EMBEDDED SYSTEM IMPLEMENTATION OF IEEE 802.15.4 FOR WIRELESS SENSOR NETWORKS (WSNs)

By

Arslan Azad

Muhammad Hamza Ehsan

Syed Arsalan Hasan

Tabish Tanveer

Submitted to the Faculty of Electrical (Telecomm) Engineering

National University of Sciences and Technology, Islamabad in partial fulfillment

For the requirements of a B.E. Degree in Electrical (Telecomm) Engineering

JULY 2018

ABSTRACT

The project aims at implementing the open source ZigBee protocol by implementing IEEE Standard 802.15.4/Zigbee on a USRP and then implementation for a small microcontroller and RF Chip based indigenous ZigBee module. It will result in surprising reduction in cost during future R&D/commercial projects. ZigBee is a communication module used widely for creating Wireless Sensor Networks due to its dramatically low power consumption (allowing battery to last for years), moderate cost and relatively secure wireless communication. However, not having ZigBee modules in Pakistan demands its import from countries like Germany which doubles the cost from 30 USD to 60 USD. This project provides a solution that not only reduces the cost to 8 USD each module, but it also provides the research and development sector in Pakistan with full control over the Stack (which costs an additional 300 USD when obtained from a foreign manufacturer) by developing ZigBee modules locally. This, as a result, opens gates for future research through possible changes in the protocol stack.

CERTIFICATE FOR CORRECTNESS AND APPROVAL

Certified that work contained in the thesis – Embedded System Implementation of IEEE Standard 802.15.4 for Wireless Sensor Networks (WSNs) carried out by Arslan Azad, M Hamza Ehsan, Syed Arsalan, and Tabish Tanveer in supervision of Lt. Col.Amer Ehsan Gilani for partial fulfilment of Degree of Bachelor of Electrical (Telecomm) Engineering is correct and approved.

Approved by

____

Lt. Col. Amer Ehsan Gilani

EE DEPARTMENT

MCS

DATED: July 5, 2018

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent.

To our parents, without whose unflinching support and cooperation, a work of this magnitude would not have been possible.

ACKNOWLEDGEMENTS

We would like to thank Allah Almighty for His incessant blessings which have been bestowed upon us. Whatever we have achieved, we owe it to Him, in totality. We are also thankful to our families for their continuous moral support which makes us what we are.

We are extremely grateful to our project supervisor Lt. Col. Amer Ehsan Gilani from MCS, who in addition to providing us with valuable technical help and guidance also provided us with moral support and encouraged us throughout the development of the project.

We are highly thankful to all of our teachers and staff of MCS who supported and guided us throughout our course work. Their knowledge, guidance and training enabled us to carry out this whole work.

Finally, we are grateful to the faculty of Electrical (Telecomm) Department of the Military College of Signals, NUST.

In the end we would like to acknowledge the support provided by all our friends, colleagues and a long list of well-wishers whose prayers and faith in us propelled us towards our goal.

Contents

Table of Figures

Abbreviations Used

API	Application Programming Interface
APS	Application Programming Sub layer
AF	Application Framework
BDB	Base Device Behavior
BPSK	Binary Phase Shift Keying
CSMA	Carrier Sense Multiple Access
CSMA-CA	Carrier Sense Multiple Access – Collision Avoidance
ED	End Device
FFD	Fully Functional Device
GTS	General Time Slot
LQI	Link Quality Indicator
MLME	MAC Layer Management Entity
MLDE	MAC Layer Data Entity
MAC	Medium Access Control
O-QPSK	Orthogonal-Quadrature Phase Shift Keying
PHY	Physical Layer
PDU	Protocol Data Unit
RF	Radio Frequency
RFD	Reduced Function Device
SoC	System on Chip
SDR	Software Defined Radio
USRP	Universal Software Radio Peripheral
WSN	Wireless Sensor Network
WPAN	Wireless Personal Area Network
ZNP	ZigBee Network Processor
ZDO	ZigBee Device Object

Chapter 1: Introduction

1.1 Overview

ZigBee is a wireless network communication protocol used widely for creating Wireless Sensor Networks due to its dramatically low power consumption, moderate cost and relatively secure wireless communication. The IEEE Standard 802.15.4 lays the basis of the ZigBee protocol stack as it defines the lower two layers, namely Physical Layer (PHY) and MAC Layer. The project aims at implementing the open source ZigBee protocol by implementing IEEE Standard 802.15.4 on a USRP in order to understand the working of stack and then proceed to implementation for a small microcontroller and an RF Chip based indigenous ZigBee module. It will result in surprising reduction in cost during future R&D/commercial projects. The available ZigBee modules in Pakistan demand import from countries like Germany which doubles its cost. This project provides a solution that not only reduces the cost, but it also provides the research and development sector in Pakistan with full control over the Stack by developing ZigBee modules from simple transceivers locally. This, as a result, opens gates for future research through possible changes in the protocol stack. In order to attain the main objectives of cost reduction and control over protocol stack for future customizations, the project is divided into two objectives including GNU Radio Simulation for a USRP and then the embedded system implementation of ZigBee Protocol Stack on suitable hardware; ideally a combination of a microcontroller and an RF Chip.

Working on the first objective, in order to get a better understanding of the protocol stack, a simulation of the IEEE Standard of 802.15.4 was done on the GNU Radio on Linux platform at first. This helped in understanding the working of MAC and Physical Layer, the way data packets flow, and the transmission over a radio link. Following the GNU Radio Simulation, a network was created using the already present ANY-900 modules present to see how the mesh network was created, how the nodes joined the network and how each sent data to the other nodes. Finally an open source ZigBee protocol stack was selected from among 5 open source stacks available, due to its compliance with hardware and online support available. The stack was debugged and ported on a compliant hardware – a System on Chip (SoC) containing a built-in microcontroller for controlling and an RF Chip for radio communication. Once the ZigBee modules were formed, a network of ZigBee nodes (Coordinator and Routers) was created, and communication between these nodes was carried out by sending data to and from the coordinator, which was received successfully over a tested range of 80 meters in open space while 10 meters in obstacle environment from one room to another. Due to customizable stack, the MAC and network layers (for security) were explored in order to make the required changes. The user will now be able to do stack hardening by changing the key before creating the network, however all the nodes must have the same key configured to them. Furthermore, the coordinator is now able to permit a node to join the network as well as bind a node to it. Numerous other customizations are possible including customization of encryption techniques, modulation schemes, and topologies etc.

1.2 Problem Statement

The major problems due to which this project was chosen was the expansive nature of ZigBee and its protocol stack, delivery issues containing the uncertainty if the imported ZigBee will even work or not, unavailability of high specs of ZigBee modules and vendors in Pakistan and no control over the protocol stack.

A single ZigBee costs around 30 USD when imported from outside. But these ZigBee's come with built-in Hex that cannot be changed. So the buyer does not have much control over the stack and hence can't make much changes to the ZigBee. The only way to get control over the stack is to buy it and it is very costly to do so. There can be issues during delivery that the imported ZigBee is not in working condition or has been damaged during delivery. This issue can be solved if the ZigBee was produced and sold locally but high specs ZigBee are neither produced nor sold locally.

1.3 Proposed Solution

The solution proposed was to create a high spec, cost-effective ZigBee over which the user would have complete control. In order to do this an open source ZigBee protocol stack would be chosen and ported in a compliant RF module. Modifications would be made in the stack so that it would become unique compared to other ZigBee running on the same protocol stack. By developing this ZigBee locally, this would reduce the cost of the ZigBee for local R&D/commercial projects. Furthermore, the users would have complete control over their stack and would be able to make changes to their ZigBee as they wish. The changes in the protocol stack would mean that not only can they customize the different layers of the stack but now they can customize the hardware as well.

1.4 Objective

The goal is to provide indigenously developed, highly customized, efficient Commercial-Off-The-Shelf (COTS) ZigBee/IEEE 802.15.4 modules for future R&D/commercial projects.

The main objective of this project is to first research and understand the MAC and Physical layer of ZigBee. To simulate the ZigBee network on USRP and understand how the MAC and Physical work, how they communicate amongst themselves and how data is sent from one layer to another. Then to implement the mesh network on available ZigBee modules present with local R&D wing to better understand how a network is created, how coordinator, router and end devices are made, what their functions are and how they communicate with each other. Then to find a suitable ZigBee protocol stack and a compliant RF module to create our own ZigBee. Making changes in the protocol stack according to our specifications and requirements. And finally to implement the stack in the RF module and to test and debug our ZigBee by creating a mesh network and having them communicate with each other.

Chapter 2: Literature Review

2.1 What is IEEE 802.15.4?

It’s a ZigBee Standard which holds ability of Radio Frequency (RF) transmissions between the devices in a wireless personal area network (WPAN). These devices are comprised of low data rate, comparatively short-range and low-power. Devices are interconnected to each to other in personal area network (PAN) and they can communicate to each other via radio communication. This protocol is capable of star and peer-to-peer (P-P) topologies. In communication protocols, it is important to avoid collisions between transmitted packets, this protocol tackles this issue by using a collision avoidance technique called Carrier Sense Multiple Access (CSMA).

2.2 General Description

This protocol is capable of many things, but a reasonable battery life, extremely low cost, reliable data transfer, ease of installation, short-range operation are one of its main objectives. It also maintains a simple and flexible protocol. Its data rate is about 250kbps. Address could be as extended as 64 bits, or it could be a short address with 16 bits. It also includes Link Quality indication (LQI). Since we are operating on 2.4 GHz, so we will have 16 channels available for data transmission, in case of 915 MHz band, we have 30 channels and if a user is operating at 868 MHz band, only 3 channels will be available.

Two types of devices are eligible for taking part in a ZigBee (IEEE 802.15.4) network, Reduced Functional (RFD) and Fully Functional (FFD). It also has an End Device (ED) but it only receives signals from Coordinators via Routers and act accordingly.

A ZigBee (IEEE 802.15.4) network is made up of of numerous components with device being the most elementary component of this network. A device could be anyone of the RFD or an FFD. If a physical channel has two or device communicating to each other, we will have Wireless personal access network (WPAN). It is necessary that every WPAN should have at least one PAN coordinator.

An FFD is capable of communication with an RFD and other FFD devices meanwhile and RFD is limited to communicate only with an FFD. For extremely simple applications an RFD is used, for example a light switch.

ZigBee network works in the form of topologies, now we will discuss all the possible topologies in which a ZigBee (IEEE 802.15.4) network can operate.

2.3 Network Topologies

An IEEE 802.15.4 network can operate in either of two topologies: the peer-to-peer or the star topology. Topology in which the communication is established between a single central controller (PAN coordinator) and devices is called Star Topology. PAN coordinator is also present in P-P topology; however, device may communicate with any other device if they are in range of one another which is not possible in star topology.

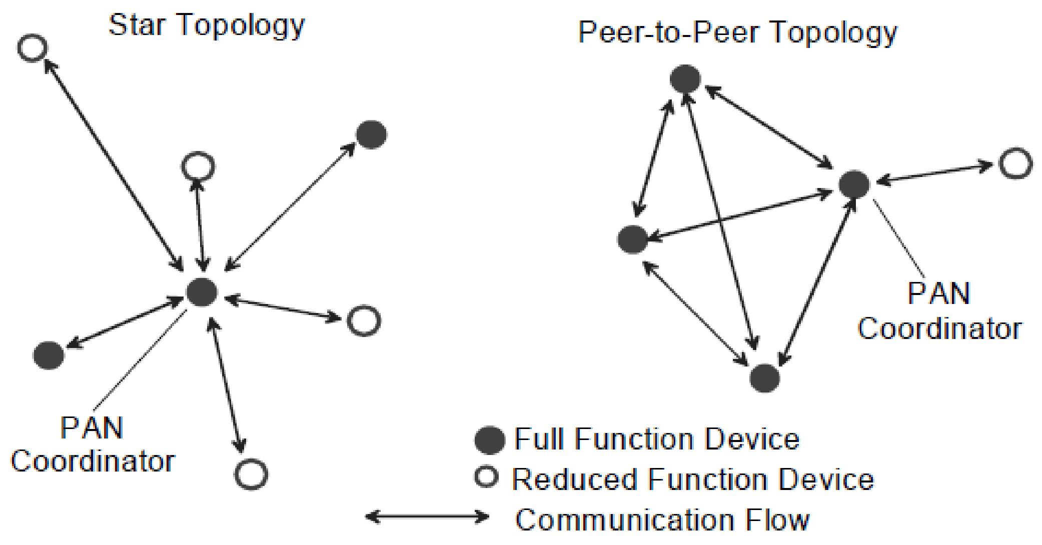


Figure 1: Star and Peer-to-Peer Topologies

2.3.1 Network Formation in Star Topology

Operating principle of star network is explained in the above Figure. Any FFD can become a PAN Coordinator and start its own network. Unlike peer-to-peer topology, two or more star networks are unable to operate together. First step in the process of creating a star topology network is choosing a Personal area Network identifier currently inactive in any other network. The network created by a Personal Area Coordinator is not allowed to join by any devices until a PAN identifier is not selected.

2.3.2 Network Formation in Peer-to-Peer Topology

In this topology, process is slightly different because first a PAN coordinator is selected, a PAN coordinator is the first device that can communicate with devices available in the network or connected to the channel. Devices (REF or FFD) can send or receive commands with any other device (RFD or FFD) within its range, which is not possible in case of star topology.

Cluster tree is very much like P-P topology. Just like a leaf is connected to the end of branch, an RFD is connected to a ZigBee (IEEE 802.15.4) network because RFDs (like leaves) do not have ability to connect with other RFDs. There are multiple FFDs in a cluster tree network but only one of these FFDs have ability to become a coordinator and a coordinator is more responsible for operations in a ZigBee network than any other device(s) in the PAN. Coordinator can control the synchronization of devices (RFDs or FFDs) in a network.

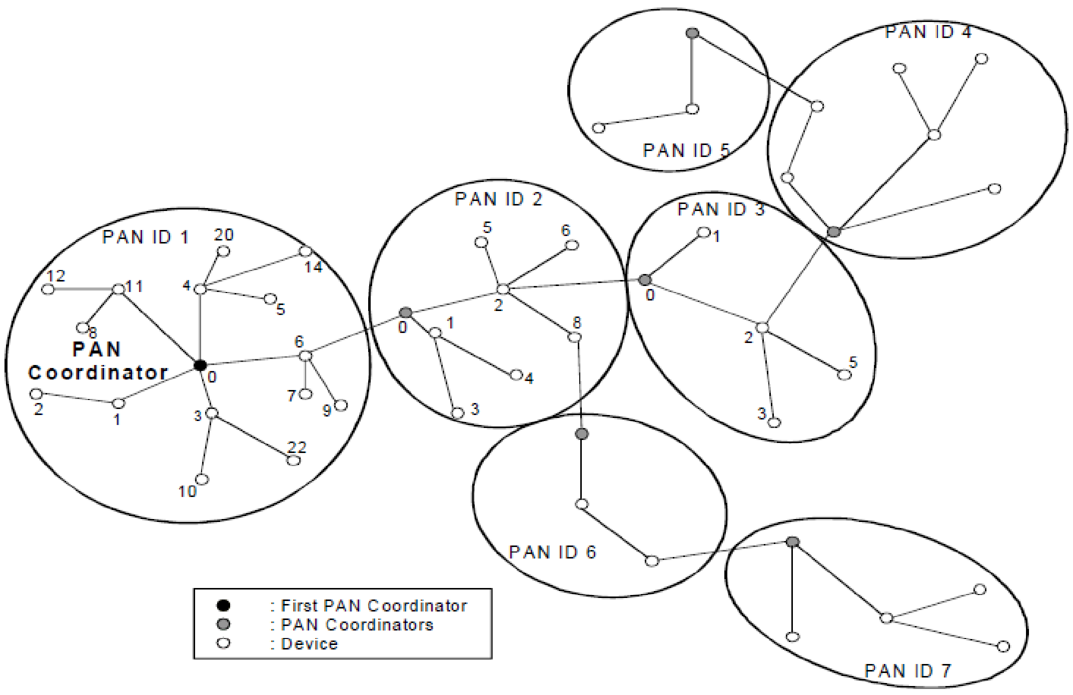


Figure 2: Cluster Tree Network

2.4 Device Structure

Network layer is the upper layer in each device of a network, which is responsible for routing of messages, manipulation, and configuration of network and an application layer is also a part of every device, which has its own responsibilities in the device functionality. Upper layers are not discussed in this standard since they are not a part of it. While working on an IEEE 802.15.4 most of the attention is emphasized on MAC and PHY layer. These layers are responsible for all the tasks in an IEEE 802.15.4 network. Both layers cooperate with each other.

2.4.1 Physical Layer

PHY layer is responsible for: Physical management and data service. Protocol data units (PDU) are transmitted and received across physical radio channel which is enabled by PHY data service.

The standard specifies 4 PHYs

- 868/915 MHz DSSS with BPSK Modulation (Mandatory) ¬ In case of 868 MHz 20 Kb/s is the data rate and in case of 915 MHz 40 Kb/s is the data rate.
- 868/915 MHz DSSS with O-QPSK Modulation (Optional) ¬ In case of 868 MHz 100 Kb/s is the data rate and in case of 915 MHz 250 Kb/s is the data rate. ¬ Signaling is like 2450 MHz band PHY
- 2450 MHz DSSS with O-QPSK Modulation

Physical Layer Specifications

PHY layer is responsible for following things

- Radio transceiver's Deactivation and Activation
- Data reception and transmission
- Use of CSMA-CA mechanism to avoid collision
- Selection of frequency for channel
- Data reception and transmission
- Link Quality Indication (LQI) for the received packets
- Transmission and reception of data
- Selection of channel frequency

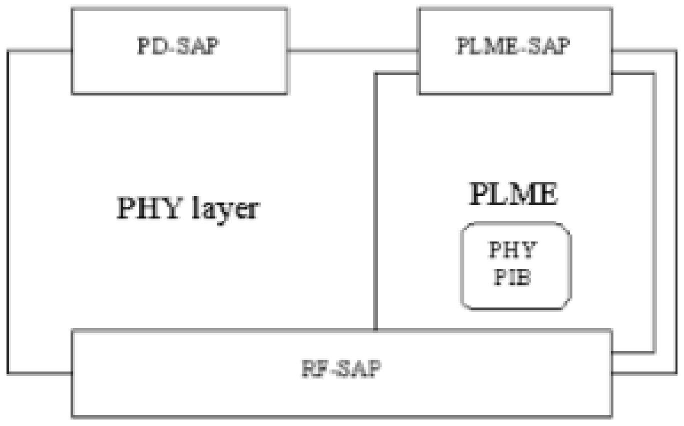


Figure 3: PHY Layer Data Service

2.4.2 MAC Layer

MAC Layer has following tasks: the MAC management and data services. Protocol data units (PDUs) travels across the PHY layer by the help of MAC data services.

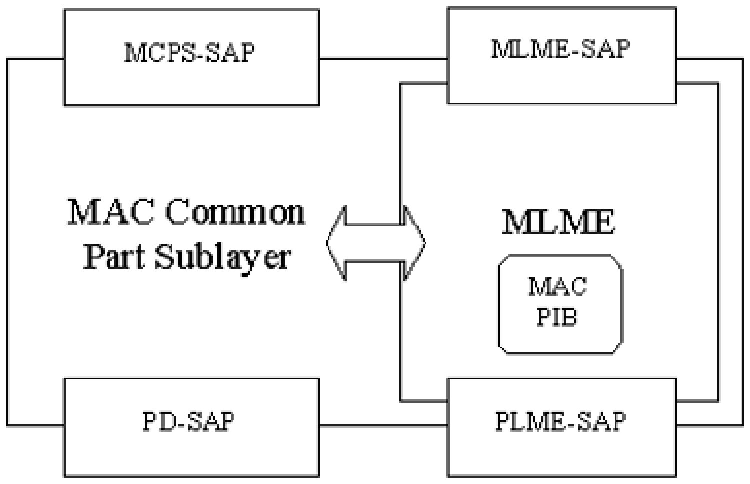


Figure 4: MAC Layer

MAC Layer Specifications

MAC layer is responsible for following things

- If device is acting as a Coordinator, it must generate network beacons
- Synchronization of generated network beacons
- Increasing the probability of robust channel access by the employment of CSMA-CA mechanism.
- MAC layer must support security of ZigBee network devices
- Supporting association and disassociation of Personal Area Network
- MAC layer is also responsible for providing a consistent connection between two MAC sub layer entities.
- Controlling and maintenance of the General Time Slot (GTS) procedure.

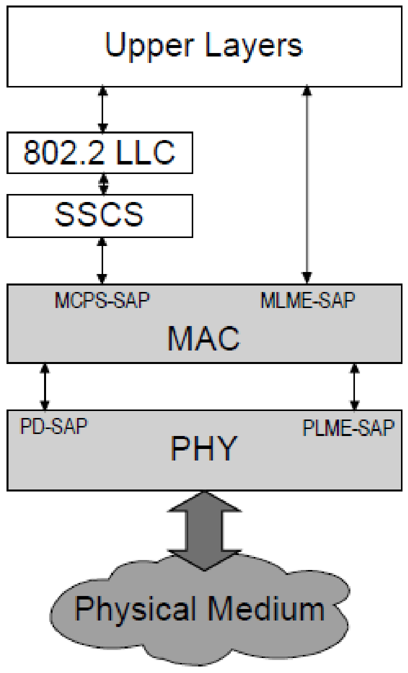


Figure 5: Architecture of IEEE 802.15.4 Layers

2.5 Functional Outline

In this part, we will study how an IEEE 802.15.4 protocol works. It also includes the information regarding structure of super frame, how data is transferred, how to improve the probability of a successful delivery, security and power consumption.

2.5.1 Superframe Structure

It is not always necessary to use of a superframe structure, this protocol provides the flexibility to use the super-frame structure only when needed. Coordinator defines the super-frame format. 16 slots of equal size are available in a super-frame structure. If needed, it is parted into active and inactive regions. During an inactive region, PAN coordinator goes into a low-power mode. First slot in each super-frame is reserved for the transmission of a beacon frame. If a super-frame structure is not needed coordinator can turn off beacon transmissions. Guaranteed time slots are available in case of applications with very low latency or if specific bandwidth is needed to run applications. Guaranteed time slots are also termed as “Contention free Period”, since it is always available.

Below is the structure of Superframe structure with Guaranteed Time slot (GTS)

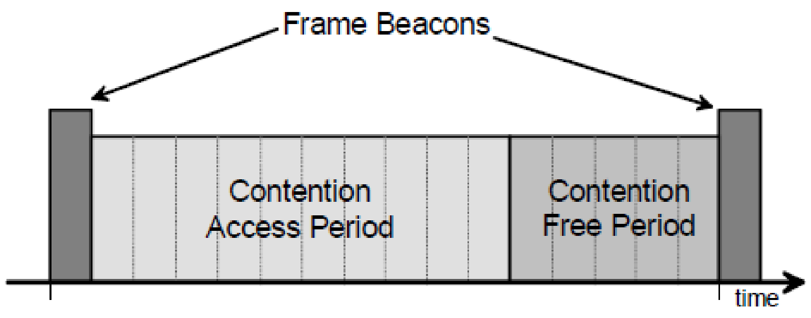


Figure 6: Super-frame with Guaranteed Time slot

Below is the structure of Super-frame structure without Guaranteed Time slot (GTS)

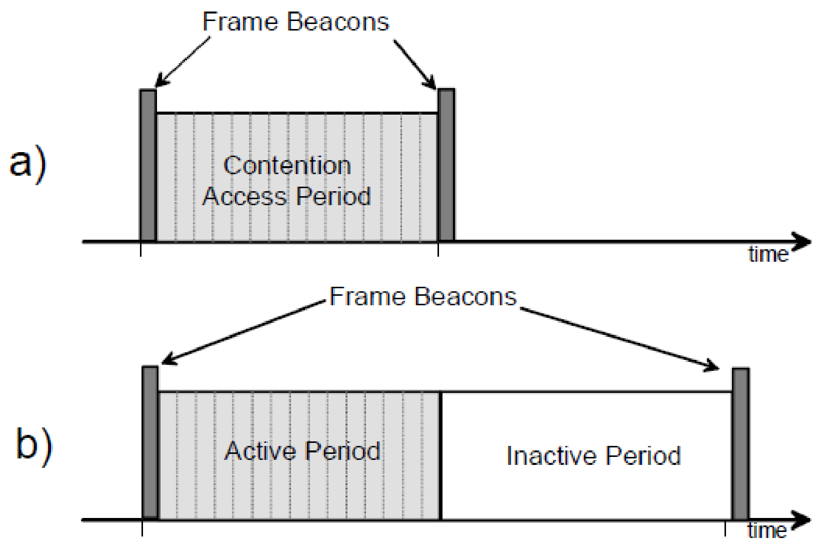


Figure 7: Super-frame without Guaranteed Time slot

2.5.2 Data Transfer Protocol

There are multiple ways in which data transmission can occur, but these three are most commonly used.

- Transmission of data from a Coordinator to other devices
- Transmission of data from other devices to a Coordinator
- Data transmission from one peer to another in a Peer-to-Peer network

However, in star topology only first and second type of transmission occurs since devices cannot communicate with devices from other star topologies.

2.5.3 Frame Structure

Frame structures are defined in such a way to decrease the complexities to a minimum level and their designs make it possible to have to robust transmission when passing through a channel with high level of noise. Frames pass through multiple layers and each layer adds their own headers and footer to the frame.

There are four kinds of frames

- **Data frame:** It is used in transmission of all type of data
- **Command Frame (MAC):** Control transfers in MAC Layer are handled by this frame
- **Beacon frame:** Used when coordinator must transfer beacons
- **Acknowledgment frame:** It is used as a confirmation in case of successful frame transmission.

2.5.4 Improving Probability of Successful Delivery

Successful data transmission is very important function. There are multiple ways in which could interrupt data transmission such as noisy channel, incorrect headers, information getting corrupt etc. IEEE 802.15.4 uses three kinds of mechanisms to improve the probability of a successful delivery and those are

- Frame Acknowledgment
- CSMA-CA mechanism (Slotted and Unslotted)
- Data verification

2.5.5 Power consumption

These devices will be powered and if they are to use a “stand alone” device, they are going to need batteries with long life spans because changing batteries after short intervals of time is practically impossible. Devices capable of operating IEEE 802.15.4 protocol are designed while keeping in view the power resource management. Whenever there is no data transmission pending, these devices go into sleep mode where they spend most of the time. They get active only when they must listen to the of radio transceiver channel if it has a new pending message. This mechanism allows the balance between message latency and battery consumption. However, High powered devices hold the capability to listen to RF channel all the time.

2.5.6 Security

While keeping in view the security, wireless networks are very much like another wireless network. Active tampering is possible because to participate in communications there is no need of physical access to the wire and passive eavesdropping attacks are also possible. Since these devices are low-cost and they have very limited available storage and computing power, it is not possible to have strong and trusted computing base. In these cases, cryptographic mechanisms are used. Symmetric key cryptography is used and higher or upper layers.

Cryptography provide these three combinations

- **Data authenticity:** It assures that data and source of data transmitted is not modified.
- **Replay protection:** It make sure the detection of duplicate information
- **Data confidentiality:** It make sure that transmitted data is available only for intended parties

Chapter 3: GNU Radio Simulation

3.1 What is USRP?

The Universal Software Radio Peripheral (USRP) makes it possible for engineers and developers to design and implement Software Defined Radio (SDR) systems which are powerful and flexible. USRP designs, when coupled with daughter boards which are operating at wide range of frequencies, makes our software radio up and running quickly. Process is very simple, it requires a Linus-Based software GNU Radio, which is a complete signal processing package and open source software radio, and as a result the USRP is in working state. After the installation of software on Linux and synchronizing the USRP with computer, it is ready for transmission and reception of signals.

Designers and engineers can design a software radio system and then implement it with a minimum of effort and least budget which is the true value of the USRP. By providing many practical software and hardware applications, large community of users and developers have contributed in the field of software radio development. Community of experienced users, open-source software and flexible hardware combines to make it an ideal platform for Software Radio Development.

3.1.1 Benefits of USRP

- It is a low cost and flexible platform
- GNU Radio – Linux Based Software, allows the users to design the Radio systems and then implement in on USRP for further Research and Development Purposes
- It has a large community of developers which makes it convenient for new developers.

3.1.2 Hardware

The USRP has two real time antennas due to which it can transmit and receive simultaneously. Creation of MIMO (multiple input, multiple output) systems is also possible since it has fully coherent local oscillators and sampling clocks. In USRP, there are two kinds of sample-rate processing that takes place;

- **High:** It happens in the field programmable gate array (FPGA)
- **Low:** It takes place in the computer.

Incoming signals in the FPGA are decimated and filtered (from 64 MS/s) by two onboard Digital Down-Converters (DDCs) mix. Before translating Baseband signals to the selected output frequency, they are interpolated to 128 MS/s by Two digital up converters (DUCs). High sample rates when combined with the DDCs and DUCs also simplify analog filtering requirements. Flexible and fully integrated RF front-ends are provided by the daughterboard mounted on the

USRP. For a broad range of applications, different frequencies are required and this problem is solved by a wide variety of available daughterboard. To transmit and receive data packets the USRP can hold up to two daughterboard for RF I/O (2 for transmission and 2 for reception).

3.2 GNU Radio

GNU Radio is a Linux-Based toolkit used for development for software. It is open-source & free. It is used to implement software radio systems by providing signal processing blocks. It could be used in two ways

- **With Hardware (USRP):** Creation of software-defined radios on already available RF hardware (low cost, usually external).
- **Without hardware:** It is more of a software simulation where results are available only on the host computer not on a practical hardware

It's usage in not limited to research and development, it is used in academia, industry, hobbyist, and government environments. Projects designed and simulated in GNU Radio are used to support both real-world radio systems and wireless communications research.

3.2.1 Software Radio

A radio system which has the ability to perform signal processing in software without using hardware is called Software Radio. Major benefit of using a Software Radio is that user doesn't have to change hardware whenever a new experiment is being done, only changes in software are required. As a result, one radio system could be used to create many kinds of software radio because software could easily be replaced in the radio system; thus, a single software radio is enough for multiple applications.

3.2.2 GNU Radio Application

Gnu Radio holds the ability to perform all kind of signal processing. It could be used for:

- Transmission and reception of the data with radio hardware by writing applications
- Simulation-based applications e.g. IEEE 802.15.4 Implementation

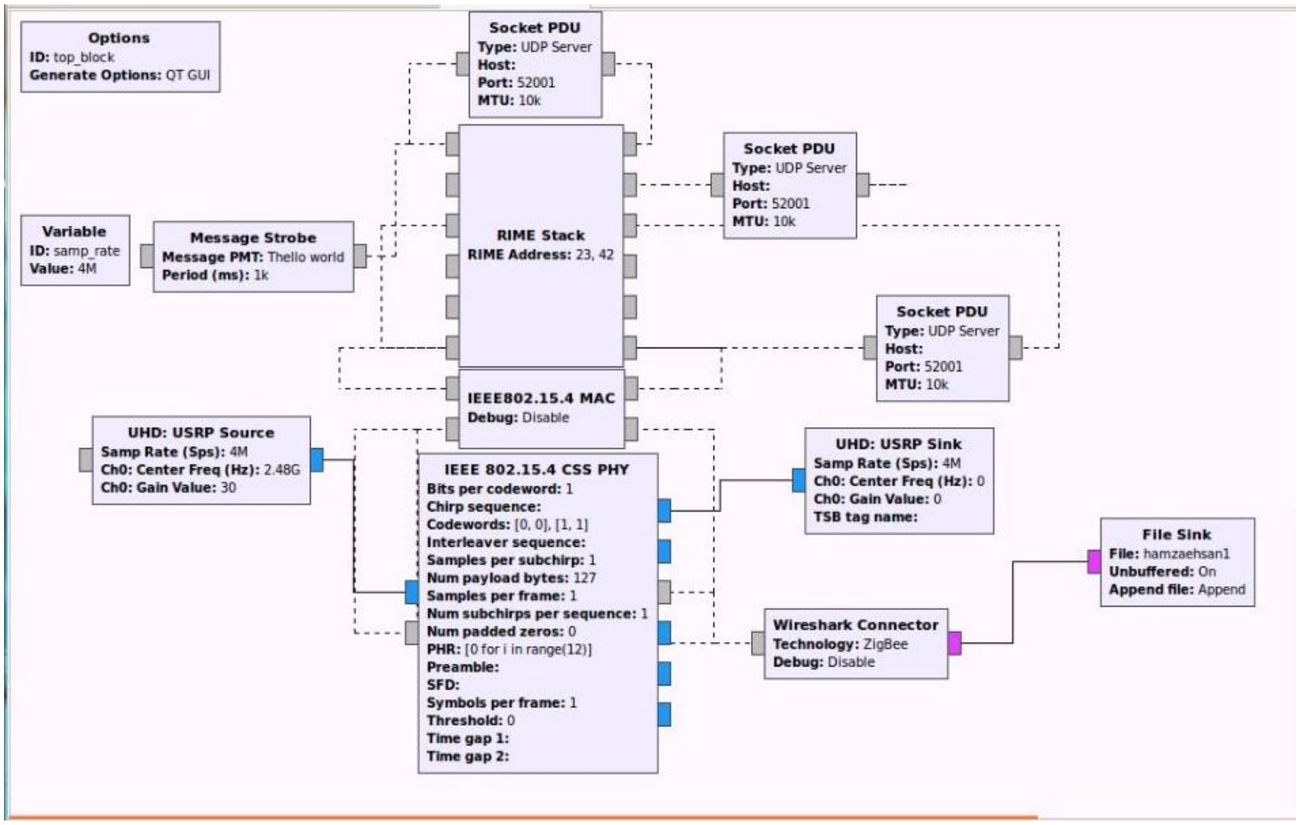
By default, in GNU Radio a broad range of blocks are available which are used in signal processing systems, such as filters, decoders, demodulators, modulators, synchronization elements, channel codes equalizers, vocoders, and many other types of blocks. Block extension in GNU Radio is also a possible, if a certain block of user's choice or need is missing then the user can create and add it manually which makes it much more user friendly than other software. Another important thing is that, GNU Radio has way of connecting these blocks according to their requirements, for example you cannot connect two blocks if they don't mutually operate and if also control data management as it passes from one block (or layer) to another in radio systems.

Two kinds of languages are used to write applications in GNU Radio, either C++ or Python. Developer must use C++ by using processor floating-point extensions where available, if he wants to implement high throughput and real time radio system in which performance-critical signal processing is required.

3.3 Software Simulation

Main objective behind this software simulation is to understand how IEEE 802.15.4 practically works. How a message is generated and transmitted to other nodes. How message is changed while going through multiple layered structure or protocol.

First, the open source ZigBee protocol was acquired, and blocks were generated in the GNU Radio software corresponding to the code.



Options
ID: top_block
Generate Options: QT GUI

Socket PDU
Type: UDP Server
Host:
Port: 52001
MTU: 10k

Socket PDU
Type: UDP Server
Host:
Port: 52001
MTU: 10k

Variable
ID: samp_rate
Value: 4M

Message Strobe
Message PMT: Thello world
Period (ms): 1k

RIME Stack
RIME Address: 23, 42

Socket PDU
Type: UDP Server
Host:
Port: 52001
MTU: 10k

IEEE802.15.4 MAC
Debug: Disable

UHD: USRP Source
Samp Rate (Sps): 4M
Ch0: Center Freq (Hz): 2.48G
Ch0: Gain Value: 30

IEEE 802.15.4 CSS PHY
Bits per codeword: 1
Chirp sequence:
Codewords: [0, 0], [1, 1]
Interleaver sequence:
Samples per subchirp: 1
Num payload bytes: 127
Samples per frame: 1
Num subchirps per sequence: 1
Num padded zeros: 0
PHR: [0 for i in range(12)]
Preamble:
SFD:
Symbols per frame: 1
Threshold: 0
Time gap 1:
Time gap 2:

UHD: USRP Sink
Samp Rate (Sps): 4M
Ch0: Center Freq (Hz): 0
Ch0: Gain Value: 0
TSB tag name:

File Sink
File: hamzaehsan1
Unbuffered: On
Append file: Append

Wireshark Connector
Technology: ZigBee
Debug: Disable

Figure 8: IEEE 802.15.4 Simulation on GNU Radio

3.3.1 Message Strobe

This Block is used to generate a message. Message could be a text, numbers or special characters, based on the needs of user. **Message PMT** is used for typing any kind of message. **Period (ms)** on the other hand defines the time interval after which Message Strobe is reset. Sending interval is reset after the specified time period (1K milliseconds in this case). After 1k milliseconds, (in this case) message will be resent and the process will go on until stopped manually.

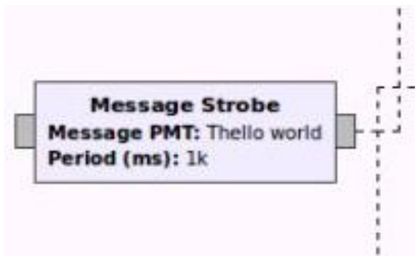

Message Strobe
Message PMT: Thello world
Period (ms): 1k

Figure 9: Message Strobe

3.3.2 Socket PDU

Socket Packet Data Unit is used to create a socket interface and it translates the traffic to Packet data units (PDUs). It is not possible to send the Message generated by Message Strobe as a single unit because the channels bandwidth doesn’t fulfill the requirements. For this purpose, Socket PDU is used to divide the Message into further packets which are then send through further layers i.e. MAC and PHY layer. If we go into details,

Type: UDP – it is a transmission protocol which is an alternative to TCP. UDP has different working principle as compared to TCP as it is used to send short packets of data known as datagrams. In our case, we also need to send short packets (PDUs) so UDP is the most convenient way forward

Port: It is used to assign a route to network which will be then used for data transmission. In this case, port number 52001 is open and will be used for data transmission

MTU: Maximum Transmission Unit – MTU is used to define the maximum size of data that can be communicated in a network transaction. In this case, it is 10kb which means that data transmission with size greater than 10kb is not possible. It is an important factor to know the limits as it will not cause problem in the later stages such collision, channel burdening etc.

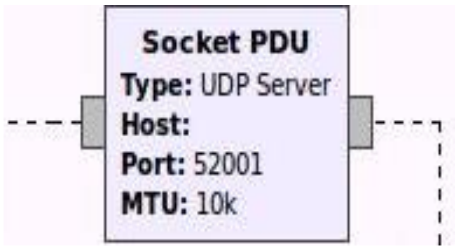

Socket PDU
Type: UDP Server
Host:
Port: 52001
MTU: 10k

Figure 10: Socket PDU

3.3.3 RIME Stack

Rime stack is a light-weight communication stack used in Wireless Sensor Nodes (WSNs). It has a layered structure that uses simple protocols to perform complex actions. Data is sent to and from Socket PDUs by use of a set of primitives. RIME address is used to address nodes in the RIME stack. It uses single hop and multi-hop communication primitives

Hop defines the number of networks a packet must go through to reach its destination. It has two types

Single hop network

In a single hop network, a packet passes through a single hope when it leaves the source before reaching its destination address. That single hop could be anything from Coordinator to Router etc.

Multi-hop network

In a multi-hop network, a packet must go through 2 or more networks to reach its destination address.

Hop also means re-routing. Each hop possesses an ability to slow traffic due to the processing overhead while sending data across greater physical distances and through multiple channels.

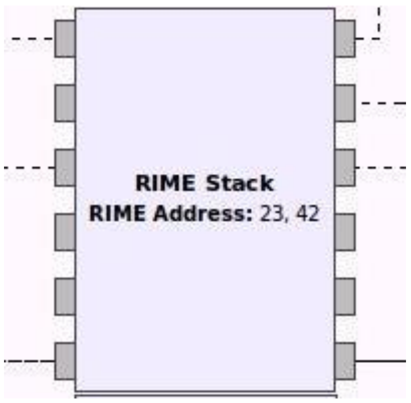


Figure 11: RIME Stack

3.3.4 MAC Layer

MAC layer has ability to enable the connectivity by encapsulating packets from higher layers with a valid header. It has all the features defined in the above Literature Review section such as management of generated beacons, acknowledged frame delivery, General Time Slot management, channel access, frame validation, association, and disassociation. MAC Layer is also responsible for security of network

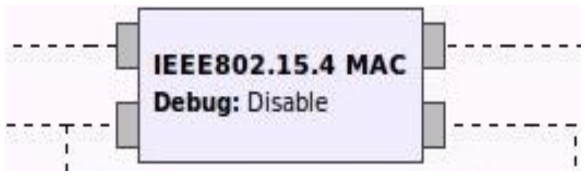

IEEE802.15.4 MAC
Debug: Disable

Figure 12: MAC Layer

3.3.5 Physical Layer (PHY)

Physical layer (PHY) is responsible for the connection between the MAC Layer and the physical radio channel through RF Firmware and Hardware. All modulation processes are carried out in PHY layer. No. of samples are inserted to avoid a complex issue. PHY Layer uses Chirp Spread Spectrum (CSS) in which information is encoded using wideband linear frequency. Chirp is usually a sinusoidal signal whose magnitude increases or decreases with time. It has a long list of parameters which perfectly defines the use of PHY layer.

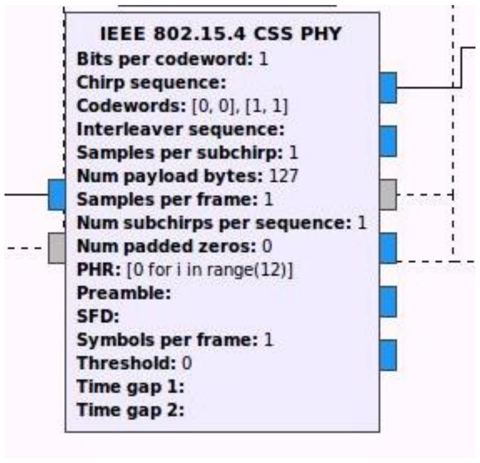

IEEE 802.15.4 CSS PHY
Bits per codeword: 1
Chirp sequence:
Codewords: [0, 0], [1, 1]
Interleaver sequence:
Samples per subchirp: 1
Num payload bytes: 127
Samples per frame: 1
Num subchirps per sequence: 1
Num padded zeros: 0
PHR: [0 for i in range(12)]
Preamble:
SFD:
Symbols per frame: 1
Threshold: 0
Time gap 1:
Time gap 2:

Figure 13: Physical Layer

3.3.6 USRP Sink and USRP Source

USRP Source is the point from which data is being transmitted and USRP sink defines the properties of the USRP to which data is being transmitted. They are like two USRPs, one of them is receiving the data and other one is sending it.

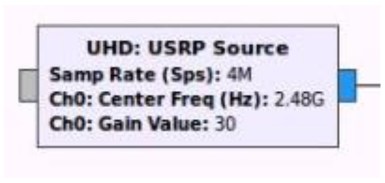

UHD: USRP Source
Samp Rate (Sps): 4M
Ch0: Center Freq (Hz): 2.48G
Ch0: Gain Value: 30

Figure 15: USRP Source

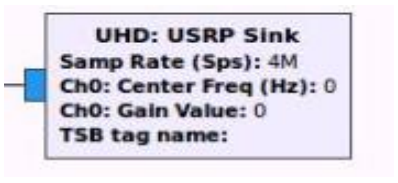
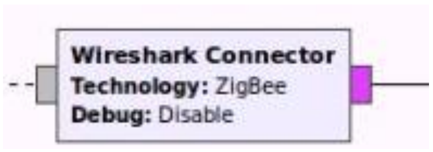

UHD: USRP Sink
Samp Rate (Sps): 4M
Ch0: Center Freq (Hz): 0
Ch0: Gain Value: 0
TSB tag name:

Figure 14: USRP Sink

3.3.7 Wireshark Connector

Wireshark is a well-known software which is used to capture packet that are being transmitted from one point to another. In this case, Wireshark is used to capture the data transmission between two USRPs. Data capturing is very useful technique as it enables us to check if encryption is working or not. It also tells us the nature of packets that are travelling through the air.



Technology: ZigBee

Debug: Disable

Figure 16: Wireshark Connector

Chapter 4: Hardware and Software Selection

4.1 Background Study

The problem with IEEE standard 802.15.4 is that its stack is either provided by the hardware manufacturers so therefore are limited to their products, or are licensed and need to be bought for large fees. This calls for some major problems for individual users who want to create and implement their own ZigBee designs as they either cannot afford the proprietary fees or the costs of the needed for creating a ZigBee application. Individual enthusiasts with specific domain knowledge have in many cases come up with some of the most innovative creations. This project aims to provide a free stack to allow users the freedom to create new and innovative projects that can improve people’s lives.

Some ZigBee designs have a limitation that they can only use a specific microcontroller as that controller may be part of a device that's needed for a specific application so it's next to impossible to use different hardware to optimize an application. However, the addition of wireless communications would greatly benefit the application. Currently, to make a ZigBee application, there isn't any easy way to take a microcontroller such as an ARM based one and mix it with an 802.15.4 radio from a different vendor due to the reason that the ZigBee software created by specific manufacturers are either in binary form or contain a license clause which only permits the use of the software with their hardware. Licensed stack owned by software companies usually charge stack licensing fees that, not only are expansive in itself (can go upwards of $50k) but also require fees for driver modification for specific microcontroller. One of the goals of this project is to provide a free, customizable ZigBee stack which will give the user enough flexibility to choose their own components without having to worry about proprietaries.

So a list of open source protocol stacks were studied and tested on to see which one would provide the most flexibility and compliance with other modules without diminishing any control over the stack.

4.2 Protocol Selection

So a list of open source protocol stacks were studied and tested on to see which one would provide the most flexibility and compliance with other modules without diminishing any control over the stack.

4.2.1 Available ZigBee protocol stack

4.2.1.1 ZBoss

ZBOSS: the ZigBee® Open Source Stack was one of the first open-source stack that was certified by the ZigBee alliance. Its goal is to resolve the compatibility issues being face by global manufacturing and research communities. ZBOSS was certified by partner companies since it is a high-performance, small memory footprint, cross-platform solution. This stack is compliant with chipsets from different manufacturers. Its architecture minimizes retransmits and packet loss. Currently ZBOSS is available on Linux and different OS-less platforms. This protocol stack aims to better customers' daily lives by helping product vendors manufacture green, smart products.

4.2.1.2 FreakZ

The FreakZ ZigBee stack is an open source stack that is compliant with many different RF modules and gives users high control over the stack. The problem with this stack though is that it wasn't completed and was left underdeveloped back in 2008.

4.2.1.3 Z-Stack by Texas Instruments

Z-Stack 3.0 created by TI is a ZigBee 3.0 compliant protocol suite. Z-Stack 3.0 has combined multiple previous ZigBee profiles into one unified standard. This stack allows cost and energy efficient as well as ultra-low power devices to connect to a single ZigBee network. It's based on ZigBee PRO 2015 stack and provides new and improved security modes, including Install Codes for out of band key exchange. Compatibility has been maintained with all the previous ZigBee PRO application profiles.

4.2.2 Usability of the selected stack

Hardware Compliance

- ZBoss: The ZBOSS stack currently supports two ZigBee® chipsets: the Texas Instruments' TI CC253x and UBEC's UBEC 2400, UBEC 2410.

- Z-Stack by Texas Instruments: The Z-Stack 3.0 is currently compliant with the Texas Instruments' TI CC253x chipsets.

Online Resources

- ZBOSS: ZBOSS' website provides resources and help material for this particular stack.

- FreakZ: Support is available on FreakZ official website and on GitHub.

- Z-Stack by Texas Instruments: Texas Instruments' official website provides extensive resources plus an online forum for queries to be answered by professional TI technical staff.

Most up to date:

- ZBOSS: ZBoss, although a good protocol stack, was developed in 2013 and hasn't been updated since then. Because of this stack is not compatible on newer hardware.

- FreakZ: FreakZ was left underdeveloped in 2008 and hasn't been updated since.
- Z-Stack by Texas Instruments: Z-Stack was developed in 2017 and has been updated constantly since so out of all the stacks, this one is the most advanced and the most compatible with newer hardware.

4.3 Hardware Shortlisted

4.3.1 Digi XBee

Digi XBee ZigBee RF modules provide cost effective wireless connectivity to electronic devices. They're compatibility has been tested against other ZigBee compliant devices, including devices from other manufacturers. Since Digi XBee ZigBee modules are highly energy efficient, they are ideal for applications in the energy and controls markets as they require critical manufacturing efficiencies. Digi ZigBee Development Kits are the perfect way to begin ZigBee application development. The Serial Peripheral Interface (SPI) provides an optimized integration with embedded microcontrollers as well as a high-speed interface and lowers development costs. The Digi XBee family hardware do not need any configuration. The programmable versions of the Digi XBee ZigBee module make customization of applications quite easy. A separate processor just for programming the module is now no longer needed since the module can be programmed directly. The risk to RF performance or security by the development of applications has been minimized by isolating the wireless software. Digi's ZigBee are compatible with the Ember EM35x (EM357 and EM3587) system on chip (SoC) radio ICs from SiliconLabs, utilizing 32-bit ARM CortexTM M3 processor.

Figure 17: DigiXBee Module

4.3.2 ANY-900

ANY900 is an ultra-low power 802.15.4/ZigBee RF modules for Sub-1 GHz ISM band created by Adaptive Network Solution. The tiny modules feature an exceptional sensitivity of -110 dBm that results in the line-of-sight range of up to 500m. Featuring the built-in chip antenna, ANY900 module presents a fully integrated solution for the system integrators and OEMs. These modules eliminate the need for costly and time-consuming RF development, and shorten time to market for a wide range of standards based wireless products.

Figure 18: ANY-900 Module

4.3.3 Zigbit

Atmel provides a variety of wireless MCU solutions in the 802.15.4 platform. They provide multiple RF transceiver solutions in the 2.4GHz band and the 700/800/900MHz band. These radios can be very easily interfaced with any of Atmel's AVR or SAM microcontrollers through the digital SPI port. Atmel also provides single chip RF solutions which combine the microcontroller and RF transceiver into a single chip. ZigBit modules combine these wireless MCU solutions with an approved antenna interface in an easy to implement platform. Using ZigBit modules can save the developer time and money by not having to pursue quite as much FCC testing. Atmel also provides wireless stacks and example code in Atmel Studio for these ZigBit modules. The purpose of this page is to familiarize the user with these modules and the differences between them.

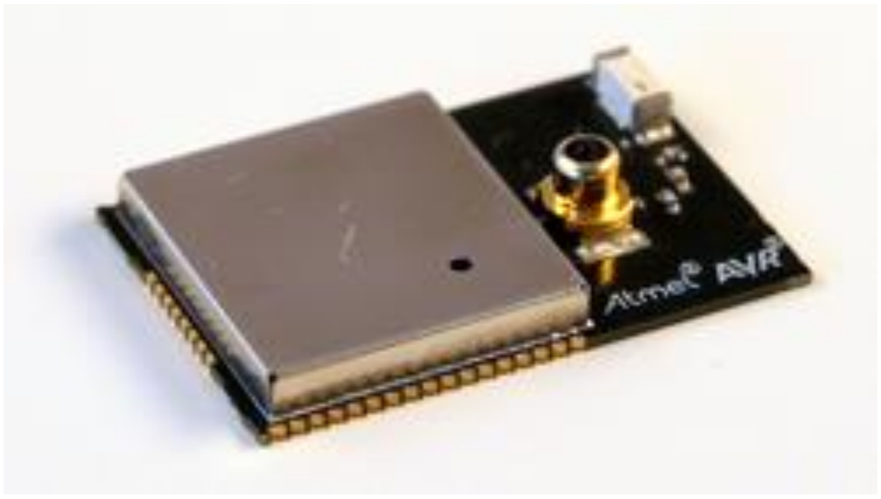


Figure 19: DigiBit Module

4.3.4 CC 2531 System on Chip (SoC)

The CC2531 USB Dongle created by the Texas Instruments support a PC interface to 802.15.4 / ZigBee applications. The dongle can be plugged directly into a PC and can be used as an IEEE 802.15.4 packet sniffer or for other purposes. With the CC2531 USB Firmware Library available on the web, users can develop their own software to utilize this part. The USB dongle can be used as a reference module for prototyping of USB devices and for testing the RF performance of CC2531 with a small size PCB antenna.

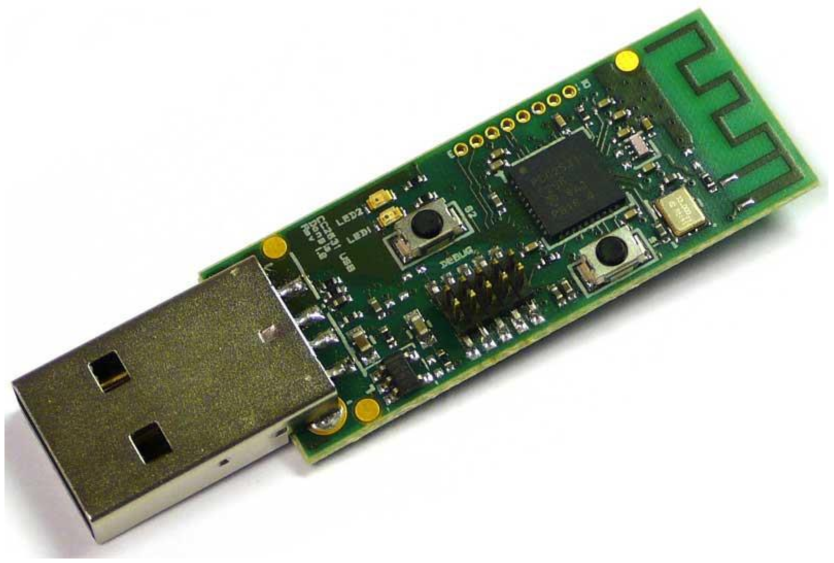


Figure 20: CC-2531 USB Dongle

4.4 Specifications

4.4.1 Digi Xbee

TRANSCEIVER CHIPSET	Silicon Labs EM357 SoC
DATA RATE	RF 250 Kbps, Serial up to 1 Mbps
INDOOR RANGE*	Up to 200 ft (60 m)
OUTDOOR/RF LINE-OF-SIGHT RANGE	Up to 4000 ft (1200 m)
TRANSMIT POWER	3.1 mW (+5 dBm)
SERIAL DATA INTERFACE	UART, SPI
FREQUENCY BAND	ISM 2.4 GHz
INTERFERENCE IMMUNITY	DSSS (Direct Sequence Spread Spectrum)
DIGITAL I/O	15
MEMORY	Programmable: 32 KB Flash/2 KB RAM
PROTOCOL	ZigBee PRO 2007, HA-Ready with support for binding/multicasting
ENCRYPTION	128-bit AES
CHANNELS	16 channels
SUPPLY VOLTAGE	2.1 to 3.6V

COST	Approx. 60 USD

4.4.2 ANY-900

TRANSCEIVER CHIPSET	
DATA RATE	up to 1 Mbit/s
INDOOR RANGE*	
OUTDOOR/RF LINE-OF-SIGHT RANGE	Up to 16400 ft (5000 m)
TRANSMIT POWER	3.1 mW (+5 dBm)
SERIAL DATA INTERFACE	
FREQUENCY BAND	779-787 MHz / 863-870 MHz / 902-928 MHz
INTERFERENCE IMMUNITY	
DIGITAL I/O	
MEMORY	Programmable: 128 KB Flash/8 KB RAM
PROTOCOL	
ENCRYPTION	128-bit AES

CHANNELS	
SUPPLY VOLTAGE	1.8 to 3.6 V
COST	

4.4.3 Zigbit

TRANSCEIVER CHIPSET	ATmega256RFR2 SoC
DATA RATE	250, up to 2000 kbps
OUTDOOR/RF LINE-OF-SIGHT RANGE	170 - 570 m
TRANSMIT POWER	-16.5 to +3.5 dBm
FREQUENCY BAND	2.4000 to 2.4835
INTERFERENCE IMMUNITY	DSSS (Direct Sequence Spread Spectrum)
DIGITAL I/O	8
MEMORY	256KB Flash, 32KB RAM
ENCRYPTION	128-bit AES
CHANNELS	16

SUPPLY VOLTAGE	1.8 to 3.6 V
COST	

4.4.4 CC 2531 USB Dongle

TRANSCEIVER CHIPSET	Texas Instruments CC 2531 SoC
DATA RATE	250 Kbps
INDOOR RANGE*	Up to 33 ft (10 m)
OUTDOOR/RF LINE-OF-SIGHT RANGE	Up to 260 ft (80 m)
OUTPUT POWER	4.5 dBm
SERIAL DATA INTERFACE	USB, SPI, UART
FREQUENCY BAND	ISM 2.4 GHz
INTERFERENCE IMMUNITY	DSSS (Direct Sequence Spread Spectrum)
DIGITAL I/O	6
MEMORY	Programmable: 256 KB Flash/ 8 KB RAM
PROTOCOL	ZStack 3.0

ENCRYPTION	128-bit AES
CHANNELS	8 channels
SUPPLY VOLTAGE	2.1 to 3.6V
COST	Approx. 60 USD

4.5 CC 2531 vs. ANY-900:

By comparing the specifications of CC 2531 with ANY-900 it is quite clear that the CC 2531 is more suitable for this project. First, ANY-900 only works on the 779-787 MHz / 863-870 MHz / 902-928 MHz frequency bands whereas the CC 2531 also works on the 2400 MHz band. As frequency band that'll be used for this project is the 2.4 GHz band, it is already clear that the CC 2531 is more suitable. There's not much difference when it comes to the encryption, memory and transmit power. The ANY-900 module has a far better range (5000 m in open air) compared to the CC 2531 (80 m in open air) but that can be resolved by using a better antenna. The CC 2531 is capable of both serial communication and RF transmissions compared to the ANY-900 module having only serial communication.

4.6 CC-debugger

A small debugger and programmer for low-power SoC devices by texas instruments, known as CC Debugger. Together with the Embedded Workbench for 8051 by IAR (version 7.51A or later), it is used to debug while the SmartRF Flash Programmer is used for flash programming (hex-file uploading or otherwise known as porting). Some selected devices can also be controlled using SmartRF Studio.

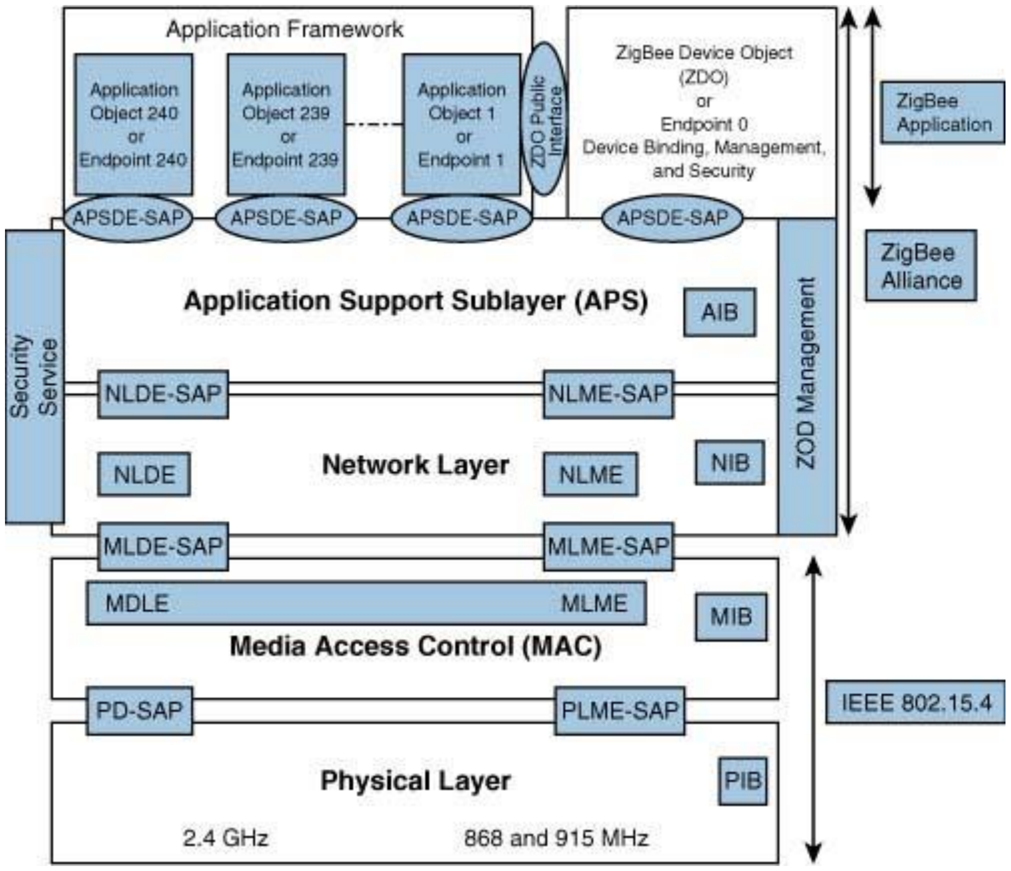
Chapter 05: Z-Stack

5.1 What is Z-Stack?

Z-Stack is a CC-2531 compatible protocol stack. It supports several other wireless MCUs including CC 2530 and CC2538, just to mention a few. However, the one that supports this project is the creation of wireless sensor nodes by implementing the Physical (PHY) Layer and MAC Layer. Supporting the RF front ends CC2590 and CC2592, it provides a transmit power of up to +14dBm and +20dBm respectively, thus providing better receiver sensitivity.

The architecture of ZigBee protocol stack is split into three sections, which are as follows:

- IEEE 802.15.4, consisting of the physical layers and MAC layers.
- Layers constituted of network layer, security management, application sub layer, and ZigBee Device Object layer.
- Manufacturer application: ZigBee device manufacturers are at liberty to either create their own application profile or use the ZigBee application profile.



Application Framework
Application Object 240 or Endpoint 240
Application Object 239 or Endpoint 239
Application Object 1 or Endpoint 1
ZDO Public Interface
ZigBee Device Object (ZDO) or Endpoint 0 Device Binding, Management, and Security
ZigBee Application
APSDE-SAP
Application Support Sublayer (APS)
AIB
ZigBee Alliance
Security Service
NLDE-SAP
NLME-SAP
NLDE
Network Layer
NLME
NIB
ZOD Management
MLDE-SAP
MLME-SAP
MDLE
MLME
MIB
Media Access Control (MAC)
PD-SAP
PLME-SAP
IEEE 802.15.4
Physical Layer
PIB
2.4 GHz
868 and 915 MHz

Figure 21: Layered Z-Stack

5.2 Specifications

The fully customizable protocol stack, ZigBee Protocol Stack, enables the programmer to control the way a ZigBee device behaves. New features of Z-Stack's latest version are highlighted as follows:

- All of the previous profiles are unified into one standard in Z-Stack 3.0
    - A unified Cluster Library is incorporated by the Z-Stack, setting foundation of the IoT applications' universal language **dotdot.** It defines models, data objects, and embedded IoT applications
    - A set of mechanisms to form and discover the network by implementing ZigBee Device Behavior specification, and the use of application provision is allowed.
    - Better and new security modes as provided on the basis of ZigBee PRO 2015 stack, which includes Install Codes for exchanging key out-of-band, and a network topology without a coordinator using Distributed Security Networks.
    - Green power proxy is also supported, which allows connection of ultra-low power devices to a ZigBee network and energy harvesting.
    - Application profiles and ZigBee PRO are forward and backward compatible in this version of Z-Stack.
- For the purpose of prototyping, Sample Applications are included. These include thermostat, temperature sensor, light and switch, and door lock.
- For ZigBee applications with 2-chip architectures, an access to the Base Device Behavior and ZigBee Pro 2015 stack is provided through a serial port in ZNP firmware.
- Deployed systems can be updated in the future because of Z-Stack being capable of bootloading and firmware upgrade.
- Application integrations on an IP based interface are made convenient because there is Ethernet-to-ZigBee gateway implementation for reference by using a Linux Gateway; it is named as ZigBee Linux Gateway by Texas Instruments.

5.3 Layers

5.3.1 BDB

The Base Device Behavior layer specifies the environment, initialization, commissioning and operating procedures of a base device operating on the ZigBee-PRO stack to ensure profile interoperability. The Z-Stack BDB provides the data structures and interface that an application developer requires to comply with this specification. It also provides a generic implementation of reporting attributes for applications that supports clusters with reportable attributes.

BDB describes how a general ZigBee Device will enter and behave in the ZigBee network ensuring profile interoperability between devices. More specifically, the BDB specification defines the following functionalities:

- The initialization procedures
- The commissioning procedures
- The reset procedures
- The security procedures
- Reporting attributes

All these procedures are performed by using the existing API from other layers such as ZDO, ZCL, NWK and APS

5.3.2 ZDO

The functionality of device management is supplied by the ZigBee Device Objects (ZDO) layer of the Z-Stack. Application endpoints are provided with interfaces that contribute to managing functionality for ZigBee Routers, Coordinators, or End Devices. This inculcates discovering the network after creation, and then joining it; managing security and binding application endpoints.

ZDP describes how general ZigBee Device features are implemented within ZDO. It defines Device Description and Clusters which employ command and response pairs. Through the definition of messages in command structure, ZDP provides the following functionality to the ZDO and applications.

- Service & Device Discovery
- Device Network Startup
- Network Management Service
- Bind and Unbind Service, End Device Bind

5.3.3 AF

The Application Framework (AF) interface supports an Endpoints (including the ZDO) interface to the underlying stack. The Z-Stack AF provides the data structures and helper functions that a developer requires to build a device description and is the endpoint multiplexor for incoming messages.

The Application Framework layer is the application's over-the-air data interface to the APS layer. It contains the functions an application uses to send data out over the air (through the APS and NWK) layers. This layer is also the endpoint multiplexer for incoming data messages.

The AF provides the following functionality to applications:

Endpoint Management

Each device is a node in the ZigBee. Each node has a long and short address, the short address of the node is used by other nodes to send it data. Each node has 241 endpoint (0 reserved, 1-240 application assigned). Each endpoint is separately addressable; when a device sends data, it must specify the destination node's short address and the endpoint that will receive that data. An application must register one or more endpoints to send and receive data in a ZigBee network.

Sending and Receiving Data

Sending Data

In order to transmit data to other devices in the network, the function call AF_DataRequest () is used.

Receiving Data

Data packets are sent to an application's registered endpoint. The application will receive the AF_INCOMING_MSG_CMD OSAL message in its OSAL event processing function

5.3.4 APS

General support services used by manufacturer-defined objects as well as ZDO are provided by the Application Support Sub layer (APS) API. The APS provides the following management functionality accessible to the higher layers:

Binding Table Management

The APS Binding table is a table defined in static RAM (not allocated). The table size can be controlled by 2 configuration items in f8wConfig.cfg [MAX_BINDING_CLUSTER_IDS & NWK_MAX_BINDING_ENTRIES]. The number of entries in the table are defined by NWK_MAX_BINDING_ENTRIES and MAX_BINDING_CLUSTER_IDS defines the number of clusters (16 bits each) in each entry. The table is defined in nwk_globals.c. The table is only included (along with the following functions) if REFLECTOR or COORDINATOR_BINDING is defined. Using the compiler directive REFLECTOR enables the source binding features in the APS layer

Group Table Management

The APS group table is a link list defined with allocated RAM [osal_mem_alloc()], so as groups are added to the table the amount of OSAL heap used increases. The maximum table size can be changed by adjusting APS_MAX_GROUPS in f8wConfig.cfg. The table is defined in nwk_globals.c. To use the group table functions include "aps_groups.h" in your application.

Quick Address Lookup

The APS provides a couple of functions to provide quick address conversion (lookup). These functions allow you to convert from short to IEEE address (or IEEE to short) if the lookup has already been done and stored in the address manager (ref. network layer) or if it's your own address.

Besides the management functions, APS also provides data services that are not accessible to the application. Applications should send data through the AF data interface [AF_DataRequest()]. To use the binding table functions include "BindingTable.h" in your application.

5.3.5 NWK

The ZigBee network (NWK) layer provides management and data services to higher layer (application) components. Following functionalities in the Network Layer are accessible to the higher layers:

- Address Management
- Network Management
- Utility Functions and Network Variables

Besides the management functions, NWK also provides data services that are not accessible to the application. Applications should send data through the AF data interface [AF_DataRequest()].

5.3.6 Green Power

The Green Power layer data services that allows the higher layer to send or receive data from the Green Power stubs. These stubs are used to interact with green power devices either for commissioning or normal operation in the network.

5.3.7 ZMAC

An interface between the ZigBee NWK layer and 802.15.4 MAC layer is provided by the ZMAC.

Interface Mechanisms

The following interface mechanisms are used in the MAC API.

Message Passing Function Calls

These API functions provide a message passing interface to the MAC by sending an OSAL message to the MAC event handler. Unless otherwise noted, the API functions described in this document are message passing functions. These functions do not contain critical sections and do not directly access MAC data.

Direct Execute Function Calls

These API functions directly execute code that performs a MAC operation. The function executes in the context of the caller. These functions may have critical sections and may directly access MAC data.

Callback Functions

Implementation of Callback functions must occur by application and are utilized to pass events and data from the MAC to the application. The validity of data that is accessed through callback function parameters, pointer to data for instance, only exists for the execution of the function and should be considered invalid when the function returns. The execution of these functions occurs in the MAC context. The callback function implementation should avoid using critical sections and CPU intensive operations.

Zero Copy Data Interfaces

The interfaces for sending and receiving data between the MAC and the application are designed to not require a data copy, i.e. they are “zero copy”. This results in a more CPU-efficient implementation. However, the application must follow certain rules on when to allocate and de-allocate data buffers.

For additional details regarding the functions used in each of the layers above, refer to “Z-Stack API”

5.4 Other Components

OSAL

TI Stack Software Components are shielded from the processing environment specifics by the OS abstraction layer. Following is the functionality provided, which does not depend on the processing environment.

- Message exchange between tasks
- Initialization, Task registration, starting
- Interrupt handling

- Task synchronization
- Memory allocation
- Timers

Message Management API

The mechanism for exchanging messages among processing elements and tests is provided by the Message Management API in the OSAL with different processing environments; for instance, functions called in the current loop or interrupt service routines. The allocation and de-allocation of message buffers, sending commands, and receiving response is enabled by the functions in Message Management API.

Task Synchronization API

A task is enabled in this API to wait for happening of events in order to return control. Events can be set for a task and the task can be notified as the event is set.

Timer Management API

The use of timers by external and internal tasks is enabled in this API. Functions to stop or start a timer are provided in this API where the timer can be set with 1 millisecond of increments

Interrupt Management API

Interfacing of a task with external interrupts is enabled in this API. A task can associate each interrupt with a service routine when the functions in the API are used.

Task Management API

Tasks are managed and added to the OS Abstraction Layer, using this API. An event processing function and an initialization function makes up every task. OSAL calls osalInitTasks() [application supplied] to initialize the tasks and OSAL uses a task table (const pTaskEventHandlerFn tasksArr[]) to call the event processor for each task (also application supplied).

Memory Management API

Memory Management API signifies as a representative of simple memory allocation system, where a dynamic memory allocation is allowed by the functions.

Power Management API

OSAL power management system is described in this API. When it is safe to turn off the external hardware and the receiver, the system helps tasks/applications to notify OSAL, and takes the processor to sleep. Power management is controlled by two functions. The first is osal_pwrmgr_device() which is called to set the device level mode (no power savings or power save). Following this, task power state exists, where using osal_pwrmgr_task_state(PWRMGR_HOLD) call, each task is capable of holding off the power manager from conserving power. For power manager to remain in the power conserve mode, if the power manager is in "Hold" by a task, osal_pwrmgr_task_state(PWRMGR_CONSERVE) will need to be called. Power state of a task is set to PWRMGR_CONSERVE when a task is initialized. Hence, no change in the power conservation function is required if a task wishes not to hold off power conservation; in this case there's no need for osal_pwrmgr_task_state() function call. In addition, by default, a battery-operated device will be in PWRMGR_ALWAYS_ON state until it joins the network, then it will change its state to PWRMGR_BATTERY. This means that if the device cannot find a device to join it will not enter a power save state. If you want to change this behavior, add osal_pwrmgr_device(PWRMGR_BATTERY) in your application's task initialization function or when your application stops/pauses the joining process. Before the power manager goes into the state of power conserve, it will note the collective power state and the device mode of all tasks.

For more details on individual functions and parameters in the OS Abstraction Layer (OSAL), refer to "Z-Stack OSAL API"

Chapter 06: Embedded System Implementation

6.1 Hands-on Working with Off-the-shelf ZigBee

For better understanding and analysis on how data transmission from one ZigBee to another occurs in a network, communication in a wireless sensor network of ZigBee modules was performed. For this purpose, off-the-shelf ANY-900 modules were acquired and a network was formed with four of the ANY-900 ZigBee modules, where Star and Peer-to-Peer topologies were implemented respectively. The serial communication tool that was used in the process is “HyperTerminal” which was configured with the commands from SMS Command Reference Guide. In order to perform various data transmission schemes, refer to “SMS Command Reference Guide” by ANY Solutions.

An antenna was used with each ANY-900 ZigBee device. USB to TTL converters were used to connect the ZigBee devices to the computer.

6.1.1 ANY-900 Module by AN Solutions

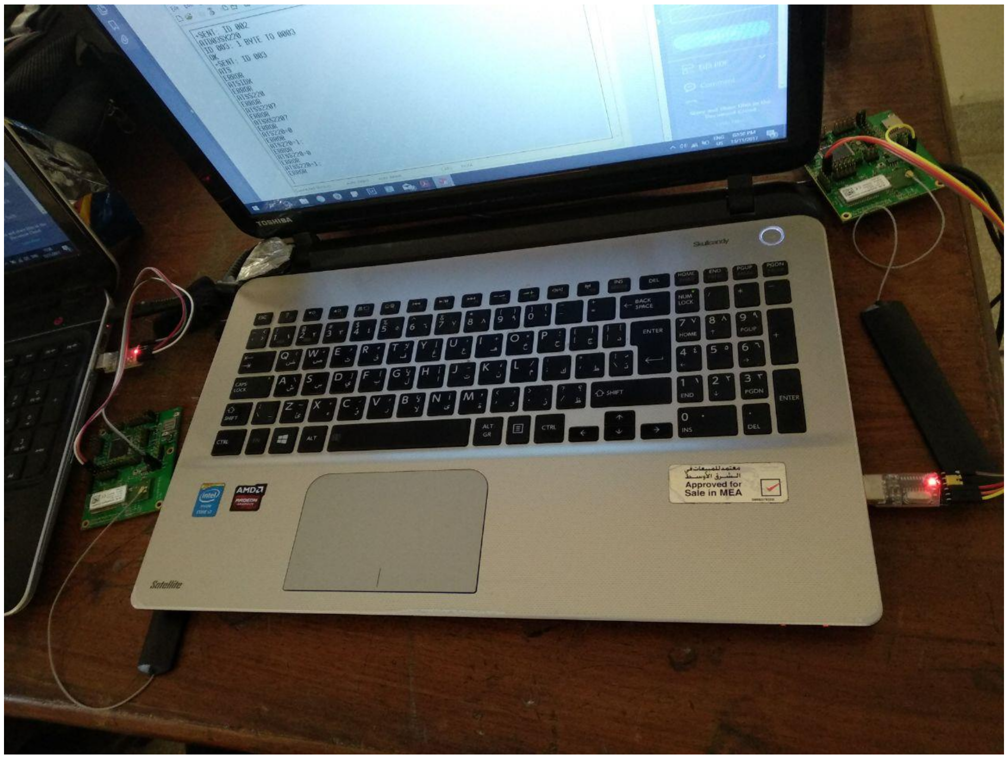
ANY900-2 is an 802.15.4/ZigBee RF module with ultra-low power consumption. It has a sensitivity of -110 dBm and a scalable output power of up to +10dBm, resulting into the range of up to 5,000m in line-of-sight. U.FL antenna connector in this module enables a rapid design-in. For every application, it allows the use of a different external antenna, hence extending greater flexibility for developers. ISM band applications of 783, 868, and 915 MHz are supported by the adjustable software. While the ANY-900 module helps avoid hectic RF-development, it lacks a control over stack since it is supplied with a pre-built firmware.

6.1.2 HyperTerminal

HyperTerminal is a computer software program. It connects to computer systems using internet through SSH or Telnet, by either the Dial-Up Modem, or it can be directly connected to the computer by a COM port and an RS232 serial cable. Activation procedure for HyperTerminal involves clicking at Start, choosing "Programs", clicking on "Accessories" on the submenu and then choosing "Communications." If communications menu in your computer shows HyperTerminal, you can just click at it to get started. Further details on commands required to configure ANY-900 ZigBee modules are given in the SMS Command Reference Guide.

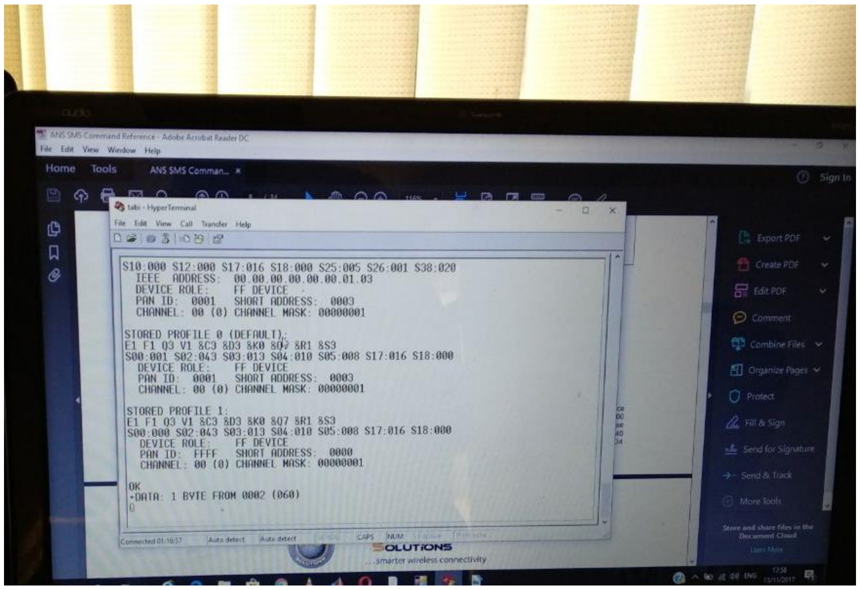
6.1.3 Data Transmission

The data transmission requires creation of a network. At first, use the command reference guide to configure the network with your required specifications. The network topologies experimented with in “Hands-on working” on off-the-self ZigBee device (ANY-900) in this project are Star and Peer-Peer topologies. Other parameters that need to be configured include a PAN ID which is specified by the user, the receiver's PAN ID, and the network address. All of the information regarding these parameters and the commands required for that purpose are explained in the SMS Command Reference Guide.



```
+SENT: ID 002
ATD003SX220
ID 003: 1 BYTE TO 0003
OK
+SENT: ID 003
ATS
ERROR
ATS1DK
ERROR
ATSS220
ERROR
ATSS220?
ERROR
ATSXS220?
ERROR
ATS220=0
ERROR
ATS220=1:
ERROR
ATSS220=0
ERROR
ATSS220=1:
ERROR
```

Figure 22: ANY-900 Modules Communicating



```
S10:000 S12:000 S17:016 S18:000 S25:005 S26:001 S38:020
  IEEE  ADDRESS:  00.00.00.00.00.00.01.03
  DEVICE ROLE:    FF DEVICE
  PAN ID:  0001   SHORT ADDRESS:  0003
  CHANNEL: 00 (0) CHANNEL MASK: 00000001

STORED PROFILE 0 (DEFAULT):
E1 F1 Q3 V1 &C3 &D3 &K0 &Q7 &R1 &S3
S00:001 S02:043 S03:013 S04:010 S05:008 S17:016 S18:000
  DEVICE ROLE:    FF DEVICE
  PAN ID:  0001   SHORT ADDRESS:  0003
  CHANNEL: 00 (0) CHANNEL MASK: 00000001

STORED PROFILE 1:
E1 F1 Q3 V1 &C3 &D3 &K0 &Q7 &R1 &S3
S00:000 S02:043 S03:013 S04:010 S05:008 S17:016 S18:000
  DEVICE ROLE:    FF DEVICE
  PAN ID:  FFFF   SHORT ADDRESS:  0000
  CHANNEL: 00 (0) CHANNEL MASK: 00000001

OK
+DATA: 1 BYTE FROM 0002 (060)
0
```

Figure 23: Data Byte Transmission in ANY-900 ZigBee Network

6.2 Tools & Software

This section explains the tools and software used in the process of achieving the project goals. Further details are given on their respective websites.

6.2.1 IAR Embedded Workbench

A C/C++ debugger and compiler which supports several processor families. In this project, the software is used as C/C++ compiler for 8051.

Following are the targets supported:

8051, 78K, AVR, ARM, CR16C, AVR32, Coldfire, HCS12, H8, M16C, MSP430, M32C, Maxim, R32C, MAXQ, R8C, RL78, RH850, RX, S08, STM8, V850, SAM8, SuperH,.

IAR Embedded Workbench finds its application in this project as a C Compiler for 8051 Microcontroller.

For more details on the software and support, visit https://www.iar.com/iar-embedded-workbench/.

6.2.2 SmartRF Flash Programmer

In order for programming of flash memory in 8051 based RF MCUs by Texas Instruments, SMartRF Flash Programmer is used to upgrade the bootloader and firmware on the SmartRF Transceiver Evaluation Board (TrxEB), the CC-Debugger, SmartRF05 Evaluation Board.

Features

- Updating bootloader and firmware on ISB MCU of Evaluation Boards
- SW-image programming on wireless MCUs
- On available software on device, append the software image
- On-device software image verification against a file
- Software image reading from the device to ELF, hex or binary files.
- MAC address read/write
- Information Page reading on device
- Command Line Interface
- Flash lock bits' programming

Supported interfaces & debuggers

- CC Debugger
- SmartRF04EB

- SmartRF Flash Programmer
- SmartRF TrxEB
- SmartRF05EB

In order to get started with Flash Programmer Software, refer to the User Manual under the "User Guides" heading at http://www.ti.com/tool/FLASH-PROGRAMMER.

6.2.3 Z-Tool

Z-Tool is a PC-based software that has been made to operate ZigBee devices containing the protocol stack that has been used in this project. Z-Tool can be used to create networks using the appropriate commands and parameters. It allows to configure devices as end points and communicate the devices within a network. The software has been made compliant with the protocol stack used, and hence it contains all the functions of the stack.

Unzip the Z-Stack 3.0.1 file allows you to extract Texas Instruments Z-Stack folder. The Z-Tool can be found in the "Tools" folder of this Texas Instruments Z-Stack folder.

6.2.4 SmartRF Packet Sniffer

A software application based on PC, which is capable of storing, displaying the captured packets by an RF listening device. The connection to the capturing device is a USB connection, and number of RF protocols are supported. Packets are filtered and decoded by the Packet Sniffer and are then displayed in an easy manner. Options to store and filter are also available.

Features:

- Packet sniffer for:

  - ZigBee and IEEE 802.15.4 networks.
  - RF4CE networks
  - Bluetooth® low energy networks.
  - Generic protocols (raw packet data)
  - SimpliciTI™ networks.

- Hidden and displayed fields selection
- With captured packets, save or open a file.
- Packet details by display of raw data received by the radio.
- Received packets timestamping.

- Packet filtering display
- An address book comprising of known network nodes
- For real-time monitoring of packets, the possibility of forwarding the captured data to a UDP socket
- Display of packets in received sequence displayed by a simple time line

In order to get started with SmartRF Packet Sniffer, refer to the user manual under the "User Guides" heading at http://www.ti.com/tool/PACKET-SNIFFER.

6.3 Testing & Debugging

This heading describes the testing and debugging process for hardware implementation of ZigBee Standard.

6.3.1 Protocol Porting

Procedure

The porting of the (debugged) open source protocol stack is split into two steps, which are as follows:

Software Simulation and creation of HEX File in IAR

After selecting Z-Stack, an open source ZigBee protocol stack (refer to Chapter 5), the protocol is made ready for implementing on an embedded system. The selected SOC CC-2531 RF Transceiver (refer to Chapter 4: Hardware and Software Selection) needs to be ported with the required firmware (i-e ZigBee Protocol). For that purpose, a hex file is generated using IAR Embedded Workbench. The procedure is explained in the user manual; refer to "IAR C/C++ Development Guide" at https://www.iar.com/support/user-guides/user-guide-iar-embedded-workbench-for-arm/. After you generate the HEX file to a known location, move to the next step: Protocol Porting (Uploading the HEX File).

Uploading HEX file in SmartRF Programmer

The SmartRF Flash Programmer (refer to "Tools and Software" heading in this chapter) is used to port the hex file on to the embedded chip CC-2531 containing the SOC (a combination of an 8051 Microcontroller and an RF Chip).

Following are the steps to be performed in Smart RF Flash Programmer:

- Connect the CC-Debugger to your PC
- Power-up the CC-2531 transceiver to another PC
- Open the SmartRF Flash Programmer on the PC connected to CC-Debugger

- The light on CC-Debugger should turn green. If it is still red, press the reset button on CC-Debugger; the light should turn green. If the problem persists, go to Device Manager and click at Cebal Connected Devices. Remove the warning sign, if any.
- Once the CC-debugger is correctly configured, in Flash programmer, click at “Search IEEE” in the Flash Programmer Software. If there appears an IEEE address in front of it, the CC-2531 is connected. Otherwise, remove the CC-2531 and connect it again.
- Now click at “Erase Memory” in the SmartRF Flash Programmer.
- Browse for the hex file, wherever you generated it. You can also use the “ZNP with SBL” file in the projects in Z-Stack folder. This file is supplied by the Texas Instruments themselves
- Click at “Erase, Program, and Verify”. Once the programming is complete, you CC-2531 transceiver becomes a fully compliant ZigBee device.

The steps can also be seen in the Flash Programmer User Manual.

6.3.2 Creation of a Wireless Sensor (Mesh) Network

After porting the CC-2531 with the ZigBee Protocol Stack firmware, the next step is communication between ZigBee modules (Wireless Sensor Nodes) in a Network. However, it requires creation of a network at first.

Create Coordinator

The first step in creation of a network is the creation of a coordinator. The procedure for configuring a CC-2531 ZigBee device as a coordinator is as follows:

- Power up your ZigBee device to your computer through the USB port.
- Wait for the green light to turn off.
- Once the light has gone, open the Z-Tool (refer to Z-Tool sub-heading in this chapter)
- Run the following commands in the Z-Tool to configure your device as a coordinator.

o SYS_OSAL_NV_WRITE

Id: 0x0087

Offset: 0x00

Len: 0x01

Value: (0x00)

o APP_CNF_BDB_SET_CHANNEL

isPrimary: TRUE (0x1)

Channel: CHNL_0x00002000 (0x2000)

- APP_CNF_BDB_SET_CHANNEL

  isPrimary: FALSE (0x0)

  Channel: NONE (0x0)

- APP_CNF_BDB_START_COMMISSIONING

  Commissioning Mode: (0x04) Network Formation (0x4)

- SYS_OSAL_NV_WRITE

  Id: 008F

  Offset: 0x00

  Len: 0x01

  Value: 0x00

- APP_CNF_BDB_START_COMMISSIONING (0x2F05)

  Commissioning Mode: (0x02) Network Steering

As soon as the coordinator is configured, the network forms by itself and each device gets a PAN ID automatically.

Create Router (Write Commands)

In order to configure another device, which you may connect to the same computer or another one. Z-Tool opens another window for if another CC-2531 ZigBee device is connected to some other COM port on the same PC, allowing to connect as many CC-2531 devices as the number of USB ports on your computer.

- SYS_OSAL_NV_WRITE

  Id: 0x0087

  Offset: 0x00

  Len: 0x01

  Value: (0x01)

- APP_CNF_BDB_SET_CHANNEL

  isPrimary: TRUE (0x1)

  Channel: CHNL_0x00002000 (0x2000)

- APP_CNF_BDB_SET_CHANNEL

  isPrimary: FALSE (0x0)

  Channel: NONE (0x0)

- APP_CNF_BDB_START_COMMISSIONING (0x2F05)

  Commissioning Mode: (0x02) Network Steering

If the router is successful in joining the network, a "message incoming" dialogue appears on the coordinator side. Now run the get device info command under "Util"; this will give you the information about the router. Source address will be required later for data transmission from coordinator to the router.

IMPORTANT TIP:

It must be ensured that the "network steering" command on the router side is run within 260 seconds after the "network steering" command is run on the coordinator side. After this interval, the network doesn't allow joining devices unless used Permit_Join_Request which will later be discussed.

6.3.3 Communication between two nodes

Data transmission between two nodes is done using the Application Framework (AF) commands.

Communication in decimal

Following are the steps for sending data in decimal format

- Run the AF_Register command with following parameters on both the coordinator and the router.

- AF_REGISTER (0x2400)

  EndPoint: 0x01

AppProfID: 0x0104

AppDeviceId: 0x0100

AppDevVer: 0x00

LatencyReq: NO_LATENCY_REQS (0x0)

AppNumInClusters: 0x02

AppInClusterList: 0x0000, 0x0006

AppNumOutClusters: 0x02

AppOutClusterList: 0x0000, 0x0006

- Run the AF_DataRequest command on the coordinator side with following parameters. The DstAddr here is the "Source address" or "Short Address" of the router that can be seen in Util_get_device_info.

o AF_DATA_REQUEST (0x2401)

DstAddr: 0xDB40

DestEndpoint: 0x01

SrcEndpoint: 0x01

ClusterID: 0x0006

TransID: 0x00

Options: 0x00

Radius: 0x00

Len: 0x03

Data: ... (0x02, 0x02, 0x02)

The data sent in decimal format was (02, 02, 02). You'll see on the router side, the following incoming message

o AF_INCOMING_MSG (0x4481)

GroupID: 0x0000

ClusterID: 0x0006

SrcAddr: 0x0000

SrcEndpoint: 0x01

DstEndpoint: 0x01

WasBroadcast: 0x00

LinkQuality: 0x3F

SecurityUse: 0x00

Timestamp: 0x00156CD2

TransSeqNumber: 0x00

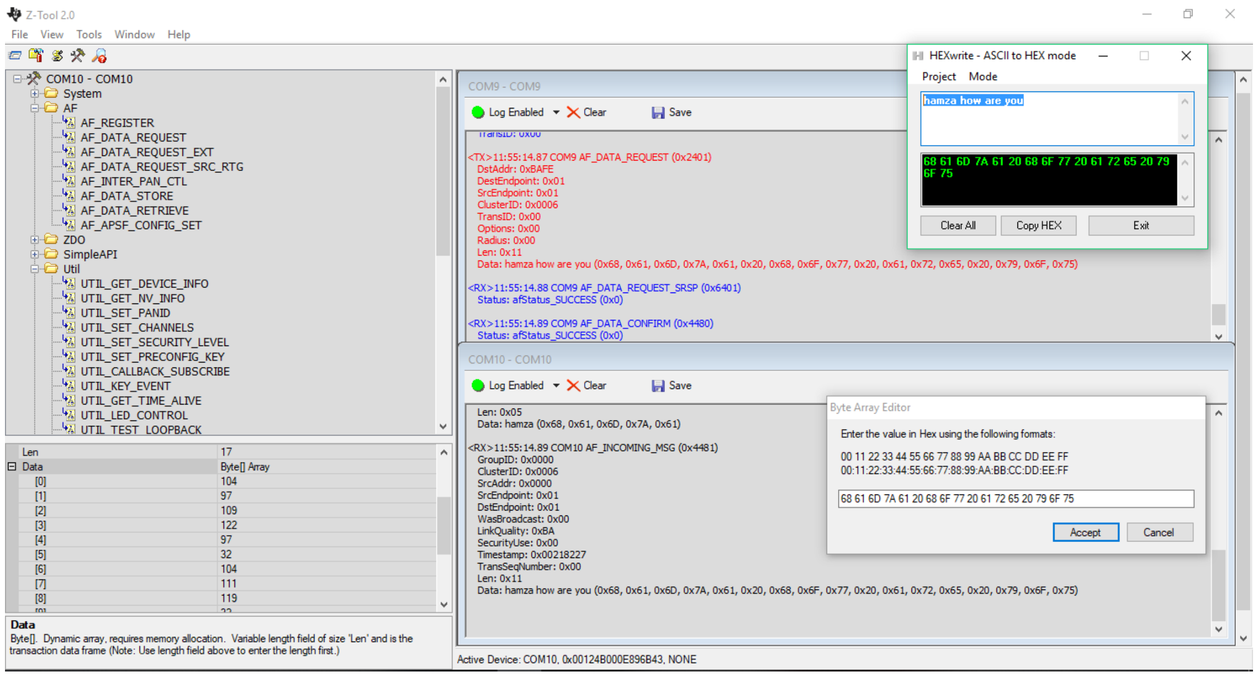
Len: 0x03

Data: ... (0x02, 0x02, 0x02)

String transmission

If you wish data received on the receiver side to be in text format, the process is explained in this heading. The steps are as follows:

- Create a network by configuring the coordinator
- Join the network through router or end device using network steering
- Register using the AF_Register command on both the router and the coordinator
- Use an ASCII to HEX converter to convert the string you want to transmit into HEX format. Copy the hex data
- In AF_Data Request, specify the length depending on the number of HEX data bytes and right click on the ByteArray field
- In the ByteArray field, after right clicking at it, click at edit.
- Now paste the hex you had copied into the small window appearing in front
- Send the data
- The data will now be received on the receiver side in the text format.



Len	17
Data	Byte[] Array
[0]	104
[1]	97
[2]	109
[3]	122
[4]	97
[5]	32
[6]	104
[7]	111
[8]	119

Data
Byte[]. Dynamic array, requires memory allocation. Variable length field of size 'Len' and is the transaction data frame (Note: Use length field above to enter the length first.)

```
<TX>11:55:14.87 COM9 AF_DATA_REQUEST (0x2401)
  DstAddr: 0xBAFE
  DestEndpoint: 0x01
  SrcEndpoint: 0x01
  ClusterID: 0x0006
  TransID: 0x00
  Options: 0x00
  Radius: 0x00
  Len: 0x11
  Data: hamza how are you (0x68, 0x61, 0x6D, 0x7A, 0x61, 0x20, 0x68, 0x6F, 0x77, 0x20, 0x61, 0x72, 0x65, 0x20, 0x79, 0x6F, 0x75)

<RX>11:55:14.88 COM9 AF_DATA_REQUEST_SRSP (0x6401)
  Status: afStatus_SUCCESS (0x0)

<RX>11:55:14.89 COM9 AF_DATA_CONFIRM (0x4480)
  Status: afStatus_SUCCESS (0x0)
```

```
hamza how are you
```

```
68 61 6D 7A 61 20 68 6F 77 20 61 72 65 20 79
6F 75
```

```
  Len: 0x05
  Data: hamza (0x68, 0x61, 0x6D, 0x7A, 0x61)

<RX>11:55:14.89 COM10 AF_INCOMING_MSG (0x4481)
  GroupID: 0x0000
  ClusterID: 0x0006
  SrcAddr: 0x0000
  SrcEndpoint: 0x01
  DstEndpoint: 0x01
  WasBroadcast: 0x00
  LinkQuality: 0xBA
  SecurityUse: 0x00
  Timestamp: 0x00218227
  TransSeqNumber: 0x00
  Len: 0x11
  Data: hamza how are you (0x68, 0x61, 0x6D, 0x7A, 0x61, 0x20, 0x68, 0x6F, 0x77, 0x20, 0x61, 0x72, 0x65, 0x20, 0x79, 0x6F, 0x75)
```

Enter the value in Hex using the following formats:

```
00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF
00:11:22:33:44:55:66:77:88:99:AA:BB:CC:DD:EE:FF
```

```
68 61 6D 7A 61 20 68 6F 77 20 61 72 65 20 79 6F 75
```

Active Device: COM10, 0x00124B000E896B43, NONE

Figure 24: Data Transmission in HEX format

6.3.4 Packet Sniffing using Packet Sniffer

In this heading, the packet sniffing procedure is explained. Packet sniffing is done using SmartRF Packet Sniffer Firmware supplied by the Texas Instruments and SmartRF Packet Sniffer Software (PC Tool). The firmware allows to convert any transceiver into a packet sniffer using the SmartRF Flash Programmer and then use the Packet Sniffer Software to analyze the packets.

For displaying and storing RF packets captured, the software application Packet Sniffer is used, which supports multiple RF protocols. Packet are first filtered, decoded, and then displayed in an easy to understand manner. It also provides options for storage to a binary file. Packets are stored on the disk on the system (PC) side. The size of the hard disk and the packet size determine the total allowed number of packets which can be stored. RAM buffer has cached packets during the operation. While GUI has to display the packets, this is done to make the access time better.

Firmware Porting

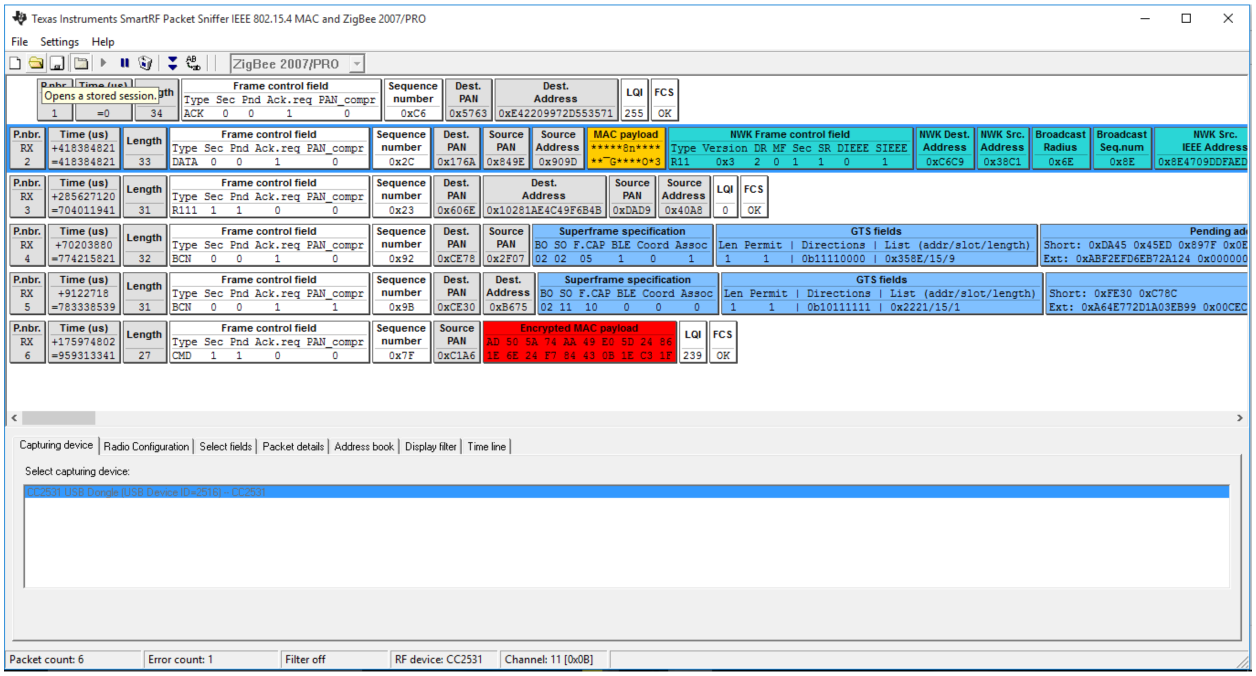
In this project, the CC-2531 was ported with the sniffer firmware in order to perform sniffing in the network of wireless sensor nodes. First of all, the hex file supplied by Texas Instruments. The hex file can be found in the SmartRF Packet Sniffer Software folder after you install it; the packet sniffer firmware can be downloaded from http://www.ti.com/tool/PACKET-SNIFFER. For CC-2531, use “Packet Sniffer” not “Packet Sniffer-2”.

Channel selection

Remember that the channel selected while configuring the router was 0x2000 which in decimal "Channel 13". Therefore, select the "channel 13" for correct sniffing of packets. Otherwise, you'll receive very few packets of junk data.

Packet Sniffing using SmartRF Sniffer

Once you select the COM Port and the Channel, start the packet capturing. The software in conformance with the CC-2531 Packet Sniffer firmware loaded device will start capturing packets. With minimal information required, it is possible to get a lot of information about the ZigBee devices communicating in the network.



Texas Instruments SmartRF Packet Sniffer IEEE 802.15.4 MAC and ZigBee 2007/PRO

```
P.nbr. | Time (us) | Length | Frame control field                    | Sequence | Dest.  | Dest.              | LQI | FCS
       |           |        | Type Sec Pnd Ack.req PAN_compr         | number   | PAN    | Address            |     |
1      | =0        | 34     | ACK  0   0   1       0                 | 0xC6     | 0x5763 | 0xE42209972D553571 | 255 | OK

P.nbr. | Time (us)  | Length | Frame control field            | Sequence | Dest.  | Source | Source  | MAC payload | NWK Frame control field                        | NWK Dest. | NWK Src. | Broadcast | Broadcast | NWK Src.
RX     | +418384821 |        | Type Sec Pnd Ack.req PAN_compr | number   | PAN    | PAN    | Address | *****8n**** | Type Version DR MF Sec SR DIEEE SIEEE          | Address   | Address  | Radius    | Seq.num   | IEEE Address
2      | =418384821 | 33     | DATA 0   0   1       0         | 0x2C     | 0x176A | 0x849E | 0x909D  | **¯G****O*3 | R11  0x3     2  0  1   1  0     1              | 0xC6C9    | 0x38C1   | 0x6E      | 0x8E      | 0x8E4709DDFAED

P.nbr. | Time (us)  | Length | Frame control field            | Sequence | Dest.  | Dest.              | Source | Source  | LQI | FCS
RX     | +285627120 |        | Type Sec Pnd Ack.req PAN_compr | number   | PAN    | Address            | PAN    | Address |     |
3      | =704011941 | 31     | R111 1   1   0       0         | 0x23     | 0x606E | 0x10281AE4C49F6B4B | 0xDAD9 | 0x40A8  | 0   | OK

P.nbr. | Time (us)  | Length | Frame control field            | Sequence | Dest.  | Source | Superframe specification     | GTS fields                                           | Pending ad
RX     | +70203880  |        | Type Sec Pnd Ack.req PAN_compr | number   | PAN    | PAN    | BO SO F.CAP BLE Coord Assoc  | Len Permit | Directions | List (addr/slot/length) | Short: 0xDA45 0x45ED 0x897F 0x0E
4      | =774215821 | 32     | BCN  0   0   1       0         | 0x92     | 0xCE78 | 0x2F07 | 02 02  05    1    0     1    | 1   1      | 0b11110000 | 0x358E/15/9             | Ext: 0xABF2EFD6EB72A124 0x000000

P.nbr. | Time (us)  | Length | Frame control field            | Sequence | Dest.  | Dest.   | Superframe specification     | GTS fields                                           |
RX     | +9122718   |        | Type Sec Pnd Ack.req PAN_compr | number   | PAN    | Address | BO SO F.CAP BLE Coord Assoc  | Len Permit | Directions | List (addr/slot/length) | Short: 0xFE30 0xC78C
5      | =783338539 | 31     | BCN  0   0   1       1         | 0x9B     | 0xCE30 | 0xB675  | 02 11  10    0    0     0    | 1   1      | 0b10111111 | 0x2221/15/1             | Ext: 0xA64E772D1A03EB99 0x00CEC

P.nbr. | Time (us)  | Length | Frame control field            | Sequence | Source | Encrypted MAC payload         | LQI | FCS
RX     | +175974802 |        | Type Sec Pnd Ack.req PAN_compr | number   | PAN    | AD 50 5A 74 AA 49 E0 5D 24 86 |     |
6      | =959313341 | 27     | CMD  1   1   0       0         | 0x7F     | 0xC1A6 | 1E 6E 24 F7 84 43 0B 1E C3 1F | 239 | OK
```

Select capturing device:

CC2531 USB Dongle (USB Device ID=2516) -- CC2531

Packet count: 6 | Error count: 1 | Filter off | RF device: CC2531 | Channel: 11 [0x0B]

Figure 25: Packet Capturing of ZigBee Devices Communicating in a Network

- **Uses**

Packet sniffing in a network finds many uses, the most prominent of which are explained later in the text.

Verification of encryption

The packet sniffer is able to capture packets but if they are encrypted, the person sniffing the packets will not be able to easily take meaningful information. The network that was sniffed in this project was based on ZigBee Protocol by Texas Instruments which has AES-128 encryption in it. It was prominent when the packets were captured that the payload was appearing encrypted in the form of asterisks or data that wasn't actually transmitted. However, the network address and PAN ID were showing correctly.

Research and Development

The packet sniffing capability can be used to make the network even secure, which opens gates for research and development in the area.

Chapter 7: Customizations, Impact & Achieved Goals, Future Prospects

7.1 Customizations

7.1.1 Network Layer

In the network layer, the stack hardening has been done as a customization.

Stack Hardening (Changing Key)

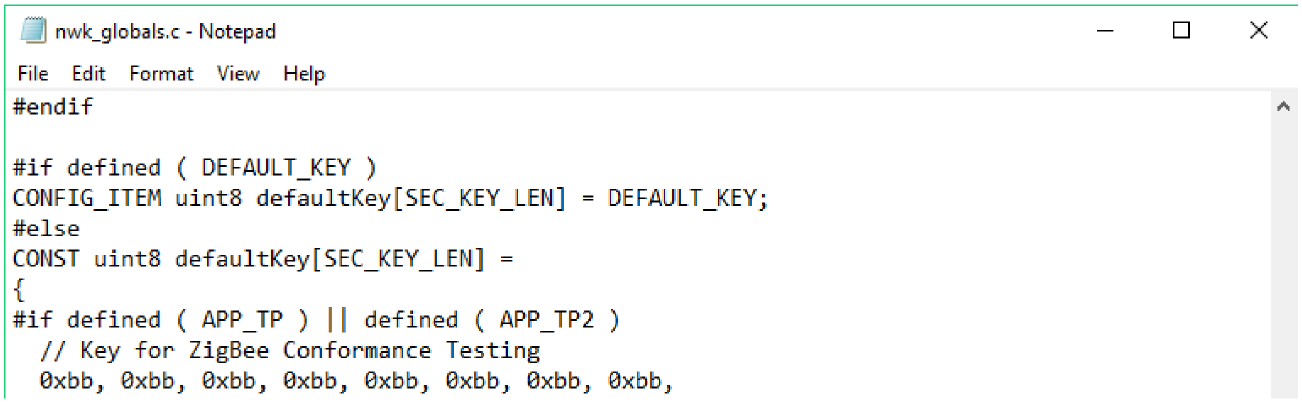
What is Stack Hardening?

Stack hardening is a process of enhancing security by customizations or modifications in the Security Layer (MAC Layer in this case). It results in our network operating in more secure environment as a result keeping it safe for multiple kind of attacks, phishing, eavesdropping etc. It is an important process since most of the communication is confidential and if leaked, it could cause a lot of damage. Stack hardening has multiple types, it could be a change in Encryption schemes (AES, DES), it could be a change in the Key.

How it is modified in our Project

Another one of the extended objectives of the project was to gain control of the security layer of the ZigBee protocol stack. To make it possible, we explored the security layer of the acquired ZigBee protocol stack and identified that Trust Center Link Key (TCLK) can be changed, providing control over the encryption. User can change the encryption key and set the desired key. This Trust Center Link Key is the first key used when a network is formed.

The preconfigured key for network layer encryption can be changed in the function nwk_globals.c in the Z-Stack. However, it must be ensured that the key remains the exact same in all the communicating devices.



```
#endif

#if defined ( DEFAULT_KEY )
CONFIG_ITEM uint8 defaultKey[SEC_KEY_LEN] = DEFAULT_KEY;
#else
CONST uint8 defaultKey[SEC_KEY_LEN] =
{
#if defined ( APP_TP ) || defined ( APP_TP2 )
  // Key for ZigBee Conformance Testing
  0xbb, 0xbb, 0xbb, 0xbb, 0xbb, 0xbb, 0xbb, 0xbb,
```

Figure 26: Key structure in MAC Layer on Z-Stack

7.1.2 MAC Layer

MAC Binding and Joining Permission

Why Needed?

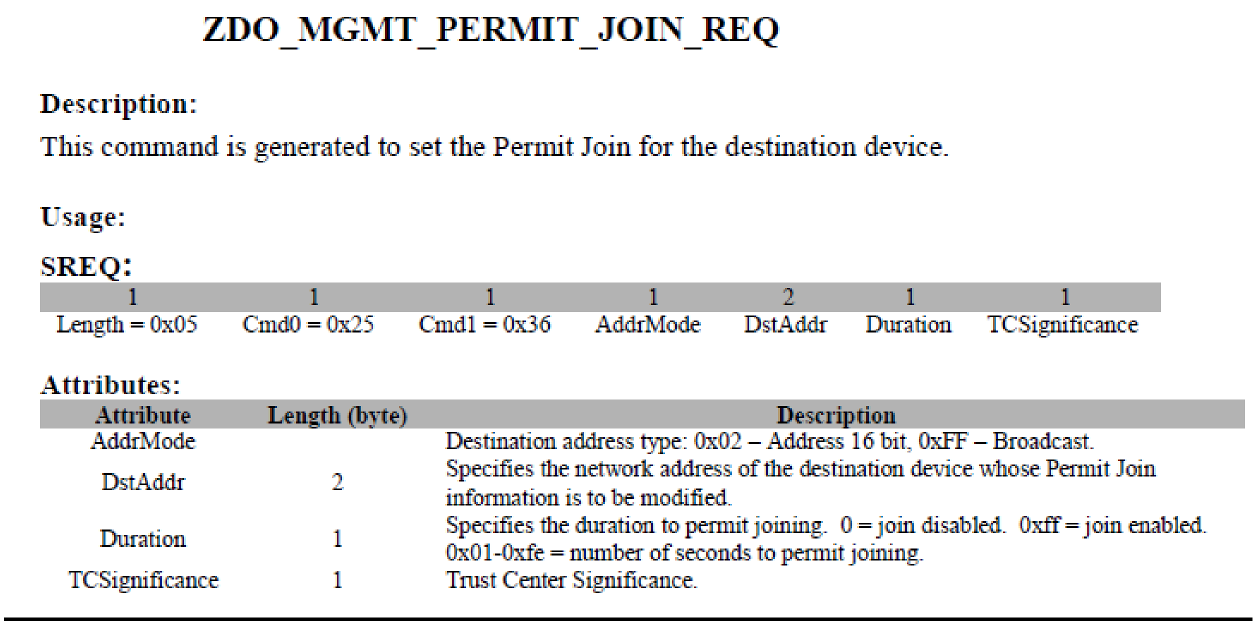
After making a ZigBee network whenever Coordinator or Router sends commands to each other, if anyone in the area is listening to the conversation using Packet Sniffer, that person will get to know whenever Coordinator allows Routers to join the network. As a result, that person can also join the network since there is no key or security measure involved. To prevent this problem, we proposed a solution "MAC Binding"

What is MAC binding?

MAC Binding is a process in which the MAC address of a router or an end device is statically saved by the coordinator. In DHCP servers, MAC binding saves the MAC address of the client such that each time the client connects to that server it is assigned the same IP-address. In 802.15.4, a similar procedure happens. MAC binding is used so that each time an end device joins a parent node, it is assigned the same network address by the coordinator.

In Wi-Fi, the server can however be configured such that only those clients whose MAC address have been saved are able to join the network. This, however, is not possible within the ZigBee network. In the ZigBee network, the coordinator creates a table where it saves the MAC addresses of its end devices. This is handy in situations where the end device is mobile and will be joining multiple parent nodes. By binding the MAC address of the end device to the parent nodes, it will easily connect to those coordinators instead of initializing a joining procedure each time it moves to a new coordinator.

MAC Binding is a highly effective method to stop man in the middle attacks from occurring. But since ZigBee networks don't have such an option a different method was implemented. Whenever a coordinator creates a network, any end device that know which the channel the coordinator is using, they can easily join the network. But a coordinator can be set to allow the end devices to join the network for a specific time ranging from 0-255 seconds. After the time limit coordinator doesn't allow any end devices to join the network until or unless the coordinator isn't given a command to start accepting new end devices in the network. The command ***"ZDO_MGMT_PERMIT_JOIN_REQ"*** asks for the MAC address of the end device that wants to join and the time limit in which it can connect to coordinator before it can be sent to the coordinator. Since this command only allows the end device that has the same MAC address as the one that was given to the command to join, this prevents any unknown end devices to try to join the network and so eliminates the possibility of any man in the middle attack.



ZDO_MGMT_PERMIT_JOIN_REQ

Description:

This command is generated to set the Permit Join for the destination device.

Usage:

SREQ:

1	1	1	1	2	1	1
Length = 0x05	Cmd0 = 0x25	Cmd1 = 0x36	AddrMode	DstAddr	Duration	TCSignificance

Attributes:

Attribute	Length (byte)	Description
AddrMode		Destination address type: 0x02 – Address 16 bit, 0xFF – Broadcast.
DstAddr	2	Specifies the network address of the destination device whose Permit Join information is to be modified.
Duration	1	Specifies the duration to permit joining. 0 = join disabled. 0xff = join enabled. 0x01-0xfe = number of seconds to permit joining.
TCSignificance	1	Trust Center Significance.

Figure 27: Attributes of ZDO_MGMT_PERMIT_JOIN_REQ

There are three simple thing required for this progess

- Destination Address
- IEEE Address
- Time Period in which that specific device are allowed to join

7.2 Impact & Goals Achieved

7.2.1 Impact

By using an open source Z Stack and hardware of our choosing, this project has created a ZigBee that is very low in cost, highly customizable according to user’s needs and functionality. The Stack can be changed as per the requirements of the user. The cost for buying a fully customizable ZigBee has been reduced by a lot now that users only have to pay for the cost of the hardware which is very cheap compared to the Stack. Users can fully customize the stack in accordance to the type of hardware they’re using and the type of function the ZigBee has.

7.2.2 Goals Achieved

- Literature review
- Selection of open-Source stack from all stacks available
- Selection of hardware
- Hands-on working with ZigBee by creating a network of nodes
- Simulation of ZigBee stack in software to implement ZigBee stack on USRP

- Hardware implementation of ZigBee protocol stack
- Successful Debugging and Testing of Stack on hardware
- Creation of a ZigBee network
- Communication between two nodes in the CC-2531 ZigBee modules
- String data transmission in a network
- Stack hardening
- Joining Permissions
- Packet Sniffing of a ZigBee Network

7.3 Future Prospects

7.3.1 MAC & Security Layer - New/customized encryption techniques

This project has been able find a cost-efficient solution for creating wireless sensor networks of ZigBee modules. In addition to that, it has also been able to introduce some modifications (explained in Chapter 6). Nevertheless, there are several things that can be introduced in the future, two of which are MAC and Security layer modifications. In MAC and Security layer, encryption schemes can be tweaked since the protocol is customizable. The key exchange methods can also be modified. In addition to that, only selected devices can also be allowed to join the network using install codes.

7.3.2 PHY Layer - changes & required hardware changes accordingly

The modulation scheme currently employed in Z-Stack is O-QPSK. New modulation scheme can be introduced by working on the Physical layer. However, this may require hardware changes as well.

7.3.3 Network Layer – Topologies

At present, Mesh network topology is implemented Z-Stack, which can be altered to any desired network topology by working on the network layer.

7.3.4 Audio/File Transmission

Currently, this project demonstrates the transmission of data in text or HEX format, but the functionality can be enhanced by writing a zjs script for transmission of a text file or a small audio file.

Bibliography

Amaro, J. P. F. J. F. R. C. a. J. L., 2012. Powering Wireless Sensor Networks Nodes for Complex Protocols on Harvested Energy.

Anon., March 2010. Adaptive Network Solutions Unveils Europe's First Sub-1 GHz IEEE 802.15.4.

Anon., n.d. *Wireless Solutions.* [Online]
Available at: https://wireless-solutions.de/

AN-Solutions, n.d. [Online]
Available at: http://www.an-solutions.de/products/900_mhz/900_zigbee_modules.html

Bastian Bloessl, C. L. F. D. a. C. S., 2013. [Online]
Available at: www.ccs-labs.org/bib/bloessl2013gnu/bloessl2013gnu.pdf

Dalibor Dobrilovic, Z. S. V. B. Z. C. N. B., 2014. Software Application for Analyzing ZigBee Network Performance in University Courses.

David Martins, H. G., 2010. Attacks with Steganography in PHY and MAC Layers of 802.15.4 Protocol.

Digi, n.d. *Digi Official Site.* [Online]
Available at: www.digi.com

Fei, Y. Y. S. L. X. a. G. S., 2014. Micro-IMU based Wireless Body Sensor Network.

IEEE, 2006. [Online]
Available at: https://standards.ieee.org/findstds/standard/802.15.4-2006.html

IEEE, 2015. [Online]
Available at: https://standards.ieee.org/findstds/standard/802.15.4-2015.html

Instruments, T., 2016. *Developer's Guide.* [Online]
Available at: http://www.ti.com/tool/Z-STACK

Instruments, T., 2016. *OSAL API.* [Online]
Available at: http://www.ti.com/tool/Z-STACK

Instruments, T., 2016. *Z-Stack API.* [Online]
Available at: http://www.ti.com/tool/Z-STACK

Instruments, T., 2016. *Z-Stack Monitor & Test API.* [Online]
Available at: http://www.ti.com/tool/Z-STACK

Laura Victoria Escamilla Del Río, J. M. G. D., 2014. Ad Hoc Communications for Wireless Robots in Indoor Environments.

Lee, J.-S., 2005. An Experiment on Performance Study of IEEE 802.15.4 Wireless Networks.

Lucchi, M., 2011. Cooperative communication and distributed detection in wireless sensor networks.

Site, C., n.d. *Set-up a ZigBee Network.* [Online]
Available at: https://sunmaysky.blogspot.com/2017/02/use-ztool-z-stack-30-znp-to-set-up.html

US, T., n.d. [Online]
Available at: http://www.theseus.fi/

Wiki, E. E., n.d. [Online]
Available at: eewiki.net

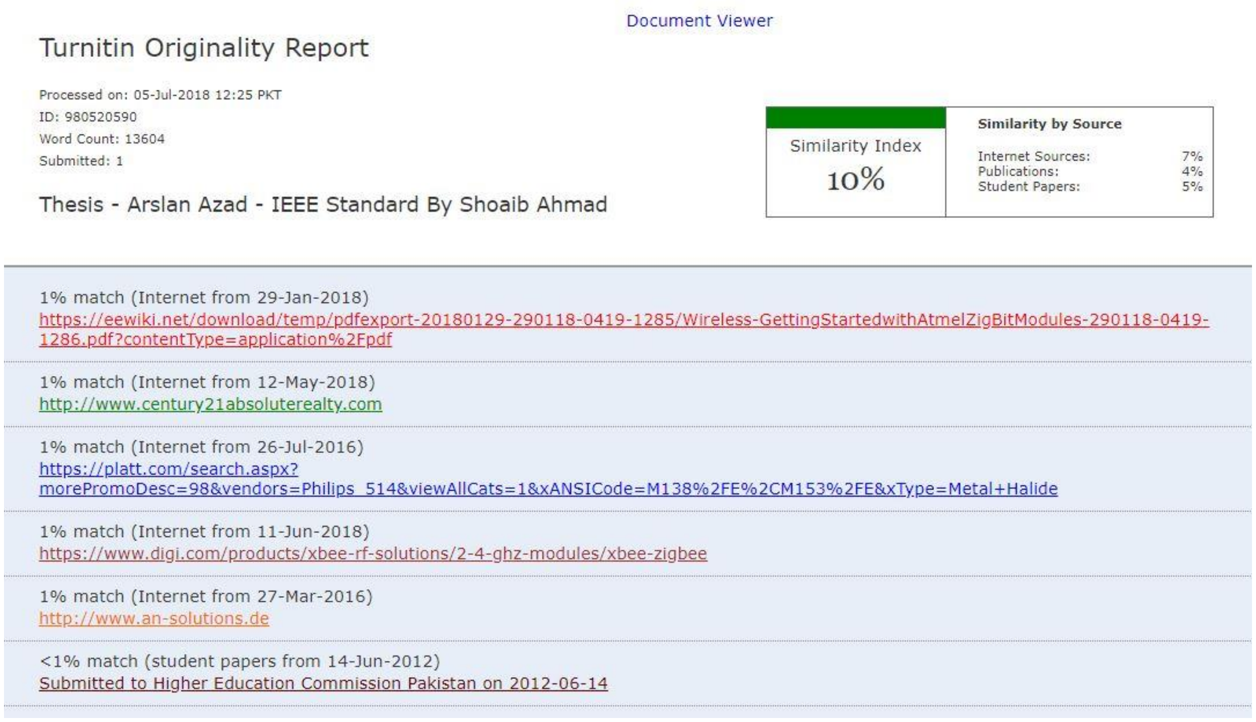
Wikipedia, n.d. [Online]
Available at: https://en.wikipedia.org/wiki/IEEE_802.15.4

Xia, F. a. A. R., 2015. Evaluating IEEE 802.15.4 for CPS.

Ying-Chih Chen, P.-Y. C. C.-Y. W., 2016. Request-Centric Wireless Bus Information Management System.

ZBoss, n.d. *ZBoss Open Source Stack.* [Online]
Available at: www.zboss.dsr-wireless.com

Plagiarism Report



Document Viewer

Turnitin Originality Report

Processed on: 05-Jul-2018 12:25 PKT
ID: 980520590
Word Count: 13604
Submitted: 1

Thesis - Arslan Azad - IEEE Standard By Shoaib Ahmad

Similarity Index	Similarity by Source	
10%	Internet Sources:	7%
	Publications:	4%
	Student Papers:	5%

1% match (Internet from 29-Jan-2018)
https://eewiki.net/download/temp/pdfexport-20180129-290118-0419-1285/Wireless-GettingStartedwithAtmelZigBitModules-290118-0419-1286.pdf?contentType=application%2Fpdf

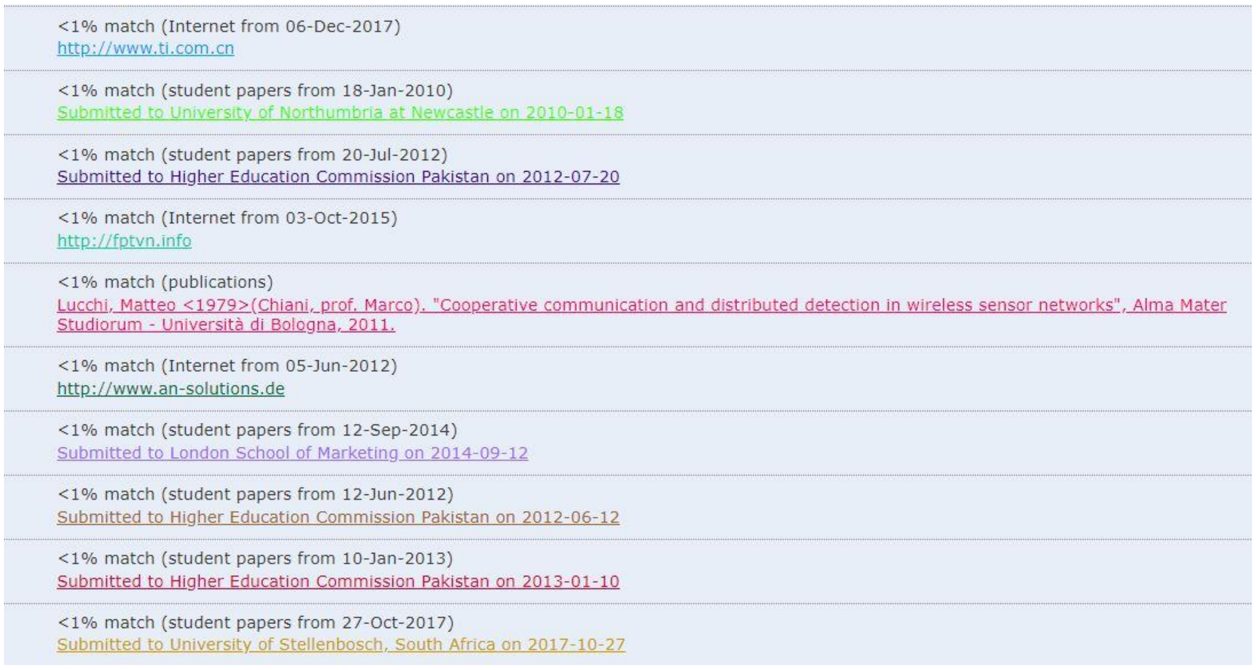
1% match (Internet from 12-May-2018)
http://www.century21absoluterealty.com

1% match (Internet from 26-Jul-2016)
https://platt.com/search.aspx?morePromoDesc=98&vendors=Philips_514&viewAllCats=1&xANSICode=M138%2FE%2CM153%2FE&xType=Metal+Halide

1% match (Internet from 11-Jun-2018)
https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-zigbee

1% match (Internet from 27-Mar-2016)
http://www.an-solutions.de

<1% match (student papers from 14-Jun-2012)
Submitted to Higher Education Commission Pakistan on 2012-06-14



<1% match (Internet from 06-Dec-2017)
http://www.ti.com.cn

<1% match (student papers from 18-Jan-2010)
Submitted to University of Northumbria at Newcastle on 2010-01-18

<1% match (student papers from 20-Jul-2012)
Submitted to Higher Education Commission Pakistan on 2012-07-20

<1% match (Internet from 03-Oct-2015)
http://fptvn.info

<1% match (publications)
Lucchi, Matteo <1979>(Chiani, prof. Marco). "Cooperative communication and distributed detection in wireless sensor networks", Alma Mater Studiorum - Università di Bologna, 2011.

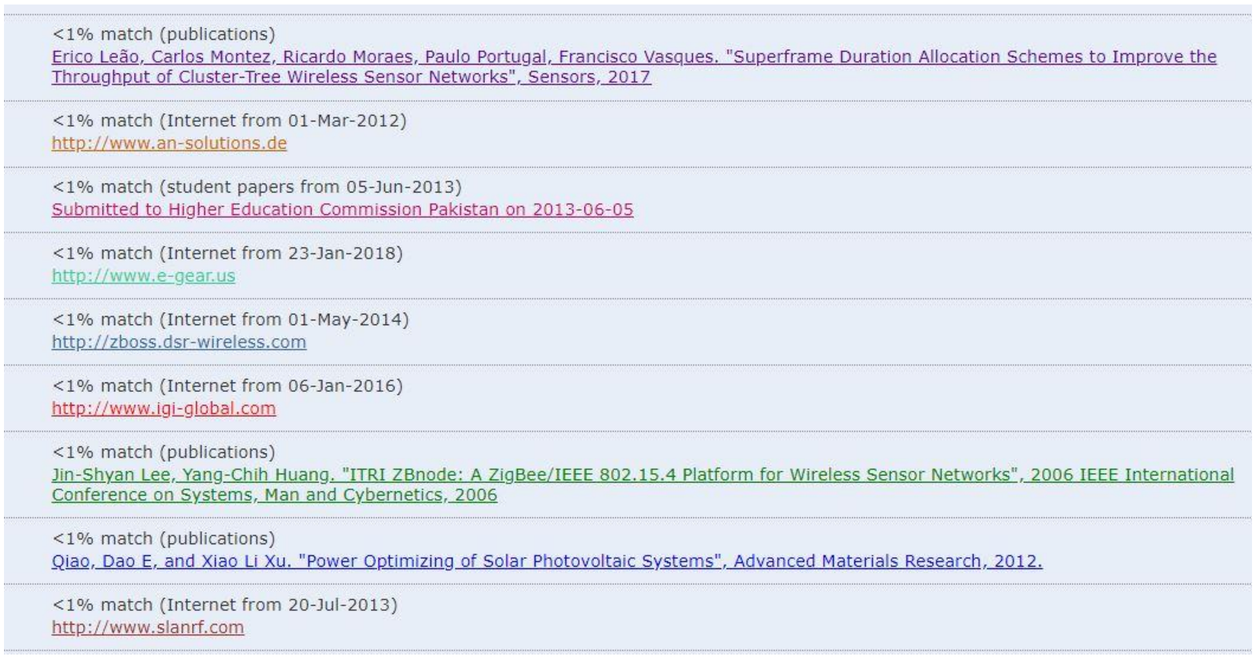
<1% match (Internet from 05-Jun-2012)
http://www.an-solutions.de

<1% match (student papers from 12-Sep-2014)
Submitted to London School of Marketing on 2014-09-12

<1% match (student papers from 12-Jun-2012)
Submitted to Higher Education Commission Pakistan on 2012-06-12

<1% match (student papers from 10-Jan-2013)
Submitted to Higher Education Commission Pakistan on 2013-01-10

<1% match (student papers from 27-Oct-2017)
Submitted to University of Stellenbosch, South Africa on 2017-10-27



<1% match (publications)
Erico Leão, Carlos Montez, Ricardo Moraes, Paulo Portugal, Francisco Vasques. "Superframe Duration Allocation Schemes to Improve the Throughput of Cluster-Tree Wireless Sensor Networks", Sensors, 2017

<1% match (Internet from 01-Mar-2012)
http://www.an-solutions.de

<1% match (student papers from 05-Jun-2013)
Submitted to Higher Education Commission Pakistan on 2013-06-05

<1% match (Internet from 23-Jan-2018)
http://www.e-gear.us

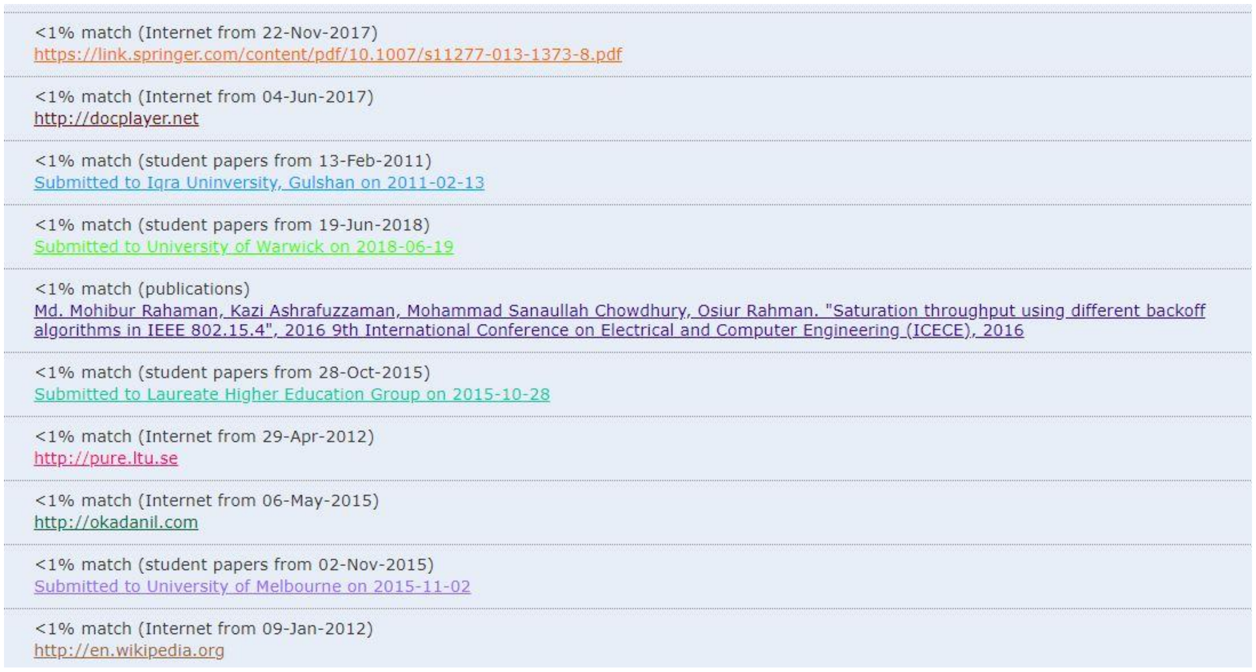
<1% match (Internet from 01-May-2014)
http://zboss.dsr-wireless.com

<1% match (Internet from 06-Jan-2016)
http://www.igi-global.com

<1% match (publications)
Jin-Shyan Lee, Yang-Chih Huang. "ITRI ZBnode: A ZigBee/IEEE 802.15.4 Platform for Wireless Sensor Networks", 2006 IEEE International Conference on Systems, Man and Cybernetics, 2006

<1% match (publications)
Qiao, Dao E, and Xiao Li Xu. "Power Optimizing of Solar Photovoltaic Systems", Advanced Materials Research, 2012.

<1% match (Internet from 20-Jul-2013)
http://www.slanrf.com



<1% match (Internet from 22-Nov-2017)
https://link.springer.com/content/pdf/10.1007/s11277-013-1373-8.pdf

<1% match (Internet from 04-Jun-2017)
http://docplayer.net

<1% match (student papers from 13-Feb-2011)
Submitted to Iqra Uninversity, Gulshan on 2011-02-13

<1% match (student papers from 19-Jun-2018)
Submitted to University of Warwick on 2018-06-19

<1% match (publications)
Md. Mohibur Rahaman, Kazi Ashrafuzzaman, Mohammad Sanaullah Chowdhury, Osiur Rahman. "Saturation throughput using different backoff algorithms in IEEE 802.15.4", 2016 9th International Conference on Electrical and Computer Engineering (ICECE), 2016

<1% match (student papers from 28-Oct-2015)
Submitted to Laureate Higher Education Group on 2015-10-28

<1% match (Internet from 29-Apr-2012)
http://pure.ltu.se

<1% match (Internet from 06-May-2015)
http://okadanil.com

<1% match (student papers from 02-Nov-2015)
Submitted to University of Melbourne on 2015-11-02

<1% match (Internet from 09-Jan-2012)
http://en.wikipedia.org