IEEE 802.15.4e-TSCH MAC Stack Extensions to GNU Radio

Ipsita Sanyal, TV Prabhakar
Dept. of Electronic Systems Engineering
Indian Institute of Science
Bangalore, India
ipsitasanyal@iisc.ac.in, tvprabs@iisc.ac.in

2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS) | 978-1-6654-2104-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/COMSNETS53615.2022.9668390

***Abstract*—In this work we have designed and implemented the IEEE 802.15.4e-TSCH protocol using GNU radio systems. The goal was to create a user configurable tool to test several future applications that might use deterministic MAC protocol. The main blocks that were implemented include: (a) Clear Channel Assessment using energy thresholding (b) Frequency and Channel hopping (c) Frame structure for Enhanced Beacon, Data frame and Enhanced acknowledgment. Results indicate that IEs and their transmission order are satisfied, Channel hopping and TDMA are validated, and interoperability with COTS radio systems performs well. The timeslot for Tx and Rx were implemented to remain within the standard's specification.**

***Index Terms*—Frequency hopping, IEEE 802.15.4e, IIOT, Out of Tree module, TSCH, WSAN**

I. Introduction and Motivation

Large scale deployment of interconnected sensors, actuators and other networked devices is a new paradigm to achieve high efficiency, low production costs and safety to humans and machines. This is made possible due to the affordability of sensor and wireless communication technologies and thus has benefited manufacturing and industrial processes. The Industrial Internet of things (IIoT) is realised due to the fusion of Information Technology (IT) data and Operational Technology(OT) data. For example, while tracking the usage of the equipment provides an insight into the utilization of the equipment (IT), monitoring the health of equipment ensures that the equipment is safe and operational throughout its life (OT).

The OT data is exacting in terms of time sensitivity, reliability, and deterministic operation. For example, if a sensor detects a slippage of a conveyor belt, the actuators are required to be operated in real time. Typically, a wireless network would be required to carry such anomalous sensor data and an action is required in real time. Usually, untethered operation of these sensors requires a wireless communication device. The IEEE 802.15.4 – 2006 wireless technology is piqued for its resilience and robustness to noise. However, the problem appears to be the limitation of its Media Access Control (MAC) layer in providing deterministic operation. The IEEE 802.15.4e [1] amendments were proposed in the year 2012. The IEEE 802.15.4e is continuously evolving and has multifaceted MAC behavior modes. The IEEE 802.15.4e-TSCH MAC behavior is the Time slotted channel Hopping (TSCH), has improved reliability due to its diversity in time and frequency. Owing to its characteristic frequency diversity and time slot frame structure, the TSCH MAC fortifies the drawbacks of IEEE 802.15.4 by instituting a highly scalable, reliable, deterministic and energy efficient network.

While the IEEE 802.15.4e-TSCH standard has all the necessary and required provisions, there is no proof of the entire TSCH mode performance due to lack of a full fledged implementation. The work in [2], report a limited implementation and some test results related to characteristics of IEEE 802.15.4e TSCH mode such as the CSMA-CA. Therefore, there is an urgent need for a platform to effectively prototype the standard to test application performances. Lately, the flexibility offered by open-source Software Defined Radio (SDR) platforms [3] and hardware reusability have the potential for laboratory scale prototyping as well as testing for real time performance. In this work, we implement the IEEE 802.15.4e TSCH MAC behavior mode extensions for the GNU Radio. To test our implementation, this extension to GNU radio is deployed alongside a network consisting of commercial nodes. Our goal is to demonstrate some of the key features of this standard. This work to best of our knowledge is the first implementation of IEEE 802.15.4e-TSCH mode in the GNU Radio platform.

II. Related Works

The open source implementation on IEEE 802.15.4e-TSCH mode such as Contiki [4] and openwsn [5] supports a multitude of hardware platforms. They have a robust and extensive implementation of all the communication layers supporting IEEE 802.15.4e protocol and provide APIs for input parameters. However, there are restrictions in terms of parameter settings. For example, Contiki provides 4 channel hopping sequences and has a fixed time slot parameters. Although the source file is available to users, making changes is non trivial. Also, due to lack of any thorough testing, there can be a compromise to the stability of the stack. Whereas the proposed GNU radio implementation provides the flexibility to change these parameters through the graphic user interface. Also, the flow graph debug reports are comprehensive enough when there

We thank Ministry of Electronics and Information Technology (MeitY)for sponsoring the project "Design of Dynamic MAC and PHY SoC for Low Power and Long Range Networks"- MITO03

TABLE I
EXISTING AND ENHANCED MAC LAYER FEATURES

Existing 802.15.4 flow Graph	**Enhancement to 802.15.4e TSCH Mode**
UDP socket (to connect with the applications),Rime stack (network layer), 802.15.4 compliant MAC frame (256 bytes) + 16bit CRC + OQPSK modulation in Phy layer.	UDP socket (to connect with the applications), Rime stack (network layer), MAC frame compliant to 802.15.4e std (512 bytes) including the ASN, IE's needed for different types of frames + 16bit CRC + OQPSK modulation in Phy layer.
Absence of Guaranteed Timeslot.	Frequency hopping and Time slotted multiple access.
Absence of CCA, ACK and ARQ implementation.	Energy based Clear Channel Assessment Generation and transmission of Enhanced ACK frames.
Interoperability with 802.15.4 acquiescent commercial motes.	Backward compatibility maintained with commercial motes, together with TSCH enhancements

is a rapid development of the software stack. The execution delays reported in [6] and are mostly due to hardware and software systems. The work provides a detailed study of these delays and include processing delays by the hardware, and transmit and receive of packets in the software stack. The net timeslot values encompassing the hardware and software delays specified by the standard at a sample rate of 4 MHz is 7760 μs. The delays computed in [6] is limited to a packet size of 112 bytes. If we wish to experiment and transmit a larger packet (i.e. greater than 112 bytes) then the delay increases. In this work we have implemented a 10000 μs timeslot to be able to transmit a packet size greater than 112 bytes at a sample rate of 4 MHz.

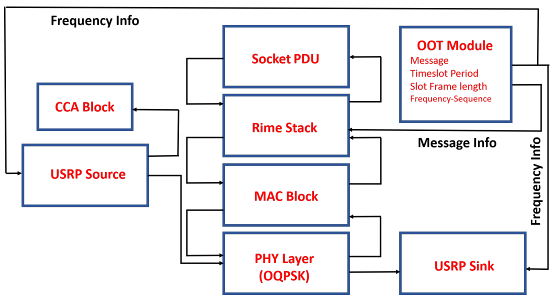


Fig. 1. GNU Radio flow graph Block Diagram

Bastian Bloessl et al [7] developed a GNU radio flow graph compliant to 802.15.4 protocol, which is interoperable with commercial 802.15.4 motes. We build on this basic flow graph and have further developed the 802.15.4e-TSCH MAC behavior. Our implementation is fully cross platform compliant with commercial IEEE 802.15.4 RF Transceiver, such as the ADF7242 [8].

III. IMPLEMENTATION

The GNU Radio flowgraph implemented in this work is in agreement with the IEEE 802.15.4e – 2020 standard [9] the block diagram representation of the flow graph is as shown in Fig 1. Our enhancements to the MAC layer of the 802.15.4e-TSCH mode reuses the UDP Socket, rime stack, Phy layer and 16-bit CRC from the existing work [10]. The first column shows that the implementation is limited to IEEE 802.15.4 with no support for beacon enabled mode. There are no interference mitigation mechanisms as well. Whereas, the novelties associated with our implementation include the ASN, IEs, frequency hopping, time slotted access, interference detection in shared slots and backward compatibility.

TABLE II
OUT OF TREE MODULE FEATURES

OOT Module Signature	**Description**
Message	The Data frame payload field. This has a maximum size of 512 bytes
Timeslot Period	The Time slot period can be specified through this parameter with a minimum value of 10ms.
Slot frame length	The slotframe length specifies the number of timeslots to be repeated periodically. The minimum value of the slotframe must be 2 timeslots.
Frequency sequence	The frequency sequence is the channel lookup table which is used for determining the sequence of channel hopping. A minimum of two channel values should be passed in the array.

There are notable improvements in our implementation. The Absolute slot number (ASN) is 5 bytes and implemented as per the standard. The ASN provides information related to frequency hopping and facilitates a new node to join an existing network. The Information Elements (IEs) for enhanced beacons (EB), data frames and acknowledgement frames are available as specified by the standard. For instance, the EB requires 7 IEs namely Global time IE, header termination IE, TSCH Synchronization IE, TSCH Slotframe & Link IE, TSCH Timeslot IE, Channel Hopping IE and payload termination IE , which are included in the EB structure deployed in the MAC stack of the flow graph. A basic energy based clear channel assessment is also implemented. Implementation of acknowledgement provides support for synchronisation. The slot frame structure is fully supported in terms of time slots and channel hopping. A highlight of our SDR based implementation is the interoperability of the standard with commercial motes. Fig.1 displays the flow graph block diagram of IEEE 802.15.4e-TSCH. The GNU radio flow graph shows the OOT modules, hierarchical blocks and the built-in modules. There are three important blocks which reflect the enhancements.

These include: (a) The MAC block (b) The OOT block and finally (c) The CCA block. In the OOT module the port generating the frequency change command is connected to the command input port of ”USRP SINK” and ”USRP SOURCE” block in the flow graph whereas the port issuing the payload is connected to the input port of the rime module.

The fundamental characteristics of the IEEE 802.15.4e-TSCH is indeed the TDMA and channel hopping. In our work, this is implemented as an Out of tree (OOT) module. The OOT module has two outputs. The first one is the frequency change command which is issued periodically at the beginning of a timeslot. The other output is the message payload which is also issued periodically at the beginning of each timeslot period. Table II lists and delineates the signatures of the OOT modules. These signatures can be initialized explicitly by the users through the graphic user interface. Thus, the system is customisable for testing application specific parameters such as slot frame structure, time slot period, choice of participating channels, data payload etc.
The Clear Channel assessment (CCA) block is a Hierarchical block which computes 1024-point FFT of the received samples followed by magnitude computation and averaging for determining the power of the received packets or the channel energy. The Enhanced Acknowledgement frame is generated complying to the standard and transmitted by the receiver when a data frame is received successfully.

IV. Results

A. Time Slot Measurements

The standard specifies timing attributes for transmitter and receiver time slots with default standard values. Timing measurements of the attributes were done for the implemented flow graph. The timing measurements of the flow graph is the average of 5000 transitions and the measurements encompasses the delays in function call and data propagation from one module to another module. The timing results are encouraging as they are well within the standard specified values.

TABLE III
Tx Time Slot Measurements

Attributes	**Standard Timings (μs)**	**Timings achieved in Gnu Radio (μs)**
macTsCCAoffset	1800	1702.96
macTsCCA	128	
macTsRxTx	192	8297.23
macTsMaxTx	4256	
macTsRxAckDelay	800	
macTsAckWait	400	
RxofAck	2424	
Total Time	10000	10000.19

Table III represents the transmitter time slot duration of each attributes. In the table the standard default values are specified in the second column and the timings as measured from the test bed is tabulated in column 3. The time taken for the clear channel assessment is 1.70296 ms, the time taken for the generation, transmission of the data frame and reception of the ACK is 8.29723 ms. Hence, the net transmitter timeslot period sums upto 10.00019 ms.

TABLE IV
Rx Time Slot Measurements

Attributes	**Standard Timings (μs)**	**Timings achieved in Gnu Radio (μs)**
macTsRxOffset	1020	8476.23
macTsRXWait	2200	
Rx of frame	1724	
macTsTxAckDelay	800	1708.46
macTsMaxAck	4256	
Total Time	10000	10184.69

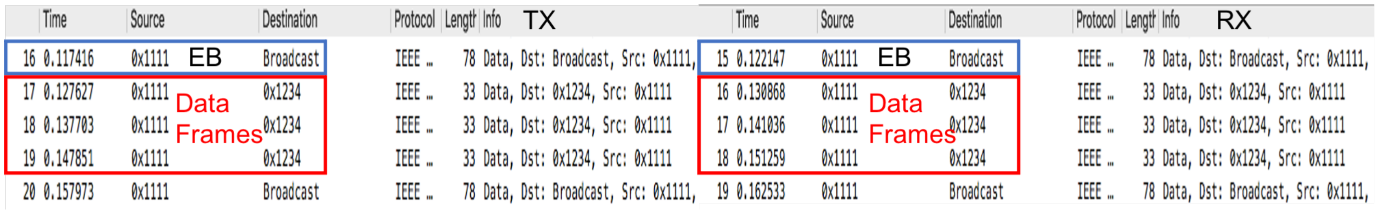
Table IV represents the timing attributes of the receiver timeslot. It is noted that the time taken for receiving and processing the data frame is 8.47623 ms. Whereas, the generation and transmission of the acknowledgment takes 1.70846 ms. The total receiver timeslot has exceeded by 0.18469ms. We think that this can be easily handled in future releases of the module.

B. IE implementation

Fig. 3 shows the 7 IEs in the broadcasted EB packet. The snapshot was captured at the receiver side using the wireshark tool. The standard expects that the IEs should be transmitted in a specific sequence and this is shown in the figure. Furthermore, the IE size is as per the standard. For example, the Global Time IE is of 4 octets.

C. Experimental Scenarios

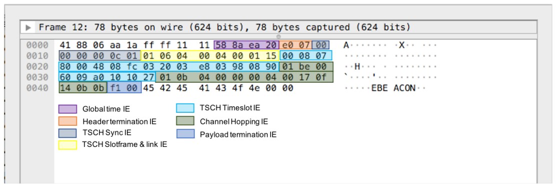
There were two experimental setups for the verification and cross-compatibility testing of the proposed implementation. In Scenario 1, the implemented OOT module is tested. In this scenario, the receiver (USRP N210) [10], and transmitter (USRP B210) [11] are connected to a high accuracy clock distribution module called the octaclock [12]. With this setup, since the receiver and transmitter are synchronised, our implementation of channel hopping was experimented. We used a 40ms slot frame structure with 4 timeslots, with the system hopping over 4 randomly chosen channels. The OOT module provides the payload, timings, and frequency change information to the USRP source and sink blocks after every time slot period.
Fig 4 depicts the change in the frequency at every time slot highlighted by the yellow box. The results indicate that channel 23, 15, 20 and 11 are chosen. The figure also shows the ASN, which increments at every 10ms time slot. Fig 2 illustrates the packet capture logs at the transmitter (left side) and the receiver side(right side). In this figure, the sequence 16 in the transmitted data packet logs is the EB frame which is broadcasted from the source with the address 0x111 (i.e B210), while in the rest of the three time slots of the slot frame structure a data frame with an appropriate payload is sent to the receiver (USRP N210) with the address 0x1234 every 10ms over different frequency channels. The other half of the figure depicts the reception of the EB followed by the data frames in the three consecutive 10ms time slots.



```
 Time         Source    Destination  Protocol Length Info
 16 0.117416  0x1111    Broadcast    IEEE ..  78 Data, Dst: Broadcast, Src: 0x1111,
 17 0.127627  0x1111    0x1234       IEEE ..  33 Data, Dst: 0x1234, Src: 0x1111
 18 0.137703  0x1111    0x1234       IEEE ..  33 Data, Dst: 0x1234, Src: 0x1111
 19 0.147851  0x1111    0x1234       IEEE ..  33 Data, Dst: 0x1234, Src: 0x1111
 20 0.157973  0x1111    Broadcast    IEEE ..  78 Data, Dst: Broadcast, Src: 0x1111,
```

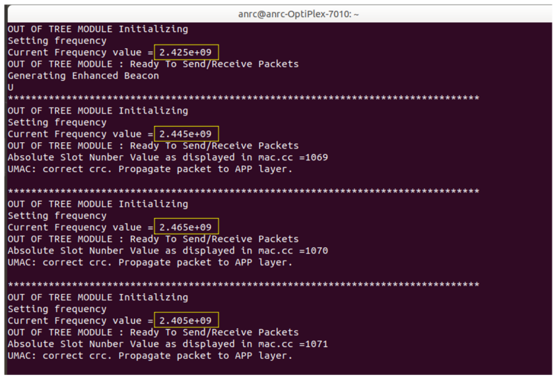
```
 Time         Source    Destination  Protocol Length Info
 15 0.122147  0x1111    Broadcast    IEEE ..  78 Data, Dst: Broadcast, Src: 0x1111,
 16 0.130868  0x1111    0x1234       IEEE ..  33 Data, Dst: 0x1234, Src: 0x1111
 17 0.141036  0x1111    0x1234       IEEE ..  33 Data, Dst: 0x1234, Src: 0x1111
 18 0.151259  0x1111    0x1234       IEEE ..  33 Data, Dst: 0x1234, Src: 0x1111
 19 0.162533  0x1111    Broadcast    IEEE ..  78 Data, Dst: Broadcast, Src: 0x1111,
```

Fig. 2. EB and data frame transmission and reception wireshark logs



```
▶ Frame 12: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
0000  41 88 06 aa 1a ff ff 11  11 58 8a ea 20 e0 07 00   A·······  ·X·· ···
0010  00 00 00 0c 01 01 06 04  00 04 00 01 15 00 08 07   ········  ········
0020  80 00 48 08 fc 03 20 03  e8 03 98 08 90 01 be 00   ··H··· ·  ········
0030  60 09 a0 10 10 27 01 0b  04 00 00 00 04 00 17 0f   `·····'··  ········
0040  14 0b 0b f1 00 45 42 45  41 43 4f 4e 00 00         ·····EBE ACON··
```

Fig. 3. Illustration of the Information Elements in the Enhanced Beacon structure



```
OUT OF TREE MODULE Initializing
Setting frequency
Current Frequency value = 2.425e+09
OUT OF TREE MODULE : Ready To Send/Receive Packets
Generating Enhanced Beacon
U
*********************************************************************************
OUT OF TREE MODULE Initializing
Setting frequency
Current Frequency value = 2.445e+09
OUT OF TREE MODULE : Ready To Send/Receive Packets
Absolute Slot Nunber Value as displayed in mac.cc =1069
UMAC: correct crc. Propagate packet to APP layer.

*********************************************************************************
OUT OF TREE MODULE Initializing
Setting frequency
Current Frequency value = 2.465e+09
OUT OF TREE MODULE : Ready To Send/Receive Packets
Absolute Slot Nunber Value as displayed in mac.cc =1070
UMAC: correct crc. Propagate packet to APP layer.

*********************************************************************************
OUT OF TREE MODULE Initializing
Setting frequency
Current Frequency value = 2.405e+09
OUT OF TREE MODULE : Ready To Send/Receive Packets
Absolute Slot Nunber Value as displayed in mac.cc =1071
UMAC: correct crc. Propagate packet to APP layer.
```

Fig. 4. Terminal log depicting channel hopping

Scenario 2 validates the cross-platform compatibility of the developed flow graph. The experimental setup consisting of a transmitter (USRP B210), receiver USRP X310) , and Commercially Off the Shelf(COTS) transceiver component, ADF7242. The ADF7242 is interfaced with a cortex M4 controller that implements the IEEE 802.15.4e-TSCH MAC. The transmitter was programmed to channel hop on channels 11 and 15 for every 10ms. The USRP based receiver and the COTS ADF7242 are set to Rx mode at channels 11 and 15, respectively. Fig. 5 has two parts: (a) This part illustrates the ASN and the payload of data frame received by ADF 7242 at channel 11 and (b) depicts the data frame received by USRP X310 [13] at channel 15. From the column that indicates the time, it can be seen that the data at each of the receivers is available after every 20ms. Hence, we may conclude the fact that the USRP B210 is channel hopping at every 10ms timeslot period.

V. Challenges

Since the IEEE 802.15.4e-TSCH standard does not specify how to implement the processes, we had to solve several challenges during the development stage. For example, for local loop test, the ASN specified variable was declared as a shared memory variable. Furthermore, we had to use polymorphic data type since they facilitate safe inter thread and inter module data transfer between the USRP sink and source modules. Moving on, the existing "QT GUI CHOOSER" block in GNU radio had to be equipped for centre frequency change at run time. We had to overcome the manual intervention and hence, our implementation of the OOT module which will change the frequency automatically and periodically during run time.

VI. Conclusion and Future Work

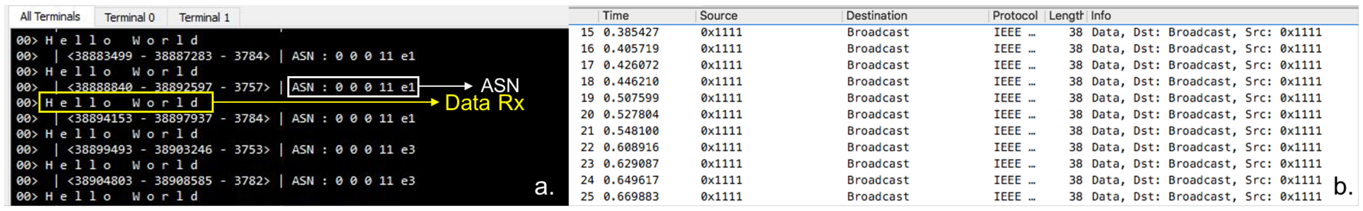
This work deploys a GNU Radio based IEEE 802.15.4e flow graph. The flow graph provides ease of re-configuration of various parametric values as mentioned in Table II with the aid of a graphical user interface. The code of the OOT module can be acquired from the Zenlabs IISc GitHub repository [14]. This implementation also allows the formation of a heterogeneous network containing COTS transceiver such as the ADF7242 and SDR systems. The monitoring and logging of the trans-receiving mechanism is facilitated by the built in wireshark module in GNU Radio. The developed system may be considered as a mandatory laboratory tool for embedded MAC stack developers. The timeslot for Tx and Rx were implemented to remain within the standard's specification. In future work we aim to use our base code towards an open source implementation of IEEE 802.15.4e-TSCH MAC behavior for industrial environment with a scheduling algorithm for EB broadcasting and data frame transfer.

Acknowledgment

We express our profound gratitude to the Ministry of Electronics and Information technology for sponsoring the project. We would like to express our sincere thanks to several colleagues from Zenlab for their valuable guidance and encouragement throughout the project development.

References

[1] "IEEE Standard for Low-Rate Wireless Networks," in IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011) , vol., no., pp.1-709, 22 April 2016, doi: 10.1109/IEEESTD.2016.7460875.
[2] Guglielmo, Domenico & Al Nahas, Beshr & Duquennoy, Simon & Voigt, Thiemo & Anastasi, Giuseppe. (2016). Analysis and experimental evaluation of IEEE 802.15.4e TSCH CSMA-CA Algorithm. IEEE Transactions on Vehicular Technology. 66. 1-1. 10.1109/TVT.2016.2553176.
[3] Popescu, Otilia & El-Tawab, Samy & Abraham, Shiny. (2017). A Mobile Platform Using Software Defined Radios for Wireless Communication Systems Experimentation. 10.18260/1-2–27482.
[4] https://github.com/contiki-ng/contiki-ng/wiki.
[5] https://team.inria.fr/eva/berkeleys-openwsn/.



```
00> H e l l o   W o r l d
00>  | <38883499 - 38887283 - 3784> | ASN : 0 0 0 11 e1
00> H e l l o   W o r l d
00>  | <38888840 - 38892597 - 3757> | ASN : 0 0 0 11 e1
00> H e l l o   W o r l d
00>  | <38894153 - 38897937 - 3784> | ASN : 0 0 0 11 e1
00> H e l l o   W o r l d
00>  | <38899493 - 38903246 - 3753> | ASN : 0 0 0 11 e3
00> H e l l o   W o r l d
00>  | <38904803 - 38908585 - 3782> | ASN : 0 0 0 11 e3
00> H e l l o   W o r l d
```

	Time	Source	Destination	Protocol	Length	Info
15	0.385427	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111
16	0.405719	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111
17	0.426072	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111
18	0.446210	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111
19	0.507599	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111
20	0.527804	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111
21	0.548100	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111
22	0.608916	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111
23	0.629087	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111
24	0.649617	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111
25	0.669883	0x1111	Broadcast	IEEE …	38	Data, Dst: Broadcast, Src: 0x1111

Fig. 5. a. Packets received by ADF 7242 in channel 11, b. Packets received by X310 in channel 15.

[6] Hazra, Saptarshi. “Timing delay characterization of GNU Radio based 802.15.4 network using LimeSDR.” (2018).
[7] Bloessl, Bastian & Leitner, Christoph & Dressler, Falko & Sommer, Christoph. (2013). A GNU Radio-based IEEE 802.15.4 Testbed. 37-40.
[8] https://www.analog.com/en/products/adf7242.html
[9] ”IEEE Standard for Low-Rate Wireless Networks,” in IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015) , vol., no., pp.1-800, 23 July 2020, doi: 10.1109/IEEESTD.2020.9144691.
[10] https://www.ettus.com/all-products/un210-kit/
[11] https://kb.ettus.com/B200/B210/B200mini/B205mini
[12] https://www.ettus.com/all-products/octoclock-g/
[13] https://files.ettus.com/manual/page_usrp_x3x0.html
[14] https://github.com/zenlabdeseiisc/GNUradio_TSCH_4e.git