B. Tackling Interference

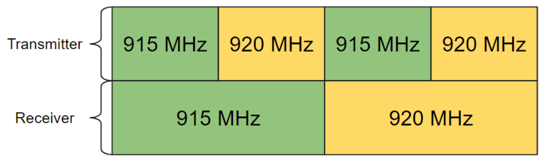
Jamal Davis

Interference was probably the biggest and most challenging design obstacle our team needed to overcome. Our team employed two strategies to help counter the effects of interference in the channel.

The first method used a timing based approach. The timing of when a packet is sent in relation to when an interferer is transmitting is a key factor in determining if that packet will successfully make it to the receiver. If transmitting same time as interferer, then it is very possible that the packet will be corrupted. In the cases we observed, when packets were corrupted, they were done so in an "All-or-Nothing" manner. Meaning we either got the entire packet or we received no usable data at all. Because of this fact, we decided against sending larger packets that contained redundant data in favor of using small packets that could be sent in bursts. We placed a focus on ensuring that a least some number of our packets are transmitted while the interferers arenâĂŹt transmitting. We also knew the rates at which the interferers were transmitting (a rate of 10Hz), so it was decided to send multiple packets at a significantly faster rate to ensure that some make it though. In the final competition we settled on sending 8 packet bursts, sent at 20Hz.

Frequency hopping was used as an additional measure to combat interference. We know ahead of time the frequencies that our transmitter and interferers were allowed to transmit on, 915MHz and 920MHz. Because of the distance between the carrier frequencies and since LoRa uses a bandwidth of 125KHz for its transmissions, we knew our transmitter would only be interfered with by an interferer on the same transmission frequency. To take advantage of this, as the transmitter was sending packets, we set it up so it would periodically change frequencies back and forth between 915Mhz and 920Mhz at a rate of 10Hz. The receiver switches between 915 Mhz and 920 Mhz at a rate of 5 Hz. Ensuring that the receiver always has a chance to receive a packet at each frequency, as shown in figure 1. This scheme also doesn't require synchronization between the transmitter and receiver for when to switch frequencies, regardless of when each starts, there will always be a period where the receiver and transmitter are on the same frequency when a packet is being transmitter.

Fig. 1: Frequency hopping scheme



C. *Packet Creation*

One key design parameter for the competition was to minimize the energy consumption of the radio module. We observed in our previous labs that the energy consumption of the radio was directly proportional to amount of data being transmitted. Armed with this knowledge, we spend a good amount of effort to try and reduce the packet size as much as possible.

The FFT code that extracts the two frequencies returns the values as 4 byte floats, however the frequency range of the audio source in the competition was from 200Hz to 19KHz, and with a minimum tolerance of 5Hz. For this range, a float much more dynamic range than is actually needed, and thus it was a waste of power to send a full 4 bytes per frequency extracted. In comparison, a 2 byte short can go up 65535, which is more than enough to represent each frequency in the specified range. By casting the float results into shorts, the payload size can be brought down from 8 bytes to 4 bytes.

We further explored the possibility of reducing the payload down from 4 bytes down to 3 bytes by developing our own floating point encoding scheme. In this scheme, each frequency could be represented as a 12 bit number for a total of 24 bits for 2 frequencies, which could be packed into a 3 byte buffer for transmitting. This scheme involved using 3 bits for the exponent, 9 bits for the mantissa, and a value of 7 for the bias. Allowing it to represent frequencies as low as 128 Hz and up to 32736 Hz. Fully covering the specified frequency range. Unfortunately, the competition required that the reported frequencies have a tolerance of +/- 10 Hz of the actual frequency value, and at the upper ranges of our floating point representation there is a 32 Hz step size between bit patterns, meaning that it could not meet the tolerance requirements in all cases. Because of this fact, this method was dropped in favor of the short conversion, which resulted a payload size of 4 bytes for the frequency data.

D. *Packet Validation*

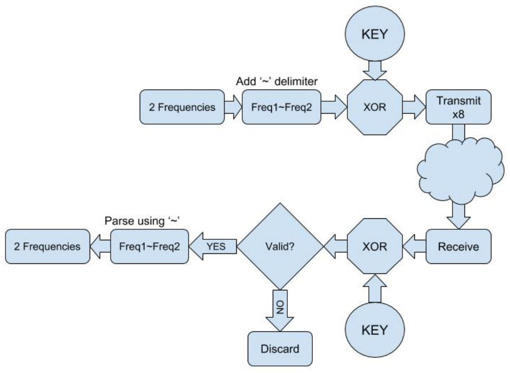
Another important challenge to overcome for the competition was being able to validate the packets that were being received, before sending the results to the server. Even if it was possible to guarantee that all packets were received without corruption, it is still extremely import to be able to distinguish the packets that were transmitted by our team, and the ones transmitted by other teams and the interferers. Not doing so would lead us to blindly accepting any packet, and potentially sending the wrong data to the server.

To validate the packets our team transmitted, we first encoded them in a specific way. On the receiving end, we would decode any received packets and check that the decoded packet was still in the expected form. If it was, we used the packet data to send to the server, and if not, then we could safely say that the packet was not sent by our team.

The process for encoding and decoding is shown in figure 2 and goes as follows:

1) Append the two extracted frequencies together using the tilde character as a delimiter, and write result into 5 byte buffer
2) XOR each byte of the buffer with its corresponding element in a 5 byte key
3) Transmit the encoded packet 8 times
4) Packets seen by receiver are XORed by the same encryption key
5) Check if decoded packet is in expected form. (5 Byte length, Delimited by tilde character)
6) If packet is in correct form, parse out the frequency data and send to the server. If not, then the packet is ignored

Fig. 2: Block diagram of packet encoding and decoding scheme



Sai Vineeth Kalluru Srinivas

After the frequencies from FFT are calculated, they are transmitted over the wireless channel to the receiver. The packet is affected by free space path loss in the wireless channel. The simplified formula for FSPL is given by equation 1.

$$FSPL = 20 * log_{10}(d) + 20 * log_{10}(f) - 147.55dB \quad (1)$$

We initially believed that making an antenna would resolve the issue of interference by an interferer. This is because an antenna would provide enough gain for the packets from tx would reach the rx with better RSSI value than the packets from interferer. The packets from tx would have an advantage over the packets from interferer. It was speculated that the better the RSSI value, the better the chances of decoding. However, we wanted to conduct an experiment to check if this hypothesis would be true.

The FSPL equation is important because it allows us to simulate the effects of building an antenna without actually building the antenna in order to determine how effective building an antenna would be.

In the FSPL equation, d stands for distance from the transmitter and f stands for the center frequency of the transmission. The problem statement already defines the value for frequency to be either 915Mhz or 920Mhz. The distance d is calculated to be 0.24km or 240meters. The distance was calculated by determining the latitude and longitude values of the transmitter and receiver positions by using Google Maps mobile application. The distance between the (lat,long) pairs are calculated using an online application [].

Let us assume the antenna would give around 2dB gain. The effective path loss that the packet goes through would be $fspl_{channel} - 2db$. The FSPL of the channel was calculated to be 73.97dB. Therefore, the effective path loss would decrease to 71.97db. In other words the antenna would give us 2dB advantage over the interferer.

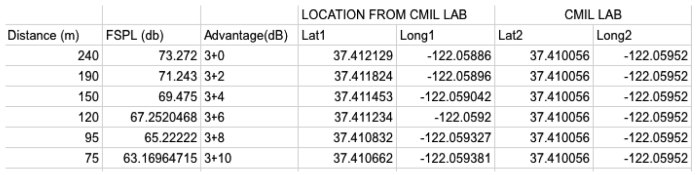
Interestingly, we can provide the same advantage by bringing the transmitter x meters closer, such that the distance $d - x$ between the transmitter and the receiver would equate to 2dB lesser path loss than the original case. For example, it was calculated that if receiver was positioned at a distance of 190m instead of 240m, it indirectly mimics the advantage of having a 2dB gain antenna.

Given that the interferer correctly interferes, then the packets sent by the transmitter should be corrupted. If 100 packets were sent by the transmitter, fewer than 100 of them would decode properly.

This idea was utilized to understand the effect of an indirect increase in gain. The steps taken for the experiment is as follows.

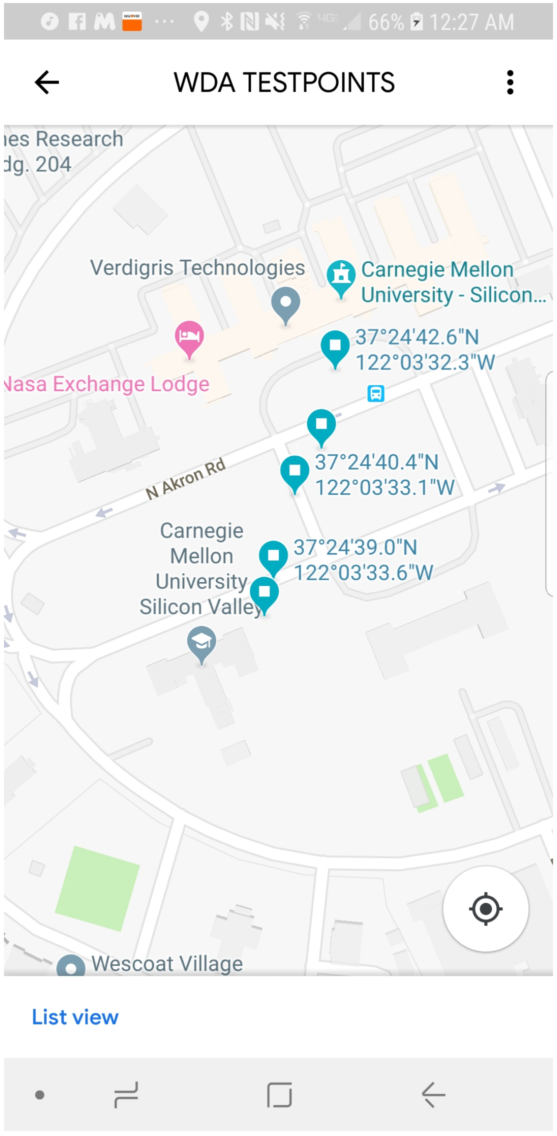
1) The receiver program was modified to receive 100 packets and validate them after decoding.
2) The program would report the percentage of valid packets received at the receiver. Any change to this percentage would indicate the advantage offered.
3) The (lat,long) pairs of the positions were calculated. These positions would correspond to xdB gain of the antenna. The range of x is 2dB to 10db at increments of two.
4) The locations were saved on Google Maps application and shared to our teammate holding the transmitter. Refer to figure 4.
5) The teammate was asked to move the position indicated by the Google Map application.
6) The percentage of the valid packets received at every position was reported by the program.

Fig. 3: The following figure shows the positions that would provide advantage over the interferer. This advantage would correlate to having an antenna of the same gain.



			LOCATION FROM CMIL LAB		CMIL LAB	
Distance (m)	FSPL (db)	Advantage(dB)	Lat1	Long1	Lat2	Long2
240	73.272	3+0	37.412129	-122.05886	37.410056	-122.05952
190	71.243	3+2	37.411824	-122.05896	37.410056	-122.05952
150	69.475	3+4	37.411453	-122.059042	37.410056	-122.05952
120	67.2520468	3+6	37.411234	-122.0592	37.410056	-122.05952
95	65.22222	3+8	37.410832	-122.059327	37.410056	-122.05952
75	63.16964715	3+10	37.410662	-122.059381	37.410056	-122.05952

Fig. 4: The following figure shows the positions indicated on the Google Maps mobile application. The locations with squares are the positions with advantage.

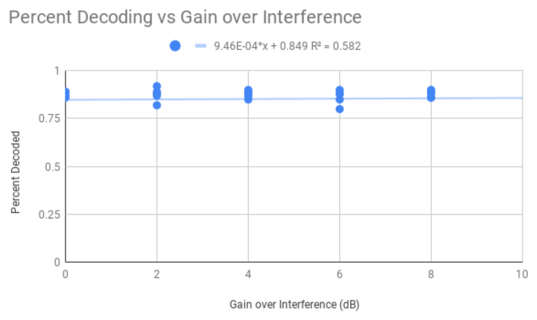

66% 12:27 AM
WDA TESTPOINTS
ies Research
Bldg. 204
Verdigris Technologies
Carnegie Mellon University - Silicon...
37°24'42.6"N 122°03'32.3"W
Nasa Exchange Lodge
N Akron Rd
37°24'40.4"N 122°03'33.1"W
Carnegie Mellon University Silicon Valley
37°24'39.0"N 122°03'33.6"W
Wescoat Village
List view

From the figure 3, CMIL lab is the fixed location of the receiver. The *distance* column corresponds to the distance of the transmitter from the receiver in the CMIL Lab. The *advantage* column is represented in $3 + xdB$ format. The constant 3dB is from the gain of the transmitting antenna that is already present. The extra x dB is the advantage offered in being closer to the receiver. In this experiment the interferer is always placed at 240 meters away from the receiver, whereas, the transmitter is moved along the calculated positions, in order to simulate the effect of antenna gain.

Before beginning with the experiment, the best case decoding and the worst case decoding percentage values were recorded. The best case decoding was observed to be 99 out of 100, which was taken when there was no interferer present. This implies 99 out of the 100 values were decoded to be *valid*. The worst case decoding was observed to be 75 out of 100, which implies only 75 percent of the packets were decoded as *valid*. Other packets were either corrupted or received from interferer.

Now, it was expected that bringing the transmitter closer to the receiver would increase the percentage decoded at the receiver. However, to our surprise we can notice from figure 5 that there is literally zero scope of improvement with increased gain at the transmitter side. The slope of the trend line is effectively zero. The advantage of having the transmitter closer to the receiver, does not help at all. The interferer is always found to interfere with the transmitter's packets. The advantage of gain and therefore, building an antenna for gain was found to be a poor solution for the current problem.

Fig. 5: The following figure shows the percentage of packets valid for every advantage in gain value.



The above experiment highlights the effectiveness of LoRa protocol. The protocol is so effective in the sense that even a very little gain in the direction of the receiver is enough for the packets to reach the receiver. Therefore, the interferer packets did always reach the receiver with same efficiency. The transmitter packets even though had more RSSI did not eliminate the effect of the interferer. This experiment and its observation was landmark to our project and saved us the time spent in making an antenna.

Therefore to tackle the problem of interference we have resorted to the concept of frequency hopping.

III. Results

A. FFT Robustness

The following are the results for the robustness of the FFT system. As stated before, the FFT model involves eliminating frequencies that fall below a particular threshold. Furthermore, the frequency values need to be found twice in order to be reported as the top two peak values. Figure 6 shows the frequency of number of iterations for each of the amplitudes. Figure 7 shows the average number of iterations for each particular noise level. Finally, Figure 8 shows the successful percent decoding versus the noise level. If at least one of the reported frequencies was incorrect, then it is recorded as a failure. However, if the both of the reported frequencies are correct, then a success is reported. The success rate is the number of successes divided by the number of frequency calculations.

Fig. 6: The following plot shows a histogram, where for each of the amplitudes for noise level, the height of the bar involves the frequency for the number of iterations. For example, for a noise level of 0.0, 2 iterations was the most frequent occurrence, which is the best case scenario.

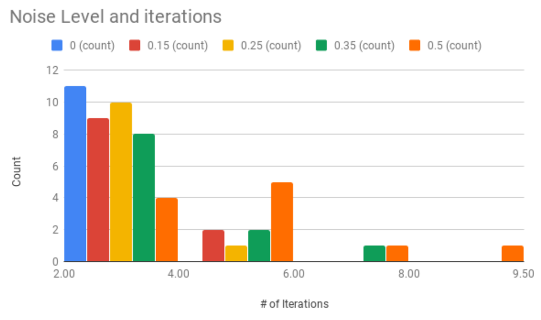


Fig. 7: The following plot shows the average number of iterations for each particular noise level. As the noise level increases, the average number of iterations also increases.

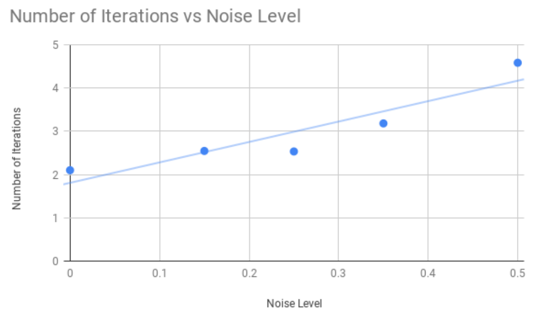
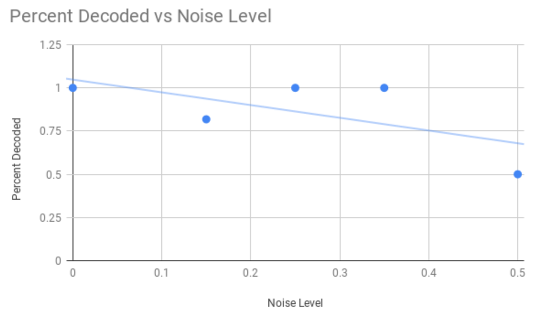


Fig. 8: The following chart shows the percent decoding versus the noise level. As the noise level increases, the success rate at decoding the proper frequencies decreases.



B. Energy Consumption

Many aspect of energy consumption were measured. Two baseline measurements were the energy consumption of a single FFT calculation and a energy for transmission at different TX Powers. Figure 9 shows an output from the PowerDue energy plot. The fourth channel shows the magnitude of the power consumption in the processor channel, while the third channel shows the start and end of different FFT functions. Overall, the FFT compute time was and the energy consumption was .777s. The energy consumption by a single FFT calculation was 115 mJ.

The different TX Powers can be seen in Figure 10. According to this plot, a single transmission of 5 bytes at TX_power of 6 requires 31 ms and 5.7 mJ of energy. According to Figure 11, a single transmission of 5 bytes at TX_power of 20 requires 32 ms and 11.2 mJ.

Fig. 9: The vertical bars in the third channel indicate the start and end of different FFT functions. The gap between 0.5 s and 0.75s represents the Windowing function. The gap between 0.78 s and 1.2 s represents the Compute function. The gap between 1.22 s and 1.35 s represents the ComplexToMagnitude function.

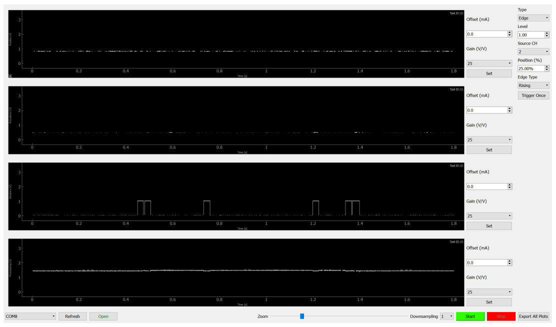


Fig. 10: The following is an energy plot for a single transmission of 5 bytes at TX_Power of 6. The time was 31 ms and the energy was 5.7 mJ.

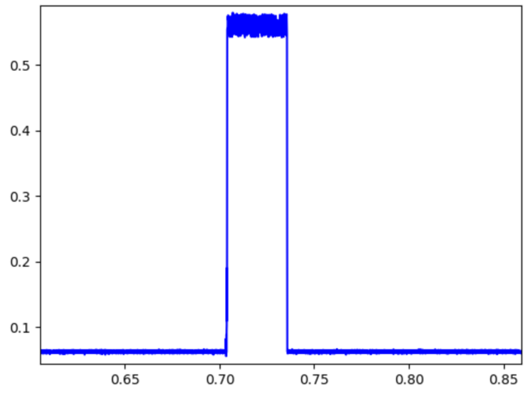
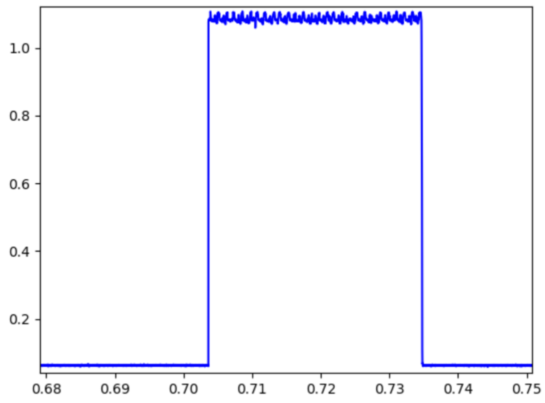


Fig. 11: The following is an energy plot for a single transmission of 5 bytes at TX_Power of 6. The time was 32 ms and the energy was 11.2 mJ.



IV. Analysis

A. FFT Analysis

The first quality to be noticed for the FFT is its robustness. This can be identified by looking at the successful percent decoding for each of the noise levels. The worst case scenario is for a noise level of 0.5. This indicates a 50% success rate. This means that if the Sawtooth wave is its the peak when the FFT is calculated, then it is likely that the reported frequencies will be incorrect. This is not an issue, however, because in the next FFT iteration, the sawtooth wave will be at a lower level. As noted by the chart, all of the lower noise levels, with exception to the 0.1 noise level, will have a 100% successful decoding rate.

If the noise was a constant source at 0.5 magnitude, then the robustness would fail. However, because the actual scenario for the competition is a sawtooth wave, then this scheme is successful for providing robustness. This correlates well with the actual results of the competition, where the reported frequencies for all three trials were correct, even if the final trial did not actually send the frequencies.

Because the accuracy for the noise levels of 0.5 is only 50%, to provide better robustness at high noise levels, one must increase the required number of repeat frequencies before reporting the actual frequency value. For example, given a constant noise amplitude at 0.5, it may be preferable to wait for 4 matches of frequencies before reporting the actual value.

The next portion of analysis is the histogram that indicates the most frequent number of iterations required for each noise amplitude. At a zero noise level, the best case was always reported, which is only requiring two FFT calculations. The 0.15 noise level was mostly optimal. However, in some cases, 4 iterations of FFT calculations were required. This is the same case for the 0.25 noise level. As the noise level increased to 0.35 and 0.5, the number of iterations required drastically increased. Many cases of the 0.35 noise level were in the 3-4 range for number of iterations, but there were a significant amount of cases where the number of iterations went as high as 5 and 7. For the 0.5 noise level, one of the cases even required 9 iterations.

What is apparent from the histogram that is not apparent from the scatter plot of iterations versus noise level is the variance for the number of iterations. The variance of the number of iterations for each noise level was very high as the noise level increased. Also, the worst case scenario is reported.

This would be important if someone was concerned with the worst case scenario, such as only running three trials. If a worst case scenario occurred in any of the three trials, this would provide a significant detriment to the score. However, if a large number of trials were run, and the average time across all the trials was considered, then only the average case plot would be important.

B. Frequency hopping Analysis

To verify that using frequency hopping would help overcome channel interference and allow more packets to make it to the receiver, we started first by testing the assumption that an interferer transmitting on a different frequency than our transmitter would not be able to corrupt our packets. We tested this by setting up an interferer running at 915MHz, with our transmitter running at 920MHz. In this case we saw 100% of our packets reach the receiver, which was promising as the theory held up.

We then setup a new experiment with our transmitter and receiver running in the frequency hopping scheme described in the approach section, and the interferer continuing to run at 915MHz. With this setup we received an average of 6 out of 8 packets. Prior to using frequency hopping, we were receiving 4 out of 8 packets on average. Showing that frequency hopping gave us about a 25% increase in packet success rate on average, which was a substantial improvement, and required very little overhead to implement. I just required a few calls to the setFrequency() function, which executed very quickly and thus used very little extra power or time.

C. Energy consumption Analysis

Since it appeared that the TX_power of 6 performs equally well as a TX_power of 20 in terms of robustness due to higher RSSI having no effect on decoding, we decided to only use a TX_power of 6 for our transmissions. With our protocol, the energy consumption of the processor was 115 mJ * 2 iterations = 330 mJ in the best / average case scenario. Since we sent our packets 8 times at a TX_power of 6, our energy consumption was 5.7 * 8 mJ which is 45.6 mJ. According to this, the percent of the radio energy to the processor energy for the calculation itself is only 12.1%. Furthermore, a single radio transmission is only 1.7% of the total energy cost. In the case of a noisy audio signal, the relative proportion of the radio energy to the processor energy vastly decreases due to the greater number of iterations required to converge on a value. Since an FFT calculation requires so much energy, this algorithm is processor dominant.

V. Conclusion

We can come to four key conclusions on different aspects of the project. These conclusions will be on the robustness on the FFT, the effectiveness of an antenna, the results of frequency hopping, and overall the energy consumption analysis.

The first conclusion about the FFT analysis shows that it is critical to know the channel conditions for the audio signal and the degree of reliability required for the FFT calculation. For this particular competition using a sawtooth wave, because the noise level decreased over time, generally only two FFT calculations were needed in order to secure a correct result. The FFT decoding plots showed that generally, with the exception for the worst case noise level, successful decoding in only 2 FFT calculations was generally the common case. Although the number of FFT calculations could increase at higher noise levels, the frequency of their occurrence was not as high as to cause worry for this particular competition.

However, what is critical to know is that in other situations, it may be more important to consider those worst case scenarios, where the number of FFT iterations is much higher and the channel could be significantly more noisy. Due to the high variance in the FFT calculations, in a time critical scenario such as a gunshot, it may be more important to seriously consider those cases where it may take a long time for the FFT to converge. Even if on average, the FFTs converge quickly, understanding the worst case scenario could be of critical importance depending on the system.

Furthermore, we believe that our system was generally robust in terms of accuracy. However, the accuracy had a sharp dropoff at the noise level of 0.5. In the case that the noise level of 0.5 could be expected to last longer, it would be more advantageous to require a larger number of frequency matches before reporting the two frequencies. Otherwise, you are sacrificing robustness for the sake of performance. In general however, the plots of percent decoding versus noise level and number of iterations versus noise level provide adequate guidance for people designing systems to determine how robust they would like their system to be and type of tradeoffs that are reasonable.

The second conclusion to be made is regards to the effectiveness of the antenna. It is clear that within this particular competition, an antenna does not have any significant effect. In fact, what appeared to have a larger effect was the timing of information. Due to the length of time that LoRa takes in order to encode and decode packets, the time scale is significant enough that two packets can directly interfere with each other. What is not necessarily clear, however, is why ever a small interfering noise signal is able to interfere with the transmitting signal no matter the gain. This points both to the effectiveness of LoRa and its downsides in a large scale deployment of LoRa devices that communicate over an ALOHA protocol.

It is likely that given the low values for the FSPL and the same location of the transmitter and interferer on the same line of transmission that the distance was not great enough to merit the effectiveness of the antenna. Given a large competition distance along with freedom to modify the polarization of signals and null out the interferer in a particular direction, an antenna could be of use by entirely nulling out the interferer.

What is clear, however, is the time and productivity effectiveness of using a simulated antenna over building a physical one. Compared to other teams that invested a large portion of their time and resources into building an actual antenna, by running analysis of the performance first, a significant portion of time was saved. The overall experiment only took a few hours and zero dollars. This provides a lesson to IoT designers to first perform an analysis of the expected improvement that can be had from a particular design choice before going along with the actual design choice. Even if the simulation does not perfectly reflect reality, by testing the best and worst case scenarios in the simulation, high confidence can be had in the results.

A third conclusion is the effectiveness of frequency hopping. Given that there was a substantial improvement in the decoding once frequency hopping was applied, frequency hopping is truly an advantageous strategy. There is a low cost and high reward in our implementation. Changing the frequency only requires writing to a register. Because the transmitter frequency was switched at twice the rate of the receiving frequency, there was a mathematical guarantee that there would be alignment of the frequencies every few trials. No real synchronization is required. In the competition deployment, this is effective because there are only 2 frequencies to hop between. However, in a real deployment, this strategy could falter since it relies on randomly aligning the frequencies in a particular window. As the number of frequencies to hop between increases, the lower the probability that there will be a random overlap of frequencies for the transmitter and the receiver. Furthermore, future analysis could explore what happens if the receiver antenna switches frequencies while a packet is being decoded.

The fourth and final conclusion is on the energy analysis. One thing to note is that it is important to be able to classify the time and energy for not only the entire FFT function, but also the different functions for the FFT calculation. By classifying the Windowing, Compute, ComplexToMagnitude, and MajorPeak functions, it becomes clear which particular functions are slowing down the calculation the most and how advantageous it is to modify a particular function. For example, the only function that we were really allowed to modify was MajorPeak due to the restriction on sampling rate. However, MajorPeak takes very little time relative to the other functions in order to compute. Therefore, it is illogical to try to optimize this function, and it is not of any worry to add an additional pass-through of the function in order to calculate the two peak values.

Another thing to note is the relative dominance of the processor to the radio module. The calculations were very processor dominant. As a result of this, sending additional packets were not of too much worry. By classifying the dominance of the system, one can make design choices where one can have freedom to create some slowdown on the less dominant module for the sake of robustness. This also creates pressure for designers of IoT to improve the performance of local computation.

With all these conclusions in mind, IoT designers can understand that given the certain constraints of the deployment environment, tradeoffs can be made. These tradeoffs can be quantified. Furthermore, as a whole, projects should be guided in a data-first approach rather than a build-first approach. This will save IoT designers time and resources.