VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Shahid Mahmood

SOFTWARE DEFINED RADIO FOR WIRELESS SENSOR &COGNITIVE NETWORKS

Faculty of Information Technology
2011

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Programme in Information Technology

ABSTRACT

Author	Shahid Mahmood
Title	Software Defined Radio for Wireless Sensors & Cognitive Networks
Year	2011
Language	English
Pages	68
Name of Supervisor	Gao Chao

Software Defined Radio is a communication system where major parts of the signal processing are done with the help of software compare to the application specific hardware. This kind of system consists of computer, a radio frequency front-end and Analog-to-Digital Converter and Digital-to-Analog Converter. Software Defined Radio modifies new protocols which are flexible and quick to deploy [1].

The ComNet department at Aalto University is exploring the feasibility of using Software Defined Radio for sensor and cognitive networks. The aim is to develop a flexible platform for verifying the customized MAC and PHY layer protocols and algorithms.

The aim of this project is to integrate the SDR nodes into our wireless sensor network test-bed that uses Sensinode [2] sensor nodes. Our SDR nodes are comprised of USRP2 hardware [3] and GNU radio [4] software. The Sensinode nodes are IEEE 802.15.4 compliant and use CC2420/CC2431 [5] transceiver. Provided the SDR nodes can be integrated into the sensor network without any loss of performance, it will give us the opportunity to enhance the functionality of the network by integration of new PHY layer algorithms and techniques such as interference cancellation [6], network cooperation and spectrum sensing for interference avoidance.

This thesis focuses on understanding the Software Defined Radio components (GNU Radio and USRP2), architecture of the operating system, signal processing blocks, creating the desired application-specific scenarios, installation and configuration of the SDR in Linux environment, integration of IEEE 802.15.4 standard into SDR, and configuring the SDR to enable communication with existing sensor networks platform.

A Software Defined Radio running IEEE 802.15.4 was configured as a receiver and it received IEEE 802.15.4 packets from an existing sensor platform. The task was successfully completed and detailed report is in results.

Keywords		Learning, stimulation, teaching methods

CONTENTS

ACKNOWLEDGEMENT

This Bachelor thesis is the most challenging project i have done during my studies at Vaasa University of Applied Sciences.

First of all, I would like to thank Almighty God, the most merciful, the most beneficent for his guidance and blessings in making this thesis successful.

I would like to express my deep gratitude to my supervisor Gao Chao for his time, detailed and constructive comments, feedback and for his important support throughout this project.

I am heartily grateful to Aamir Mahmood, ComNet department of Aalto University, who was always helpful by all the means to complete this project.

I would like to thanks to my beloved Mother from the bottom of my hearts for her unconditional love and support, i would also like to thanks all of my friends and families who help me in any way during my stay in Vaasa. Last but certainly not least I would like to thanks my wife for all the support she given me during this time.

Vaasa, 14, June 2011.

Shahid Mahmood.

ABBREVIATIONS

ADC	Analog-to-Digital Converter
AFH	Adaptive Frequency Hopping
AP	Access Point
CIC	Cascade Integration Comb
CLI	Command Line Interface
ComNet	Communication and Networking
CPLD	Programmable Logic Device
CR	Cognitive Radio
CRA	Cognitive Radio Architecture
DAC	Digital-to-Analog Converter
DDC	Digital Down Converter
DFS	Dynamic Frequency Selection
DUC	Digital Up Convertor
FCC	Federal Communications Commission
FCF	Frame Control Field
FCS	Frame Check Sequence
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
FTDI	Future Technology Device International

GigE	Gigabit Ethernet
GMSK	Gaussian Minimum Shift Keying
GPL	General Public License
GRC	GNU Radio Companion
GUI	Graphic User Interface
HDTV	High Definition Television
IF	Intermediate Frequency
IP/UDP	Internet Protocol/User Datagram Protocol
ISM	Industrial Scientific and Medical Band
LR-WPAN	Low Rate Wireless Personal Area Network
LQI	Link Quality Indicator
MAC	Media Access Control Layer
MCU	Micro Controller Unit
MIMO	Multiple-Input/Multiple-Output
MPDU	MAC Protocol Data Unit
NIC	Network Interface Card
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open System Interconnection
PAN	Personal Area Network
PC	Personal Computer

PDA	Personal Digital Assistant
PER	Packet Error Rate
PHY	Physical Layer
POS	Personal Operating Space
PPDU	Physical Protocol Data Unit
PSK	Phase Shift Keying
PU	Physical Unit
QAM	Quadrature Amplitude Modulation
RF	Radio Frequency
SDR	Software Defined Radio
TPC	Transmitter Power Control
USA	United State of America
VHF/UHF	Very High Frequency/Ultra High Frequency
WLAN	Wireless Local Area Network
WRAN	Wireless Regional Area Network
USRP	Universal Software Radio Peripheral

LISTS OF PICTURES, GRAPHS AND TABLES

1 INDRODUCTION

1.1 Background Information

The conventional radio transceivers are hardwired with application specific signal processing blocks and the programmability of encoding and modulation is limited and inflexible due to the hardware constraints. On the other hand, Software Defined Radio (SDR) overcomes these constraints imposed by standard-specific hardware. The idea of SDR is to move the hardware-software boundary as close to the antenna as possible [1]. The ideal Software Defined Radio would have an antenna sampled by an ADC and the rest is done in software, turning radio hardware design problems into software problems. The fundamental characteristic of software radio is that software defines the transmitted waveforms, and software demodulates the received waveforms. Implementation of modulation and demodulation is done using software instead of dedicated circuits. Leveraging extra programmability at the physical layer, the system can handle different radio signals without changing hardware.

Wireless sensor networks consist of small, battery-powered, sensor-equipped embedded devices that communicate wirelessly using an on-board low power radio. Sensor networks enable numerous surveillance, monitoring, and other applications. Similarly, the Cognitive Radios enable to share the spectrum, usually underutilized, with the primary user. For this purpose the Cognitive Radios must be able to learn the environment and utilize the spectrum whenever primary is absent or the interference to the primary is kept tolerable. The current research community is very active in exploring the new ideas which can enhance the performance of the existing standards. The validation of these concepts and ideas require a flexible and programmable solutions and the SDR is seen as the most appropriate prototyping platform.

1.2 Working Environment of the Project

The project work was carried out at Department of Communications and Networking (ComNet) at Aalto University, School of Science and Technology.

ComNet department at Aalto University is actively participating in projects such as Cognitive Radio, Wireless Sensor Networks and 3G/4G wireless networks on inter-disciplinary and industrial levels. It gives an opportunity to the students to work very close to the current state-of-the-art research problems.

1.3 Purpose of the Project

The ComNet is exploring the feasibility of using Software Defined Radio for sensor and cognitive networks. The objective is to prepare a flexible platform for validating the customized MAC and PHY layer protocols and algorithms. For this purpose, the first step is the compatibility validation of the existing wireless sensor platform with the software defined radio

The existing sensor platform hardware is limited in its processing power, memory and hardware interfaces for external environmental sensors. Moreover, the conventional design of the (Sensinode) sensor nodes involves hardwiring of application-specific signal processing blocks and the programmability of the innovative modulation and coding techniques as well as the environment sensing and learning of the environment is quite limited. Therefore, the integration of the existing standard into the GNU Radio will not only validate its performance but also it will pave the way to future enhancements and large scale network establishment.

The programmable and flexible platform is replacement of the standard-specific hardware’s. This will give the opportunity to the researchers to quickly validate their innovative concepts and solutions.

1.4 Objectives of the Project

- learn the architecture of an open source Software Defined Radio and the signal processing blocks

- Integrate the IEEE 802.15.4 standard into the GNU Radio
- Perform the compatibility tests of GNU Radio with the existing sensor platform such as Sensinode Micro/Nano nodes

1.5 Outline

This thesis consists of five main chapters, and each chapter is subdivided into further sections. This thesis begins with an overview of the background information. Moreover purpose and objectives of this work have been presented and an outline briefly presenting the contents of the study. The second chapter which is project description is mostly covered the theoretical parts of the thesis. This chapter describes Cognitive Radio history, its types and also define Software Defined Radio, brief description of GNU Radio which is software component of SDR, brief description of the architecture of USRP/USPR2 which are the hardware components of SDR, description and application of Zigbee and IEEE 802.15.4 standard, introduction of Sensinode and there features as well as some detail of capturing IEEE 802.15.4 multiple-channels with SDR system. The third chapter describes how the project is physical implemented, starting form installation and configuration of software and hardware components of SDR into the system and ends on developing a communication between existing Sensinode platform and SDR platform. Comments of test results and final conclusions are finally inserted in chapters fourth and fifth respectively.

2 PROJECT DESCRIPTION

2.1 Cognitive Radio

2.1.1 History of Cognitive Radio

Joseph Mitola and Maguire officially introduced the Cognitive Radio paradigm [7], which has a built –in programmable and optimized new wireless system. In his dissertation, Mitola described this innovative approach in wireless system as follows:

“The term Cognitive Radio (CR) identifies the point at which wireless Personal Digital Assistants (PDA) and the related networks are sufficiently computationally intelligent about radio resources and related computer-to-computer communications to detect user communications needs as a function of use context, and to provide radio resources and wireless services most appropriate to hose needs.” [8] After Mitola officially work, different countries departments of regulatory found that most of the radio frequency spectrum was used ineffectively. In December 2003, Federal Communication Commission (FCC) advised regulation that would consider Cognitive Radio to apply opportunistic spectrum sharing [9]. After that some of the cognitive feature has been included in wireless standards. The First one is IEEE 802.11K standard, which is updated version of the IEEE 802.11 standard for radio resource management. IEEE 802.11K provides (measurement or sensing) information. This information include noise histogram report, channel load report and station statistic report and used as well to increase the progress of traffic distribution within a network. Devices like Wireless Local Area Network (WLAN) prefer to connect to the Access Point (AP) that provides strongest signals. This type of strategic arrangement some time causes one Access Point (AP) to be overloaded and others to be underutilized. In 802.11K standard, if the AP which has the strongest signal is overloading to its full capacity, then wireless device will connect to some other AP which is underutilized. Despite of that the wireless device receives the weaker signal the overall network is working more efficiently. The second example is Bluetooth. In Bluetooth standard Adaptive

Frequency Hopping (AFH) is added to avoid interference with other wireless technologies in the 2.4 Ghz unlicensed radio spectrum [10]. Thus IEEE 802.11 b/g devices, Bluetooth, Cordless telephones and Microwave ovens can use the same 2.4 GHz wireless frequencies. Because of the sensing algorithms used by AFH it determine if there are any other devices present in the Industrial Scientific and Medical (ISM) radio bands and decide whether to avoid them. AFH avoid taking frequencies which are already being used by WLAN. The third example is IEEE 802.22 standard which formed in 2004 and based on Cognitive Radio for Wireless Regional Area Networks (WRAN). This WRAN system operates on the VHF/UHF bands which are currently assigned for TV broadcasting services as well as wireless microphones. Practically WRAN system is able to sense the spectrum, identify unused TV channels and use these channels to supply broadband services for fixed wireless subscribers. [11-12].

2.1.2 Definition of Cognitive Radio

A radio type who has the ability to aware of its environment and decide intelligently about its operations based on that predefined information.

Cognitive Radio changes its behavior and operations to accomplish its objectives within the technologies like Software Defined Radio, adaptive radio [13].

Mitola Radio is a full Cognitive Radio (CR) which is fully reconfigurable device. It cognitively acquires itself to its local environment. Cognitive Radio (CR) has potential to supply greater benefits to telecommunication systems. Cognitive Radio discovers unused portion within the spectrum and bypass the interference with the PU, and communicate to a network in its preferred protocols. So it is intelligent enough to bypass over-crowding spectral. Cognitive Radio has the ability which can improve spectrum access and communication services. Cognitive Radio also has the ability to work as a bridge between two systems, it receive signal in one format and transmit it to another mode. CR amends spectrum management that is changing dynamically the operating frequency with the help of Dynamic Frequency Selection (DFS). One important feature of CR is Transmitter Power Control (TPC) it grand permission to transmission at the allowable limits when it is essential. It decreases the power in mobile device when it is in airport for safety reason. [14]

2.1.3 Types of Cognitive Radio

Cognitive Radio has two main types. The first type is (Mitola radio) where all the parameter are under consideration in a wireless network. [7]. the second type is (WRAN) where only radio frequency spectrum is considered [15]. Cognitive Radio also classified in terms of parts of the spectrum available.

- Licensed Band

These are the bands which are sold by license and used by Cognitive Radio. IEEE802.22 develops a standard for (WRAN) that will work in spectrum of unused television channels [16].

- Unlicensed Band

These are the bands which only used unlicensed parts of the radio frequency spectrum. IEEE 802.15 develops a standard for Bluetooth and WLAN [17]. The functional Cognitive Radio Architecture (CRA) is shown in Figure 1.

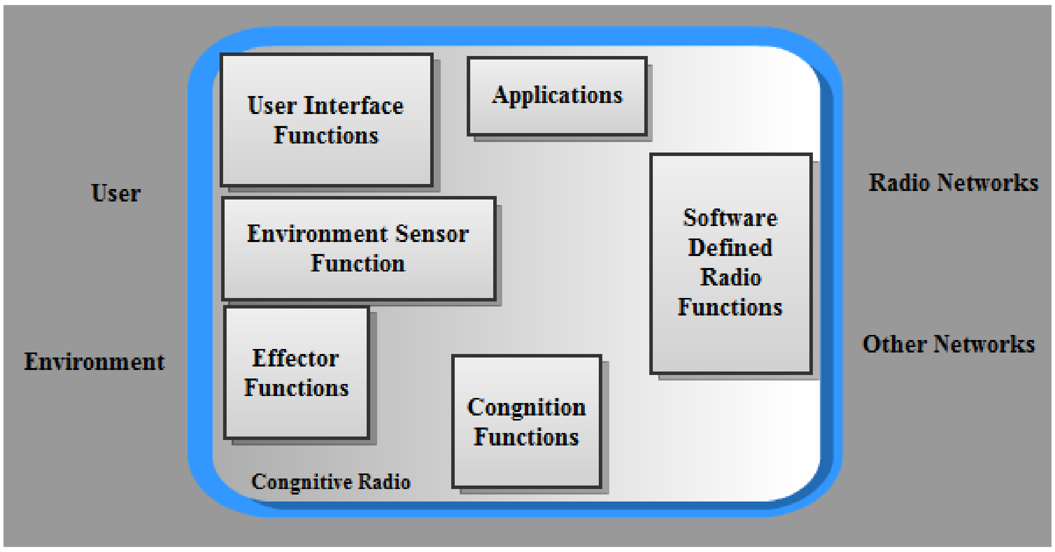


Figure 1: Functional Cognitive Radio Architecture (CRA) [14]

2.1.4 Software Defined Radio

The ideal SDR is where software pieces and not the hardware devices take care of the signal to extract information. Such a system is consisting of computer or some embedded device. In SDR system Analog-to-Digital and Digital-to-Analog Converter converts signals to and from Radio Frequency front-end and modulation, demodulation, mixing and filtering done by the software rather than electronic devices. SDR has a RF front-end which has the capacity to access more than 2.5 GHz for supporting different kind of communication services. In SDR instead of installing extra circuitry to handle different kind of radio signals, we just load appropriate software. It can be an AM radio at one instance and next instance it can be a wireless data receiver or may be a HDTV set. A Software Defined Radio could remove the drawbacks of current radios [18]. The basic architecture of SDR is shown in Figure 2.

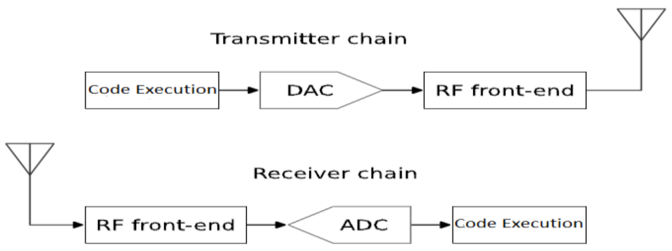


Figure 2: Basic architecture of SDR

A universal SDR structure with the specific software (GNU Radio) and hardware (USRP2) is shown in figure 3.The structure of Software Defined Radio has three parts. The left one build the RF frontend of the hardware assist as interface to the analog RF domain. In the second part, the intelligence of the hardware part is applied, forming the interface between the digital and making the interface between the digital and the analog world. In the third part, the whole signal processing is done-fully designed in software.

In the first part which is situated on the left side of the Figure 3. This analog RF signal can be received or transmitted over antennas. The upper path marks the receive path (Rx) and the lower path is marks the transmitter (Tx). Both parts can operate automatically. The operational frequency range is varying from DC to 5.9 GHz, depending on the available daughter-boards for USRP2. These daughter-boards which form the RF frontend of USRP2 are connected to USRP2 mother-board. On the motherboard of USRP2, an Analog to Digital Converter (ADC) samples the received signal and converts it to digital samples depending on the ADCs dynamic range of 14 bit (100MS/s) USRP2. These digital samples are then transferred to the FPGA and processed with Digital down Converters (DDC) to meet the exactly requested output frequency and sample rate.

The digital samples from ADC are mixed down to the desired IF by being multip-lied with sine respectively cosine function resulting in the, I and Q path. The fre-quency is generated with Numerically Controlled Oscillator (NCO) which synthe-sizes a discrete time, discrete amplitude waveform within the FPGA. Afterwards a decimation of the sampling rate is performed by an arbitrary decimation factor N. The sampling rate (fs) divided by N results in the output sample rate, send to host by Gigabit Ethernet. Using 4 byte complex samples (16-bit I and 16-bit Q) and respecting the Nyquist criterion leads to usable (complex) spectral bandwidth of 8 MHz for Gigabit Ethernet in USRP2. GNU Radio framework controls the further signal processing capabilities. GNU Radio is an open source framework, provid-ing various pre-assembled signal processing blocks for waveform creation and analysis in the software radio development.

In the transmit path, the same procedure is done vice versa using Digital up Con-verter (DUC) and Digital-to-Analog Converters [19].

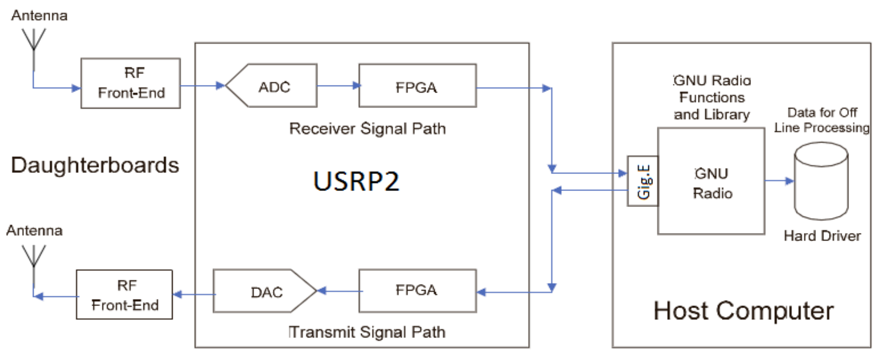


Figure 3: Block diagram SDR using GNU Radio, USRP2, and daughterboard.

2.2 GNU Radio

2.2.1 Background of GNU Radio

The GNU Radio project was started by Eric Blossom, an electric engineer in early 2000. Eric Blossom and his development colleague Matt Ettaus created an advanced software infrastructure to explore software radio. The motivation came for this software development that he wanted to design software based HDTV receiver. This software was developed as a reaction to the restriction on hardware receivers in the broadcast flag legislation that time within the USA. The main idea behind the GNU Radio development was to turn all the Hardware problems into software problems and take the software as close to the antenna as possible. The GNU Radio project founder, Richard Stallman liked Eric Blossom idea and they agreed to take the project under GNU aegis 34. Eric Blossom and his colleague developed a project and turned an ordinary computer into a quality radio receiver. They made it possible that with the combination of appropriate software modules they can develop a non-commercial radio receiver. GNU Radio developed only for radio amateurs but the interest of more and more researchers made GNU Radio more sophisticated. Now quite many developers affirm GNU Radio test-bed platform. GNU Radio support various Modulations (QAM, OFDM, GMSK, PSK), Signal processing technics (Filters, Equalizers, FFTs, Timing Recovery), and error corrections codes (Viterbi, Reed Solomon, Turbo codes) etc. GNU Radio runs under several operating systems like Linux, Debian, Fedora, Gentoo, Mandriva, SuSE, Ubuntu, Mac OS X, NetBSD, and Windows [20].

2.2.2 Introduction of GNU Radio

GNU Radio is an open source development toolkit which when combined with low cost RF hardware allows constructing radios, and it turns hardware problems into software problems. GNU Radio provides tons of libraries of signal processing blocks written in C/C++ programming language and the glue to tie it all together for building and deploying Software Defined Radio. These blocks are widely used in real time implementation of SDR and wireless communication research. The programmer builds a radio by creating a graph where vertices are signal processing blocks and edges defined the data flow between them. The signal processing blocks are implemented in C++ and the graphs are constructed and run in Python. Python supply a user friendly frontend environment to the developer to write routines in a rapid way. In GNU Radio some of the blocks have only input ports and some has only output posts. They work as data source and sink within the graph. The sources are used to read from a file or ADC and sink that write to a file, DAC or Graphical display. More than 100 blocks come with the GNU Radio and it is also possible to write new blocks [21]. The connections of GNU Radio component are shows in Figure 4.

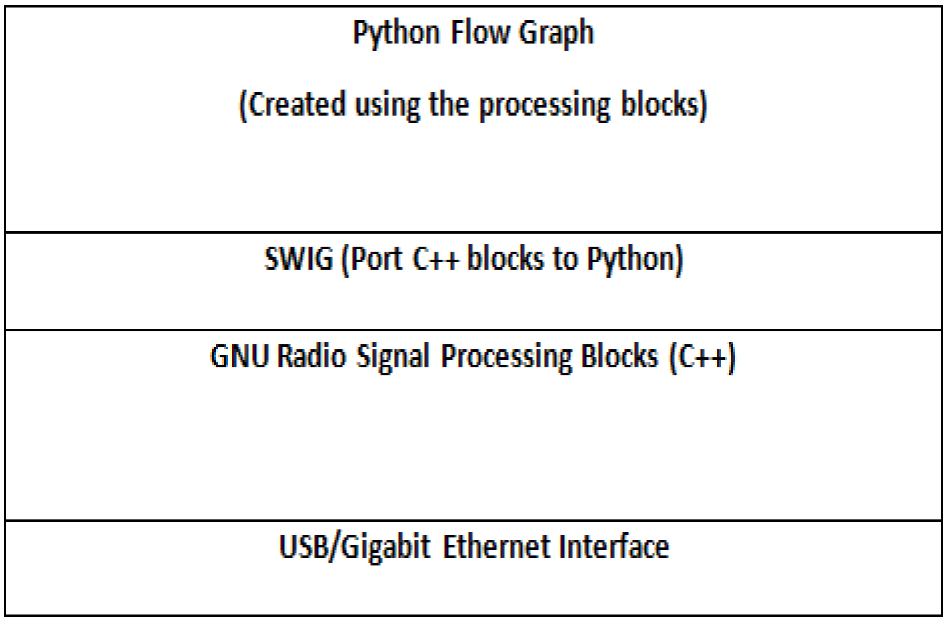

Python Flow Graph
(Created using the processing blocks)
SWIG (Port C++ blocks to Python)
GNU Radio Signal Processing Blocks (C++)
USB/Gigabit Ethernet Interface

Figure 4: Block Diagram of GNU Radio Components

The development of GNU Radio can be done using Object Oriented Approach and Procedural Approach Depending on the complexity of the problem. Here are some modules available in the current release of GNU Radio which shown in Figure 5 [22].

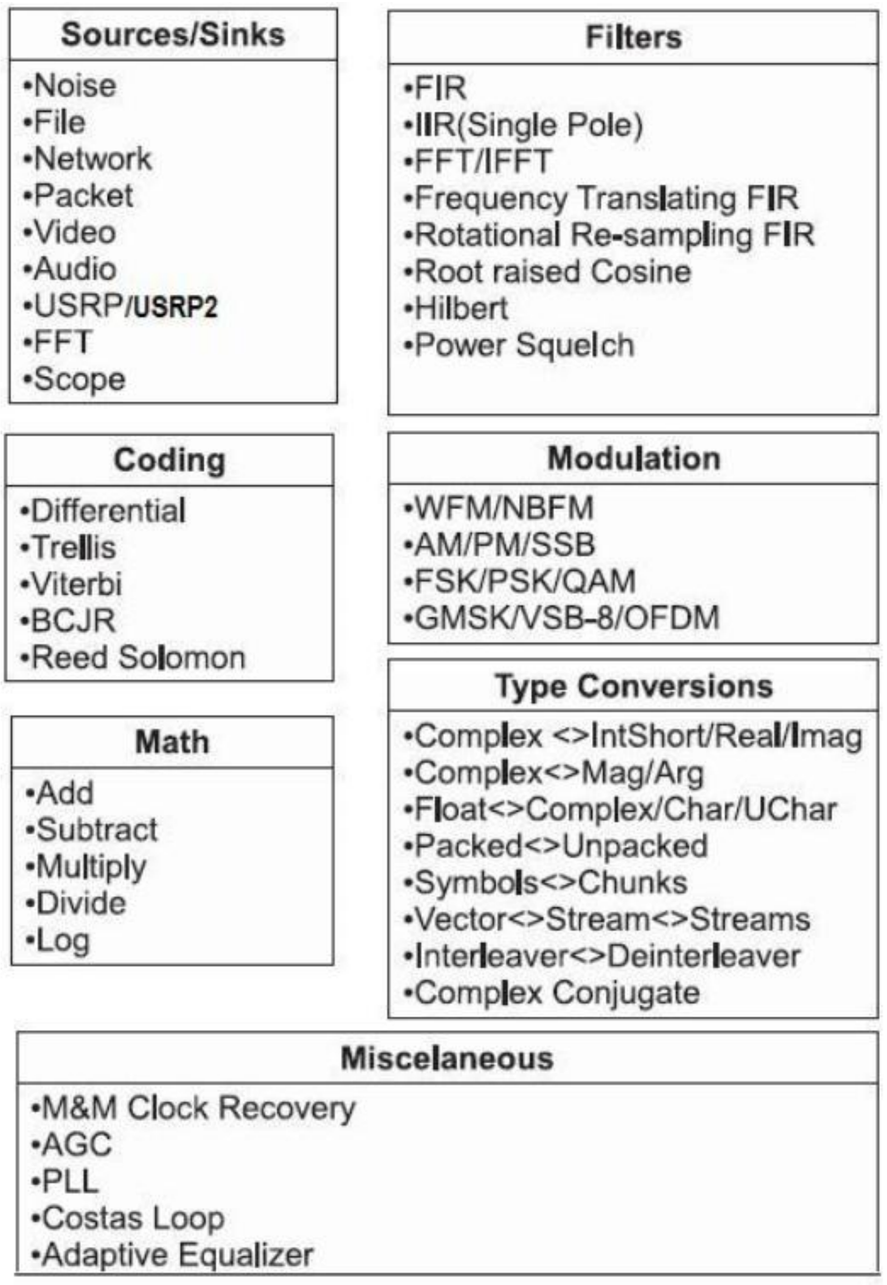


Figure 5: GNU Radio Modules [22]

GNU Radio is not very useful unless it has attached some hardware to interact with real world. It supports much different hardware, like sound card, and many different RF front-ends to received different bands of the telecommunication spectrum. The most commonly used one is Universal Software Radio Peripheral (USRP). The more detail about USRP will cover in Section 2.3.

2.2.3 GNU Radio Packages

The packages which required for compiling various parts of GNU Radio on Ubuntu are:

Development Tools (need for compilation)

- g++
- git
- make
- autoconf, automake, libtool
- sdcc (from “universe”; 2.4 or newer)
- guile (1.6 or newer)
- ccache (not required, but recommended if you compile frequently)
- Liberaries (need for runtime and for compilation)
- python-dev
- FFTW 3.X (ffw3, fftw3-dev)
- cppunit (libcppunit and libcppunit-dev)
- Boost 1.35 (or later)
- libusb and libusb-dev
- wxWidgets (wx-common) and wxPython (python-wxgtk2.8)
- python-numpy (via python-numpy-ext) (for SVN on or after 2007-May-28)
- ALSA (alsa-base, libasound2 and libasound2-dev)

- Qt (libqt3-mt-dev for versions earlier than 8.04; version 4 works for 8.04 and later)
- SDL (libsdl-dev)
- GSL GNU Scientific Library (libgsl0-dev >= 1.10 required for SVN trunk, not in binary repositories for 7.10 and earlier)
- Swing (1.3.31 or newer required)
- Edgy or previous: requires installation from source
- Feisty or newer: use the standard package install (swig)
- QWT and QWT PLot3d liberaries (optional for QT Gui)
- Polyphase Filter Bank examples
- For the examples in gnuradio-examples/python/pfb to work it is important to install python-scipy, python-matpoltlib, and python-tk

Other useful packages:

- doxygen (for creating documentation from source code)
- octave (from "universe") [22]

2.2.4 A Hello World Example Application

The “Hello World” program in GNU Radio of dial_tone.py is found in the directory <top gr dir>/gnuradio-example/python/audio/. In GNU Radio this application generate US dial tone and playing it using a computer sound card. The dial tone is generated by sine waves; one of them is used left channel and the other one used right channel of the sound card.

Hello World Example Python code:

```
#! /usr/bin/env python
```

```
from   gnuradio   import gr

from   gnuradio   import audio


def   build_graph ():

    sampling_freq = 48000

    ampl = 0.1

    fg = gr.flow_graph ()  # get empty flow graph


    # Instantiate source and sink blocks

    src0 = gr.sig_source_f (sampling_freq,

   gr.GR_SIN_WAVE, 350, ampl)

    src1 = gr.sig_source_f (sampling_freq,

   gr.GR_SIN_WAVE, 440, ampl)

    dst = audio.sink (sampling_freq)


    # connect the blocks

    fg.connect ((src0, 0), (dst, 0))

    fg.connect ((src1, 0), (dst, 1))


    return  fg


if __name__ == '__main__':

    fg = build_graph ()

    fg.start ()

    raw_input ('Press Enter to quit: ')
```

```
fg.stop ()
```

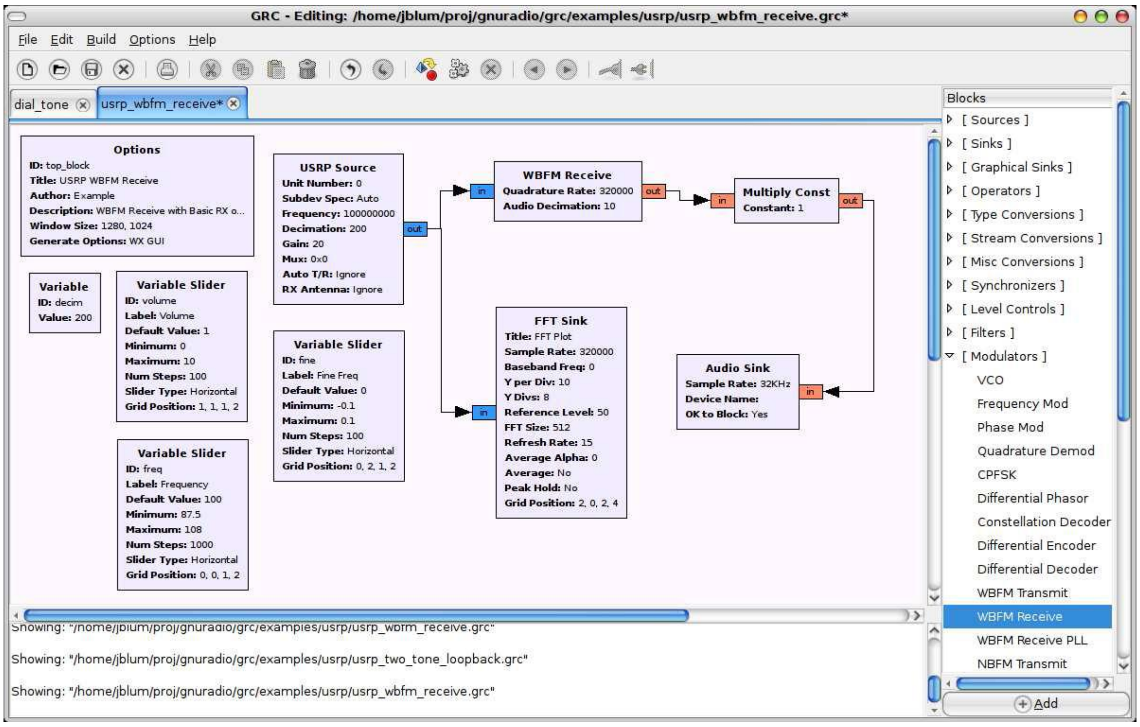
A flow graph is produced to hold the blocks and connection between them. The two sine waves are generated by the gr.sig_source_f calls. The f suffix shows the source produces floating numbers. One of the two sine waves is at 350 Hz, and the other one is at 440 Hz. Together, they sound like a US dial tone. The audio sink creates a sink and writes the input to the sound card. It carries one or more streams of floats in the range as its input. The three blocks are connected together using the connect() method of the flow graph. The connect() method carry two parameters, one of them is source endpoint and other one is destination endpoint and creates a connection from source to the destination. An endpoint has two components: a port number and signal processing block. The port number defines which input or output port of the specified block is to be connected. In the general form, an endpoint is represented as a python tuple like this: (block, port_number). When the port_number is zero, the block may be used alone. For example these following expressions are equivalent:

```
fg.connect ((src1, 0), (dst, 1))

fg.connect (src1, (dst, 1))
```

Once we get the built graph. We start it. The start() method forks one or more threads to run the computation described by the graph and returns control immediately to the caller. In this case, the software waits for any keystroke [18].

2.2.5 GNU Radio Companion

Since programming on GNU Radio is a hard process, due to its command line interface. Josh Blum from Johns Hopkins University developed a graphical interface called GRC. It permits users to interact with signal processing blocks of GNU Radio in a way similar to Simulink or Labview. In the process of completing its designed interface GNU Radio was taken in to mind and cover over 150 blocks from the GNU Radio project. Integration of GNU Radio blocks are done manually into GRC through Descriptive python definitions. The definitions are so flexible, and it permits multiple GNU Radio blocks to group into a single GRC super block. The GRC components are shown in Figure 6.



```
Options
ID: top_block
Title: USRP WBFM Receive
Author: Example
Description: WBFM Receive with Basic RX o...
Window Size: 1280, 1024
Generate Options: WX GUI

Variable
ID: decim
Value: 200

Variable Slider
ID: volume
Label: Volume
Default Value: 1
Minimum: 0
Maximum: 10
Num Steps: 100
Slider Type: Horizontal
Grid Position: 1, 1, 1, 2

Variable Slider
ID: freq
Label: Frequency
Default Value: 100
Minimum: 87.5
Maximum: 108
Num Steps: 1000
Slider Type: Horizontal
Grid Position: 0, 0, 1, 2

USRP Source
Unit Number: 0
Subdev Spec: Auto
Frequency: 100000000
Decimation: 200
Gain: 20
Mux: 0x0
Auto T/R: Ignore
RX Antenna: Ignore

Variable Slider
ID: fine
Label: Fine Freq
Default Value: 0
Minimum: -0.1
Maximum: 0.1
Num Steps: 100
Slider Type: Horizontal
Grid Position: 0, 2, 1, 2

WBFM Receive
Quadrature Rate: 320000
Audio Decimation: 10

Multiply Const
Constant: 1

FFT Sink
Title: FFT Plot
Sample Rate: 320000
Baseband Freq: 0
Y per Div: 10
Y Divs: 8
Reference Level: 50
FFT Size: 512
Refresh Rate: 15
Average Alpha: 0
Average: No
Peak Hold: No
Grid Position: 2, 0, 2, 4

Audio Sink
Sample Rate: 32KHz
Device Name:
OK to Block: Yes
```

```
Showing: "/home/jblum/proj/gnuradio/grc/examples/usrp/usrp_wbfm_receive.grc"
Showing: "/home/jblum/proj/gnuradio/grc/examples/usrp/usrp_two_tone_loopback.grc"
Showing: "/home/jblum/proj/gnuradio/grc/examples/usrp/usrp_wbfm_receive.grc"
```

Figure 6: GRC components

Flow Graph: It Interconnect the signal processing blocks, GRC has a scrollable window to place and connect different signal blocks.

Parameters: Parameters make influence to a function of a signal block. For Example, a parameter can be a gain, a flag, or a sampling rate. Parameters are displayed below its label for a signal block.

Connections: A connection links an input and output socket. The representation of connections is done by drawing a line between the two sockets. Connections must be between matching data types, including vectors.

Signal Blocks: All of the processing is done by signal block in a flow graph. For example, a signal block can be a source, an address, a filter or a sink. The representation of signal blocks in GRC is as a rectangular blocks. Each block has a label that points the list of parameters and name of the block.

Sockets: The inputs and outputs of the signal block are called sockets. Each signal block has a certain amount of sockets link up with it. For example, an adder has one output socket and two input sockets. A small rectangle attached to a signal block is representing a socket. The socket has a label indicating its function. Usually labels are named as “in” or “out”. Some other labels which indicate a vector type named as “vin” or “vout”. Sockets are also coloured to point their data type. For example, Red for float, Yellow for short, Blue for complex, Green for int, and Purple for byte.

Variables: They hold a number which is available to all components in the flow graph. They have two purposes. First one is, parameters can use a variable to share values. Second one is, Variable can also have a range (min and max) affiliated with them. While the flow graph is running, variable with ranges can be dynamically changed [23].

2.3 Universal Software Radio Peripheral

2.3.1 USRP

The Universal Software Radio Peripheral, (USRP) is an integrated circuit board that allows together with daughter boards, creation of a Software Defined Radio using any computer with an USB 2.0 port. Different types of plug-on daughter-boards permit the USRP to be used on different radio frequency bands. Now a days daughter-boards operating from DC (logic zero) to 5.9 GHz are available [21].

The USRP has been developed by Matt Ettus and his development team at Etters Research. There are quite many hardware available to interface GNU Radio altogether with electromagnetic spectrum but the USRP has become the standard one. The entire schematics design of the USRP is open source altogether with firmware, drivers and even FPGA and daughter-board designs. A USRP board setup consists of one mother board and up to four daughter-boards which is shown in Figure 7 [24].

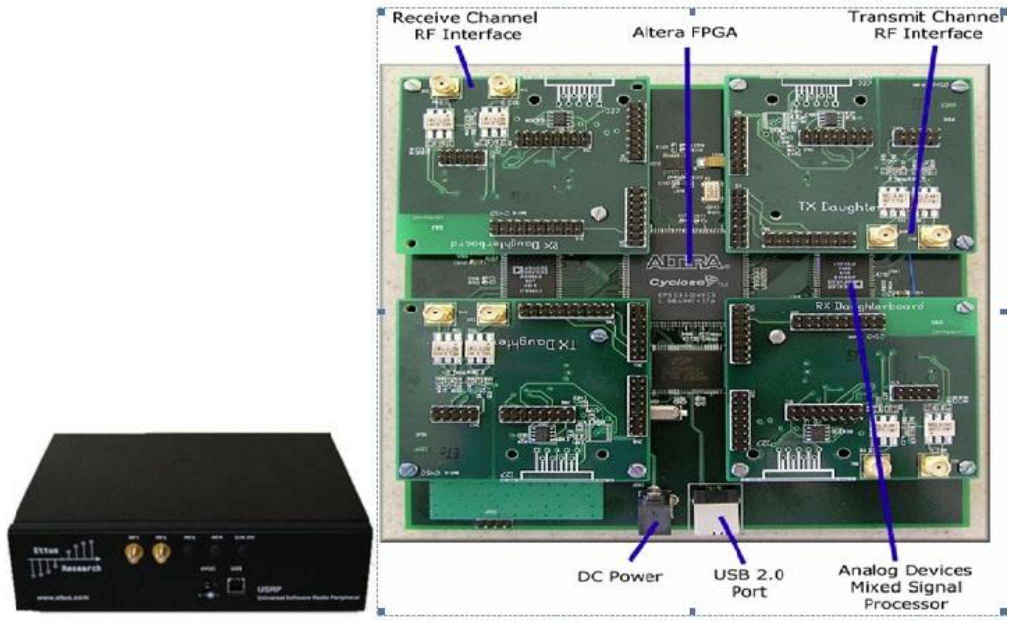
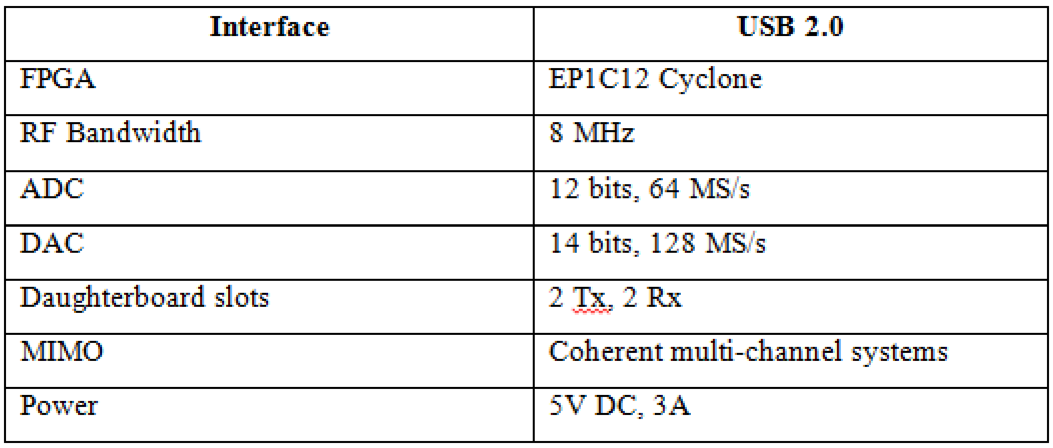


Figure 7: USRP together with Daughter-boards

The USRP motherboard has up to four 12-bit, 64M sample/sec ADCs, four 14-bit, 128M sample/sec DACs, and million gates, Filed programmable Gate Array (FPGA) and a programmable USB 2.0 controller. USRP motherboard supports four daughter-boards, two is used for receiving and two is used for transmitting. The daughter-boards containing the RF front ends. The Main features of USRP are shown in Figure 8 [25].



Interface	USB 2.0
FPGA	EP1C12 Cyclone
RF Bandwidth	8 MHz
ADC	12 bits, 64 MS/s
DAC	14 bits, 128 MS/s
Daughterboard slots	2 Tx, 2 Rx
MIMO	Coherent multi-channel systems
Power	5V DC, 3A

Figure 8: Features of USRP [25]

One USRP can received and transmit on two antennas in real time. The creation of MIMO (Multiple Inputs, Multiple Outputs) system is possible when all sampling clocks and local oscillators are fully coherent. FPGA provide a platform where high sampling rate processing is done and host computer provide a plate form for low rate sampling. The two Digital down Converters (DDCs) which are mounted onto the board mix, filter, and decimate (from 64M samples /s) incoming signals in the FPGA. The two Digital up Converters (DUCs) interpolate to the baseband signals to 128MS/s before translating them to the selected output frequency. USRP daughter-boards provide integrated RF front-ends platform. The USRP daughter-boards accommodate up to two transmit and two receive for RF I/O. The block diagram of USRP is shown in Figure 9 [21].

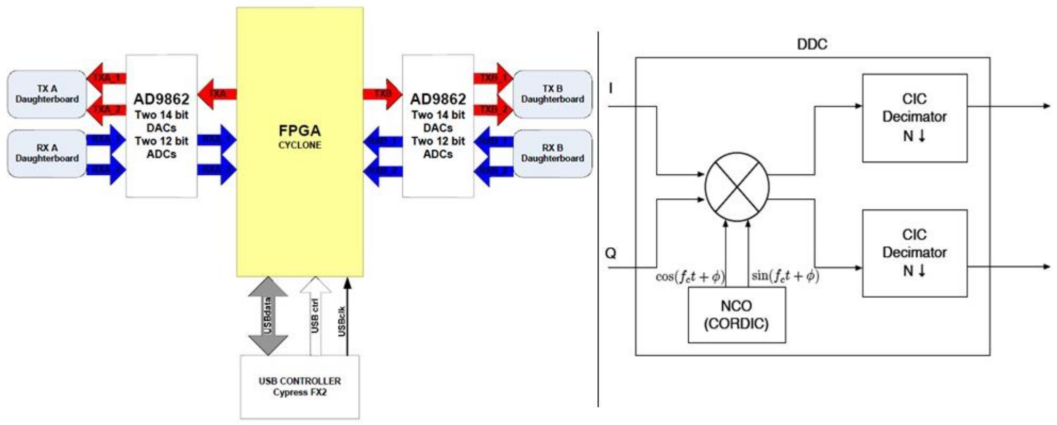


Figure 9: USRP Block diagram and USRP Digital down Convertor

The received analog radio signal enters the system through antenna which is connected to either A or B side of the daughter-board. On the daughter-board the signal is amplified and fed into a mixer which moves the desired frequency band down to Intermediate Frequency which is acceptable for ADC. This mixing process is known as down conversion which is shown in figure 9. The signal then enters the ADC interface module that converts the signal into digital samples. The ADCs sample at a rate of 64MS/s. The samples data width is 12bits. For each of the two daughter-boards there is one dual channel ADC is available. These can be used as two separate real channels but normally used in parallel for complex sampling. These digital samples then send to FPGA where further down conversion is take place and digital samples decimate to 32MS/s. And finally, the signal go through the RX buffer module where they get interleaved into a 16 bit value and send to the PC through USB bus.

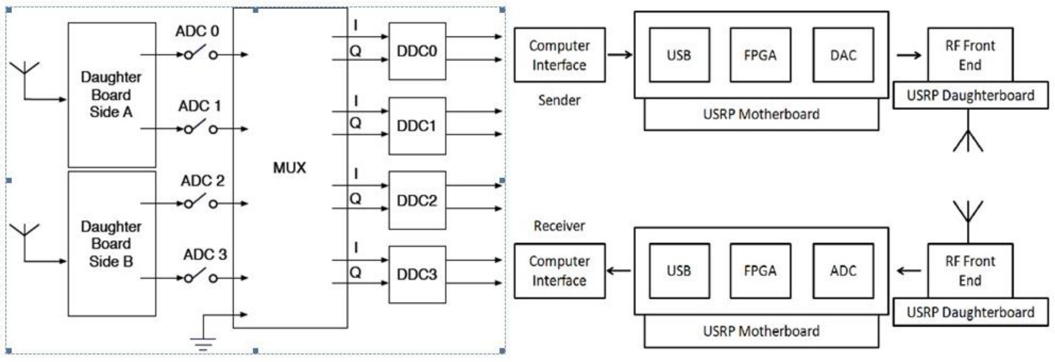


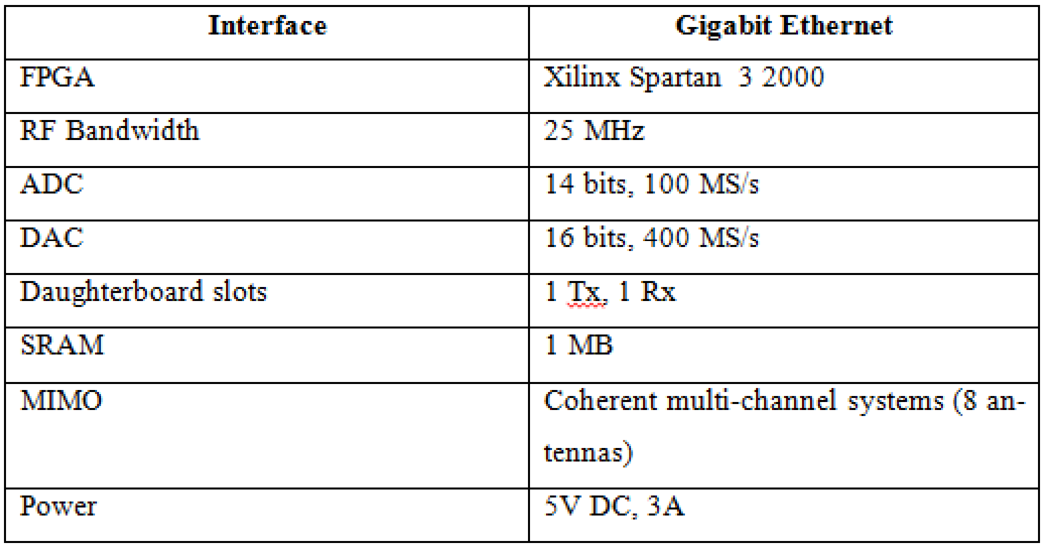
Figure 10: USRP received path block diagram and USRP sending and received application block diagram [21-27]

The transmit path is very similar to the receive path, the data comes as a 16 bit complex pair of signals I, Q from USB and fed into the FPGA through TX buffer module. In the FPGA the signal is raise to a rate of 32MS/s with the help of Cascaded Integrator Comb (CIC) unit, which is the necessary input rate of the DAC. The DAC is configured for complex sampling and can’t be used to transmit four real channels simultaneously but it can be possible to make changing into the configuration of GNU Radio and USRP. The bus runs at 64MHz between FPGA and two ADC/DAC codecs and on it two 32 MS/s streams of samples are multiplexed. The AD 9862 interpolated the signal by factor of 4, and then up converted to an Intermediate Frequency. For the transmit case no up conversion is done in FPGA. Instead it is done in the AD9862. This Intermediate Frequency finally converted to an analog signal. The block diagram of received and transmit path is shown in Figure 10.

The USRP is bounded to 32 MB/s when transferring data over USB2 link. This means in order to keep a constant steam of samples at least a factor of 8, decimation is required. This can be done by initializing parameters when interfacing with USRP using GNU Radio. A minimum decimation of 8 of 64 MS/s gives us 8 MS/s (complex samples), which is equal to sampling window of 8 MHz. This window gives us the chance to cover 2 (IEEE 802.15.4) channels at once in the 2.4 GHz band. Because of this limitation we choose USRP2 for multi-channel decoding [28].

2.3.2 USRP2

In May 2009, a new version of Universal Software Radio Peripheral (USRP) introduced and named USRP2. USRP2 is also open source hardware and all schematics and components information can be downloaded from the manufacturer website. USRP2 contains Field Programmable Gate Array (FPGA), Gigabit Ethernet controller, SD card slot, MIMO expansion slot with 8 LED indicators and daughter-boards which are connected over FPGA in the form of RF transceiver. SD card holds the driver for USRP2 motherboard and RF transceiver. 5V, 6A DC power is required to turn on the USRP2. The expansion MIMO port allows multiple USRP2 systems: it’s possible to connect together more USRP2 to form fully coherent multiple antenna systems for MIMO with as many as 8 antennas. The main features of USRP2 are shown in Figure 11 [25].



Interface	Gigabit Ethernet
FPGA	Xilinx Spartan 3 2000
RF Bandwidth	25 MHz
ADC	14 bits, 100 MS/s
DAC	16 bits, 400 MS/s
Daughterboard slots	1 Tx, 1 Rx
SRAM	1 MB
MIMO	Coherent multi-channel systems (8 antennas)
Power	5V DC, 3A

Figure 11: Features of USRP2 [25]

The received analog signal is enters into the system through antenna which is connected to the daughter-board. RF transceiver in the form of daughter-board fetches the signal and converts it to Intermediate Frequency (IF) around Direct

Current (DC) using Digital down Converter (DDC). This IF signal then fed to ADC. USRP2 holds 14-bit ADC converter supply 100MS/s sampling rate. These digital samples then send to FPGA. FPGA further down convert the remaining frequency and data rate conversion. After processing the signal, FPGA send the resulted signal to Gigabit Ethernet controller which passes it over to the computer where with the help of GNU Radio rest of the signal processing tasks are performed. The receiver and transmitter block diagram of USRP2 is shown in Figure 12.

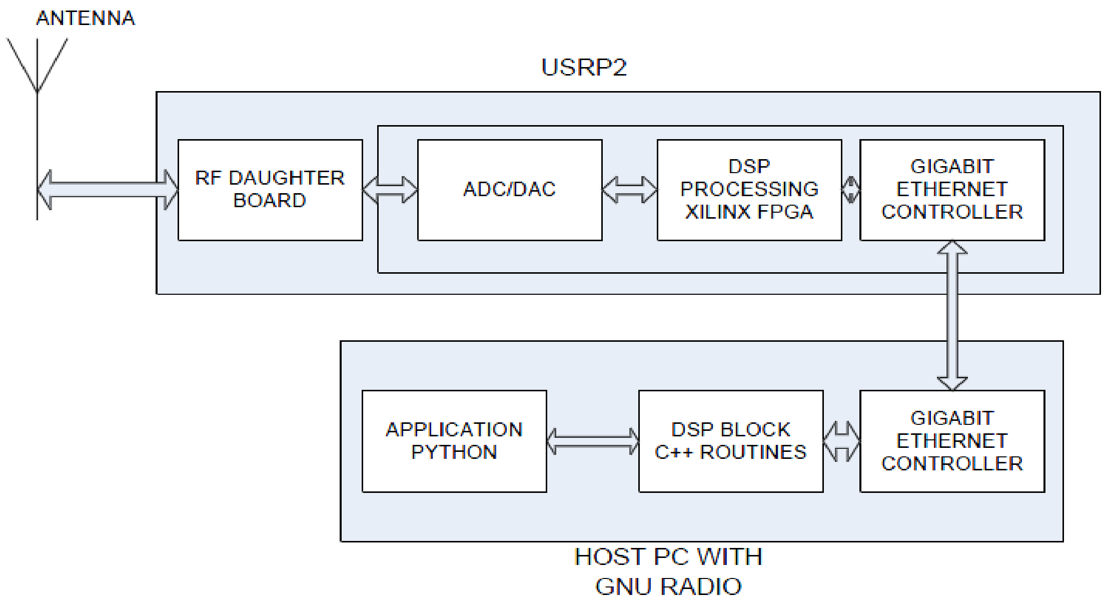


Figure 12: USRP2 operation with GNU Radio block diagram [29]

The transmission path is very similar to the receiver path, firstly Gigabit Ethernet controller of the host computer transfer the input parameters to USRP2. This received complex signal is Digital up Converted to IF signal by the (DUC) and transferred it to DAC. The DAC transfer the IF converted signal to the daughterboard which is RF transceiver where it is converted to RF signal and transmitted over the air. The FPGA and USRP2 are shown in Figure 13 [29].

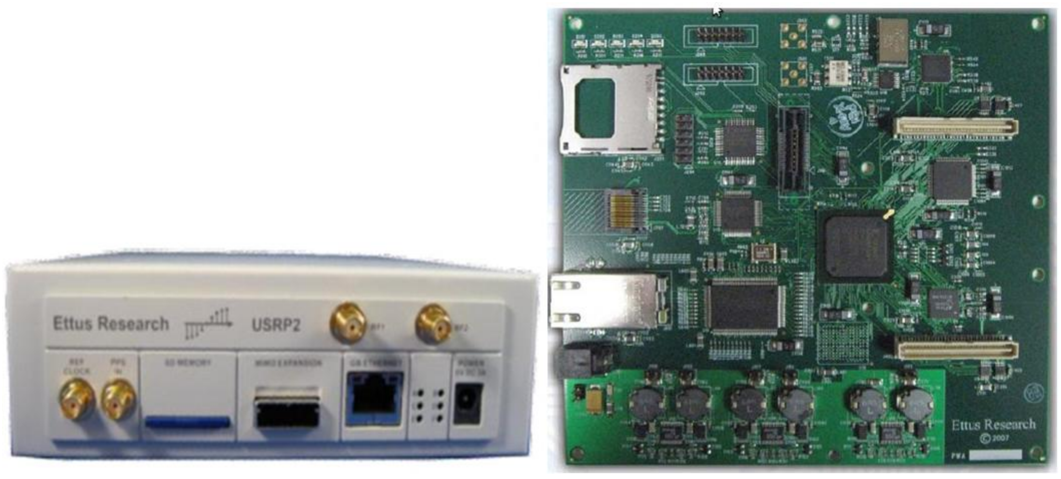
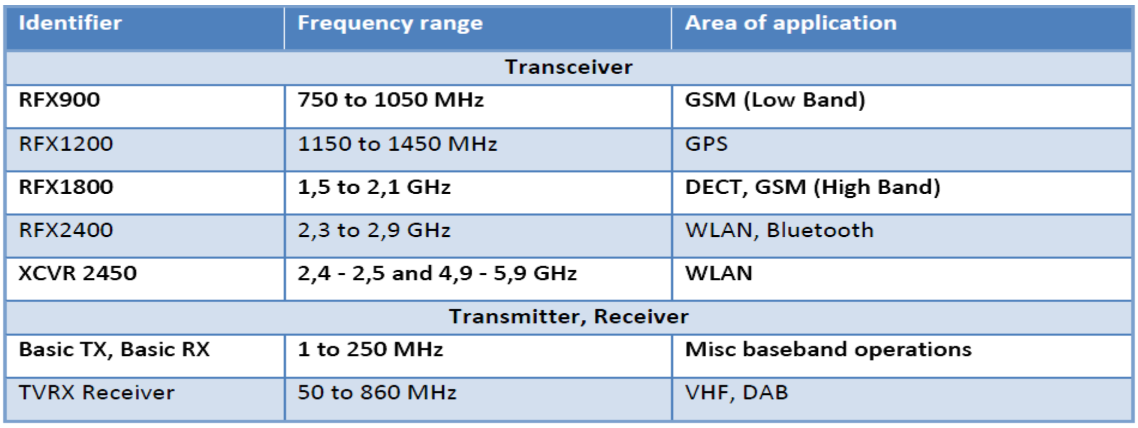


Figure 13: USRP2 and FPGA of USRP2 [25].

The USRP2 is used for multi-channel decoding. In USRP2 ADCs capacity is changed and now capable of 14-bit 100MS/s, and DACs is changed as well and now capable of 16-bit 400MS/s, and Gigabit Ethernet (GigE) is added instead of USB2. Gigabit Ethernet has the capacity to transfer data rate of 125MB/s which is equivalent to 30MS/s. When we decimation signal at rate of 4, USRP2 gives us sampling window of 25MHz. This window can cover 5 consecutive IEEE 802.15.4 channels in the 2.4 GHz band. In ISM band in total 16 channels available but with the help of USRP2 it is a good start to utilized 5 channels to seeing the power of SDR. AS the technology improves, SDR will capable to sample the entire 16 channels of ISM band. Daughter-boards available for USRP/USRP2 are shown in Figure 14 [25].

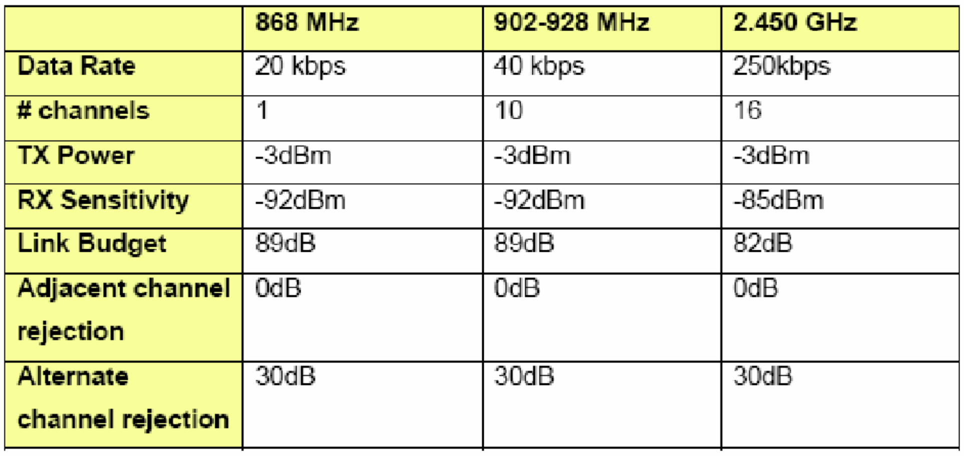


Identifier	Frequency range	Area of application
	Transceiver	
RFX900	750 to 1050 MHz	GSM (Low Band)
RFX1200	1150 to 1450 MHz	GPS
RFX1800	1,5 to 2,1 GHz	DECT, GSM (High Band)
RFX2400	2,3 to 2,9 GHz	WLAN, Bluetooth
XCVR 2450	2,4 - 2,5 and 4,9 - 5,9 GHz	WLAN
	Transmitter, Receiver	
Basic TX, Basic RX	1 to 250 MHz	Misc baseband operations
TVRX Receiver	50 to 860 MHz	VHF, DAB

Figure 14: Frequency range of several USRP/USRP2 daughter-boards [25]

2.4 ZIGBEE and IEEE 802.15.4

The IEEE 802.15.4 standard specifies the Physical Layer and Medium Access Control Layer of a Low Rate Wireless Personal Area Networks (LR-WPAN) for multiple Radio Frequency (RF) bands, including 868 MHz, 915MHz, and 2.4 GHz. The hardware for network communication is Physical Layer and Medium Access Control Layer is representing to the data link layer of the Open System Interconnection (OSI) reference model. Its main goal is to make a low data rate protocol for low power applications. Low power consumption can be achieved by allowing a device to sleep, only waking into active mode for brief periods. Enabling such low duty cycle operation is at the heart of the IEEE802.15.4 standard. The Zigbee Alliance constructed on top of the IEEE 802.15.4 protocols by further specifying the higher layers of the stack and releasing the Zigbee protocol. IEEE 802.15.4 parameters are shown in Figure 15 [31].



	868 MHz	902-928 MHz	2.450 GHz
Data Rate	20 kbps	40 kbps	250kbps
# channels	1	10	16
TX Power	-3dBm	-3dBm	-3dBm
RX Sensitivity	-92dBm	-92dBm	-85dBm
Link Budget	89dB	89dB	82dB
Adjacent channel rejection	0dB	0dB	0dB
Alternate channel rejection	30dB	30dB	30dB

Figure 15: IEEE 802.15.4 PHY parameters [30]

2.4.1 Zigbee Application

The Zigbee Alliance design and create products for home and industrial use. The Wireless sensor network is a key technology for several applications like home automation, building control, energy saving and automobile monitoring. These devices can transmit sensor data, sensor health, commands, and update wirelessly. Devices from different manufacturer can inter operate to each other by following

Zigbee standards. Zigbee technology enable two way communications, with the help of this technology consumer is not only able to monitor and keep track of domestic utilities usage but able to feed it to computer system for data analysis. The Zigbee applications are shown in Figure 16 [32].

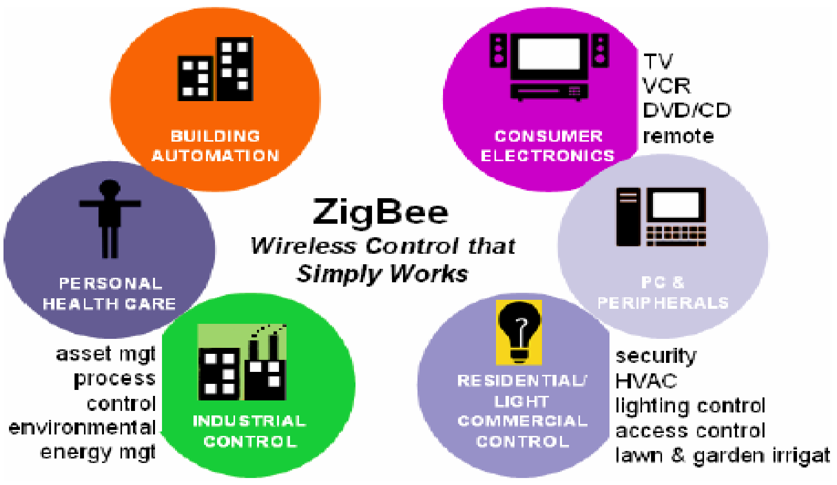


Figure 16: Zigbee Applications [30].

The West Technology Research solution issued a report that Zigbee chipsets into the home automation segment annul shipment is exceed 339 million units and showed in light switches, fire and smoke detectors, thermostats, appliances in the kitchen, video and audio remote controls, lands capping and security systems.

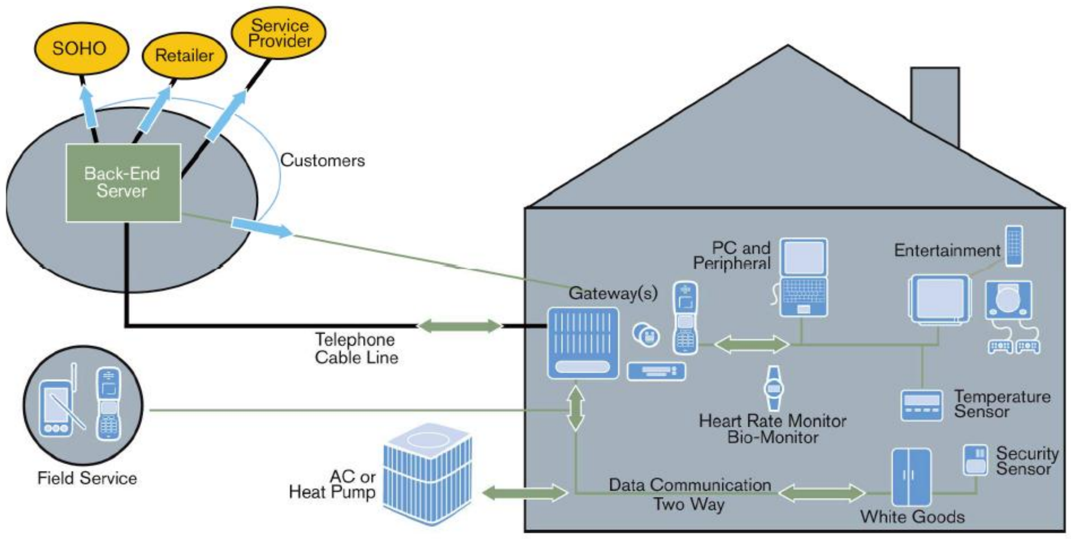


Figure 17: Zigbee Application in Home environment [30].

One of the Zigbee technology examples is home control centre which is shown in Figure 17. The central console combined data of room light, humidity, temperature, and air movement. The room control system can automatically control lights, blinds, and air conditioning of the home with the help of available data. In future more energy efficient homes and building can be built with the help of this technology where rooms need to kept optimal lighting and temperature with minimal amount of resources [32].

2.4.2 IEEE 802.15.4 Standard

The new IEEE 802.15.4 standard, defines the specification of Physical Layer (PHY) and Medium Access Control sublayer (MAC) for low data rate wireless connectivity within the devices which use minimal power and operate in the Personal Operating Space (POS) of 10 meters or less. A device use either 64 bit IEEE address or a 16 bit short address during the association procedure in an IEEE 802.15.4 network, and can accommodate up to 64k (2power 16) devices. There are two different network topologies (Star topology and peer to peer topology) that are useful for wireless network. In the star topology a single node is selected as a coordinator of Personal Area Network (PAN). All other nodes which are link to this network must communicate through the coordinator. It is possible that node as a coordinator of PAN has more computing resources and may be mains powered and other linked node probably battery powered. This kind of setup is used in home automation applications where there is a central control point. In peer to peer topology model, there is still a coordinator but in this topology every node can communicate to its neighbouring nodes within a reception range. This is a complicated mesh network topology and used in industrial production, inventory tracking [31].

The Wireless communication under IEEE 802.15.4 operates in three license free ISM frequency bands. In 2.4 GHz band 16 channels (start from channel number 11 and go to channels number 26) are allowed and it hold data rates of 250Kb/sec, in 915 MHz band 10 channels (start from channel number 0 and go to channel number 10) are allowed and it hold data rate of 40kb/sec, and in 868 MHz band only one channels is allowed and it hold data rate of 20kb/sec. So in total there are 27 channels are allowed in 802.15.4. These channels are spaced 5 MHz apart with a spectral widow of 2 MHz. The block diagram of 2.4 GHz of IEEE 802.15.4 corresponding bit/symbol/chip/ rate is shown in Figure 18 [21].

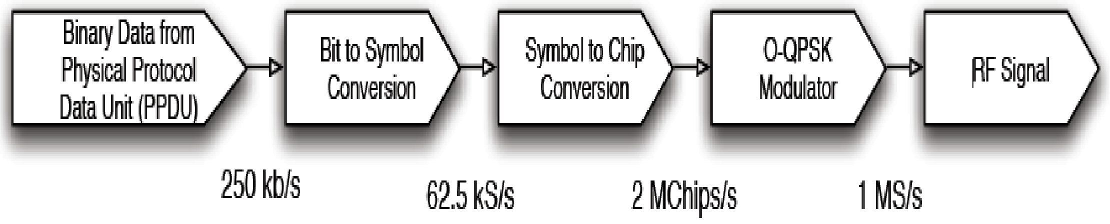


Figure 18: Block diagram of 2.4 GHz of IEEE 802.15.4 corresponding bit/symbol/chip rates the modules produce [21]

The 2.4 GHz IEEE 802.15.4 band has seen major developed because of its availability worldwide. After the general introduction about IEEE 802.15.4 now the discussion is more focus on 2.4 GHz band because this band is used in this project. This band has a transmission data rate 250kbit/s. First transmitted data is converted into 4 data symbols, than it is spread according to 32 bit sequence at a rate of 2 MChip/s. The Offset Quadrature Phase Shift Key than modulated these chipping sequence and the result of the signal is sent out centred at the channel frequency.

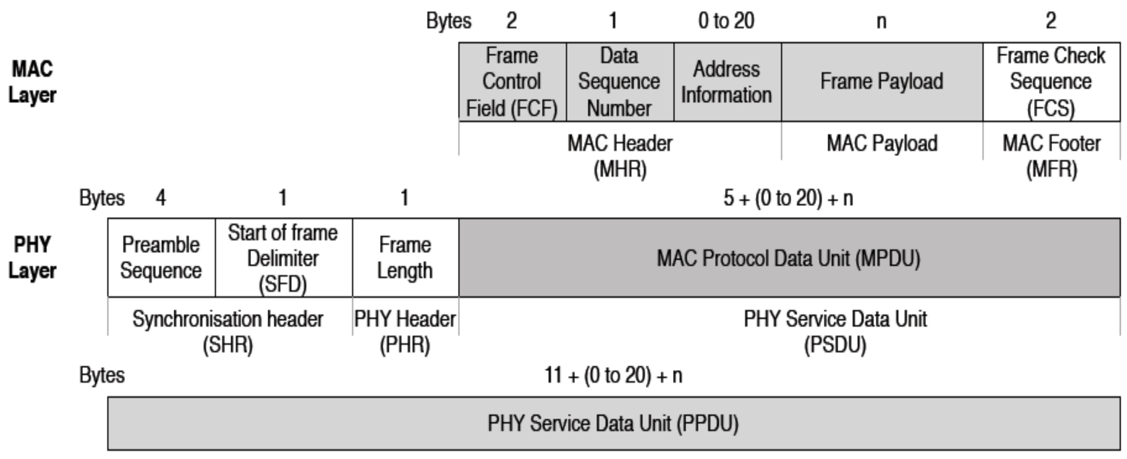

Bytes 2 1 0 to 20 n 2
MAC Layer
Frame Control Field (FCF)
Data Sequence Number
Address Information
Frame Payload
Frame Check Sequence (FCS)
MAC Header (MHR)
MAC Payload
MAC Footer (MFR)
Bytes 4 1 1 5 + (0 to 20) + n
PHY Layer
Preamble Sequence
Start of frame Delimiter (SFD)
Frame Length
MAC Protocol Data Unit (MPDU)
Synchronisation header (SHR)
PHY Header (PHR)
PHY Service Data Unit (PSDU)
Bytes 11 + (0 to 20) + n
PHY Service Data Unit (PPDU)

Figure 19: IEEE 802.15.4 (PHY & MAC Layer) Packet Frame Layout [21]

A Physical Protocol Data Unit (PPDU) contains synchronized header, a frame length field and the MAC Protocol Data Unit (MPDU). The IEEE 802.15.4 packet has a maximum MPDU size of 127 bytes. MPDU consist of Frame Control Field (FCF), sequence number, address field, frame payload, and finally the Frame Check Sequence (FCS). The field before payload holds metadata regarding the contents of payload. The FCS is ensuring the integrity of the data. The IEEE 802.15.4 (PHY & MAC Layer) packet frame layout is shown in Figure 19 [21].

2.4.3 802.15.4 ISM Band Co-Existence

The 2.4 GHz band which is part of Industrial Scientific and Medical (ISM) band. Many different communication standards use this unlicensed part of spectrum. Some of the common technologies which use this band are IEEE 802.15.1 (Bluetooth), IEEE 802.15.4 (Zigbee), and IEEE 802.11b/g used by Wireless Local Area Networks (WLAN) [32].

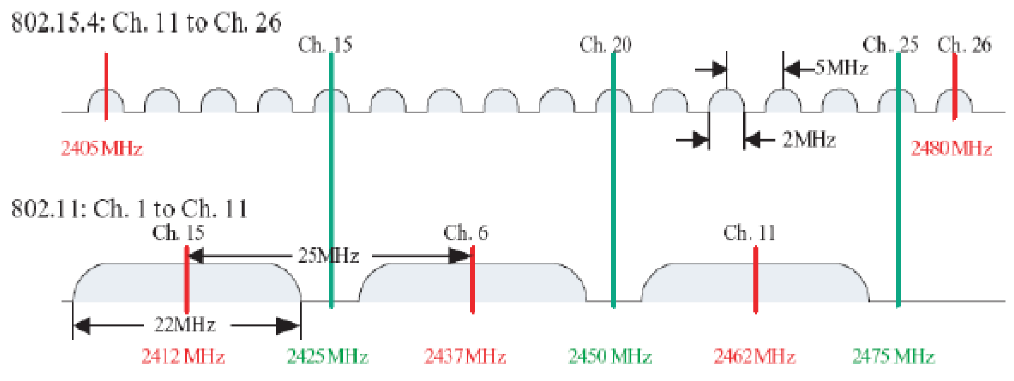


Figure 20: IEEE 802.15.4 and WLAN Spectrum Overlap [32]

These technologies work in the same communication spectrum and resulting overlapping bands. The number of different wireless technologies reasonably concerns about interference in the unlicensed 2.4 GHz ISM band. The Figure 20 shows the overlap between WLAN and Zigbee. The WLAN devices have much higher power output as compare to IEEE 802.15.4 nodes. The IEEE 802.15.4 channels experience significant frame loss that was centred within 7 MHz of the WLAN channel. The longer packet has a higher Packet Error Rate (PER) as they are longer in size and have more chance of corruption because of the interference. A number of solutions are advised for IEEE 802.15.4 to continue operating despite of possible communication spectral interference. All of these solutions require channel hopping which advice to move channel that the LR-WPAN is running on to a channel that has less interference [33].

A more adopted approach is Cognitive Radio approach. The PAN Coordinator will sense the current communication spectrum and then make the decisions to change channels based on the measurements. Maintaining the LR-WPAN while still hopping channels becomes the difficult part of applying these protocols [32].

2.5 Micro Series Sensinode

The cc2420 is a single-chip 2.4 GHz IEEE 802.15.4 RF transceiver designed for low power and low voltage wireless applications at the data rate of 250kbps.

2.5.1 U100 Micro.2420

Micro.2420 is the core of Micro Series node. It is a fully operational communication node which has accessible connectors for easy sensor and UI element operation. It runs on 2 NiMH batteries and also run on bus supplied 3.3v power. It integrates TI MSP430 microcontroller with a Chipcon 802.15.4 radio which has a capacity of 100 m transmit range. This radio offers ad hoc communication with a wide variety of topologies at 250kbps data rate.

The main functional blocks of Micro.2420 are the Microcontroller (MCU), programmable logic device (CPLD), radio transceiver and the power supply which is battery powered. The MCU has 10kB of ram and 256kB of FLASH memory. It can be programmed by using Micro series devel module D100 or the USB programmer A100. These programs can sport both JTAG and bootstrap programming modes. The hardware of Micro.2420 also supports MSP430 self-programming where MCU programs its own FLASH memory. An external 4Mbit serial FLASH memory is connected to MCU. It can be used to store measurement data. The purpose of CPLD is to connect the Micro.2420 system together as well as routes reset and clock signals and generates the select signals for the modules. The RF system consists of TI cc2420 RF transceiver, RF matching circuitry and antenna [34]. The Micro.2420 node is shown in Figure 21.

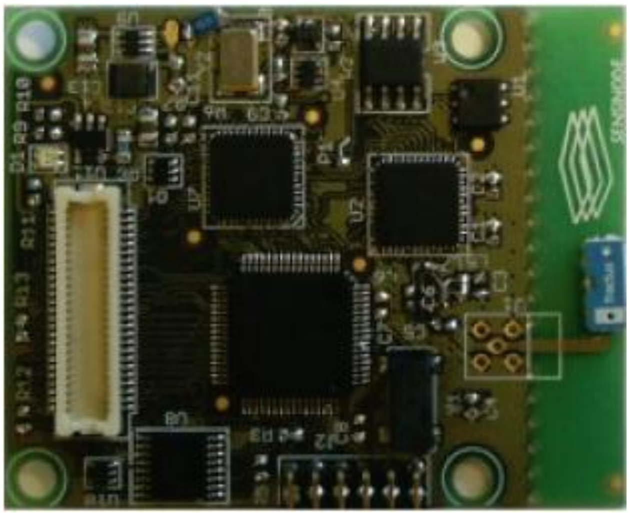


Figure 21: Micro.2420 node [35]

The features of Micro.2420 are given below [35]:

- Powerful MSP430 microcontroller including 10kB RAM and 256kB FLASH memory, running at 8Mhz.
- Multiple 12 bit ADC and two 12 bit DAC.
- Chipcon cc2420 802.15.4 RF transceiver with 250kbps data rate
- 4 Mbit serial data FLASH memory
- On-board antenna and optional connector for an external antenna.
- Expandable by adding other micro modules.
- Low power consumption, available modes are ultra-low power sleep and idle.
- Each module has a unique ID number

- Case connector with 8 digital/analogue IOs.
- RoHS compatible
- 3.3V operation, minimum battery voltage 1.5V
- Has support for running on 2 NiMH batteries
- Sleep mode current <50uA
- Operating current <25mA

2.5.2 U600 Micro.usb

The Micro.usb modules give the opportunity to Micro Sensinode that it can interface its node with for example a PC, provide a serial connection over USB for debugging, supplies power to the stack, and enables programming the microcontroller. The Micro.usb can be used with a rechargeable battery, which charged him when the node is powered by USB. The block diagram of Micro.usb is shown in Figure 22.

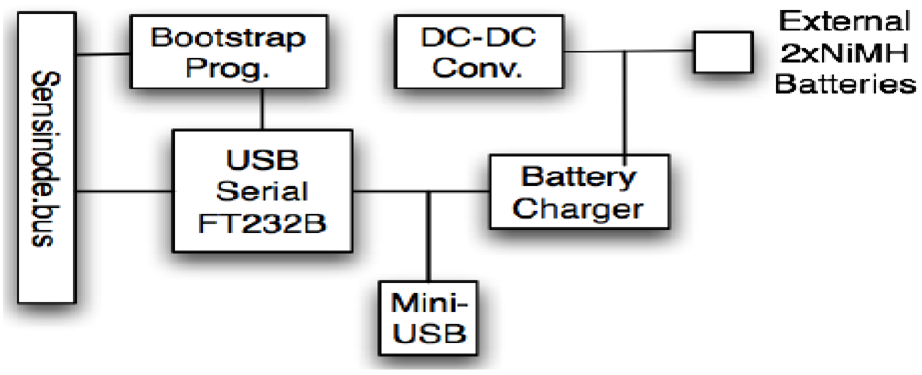


Figure 22: Block diagram of Micro.usb [36]

Micro.usb widely supported FTDI USB chip, and has drivers in all the major operating systems. When connected to PC, the FTDI driver provides a serial port to the Sensinode which can be used with a terminal for debugging or control purposes, or with the NanoStack PC tools [34]. The Micro.usb node is shown in Figure 23.

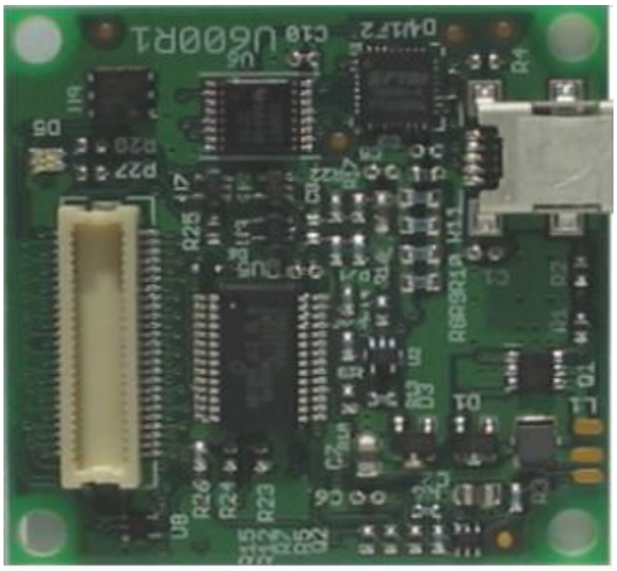


Figure 23: Micro.usb node [36]

The features of Micro.usb are given bellow [36]:

- USB serial adapter for the Sensinode Micro Series
- Used FTDI USB chip solution
- Drivers available for Linux, OS X, and Windows
- Full speed USB device data rate of 1MB/s
- The USB bus charges 2 NiMH batteries
- Enables bootstrap programming of the Micro.2420 over USB
- Each module has a unique ID number
- RoHS compatible.

2.5.3 NanoStack

NanoStack is built upon a stable, portable real-time operating system called FreeRTOS. NanoStack is a flexible 6LoWPAN protocol stack for wireless sensing and control using power devices with full IEEE 802.15.4 implementation. The architecture of NanoStack framework which is shown in Figure 24 is design to runs on the embedded wireless nodes, drivers and tools for accessing the wireless nodes from a PC.

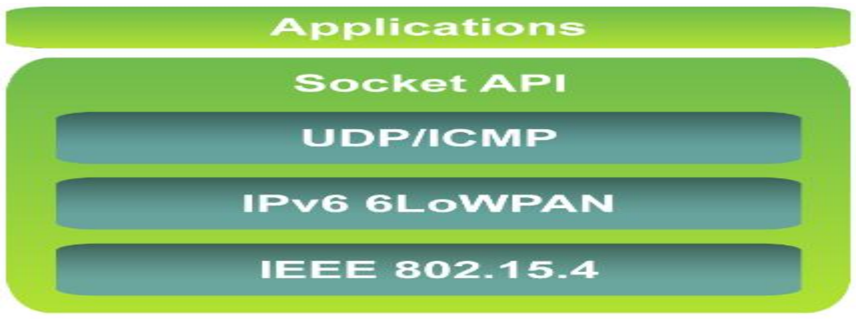


Figure 24: Architecture of NanoStck

NanoStack is built-in radio chip drivers for TI cc2420 and cc2430 radios. These radios apply part of the IEEE 802.15.4 standard in hardware and the rest of the IEEE 802.15.4 standard is applied inside NanoStack. The nRoute Protocol is used to communicate between a host PC and serial device which allow access to the local sensor network for network monitoring, system diagnosis and data collection [37].

2.6 GNU Radio for Multichannel Monitoring

There are quite many different packet analysing tools are available. There types vary in a way that how many number of channels they can capture and the types of visualization they can produce.

In the interest of exploring IEEE 802.15.4 demodulation and modulation with SDR, Thomas Schmid created a block for GNU Radio. This implementation for single channel worked and this block design can be adapted to multiple channels implementation.

After extending the Schimed work multichannel packet capture solution was implemented. The USRP2 is chosen over the USRP because it has the capability to sample a spectral window which contains 5 channels as compare to USRP which has capability of 2 channels. GNU Radio was used to demodulate the packet and separate the sampling window into the different channels. Decimating at a factor of 4, the USRP2 streams 25 MS/s over the GigE link. When putting the USRP2 in the middle of a centre channel which is shown in Figure 25, it is possible to capture 5 IEEE 802.15.4 channels. These samples enter the GNU Radio block which is shown in Figure 26. The incoming samples coming from USRP2 are transfer into a power squelch block. This squelch block limit the amount of processing which host computer has to perform by dropping all signals which do not pass threshold strength in dB. It is necessary to turn this squelch filter just above the noise level to make sure incoming packets are not dropped. These samples then forward parallel to 5 software based DDCs. The work of these DDCs to translate the signal by a frequency offset. After the translation, these blocks down-sample and low pass filter the signal to select a narrow band. These blocks translate the signal by different amount corresponding to the centre of the desired frequency. These samples are then decimated by factor of 5 and low pass filtered to a 2 MHz windows [32].

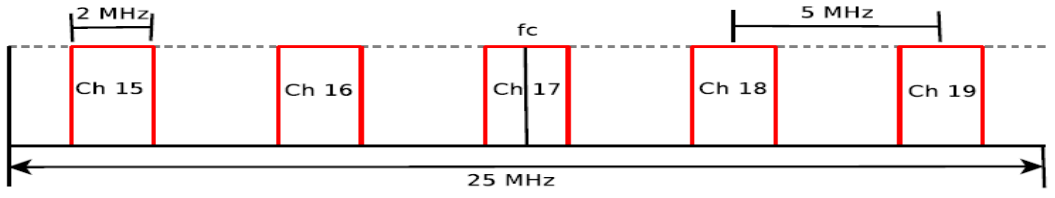


Figure 25: USRP2 capture 5 channels. Centre frequency is denoted by fc [32]

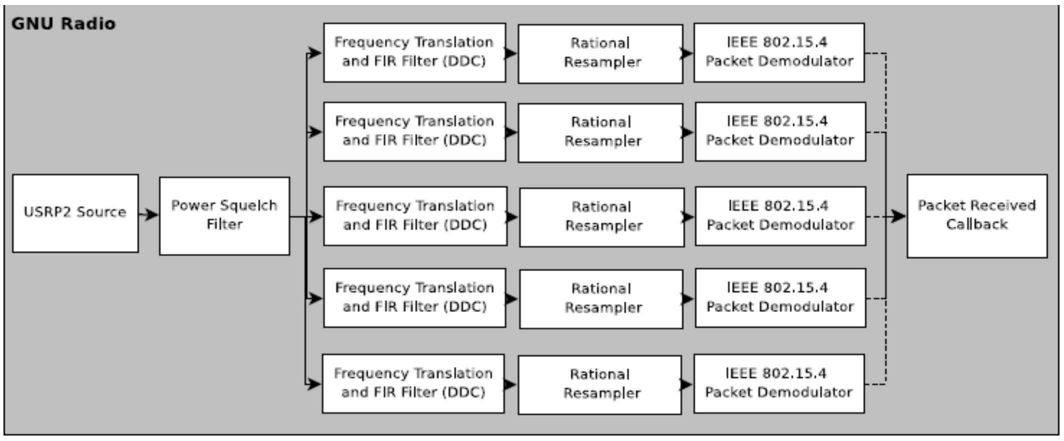


Figure 26: Architecture of GNU Radio IEEE802.15.4 Multichannel Demodulator [32]

At the output of DDCs, samples are produced a rate of 5MS/s. It is necessary to resample these samples at the rate of 4MS/s because IEEE 802.15.4 demodulation block requires 2 samples per chip and IEEE 802.15.4 standard has chipping rate of 2MChip/sec. This is done by rational resampler block and these samples at 4MS/s are sent to IEEE 802.15.4 demodulator. The IEEE 802.15.4 specification specifies that the Link Quality Indicator (LQI) must range from 0 to 255 and should relate to the relative signal quality of the channels. On TI cc2420 transceiver chips, LQI is calculated by averaging the chip correlation value of 8 consecutive symbols.

In GNU Radio LQI is implemented by summing the number of matched chip with in the first 8 symbols after the SFD is detected because each symbol is framed of 32 chips then adding 8 symbols the maximum sum of it is 256. The result then prepended to the demodulated packet. This demodulated packets then forwarded to a packet received callback exiting GNU Radio architecture. This callback then forwards the packet in a libcap format to a buffer. The libcap is a packet capture format that can be read by Wireshark and able to support multiple channels [32].

2.6.1 Wireshark

Wireshark is real-time network packet analyser software and is releases under the General Public License (GPL). It means Wireshark can be used on multiple computers without worrying about license fees. Because of GPL license, addition of new protocols to Wireshark is open and anyone can add new protocol and built into the source code. Wireshark provides packet dissection, packet capture, open / save captured data packets, import / export packet data, filter specific packet, support Linux and Windows, support GUI & CLI and various statistical reports.

The Figure 27 shows the overview of Wireshark architecture. GTK2 library is handling all the Graphical User Interface (GUI) such as input and output. The core block works as backbone; it attaches all the blocks together. Wiretap is dealing with captured files and many other formats. Capture block is used to capture data from network interface and it works with capture engine Dumpcap. Winpcap / Libpcap are an open source packet capture platform dependent libraries, which has the capture filter engine as well. These libraries are separate from the Wireshark package. Libpcap can capture packets through Network Interface Card (NIC). The Epan block of the figure contains the logic of packet dissection method. Protocol-Tree block keeps the protocol information of the captured data file. Dissector block opens the captured packets and handoff it to different dissectors to analyse the different parts of the data packet. For example, an incoming IEEE 802.15.4 packet can be dissected by physical layer dissector, then handed off to Media Access Control (MAC) layer dissector and then handed off to an IEEE 802.15.4 (Zigbee) protocol dissector.

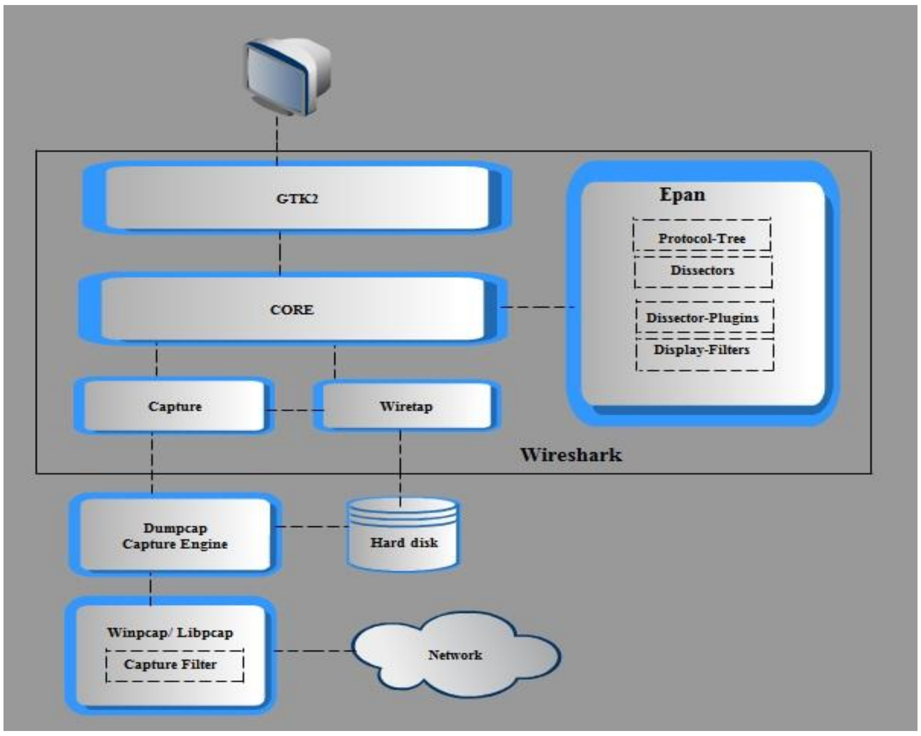


Figure 27: Wireshark internal structure [32]

Wireshark supports the dissection of MAC Protocol Data Unit (MPDU) of an IEEE 802.15.4 (Zigbee) packet. The support of IEEE 802.15.4 was added to Wireshark project in 2008 that enhanced Wireshark capabilities to dissect MPDU and recognize the fields inside the MPDU such as FCS, sequence number, addresses and payload data. Since Wireshark has a variety of built-in packet analysis features and provides support for IEEE 802.15.4 makes Wireshark best packet analyser software [32].

3 IMPLEMENTATION OF THE PROJECT

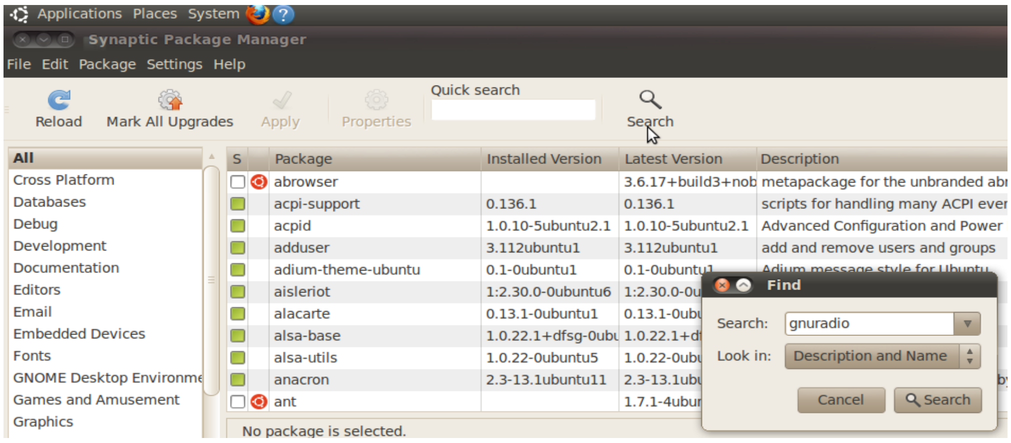
Once the theory about the SDR software architecture, USRP2 hardware architecture as well as IEEE 802.15.4 standard and Zigbee gone through, the project setup needs to be created. The implementation of project is subdivided into four parts:

3.1 Installation & Configuration

- Installation & Configuration of GNU Radio.
- Installation & Configuration of USRP2.
- Installation & Configuration of IEEE 802.15.4 Ucla_Zigbee_phy Code Examples.
- Installation of Python-Numeric Package.

3.1.1 Installation & Configuration of GNU RADIO

GNU Radio installation on any recent Ubuntu is easy and two different ways are given below:



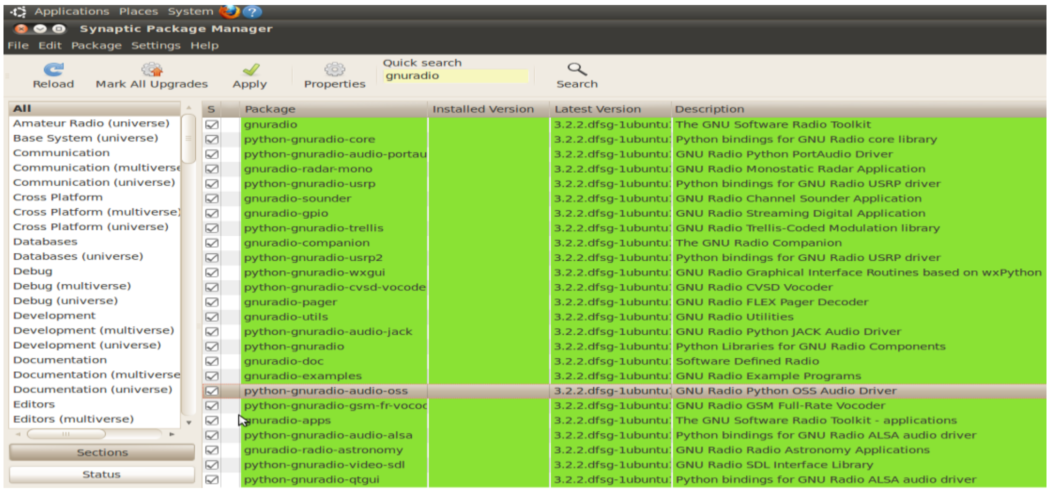
S	Package	Installed Version	Latest Version	Description
☐	abrowser		3.6.17+build3+nob	metapackage for the unbranded abr
🟩	acpi-support	0.136.1	0.136.1	scripts for handling many ACPI ever
🟩	acpid	1.0.10-5ubuntu2.1	1.0.10-5ubuntu2.1	Advanced Configuration and Power
🟩	adduser	3.112ubuntu1	3.112ubuntu1	add and remove users and groups
🟩	adium-theme-ubuntu	0.1-0ubuntu1	0.1-0ubuntu1	Adium message style for Ubuntu
🟩	aisleriot	1:2.30.0-0ubuntu6	1:2.30.0-0u	
🟩	alacarte	0.13.1-0ubuntu1	0.13.1-0ubu	
🟩	alsa-base	1.0.22.1+dfsg-0ubu	1.0.22.1+df	
🟩	alsa-utils	1.0.22-0ubuntu5	1.0.22-0ubu	
🟩	anacron	2.3-13.1ubuntu11	2.3-13.1ubu	
☐	ant		1.7.1-4ubur	

No package is selected.

Search: gnuradio
Look in: Description and Name

Figure 28: Installation of GNU Radio through Synaptic Package Manager

The first and easiest way to install GNU Radio is through “synaptic package manager” which is shown in Figure 28. Write gnuradio into synaptic package manager search and click onto search tab then it searched all the required gnuradio packages. After selecting and mark the gnuradio packages which are shown in Figure 29, click Apply tab and it downloaded and installed GNU Radio onto Ubuntu.



S	Package	Installed Version	Latest Version	Description
✓	gnuradio		3.2.2.dfsg-1ubuntu	The GNU Software Radio Toolkit
✓	python-gnuradio-core		3.2.2.dfsg-1ubuntu	Python bindings for GNU Radio core library
✓	python-gnuradio-audio-portau		3.2.2.dfsg-1ubuntu	GNU Radio Python PortAudio Driver
✓	gnuradio-radar-mono		3.2.2.dfsg-1ubuntu	GNU Radio Monostatic Radar Application
✓	python-gnuradio-usrp		3.2.2.dfsg-1ubuntu	Python bindings for GNU Radio USRP driver
✓	gnuradio-sounder		3.2.2.dfsg-1ubuntu	GNU Radio Channel Sounder Application
✓	gnuradio-gpio		3.2.2.dfsg-1ubuntu	GNU Radio Streaming Digital Application
✓	python-gnuradio-trellis		3.2.2.dfsg-1ubuntu	GNU Radio Trellis-Coded Modulation library
✓	gnuradio-companion		3.2.2.dfsg-1ubuntu	The GNU Radio Companion
✓	python-gnuradio-usrp2		3.2.2.dfsg-1ubuntu	Python bindings for GNU Radio USRP driver
✓	python-gnuradio-wxgui		3.2.2.dfsg-1ubuntu	GNU Radio Graphical Interface Routines based on wxPython
✓	python-gnuradio-cvsd-vocode		3.2.2.dfsg-1ubuntu	GNU Radio CVSD Vocoder
✓	gnuradio-pager		3.2.2.dfsg-1ubuntu	GNU Radio FLEX Pager Decoder
✓	gnuradio-utils		3.2.2.dfsg-1ubuntu	GNU Radio Utilities
✓	python-gnuradio-audio-jack		3.2.2.dfsg-1ubuntu	GNU Radio Python JACK Audio Driver
✓	python-gnuradio		3.2.2.dfsg-1ubuntu	Python Libraries for GNU Radio Components
✓	gnuradio-doc		3.2.2.dfsg-1ubuntu	Software Defined Radio
✓	gnuradio-examples		3.2.2.dfsg-1ubuntu	GNU Radio Example Programs
✓	python-gnuradio-audio-oss		3.2.2.dfsg-1ubuntu	GNU Radio Python OSS Audio Driver
✓	python-gnuradio-gsm-fr-vococ		3.2.2.dfsg-1ubuntu	GNU Radio GSM Full-Rate Vocoder
✓	gnuradio-apps		3.2.2.dfsg-1ubuntu	The GNU Software Radio Toolkit - applications
✓	python-gnuradio-audio-alsa		3.2.2.dfsg-1ubuntu	Python bindings for GNU Radio ALSA audio driver
✓	gnuradio-radio-astronomy		3.2.2.dfsg-1ubuntu	GNU Radio Radio Astronomy Applications
✓	python-gnuradio-video-sdl		3.2.2.dfsg-1ubuntu	GNU Radio SDL Interface Library
✓	python-gnuradio-qtgui		3.2.2.dfsg-1ubuntu	Python bindings for GNU Radio ALSA audio driver

Figure 29: Installation with Synaptic Package Manager Example

For Ubuntu 10.04 the following command line scripts which is shown in Figure 30, installed all the required dependencies but before running make ensure that the optional repositories are enabled in “Software Sources”.

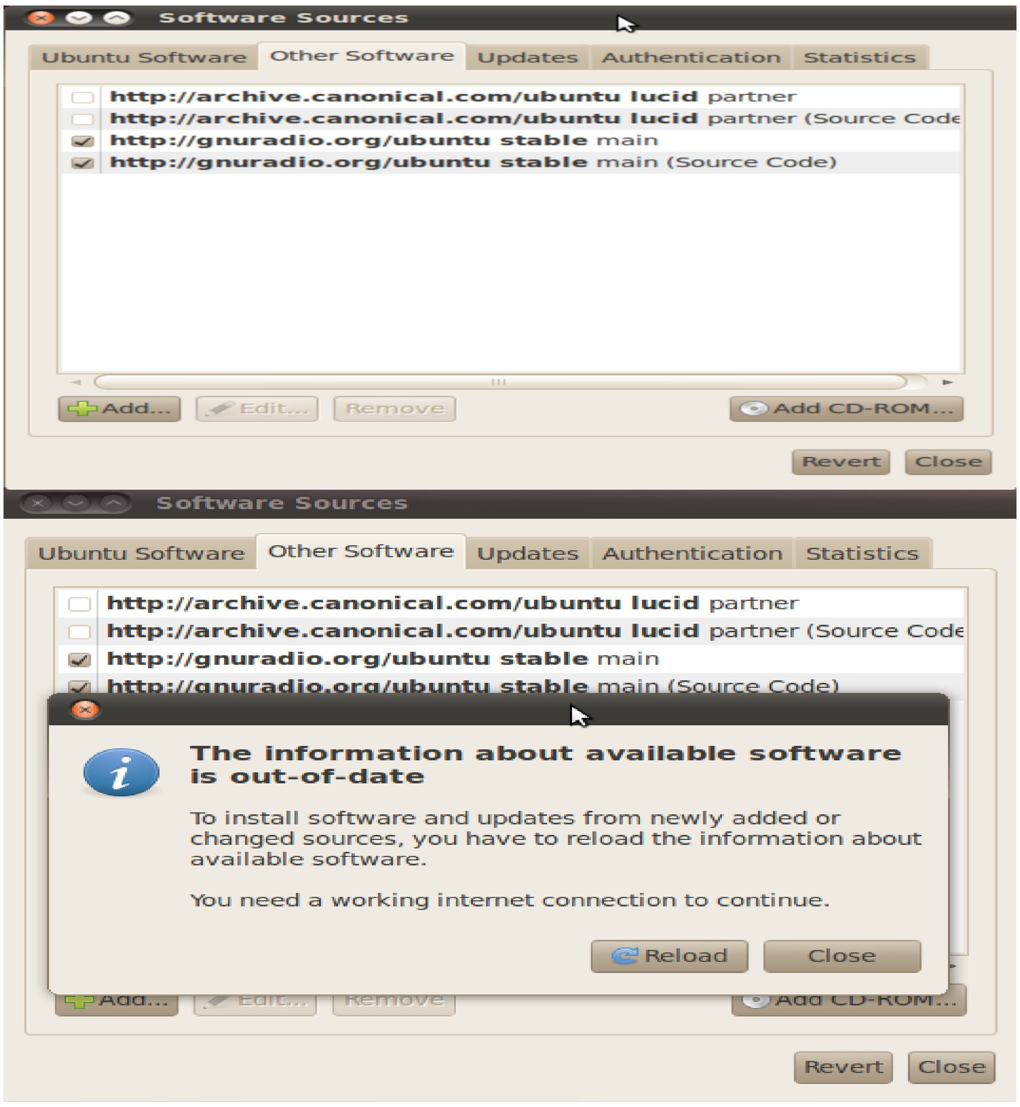
```
sudo apt-get -y install libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake auto-
conf libtool python-dev libfftw3-dev \

libcppunit-dev libboost-all-dev libusb-dev fort77 sdcc sdcc-libraries \

libsdl1.2-dev python-wxgtk2.8 git-core guile-1.8-dev \

libqt4-dev python-numpy ccache python-opengl libgsl0-dev \

python-cheetah python-lxml doxygen qt4-dev-tools \

libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4
```

Figure 30: Commands for updating installed GNU Radio [22]

The second and prefers way to install GNU Radio stable release is go to www.gnuradio.org copy and add these two links (shown in Figure 31) into System > Administration > Software Sources > Other software tab.

```
deb http://gnuradio.org/ubuntu unstable main

deb-src http://gnuradio.org/ubuntu unstable main
```

Figure 31: GNU Radio download link [22].



Software Sources

Ubuntu Software | Other Software | Updates | Authentication | Statistics

- [ ] http://archive.canonical.com/ubuntu lucid partner
- [ ] http://archive.canonical.com/ubuntu lucid partner (Source Code
- [✓] http://gnuradio.org/ubuntu stable main
- [✓] http://gnuradio.org/ubuntu stable main (Source Code)

Add... Edit... Remove Add CD-ROM...

Revert Close

Software Sources

Ubuntu Software | Other Software | Updates | Authentication | Statistics

- [ ] http://archive.canonical.com/ubuntu lucid partner
- [ ] http://archive.canonical.com/ubuntu lucid partner (Source Code
- [✓] http://gnuradio.org/ubuntu stable main
- [✓] http://gnuradio.org/ubuntu stable main (Source Code)

The information about available software is out-of-date

To install software and updates from newly added or changed sources, you have to reload the information about available software.

You need a working internet connection to continue.

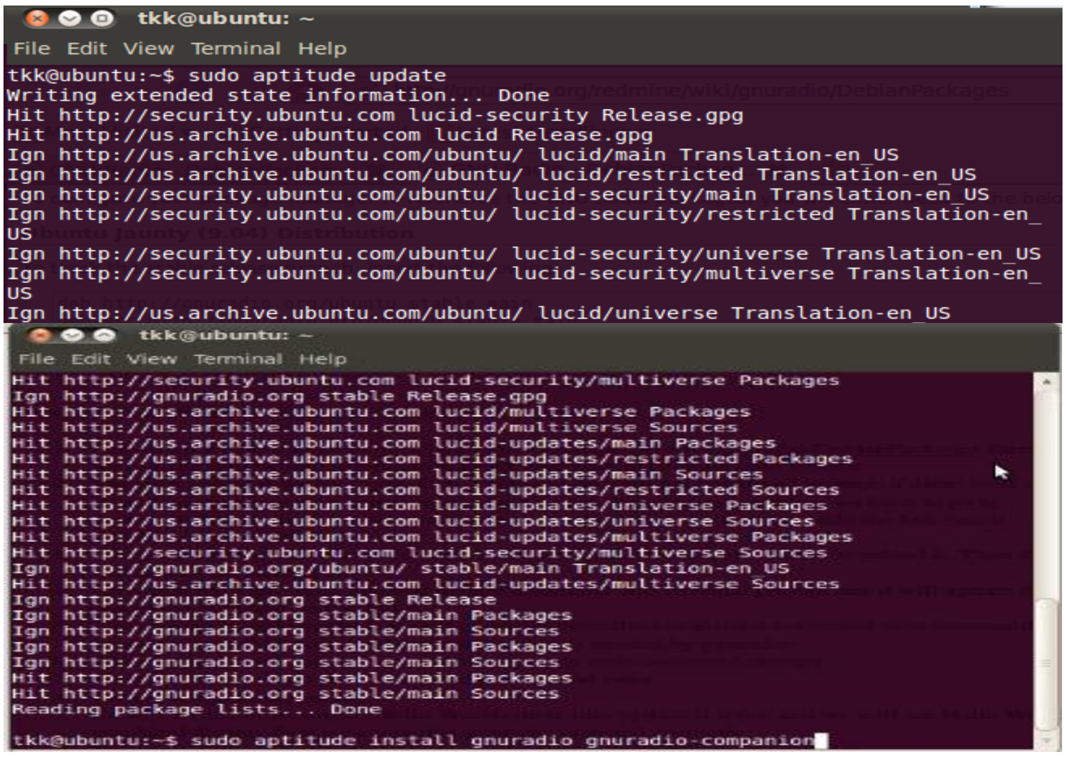
Reload Close

Revert Close

Figure 32: Downloading GNU Radio through Other software

After adding these links into other software tab which is shown in Figure 32, it asked reload and when reload applied it downloaded GNU Radio stable release.

When Applied “sudo aptitude update” command into terminal prompt which is shown in Figure 33, it started update the downloaded GNU Radio and installed it and applying “sudo aptitude install gnuradio gnuradio-companion” onto command prompt installed libraries and examples which were needed by GNU Radio.

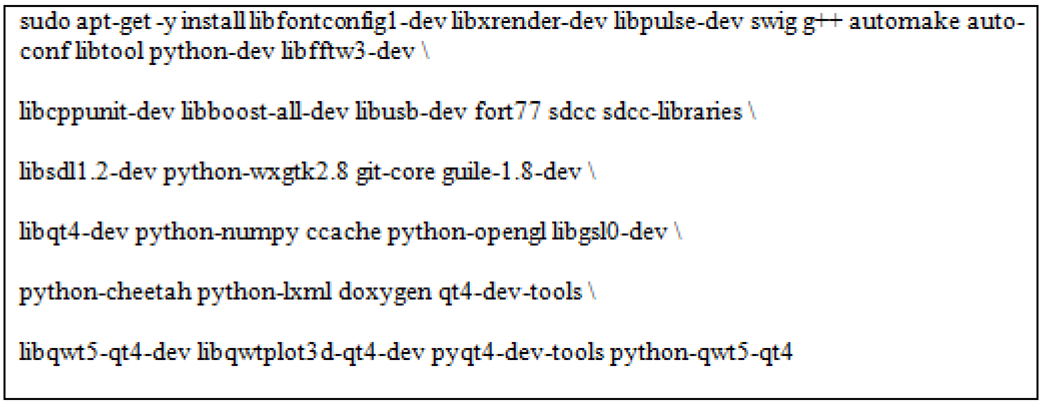


```
tkk@ubuntu:~$ sudo aptitude update
Writing extended state information... Done
Hit http://security.ubuntu.com lucid-security Release.gpg
Hit http://us.archive.ubuntu.com lucid Release.gpg
Ign http://us.archive.ubuntu.com/ubuntu/ lucid/main Translation-en_US
Ign http://us.archive.ubuntu.com/ubuntu/ lucid/restricted Translation-en_US
Ign http://security.ubuntu.com/ubuntu/ lucid-security/main Translation-en_US
Ign http://security.ubuntu.com/ubuntu/ lucid-security/restricted Translation-en_
US
Ign http://security.ubuntu.com/ubuntu/ lucid-security/universe Translation-en_US
Ign http://security.ubuntu.com/ubuntu/ lucid-security/multiverse Translation-en_
US
Ign http://us.archive.ubuntu.com/ubuntu/ lucid/universe Translation-en_US
```

```
Hit http://security.ubuntu.com lucid-security/multiverse Packages
Ign http://gnuradio.org stable Release.gpg
Hit http://us.archive.ubuntu.com lucid/multiverse Packages
Hit http://us.archive.ubuntu.com lucid/multiverse Sources
Hit http://us.archive.ubuntu.com lucid-updates/main Packages
Hit http://us.archive.ubuntu.com lucid-updates/restricted Packages
Hit http://us.archive.ubuntu.com lucid-updates/main Sources
Hit http://us.archive.ubuntu.com lucid-updates/restricted Sources
Hit http://us.archive.ubuntu.com lucid-updates/universe Packages
Hit http://us.archive.ubuntu.com lucid-updates/universe Sources
Hit http://us.archive.ubuntu.com lucid-updates/multiverse Packages
Hit http://security.ubuntu.com lucid-security/multiverse Sources
Ign http://gnuradio.org/ubuntu/ stable/main Translation-en_US
Hit http://us.archive.ubuntu.com lucid-updates/multiverse Sources
Ign http://gnuradio.org stable Release
Ign http://gnuradio.org stable/main Packages
Ign http://gnuradio.org stable/main Sources
Ign http://gnuradio.org stable/main Packages
Ign http://gnuradio.org stable/main Sources
Hit http://gnuradio.org stable/main Packages
Hit http://gnuradio.org stable/main Sources
Reading package lists... Done

tkk@ubuntu:~$ sudo aptitude install gnuradio gnuradio-companion
```

Figure 33: Installation of GNU Radio and its companion

For Ubuntu 10.04 the following command line scripts which is shown in Figure 34, installed all the required dependencies but before running make ensure that the optional repositories are enabled in “Software Sources”.



```
sudo apt-get -y install libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake auto-
conf libtool python-dev libfftw3-dev \

libcppunit-dev libboost-all-dev libusb-dev fort77 sdcc sdcc-libraries \

libsdl1.2-dev python-wxgtk2.8 git-core guile-1.8-dev \

libqt4-dev python-numpy ccache python-opengl libgsl0-dev \

python-cheetah python-lxml doxygen qt4-dev-tools \

libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4
```

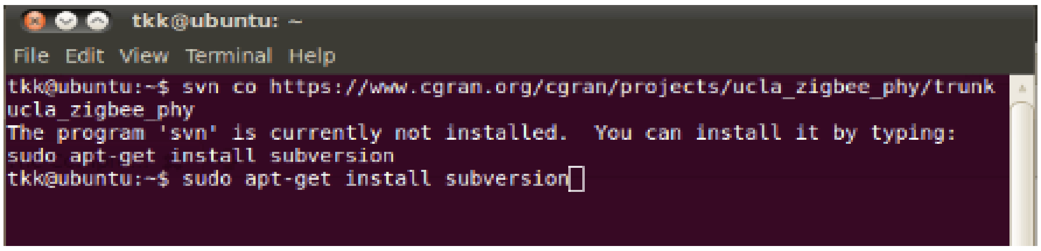
Figure 34: Commands for updating installed GNU Radio [22]

3.1.2 Installation & Configuration of USRP2

The USRP2 connect to the laptop gigabit Ethernet card through CAT6 cable which has RJ-45 jack at both ends. The USRP2 is different from its Predecessor, it need certain configuration at Linux terminal to work with GNU Radio. First the drivers of USRP2 were updated with XCVR2450. USRP2 communicates at IP/UDP layer. The default IP address for USRP2 is 192.168.10.2 and for making it work with a laptop, the laptop should assigned an IP address in the same subnet. When there is no IP address assigned to USRP2 then it communicate with laptop using UDP broadcast packets, so it is important to turn off the firewall before making connection with USRP2 [22].

3.1.3 Installation & Configuration of IEEE 802.15.4 Ucla_Zigbee_Phy code Examples.

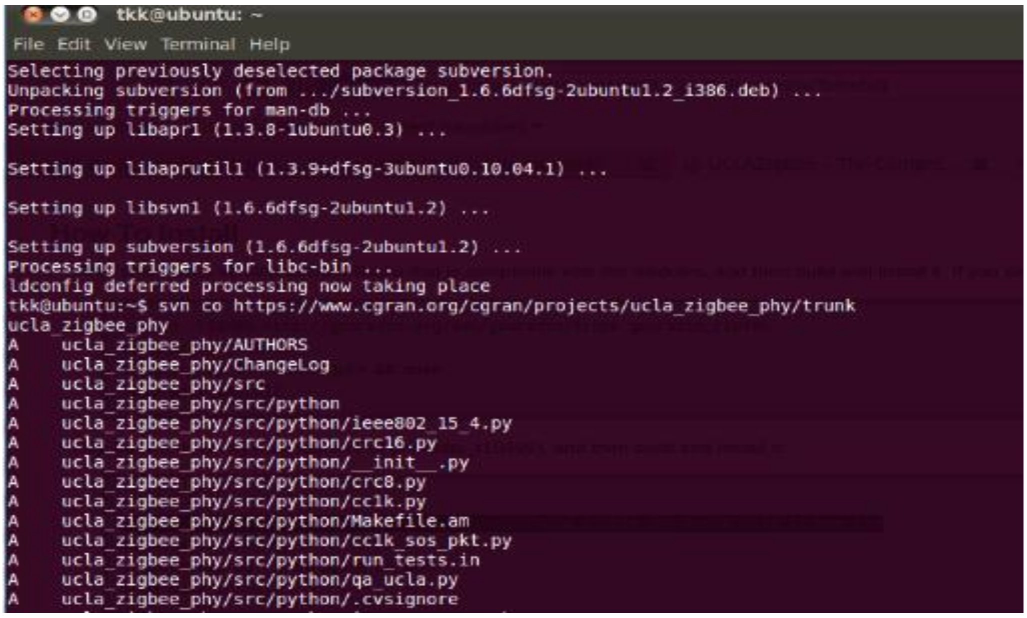
GNU Radio combined with USRP2 to computing with the CC2420 radio found in Sensinodes. IEEE 802.15.4 Ucla_zigbee_phy code examples were needed which were developed by Thomas Schmid and later updated by Lesile Choong and Sanna Leidelof. For downloading Ucla_Zigbee_Phy code examples, first installed “SVN” program through typing “sudo apt-get install subversion” command into terminal prompt (shown in Figure 35) which was needed to download Ucla_Zigbee_phy code examples.



```
tkk@ubuntu:~$ svn co https://www.cgran.org/cgran/projects/ucla_zigbee_phy/trunk
ucla_zigbee_phy
The program 'svn' is currently not installed.  You can install it by typing:
sudo apt-get install subversion
tkk@ubuntu:~$ sudo apt-get install subversion
```

Figure 35: Installation of SVN Program

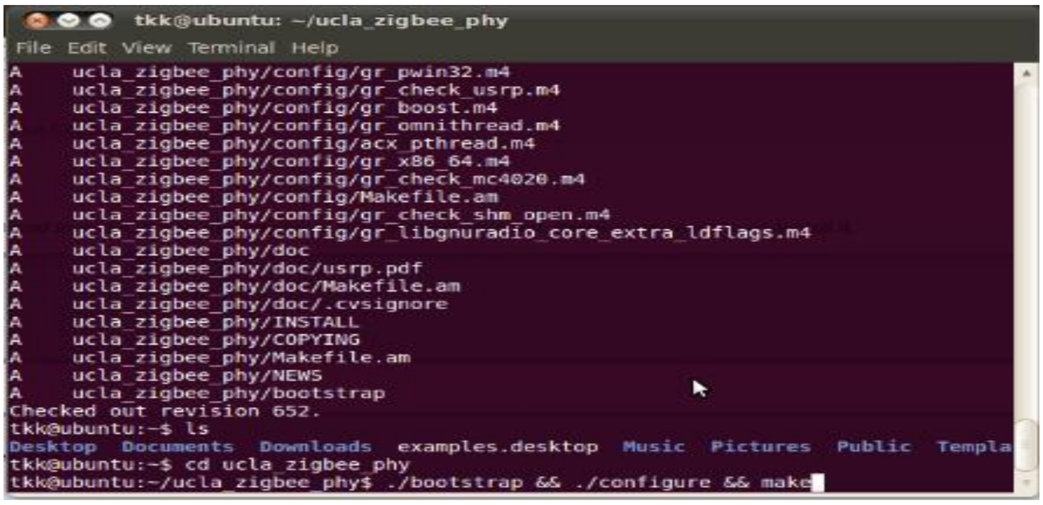
After installation of SVN program, the following link (svn co https://www.cgran.org/cgran/projects/ucla_zigbee_phy/trunk ucla_zigbee_phy) entered into terminal prompt which is shown in Figure 36, and it downloaded Ucla_Zigbee_Phy code examples [38].



```
tkk@ubuntu: ~
File Edit View Terminal Help
Selecting previously deselected package subversion.
Unpacking subversion (from .../subversion_1.6.6dfsg-2ubuntu1.2_i386.deb) ...
Processing triggers for man-db ...
Setting up libapr1 (1.3.8-1ubuntu0.3) ...

Setting up libaprutil1 (1.3.9+dfsg-3ubuntu0.10.04.1) ...

Setting up libsvn1 (1.6.6dfsg-2ubuntu1.2) ...

Setting up subversion (1.6.6dfsg-2ubuntu1.2) ...
Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
tkk@ubuntu:~$ svn co https://www.cgran.org/cgran/projects/ucla_zigbee_phy/trunk
ucla_zigbee_phy
A    ucla_zigbee_phy/AUTHORS
A    ucla_zigbee_phy/ChangeLog
A    ucla_zigbee_phy/src
A    ucla_zigbee_phy/src/python
A    ucla_zigbee_phy/src/python/ieee802_15_4.py
A    ucla_zigbee_phy/src/python/crc16.py
A    ucla_zigbee_phy/src/python/__init__.py
A    ucla_zigbee_phy/src/python/crc8.py
A    ucla_zigbee_phy/src/python/cc1k.py
A    ucla_zigbee_phy/src/python/Makefile.am
A    ucla_zigbee_phy/src/python/cc1k_sos_pkt.py
A    ucla_zigbee_phy/src/python/run_tests.in
A    ucla_zigbee_phy/src/python/qa_ucla.py
A    ucla_zigbee_phy/src/python/.cvsignore
```

Figure 36: Downloading IEEE802.15.4 Ucla_Zigbee_phy code examples

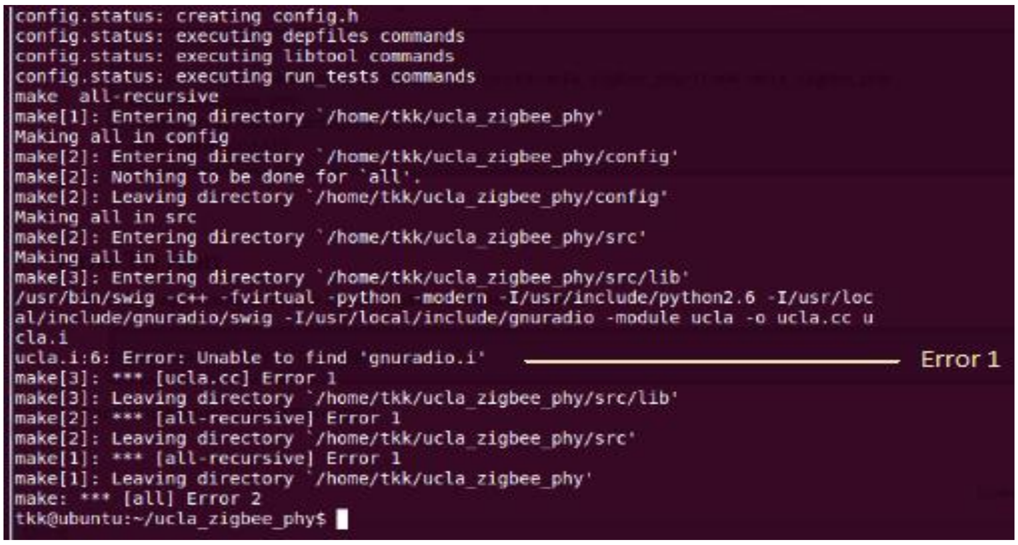
For Installation of IEEE 802.15.4 Ucla_Zigbee_Phy code examples, first went to ucla_zigbee_phy directory and then entered "./bootstrap && ./configure && make" into terminal prompt which is shown in Figure 37, it started installing Ucla_Zigbee_phy code examples [38].



```
tkk@ubuntu: ~/ucla_zigbee_phy
File Edit View Terminal Help
A    ucla_zigbee_phy/config/gr_pwin32.m4
A    ucla_zigbee_phy/config/gr_check_usrp.m4
A    ucla_zigbee_phy/config/gr_boost.m4
A    ucla_zigbee_phy/config/gr_omnithread.m4
A    ucla_zigbee_phy/config/acx_pthread.m4
A    ucla_zigbee_phy/config/gr_x86_64.m4
A    ucla_zigbee_phy/config/gr_check_mc4020.m4
A    ucla_zigbee_phy/config/Makefile.am
A    ucla_zigbee_phy/config/gr_check_shm_open.m4
A    ucla_zigbee_phy/config/gr_libgnuradio_core_extra_ldflags.m4
A    ucla_zigbee_phy/doc
A    ucla_zigbee_phy/doc/usrp.pdf
A    ucla_zigbee_phy/doc/Makefile.am
A    ucla_zigbee_phy/doc/.cvsignore
A    ucla_zigbee_phy/INSTALL
A    ucla_zigbee_phy/COPYING
A    ucla_zigbee_phy/Makefile.am
A    ucla_zigbee_phy/NEWS
A    ucla_zigbee_phy/bootstrap
Checked out revision 652.
tkk@ubuntu:~$ ls
Desktop  Documents  Downloads  examples.desktop  Music  Pictures  Public  Templa
tkk@ubuntu:~$ cd ucla_zigbee_phy
tkk@ubuntu:~/ucla_zigbee_phy$ ./bootstrap && ./configure && make
```

Figure 37: Installation of IEEE802.15.4 Ucla_Zigbee_phy code examples

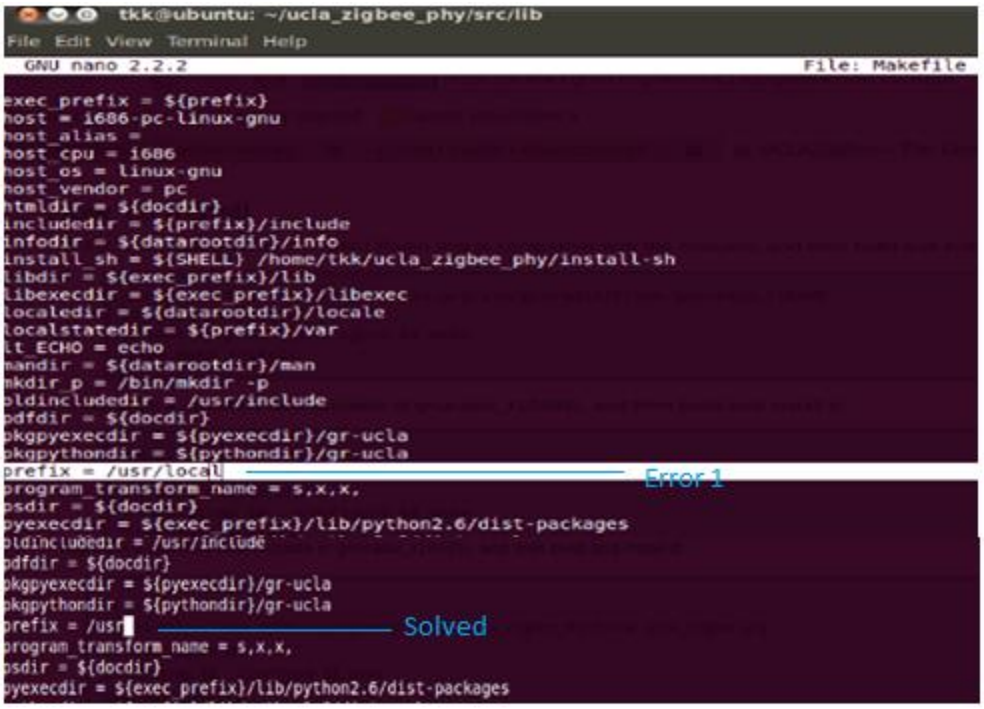
During the installation the first error we faced was "unable to find gnuradio.i" which is shown in Figure 38, and the installation terminated.



```
config.status: creating config.h
config.status: executing depfiles commands
config.status: executing libtool commands
config.status: executing run_tests commands
make  all-recursive
make[1]: Entering directory `/home/tkk/ucla_zigbee_phy'
Making all in config
make[2]: Entering directory `/home/tkk/ucla_zigbee_phy/config'
make[2]: Nothing to be done for `all'.
make[2]: Leaving directory `/home/tkk/ucla_zigbee_phy/config'
Making all in src
make[2]: Entering directory `/home/tkk/ucla_zigbee_phy/src'
Making all in lib
make[3]: Entering directory `/home/tkk/ucla_zigbee_phy/src/lib'
/usr/bin/swig -c++ -fvirtual -python -modern -I/usr/include/python2.6 -I/usr/loc
al/include/gnuradio/swig -I/usr/local/include/gnuradio -module ucla -o ucla.cc u
cla.i
ucla.i:6: Error: Unable to find 'gnuradio.i'
make[3]: *** [ucla.cc] Error 1
make[3]: Leaving directory `/home/tkk/ucla_zigbee_phy/src/lib'
make[2]: *** [all-recursive] Error 1
make[2]: Leaving directory `/home/tkk/ucla_zigbee_phy/src'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/home/tkk/ucla_zigbee_phy'
make: *** [all] Error 2
tkk@ubuntu:~/ucla_zigbee_phy$
```

Figure 38: Installation Error 1.

The file which is needed “gnuradio.i” is situated into directory “usr/include/gnuradio/swing” and the path which is written in Makefile is “usr/local/include/gnuradio/swig”, so we changed the path manually into Makefile by entering “ucla_zigbee_phy/src/lib pico Makefile” command into terminal prompt which is shown in Figure 39.

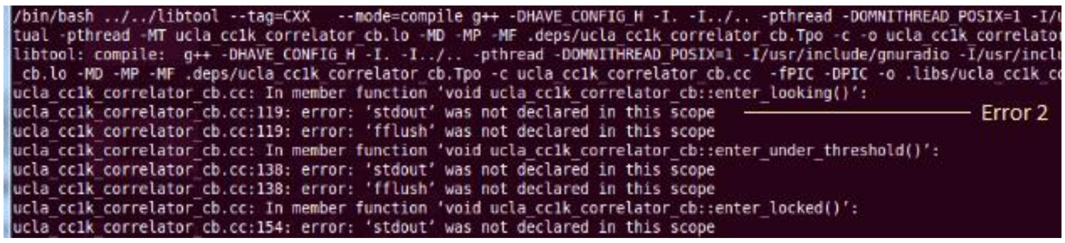


tkk@ubuntu: ~/ucla_zigbee_phy/src/lib
File Edit View Terminal Help

```
  GNU nano 2.2.2                                                    File: Makefile

exec_prefix = ${prefix}
host = i686-pc-linux-gnu
host_alias =
host_cpu = i686
host_os = linux-gnu
host_vendor = pc
htmldir = ${docdir}
includedir = ${prefix}/include
infodir = ${datarootdir}/info
install_sh = ${SHELL} /home/tkk/ucla_zigbee_phy/install-sh
libdir = ${exec_prefix}/lib
libexecdir = ${exec_prefix}/libexec
localedir = ${datarootdir}/locale
localstatedir = ${prefix}/var
lt_ECHO = echo
mandir = ${datarootdir}/man
mkdir_p = /bin/mkdir -p
oldincludedir = /usr/include
pdfdir = ${docdir}
pkgpyexecdir = ${pyexecdir}/gr-ucla
pkgpythondir = ${pythondir}/gr-ucla
prefix = /usr/local
program_transform_name = s,x,x,
psdir = ${docdir}
pyexecdir = ${exec_prefix}/lib/python2.6/dist-packages
oldincludedir = /usr/include
pdfdir = ${docdir}
pkgpyexecdir = ${pyexecdir}/gr-ucla
pkgpythondir = ${pythondir}/gr-ucla
prefix = /usr
program_transform_name = s,x,x,
psdir = ${docdir}
pyexecdir = ${exec_prefix}/lib/python2.6/dist-packages
```

Figure 39: Installation error 1 solved

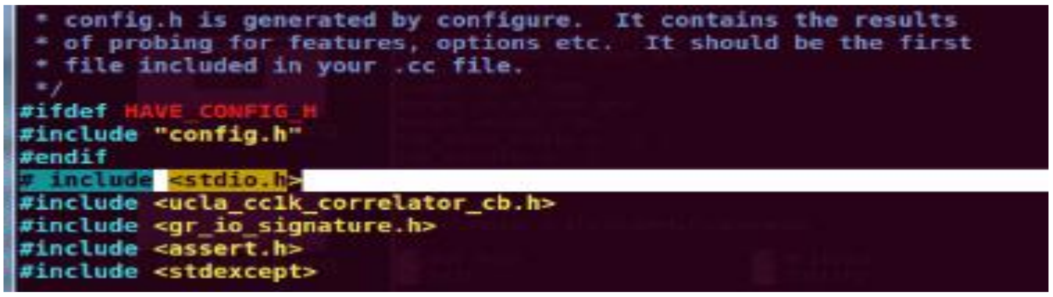
After solving the first error it is necessary to go back to the ucla_zigbee_phy directory and entered “sudo make install” command into terminal prompt which started to install Ucla_Zigbee_Phy code examples. After some further installation second error came saying that “stdio” was not declared in the “ucla_cc1k_correlator_cb.cc” file and installation terminated. The Error 2 is shown in Figure 40.



```
/bin/bash ../../libtool --tag=CXX   --mode=compile g++ -DHAVE_CONFIG_H -I. -I../.. -pthread -DOMNITHREAD_POSIX=1 -I/u
tual -pthread -MT ucla_cc1k_correlator_cb.lo -MD -MP -MF .deps/ucla_cc1k_correlator_cb.Tpo -c -o ucla_cc1k_correlato
libtool: compile:  g++ -DHAVE_CONFIG_H -I. -I../.. -pthread -DOMNITHREAD_POSIX=1 -I/usr/include/gnuradio -I/usr/incl
 cb.lo -MD -MP -MF .deps/ucla_cc1k_correlator_cb.Tpo -c ucla_cc1k_correlator_cb.cc  -fPIC -DPIC -o .libs/ucla_cc1k_c
ucla_cc1k_correlator_cb.cc: In member function ‘void ucla_cc1k_correlator_cb::enter_looking()’:
ucla_cc1k_correlator_cb.cc:119: error: ‘stdout’ was not declared in this scope
ucla_cc1k_correlator_cb.cc:119: error: ‘fflush’ was not declared in this scope
ucla_cc1k_correlator_cb.cc: In member function ‘void ucla_cc1k_correlator_cb::enter_under_threshold()’:
ucla_cc1k_correlator_cb.cc:138: error: ‘stdout’ was not declared in this scope
ucla_cc1k_correlator_cb.cc:138: error: ‘fflush’ was not declared in this scope
ucla_cc1k_correlator_cb.cc: In member function ‘void ucla_cc1k_correlator_cb::enter_locked()’:
ucla_cc1k_correlator_cb.cc:154: error: ‘stdout’ was not declared in this scope
```

Figure 40: Installation Error 2

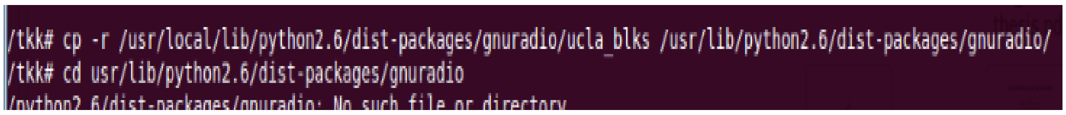
To solved this error, the “# include <stdio.h>” was manually entered into the file by entering “ucla_zigbee_phy/src/lib pico ucla_cc1k_correlator_cb.cc” command into terminal prompt which is shown in Figure 41.



```
 * config.h is generated by configure.  It contains the results
 * of probing for features, options etc.  It should be the first
 * file included in your .cc file.
 */
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif
# include <stdio.h>
#include <ucla_cc1k_correlator_cb.h>
#include <gr_io_signature.h>
#include <assert.h>
#include <stdexcept>
```

Figure 41: Installation error 2 solved

After solving the second error it is necessary to go back to the ucla_zigbee_phy directory and started installation again with the help of “sudo make install” command. After some further installation of IEEE 802.15.4 Ucla_Zigbee_phy code examples, faced a third error that “ucla_blks” was not found. This ucla_blks directory is situated in “/usr/local/lib/python2.6/dist-packages/gnuradio/ucla_blks” but for the completion of IEEE 802.15.4 Ucla_zigbee_phy code examples installation this directory was needed in “/usr/lib/python2.6/dist-packages/gnuradio/” so copy this directory with the help of entering the “cp –r /usr/local/lib/python2.6/dist-packages/gnuradio/ucla_blks /usr/lib/python2.6/dist-packages/gnuradio/” command into terminal prompt which is shown in Figure 42, it copy the ucla_blks directory to the required place.



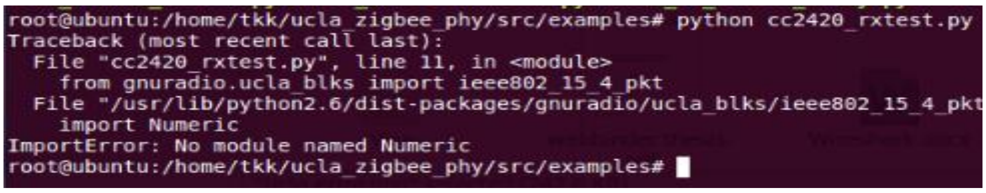
```
/tkk# cp -r /usr/local/lib/python2.6/dist-packages/gnuradio/ucla_blks /usr/lib/python2.6/dist-packages/gnuradio/
/tkk# cd usr/lib/python2.6/dist-packages/gnuradio
/python2.6/dist-packages/gnuradio: No such file or directory
```

Figure 42: Installation error 3 solved

After solving the third error no further error came and installation of IEEE 802.15.4 Ucla_zigbee_phy code examples are successfully done.

3.1.4 Installation of Python-Numeric Package

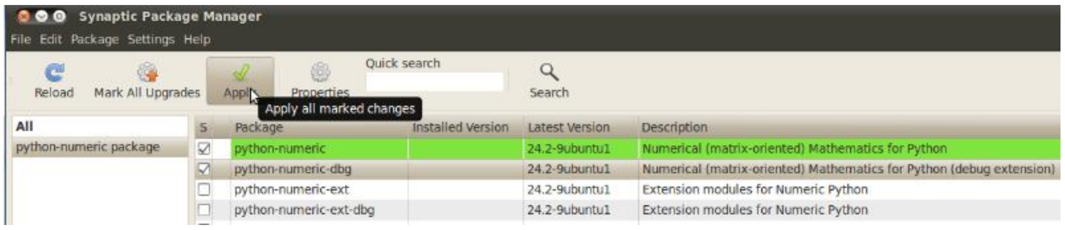
To run IEEE 802.15.4 Ucla_Zigbee_phy code examples it is necessary to install python-numeric package before running examples otherwise it gave error “import error: No module named Numeric”. The numeric error is shown in Figure 43.



```
root@ubuntu:/home/tkk/ucla_zigbee_phy/src/examples# python cc2420_rxtest.py
Traceback (most recent call last):
  File "cc2420_rxtest.py", line 11, in <module>
    from gnuradio.ucla_blks import ieee802_15_4_pkt
  File "/usr/lib/python2.6/dist-packages/gnuradio/ucla_blks/ieee802_15_4_pkt
    import Numeric
ImportError: No module named Numeric
root@ubuntu:/home/tkk/ucla_zigbee_phy/src/examples# 
```

Figure 43: Import Numeric Error

Open synaptic package manager and write python-numeric package and click on search then it found the package which selected and installed through synaptic package manager which is shown in Figure 44.



S	Package	Installed Version	Latest Version	Description
✓	python-numeric		24.2-9ubuntu1	Numerical (matrix-oriented) Mathematics for Python
✓	python-numeric-dbg		24.2-9ubuntu1	Numerical (matrix-oriented) Mathematics for Python (debug extension)
☐	python-numeric-ext		24.2-9ubuntu1	Extension modules for Numeric Python
☐	python-numeric-ext-dbg		24.2-9ubuntu1	Extension modules for Numeric Python

Figure 44: Import Numeric error solved

All the installation and configuration were completed which was needed for the communication with Sensinode test-bed platform.

3.2 Test-Bed Platform – Hardware and Operating System

Keeping the following points in mind the test-bed is developed and various components of the test-bed were chosen:

- It should allow changes in the experimental setup
- The adapting functionality to experiment needs and modification of system parameters are possible.
- The produced results are reliable.

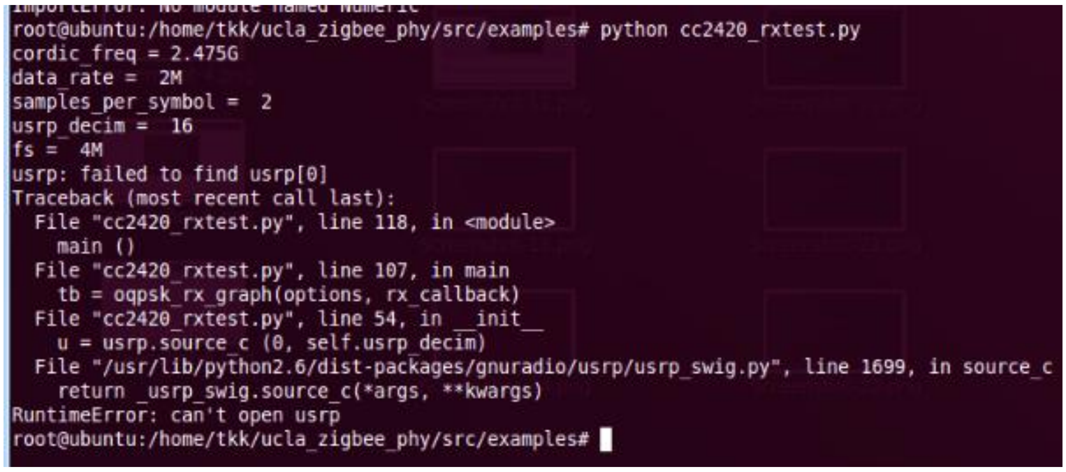
The test-bed is built up with laptop (Intel Core 2 Duo) that is running Linux-based operating system, Ubuntu 10.04. Ubuntu provides several setup customizations and a set of open source tools and APIs for modifications in the experimental setup.

The hardware which is used for this test-bed is USRP2 combined with daughter board (XCVR 2450 Transceiver) and GNU Radio, it used for multi-channel decoding. Its ADCs capacity is 14-bit 100MS/s, and DACs capacity is 16-bit 400MS/s, and Gigabit Ethernet capacity is 125MB/s which is equivalent to 30MS/s. When we decimation signal at rate of 4, USRP2 gives us sampling window of 25MHz. This window can cover 5 consecutive IEEE 802.15.4 channels in the 2.4GHz band. In ISM band the total 16 channels available but with the help of USRP2 it is a good start to utilized 5 channels to seeing the power of SDR.

3.3 Analysing Ucla_Zigbee_phy Code Examples with IEEE 802.15.4 Standard

After completed installation of ucla_zigbee_phy code examples, we open code files of cc2420_txtest.py (Transmitter), cc2420_rxtest.py (Receiver), and ieee802_15_4_pkt.py in python and check them that they were meeting the IEEE 802.15.4 standard. The code files were created by Thomas Schmid and later modified by Leslie Choong and Sanna Leidelof. In the file “ieee802_15_4_pkt” packet structure of IEEE 802.15.4 is defined according IEEE 802.15.4 standard.

When we opened code files of cc2420-txtest.py (Transmitter), cc2420_rxtest.py (Receiver) and try to understand them we found that the installed code files are not updated and they are written for USRP which was the precious version of USRP2. When we tried to run Receiver example code with the help of python on USRP2 it didn’t work and gave the RuntimeError: can´t open USRP which is shown in Figure 45. We already checked that our USRP2 is working before running this example.



```
ImportError: No module named Numeric
root@ubuntu:/home/tkk/ucla_zigbee_phy/src/examples# python cc2420_rxtest.py
cordic_freq = 2.475G
data_rate =  2M
samples_per_symbol =  2
usrp_decim =  16
fs =  4M
usrp: failed to find usrp[0]
Traceback (most recent call last):
  File "cc2420_rxtest.py", line 118, in <module>
    main ()
  File "cc2420_rxtest.py", line 107, in main
    tb = oqpsk_rx_graph(options, rx_callback)
  File "cc2420_rxtest.py", line 54, in __init__
    u = usrp.source_c (0, self.usrp_decim)
  File "/usr/lib/python2.6/dist-packages/gnuradio/usrp/usrp_swig.py", line 1699, in source_c
    return _usrp_swig.source_c(*args, **kwargs)
RuntimeError: can't open usrp
root@ubuntu:/home/tkk/ucla_zigbee_phy/src/examples# 
```

Figure 45: Error can’t open USRP

So we know that we need to update these code files for USRP2. We tried to look that if someone already updated these code files onto internet and after a bit struggle we were able to found updated version of Receiver code file from “The University of UTAH weblink” and the updated version of Transmitter code file from a discussion forum where Sanna leidelof upload the file. After downloading and updating these files our cc2420_txtest.py code file for transmitter and cc2420_rxtest.py code file for receiver is ready.

After updating these codes file we run cc2420_txtest.py code file which was for transmitter with the help of python and we found that it was working fine and sending IEEE 802.15.4 packets through USRP2. Now our transmitter is ready for transmit signal through USRP2 to existing Sensinode test-bed platform but when we tried to run cc2420_rxtest.py code file which is for receiver then it found USRP2 but later it gave error that chan_802_15_4 attribute not found .

We look through cc2420_rxtest.py code file and tried to found this problem but couldn't found and the code seem ok so we know that the problem was not into this file, so we went through ieee802_15_4.py code file which was also ok and then ieee802_15_4_pkt.py code files and found that the chan_802_15_4 class which was needed for cc2420_rxtest.py code file is missing. This ieee802_15_4_pkt.py code file with up gradation of missing part was also found on "The University of UTAH weblink". After downloading and updating the ieee802_15_4_pkt.py code file we run cc2420_rxtest.py code file with the help of python it worked and saying waiting for packets.

Now our platform was ready to receive and transmit IEEE 802.15.4 packets through USRP2 to existing Sensinode platform.

3.4 Experimental Setup

The physical setup of the test-bed is shown in Figure 46. One laptop was configured and connected through gigabit Ethernet card with USRP2 running Linux-based operating system, Ubuntu 10.04 as well as GNU Radio and IEEE 802.15.4 Ucla_zigbee_phy code examples install on it and the other laptop is configured and connected through USB 2.0 with Sensinode running Linux-based operating system, Ubuntu 10.04.

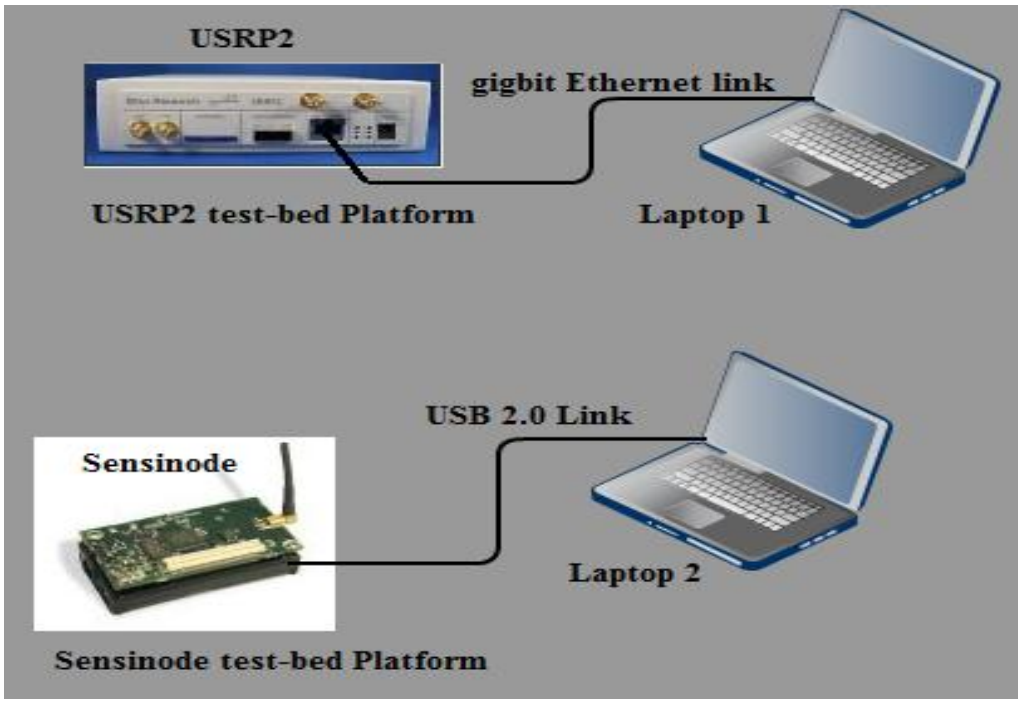


Figure 46: Experimental setup diagram

We transmit IEEE 802.15.4 packets on channel 15 from Sensinode test-bed platform and received it on USRP2 test-bed platform by running receiver file "cc2420_rxtest.py" code into python. The IEEE 802.15.4 packets are successfully received on USRP2 test-bed platform which showed that USRP2 test-bed platform was compatible and ready to communicate with existed Sensinode test-bed platform. IEEE 802.15.4 received packets are shown in Figure 47.

```
root@tkk-laptop:/home/tkk/ucla_zigbee_phy/src/examples#
./cc2420_rxtest.py
Enabled Realtime
cordic_freq = 2.425G
data_rate =  2M
samples_per_symbol =  2
usrp_decim =  25
>>> gr_fir_fff: using SSE
802_15_4_pkt: waiting for packet
received packet
checksum: 27663, received: 12058
/usr/lib/python2.6/dist-
packages/gnuradio/ucla_blks/ieee802_15_4_pkt.py:280: DeprecationWarning:
integer argument expected, got float
  self.callback(ok, msg_payload, self.chan_num)
802_15_4_pkt: waiting for packet
received packet
checksum: 10783, received: 26890
802_15_4_pkt: waiting for packet
received packet
checksum: 42559, received: 58666
```

```
802_15_4_pkt: waiting for packet
received packet
checksum: 23804, received: 8169
802_15_4_pkt: waiting for packet
received packet
checksum: 15642, received: 32271
802_15_4_pkt: waiting for packet
received packet
checksum: 63274, received: 46143
802_15_4_pkt: waiting for packet
received packet
checksum: 19449, received: 2284
802_15_4_pkt: waiting for packet
received packet
checksum: 33225, received: 49884
802_15_4_pkt: waiting for packet
received packet
checksum: 30651, received: 13486
802_15_4_pkt: waiting for packet
received packet
checksum: 54050, received: 36919
802_15_4_pkt: waiting for packet
received packet
checksum: 45486, received: 62139
802_15_4_pkt: waiting for packet
received packet
checksum: 64895, received: 48746
802_15_4_pkt: waiting for packet
received packet
checksum: 43838, received: 59435
802_15_4_pkt: waiting for packet
received packet
checksum: 20989, received: 4840
802_15_4_pkt: waiting for packet
received packet
checksum: 56797, received: 40648
802_15_4_pkt: waiting for packet
received packet
checksum: 39885, received: 55512
802_15_4_pkt: waiting for packet
received packet
checksum: 1980, received: 17577
802_15_4_pkt: waiting for packet
received packet
checksum: 35740, received: 51337
802_15_4_pkt: waiting for packet
received packet
checksum: 12315, received: 29454
802_15_4_pkt: waiting for packet
received packet
checksum: 64043, received: 47422
802_15_4_pkt: waiting for packet
received packet
```

Figure 47: IEEE 802.15.4 received packets.

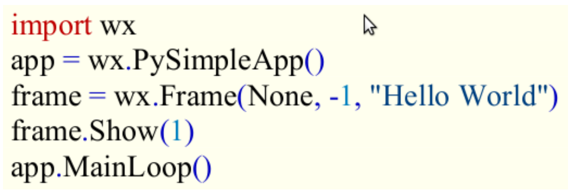
4 TEST, RESULTS AND ANALYSYS

During the implementation of each task, it is necessary to test it before proceed to the next tasks. The tests are divided into four parts:

- Test of GNU Radio
- Test of USRP2
- Test of IEEE 802.15.4 Ucla_Zigbee_phy code examples
- Test of communication between USRP2 and Sensinode

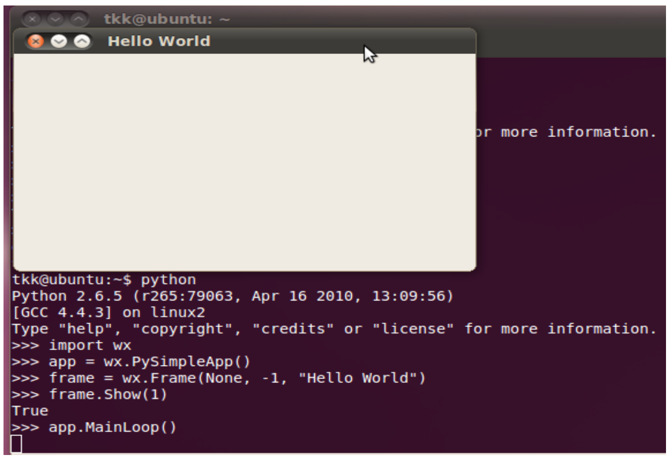
4.1 Test of GNU Radio

First GNU Radio test has to been done to make sure that it is install and working properly. To do this we run a small program in Python:



```
import wx
app = wx.PySimpleApp()
frame = wx.Frame(None, -1, "Hello World")
frame.Show(1)
app.MainLoop()
```

Here is what should get with wx, which proves that GNU Radio is working properly and shown in the Figure 48.



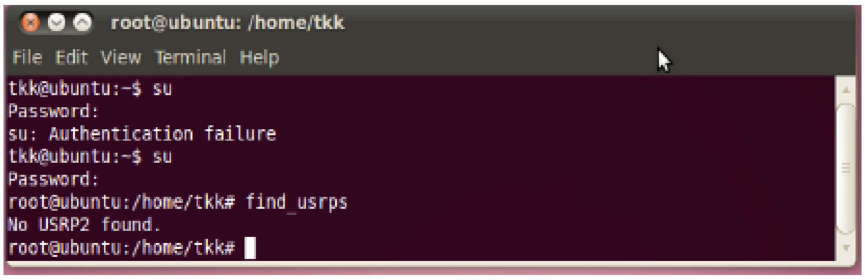
```
or more information.
tkk@ubuntu:~$ python
Python 2.6.5 (r265:79063, Apr 16 2010, 13:09:56)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import wx
>>> app = wx.PySimpleApp()
>>> frame = wx.Frame(None, -1, "Hello World")
>>> frame.Show(1)
True
>>> app.MainLoop()
```

Figure 48: GNU Radio Test output.

4.2 Test of USRP2

The working of USRP2 can be test with following two commands:

1. Applying “find_usrps” command into terminal prompt will give one of these two results. USRP2 is not connected; the display is shown in Figure 49.



```
tkk@ubuntu:~$ su
Password:
su: Authentication failure
tkk@ubuntu:~$ su
Password:
root@ubuntu:/home/tkk# find_usrps
No USRP2 found.
root@ubuntu:/home/tkk# 
```

Figure 49: USRP2 not connected test output

USRP2 is connected the display will be

00:50:c2:85:32:95 hw_rev = 0xo400

2. Applying “usrp2_fft.py” command into terminal prompt will show this graph if USRP2 is connected. The graph is shown in Figure 50.

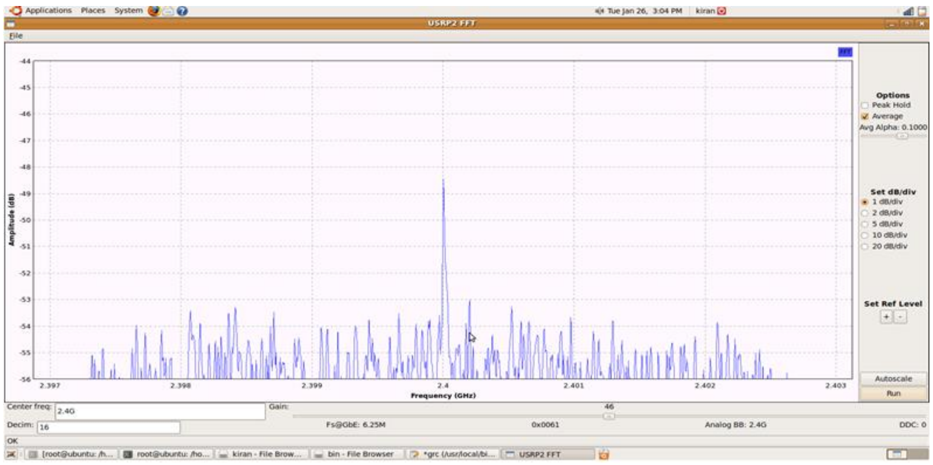


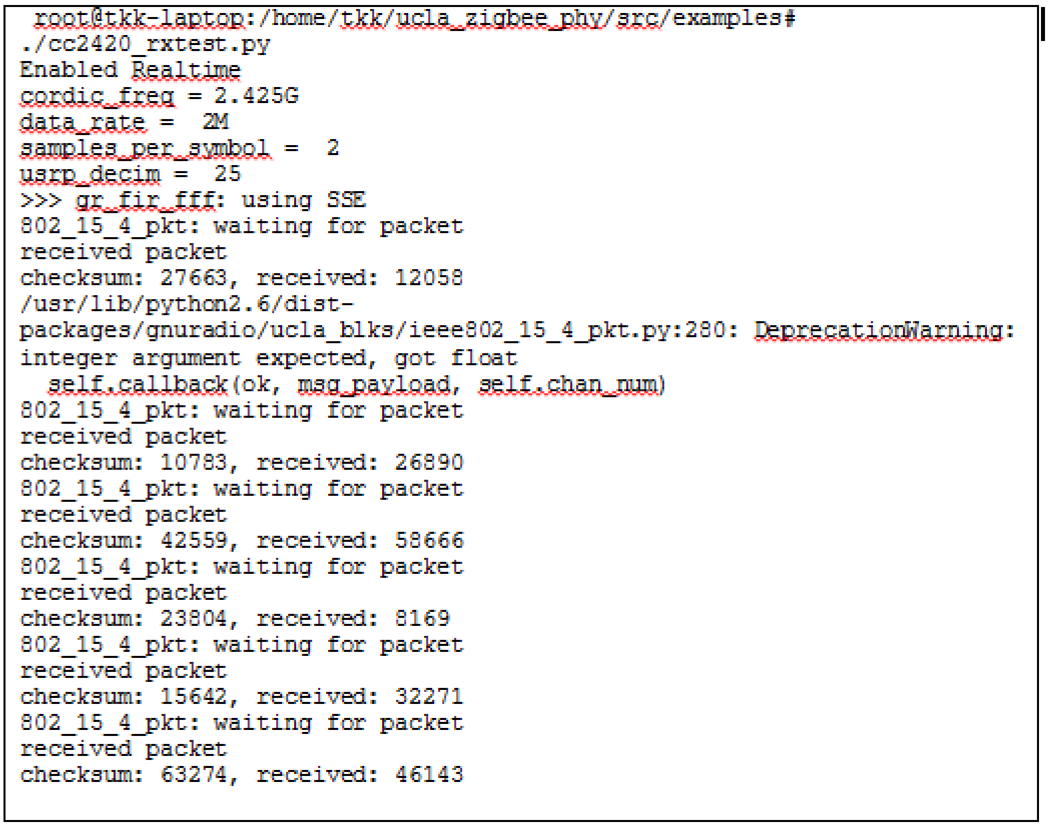
Figure 50: USRP2 connected test output.

4.3 Test of IEEE 802.15.4 Ucla_Zigbee_phy code Examples

The installation and working of IEEE 802.15.4 Ucla_zigbee_phy code examples can be test by running transmitter and receiver examples into python. When run “cc2420_txtest.py” into python, it starts transmitting IEEE 802.15.4 packets through USRP2 and when run “cc2420_rxtest.py” into python, it starts receiving IEEE 802.15.4 packets and if packet is not available it start saying waiting for the packet.

4.4 Test of Communication between USRP2 and Sensinode

To test the communication between existing Sensinode platform and USRP2 platform, we transmitted IEEE 802.15.4 packets from Sensinode and received it to USRP2 by running “cc2420_rxtest.py into python. It shows onto the screen that packets are receiving. The IEEE 802.15.4 received packets are shown in Figure 51.



```
 root@tkk-laptop:/home/tkk/ucla_zigbee_phy/src/examples#
./cc2420_rxtest.py
Enabled Realtime
cordic_freq = 2.425G
data_rate =  2M
samples_per_symbol =  2
usrp_decim =  25
>>> gr_fir_fff: using SSE
802_15_4_pkt: waiting for packet
received packet
checksum: 27663, received: 12058
/usr/lib/python2.6/dist-
packages/gnuradio/ucla_blks/ieee802_15_4_pkt.py:280: DeprecationWarning:
integer argument expected, got float
  self.callback(ok, msg_payload, self.chan_num)
802_15_4_pkt: waiting for packet
received packet
checksum: 10783, received: 26890
802_15_4_pkt: waiting for packet
received packet
checksum: 42559, received: 58666
802_15_4_pkt: waiting for packet
received packet
checksum: 23804, received: 8169
802_15_4_pkt: waiting for packet
received packet
checksum: 15642, received: 32271
802_15_4_pkt: waiting for packet
received packet
checksum: 63274, received: 46143
```

Figure 51: IEEE 802.15.4 packets received on USRP2

5 CONCLUSION

ComNet department at Aalto University is working on Sensinode platform. This platform is limited in its processing power, memory and hardware interface for external environment sensor, so ComNet is exploring the feasibility of using Software Defined Radio for sensor and cognitive networks.

In this thesis, the purpose of exploring GNU Radio is that to use it for wireless network platform. GNU Radio is a sophisticated and rich programming environment. GNU Radio together with USRP2 is a low cost platform with huge implementation flexibility and support.

The Main goal of this project was to develop a Software Defined Radio platform running with IEEE 802.15.4 and to make sure it communicates with existing Sensinode platform. This has been achieved by developing a Software Defined Radio platform on a Laptop running Ubuntu 10.04 operating system following with the installation of GNU Radio, USRP2, and IEEE 802.15.4 Ucla_zigbee_phy code examples. This Software Defined Radio was configured as a receiver and the existing Sensinode platform was configured as a transmitter. The IEEE 802.15.4 packets were transmitted from existing Sensinode platform and successfully received on Software Defined Radio platform.

This Software Defined Radio platform setup is a first step towards inquiring Software Defined Radio techniques for wireless sensor networks and low power communication protocols such as IEEE 802.15.4. It permits to discover problems and differences with which one has to deal in order to use Software Defined Radio solutions as they exist today as well as give the opportunity to the researchers to quickly validate their innovative concepts and solutions.

REFERENCES

[1]: D. Saha, D. Grunwald, and D. Sicker, “Wireless innovation through software radios,” SIGCOMM Comput. Commun. Rev., vol. 39, no. 1, pp. 62–67, 2009 Available in www-form:

<URL:http://delivery.acm.org/10.1145/1500000/1496102/p62-aha.pdf?ip=193.166.120.39&CFID=28337591&CFTOKEN=27239160&__acm__=1307881994_a6e6b489bca680ab3baf712217c6ff4c>

[2]: Siensinode co Ltd “Sensinode Micro Series and NanoStack” Available in www-form:

<URL:http://www.sensinode.com/ >

[3]: Ettus Research LLC “USRP/USRP2 from Ettus research LLC” Available in www-form:

<URL:http://www.ettus.com/>

[4]: GNU Radio community “GNU Radio for Software Defined Radio” Available in www-form:

<URL:http://gnuradio.org/redmine/wiki/gnuradio>

[5]: Chipcon productions “2.4 GHz IEEE 802.15.4/ Zigbee-ready RF transceiver” Texas Instruments. Available in www-form:

<URL:http://focus.ti.com/lit/ds/symlink/cc2420.pdf>

[6]: David Wetherall “Taking the sting out of Carrier Sense: Interference Cancellation for Wireless LANs” University of Washington. Available in www-form:

<URL:http://www.seattle.intel-research.net/pubs/sic-mobicom.pdf>

[7]: Joseph Mitola III and Gerald Q.Maguire, "Cognitive Radio: Making software Radios More Personal," IEEE Personal communications, August 1999. Available in www-form:

<URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=788210&userType=&tag=1>

[8]: Joseph Mitola, "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio," in Royal Institute of Technology (KTH) Sweden: Royal Institute of Technology (KTH), 2000, pp. xi, 313 p.

[9]: Fcc, "Facilitating Opportunities for Flexible, Efficient, and Reliable Spectrum Use Employing Cognitive Radio Technologies" in 2003. Available in www-form:

<URL:http://www.cs.ucdavis.edu/~liu/289I/Material/FCC-03-322A1.pdf>

[10]: Bluetooth Sig, "Specification of the Bluetooth system, Master Table of Contents and Compliance Requirements," Nov 2004. Available in www-form:

<URL:http://netlab.cs.ucla.edu/wiki/files/btv12.pdf>

[11]: IEEE 802.22 Wrieless Ran, "Functional requirements for the 802.22 WRAN standard," IEEE 802.22- 05/0007r46, Oct 2005

[12]: Yonghong Zeng and Ying-Chang Liang, "Covariance Based Signal Detections For Cognitive Radio" Institute for Infocomm Research, A*STAR, IEEE 2007. Available in www-form:

<URL:http://www1.i2r.a-star.edu.sg/~yhzeng/Zeng-DySAPN-2007.pdf>

[13]: "IEEE Standard Definitions and Concepts for Dynamic Spectrum Access: Terminology Relating to Emerging Wireless Networks, System functionality, and Spectrum Management," IEEE Communications Society, vol. IEEE Std 1900.1 TM, September 2008.

[14]: Aldo Buccardo “A Signal Detector for Cognitive Radio System” 11, 2010 University of Gävel.

[15]: Simon Haykin and Life Fellow, "Cognitive Radio: Brain-Empowered Wireless communications," IEEE Journal in communications, Feb 2005.

[16]: IEEE 802 group “IEEE 802.22 working group on Wireless Regional Area Networks” Available in www-form:

<URL:http://ieee802.org/22/ >

[17]: IEEE 802.15 task group “IEEE 802.15 WPAN (TG2)” Available in www-form:

<URL:http://ieee802.org/15/pub/TG2.html >

[18]: GNU operating system community “Exploring GNU Radio” Available in www-form:

<URL:http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html>

[19]: Matthias Fähnle “SDR with GNU Radio and USRP2 hardware frontend, setup FM/GSM application” 02, 2010 Hochschule Ulm University of Applied Sciences. Available in www-form:

<URL:http://www.hsulm.de/opus/volltexte/2010/27/pdf/SDR_GNURadio_USRP_Feb2010.pdf>

[20]: Norton Quinn Wired “GNU Radio Opens an Unseen World”. Available in www-form:

<URL:http://www.wired.com/science/discoveries/news/2006/06/70933?currentPage=1>

[21]: Thoms Schmid “GNU Radio 802.15.4 En-and Decoding” 2006. Available in www-form:

<URL:http://nesl.ee.ucla.edu/fw/thomas/thomas_project_report.pdf>

[22]: GNU Radio community “GNU Radio modules” Available in www-form:

<URL:http://gnuradio.org/redmine/wiki/gnuradio>

[23]: Joshknows community “Exploring GRC” Available in www-form:

<URL:http://www.joshknows.com/grc>

[24]: GNU Radio community “Exploring USRP for Software Defined Radio” Available in www-form:

<URL:http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html#usrp>

[25]: Ettus Research LLC “USRP/USRP2 Datasheets” Available in www-form:

<URL:http://www.ettus.com/download>

[26]: Ettus Research LLC “Exploring USRP2 for Software Defined Radio” Available in www-form:

<URL: http://www.ettus.com/orderpage.html, 2009.>

[27]: Sriram Sanka “Communication between Wireless Sensor Devices and GNU Radio” Available in www-form:

<URL:http://academic.csuohio.edu/yuc/mobile09/gaurav.pdf>

[28] Carl Ingemarsson “Hardware evalution platform based on GNR Radio and USRP” University of Linköping. Available in www-form:

<URL:http://liu.diva-portal.org/smash/record.jsf?pid=diva2:218764>

[29]: Adnan Aftab “Spectrum Sensing Trough Implementation of USRP2” School of Computing, Sweden. Available in www-form:

<URL:http://www.seamist.se/com/mscee.nsf/attachments/webbinder_pdf/$file/webbinder.pdf>

[30]: Elaheh Rahmani “Zigbee/IEEE 802.15.4” 05, 2005 University of Tehran. Available in www-form:

<URL: http://ece.ut.ac.ir/silab/erahmani/local/pp01.pdf>

[31]: Jianling Zheg "A Comprehensive Performance Study of IEEE 802.15.4" Available in www-form:

<http://www-ee.ccny.cuny.edu/zheng/papers/paper1_wpan_performance.pdf>

[32]: Leslie Choong "Multi-Channel IEEE 802.15.4 Packet Capture Using Software Defined Radio" University of California. Available in www-form:

<URL:http://nesl.ee.ucla.edu/fw/thomas/leslie_choong_multichannel_ieee802154.pdf>

[33]:http://www.jennic.com/files/support_files/JN-AN-1079%20Coexistence%20of%20IEEE%20802.15.4%20In%20The%202.4GHz%20Band-1v0.pdf>

[34]: Sensinode Co Ltd "Micro Hardware Manual" Available in www-form:

<URL:http://www.ee.oulu.fi/~ikram/microseries/sensinode-manual-hw.pdf>

[35]: Sensinode Co Ltd "Micro.2420 U100 datasheet" Available in www-form:

<URL:http://www.ee.oulu.fi/~ikram/microseries/sensinode-datasheet-U100R2-20070923.pdf>

[36]: Sensinode Co Ltd "Micro.usb U600 datasheet" Available in www-form:

<URL: http://www.alldatasheet.com/datasheet-pdf/pdf/187781/ETC2/U600.html>

[37]: Sensinode Co Ltd "NanoStack Manual" Available in www-form:

<Sensinode NanoStack Manual v1.0.1.pdf>

[38]: Trac integrated SCM & Project Management "IEEE 802.15.4 Ucla_Zigbee_phy Code Examples" Available in www-form:

<URL: https://www.cgran.org/wiki/UCLAZigBee#HowToRun>