Local Packet Detection in SDR based IoT Gateways

G Prabhu
CS18M004
IIT Madras
cs18m004@smail.iitm.ac.in

M Dileep Kumar
CS18M031
IIT Madras
cs18m031@smail.iitm.ac.in

P Ohm Prakash
CS15B030
IIT Madras
cs15b030@smail.iitm.ac.in

Vedant Somani
CS14B053
IIT Madras
cs14b053@smail.iitm.ac.in

***Abstract*—The aim of this project is to get a systems level understanding of how wireless signals are detected in real-world wireless paradigms. This is achieved by implementing a local packet detection on a low cost SDR based IoT gateway platform.**

I. Introduction

Software Defined Radios (SDRs) are devices that can be tuned to retrieve wireless signals from a particular frequency band; the idea of its invention aiming at shifting the processing paradigms of wireless signals to software rather than hardware. This provides SDRs the flexibility to retrieve any wireless signal arriving at a particular frequency, irrespective of its technology, modulation scheme, bandwidth etc.

SDRs can hence act as universal gateways operating across technologies. Past works have developed smart home gateways using the expensive USRP radio with GNU radio support to effectively decode the packets in 2.4GHz ISM bands. This implementation however failed to cope up with the latency requirements of Wi-Fi and incurred significant cost (thousands of dollars). In contrast, focusing on low-power IoT technologies that can be implemented on cheap programmable radios like RTL-SDRs provide cost based benefits with the additional advantage of scalability with the help of a cloud based backhaul.

An architecture with an inexpensive RTL-SDR, connected to a low-cost hardware board like Raspberry Pi, provided with an ethernet based cloud backhaul can hence work as a cheap, universal IoT gateway. However, even for IoT technologies which transmit with a smaller bandwidth of the order of tens to hundreds of kilohertz, retrieving all bit streams without sample loss is highly challenging. While a naive receiver would simply upload all received bit streams to the cloud, this poses immense strain on backhaul bandwidth, which is often a commodity cable backhaul at most homes.

This can be resolved by implementing a local wireless packet detection in real-time at the Raspberry Pi. This implies that rather than transmitting the entire reception over cloud backhaul, the Pi performs a signal detection locally, identifying and transmitting only the actual signal over the ethernet backhaul in real-time.

Packet detection can be performed using two methods. One is a typical energy based detection where the entire signal stream is monitored against varying power levels. The points where the power is higher is considered to be signal while the rest is discarded as noise. The more efficient method which shows better noise resilience is correlation based approach. Here the entire signal is correlated with the preamble of the wireless packet to identify the signal position (Correlation is simply the sum of products of the signal with its preamble).

II. First Deliverable

A. Aim

Implement a basic signal file retrieval over the RTL-SDR and send it to laptop. Implement a C-code that can perform energy-based detection on the file retrieved. Find the preamble of the transmitted signal and implement a correlation based detection in C.

B. Outcomes Expected

Show the performance of the two algorithms across varying levels of transmit power from the transmitter device. Compare the efficiency of the detection algorithms in terms of error in detection.

C. Preparation

First hand understanding of the signal characteristics is important to start with the signal detection. The team studied the signal characteristics using the following devices and software:

- LoRa Transmitter (SX1276)
- RTL-SDR Radio Receiver (RTL2832U)
- MATLAB 2018a

Figure 1. LoRa Transmitter Module

LoRa is the signal transmitter used in this project. The RTL-SDR device was used to receive the LoRa signal, sample the received signal and save it in a .bin file. For this, the SDR has

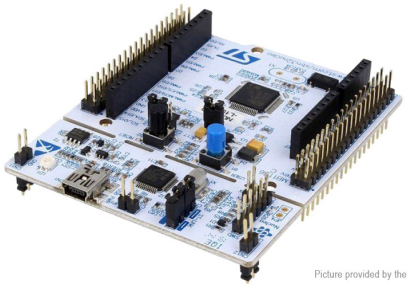
to be provided with the frequency, bandwidth of the signal and the number of samples to be taken. Sampled signal file can be plotted using MATLAB thereby enabling us to view the signal in time domain.

D. Set-Up

The transmitter device was set to transmit signal at minimal power as the detected signals were getting clipped off due to the high power. Online embedded compiler (mbed) was used to make changes to the transmitter device settings. The preamble of the signal was also changed as per project requirement. The preamble to be detected will have 14 symbols with each symbol of 2048 sample size. Out of the 14, 10 symbols will be up chirps and the last four symbols are the start delimiter. The start delimiter has 2 x Up chirps and 2 x down chirps.

Figure 2. Raspberry Pi Device

Figure 3. Nucleo Module



E. Energy Based Detection

The sampled signal data by the SDR is a raw data in binary format which has to be read and converted into complex number format by reading 8bits at a time. First eight bits represent the real component and next eight bits represent the imaginary component of the complex number. The real numbers of the sampled data can be easily plotted in MATLAB to study the signal characteristics.

Energy based detection uses a sliding window checking of the sampled received signal to locate the desired signal. In every iteration, the average amplitude of all samples of the window is maintained and the same is compared with the next window. If the ratio of two consecutive window crosses a particular threshold then it can be assumed that the signal is present in that particular window. Setting a threshold for this detection requires repeated testing of the various samples of the signal. Though the energy-based detection was able to detect the signal in the samples where the power of the signal is higher than the noise level, the same was prone to errors when the noise levels were high and equal to the signal power. The output of the energy-based detection is given in succeeding paras.

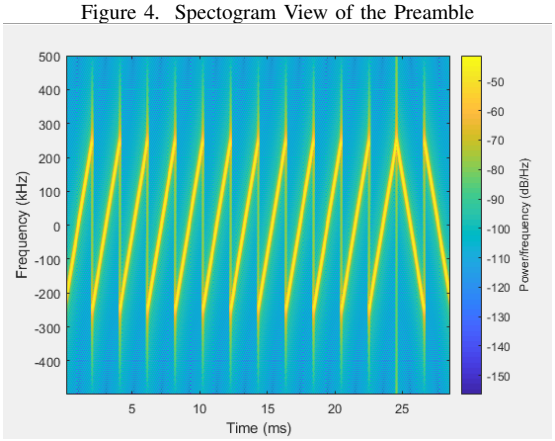
Table I
SIGNALS DETECTED BY ENERGY ALGO FOR VARIOUS WINDOW AND STEP SIZES

S.No	Actual No of Signal	Window Size	Stride size	No of Signals detected
1	3	512	512	3
2	3	512	256	3
3	3	512	128	3
4	3	512	64	3
5	3	512	32	3
6	3	512	16	3
7	3	512	1	2
8	3	256	64	3
9	3	128	64	3
10	3	64	64	3
11	3	64	1	3
12	3	32	32	3
13	3	16	16	3
14	3	1	1	0

F. Correlation Based Detection

A more advanced and better way to detect the signal is to correlate the original preamble of the signal with the data of the received signal. The LoRa transmitter was programmed using the mbed online compiler to transmit using a preamble of 14 symbols length with each symbol of 2048 sample size. Each preamble has 10 up chirps and a start delimiter. The start delimiter has two up chirps and two down chirps. Visualization of the preamble was seen using the spectogram package in Matlab. The same was compared with the original signal to identify the difference. Screenshot of spectogram view of the preamble is given below.

Figure 4. Spectogram View of the Preamble



The received is signal is correlated with the original preamble of the signal to detect the starting point of the signal. Whenever a preamble is detected then the following sequence of data before the next preamble is the data. Here a window size of the size of the preamble (2048 x 48 symbols) was used. For every iteration the correlation value (sum of products of both the signal and the preamble values) is calculated and the ratio is checked for a threshold. If it crosses the threshold then signal starting point is identified. Otherwise the window is shifted by one sample to repeat the same procedure. The received signal was normalized and demean was calculated to make the correlation work effectively.

G. *Efficiency of Detection Algorithms in Non-Realtime*

1. The efficiency of both energy based detection and correlation based approach was tested using various signals samples of various power outputs. Distance of the transmitter and the receiver were also altered to test both the methods. Though the energy based detection was fast enough to detect the signal in the sample were the signal power was higher than the noise. But it was facing issues in detecting signal in a noisy environment/low power signal. The correlation based approach was able to exactly identify the starting point of the signal in all the environment.

2. Correlation based approach was able to detect the exact starting point of the signal but in energy based detection the starting point was varying for various conditions. On an average the starting point offset of energy based detection was about 500-1000 samples and for correlation it was about 10-20 samples.

The details of the results of the algorithms is given below:-

Table II
SIGNAL DETECTED BY BOTH ALGORITHMS

S.No	Actual Number of Signals Present	Detected	
		Energy Based	Correlation
High SNR			
1	2	2	2
2	2	2	2
3	3	3	3
4	2	2	2
5	3	3	3
Low SNR			
1	3	1	3
2	4	2	4
3	5	2	5
4	7	3	7
5	9	5	9

Figure 5. Signals Detected in High SNR Condition

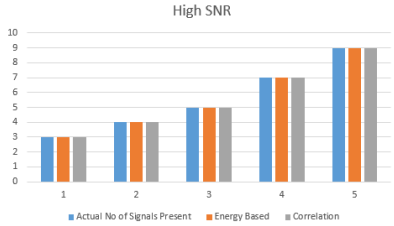
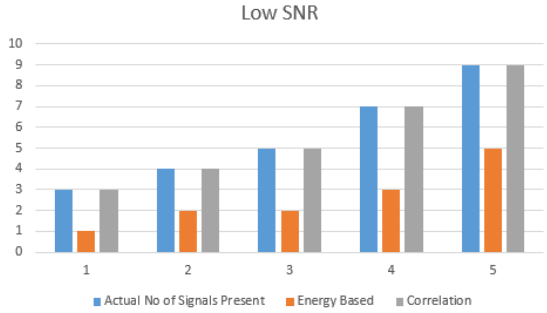


Figure 6. Signals Detected in Low SNR Condition by keeping time interval between the signals as constant (3000ms) and increasing the number of sample collected



III. SECOND DELIVERABLE

A. *Aim*

Local detection of the signal using energy based detection in the bit stream as received by the RTL-SDR and sending the signal to the cloud using socket programming.

B. *Outcomes Expected*

Show the efficiency of the energy based local detection in real time by implementing it to work in real-time and measuring the sample loss.

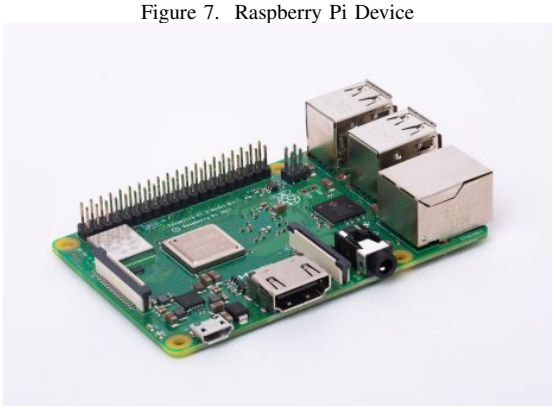
C. *Preparation*

As the first step, the Raspbian Operating system was installed in the RPi device to make it functional and able to work as a gateway device. RTL-SDR device drivers were installed and checked for functioning in the RPi device.

D. *Incorporating Energy Based Detection in RTL-SDR*

The driver source code of RTL-SDR has been modified in the next phase to incorporate the energy based detection program. For this task, the team has studied the source code of the driver to identify the specific location and variables that are used to write the detected samples into a ".bin" file.

Figure 7. Raspberry Pi Device



In the driver program, `rtl_sdr.c` is the file used to detect the signal and write it to a local file. In this program, `rtl_sdr_callback()` is the function used to read the detected samples and write it to the local file. Without modifying

the original C program, a copy of the same was created in the name of `rtl_sdr_energy.c` for the project. Energy based detection program was implemented as a function in the name of findsignal() in the `rtl_sdr_energy.c` program. Once the signal is detected by the function, the samples are read and copied into a buffer. The sample values can be then written to a file or send to a socket program to send it to the network. In the first deliverable, energy based detection program has detected starting point of the signal and reported the sample number of the same. In the real-time detection, the program has to send the entire signal data to the network. Accordingly, the data between two preamble has been taken as the signal data and the same has been copied to the buffer for further sending. Appropriate modifications were made in CMakeLists.txt file and Makefile.am files to incorporate the changes in the device driver program thereby avoiding any errors while compiling the actual program.

E. Socket Programming

RPi is a memory constrained device and the programs running in it has to be adequately optimized to get a desired output. The socket program written and tested for sending data from one computer to another has to be modified as per the requirement of RPi. Unlike other devices, where the data generated by the system is occasional, the real-time signal receiver generates voluminous data to be sent through the socket. This leads to loss in the signal data during transmission. The loss of signal while transmitting is due to the reason that the socket busy in sending the previous data when the current signal is ready to be sent. Also there is a limitation in the size of data that can be transmitted through socket at a given point of time.

A separate function was written to send the data from RPi device to the cloud. The each received signal samples data was more than 100k bytes, which was leading to loss of samples while transmission through the socket. So the "send" function was modified to split the data into chunks of 50k each and send it to the socket.

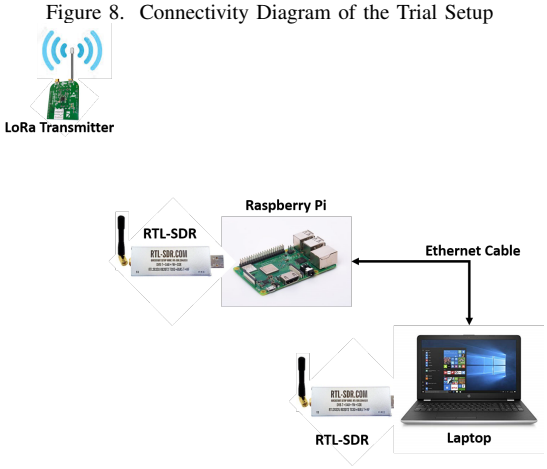
The RTL-SDR was connected to RPi device which was further connected to the laptop/cloud on Ethernet port. The signal transmitted by LoRa device was sampled and sent by the RPi to the laptop where the received signals from socket are saved in separate signal file in CSV format. A separate signal file is being created for each signal data received.

F. Efficiency of Energy Based Detection in Real-Time

The efficiency of the energy based detection in real-time was tested using an extra RTL-SDR device which was kept close to the RTL-SDR connected to the RPi device. RPi was taken control using the SSH connection from the laptop and the real-time energy based detection program was run in RPi to detect the signals. The RTL-SDR connected to the laptop was used to receive the signals into a bin file and then detect the signals from the bin file using energy based detection in non-realtime. Both the realtime and non-realtime programs were controlled from the laptop in a time synchronized manner by using "Terminator" software. This setup was done to enable verification of the signal being detected in real-time energy based detection algorithm. Two receivers are kept closeby to avoid any variation in reception due to multi-path propagation.

The block diagram of the trial setup is given below:-

Figure 8. Connectivity Diagram of the Trial Setup

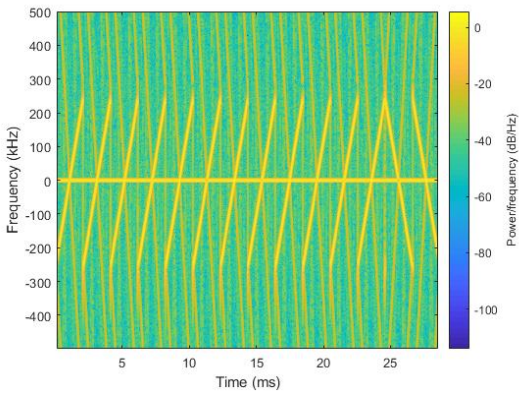


The number of signals detected by real-time and non-real-time programs were checked by changing the following parameters:-

- Varying the interval between signals being transmitted by the LoRa device.
- Increasing the running time of the programs.

Screenshot of the preamble of the received signal plotted in MATLAB using spectogram is given below:-

Figure 9. Preamble of the Received signal



The total number of signals detected by real-time and non-real-time programs were plotted against the parameters like time interval between the signal transmission and the duration of running the program. The same is produced below for reference.

Figure 10. Signals detected by varying the time interval between signals (each time the program was run for 10seconds)

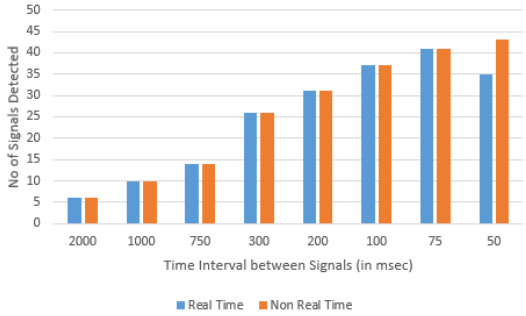
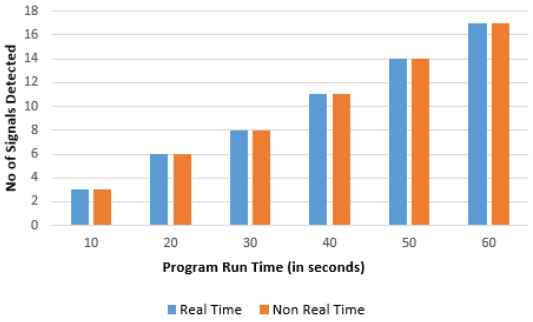


Figure 11. Signals detected by increasing the duration of the program running time (Time interval for signals is 3000msecs)



G. Observations

1. When the time interval between the signals transmitted by the LoRa device was reduced below 50ms, the real-time program was giving packet losses. This packet loss is due to the large amount of data that is being sent to the socket for sending it to the cloud. In lesser time intervals, the frequency of signals received by RTL-SDR increases thereby leading to the increase in number of signals detected per second.

2. The program was run for prolonged duration with a signal interval of 3000msec and found to be detecting and transferring the packets without any loss of packets.

IV. Third Deliverable

A. Aim

Local detection of the signal using correlation based method in the bit stream as received by the RTL-SDR and sending the signal to the cloud using socket programming.

B. Outcomes Expected

Show the efficiency of the correlation based local detection in real time by implementing the algorithm and checking for the sample loss.

C. Preparation

The device setup for real-time correlation based signal detection is similar to that of the second deliverable. Hence the procedure is not repeated in this section. Only modification is that a separate program file `rtl_sdr_correlation.c` is created for correlation based detection.

Correlation based method for signal detection involves calculating sum of products of the signal and preamble for a large number of sample values. This computation takes more time compared to the rate at which the samples are received at the SDR. To overcome this issue, two buffers were used, one large sized buffer to receive the incoming samples and other small sized buffer for processing the samples for correlation. Usage of separate buffer helped in receiving all samples and processing them correctly without missing any samples.

A buffer of length equal to half of the length of the preamble was taken and the samples are correlated for the value. If the correlated value is above a particular threshold, then the samples in the particular buffer are further correlated one by one for the entire length of the preamble to find the starting point of the signal. The threshold value was obtained after repeated trials of the sample values with the preamble values. If the correlated value is below the threshold then half of the buffer length is skipped and the procedure is continued in similar way.

D. Efficiency of Correlation Based Detection in Real-Time

Trial setup for checking efficiency of the real-time correlation based detection is same as the setup made for second deliverable. Hence the details of the same is not produced again in this section.

The number of signals detected by real-time and non-real-time programs were checked by changing the following parameters:-

- Varying the interval between signals being transmitted by the LoRa device.
- Increasing the running time of the programs.

Under low SNR conditions, both the energy based and correlation based methods were run by keeping a constant time interval of 15 seconds and varying the duration of program run. The results are shown in Fig. 14.

The total number of signals detected by real-time and non-real-time programs were plotted against the parameters like time interval between the signal transmission and the duration of running the program. The same is produced below for reference.

Figure 12. Real-Time Correlation - Signals detected by varying the time interval between signals (each time the program was run for 5 minutes)

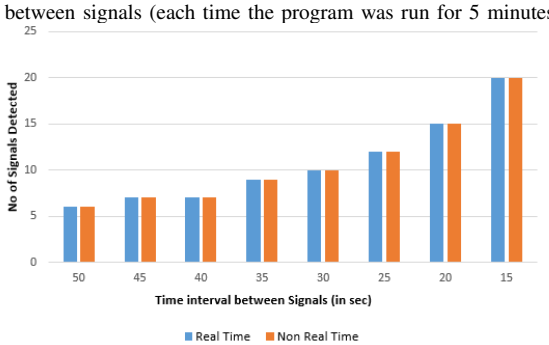


Figure 13. Real-Time Correlation - Signals detected by increasing the duration of the program running time (Time interval for signals is 15 seconds)

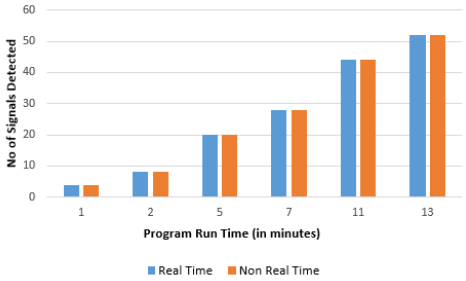
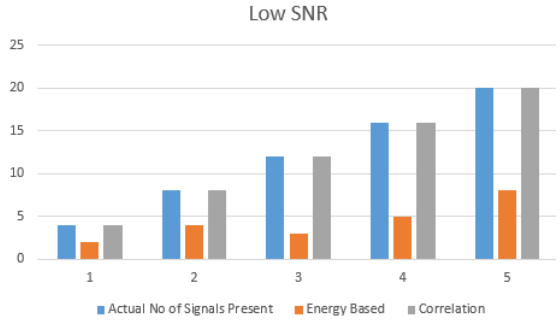


Figure 14. Real-Time (Energy and Correlation) - Signals detected in low SNR conditions, constant time interval - 15 seconds and varying program running time



E. Observations

1. When the time interval between the signals transmitted by the LoRa device was reduced below 15seconds, the real-time program was giving packet losses. This packet loss is due to the large number of calculations that is being carried out by the caorrelation program to correlate the incoming signal with the preamble. In lesser time intervals, the frequency of signals received by RTL-SDR increases thereby leading to the increase in number of signals detected per second.

2. The program was run for prolonged duration with a signal interval of 15secs and found to be detecting and transferring the packets without any loss of packets.

3. As observed earlier, under low SNR conditions, the correlation based method is more effective in correctly identifying the signals without any false positives.

V. Conclusion

Local packet detection in SDR based IoT gateways can be used to detect any signal in realtime with/without preamble of the signal. Due to the power and memory constraints of the RPi device, the gateway can be used to receive signals that carry data less frequently like wireless sensor networks.

Acknowledgment

We thank Revathy Narayanan for being a constant help and mentor through the course of this work.

References

[1] *Installing Raspbian Operating System Images* https://www.imore.com/how-get-started-using-raspberry-pi

[2] *Nagaraj, Sumeeth, et al. "Differential preamble detection in packet-based wireless networks." IEEE Transactions on Wireless Communications 8.2 (2009): 599-607.*

[3] *Modifying the LoRa transmitter features* https://os.mbed.com/handbook/mbed-Compiler

[4] *Installing and checking RTL-SDR in PC* https://inst.eecs.berkeley.edu/~ee123/sp16/rtl_sdr_install.html

[5] *Setting up RTL-SDR in RPi* https://youtube.com/watch?v=1pr319FvOwI